

## new/exception\_lists/copyright

\*\*\*\*\*

18553 Fri Jan 17 07:29:12 2014

## new/exception\_lists/copyright

### 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 # Copyright (c) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
22 # Copyright 2011 Nexenta Systems, Inc. All rights reserved.
23 # Copyright (c) 2011 by Delphix. All rights reserved.
24 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
```

```
26 syntax: glob
27 exception_lists/closed-bins
28 exception_lists/copyright
29 exception_lists/cstyle
30 exception_lists/hdrchk
31 usr/src/cmd/dtrace/test/tst/common/*.out
32 usr/src/cmd/krb5/kadmin/cli/kadmin_ct.c
33 usr/src/cmd/krb5/kadmin/cli/kadmin.h
34 usr/src/cmd/krb5/kadmin/cli/ss_wrapper.c
35 usr/src/cmd/krb5/kadmin/dbutil/nstrtok.h
36 usr/src/cmd/krb5/kadmin/dbutil/ovload.c
37 usr/src/cmd/krb5/kadmin/dbutil/string_table.c
38 usr/src/cmd/krb5/kadmin/dbutil/strtok.c
39 usr/src/cmd/krb5/kadmin/dbutil/util.c
40 usr/src/cmd/krb5/kadmin/kpasswd/kpasswd_strings.h
41 usr/src/cmd/krb5/kadmin/kpasswd/kpasswd.h
42 usr/src/cmd/krb5/kadmin/ktutil/ktutil.h
43 usr/src/cmd/krb5/kadmin/server/server_glue_v1.c
44 usr/src/cmd/krb5/krb5kdc/extern.c
45 usr/src/cmd/krb5/krb5kdc/kdc_util.c
46 usr/src/cmd/krb5/krb5kdc/policy.c
47 usr/src/cmd/krb5/krb5kdc/replay.c
48 usr/src/cmd/krb5/krb5kdc/sock2p.c
49 usr/src/cmd/krb5/ldap_util/kdb5_ldap_list.c
50 usr/src/cmd/krb5/ldap_util/kdb5_ldap_list.h
51 usr/src/cmd/krb5/ldap_util/kdb5_ldap_policy.h
52 usr/src/cmd/krb5/ldap_util/kdb5_ldap_realm.h
53 usr/src/cmd/krb5/ldap_util/kdb5_ldap_services.h
54 usr/src/cmd/smbsrv/smbd/eventlog.dll
55 usr/src/cmd/terminfo/termcap.src
56 usr/src/cmd/terminfo/terminfo.src
57 usr/src/common/acpica
58 usr/src/common/bzip2/LICENSE
59 usr/src/common/bzip2/Solaris.README.txt
60 usr/src/common/bzip2/bzlib.h
```

1

## new/exception\_lists/copyright

```
61 usr/src/common/bzip2/crcTable.c
62 usr/src/common/bzip2/randTable.c
63 usr/src/common/bzip2/blocksort.c
64 usr/src/common/bzip2/compress.c
65 usr/src/common/bzip2/bzlib.c
66 usr/src/common/bzip2/decompress.c
67 usr/src/common/bzip2/bzlib_private.h
68 usr/src/common/bzip2/huffman.c
69 usr/src/common/openssl/crypto/krb5/krb5_asn.c
70 usr/src/common/openssl/crypto/krb5/krb5_asn.h
71 usr/src/grub/grub-0.97/stage2/Makefile.am
72 usr/src/lib/gss_mechs/mech_krb5/crypto/cksumtype_to_string.c
73 usr/src/lib/gss_mechs/mech_krb5/crypto/coll_proof_cksum.c
74 usr/src/lib/gss_mechs/mech_krb5/crypto/entype_compare.c
75 usr/src/lib/gss_mechs/mech_krb5/crypto/keyed_checksum_types.c
76 usr/src/lib/gss_mechs/mech_krb5/crypto/keyed_cksum.c
77 usr/src/lib/gss_mechs/mech_krb5/crypto/keylengths.c
78 usr/src/lib/gss_mechs/mech_krb5/crypto/old/des_stringtokey.c
79 usr/src/lib/gss_mechs/mech_krb5/crypto/random_to_key.c
80 usr/src/lib/gss_mechs/mech_krb5/crypto/string_to_cksumtype.c
81 usr/src/lib/gss_mechs/mech_krb5/crypto/string_to_entype.c
82 usr/src/lib/gss_mechs/mech_krb5/crypto/valid_cksumtype.c
83 usr/src/lib/gss_mechs/mech_krb5/et/error_table.h
84 usr/src/lib/gss_mechs/mech_krb5/et/internal.h
85 usr/src/lib/gss_mechs/mech_krb5/et/mit-sipb-copyright.h
86 usr/src/lib/gss_mechs/mech_krb5/include/cache-addinfo.h
87 usr/src/lib/gss_mechs/mech_krb5/include/cm.h
88 usr/src/lib/gss_mechs/mech_krb5/include/db-config.h
89 usr/src/lib/gss_mechs/mech_krb5/include/db.h
90 usr/src/lib/gss_mechs/mech_krb5/include/fake-addinfo.h
91 usr/src/lib/gss_mechs/mech_krb5/include/foreachaddr.h
92 usr/src/lib/gss_mechs/mech_krb5/include/k5-int-pkinit.h
93 usr/src/lib/gss_mechs/mech_krb5/include/k5-utf8.h
94 usr/src/lib/gss_mechs/mech_krb5/include/kdb_kt.h
95 usr/src/lib/gss_mechs/mech_krb5/include/krb5_libinit.h
96 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm_defs.h
97 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm_proto.h
98 usr/src/lib/gss_mechs/mech_krb5/include/krb5/adm.h
99 usr/src/lib/gss_mechs/mech_krb5/include/krb5/copyright.h
100 usr/src/lib/gss_mechs/mech_krb5/include/krb5/k5-err.h
101 usr/src/lib/gss_mechs/mech_krb5/include/krb5/k5-plugin.h
102 usr/src/lib/gss_mechs/mech_krb5/include/locate_plugin.h
103 usr/src/lib/gss_mechs/mech_krb5/include/port-sockets.h
104 usr/src/lib/gss_mechs/mech_krb5/include/preauth_plugin.h
105 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_decode.c
106 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_decode.h
107 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_encode.c
108 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_encode.h
109 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_get.c
110 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_get.h
111 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_decode.h
112 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_encode.h
113 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_make.c
114 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_make.h
115 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_misc.c
116 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_misc.h
117 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1buf.h
118 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krb5_decode.c
119 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krb5_encode.c
120 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krbasn1.h
121 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/ldap_key_seq.c
122 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc-int.h
123 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cccopy.c
124 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccdefault.c
125 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccdefops.c
126 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccfns.c
```

2

```
127 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/fcc.h
128 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/scc.h
129 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ser_cc.c
130 usr/src/lib/gss_mechs/mech_krb5/krb5/error_tables/adm_err.h
131 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/kt-int.h
132 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktadd.c
133 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktdefault.c
134 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktfns.c
135 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktremove.c
136 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/read_servi.c
137 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_comp.c
138 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_order.c
139 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_srch.c
140 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/appdefault.c
141 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/bld_pr_ext.c
142 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/bld Princ.c
143 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/chk_trans.c
144 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/cleanup.h
145 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_addrs.c
146 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_creds.c
147 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_data.c
148 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_tick.c
149 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/decode_kdc.c
150 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/decrypt_tk.c
151 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/deltat.c
152 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/enc_helper.c
153 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/encode_kdc.c
154 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/encrypt_tk.c
155 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/free_rtree.c
156 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gc_via_tkt.c
157 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gen_subkey.c
158 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gic_opt.c
159 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/init_keyblock.c
160 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/int-prot.h
161 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/kdc_rep_dc.c
162 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/kerrs.c
163 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_error.c
164 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_rep.c
165 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/pr_to_salt.c
166 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/preauth.c
167 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_error.c
168 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_rep.c
169 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_req.c
170 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_safe.c
171 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/tgtname.c
172 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/valid_times.c
173 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/vic_opt.c
174 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/walk_rtree.c
175 usr/src/lib/gss_mechs/mech_krb5/krb5/os/accessor.c
176 usr/src/lib/gss_mechs/mech_krb5/krb5/os/changepw.c
177 usr/src/lib/gss_mechs/mech_krb5/krb5/os/dnssrv.c
178 usr/src/lib/gss_mechs/mech_krb5/krb5/os/free_hstrl.c
179 usr/src/lib/gss_mechs/mech_krb5/krb5/os/free_krbhs.c
180 usr/src/lib/gss_mechs/mech_krb5/krb5/os/full_ipadr.c
181 usr/src/lib/gss_mechs/mech_krb5/krb5/os/gen_port.c
182 usr/src/lib/gss_mechs/mech_krb5/krb5/os/gen_rname.c
183 usr/src/lib/gss_mechs/mech_krb5/krb5/os/genaddrs.c
184 usr/src/lib/gss_mechs/mech_krb5/krb5/os/get_krbhst.c
185 usr/src/lib/gss_mechs/mech_krb5/krb5/os/gmt_mktime.c
186 usr/src/lib/gss_mechs/mech_krb5/krb5/os/hostaddr.c
187 usr/src/lib/gss_mechs/mech_krb5/krb5/os/localaddr.c
188 usr/src/lib/gss_mechs/mech_krb5/krb5/os/lock_file.c
189 usr/src/lib/gss_mechs/mech_krb5/krb5/os/mk_faddr.c
190 usr/src/lib/gss_mechs/mech_krb5/krb5/os/net_read.c
191 usr/src/lib/gss_mechs/mech_krb5/krb5/os/net_write.c
192 usr/src/lib/gss_mechs/mech_krb5/krb5/os/os-prot.h
```

```
193 usr/src/lib/gss_mechs/mech_krb5/krb5/os/osconfig.c
194 usr/src/lib/gss_mechs/mech_krb5/krb5/os/port2ip.c
195 usr/src/lib/gss_mechs/mech_krb5/krb5/os/prompter.c
196 usr/src/lib/gss_mechs/mech_krb5/krb5/os/promptusr.c
197 usr/src/lib/gss_mechs/mech_krb5/krb5/os/read_msg.c
198 usr/src/lib/gss_mechs/mech_krb5/krb5/os/read_pwd.c
199 usr/src/lib/gss_mechs/mech_krb5/krb5/os/realmdom.c
200 usr/src/lib/gss_mechs/mech_krb5/krb5/os/realmd_iter.c
201 usr/src/lib/gss_mechs/mech_krb5/krb5/os/thread_safe.c
202 usr/src/lib/gss_mechs/mech_krb5/krb5/os/unlck_file.c
203 usr/src/lib/gss_mechs/mech_krb5/krb5/os/ustime.c
204 usr/src/lib/gss_mechs/mech_krb5/krb5/os/write_msg.c
205 usr/src/lib/gss_mechs/mech_krb5/krb5/posix/daemon.c
206 usr/src/lib/gss_mechs/mech_krb5/krb5/posix/setenv.c
207 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_base.h
208 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_conv.c
209 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_io.h
210 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc_none.c
211 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rc-int.h
212 usr/src/lib/gss_mechs/mech_krb5/krb5/rcache/rcfns.c
213 usr/src/lib/gss_mechs/mech_krb5/mech/add_cred.c
214 usr/src/lib/gss_mechs/mech_krb5/mech/compare_name.c
215 usr/src/lib/gss_mechs/mech_krb5/mech/context_time.c
216 usr/src/lib/gss_mechs/mech_krb5/mech/copy_ccache.c
217 usr/src/lib/gss_mechs/mech_krb5/mech/disp_com_status.c
218 usr/src/lib/gss_mechs/mech_krb5/mech/disp_major_status.c
219 usr/src/lib/gss_mechs/mech_krb5/mech/disp_name.c
220 usr/src/lib/gss_mechs/mech_krb5/mech/disp_status.c
221 usr/src/lib/gss_mechs/mech_krb5/mech/export_name.c
222 usr/src/lib/gss_mechs/mech_krb5/mech/export_sec_context.c
223 usr/src/lib/gss_mechs/mech_krb5/mech/gss_libinit.h
224 usr/src/lib/gss_mechs/mech_krb5/mech/import_name.c
225 usr/src/lib/gss_mechs/mech_krb5/mech/indicate_mechs.c
226 usr/src/lib/gss_mechs/mech_krb5/mech/inq_cred.c
227 usr/src/lib/gss_mechs/mech_krb5/mech/inq_names.c
228 usr/src/lib/gss_mechs/mech_krb5/mech/oid_ops.c
229 usr/src/lib/gss_mechs/mech_krb5/mech/process_context_token.c
230 usr/src/lib/gss_mechs/mech_krb5/mech/rel_cred.c
231 usr/src/lib/gss_mechs/mech_krb5/mech/rel_name.c
232 usr/src/lib/gss_mechs/mech_krb5/mech/rel_oid.c
233 usr/src/lib/gss_mechs/mech_krb5/mech/set_allowable_etypes.c
234 usr/src/lib/gss_mechs/mech_krb5/mech/util_buffer.c
235 usr/src/lib/gss_mechs/mech_krb5/mech/util_ctxsetup.c
236 usr/src/lib/gss_mechs/mech_krb5/mech/util_dup.c
237 usr/src/lib/gss_mechs/mech_krb5/mech/utl_nohash_validate.c
238 usr/src/lib/gss_mechs/mech_krb5/profile/prof_err.h
239 usr/src/lib/gss_mechs/mech_krb5/profile/prof_get.c
240 usr/src/lib/gss_mechs/mech_krb5/profile/prof_set.c
241 usr/src/lib/gss_mechs/mech_krb5/support/errors.c
242 usr/src/lib/gss_mechs/mech_krb5/support/fake-addrinfo.c
243 usr/src/lib/gss_mechs/mech_krb5/support/init-addrinfo.c
244 usr/src/lib/gss_mechs/mech_krb5/support/supp-int.h
245 usr/src/lib/gss_mechs/mech_krb5/support/threads.c
246 usr/src/lib/gss_mechs/mech_krb5/support/utf8.c
247 usr/src/lib/krb5/dyn/dyn_append.c
248 usr/src/lib/krb5/dyn/dyn_create.c
249 usr/src/lib/krb5/dyn/dyn_debug.c
250 usr/src/lib/krb5/dyn/dyn_delete.c
251 usr/src/lib/krb5/dyn/dyn_initzero.c
252 usr/src/lib/krb5/dyn/dyn_insert.c
253 usr/src/lib/krb5/dyn/dyn_paranoid.c
254 usr/src/lib/krb5/dyn/dyn_put.c
255 usr/src/lib/krb5/dyn/dyn_realloc.c
256 usr/src/lib/krb5/dyn/dyn_size.c
257 usr/src/lib/krb5/kadm5/admin_internal.h
258 usr/src/lib/krb5/kadm5/admin_xdr.h
```

## new/exception\_lists/copyright

5

```
259 usr/src/lib/krb5/kadm5/chpass_util_strings.h
260 usr/src/lib/krb5/kadm5/clnt/client_handle.c
261 usr/src/lib/krb5/kadm5/clnt/clnt_chpass_util.c
262 usr/src/lib/krb5/kadm5/kadm_rpc.h
263 usr/src/lib/krb5/kadm5/misc_free.c
264 usr/src/lib/krb5/kadm5/srv/server_dict.c
265 usr/src/lib/krb5/kadm5/srv/server_handle.c
266 usr/src/lib/krb5/kadm5/srv/server_misc.c
267 usr/src/lib/krb5/kadm5/srv/svr_iters.c
268 usr/src/lib/krb5/kadm5/srv/svr_misc_free.c
269 usr/src/lib/krb5/kadm5/srv/svr_policy.c
270 usr/src/lib/krb5/kadm5/srv/xdr_alloc.c
271 usr/src/lib/krb5/kdb/adb_err.h
272 usr/src/lib/krb5/kdb/kdb5.h
273 usr/src/lib/krb5/kdb/keytab.c
274 usr/src/lib/krb5/plugins/kdb/db2/adb_policy.c
275 usr/src/lib/krb5/plugins/kdb/db2/kdb_db2.h
276 usr/src/lib/krb5/plugins/kdb/db2/kdb_xdr.c
277 usr/src/lib/krb5/plugins/kdb/db2/kdb_xdr.h
278 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_close.c
279 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_conv.c
280 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_delete.c
281 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_get.c
282 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_open.c
283 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_overflow.c
284 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_page.c
285 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_put.c
286 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_search.c
287 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_split.c
288 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_utils.c
289 usr/src/lib/krb5/plugins/kdb/db2/libdb2/db/db.c
290 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_bigkey.c
291 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_func.c
292 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_log2.c
293 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_page.c
294 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash.c
295 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hsearch.c
296 usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-ndbm.h
297 usr/src/lib/krb5/plugins/kdb/db2/libdb2/mpool/mpool.c
298 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_close.c
299 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_delete.c
300 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_get.c
301 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_open.c
302 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_put.c
303 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_search.c
304 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_seq.c
305 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_utils.c
306 usr/src/lib/krb5/plugins/kdb/db2/pol_xdr.c
307 usr/src/lib/krb5/plugins/kdb/db2/policy_db.h
308 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_xdr.c
309 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_xdr.h
310 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_err.c
311 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_err.h
312 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_fetch_mkey.c
313 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_handle.c
314 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_handle.h
315 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_krbcontainer.c
316 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_krbcontainer.h
317 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_principal.h
318 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_pwd_policy.h
319 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_realm.h
320 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_rights.c
321 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_stash.h
322 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_services.c
323 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_services.h
324 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_tkt_policy.c
```

## new/exception\_lists/copyright

6

```
325 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_tkt_policy.h
326 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_accessor.c
327 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_accessor.h
328 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_lib.c
329 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_profile.c
330 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_srv.c
331 usr/src/lib/krb5/ss/copyright.h
332 usr/src/lib/krb5/ss/mit-sipb-copyright.h
333 usr/src/lib/krb5/ss/options.c
334 usr/src/lib/krb5/ss/std_rqs.c
335 usr/src/lib/krb5/ss/utls.c
336 usr/src/lib/librtp/common/*.ch
337 usr/src/lib/librtp/common/[CRT]*
338 usr/src/test/zfs-tests/tests/functional/history/*Z
339 usr/src/test/zfs-tests/tests/functional/history/*txt
340 usr/src/uts/intel/nsmb/ioc_check.ref
341 usr/src/uts/intel/os/splashimage.xpm
342 usr/src/uts/common/gssapi/mechs/krb5/crypto/block_size.c
343 usr/src/uts/common/gssapi/mechs/krb5/crypto/checksum_length.c
344 usr/src/uts/common/gssapi/mechs/krb5/crypto/des/f_parity.c
345 usr/src/uts/common/gssapi/mechs/krb5/crypto/des/weak_key.c
346 usr/src/uts/common/gssapi/mechs/krb5/crypto/encrypt_length.c
347 usr/src/uts/common/gssapi/mechs/krb5/crypto/hash_provider/hash_crc32.c
348 usr/src/uts/common/gssapi/mechs/krb5/include/aes_s2k.h
349 usr/src/uts/common/gssapi/mechs/krb5/include/auth_con.h
350 usr/src/uts/common/gssapi/mechs/krb5/include/cksumtypes.h
351 usr/src/uts/common/gssapi/mechs/krb5/include/crc-32.h
352 usr/src/uts/common/gssapi/mechs/krb5/include/dk.h
353 usr/src/uts/common/gssapi/mechs/krb5/include/enc_provider.h
354 usr/src/uts/common/gssapi/mechs/krb5/include/etypes.h
355 usr/src/uts/common/gssapi/mechs/krb5/include/raw.h
356 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb_copy_auth.c
357 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb_ser_adata.c
358 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb_ser_addr.c
359 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb_ser_auth.c
360 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb_ser_cksum.c
361 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb_ser_ctx.c
362 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb_ser_key.c
363 usr/src/uts/common/gssapi/mechs/krb5/krb5/os/toffset.c
364 usr/src/uts/common/gssapi/mechs/krb5/mech/util_seed.c
365 usr/src/uts/common/gssapi/mechs/krb5/mech/util_seqnum.c
366 usr/src/uts/common/gssapi/mechs/krb5/mech/val_cred.c
367 usr/src/uts/common/io/ixgbe/ixgbe_82598.c
368 usr/src/uts/common/io/ixgbe/ixgbe_82599.c
369 usr/src/uts/common/io/ixgbe/ixgbe_api.c
370 usr/src/uts/common/io/ixgbe/ixgbe_api.h
371 usr/src/uts/common/io/ixgbe/ixgbe_common.c
372 usr/src/uts/common/io/ixgbe/ixgbe_common.h
373 usr/src/uts/common/io/ixgbe/ixgbe_osdep.h
374 usr/src/uts/common/io/ixgbe/ixgbe_phy.c
375 usr/src/uts/common/io/ixgbe/ixgbe_phy.h
376 usr/src/uts/common/io/ixgbe/ixgbe_type.h
377 usr/src/uts/common/fs/zfs/THIRDPARTYLICENSE.lz4.descrip
378 usr/src/uts/sparc/nsmb/ioc_check.ref
```

## new/exception\_lists/cstyle

1

\*\*\*\*\*  
34946 Fri Jan 17 07:29:13 2014

## 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/nsrtok.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
```

## new/exception\_lists/cstyle

2

```
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/crc64table.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/afstring2key.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_shal.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
```

## new/exception\_lists/cstyle

3

```
128 usr/src/lib/gss_mechs/mech_krb5/include/osconf.h
129 usr/src/lib/gss_mechs/mech_krb5/include/port-sockets.h
130 usr/src/lib/gss_mechs/mech_krb5/include/preauth_plugin.h
131 usr/src/lib/gss_mechs/mech_krb5/include/socket-utils.h
132 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_decode.c
133 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_decode.h
134 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_encode.c
135 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_encode.h
136 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_get.c
137 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_get.h
138 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_decode.c
139 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_decode.h
140 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_encode.c
141 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_k_encode.h
142 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_make.c
143 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_make.h
144 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_misc.c
145 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1_misc.h
146 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/asn1buf.h
147 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krb5_decode.c
148 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krb5_encode.c
149 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/krbasn1.h
150 usr/src/lib/gss_mechs/mech_krb5/krb5/asn.1/ldap_key_seq.c
151 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc_file.c
152 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc_memory.c
153 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc_retr.c
154 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cc_int.h
155 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccbase.c
156 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/cccopy.c
157 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccdefault.c
158 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccdefops.c
159 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ccfns.c
160 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/fcc.h
161 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/scc.h
162 usr/src/lib/gss_mechs/mech_krb5/krb5/ccache/ser_cc.c
163 usr/src/lib/gss_mechs/mech_krb5/krb5/error_tables/adm_err.h
164 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/file/ktfile.h
165 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/kt_file.c
166 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/kt_srvtab.c
167 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/kt_int.h
168 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktadd.c
169 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktbase.c
170 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktdefault.c
171 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktfns.c
172 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktfr_entry.c
173 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/ktremove.c
174 usr/src/lib/gss_mechs/mech_krb5/krb5/keytab/read_servi.c
175 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_comp.c
176 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_order.c
177 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/addr_srch.c
178 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/appdefault.c
179 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/bld_pr_ext.c
180 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/bld_princ.c
181 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/chk_trans.c
182 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/cleanup.h
183 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/conv_princ.c
184 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_addrs.c
185 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_creds.c
186 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_data.c
187 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/copy_tick.c
188 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/cp_key_cnt.c
189 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/decode_kdc.c
190 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/decrypt_tk.c
191 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/deltat.c
192 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/enc_helper.c
193 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/encode_kdc.c
```

## new/exception\_lists/cstyle

4

```
194 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/encrypt_tk.c
195 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/free_rtree.c
196 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/fwd_tgt.c
197 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gc_frm_kdc.c
198 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gc_via_tkt.c
199 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gen_seqnum.c
200 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gen_subkey.c
201 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/get_creds.c
202 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/get_in_tkt.c
203 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gic_keytab.c
204 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gic_opt.c
205 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/gic_pwd.c
206 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/init_keyblock.c
207 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/int_proto.h
208 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/kdc_rep_dc.c
209 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/kerrs.c
210 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_error.c
211 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_priv.c
212 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_rep.c
213 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_req_ext.c
214 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_req.c
215 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/mk_safe.c
216 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/pac.c
217 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/pr_to_salt.c
218 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/preauth.c
219 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/preauth2.c
220 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/princ_comp.c
221 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_cred.c
222 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_error.c
223 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_priv.c
224 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_rep.c
225 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_req_dec.c
226 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_req.c
227 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/rd_safe.c
228 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/recvauth.c
229 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/send_tgs.c
230 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/sendauth.c
231 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/set_realm.c
232 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/srv_rcache.c
233 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/str_conv.c
234 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/tgtname.c
235 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/valid_times.c
236 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/vic_opt.c
237 usr/src/lib/gss_mechs/mech_krb5/krb5/krb/walk_rtree.c
238 usr/src/lib/gss_mechs/mech_krb5/krb5/os/accessor.c
239 usr/src/lib/gss_mechs/mech_krb5/krb5/os/an_to_in.c
240 usr/src/lib/gss_mechs/mech_krb5/krb5/os/ccdefine.c
241 usr/src/lib/gss_mechs/mech_krb5/krb5/os/changepw.c
242 usr/src/lib/gss_mechs/mech_krb5/krb5/os/def_realm.c
243 usr/src/lib/gss_mechs/mech_krb5/krb5/os/dns glue.c
244 usr/src/lib/gss_mechs/mech_krb5/krb5/os/dns glue.h
245 usr/src/lib/gss_mechs/mech_krb5/krb5/os/dnssrv.c
246 usr/src/lib/gss_mechs/mech_krb5/krb5/os/foreachaddr.c
247 usr/src/lib/gss_mechs/mech_krb5/krb5/os/free_hstrl.c
248 usr/src/lib/gss_mechs/mech_krb5/krb5/os/free_krbhs.c
249 usr/src/lib/gss_mechs/mech_krb5/krb5/os/full_ipadr.c
250 usr/src/lib/gss_mechs/mech_krb5/krb5/os/gen_port.c
251 usr/src/lib/gss_mechs/mech_krb5/krb5/os/gen_rname.c
252 usr/src/lib/gss_mechs/mech_krb5/krb5/os/genaddrs.c
253 usr/src/lib/gss_mechs/mech_krb5/krb5/os/get_krbhst.c
254 usr/src/lib/gss_mechs/mech_krb5/krb5/os/gmt_mktime.c
255 usr/src/lib/gss_mechs/mech_krb5/krb5/os/hostaddr.c
256 usr/src/lib/gss_mechs/mech_krb5/krb5/os/hst_realm.c
257 usr/src/lib/gss_mechs/mech_krb5/krb5/os/ktdefine.c
258 usr/src/lib/gss_mechs/mech_krb5/krb5/os/kuserok.c
259 usr/src/lib/gss_mechs/mech_krb5/krb5/os/localaddr.c
```

## new/exception\_lists/cstyle

```
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/realmdom.c
273 usr/src/lib/gss_mechs/mech_krb5/krb5/os/realmd_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/unlock_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
```

5

## new/exception\_lists/cstyle

```
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_paranoia.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
```

6

```
392 usr/src/lib/krb5/kdb/kdb_default.c
393 usr/src/lib/krb5/kdb/kdb5.c
394 usr/src/lib/krb5/kdb/kdb5.h
395 usr/src/lib/krb5/kdb/keytab.c
396 usr/src/lib/krb5/plugins/kdb/db2/adb_openclose.c
397 usr/src/lib/krb5/plugins/kdb/db2/adb_policy.c
398 usr/src/lib/krb5/plugins/kdb/db2/db2_exp.c
399 usr/src/lib/krb5/plugins/kdb/db2/kdb_compat.h
400 usr/src/lib/krb5/plugins/kdb/db2/kdb_db2.c
401 usr/src/lib/krb5/plugins/kdb/db2/kdb_db2.h
402 usr/src/lib/krb5/plugins/kdb/db2/kdb_xdr.c
403 usr/src/lib/krb5/plugins/kdb/db2/kdb_xdr.h
404 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_close.c
405 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_conv.c
406 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_debug.c
407 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_delete.c
408 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_get.c
409 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_open.c
410 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_overflow.c
411 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_page.c
412 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_put.c
413 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_search.c
414 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_seq.c
415 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_split.c
416 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/bt_utils.c
417 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/btree.h
418 usr/src/lib/krb5/plugins/kdb/db2/libdb2/btree/extern.h
419 usr/src/lib/krb5/plugins/kdb/db2/libdb2/db/db.c
420 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/dbm.c
421 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/extern.h
422 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_bigkey.c
423 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_func.c
424 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_log2.c
425 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash_page.c
426 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash.c
427 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hash.h
428 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/hsearch.c
429 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/page.h
430 usr/src/lib/krb5/plugins/kdb/db2/libdb2/hash/search.h
431 usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-int.h
432 usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-ndbm.h
433 usr/src/lib/krb5/plugins/kdb/db2/libdb2/include/db-queue.h
434 usr/src/lib/krb5/plugins/kdb/db2/libdb2/mpool/mpool.c
435 usr/src/lib/krb5/plugins/kdb/db2/libdb2/mpool/mpool.h
436 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/extern.h
437 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_close.c
438 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_delete.c
439 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_get.c
440 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_open.c
441 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_put.c
442 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_search.c
443 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_seq.c
444 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/rec_utils.c
445 usr/src/lib/krb5/plugins/kdb/db2/libdb2/recno/recno.h
446 usr/src/lib/krb5/plugins/kdb/db2/pol_xdr.c
447 usr/src/lib/krb5/plugins/kdb/db2/policy_db.h
448 usr/src/lib/krb5/plugins/kdb/ldap/ldap_exp.c
449 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_ldap_conn.c
450 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_ldap.c
451 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_ldap.h
452 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_xdr.c
453 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/kdb_xdr.h
454 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_create.c
455 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_err.c
456 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_err.h
457 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_fetch_mkey.c
```

```
458 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_handle.c
459 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_handle.h
460 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_krbcontainer.c
461 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_krbcontainer.h
462 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_main.h
463 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_misc.c
464 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_misc.h
465 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_principal.c
466 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_principal.h
467 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_principal2.c
468 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_pwd_policy.c
469 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_pwd_policy.h
470 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_realm.c
471 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_realm.h
472 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_rights.c
473 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_stash.c
474 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_service_stash.h
475 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_services.c
476 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_services.h
477 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_tkt_policy.c
478 usr/src/lib/krb5/plugins/kdb/ldap/libkdb_ldap/ldap_tkt_policy.h
479 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_accessor.c
480 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_accessor.h
481 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_clnt.c
482 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_crypto_openssl.c
483 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_crypto_openssl.h
484 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_crypto.h
485 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_identity.c
486 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_lib.c
487 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_matching.c
488 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_profile.c
489 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit_srv.c
490 usr/src/lib/krb5/plugins/preauth/pkinit/pkinit.h
491 usr/src/lib/krb5/ss/copyright.h
492 usr/src/lib/krb5/ss/data.c
493 usr/src/lib/krb5/ss/error.c
494 usr/src/lib/krb5/ss/execute_cmd.c
495 usr/src/lib/krb5/ss/help.c
496 usr/src/lib/krb5/ss/invocation.c
497 usr/src/lib/krb5/ss/list_rqs.c
498 usr/src/lib/krb5/ss/listen.c
499 usr/src/lib/krb5/ss/mit-sipb-copyright.h
500 usr/src/lib/krb5/ss/mk_cmds.c
501 usr/src/lib/krb5/ss/options.c
502 usr/src/lib/krb5/ss/pager.c
503 usr/src/lib/krb5/ss/parse.c
504 usr/src/lib/krb5/ss/prompt.c
505 usr/src/lib/krb5/ss/request_tbl.c
506 usr/src/lib/krb5/ss/requests.c
507 usr/src/lib/krb5/ss/ss_internal.h
508 usr/src/lib/krb5/ss/ss.h
509 usr/src/lib/krb5/ss/std_rqs.c
510 usr/src/lib/krb5/ss/utls.c
511 usr/src/lib/libgss/g_glue.c
512 usr/src/lib/libkrb5/common/base.h
513 usr/src/lib/libkrb5/common/choose.h
514 usr/src/lib/libkrb5/common/edge.c
515 usr/src/lib/libkrb5/common/edge.h
516 usr/src/lib/libkrb5/common/migrate.c
517 usr/src/lib/libkrb5/common/migrate.h
518 usr/src/lib/libkrb5/common/p2p.c
519 usr/src/lib/libkrb5/common/p2p.h
520 usr/src/lib/libkrb5/common/pcost.c
521 usr/src/lib/libkrb5/common/pcost.h
522 usr/src/lib/libkrb5/common/port.c
523 usr/src/lib/libkrb5/common/port.h
```

```
524 usr/src/lib/librstp/common/portinfo.c
525 usr/src/lib/librstp/common/portinfo.h
526 usr/src/lib/librstp/common/rolesel.c
527 usr/src/lib/librstp/common/rolesel.h
528 usr/src/lib/librstp/common/roletxns.c
529 usr/src/lib/librstp/common/roletxns.h
530 usr/src/lib/librstp/common/statmch.c
531 usr/src/lib/librstp/common/statmch.h
532 usr/src/lib/librstp/common/stp_bpdu.h
533 usr/src/lib/librstp/common/stp_in.c
534 usr/src/lib/librstp/common/stp_in.h
535 usr/src/lib/librstp/common/stp_to.h
536 usr/src/lib/librstp/common/stp_vectors.h
537 usr/src/lib/librstp/common/stpm.c
538 usr/src/lib/librstp/common/stpm.h
539 usr/src/lib/librstp/common/stpmgmt.c
540 usr/src/lib/librstp/common/sttrans.c
541 usr/src/lib/librstp/common/sttrans.h
542 usr/src/lib/librstp/common/times.c
543 usr/src/lib/librstp/common/times.h
544 usr/src/lib/librstp/common/topoch.c
545 usr/src/lib/librstp/common/topoch.h
546 usr/src/lib/librstp/common/transmit.c
547 usr/src/lib/librstp/common/transmit.h
548 usr/src/lib/librstp/common/uid_stp.h
549 usr/src/lib/librstp/common/vector.c
550 usr/src/lib/librstp/common/vector.h
551 usr/src/uts/common/gssapi/gssapi.h
552 usr/src/uts/common/gssapi/mechs/krb5/crypto/block_size.c
553 usr/src/uts/common/gssapi/mechs/krb5/crypto/checksum_length.c
554 usr/src/uts/common/gssapi/mechs/krb5/crypto/cksumtypes.c
555 usr/src/uts/common/gssapi/mechs/krb5/crypto/combine_keys.c
556 usr/src/uts/common/gssapi/mechs/krb5/crypto/crc32/crc32.c
557 usr/src/uts/common/gssapi/mechs/krb5/crypto/decrypt.c
558 usr/src/uts/common/gssapi/mechs/krb5/crypto/default_state.c
559 usr/src/uts/common/gssapi/mechs/krb5/crypto/des/d3_cbc.c
560 usr/src/uts/common/gssapi/mechs/krb5/crypto/des/f_cbc.c
561 usr/src/uts/common/gssapi/mechs/krb5/crypto/des/f_parity.c
562 usr/src/uts/common/gssapi/mechs/krb5/crypto/des/weak_key.c
563 usr/src/uts/common/gssapi/mechs/krb5/crypto/dk/checksum.c
564 usr/src/uts/common/gssapi/mechs/krb5/crypto/dk/derive.c
565 usr/src/uts/common/gssapi/mechs/krb5/crypto/dk/dk_decrypt.c
566 usr/src/uts/common/gssapi/mechs/krb5/crypto/dk/dk_encrypt.c
567 usr/src/uts/common/gssapi/mechs/krb5/crypto/enc_provider/arcfour_provider.c
568 usr/src/uts/common/gssapi/mechs/krb5/crypto/enc_provider/des.c
569 usr/src/uts/common/gssapi/mechs/krb5/crypto/enc_provider/des3.c
570 usr/src/uts/common/gssapi/mechs/krb5/crypto/encrypt_length.c
571 usr/src/uts/common/gssapi/mechs/krb5/crypto/encrypt.c
572 usr/src/uts/common/gssapi/mechs/krb5/crypto/etypes.c
573 usr/src/uts/common/gssapi/mechs/krb5/crypto/hash_provider/hash_crc32.c
574 usr/src/uts/common/gssapi/mechs/krb5/crypto/hash_provider/hash_kmd5.c
575 usr/src/uts/common/gssapi/mechs/krb5/crypto/hash_provider/hash_ksha1.c
576 usr/src/uts/common/gssapi/mechs/krb5/crypto/hmac.c
577 usr/src/uts/common/gssapi/mechs/krb5/crypto/keyhash_provider/descbc.c
578 usr/src/uts/common/gssapi/mechs/krb5/crypto/keyhash_provider/k_hmac_md5.c
579 usr/src/uts/common/gssapi/mechs/krb5/crypto/keyhash_provider/k5_kmd5des.c
580 usr/src/uts/common/gssapi/mechs/krb5/crypto/make_checksum.c
581 usr/src/uts/common/gssapi/mechs/krb5/crypto/mandatory_sumtype.c
582 usr/src/uts/common/gssapi/mechs/krb5/crypto/nfold.c
583 usr/src/uts/common/gssapi/mechs/krb5/crypto/old/old_decrypt.c
584 usr/src/uts/common/gssapi/mechs/krb5/crypto/old/old_encrypt.c
585 usr/src/uts/common/gssapi/mechs/krb5/crypto/prng.c
586 usr/src/uts/common/gssapi/mechs/krb5/crypto/raw/raw_decrypt.c
587 usr/src/uts/common/gssapi/mechs/krb5/crypto/raw/raw_encrypt.c
588 usr/src/uts/common/gssapi/mechs/krb5/crypto/verify_checksum.c
589 usr/src/uts/common/gssapi/mechs/krb5/include/aes_s2k.h
```

```
590 usr/src/uts/common/gssapi/mechs/krb5/include/auth_con.h
591 usr/src/uts/common/gssapi/mechs/krb5/include/cksumtypes.h
592 usr/src/uts/common/gssapi/mechs/krb5/include/crc-32.h
593 usr/src/uts/common/gssapi/mechs/krb5/include/des_int.h
594 usr/src/uts/common/gssapi/mechs/krb5/include/dk.h
595 usr/src/uts/common/gssapi/mechs/krb5/include/enc_provider.h
596 usr/src/uts/common/gssapi/mechs/krb5/include/etypes.h
597 usr/src/uts/common/gssapi/mechs/krb5/include/gssapi_generic.h
598 usr/src/uts/common/gssapi/mechs/krb5/include/gssapi_krb5.h
599 usr/src/uts/common/gssapi/mechs/krb5/include/gssapiP_generic.h
600 usr/src/uts/common/gssapi/mechs/krb5/include/gssapiP_krb5.h
601 usr/src/uts/common/gssapi/mechs/krb5/include/hash_provider.h
602 usr/src/uts/common/gssapi/mechs/krb5/include/k5-int.h
603 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_16.h
604 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_32.h
605 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_64.h
606 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_16.h
607 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_32.h
608 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_64.h
609 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform.h
610 usr/src/uts/common/gssapi/mechs/krb5/include/k5-thread.h
611 usr/src/uts/common/gssapi/mechs/krb5/include/keyhash_provider.h
612 usr/src/uts/common/gssapi/mechs/krb5/include/krb5.h
613 usr/src/uts/common/gssapi/mechs/krb5/include/old.h
614 usr/src/uts/common/gssapi/mechs/krb5/include/raw.h
615 usr/src/uts/common/gssapi/mechs/krb5/include/rsa-md4.h
616 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_athctr.c
617 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_auth.c
618 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_cksum.c
619 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_key.c
620 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/copy_princ.c
621 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/init_ctx.c
622 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/kfree.c
623 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/parse.c
624 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_actx.c
625 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_adata.c
626 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_addr.c
627 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_auth.c
628 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_ctxsum.c
629 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_ctx.c
630 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_key.c
631 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/ser_princ.c
632 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/serialize.c
633 usr/src/uts/common/gssapi/mechs/krb5/krb5/krb/unparse.c
634 usr/src/uts/common/gssapi/mechs/krb5/krb5/os/c_ustime.c
635 usr/src/uts/common/gssapi/mechs/krb5/krb5/os/init_os_ctx.c
636 usr/src/uts/common/gssapi/mechs/krb5/krb5/os/timeofday.c
637 usr/src/uts/common/gssapi/mechs/krb5/krb5/os/toffset.c
638 usr/src/uts/common/gssapi/mechs/krb5/mech/delete_sec_context.c
639 usr/src/uts/common/gssapi/mechs/krb5/mech/gssapi_krb5.c
640 usr/src/uts/common/gssapi/mechs/krb5/mech/import_sec_context.c
641 usr/src/uts/common/gssapi/mechs/krb5/mech/k5seal.c
642 usr/src/uts/common/gssapi/mechs/krb5/mech/k5sealv3.c
643 usr/src/uts/common/gssapi/mechs/krb5/mech/k5unseal.c
644 usr/src/uts/common/gssapi/mechs/krb5/mech/seal.c
645 usr/src/uts/common/gssapi/mechs/krb5/mech/ser_sctx.c
646 usr/src/uts/common/gssapi/mechs/krb5/mech/sign.c
647 usr/src/uts/common/gssapi/mechs/krb5/mech/unseal.c
648 usr/src/uts/common/gssapi/mechs/krb5/mech/util_crypt.c
649 usr/src/uts/common/gssapi/mechs/krb5/mech/util_ordering.c
650 usr/src/uts/common/gssapi/mechs/krb5/mech/util_seed.c
651 usr/src/uts/common/gssapi/mechs/krb5/mech/util_segnum.c
652 usr/src/uts/common/gssapi/mechs/krb5/mech/util_set.c
653 usr/src/uts/common/gssapi/mechs/krb5/mech/util_token.c
654 usr/src/uts/common/gssapi/mechs/krb5/mech/util_validate.c
655 usr/src/uts/common/gssapi/mechs/krb5/mech/val_cred.c
```



```

656 usr/src/uts/common/gssapi/mechs/krb5/mech/verify.c
657 usr/src/uts/common/gssapi/mechs/krb5/mech/wrap_size_limit.c
658 usr/src/uts/common/io/e1000api/e1000_80003es2lan.c
659 usr/src/uts/common/io/e1000api/e1000_80003es2lan.h
660 usr/src/uts/common/io/e1000api/e1000_82540.c
661 usr/src/uts/common/io/e1000api/e1000_82541.c
662 usr/src/uts/common/io/e1000api/e1000_82541.h
663 usr/src/uts/common/io/e1000api/e1000_82542.c
664 usr/src/uts/common/io/e1000api/e1000_82543.c
665 usr/src/uts/common/io/e1000api/e1000_82543.h
666 usr/src/uts/common/io/e1000api/e1000_82571.c
667 usr/src/uts/common/io/e1000api/e1000_82571.h
668 usr/src/uts/common/io/e1000api/e1000_82575.c
669 usr/src/uts/common/io/e1000api/e1000_82575.h
670 usr/src/uts/common/io/e1000api/e1000_api.c
671 usr/src/uts/common/io/e1000api/e1000_api.h
672 usr/src/uts/common/io/e1000api/e1000_defines.h
673 usr/src/uts/common/io/e1000api/e1000_hw.h
674 usr/src/uts/common/io/e1000api/e1000_i210.c
675 usr/src/uts/common/io/e1000api/e1000_i210.h
676 usr/src/uts/common/io/e1000api/e1000_ich8lan.c
677 usr/src/uts/common/io/e1000api/e1000_ich8lan.h
678 usr/src/uts/common/io/e1000api/e1000_mac.c
679 usr/src/uts/common/io/e1000api/e1000_mac.h
680 usr/src/uts/common/io/e1000api/e1000_manage.c
681 usr/src/uts/common/io/e1000api/e1000_manage.h
682 usr/src/uts/common/io/e1000api/e1000_mbx.c
683 usr/src/uts/common/io/e1000api/e1000_mbx.h
684 usr/src/uts/common/io/e1000api/e1000_nvm.c
685 usr/src/uts/common/io/e1000api/e1000_nvm.h
686 usr/src/uts/common/io/e1000api/e1000_phy.c
687 usr/src/uts/common/io/e1000api/e1000_phy.h
688 usr/src/uts/common/io/e1000api/e1000_regs.h
689 usr/src/uts/common/io/e1000api/e1000_vf.c
690 usr/src/uts/common/io/e1000api/e1000_vf.h
691 usr/src/uts/common/io/ixgbe/ixgbe_82598.c
692 usr/src/uts/common/io/ixgbe/ixgbe_82598.h
693 usr/src/uts/common/io/ixgbe/ixgbe_82599.c
694 usr/src/uts/common/io/ixgbe/ixgbe_82599.h
695 usr/src/uts/common/io/ixgbe/ixgbe_api.c
696 usr/src/uts/common/io/ixgbe/ixgbe_api.h
697 usr/src/uts/common/io/ixgbe/ixgbe_common.c
698 usr/src/uts/common/io/ixgbe/ixgbe_common.h
699 usr/src/uts/common/io/ixgbe/ixgbe_mbx.c
700 usr/src/uts/common/io/ixgbe/ixgbe_mbx.h
701 usr/src/uts/common/io/ixgbe/ixgbe_osdep.h
702 usr/src/uts/common/io/ixgbe/ixgbe_phy.c
703 usr/src/uts/common/io/ixgbe/ixgbe_phy.h
704 usr/src/uts/common/io/ixgbe/ixgbe_type.h
705 usr/src/uts/common/io/ixgbe/ixgbe_x540.c
706 usr/src/uts/common/io/ixgbe/ixgbe_x540.h
707 usr/src/common/acpica
707 usr/src/uts/intel/io/acpica/debugger/dbcmds.c
708 usr/src/uts/intel/io/acpica/debugger/dbdisply.c
709 usr/src/uts/intel/io/acpica/debugger/dbexec.c
710 usr/src/uts/intel/io/acpica/debugger/dbfileio.c
711 usr/src/uts/intel/io/acpica/debugger/dbhistry.c
712 usr/src/uts/intel/io/acpica/debugger/dbinput.c
713 usr/src/uts/intel/io/acpica/debugger/dbmethod.c
714 usr/src/uts/intel/io/acpica/debugger/dbnames.c
715 usr/src/uts/intel/io/acpica/debugger/dbstats.c
716 usr/src/uts/intel/io/acpica/debugger/dbutils.c
717 usr/src/uts/intel/io/acpica/debugger/dbxface.c
718 usr/src/uts/intel/io/acpica/disassembler/dmbuffer.c
719 usr/src/uts/intel/io/acpica/disassembler/dmnames.c
720 usr/src/uts/intel/io/acpica/disassembler/dmobject.c

```

```

721 usr/src/uts/intel/io/acpica/disassembler/dmopcode.c
722 usr/src/uts/intel/io/acpica/disassembler/dmresrc.c
723 usr/src/uts/intel/io/acpica/disassembler/dmresrc1.c
724 usr/src/uts/intel/io/acpica/disassembler/dmresrcs.c
725 usr/src/uts/intel/io/acpica/disassembler/dmutils.c
726 usr/src/uts/intel/io/acpica/disassembler/dmwalk.c
727 usr/src/uts/intel/io/acpica/dispatcher/dsargs.c
728 usr/src/uts/intel/io/acpica/dispatcher/dscontrol.c
729 usr/src/uts/intel/io/acpica/dispatcher/dsfield.c
730 usr/src/uts/intel/io/acpica/dispatcher/dsinit.c
731 usr/src/uts/intel/io/acpica/dispatcher/dsmethod.c
732 usr/src/uts/intel/io/acpica/dispatcher/dsmthdat.c
733 usr/src/uts/intel/io/acpica/dispatcher/dsobject.c
734 usr/src/uts/intel/io/acpica/dispatcher/dsopcode.c
735 usr/src/uts/intel/io/acpica/dispatcher/dsutils.c
736 usr/src/uts/intel/io/acpica/dispatcher/dswexec.c
737 usr/src/uts/intel/io/acpica/dispatcher/dswload.c
738 usr/src/uts/intel/io/acpica/dispatcher/dswload2.c
739 usr/src/uts/intel/io/acpica/dispatcher/dswstate.c
740 usr/src/uts/intel/io/acpica/dispatcher/dswstate.c
741 usr/src/uts/intel/io/acpica/events/evevent.c
742 usr/src/uts/intel/io/acpica/events/evglock.c
743 usr/src/uts/intel/io/acpica/events/evgpe.c
744 usr/src/uts/intel/io/acpica/events/evgpeblk.c
745 usr/src/uts/intel/io/acpica/events/evgpeinit.c
746 usr/src/uts/intel/io/acpica/events/evgpeutil.c
747 usr/src/uts/intel/io/acpica/events/evmisc.c
748 usr/src/uts/intel/io/acpica/events/evregion.c
749 usr/src/uts/intel/io/acpica/events/evrgnini.c
750 usr/src/uts/intel/io/acpica/events/evsci.c
751 usr/src/uts/intel/io/acpica/events/evxface.c
752 usr/src/uts/intel/io/acpica/events/evxfevent.c
753 usr/src/uts/intel/io/acpica/events/evxfge.c
754 usr/src/uts/intel/io/acpica/events/evxfregn.c
755 usr/src/uts/intel/io/acpica/executer/exconfig.c
756 usr/src/uts/intel/io/acpica/executer/exconvt.c
757 usr/src/uts/intel/io/acpica/executer/excreate.c
758 usr/src/uts/intel/io/acpica/executer/exdebug.c
759 usr/src/uts/intel/io/acpica/executer/exdump.c
760 usr/src/uts/intel/io/acpica/executer/exfield.c
761 usr/src/uts/intel/io/acpica/executer/exfldio.c
762 usr/src/uts/intel/io/acpica/executer/exmisc.c
763 usr/src/uts/intel/io/acpica/executer/exmutex.c
764 usr/src/uts/intel/io/acpica/executer/exnames.c
765 usr/src/uts/intel/io/acpica/executer/exoparg1.c
766 usr/src/uts/intel/io/acpica/executer/exoparg2.c
767 usr/src/uts/intel/io/acpica/executer/exoparg3.c
768 usr/src/uts/intel/io/acpica/executer/exoparg6.c
769 usr/src/uts/intel/io/acpica/executer/exprep.c
770 usr/src/uts/intel/io/acpica/executer/exregion.c
771 usr/src/uts/intel/io/acpica/executer/exresnt.c
772 usr/src/uts/intel/io/acpica/executer/exresolv.c
773 usr/src/uts/intel/io/acpica/executer/exresop.c
774 usr/src/uts/intel/io/acpica/executer/exstore.c
775 usr/src/uts/intel/io/acpica/executer/exstoren.c
776 usr/src/uts/intel/io/acpica/executer/exstorob.c
777 usr/src/uts/intel/io/acpica/executer/exstry.c
778 usr/src/uts/intel/io/acpica/executer/exutils.c
779 usr/src/uts/intel/io/acpica/hardware/hwacpi.c
780 usr/src/uts/intel/io/acpica/hardware/hwgpe.c
781 usr/src/uts/intel/io/acpica/hardware/hwpci.c
782 usr/src/uts/intel/io/acpica/hardware/hwregs.c
783 usr/src/uts/intel/io/acpica/hardware/hwsleep.c
784 usr/src/uts/intel/io/acpica/hardware/hwtimer.c
785 usr/src/uts/intel/io/acpica/hardware/hwvalid.c
786 usr/src/uts/intel/io/acpica/hardware/hwxface.c

```

```
787 usr/src/uts/intel/io/acpica/namespace/nsaccess.c
788 usr/src/uts/intel/io/acpica/namespace/nsalloc.c
789 usr/src/uts/intel/io/acpica/namespace/nsdump.c
790 usr/src/uts/intel/io/acpica/namespace/nsdumpdv.c
791 usr/src/uts/intel/io/acpica/namespace/nseval.c
792 usr/src/uts/intel/io/acpica/namespace/nsinit.c
793 usr/src/uts/intel/io/acpica/namespace/nsload.c
794 usr/src/uts/intel/io/acpica/namespace/nsnames.c
795 usr/src/uts/intel/io/acpica/namespace/nsobject.c
796 usr/src/uts/intel/io/acpica/namespace/nsparse.c
797 usr/src/uts/intel/io/acpica/namespace/nspredef.c
798 usr/src/uts/intel/io/acpica/namespace/nsrepair.c
799 usr/src/uts/intel/io/acpica/namespace/nsrepair2.c
800 usr/src/uts/intel/io/acpica/namespace/nssearch.c
801 usr/src/uts/intel/io/acpica/namespace/nsutils.c
802 usr/src/uts/intel/io/acpica/namespace/nswalk.c
803 usr/src/uts/intel/io/acpica/namespace/nsxfeval.c
804 usr/src/uts/intel/io/acpica/namespace/nsxfname.c
805 usr/src/uts/intel/io/acpica/namespace/nsxfobj.c
806 usr/src/uts/intel/io/acpica/parser/psargs.c
807 usr/src/uts/intel/io/acpica/parser/psloop.c
808 usr/src/uts/intel/io/acpica/parser/psopcode.c
809 usr/src/uts/intel/io/acpica/parser/psparse.c
810 usr/src/uts/intel/io/acpica/parser/psscope.c
811 usr/src/uts/intel/io/acpica/parser/pstree.c
812 usr/src/uts/intel/io/acpica/parser/psutils.c
813 usr/src/uts/intel/io/acpica/parser/pswalk.c
814 usr/src/uts/intel/io/acpica/parser/psxface.c
815 usr/src/uts/intel/io/acpica/resources/rsaddr.c
816 usr/src/uts/intel/io/acpica/resources/rscale.c
817 usr/src/uts/intel/io/acpica/resources/rscreate.c
818 usr/src/uts/intel/io/acpica/resources/rsdump.c
819 usr/src/uts/intel/io/acpica/resources/rsinfo.c
820 usr/src/uts/intel/io/acpica/resources/rsio.c
821 usr/src/uts/intel/io/acpica/resources/rsirq.c
822 usr/src/uts/intel/io/acpica/resources/rslist.c
823 usr/src/uts/intel/io/acpica/resources/rsmemory.c
824 usr/src/uts/intel/io/acpica/resources/rsmisc.c
825 usr/src/uts/intel/io/acpica/resources/rsutils.c
826 usr/src/uts/intel/io/acpica/resources/rsxface.c
827 usr/src/uts/intel/io/acpica/tables/tbfadt.c
828 usr/src/uts/intel/io/acpica/tables/tbfind.c
829 usr/src/uts/intel/io/acpica/tables/tbinstal.c
830 usr/src/uts/intel/io/acpica/tables/tbutils.c
831 usr/src/uts/intel/io/acpica/tables/tbxface.c
832 usr/src/uts/intel/io/acpica/tables/tbxfoot.c
833 usr/src/uts/intel/io/acpica/utilities/utalloc.c
834 usr/src/uts/intel/io/acpica/utilities/utcache.c
835 usr/src/uts/intel/io/acpica/utilities/utclib.c
836 usr/src/uts/intel/io/acpica/utilities/utcopy.c
837 usr/src/uts/intel/io/acpica/utilities/utdebug.c
838 usr/src/uts/intel/io/acpica/utilities/utdecode.c
839 usr/src/uts/intel/io/acpica/utilities/utdelete.c
840 usr/src/uts/intel/io/acpica/utilities/uteval.c
841 usr/src/uts/intel/io/acpica/utilities/utglobal.c
842 usr/src/uts/intel/io/acpica/utilities/utids.c
843 usr/src/uts/intel/io/acpica/utilities/utinit.c
844 usr/src/uts/intel/io/acpica/utilities/utlock.c
845 usr/src/uts/intel/io/acpica/utilities/utmath.c
846 usr/src/uts/intel/io/acpica/utilities/utmisc.c
847 usr/src/uts/intel/io/acpica/utilities/utmutex.c
848 usr/src/uts/intel/io/acpica/utilities/utobject.c
849 usr/src/uts/intel/io/acpica/utilities/utosi.c
850 usr/src/uts/intel/io/acpica/utilities/utresrc.c
851 usr/src/uts/intel/io/acpica/utilities/utstate.c
852 usr/src/uts/intel/io/acpica/utilities/uttrack.c
```

```
853 usr/src/uts/intel/io/acpica/utilities/utxface.c
854 usr/src/uts/intel/io/acpica/utilities/utxferror.c
855 usr/src/uts/intel/sys/acpi/acapps.h
856 usr/src/uts/intel/sys/acpi/accommon.h
857 usr/src/uts/intel/sys/acpi/acconfig.h
858 usr/src/uts/intel/sys/acpi/acdebug.h
859 usr/src/uts/intel/sys/acpi/acdisasm.h
860 usr/src/uts/intel/sys/acpi/acdispat.h
861 usr/src/uts/intel/sys/acpi/acevents.h
862 usr/src/uts/intel/sys/acpi/acexcep.h
863 usr/src/uts/intel/sys/acpi/acglobal.h
864 usr/src/uts/intel/sys/acpi/achware.h
865 usr/src/uts/intel/sys/acpi/acinterp.h
866 usr/src/uts/intel/sys/acpi/aclocal.h
867 usr/src/uts/intel/sys/acpi/acmacros.h
868 usr/src/uts/intel/sys/acpi/acnames.h
869 usr/src/uts/intel/sys/acpi/acnamesp.h
870 usr/src/uts/intel/sys/acpi/acobject.h
871 usr/src/uts/intel/sys/acpi/acopcode.h
872 usr/src/uts/intel/sys/acpi/acoutput.h
873 usr/src/uts/intel/sys/acpi/acparser.h
874 usr/src/uts/intel/sys/acpi/acpi.h
875 usr/src/uts/intel/sys/acpi/acpiosxf.h
876 usr/src/uts/intel/sys/acpi/acpixf.h
877 usr/src/uts/intel/sys/acpi/acpredef.h
878 usr/src/uts/intel/sys/acpi/acresrc.h
879 usr/src/uts/intel/sys/acpi/acrestyp.h
880 usr/src/uts/intel/sys/acpi/acstruct.h
881 usr/src/uts/intel/sys/acpi/actables.h
882 usr/src/uts/intel/sys/acpi/actbl.h
883 usr/src/uts/intel/sys/acpi/actbl1.h
884 usr/src/uts/intel/sys/acpi/actbl2.h
885 usr/src/uts/intel/sys/acpi/actypes.h
886 usr/src/uts/intel/sys/acpi/acutils.h
887 usr/src/uts/intel/sys/acpi/amlcode.h
888 usr/src/uts/intel/sys/acpi/amlresrc.h
889 usr/src/uts/intel/sys/acpi/platform/accygwin.h
890 usr/src/uts/intel/sys/acpi/platform/acefi.h
891 usr/src/uts/intel/sys/acpi/platform/acenv.h
892 usr/src/uts/intel/sys/acpi/platform/acfreebsd.h
893 usr/src/uts/intel/sys/acpi/platform/acgcc.h
894 usr/src/uts/intel/sys/acpi/platform/acintel.h
895 usr/src/uts/intel/sys/acpi/platform/aclinux.h
896 usr/src/uts/intel/sys/acpi/platform/acmsvc.h
897 usr/src/uts/intel/sys/acpi/platform/acnetbsd.h
898 usr/src/uts/intel/sys/acpi/platform/acos2.h
899 usr/src/uts/intel/sys/acpi/platform/acsolaris.h
900 usr/src/uts/intel/sys/acpi/platform/acwin.h
901 usr/src/uts/intel/sys/acpi/platform/acwin64.h
```

## new/exception\_lists/hdrchk

1

```
*****
8480 Fri Jan 17 07:29:13 2014
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
```

## new/exception\_lists/hdrchk

2

```
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/dns glue.h
66 usr/src/lib/gss_mechs/mech_krb5/krb5/os/os-protocol.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/chpasswd_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/utls.h
122 usr/src/lib/librtp/common/base.h
123 usr/src/lib/librtp/common/choose.h
124 usr/src/lib/librtp/common/edge.h
125 usr/src/lib/librtp/common/migrate.h
126 usr/src/lib/librtp/common/p2p.h
127 usr/src/lib/librtp/common/pcost.h
```

```
128 usr/src/lib/librstp/common/port.h
129 usr/src/lib/librstp/common/portinfo.h
130 usr/src/lib/librstp/common/rolesel.h
131 usr/src/lib/librstp/common/roletrns.h
132 usr/src/lib/librstp/common/statmch.h
133 usr/src/lib/librstp/common/stp_bpdu.h
134 usr/src/lib/librstp/common/stp_in.h
135 usr/src/lib/librstp/common/stp_to.h
136 usr/src/lib/librstp/common/stp_vectors.h
137 usr/src/lib/librstp/common/stpm.h
138 usr/src/lib/librstp/common/sttrans.h
139 usr/src/lib/librstp/common/times.h
140 usr/src/lib/librstp/common/topoch.h
141 usr/src/lib/librstp/common/transmit.h
142 usr/src/lib/librstp/common/uid_stp.h
143 usr/src/lib/librstp/common/vector.h
144 usr/src/uts/common/gssapi/mechs/krb5/include/aes_s2k.h
145 usr/src/uts/common/gssapi/mechs/krb5/include/auth_con.h
146 usr/src/uts/common/gssapi/mechs/krb5/include/cksumtypes.h
147 usr/src/uts/common/gssapi/mechs/krb5/include/crc-32.h
148 usr/src/uts/common/gssapi/mechs/krb5/include/des_int.h
149 usr/src/uts/common/gssapi/mechs/krb5/include/dk.h
150 usr/src/uts/common/gssapi/mechs/krb5/include/enc_provider.h
151 usr/src/uts/common/gssapi/mechs/krb5/include/etypes.h
152 usr/src/uts/common/gssapi/mechs/krb5/include/gssapi_generic.h
153 usr/src/uts/common/gssapi/mechs/krb5/include/gssapi_krb5.h
154 usr/src/uts/common/gssapi/mechs/krb5/include/gssapiP_generic.h
155 usr/src/uts/common/gssapi/mechs/krb5/include/gssapiP_krb5.h
156 usr/src/uts/common/gssapi/mechs/krb5/include/hash_provider.h
157 usr/src/uts/common/gssapi/mechs/krb5/include/k5-int.h
158 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_16.h
159 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_32.h
160 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-load_64.h
161 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_16.h
162 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_32.h
163 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform-store_64.h
164 usr/src/uts/common/gssapi/mechs/krb5/include/k5-platform.h
165 usr/src/uts/common/gssapi/mechs/krb5/include/k5-thread.h
166 usr/src/uts/common/gssapi/mechs/krb5/include/keyhash_provider.h
167 usr/src/uts/common/gssapi/mechs/krb5/include/krb5.h
168 usr/src/uts/common/gssapi/mechs/krb5/include/old.h
169 usr/src/uts/common/gssapi/mechs/krb5/include/raw.h
170 usr/src/uts/common/gssapi/mechs/krb5/include/rsa-md4.h
171 usr/src/uts/common/io/ixgbe/ixgbe_common.h
172 usr/src/common/acpica
172 usr/src/uts/intel/sys/acpi/acdebug.h
173 usr/src/uts/intel/sys/acpi/acdisasm.h
174 usr/src/uts/intel/sys/acpi/acevents.h
175 usr/src/uts/intel/sys/acpi/acinterp.h
176 usr/src/uts/intel/sys/acpi/acmacros.h
177 usr/src/uts/intel/sys/acpi/acnames.h
178 usr/src/uts/intel/sys/acpi/acpredef.h
179 usr/src/uts/intel/sys/acpi/acresrc.h
180 usr/src/uts/intel/sys/acpi/acstruct.h
181 usr/src/uts/intel/sys/acpi/amlresrc.h
182 usr/src/uts/intel/sys/acpi/platform/acwin64.h
```

new/usr/src/Makefile.master

1

\*\*\*\*\*

34676 Fri Jan 17 07:29:13 2014

new/usr/src/Makefile.master

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 #
23 # Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright (c) 2012 by Delphix. All rights reserved.
25 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
26 #
27 #
28 #
29 # Makefile.master, global definitions for system source
30 #
31 ROOT=          /proto
32 #
33 #
34 # Adjunct root, containing an additional proto area to be used for headers
35 # and libraries.
36 #
37 ADJUNCT_PROTO=
38 #
39 #
40 # Adjunct for building things that run on the build machine.
41 #
42 NATIVE_ADJUNCT= /usr
43 #
44 #
45 # RELEASE_BUILD should be cleared for final release builds.
46 # NOT_RELEASE_BUILD is exactly what the name implies.
47 #
48 # __GNUC toggles the building of ON components using gcc and related tools.
49 # Normally set to '#', set it to '' to do gcc build.
50 #
51 # The declaration POUND_SIGN is always '#'. This is needed to get around the
52 # make feature that '#' is always a comment delimiter, even when escaped or
53 # quoted. We use this macro expansion method to get POUND_SIGN rather than
54 # always breaking out a shell because the general case can cause a noticable
55 # slowdown in build times when so many Makefiles include Makefile.master.
56 #
57 # While the majority of users are expected to override the setting below
58 # with an env file (via nightly or bldenv), if you aren't building that way
59 # (ie, you're using "ws" or some other bootstrapping method) then you need
60 # this definition in order to avoid the subshell invocation mentioned above.
61 #
```

new/usr/src/Makefile.master

2

```
63 PRE_POUND=
64 POUND_SIGN=    pre\#
                  $(PRE_POUND:pre\%=%)

66 NOT_RELEASE_BUILD=
67 RELEASE_BUILD= $(POUND_SIGN)
68 $(RELEASE_BUILD)NOT_RELEASE_BUILD= $(POUND_SIGN)
69 PATCH_BUILD=   $(POUND_SIGN)

71 # SPARC_BLD is '#' for an Intel build.
72 # INTEL_BLD is '#' for a Sparc build.
73 SPARC_BLD_1=    $(MACH:i386=$(POUND_SIGN))
74 SPARC_BLD=      $(SPARC_BLD_1:sparc=)
75 INTEL_BLD_1=    $(MACH:sparc=$(POUND_SIGN))
76 INTEL_BLD=      $(INTEL_BLD_1:i386=)

78 # The variables below control the compilers used during the build.
79 # There are a number of permutations.
80 #
81 # __GNUC and __SUNC control (and indicate) the primary compiler.  Whichever
82 # one is not POUND_SIGN is the primary, with the other as the shadow.  They
83 # may also be used to control entirely compiler-specific Makefile assignments.
84 # __SUNC and Sun Studio are the default.
85 #
86 # __GNUC64 indicates that the 64bit build should use the GNU C compiler.
87 # There is no Sun C analogue.
88 #
89 # The following version-specific options are operative regardless of which
90 # compiler is primary, and control the versions of the given compilers to be
91 # used.  They also allow compiler-version specific Makefile fragments.
92 #

94 __GNUC=         $(POUND_SIGN)
95 $(__GNUC)__SUNC= $(POUND_SIGN)
96 __GNUC64=       $(__GNUC)

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

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

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

108 JAVA_ROOT=      /usr/java

110 SFW_ROOT=       /usr/sfw
111 SFWINCDIR=      $(SFW_ROOT)/include
112 SFWLIBDIR=      $(SFW_ROOT)/lib
113 SFWLIBDIR64=    $(SFW_ROOT)/lib/$(MACH64)

115 GCC_ROOT=       /opt/gcc/4.4.4
116 GCCLIBDIR=      $(GCC_ROOT)/lib
117 GCCLIBDIR64=    $(GCC_ROOT)/lib/$(MACH64)

119 DOCBOK_XSL_ROOT= /usr/share/sgml/docbook/xsl-stylesheets

121 RPCGEN=         /usr/bin/rpcgen
122 STABS=          $(ONBLD_TOOLS)/bin/$(MACH)/stabs
123 ELFEXTRACT=     $(ONBLD_TOOLS)/bin/$(MACH)/elfextract
124 MBH_PATCH=      $(ONBLD_TOOLS)/bin/$(MACH)/mbh_patch
125 ECHO=           echo
126 INS=            install
127 TRUE=          true
```

## new/usr/src/Makefile.master

3

```

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

183 FILEMODE=    644
184 DIRMODE=     755

186 #
187 # The version of the patch makeup table optimized for build-time use.  Used
188 # during patch builds only.
189 $(PATCH_BUILD)PMTMO_FILE=$(SRC)/patch_makeup_table.mo

191 # Declare that nothing should be built in parallel.
192 # Individual Makefiles can use the .PARALLEL target to declare otherwise.

```

## new/usr/src/Makefile.master

4

```

193 .NO_PARALLEL:

195 # For stylistic checks
196 #
197 # Note that the X and C checks are not used at this time and may need
198 # modification when they are actually used.
199 #
200 CSTYLE=       $(ONBLD_TOOLS)/bin/cstyle
201 CSTYLE_TAIL=  $(ONBLD_TOOLS)/bin/hdrchk
202 HDRCHK=       $(ONBLD_TOOLS)/bin/hdrchk
203 HDRCHK_TAIL=  $(ONBLD_TOOLS)/bin/jstyle
204 JSTYLE=       $(ONBLD_TOOLS)/bin/jstyle

206 DOT_H_CHECK=  \
207     @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL); \
208     $(HDRCHK) $< $(HDRCHK_TAIL)

210 DOT_X_CHECK=  \
211     @$(ECHO) "checking $<"; $(RPCGEN) -C -h $< | $(CSTYLE) $(CSTYLE_TAIL); \
212     $(RPCGEN) -C -h $< | $(HDRCHK) $< $(HDRCHK_TAIL)

214 DOT_C_CHECK=  \
215     @$(ECHO) "checking $<"; $(CSTYLE) $< $(CSTYLE_TAIL)

217 MANIFEST_CHECK= \
218     @$(ECHO) "checking $<"; \
219     SVCCFG_DTD=$(SRC)/cmd/svc/dtd/service_bundle.dtd.1 \
220     SVCCFG_REPOSITORY=$(SRC)/cmd/svc/seed/global.db \
221     SVCCFG_CONFIGD_PATH=$(SRC)/cmd/svc/configd/svc.configd-native \
222     $(SRC)/cmd/svc/svccfg/svccfg-native validate $<

224 INS.file=     $(RM) $@; $(INS) -s -m $(FILEMODE) -f $(@D) $<
225 INS.dir=      $(INS) -s -d -m $(DIRMODE) $@
226 # installs and renames at once
227 #
228 INS.rename=   $(INS.file); $(MV) $(@D)/$(<F) $@

230 # install a link
231 INSLINKTARGET= $<
232 INS.link=     $(RM) $@; $(LN) $(INSLINKTARGET) $@
233 INS.symlink=  $(RM) $@; $(SYMLINK) $(INSLINKTARGET) $@

235 #
236 # Python bakes the mtime of the .py file into the compiled .pyc and
237 # rebuilds if the baked-in mtime != the mtime of the source file
238 # (rather than only if it's less than), thus when installing python
239 # files we must make certain to not adjust the mtime of the source
240 # (.py) file.
241 #
242 INS.pyfile=   $(INS.file); $(TOUCH) -r $< $@

244 # MACH must be set in the shell environment per uname -p on the build host
245 # More specific architecture variables should be set in lower makefiles.
246 #
247 # MACH64 is derived from MACH, and BUILD64 is set to '#' for
248 # architectures on which we do not build 64-bit versions.
249 # (There are no such architectures at the moment.)
250 #
251 # Set BUILD64=# in the environment to disable 64-bit amd64
252 # builds on i386 machines.

254 MACH64_1=     $(MACH:sparc=sparcv9)
255 MACH64=       $(MACH64_1:i386=amd64)

257 MACH32_1=     $(MACH:sparc=sparcv7)
258 MACH32=       $(MACH32_1:i386=i86)

```

```

260 sparc_BUILD64=
261 i386_BUILD64=
262 BUILD64=      ${$(MACH)_BUILD64}

264 #
265 # C compiler mode. Future compilers may change the default on us,
266 # so force extended ANSI mode globally. Lower level makefiles can
267 # override this by setting CCMODE.
268 #
269 CCMODE=        -Xa
270 CCMODE64=      -Xa

272 #
273 # C compiler verbose mode. This is so we can enable it globally,
274 # but turn it off in the lower level makefiles of things we cannot
275 # (or aren't going to) fix.
276 #
277 CCVERBOSE=     -v

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

283 # set this to the secret flag "-Wc,-Qiselect-regsym=0" to disable register
284 # symbols (used to detect conflicts between objects that use global registers)
285 # we disable this now for safety, and because genunix doesn't link with
286 # this feature (the v9 default) enabled.
287 #
288 # REGSYM is separate since the C++ driver syntax is different.
289 CCREGSYM=      -Wc,-Qiselect-regsym=0
290 CCCREGSYM=     -Qoption cg -Qiselect-regsym=0

292 # Prevent the removal of static symbols by the SPARC code generator (cg).
293 # The x86 code generator (ube) does not remove such symbols and as such
294 # using this workaround is not applicable for x86.
295 #
296 CCSTATICSYM=   -Wc,-Qassembler-ounrefsym=0
297 #
298 # generate 32-bit addresses in the v9 kernel. Saves memory.
299 CCABS32=       -Wc,-xcode=abs32
300 #
301 # generate v9 code which tolerates callers using the v7 ABI, for the sake of
302 # system calls.
303 CC32BITCALLERS=  _gcc=-masume-32bit-callers

305 # GCC, especially, is increasingly beginning to auto-inline functions and
306 # sadly does so separately not under the general -fno-inline-functions
307 # Additionally, we wish to prevent optimisations which cause GCC to clone
308 # functions -- in particular, these may cause unhelpful symbols to be
309 # emitted instead of function names
310 CCNOAUTOINLINE= _gcc=-fno-inline-small-functions \
311                _gcc=-fno-inline-functions-called-once \
312                _gcc=-fno-ipa-cp

314 # One optimization the compiler might perform is to turn this:
315 #   #pragma weak foo
316 #   extern int foo;
317 #   if (&foo)
318 #       foo = 5;
319 # into
320 #   foo = 5;
321 # Since we do some of this (foo might be referenced in common kernel code
322 # but provided only for some cpu modules or platforms), we disable this
323 # optimization.
324 #

```

```

325 sparc_CCUNBOUND = -Wd,-xsafe=unboundsym
326 i386_CCUNBOUND  =
327 CCUNBOUND       = ${$(MACH)_CCUNBOUND}

329 #
330 # compiler '-xarch' flag. This is here to centralize it and make it
331 # overridable for testing.
332 sparc_XARCH=    -m32
333 sparcv9_XARCH=  -m64
334 i386_XARCH=
335 amd64_XARCH=    -m64 -Ui386 -U__i386

337 # assembler '-xarch' flag. Different from compiler '-xarch' flag.
338 sparc_AS_XARCH=  -xarch=v8plus
339 sparcv9_AS_XARCH= -xarch=v9
340 i386_AS_XARCH=
341 amd64_AS_XARCH=  -xarch=amd64 -P -Ui386 -U__i386

343 #
344 # These flags define what we need to be 'standalone' i.e. -not- part
345 # of the rather more cosy userland environment. This basically means
346 # the kernel.
347 #
348 # XX64 future versions of gcc will make -mmodel=kernel imply -mno-red-zone
349 #
350 sparc_STAND_FLAGS=  _gcc=-ffreestanding
351 sparcv9_STAND_FLAGS= _gcc=-ffreestanding
352 # Disabling MMX also disables 3DNow, disabling SSE also disables all later
353 # additions to SSE (SSE2, AVX, etc.)
354 NO_SIMD=           _gcc=-mno-mmx _gcc=-mno-sse
355 i386_STAND_FLAGS=   _gcc=-ffreestanding $(NO_SIMD)
356 amd64_STAND_FLAGS=  -xmodel=kernel $(NO_SIMD)

358 SAVEARGS=          -Wu,-save_args
359 amd64_STAND_FLAGS  += $(SAVEARGS)

361 STAND_FLAGS_32 = ${$(MACH)_STAND_FLAGS}
362 STAND_FLAGS_64 = ${$(MACH64)_STAND_FLAGS}

364 #
365 # disable the incremental linker
366 ILDOFF=            -xildoff
367 #
368 XDEPEND=           -xdepend
369 XFFLAG=            -xF=%all
370 XESS=              -xs
371 XSTRCONST=         -xstrconst

373 #
374 # turn warnings into errors (C)
375 CERRWARN = -errtags=yes -errwarn=%all
376 CERRWARN += -erroff=E_EMPTY_TRANSLATION_UNIT
377 CERRWARN += -erroff=E_STATEMENT_NOT_REACHED

379 CERRWARN += _gcc=-Wno-missing-braces
380 CERRWARN += _gcc=-Wno-sign-compare
381 CERRWARN += _gcc=-Wno-unknown-pragmas
382 CERRWARN += _gcc=-Wno-unused-parameter
383 CERRWARN += _gcc=-Wno-missing-field-initializers

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

388 # DEBUG v. -nd make for frequent unused variables, empty conditions, etc. in
389 # -nd builds
390 $(RELEASE_BUILD)CERRWARN += _gcc=-Wno-unused

```

```

391 $(RELEASE_BUILD)CERRWARN += -_gcc=-Wno-empty-body

393 #
394 # turn warnings into errors (C++)
395 CCERRWARN=          -xwe

397 # C99 mode
398 C99_ENABLE=         -xc99=%all
399 C99_DISABLE=        -xc99=%none
400 C99MODE=             $(C99_DISABLE)
401 C99LMODE=           $(C99MODE:-xc99%=-Xc99%)

403 # In most places, assignments to these macros should be appended with +=
404 # (CPPFLAGS.master allows values to be prepended to CPPFLAGS).
405 sparc_CFLAGS=       $(sparc_XARCH) $(CSTATICSYM)
406 sparcv9_CFLAGS=     $(sparcv9_XARCH) -dalign $(CCVERBOSE) $(V9ABIWARN) $(CCREGSYM) \
407                     $(CSTATICSYM)
408 i386_CFLAGS=        $(i386_XARCH)
409 amd64_CFLAGS=       $(amd64_XARCH)

411 sparc_ASFLAGS=      $(sparc_AS_XARCH)
412 sparcv9_ASFLAGS=    $(sparcv9_AS_XARCH)
413 i386_ASFLAGS=       $(i386_AS_XARCH)
414 amd64_ASFLAGS=      $(amd64_AS_XARCH)

416 #
417 sparc_COPTFLAG=     -xO3
418 sparcv9_COPTFLAG=  -xO3
419 i386_COPTFLAG=      -O
420 amd64_COPTFLAG=     -xO3

422 COPTFLAG= $(($(MACH)_COPTFLAG))
423 COPTFLAG64= $(($(MACH64)_COPTFLAG))

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

429 # Direct the Sun Studio compiler to use a static globalization prefix based on t
430 # name of the module rather than something unique. Otherwise, objects
431 # will not build deterministically, as subsequent compilations of identical
432 # source will yield objects that always look different.
433 #
434 # In the same spirit, this will also remove the date from the N_OPT stab.
435 CGLOBALSTATIC= -W0,-xglobalstatic

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

441 #
442 # Default debug format for Sun Studio 11 is dwarf, so force it to
443 # generate stabs.
444 #
445 DEBUGFORMAT= -xdebugformat=stabs

447 #
448 # Flags used to build in debug mode for ctf generation.  Bugs in the Devpro
449 # compilers currently prevent us from building with cc-emitted DWARF.
450 #
451 CTF_FLAGS_sparc = -g -Wc,-Qiselect-T1 $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)
452 CTF_FLAGS_i386 = -g $(C99MODE) $(CNOGLOBAL) $(CDWARFSTR)

454 CTF_FLAGS_sparcv9 = $(CTF_FLAGS_sparc)
455 CTF_FLAGS_amd64 = $(CTF_FLAGS_i386)

```

```

457 # Sun Studio produces broken userland code when saving arguments.
458 $(__GNUC)CTF_FLAGS_amd64 += $(SAVEARGS)

460 CTF_FLAGS_32 = $(CTF_FLAGS_$(MACH)) $(DEBUGFORMAT)
461 CTF_FLAGS_64 = $(CTF_FLAGS_$(MACH64)) $(DEBUGFORMAT)
462 CTF_FLAGS = $(CTF_FLAGS_32)

464 #
465 # Flags used with genoffsets
466 #
467 GOFLAGS = -_noecho \
468           $(CALLSYMS) \
469           $(CDWARFSTR)

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

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

477 #
478 # tradeoff time for space (smaller is better)
479 #
480 sparc_SPACEFLAG = -xspace -W0,-Lt
481 sparcv9_SPACEFLAG = -xspace -W0,-Lt
482 i386_SPACEFLAG = -xspace
483 amd64_SPACEFLAG =

485 SPACEFLAG = $(($(MACH)_SPACEFLAG))
486 SPACEFLAG64 = $(($(MACH64)_SPACEFLAG))

488 #
489 # The Sun Studio 11 compiler has changed the behaviour of integer
490 # wrap arounds and so a flag is needed to use the legacy behaviour
491 # (without this flag panics/hangs could be exposed within the source).
492 #
493 sparc_IROPTFLAG = -W2,-xwrap_int
494 sparcv9_IROPTFLAG = -W2,-xwrap_int
495 i386_IROPTFLAG =
496 amd64_IROPTFLAG =

498 IROPTFLAG = $(($(MACH)_IROPTFLAG))
499 IROPTFLAG64 = $(($(MACH64)_IROPTFLAG))

501 sparc_XREGSFLAG = -xregs=no%appl
502 sparcv9_XREGSFLAG = -xregs=no%appl
503 i386_XREGSFLAG =
504 amd64_XREGSFLAG =

506 XREGSFLAG = $(($(MACH)_XREGSFLAG))
507 XREGSFLAG64 = $(($(MACH64)_XREGSFLAG))

509 # dmake SOURCEDEBUG=yes ... enables source-level debugging information, and
510 # avoids stripping it.
511 SOURCEDEBUG = $(POUND_SIGN)
512 SRCDBGBLD = $(SOURCEDEBUG:yes=)

514 #
515 # These variables are intended ONLY for use by developers to safely pass extra
516 # flags to the compilers without unintentionally overriding Makefile-set
517 # flags.  They should NEVER be set to any value in a Makefile.
518 #
519 # They come last in the associated FLAGS variable such that they can
520 # explicitly override things if necessary, there are gaps in this, but it's
521 # the best we can manage.
522 #

```



```

523 CUSERFLAGS           =
524 CUSERFLAGS64          = $(CUSERFLAGS)
525 CCUSERFLAGS           =
526 CCUSERFLAGS64         = $(CCUSERFLAGS)

528 CSOURCEDEBUGFLAGS     =
529 CCSOURCEDEBUGFLAGS     =
530 $(SRCDGBLD)CSOURCEDEBUGFLAGS = -g -xs
531 $(SRCDGBLD)CCSOURCEDEBUGFLAGS = -g -xs

533 CFLAGS=                $(COPTFLAG) $($ (MACH)_CFLAGS) $(SPACEFLAG) $(CCMODE) \
534                        $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG) \
535                        $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
536                        $(CUSERFLAGS)
537 CFLAGS64=              $(COPTFLAG64) $($ (MACH64)_CFLAGS) $(SPACEFLAG64) $(CCMODE64) \
538                        $(ILDOFF) $(CERRWARN) $(C99MODE) $(CCUNBOUND) $(IROPTFLAG64) \
539                        $(CGLOBALSTATIC) $(CCNOAUTOINLINE) $(CSOURCEDEBUGFLAGS) \
540                        $(CUSERFLAGS64)
541 #
542 # Flags that are used to build parts of the code that are subsequently
543 # run on the build machine (also known as the NATIVE_BUILD).
544 #
545 NATIVE_CFLAGS=          $(COPTFLAG) $($ (NATIVE_MACH)_CFLAGS) $(CCMODE) \
546                        $(ILDOFF) $(CERRWARN) $(C99MODE) $($ (NATIVE_MACH)_CCUNBOUND) \
547                        $(IROPTFLAG) $(CGLOBALSTATIC) $(CCNOAUTOINLINE) \
548                        $(CSOURCEDEBUGFLAGS) $(CUSERFLAGS)

550 DTEXTDOM=-DTEXT_DOMAIN="\$(TEXT_DOMAIN)"      # For messaging.
551 DTS_ERRNO=-D_TS_ERRNO
552 CPPFLAGS.master=$(DTEXTDOM) $(DTS_ERRNO) \
553                $(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) $(ENVCPPFLAGS4) \
554                $(ADJUNCT_PROTO:%=-I%/usr/include)
555 CPPFLAGS.native=$(ENVCPPFLAGS1) $(ENVCPPFLAGS2) $(ENVCPPFLAGS3) \
556                $(ENVCPPFLAGS4) -I$(NATIVE_ADJUNCT)/include
557 CPPFLAGS=            $(CPPFLAGS.master)
558 AS_CPPFLAGS=          $(CPPFLAGS.master)
559 JAVAFLAGS=            -deprecation

561 #
562 # For source message catalogue
563 #
564 .SUFFIXES: $(SUFFIXES) .i .po
565 MSGROOT= $(ROOT)/catalog
566 MSGDOMAIN= $(MSGROOT)/$(TEXT_DOMAIN)
567 MSGDOMAINPOFILE = $(MSGDOMAIN)/$(POFILE)
568 DCMMSGDOMAIN= $(MSGROOT)/LC_TIME/$(TEXT_DOMAIN)
569 DCMMSGDOMAINPOFILE = $(DCMSGDOMAIN)/$(DCFILE:.dc=.po)

571 CLOBBERFILES += $(POFILE) $(POFILES)
572 COMPILER.cpp= $(CC) -E -C $(CFLAGS) $(CPPFLAGS)
573 XGETTEXT= /usr/bin/xgettext
574 XGETFLAGS= -c TRANSLATION_NOTE
575 GNUXGETTEXT= /usr/gnu/bin/xgettext
576 GNUXGETFLAGS= --add-comments=TRANSLATION_NOTE --keyword=_ \
577               --strict --no-location --omit-header
578 BUILD.po= $(XGETTEXT) $(XGETFLAGS) -d $(<F) $(<i) ;\
579            $(RM)    $@ ;\
580            $(SED) "/^domain/d" < $(<F).po > $@ ;\
581            $(RM)    $(<F).po $(<i)

583 #
584 # This is overwritten by local Makefile when PROG is a list.
585 #
586 POFILE= $(PROG).po

588 sparc_CCFLAGS=         -cg92 -compat=4 \

```

```

589 -Qoption ccfe -messages=no%anachronism \
590 $(CCERRWARN)
591 sparcv9_CCFLAGS=       $(sparcv9_XARCH) -dalign -compat=5 \
592 -Qoption ccfe -messages=no%anachronism \
593 -Qoption ccfe -features=no%conststrings \
594 $(CCREGSYM) \
595 $(CCERRWARN)
596 i386_CCFLAGS=          -compat=4 \
597 -Qoption ccfe -messages=no%anachronism \
598 -Qoption ccfe -features=no%conststrings \
599 $(CCERRWARN)
600 amd64_CCFLAGS=         $(amd64_XARCH) -compat=5 \
601 -Qoption ccfe -messages=no%anachronism \
602 -Qoption ccfe -features=no%conststrings \
603 $(CCERRWARN)

605 sparc_CCOPTFLAG=      -O
606 sparcv9_CCOPTFLAG=     -O
607 i386_CCOPTFLAG=       -O
608 amd64_CCOPTFLAG=      -O

610 CCOPTFLAG=             $($ (MACH)_CCOPTFLAG)
611 CCOPTFLAG64=           $($ (MACH64)_CCOPTFLAG)
612 CCFLAGS=               $(CCOPTFLAG) $($ (MACH)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
613                         $(CUSERFLAGS)
614 CCFLAGS64=             $(CCOPTFLAG64) $($ (MACH64)_CCFLAGS) $(CCSOURCEDEBUGFLAGS) \
615                         $(CUSERFLAGS64)

617 #
618 #
619 #
620 ELFWRAP_FLAGS =
621 ELFWRAP_FLAGS64 =      -64

623 #
624 # Various mapfiles that are used throughout the build, and delivered to
625 # /usr/lib/ld.
626 #
627 MAPFILE.NED_i386 =     $(SRC)/common/mapfiles/common/map.noexdata
628 MAPFILE.NED_sparc =
629 MAPFILE.NED =          $(MAPFILE.NED_$(MACH))
630 MAPFILE.PGA =          $(SRC)/common/mapfiles/common/map.pagealign
631 MAPFILE.NES =          $(SRC)/common/mapfiles/common/map.noexstk
632 MAPFILE.FLT =          $(SRC)/common/mapfiles/common/map.filter
633 MAPFILE.LEX =          $(SRC)/common/mapfiles/common/map.lex.yy

635 #
636 # Generated mapfiles that are compiler specific, and used throughout the
637 # build. These mapfiles are not delivered in /usr/lib/ld.
638 #
639 MAPFILE.NGB_sparc=     $(SRC)/common/mapfiles/gen/sparc_cc_map.noexeglobs
640 $(__GNU64)MAPFILE.NGB_sparc= \
641 $(SRC)/common/mapfiles/gen/sparc_gcc_map.noexeglobs
642 MAPFILE.NGB_sparcv9=   $(SRC)/common/mapfiles/gen/sparcv9_cc_map.noexeglobs
643 $(__GNU64)MAPFILE.NGB_sparcv9= \
644 $(SRC)/common/mapfiles/gen/sparcv9_gcc_map.noexeglobs
645 MAPFILE.NGB_i386=      $(SRC)/common/mapfiles/gen/i386_cc_map.noexeglobs
646 $(__GNU64)MAPFILE.NGB_i386= \
647 $(SRC)/common/mapfiles/gen/i386_gcc_map.noexeglobs
648 MAPFILE.NGB_amd64=     $(SRC)/common/mapfiles/gen/amd64_cc_map.noexeglobs
649 $(__GNU64)MAPFILE.NGB_amd64= \
650 $(SRC)/common/mapfiles/gen/amd64_gcc_map.noexeglobs
651 MAPFILE.NGB =          $(MAPFILE.NGB_$(MACH))

653 #
654 # A generic interface mapfile name, used by various dynamic objects to define

```

```

655 # the interfaces and interposers the object must export.
656 #
657 MAPFILE.INT =          mapfile-intf

659 #
660 # LDLIBS32 can be set in the environment to override the following assignment.
661 # LDLIBS64 can be set to override the assignment made in Makefile.master.64.
662 # These environment settings make sure that no libraries are searched outside
663 # of the local workspace proto area:
664 #     LDLIBS32=-YP,$ROOT/lib:$ROOT/usr/lib
665 #     LDLIBS64=-YP,$ROOT/lib/$MACH64:$ROOT/usr/lib/$MACH64
666 #
667 LDLIBS32 =      $(ENVLDLIBS1) $(ENVLDLIBS2) $(ENVLDLIBS3)
668 LDLIBS32 +=     $(ADJUNCT_PROTO:%=-L%/usr/lib -L%/lib)
669 LDLIBS.cmd =    $(LDLIBS32)
670 LDLIBS.lib =    $(LDLIBS32)
671 #
672 # Define compilation macros.
673 #
674 COMPILE.c=      $(CC) $(CFLAGS) $(CPPFLAGS) -c
675 COMPILE64.c=    $(CC) $(CFLAGS64) $(CPPFLAGS) -c
676 COMPILE.cc=     $(CCC) $(CCFLAGS) $(CPPFLAGS) -c
677 COMPILE64.cc=   $(CCC) $(CCFLAGS64) $(CPPFLAGS) -c
678 COMPILE.s=      $(AS) $(ASFLAGS) $(AS_CPPFLAGS)
679 COMPILE64.s=    $(AS) $(ASFLAGS) $(MACH64)_AS_XARCH) $(AS_CPPFLAGS)
680 COMPILE.d=      $(DTRACE) -G -32
681 COMPILE64.d=    $(DTRACE) -G -64
682 COMPILE.b=      $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))
683 COMPILE64.b=    $(ELFWRAP) $(ELFWRAP_FLAGS$(CLASS))

685 CLASSPATH=      .
686 COMPILE.java=   $(JAVAC) $(JAVAFLAGS) -classpath $(CLASSPATH)

688 #
689 # Link time macros
690 #
691 CCNEEDED        = -lC
692 CCEXTNEEDED     = -lCrun -lCstd
693 $(__GNUC)CCNEEDED = -L$(GCCLIBDIR) -lstl++ -lgcc_s
694 $(__GNUC)CCEXTNEEDED = $(CCNEEDED)

696 LINK.c=         $(CC) $(CFLAGS) $(CPPFLAGS) $(LDFLAGS)
697 LINK64.c=       $(CC) $(CFLAGS64) $(CPPFLAGS) $(LDFLAGS)
698 NORUNPATH=      -norunpath -nolib
699 LINK.cc=        $(CCC) $(CCFLAGS) $(CPPFLAGS) $(NORUNPATH) \
700                $(LDFLAGS) $(CCNEEDED)
701 LINK64.cc=      $(CCC) $(CCFLAGS64) $(CPPFLAGS) $(NORUNPATH) \
702                $(LDFLAGS) $(CCNEEDED)

704 #
705 # lint macros
706 #
707 # Note that the undefine of __PRAGMA_REDEFINE_EXTNAME can be removed once
708 # ON is built with a version of lint that has the fix for 4484186.
709 #
710 ALWAYS_LINT_DEFS = -errtags=yes -s
711 ALWAYS_LINT_DEFS += -erroff=E_PTRDIFF_OVERFLOW
712 ALWAYS_LINT_DEFS += -erroff=E_ASSIGN_NARROW_CONV
713 ALWAYS_LINT_DEFS += -U__PRAGMA_REDEFINE_EXTNAME
714 ALWAYS_LINT_DEFS += $(C99LMODE)
715 ALWAYS_LINT_DEFS += -errsecurity=$(SECLEVEL)
716 ALWAYS_LINT_DEFS += -erroff=E_SEC_CREAT_WITHOUT_EXCL
717 ALWAYS_LINT_DEFS += -erroff=E_SEC_FORBIDDEN_WARN_CREAT
718 # XX64 -- really only needed for amd64 lint
719 ALWAYS_LINT_DEFS += -erroff=E_ASSIGN_INT_TO_SMALL_INT
720 ALWAYS_LINT_DEFS += -erroff=E_CAST_INT_CONST_TO_SMALL_INT

```

```

721 ALWAYS_LINT_DEFS += -erroff=E_CAST_INT_TO_SMALL_INT
722 ALWAYS_LINT_DEFS += -erroff=E_CAST_TO_PTR_FROM_INT
723 ALWAYS_LINT_DEFS += -erroff=E_COMP_INT_WITH_LARGE_INT
724 ALWAYS_LINT_DEFS += -erroff=E_INTEGRAL_CONST_EXP_EXPECTED
725 ALWAYS_LINT_DEFS += -erroff=E_PASS_INT_TO_SMALL_INT
726 ALWAYS_LINT_DEFS += -erroff=E_PTR_CONV_LOSES_BITS

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

732 SECLEVEL=      core
733 LINT.c=         $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS) $(CPPFLAGS) \
734                $(ALWAYS_LINT_DEFS)
735 LINT64.c=       $(LINT) $(ONLY_LINT_DEFS) $(LINTFLAGS64) $(CPPFLAGS) \
736                $(ALWAYS_LINT_DEFS)
737 LINT.s=         $(LINT.c)

739 # For some future builds, NATIVE_MACH and MACH might be different.
740 # Therefore, NATIVE_MACH needs to be redefined in the
741 # environment as 'uname -p' to override this macro.
742 #
743 # For now at least, we cross-compile amd64 on i386 machines.
744 NATIVE_MACH=    $(MACH:amd64=i386)

746 # Define native compilation macros
747 #

749 # Base directory where compilers are loaded.
750 # Defined here so it can be overridden by developer.
751 #
752 SPRO_ROOT=      $(BUILD_TOOLS)/SUNWspro
753 SPRO_VROOT=     $(SPRO_ROOT)/SS12
754 GNU_ROOT=       $(SFW_ROOT)

756 # Till SS12u1 formally becomes the NV CBE, LINT is hard
757 # coded to be picked up from the $SPRO_ROOT/sunstudio12.1/
758 # location. Impacted variables are sparc_LINT, sparcv9_LINT,
759 # i386_LINT, amd64_LINT.
760 # Reset them when SS12u1 is rolled out.
761 #

763 # Specify platform compiler versions for languages
764 # that we use (currently only c and c++).
765 #
766 sparc_CC=       $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
767 $(__GNUC)sparc_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
768 sparc_CCC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
769 $(__GNUC)sparc_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
770 sparc_CPP=      /usr/ccs/lib/cpp
771 sparc_AS=       /usr/ccs/bin/as -xregsym=no
772 sparc_LD=       /usr/ccs/bin/ld
773 sparc_LINT=     $(SPRO_ROOT)/sunstudio12.1/bin/lint

775 sparcv9_CC=     $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
776 $(__GNUC64)sparcv9_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
777 sparcv9_CCC=    $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
778 $(__GNUC64)sparcv9_CCC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
779 sparcv9_CPP=    /usr/ccs/lib/cpp
780 sparcv9_AS=     /usr/ccs/bin/as -xregsym=no
781 sparcv9_LD=     /usr/ccs/bin/ld
782 sparcv9_LINT=   $(SPRO_ROOT)/sunstudio12.1/bin/lint

784 i386_CC=        $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
785 $(__GNUC)i386_CC= $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
786 i386_CCC=       $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC

```

```

787 $(__GNUC)i386_CCC=      $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
788 i386_CPP=                /usr/ccs/lib/cpp
789 i386_AS=                  /usr/ccs/bin/as
790 $(__GNUC)i386_AS=        $(ONBLD_TOOLS)/bin/$(MACH)/aw
791 i386_LD=                  /usr/ccs/bin/ld
792 i386_LINT=                $(SPRO_ROOT)/sunstudio12.1/bin/lint

794 amd64_CC=                $(ONBLD_TOOLS)/bin/$(MACH)/cw -_cc
795 $(__GNUC64)amd64_CC=    $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc
796 amd64_CCC=                $(ONBLD_TOOLS)/bin/$(MACH)/cw -_CC
797 $(__GNUC64)amd64_CCC=   $(ONBLD_TOOLS)/bin/$(MACH)/cw -_g++
798 amd64_CPP=                /usr/ccs/lib/cpp
799 amd64_AS=                  $(ONBLD_TOOLS)/bin/$(MACH)/aw
800 amd64_LD=                  /usr/ccs/bin/ld
801 amd64_LINT=                $(SPRO_ROOT)/sunstudio12.1/bin/lint

803 NATIVECC=                $(($(NATIVE_MACH)_CC))
804 NATIVECCC=                $(($(NATIVE_MACH)_CCC))
805 NATIVECPP=                $(($(NATIVE_MACH)_CPP))
806 NATIVEAS=                $(($(NATIVE_MACH)_AS))
807 NATIVELD=                $(($(NATIVE_MACH)_LD))
808 NATIVELINT=               $(($(NATIVE_MACH)_LINT))

810 #
811 # Makefile.master.64 overrides these settings
812 #
813 CC=                        $(NATIVECC)
814 CCC=                        $(NATIVECCC)
815 CPP=                        $(NATIVECPP)
816 AS=                        $(NATIVEAS)
817 LD=                        $(NATIVELD)
818 LINT=                       $(NATIVELINT)

820 # The real compilers used for this build
821 CW_CC_CMD=                 $(CC) -_compiler
822 CW_CCC_CMD=                $(CCC) -_compiler
823 REAL_CC=                   $(CW_CC_CMD:sh)
824 REAL_CCC=                  $(CW_CCC_CMD:sh)

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

829 BDIRECT=                   -Bdirect
830 BDYNAMIC=                   -Bdynamic
831 BLOCAL=                     -Blocal
832 BNODIRECT=                   -Bnodirect
833 BREDUCE=                     -Breduce
834 BSTATIC=                     -Bstatic

836 ZDEFS=                      -zdefs
837 ZDIRECT=                     -zdirect
838 ZIGNORE=                     -zignore
839 ZINITFIRST=                  -zinitfirst
840 ZINTERPOSE=                  -zinterpose
841 ZLAZYLOAD=                   -zlazyload
842 ZLOADFLTR=                   -zloadfltr
843 ZMULDEFS=                    -zmuldefs
844 ZNODEFAULTLIB=               -znodefaultlib
845 ZNODEFS=                     -znodefs
846 ZNODELETE=                   -znodelete
847 ZNODLOPEN=                   -znodlopen
848 ZNODUMP=                     -znodump
849 ZNOLAZYLOAD=                 -znolazyload
850 ZNOLDYNSYM=                  -znoldynsym
851 ZNORELOC=                    -znoreloc
852 ZNOVERSION=                  -znoversion

```

```

853 ZRECORD=                    -zrecord
854 ZREDLOCSYM=                  -zredlocsym
855 ZTEXT=                       -ztext
856 ZVERBOSE=                    -zverbose

858 GSHARED=                     -G
859 CCMT=                         -mt

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

864 sparc_C_PICFLAGS =          -K pic
865 sparcv9_C_PICFLAGS =        -K pic
866 i386_C_PICFLAGS =            -K pic
867 amd64_C_PICFLAGS =          -K pic
868 C_PICFLAGS =                 $(($(MACH)_C_PICFLAGS))
869 C_PICFLAGS64 =               $(($(MACH64)_C_PICFLAGS))

871 sparc_C_BIGPICFLAGS =       -K PIC
872 sparcv9_C_BIGPICFLAGS =     -K PIC
873 i386_C_BIGPICFLAGS =        -K PIC
874 amd64_C_BIGPICFLAGS =       -K PIC
875 C_BIGPICFLAGS =             $(($(MACH)_C_BIGPICFLAGS))
876 C_BIGPICFLAGS64 =           $(($(MACH64)_C_BIGPICFLAGS))

878 # CC requires there to be no space between '-K' and 'pic' or 'PIC'.
879 sparc_CC_PICFLAGS =          -Kpic
880 sparcv9_CC_PICFLAGS =        -KPIC
881 i386_CC_PICFLAGS =           -Kpic
882 amd64_CC_PICFLAGS =          -Kpic
883 CC_PICFLAGS =                $(($(MACH)_CC_PICFLAGS))
884 CC_PICFLAGS64 =              $(($(MACH64)_CC_PICFLAGS))

886 AS_PICFLAGS=                 $(C_PICFLAGS)
887 AS_BIGPICFLAGS=              $(C_BIGPICFLAGS)

889 #
890 # Default label for CTF sections
891 #
892 CTFCVTFLAGS=                  -i -L VERSION
893 $(SRCDGBLD)CTFCVTFLAGS      += -g

895 #
896 # Override to pass module-specific flags to ctfmerge. Currently used only by
897 # krtld to turn on fuzzy matching, and source-level debugging to inhibit
898 # stripping.
899 #
900 CTFMRGFLAGS=
901 $(SRCDGBLD)CTFMRGFLAGS      += -g

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

906 ELFSIGN_O=                    $(TRUE)
907 ELFSIGN_CRYPT=                $(ELFSIGN_O)
908 ELFSIGN_OBJECT=               $(ELFSIGN_O)

910 # Rules (normally from make.rules) and macros which are used for post
911 # processing files. Normally, these do stripping of the comment section
912 # automatically.
913 #     RELEASE_CM:                Should be edited to reflect the release.
914 #     POST_PROCESS_O:            Post-processing for '.o' files.
915 #     POST_PROCESS_A:            Post-processing for '.a' files (currently null).
916 #     POST_PROCESS_SO:          Post-processing for '.so' files.
917 #     POST_PROCESS:              Post-processing for executable files (no suffix).
918 # Note that these macros are not completely generalized as they are to be

```

```

919 # used with the file name to be processed following.
920 #
921 # It is left as an exercise to Release Engineering to embellish the generation
922 # of the release comment string.
923 #
924 # If this is a standard development build:
925 # compress the comment section (mcs -c)
926 # add the standard comment (mcs -a $(RELEASE_CM))
927 # add the development specific comment (mcs -a $(DEV_CM))
928 #
929 # If this is an installation build:
930 # delete the comment section (mcs -d)
931 # add the standard comment (mcs -a $(RELEASE_CM))
932 # add the development specific comment (mcs -a $(DEV_CM))
933 #
934 # If this is an release build:
935 # delete the comment section (mcs -d)
936 # add the standard comment (mcs -a $(RELEASE_CM))
937 #
938 # The following list of macros are used in the definition of RELEASE_CM
939 # which is used to label all binaries in the build:
940 #
941 # RELEASE          Specific release of the build, eg: 5.2
942 # RELEASE_MAJOR    Major version number part of $(RELEASE)
943 # RELEASE_MINOR    Minor version number part of $(RELEASE)
944 # VERSION          Version of the build (alpha, beta, Generic)
945 # PATCHID          If this is a patch this value should contain
946 #                  the patchid value (eg: "Generic 100832-01"), otherwise
947 #                  it will be set to $(VERSION)
948 # RELEASE_DATE     Date of the Release Build
949 # PATCH_DATE       Date the patch was created, if this is blank it
950 #                  will default to the RELEASE_DATE
951 #
952 RELEASE_MAJOR= 5
953 RELEASE_MINOR= 11
954 RELEASE= $(RELEASE_MAJOR).$(RELEASE_MINOR)
955 VERSION= SunOS Development
956 PATCHID= $(VERSION)
957 RELEASE_DATE= release date not set
958 PATCH_DATE= $(RELEASE_DATE)
959 RELEASE_CM= "@$(POUND_SIGN)SunOS $(RELEASE) $(PATCHID) $(PATCH_DATE)"
960 DEV_CM= "@$(POUND_SIGN)SunOS Internal Development: non-nightly build"
961 #
962 PROCESS_COMMENT= @?${MCS} -d -a $(RELEASE_CM) -a $(DEV_CM)
963 $(RELEASE_BUILD)PROCESS_COMMENT= @?${MCS} -d -a $(RELEASE_CM)
964 #
965 STRIP_STABS= :
966 $(RELEASE_BUILD)STRIP_STABS= $(STRIP) -x $@
967 $(SRCDBGBLD)STRIP_STABS= :
968 #
969 POST_PROCESS_O= $(PROCESS_COMMENT) $@
970 POST_PROCESS_A=
971 POST_PROCESS_SO= $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
972                  $(ELFSIGN_OBJECT)
973 POST_PROCESS= $(PROCESS_COMMENT) $@ ; $(STRIP_STABS) ; \
974               $(ELFSIGN_OBJECT)
975 #
976 #
977 # chk4ubin is a tool that inspects a module for a symbol table
978 # ELF section size which can trigger an OBP bug on older platforms.
979 # This problem affects only specific sun4u bootable modules.
980 #
981 CHK4UBIN= $(ONBLD_TOOLS)/bin/$(MACH)/chk4ubin
982 CHK4UBINFLAGS=
983 CHK4UBINARY= $(CHK4UBIN) $(CHK4UBINFLAGS) $@

```

```

985 #
986 # PKGARCHIVE specifies the default location where packages should be
987 # placed if built.
988 #
989 $(RELEASE_BUILD)PKGARCHIVESUFFIX= -nd
990 PKGARCHIVE=$(SRC)/../../packages/$(MACH)/nightly$(PKGARCHIVESUFFIX)
991 #
992 #
993 # The repositories will be created with these publisher settings. To
994 # update an image to the resulting repositories, this must match the
995 # publisher name provided to "pkg set-publisher."
996 #
997 PKGPUBLISHER_REDIST= on-nightly
998 PKGPUBLISHER_NONREDIST= on-extra
999 #
1000 # Default build rules which perform comment section post-processing.
1001 #
1002 .c:
1003     $(LINK.c) -o $@ $< $(LDLIBS)
1004     $(POST_PROCESS)
1005 .c.o:
1006     $(COMPILE.c) $(OUTPUT_OPTION) $< $(CTFCONVERT_HOOK)
1007     $(POST_PROCESS_O)
1008 .c.a:
1009     $(COMPILE.c) -o $% $<
1010     $(PROCESS_COMMENT) $%
1011     $(AR) $(ARFLAGS) $@ $%
1012     $(RM) $%
1013 .s.o:
1014     $(COMPILE.s) -o $@ $<
1015     $(POST_PROCESS_O)
1016 .s.a:
1017     $(COMPILE.s) -o $% $<
1018     $(PROCESS_COMMENT) $%
1019     $(AR) $(ARFLAGS) $@ $%
1020     $(RM) $%
1021 .cc:
1022     $(LINK.cc) -o $@ $< $(LDLIBS)
1023     $(POST_PROCESS)
1024 .cc.o:
1025     $(COMPILE.cc) $(OUTPUT_OPTION) $<
1026     $(POST_PROCESS_O)
1027 .cc.a:
1028     $(COMPILE.cc) -o $% $<
1029     $(AR) $(ARFLAGS) $@ $%
1030     $(PROCESS_COMMENT) $%
1031     $(RM) $%
1032 .y:
1033     $(YACC.y) $<
1034     $(LINK.c) -o $@ y.tab.c $(LDLIBS)
1035     $(POST_PROCESS)
1036     $(RM) y.tab.c
1037 .y.o:
1038     $(YACC.y) $<
1039     $(COMPILE.c) -o $@ y.tab.c $(CTFCONVERT_HOOK)
1040     $(POST_PROCESS_O)
1041     $(RM) y.tab.c
1042 .l:
1043     $(RM) $*.c
1044     $(LEX.l) $< > $*.c
1045     $(LINK.c) -o $@ $*.c -ll $(LDLIBS)
1046     $(POST_PROCESS)
1047     $(RM) $*.c
1048 .l.o:
1049     $(RM) $*.c
1050     $(LEX.l) $< > $*.c

```

## new/usr/src/Makefile.master

```

1051      $(COMPILE.c) -o $@ $*.c $(CTFCONVERT_HOOK)
1052      $(POST_PROCESS_O)
1053      $(RM) $*.c

1055 .bin.o:
1056      $(COMPILE.b) -o $@ $<
1057      $(POST_PROCESS_O)

1059 .java.class:
1060      $(COMPILE.java) $<

1062 # Bourne and Korn shell script message catalog build rules.
1063 # We extract all gettext strings with sed(1) (being careful to permit
1064 # multiple gettext strings on the same line), weed out the dups, and
1065 # build the catalogue with awk(1).

1067 .sh.po .ksh.po:
1068      $(SED) -n -e "a" \
1069      -e "h" \
1070      -e "s/.*gettext *\([^\"]*\).*\1/p" \
1071      -e "x" \
1072      -e "s/\(.*\)gettext *\([^\"]*\).*\1\2/" \
1073      -e "t a" \
1074      $< | sort -u | awk '{ print "msgid\t" $$0 "\nmsgstr" }' > $@

1076 #
1077 # Python and Perl executable and message catalog build rules.
1078 #
1079 .SUFFIXES: .pl .pm .py .pyc

1081 .pl:
1082      $(RM) $@;
1083      $(SED) -e "s@TEXT_DOMAIN@"$(TEXT_DOMAIN)"@" $< > $@;
1084      $(CHMOD) +x $@

1086 .py:
1087      $(RM) $@; $(CAT) $< > $@; $(CHMOD) +x $@

1089 .py.pyc:
1090      $(RM) $@
1091      $(PYTHON) -mpy_compile $<
1092      @[ $(<)c = $@ ] || $(MV) $(<)c $@

1094 .py.po:
1095      $(GNUXGETTEXT) $(GNUXGETFLAGS) -d $(<F:%.py=) $< ;

1097 .pl.po .pm.po:
1098      $(XGETTEXT) $(XGETFLAGS) -d $(<F) $< ;
1099      $(RM) $@ ;
1100      $(SED) "/^domain/d" < $(<F).po > $@ ;
1101      $(RM) $(<F).po

1103 #
1104 # When using xgettext, we want messages to go to the default domain,
1105 # rather than the specified one. This special version of the
1106 # COMPILE.cpp macro effectively prevents expansion of TEXT_DOMAIN,
1107 # causing xgettext to put all messages into the default domain.
1108 #
1109 CPPFORPO=$(COMPILE.cpp:"$(TEXT_DOMAIN)"=TEXT_DOMAIN)

1111 .c.i:
1112      $(CPPFORPO) $< > $@

1114 .h.i:
1115      $(CPPFORPO) $< > $@

```

17

## new/usr/src/Makefile.master

```

1117 .y.i:
1118      $(YACC) -d $<
1119      $(CPPFORPO) y.tab.c > $@
1120      $(RM) y.tab.c

1122 .l.i:
1123      $(LEX) $<
1124      $(CPPFORPO) lex.yy.c > $@
1125      $(RM) lex.yy.c

1127 .c.po:
1128      $(CPPFORPO) $< > $<.i
1129      $(BUILD.po)

1131 .y.po:
1132      $(YACC) -d $<
1133      $(CPPFORPO) y.tab.c > $<.i
1134      $(BUILD.po)
1135      $(RM) y.tab.c

1137 .l.po:
1138      $(LEX) $<
1139      $(CPPFORPO) lex.yy.c > $<.i
1140      $(BUILD.po)
1141      $(RM) lex.yy.c

1143 #
1144 # Rules to perform stylistic checks
1145 #
1146 .SUFFIXES: .x .xml .check .xmlchk

1148 .h.check:
1149      $(DOT_H_CHECK)

1151 .x.check:
1152      $(DOT_X_CHECK)

1154 .xml.xmlchk:
1155      $(MANIFEST_CHECK)

1157 #
1158 # Include rules to render automated sccs get rules "safe".
1159 #
1160 include $(SRC)/Makefile.noget

```

18

new/usr/src/cmd/Makefile

1

\*\*\*\*\*

10884 Fri Jan 17 07:29:14 2014

new/usr/src/cmd/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 #

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

28 include ../Makefile.master

30 #
31 # Note that the commands 'agents', 'lp', 'perl', and 'man' are first in
32 # the list, violating alphabetical order. This is because they are very
33 # long-running and should be given the most wall-clock time for a
34 # parallel build.
35 #
36 # Commands in the FIRST_SUBDIRS list are built before starting the build
37 # of other commands. Currently this includes only 'isaexec' and
38 # 'platexec'. This is necessary because $(ROOT)/usr/lib/isaexec or
39 # $(ROOT)/usr/lib/platexec must exist when some other commands are built
40 # because their 'make install' creates a hard link to one of them.
41 #
42 # Commands are listed one per line so that TeamWare can auto-merge most
43 # changes.
44 #

46 FIRST_SUBDIRS= \
47     isaexec \
48     platexec

50 COMMON_SUBDIRS= \
51     allocate \
52     availdevs \
53     lp \
54     perl \
55     man \
56     Adm \
57     abi \
58     adbgen \
59     acct \
60     acctadm \
61     acpitools \
```

new/usr/src/cmd/Makefile

2

```
62     arch \
63     asa \
64     ast \
65     audio \
66     auths \
67     autopush \
68     avs \
69     awk \
70     awk_xpg4 \
71     backup \
72     banner \
73     bart \
74     basename \
75     bc \
76     bdiff \
77     beadm \
78     bfs \
79     bnu \
80     boot \
81     busstat \
82     cal \
83     calendar \
84     captinfo \
85     cat \
86     cdrw \
87     cfgadm \
88     checkeq \
89     checknr \
90     chgrp \
91     chmod \
92     chown \
93     chroot \
94     clear \
95     clinfo \
96     cmd-crypto \
97     cmd-inet \
98     col \
99     compress \
100    consadm \
101    coreadm \
102    cpio \
103    cpc \
104    cron \
105    crypt \
106    csh \
107    csplit \
108    ctrun \
109    ctstat \
110    ctwatch \
111    datadm \
112    date \
113    dc \
114    dd \
115    deroff \
116    devfsadm \
117    syseventd \
118    devctl \
119    devinfo \
120    devmgmt \
121    devprop \
122    dfs.cmds \
123    diff \
124    diff3 \
125    diffmk \
126    dircmp \
127    dirname \
```

## new/usr/src/cmd/Makefile

```

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 /

```

3

## new/usr/src/cmd/Makefile

```

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      dlmgmt /
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      msg /
259      mkdir /

```

4

## new/usr/src/cmd/Makefile

```

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     /

```

5

## new/usr/src/cmd/Makefile

```

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      setpggrp   /
370      setuname   /
371      sgs        /
372      sh         /
373      shcomp     /
374      smbios     /
375      smbfsrv    /
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      stmftproxy /
389      stmfsvc    /
390      stmsboot   /
391      streams    /

```

6



## new/usr/src/cmd/Makefile

```

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      volrmount /
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      /

```

7

## new/usr/src/cmd/Makefile

```

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      acpihp       /
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    /

```

8

## new/usr/src/cmd/Makefile

```

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     /

```

9

## new/usr/src/cmd/Makefile

```

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      msg         /
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          /

```

10

## new/usr/src/cmd/Makefile

```

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   \

```

11

## new/usr/src/cmd/Makefile

```

722      verrpadm    \
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    \
742      \
743      sparc_MSGSUBDIRS= \
744      fruadm      \
745      prtdscp     \
746      prtfru      \
747      virtinfo    \
748      vntsd       \
749      \
750      i386_MSGSUBDIRS= \
751      ucodeadm    \
752      \
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       \
768      \
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     \
783      \
784      #
785      # commands not owned by the systems group
786      #
787      BWOSDIRS=

```

12

```
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:
```

new/usr/src/cmd/acpitools/Makefile

1

\*\*\*\*\*

786 Fri Jan 17 07:29:14 2014

new/usr/src/cmd/acpitools/Makefile

PANKOVs restructure

\*\*\*\*\*

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet at
9 # http://www.illumos.org/license/CDDL.
10 #
11 #
12 #
13 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
14 #
15 #
16 include      $(SRC)/cmd/Makefile.cmd
17 #
18 SUBDIRS=      acpiexec acpixtract iasl
19 #
20 all           := TARGET = all
21 clean        := TARGET = clean
22 clobber      := TARGET = clobber
23 install      := TARGET = install
24 #
25 .KEEP_STATE:
26 #
27 all clean clobber install: $(SUBDIRS)
28 #
29 $(SUBDIRS):   FRC
30               @cd $@; pwd; $(MAKE) $(TARGET)
31 #
32 include      $(SRC)/cmd/Makefile.targ
33 #
34 FRC:
```

new/usr/src/cmd/acpitools/acpiexec/Makefile

1

\*\*\*\*\*

4469 Fri Jan 17 07:29:15 2014  
new/usr/src/cmd/acpitools/acpiexec/Makefile  
update to acpica-unix2-20140114  
update to acpica-unix2-20130927  
acpica-unix2-20130823  
PANKOV's restructure  
\*\*\*\*\*

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet at
9 # http://www.illumos.org/license/CDDL.
10 #
11 #
12 #
13 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
14 #
15
16 PROG=          acpiexec
17 OBJS=          oslstubs.o
18 # common
19 OBJS +=        getopt.o acgetline.o
20
21 # components/debugger
22 OBJS +=        dbcmds.o dbdisply.o dbexec.o dbfileio.o dbhistory.o \
23               dbinput.o dbmethod.o dbnames.o dbstats.o dbtest.o dbutils.o
24               dbxface.o dbconvert.o
25 # components/disassembler
26 OBJS +=        dmbuffer.o dmnames.o dmbobject.o dmpopcode.o dmresrc.o \
27               dmresrc1.o dmresrcs.o dmutils.o dmwalk.o dmdeferred.o \
28               dmresrc12.o
29 # components/dispatcher
30 OBJS +=        dsargs.o dscontrol.o dsfield.o dsinit.o dsmethod.o \
31               dsmtthdat.o dsobject.o dsopcode.o dsutils.o dswexec.o \
32               dswload.o dswload2.o dswscope.o dswstate.o
33 # components/events
34 OBJS +=        evevent.o evglock.o evgpe.o evgpeblk.o evgpeinit.o \
35               evgpeutil.o evmisc.o evregion.o evrgnini.o evsci.o \
36               evxface.o evxfevnt.o evxfge.o evxfregn.o evhandler.o
37 # components/executor
38 OBJS +=        exconfig.o exconvrt.o excreate.o exdebug.o exdump.o \
39               exfield.o exfldio.o exmisc.o exmutex.o exnames.o \
40               exoparg1.o exoparg2.o exoparg3.o exoparg6.o exprep.o \
41               exregion.o exresnte.o exresolv.o exresop.o exstore.o \
42               exstoren.o exstorob.o exsystem.o exutils.o
43 # components/hardware
44 OBJS +=        hwacpi.o hwgpe.o hwpci.o hwregs.o hwsleep.o hwvalid.o \
45               hwxface.o hwsleep.o hwxfsleep.o
46 # components/namespace
47 OBJS +=        nsaccess.o nsalloc.o nsdump.o nsdumpdv.o nseval.o \
48               nsinit.o nsload.o nsnames.o nsobject.o nsparse.o \
49               nspredef.o nsrepair.o nsrepair2.o nssearch.o nsutils.o \
50               nswalk.o nsxfeval.o nsxfname.o nsxfobj.o nsarguments.o \
51               nsconvert.o nsprepkg.o
52
53 # components/parser
54 OBJS +=        psargs.o psloop.o psopcode.o psparse.o psscope.o \
55               pstree.o psutils.o pswalk.o psxface.o psubject.o psopinfo.o
56 # components/resources
57 OBJS +=        rsaddr.o rscalc.o rscreate.o rsdump.o rsinfo.o rsio.o \
58               rsirq.o rslist.o rsmemory.o rsmisc.o rsutils.o \
```

new/usr/src/cmd/acpitools/acpiexec/Makefile

2

```
59          rsxface.o rsdumpinfo.o rsserial.o
60 # components/tables
61 OBJS +=        tbfadt.o tbfind.o tbinstal.o tbutils.o tbxface.o \
62               tbxfroot.o tbprint.o tbxfload.o
63 # components/utilities
64 OBJS +=        utalloc.o utcache.o utcopy.o utdebug.o utdecode.o \
65               utdelete.o uteval.o utglobal.o utids.o utinit.o \
66               utlock.o utmath.o utmisc.o utmutex.o utobject.o \
67               utresrc.o utstate.o uttrack.o utosi.o utxferror.o \
68               utxface.o utaddress.o utbuffer.o uterror.o utexcep.o \
69               utownerid.o utpredef.o utstring.o utxfinit.o utxfmutex.o
70 # os_specific/service_layers
71 OBJS +=        osunixxf.o
72 # tools/acpiexec
73 OBJS +=        aeexec.o aehandlers.o aemain.o aetables.o
74
75 ACPICA=        $(SRC)/common/acpica
76
77 include        $(SRC)/cmd/Makefile.cmd
78
79 C99MODE=       $(C99_ENABLE)
80 CPPFLAGS +=    -DACPI_EXEC_APP -I$(ACPICA)/include -I$(ACPICA)/compiler
81
82 .KEEP_STATE:
83
84 all:           $(PROG)
85
86 clean:         $(RM) $(OBJS)
87
88 install:       all $(ROOTPROG)
89
90 $(PROG):       $(OBJS)
91               $(LINK.c) $(OBJS) -o $$@ $(LDLIBS)
92               $(POST_PROCESS)
93
94 %.o:           $(ACPICA)/common/%.c
95               $(COMPILE.c) -o $$@ $<
96               $(POST_PROCESS_O)
97
98 %.o:           $(ACPICA)/compiler/%.c
99               $(COMPILE.c) -o $$@ $<
100              $(POST_PROCESS_O)
101
102 %.o:           $(ACPICA)/components/debugger/%.c
103               $(COMPILE.c) -o $$@ $<
104               $(POST_PROCESS_O)
105
106 %.o:           $(ACPICA)/components/disassembler/%.c
107               $(COMPILE.c) -o $$@ $<
108               $(POST_PROCESS_O)
109
110 %.o:           $(ACPICA)/components/dispatcher/%.c
111               $(COMPILE.c) -o $$@ $<
112               $(POST_PROCESS_O)
113
114 %.o:           $(ACPICA)/components/events/%.c
115               $(COMPILE.c) -o $$@ $<
116               $(POST_PROCESS_O)
117
118 %.o:           $(ACPICA)/components/executor/%.c
119               $(COMPILE.c) -o $$@ $<
120               $(POST_PROCESS_O)
121
122 %.o:           $(ACPICA)/components/hardware/%.c
123               $(COMPILE.c) -o $$@ $<
124               $(POST_PROCESS_O)
```

```
125          $(POST_PROCESS_O)

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

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

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

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

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

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

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

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

new/usr/src/cmd/acpitools/acpiexec/oslstubs.c

1

\*\*\*\*\*

681 Fri Jan 17 07:29:15 2014

new/usr/src/cmd/acpitools/acpiexec/oslstubs.c

PANKOVs restructure

\*\*\*\*\*

```
1 /*
2  * This file and its contents are supplied under the terms of the
3  * Common Development and Distribution License ("CDDL"), version 1.0.
4  * You may only use this file in accordance with the terms of version
5  * 1.0 of the CDDL.
6  *
7  * A full copy of the text of the CDDL should have accompanied this
8  * source. A copy of the CDDL is also available via the Internet at
9  * http://www.illumos.org/license/CDDL.
10 */

12 /*
13  * Copyright 2013 Nexenta Systems, Inc. All rights reserved.
14 */

16 /*
17  * OSL stubs
18 */

20 #include <acpi.h>

22 UINT32
23 __acpi_acquire_global_lock(void *Facs)
24 {
25     return (0);
26 }

28 UINT32
29 __acpi_release_global_lock(void *Facs)
30 {
31     return (0);
32 }

34 void
35 __acpi_wbinvd(void)
36 {
37 }
```



new/usr/src/cmd/acpitools/acpinames/Makefile

1

\*\*\*\*\*

2989 Fri Jan 17 07:29:15 2014  
new/usr/src/cmd/acpitools/acpinames/Makefile  
acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet at
9 # http://www.illumos.org/license/CDDL.
10 #
11 #
12 #
13 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
14 #
```

```
16 PROG=          acpinames
17 # common
18 OBJS +=         getopt.o
19 #
20 OBJS +=         anmain.o anstubs.o antables.o
21 OBJS +=         dbfileio.o
22 OBJS +=         dsfield.o dsmthdat.o dsobject.o dsutils.o dswload.o \
23                dswload2.o dswscope.o dswstate.o
24 OBJS +=         excreate.o exnames.o exresnte.o exresolv.o exutils.o
25 OBJS +=         nsaccess.o nsalloc.o nsdump.o nsinit.o nsload.o \
26                nsnames.o nsobject.o nsparse.o nssearch.o nsutils.o \
27                nswalk.o nsxfeval.o nsxfname.o nsxfobj.o
28 OBJS +=         osunixxf.o
29 OBJS +=         psargs.o psloop.o psopcode.o psparse.o psscope.o \
30                pstree.o psutils.o pswalk.o psxface.o psobject.o psopinfo.o
31 OBJS +=         tbfadt.o tbfnd.o tbinstal.o tbutils.o tbxface.o \
32                tbxfroot.o tbprint.o tbxfload.o
33 OBJS +=         utalloc.o utcache.o utdebug.o utdecode.o utdelete.o \
34                utglobal.o utlock.o utmath.o utmisc.o utmutex.o \
35                utobject.o utstate.o utosi.o utxferror.o utxface.o \
36                utaddress.o uterror.o utexcep.o utownerid.o utstring.o \
37                utxfinit.o
38
39 ACPICA=          $(SRC)/common/acpica
40
41 include          $(SRC)/cmd/Makefile.cmd
42
43 C99MODE=         $(C99_ENABLE)
44 CPPFLAGS +=      -DACPI_NAMES_APP \
45                -I$(ACPICA)/include -I$(ACPICA)/tools/acpinames
46
47
48 .KEEP_STATE:
49
50 all:             $(PROG)
51
52 clean:
53                $(RM) $(OBJS)
54
55 install:         all $(ROOTPROG)
56
57 $(PROG):          $(OBJS)
58                $(LINK.c) $(OBJS) -o $@ $(LDLIBS)
59                $(POST_PROCESS)
```

new/usr/src/cmd/acpitools/acpinames/Makefile

2

```
61 %.o:            $(ACPICA)/common/%.c
62                $(COMPILE.c) -o $@ $<
63                $(POST_PROCESS_O)
64
65 %.o:            $(ACPICA)/compiler/%.c
66                $(COMPILE.c) -o $@ $<
67                $(POST_PROCESS_O)
68
69 %.o:            $(ACPICA)/components/debugger/%.c
70                $(COMPILE.c) -o $@ $<
71                $(POST_PROCESS_O)
72
73 %.o:            $(ACPICA)/components/disassembler/%.c
74                $(COMPILE.c) -o $@ $<
75                $(POST_PROCESS_O)
76
77 %.o:            $(ACPICA)/components/dispatcher/%.c
78                $(COMPILE.c) -o $@ $<
79                $(POST_PROCESS_O)
80
81 %.o:            $(ACPICA)/components/events/%.c
82                $(COMPILE.c) -o $@ $<
83                $(POST_PROCESS_O)
84
85 %.o:            $(ACPICA)/components/executer/%.c
86                $(COMPILE.c) -o $@ $<
87                $(POST_PROCESS_O)
88
89 %.o:            $(ACPICA)/components/hardware/%.c
90                $(COMPILE.c) -o $@ $<
91                $(POST_PROCESS_O)
92
93 %.o:            $(ACPICA)/components/namespace/%.c
94                $(COMPILE.c) -o $@ $<
95                $(POST_PROCESS_O)
96
97 %.o:            $(ACPICA)/components/parser/%.c
98                $(COMPILE.c) -o $@ $<
99                $(POST_PROCESS_O)
100
101 %.o:            $(ACPICA)/components/resources/%.c
102                $(COMPILE.c) -o $@ $<
103                $(POST_PROCESS_O)
104
105 %.o:            $(ACPICA)/components/tables/%.c
106                $(COMPILE.c) -o $@ $<
107                $(POST_PROCESS_O)
108
109 %.o:            $(ACPICA)/components/utilities/%.c
110                $(COMPILE.c) -o $@ $<
111                $(POST_PROCESS_O)
112
113 %.o:            $(ACPICA)/os_specific/service_layers/%.c
114                $(COMPILE.c) -o $@ $<
115                $(POST_PROCESS_O)
116
117 %.o:            $(ACPICA)/tools/acpinames/%.c
118                $(COMPILE.c) -o $@ $<
119                $(POST_PROCESS_O)
120
121 include          $(SRC)/cmd/Makefile.targ
```

new/usr/src/cmd/acpitools/acpextract/Makefile

1

\*\*\*\*\*

1004 Fri Jan 17 07:29:15 2014

new/usr/src/cmd/acpitools/acpextract/Makefile

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet at
9 # http://www.illumos.org/license/CDDL.
10 #
```

```
12 #
13 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
14 #
```

```
16 PROG=      acpextract
17 OBJS=      acpextract.o axmain.o getopt.o

19 ACPICA=     $(SRC)/common/acpica

21 include     $(SRC)/cmd/Makefile.cmd

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

26 .KEEP\_STATE:

```
28 $(PROG):    $(OBJS)
29             $(LINK.c) $(OBJS) -o $@ $(LDLIBS)
30             $(POST_PROCESS)
```

```
32 %.o:       $(ACPICA)/common/%.c
33             $(COMPILE.c) -o $@ $<
34             $(POST_PROCESS_O)
```

```
36 %.o:       $(ACPICA)/tools/acpextract/%.c
37             $(COMPILE.c) -o $@ $<
38             $(POST_PROCESS_O)
```

```
40 all:       $(PROG)
```

```
42 clean:
43             $(RM) $(OBJS)
```

```
45 install:   all $(ROOTPROG)
```

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

new/usr/src/cmd/acpitools/iasl/Makefile

1

\*\*\*\*\*

5193 Fri Jan 17 07:29:15 2014  
new/usr/src/cmd/acpitools/iasl/Makefile  
fix typo in CLEANFILES for acpitools/iasl  
acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet at
9 # http://www.illumos.org/license/CDDL.
10 #
11 #
12 #
13 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
14 #
15
16 PROG=      iasl
17 # common
18 OBJS +=    adfile.o adisasm.o adwalk.o dmextern.o \
19            dmrestag.o dmtree.o dmbdinfo.o getopt.o \
20            ahpredef.o
21 # compiler (generated)
22 OBJS +=    aslcompilerparse.o aslcompilerlex.o dtparserparse.o \
23            dtparserlex.o prexpress.o prmacros.o prscan.o prutils.o \
24            prparserparse.o prparserlex.o
25 # compiler
26 OBJS +=    aslanalyze.o aslbtypes.o aslcodegen.o aslcompile.o \
27            aslerror.o aslfiles.o aslfold.o asllength.o \
28            asllisting.o aslload.o asllookup.o aslmain.o aslmap.o \
29            aslopcode.o asloperands.o aslopt.o aslpredef.o \
30            aslresource.o aslrestype1.o aslrestype1i.o \
31            aslrestype2.o aslrestype2d.o aslrestype2e.o \
32            aslrestype2q.o aslrestype2w.o aslstartup.o aslstubs.o \
33            asltransform.o asltree.o aslutils.o asluuid.o \
34            aslwalks.o dtcompile.o dtexpress.o dtfield.o dtio.o \
35            dtsubtable.o dttable.o dttemplate.o dtutils.o aslfileio.o \
36            aslhex.o asllistsup.o aslmethod.o aslnamesp.o asloffset.o \
37            asloptions.o aslprepkg.o aslrestype2s.o aslxref.o
38 # components/debugger
39 OBJS +=    dbfileio.o
40 # components/disassembler
41 OBJS +=    dmbuffer.o dmnames.o dmopcode.o dmresrc.o dmresrc1.o \
42            dmresrcs.o dmutils.o dmwalk.o dmdeferred.o dmbobject.o \
43            dmresrc12.o
44
45 # components/dispatcher
46 OBJS +=    dsargs.o dscontrol.o dsfield.o dsobject.o dsopcode.o \
47            dsutils.o dswwxec.o dswload.o dswload2.o dswscope.o \
48            dswstate.o
49 # components/executor
50 OBJS +=    exconvrt.o excreate.o exdump.o exmisc.o exmutex.o \
51            exnames.o exoparg1.o exoparg2.o exoparg3.o exoparg6.o \
52            exprep.o exresnte.o exresolv.o exresop.o exstore.o \
53            exstoren.o exstorob.o exsystem.o exutils.o exregion.o
54 # components/namespace
55 OBJS +=    nsaccess.o nsalloc.o nsdump.o nsnames.o nsobject.o \
56            nsparse.o nssearch.o nsutils.o nswalk.o nsxfobj.o
57 # components/parser
58 OBJS +=    psargs.o psloop.o psopcode.o psparse.o psscope.o \
59            pstree.o psutils.o pswalk.o psobject.o psopinfo.o
```

new/usr/src/cmd/acpitools/iasl/Makefile

2

```
60 # components/tables
61 OBJS +=    tbfadt.o tbinstal.o tbutils.o tbxface.o tbprint.o
62 # components/utilities
63 OBJS +=    utalloc.o utcache.o utcopy.o utdebug.o utdecode.o \
64            utdelete.o utglobal.o utinit.o utlock.o utmath.o \
65            utmisc.o utmutex.o utobject.o utresrc.o utstate.o \
66            utxface.o utxferror.o utaddress.o utbuffer.o uterror.o \
67            utexcep.o utownerid.o utpredef.o utstring.o
68
69 # os_specific/service_layers
70 OBJS +=    osunixxf.o
71
72 CLEANFILES= aslcompiler.y.h aslcompilerlex.c aslcompilerparse.c \
73             dtparser.y.h dtparserlex.c dtparserparse.c prparser.y.h \
74             prparserlex.c prparserparse.c
75
76 ACPICA=     $(SRC)/common/acpica
77
78 include     $(SRC)/cmd/Makefile.cmd
79
80 C99MODE=    $(C99_ENABLE)
81 CPPFLAGS += -DACPI_ASL_COMPILER -DACPI_USE_STANDARD_HEADERS \
82             -I$(ACPICA)/include -I$(ACPICA)/compiler -I.
83
84 # Flags for flex and bison
85 LFLAGS=     -i -s
86 YFLAGS=     -d
87
88 .KEEP_STATE:
89
90 all:        $(PROG)
91
92 clean:
93             $(RM) $(CLEANFILES) $(OBJS)
94
95 install:    all $(ROOTPROG)
96
97 $(PROG):    $(OBJS)
98             $(LINK.c) $(OBJS) -o $@ $(LDLIBS)
99             $(POST_PROCESS)
100
101 aslcompilerlex.c: $(ACPICA)/compiler/aslcompiler.l
102                 $(FLEX) $(LFLAGS) -PAslCompiler -o$@ $(ACPICA)/compiler/aslcompil
103
104 aslcompilerparse.c: $(ACPICA)/compiler/aslcompiler.y
105                     $(BISON) $(YFLAGS) -pAslCompiler -o$@ $(ACPICA)/compiler/aslcomp
106                     $(RM) aslcompiler.y.h; $(MV) aslcompilerparse.h aslcompiler.y.h
107
108 dtparserlex.c:    $(ACPICA)/compiler/dtparser.l
109                 $(FLEX) $(LFLAGS) -PDtParser -o$@ $(ACPICA)/compiler/dtparser.l
110
111 dtparserparse.c: $(ACPICA)/compiler/dtparser.y
112                     $(BISON) $(YFLAGS) -PDtParser -o$@ $(ACPICA)/compiler/dtparser.y
113                     $(RM) dtparser.y.h; $(MV) dtparserparse.h dtparser.y.h
114
115 prparserlex.c:    $(ACPICA)/compiler/prparser.l
116                 $(FLEX) $(LFLAGS) -PPrParser -o$@ $(ACPICA)/compiler/prparser.l
117
118 prparserparse.c: $(ACPICA)/compiler/prparser.y
119                     $(BISON) $(YFLAGS) -PPrParser -o$@ $(ACPICA)/compiler/prparser.y
120                     $(RM) prparser.y.h; $(MV) prparserparse.h prparser.y.h
121
122 %.o:        $(ACPICA)/common/%.c
123             $(COMPILE.c) -o $@ $<
124
125             $(POST_PROCESS_O)
```

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

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

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

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

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

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

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

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

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

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

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

new/usr/src/cmd/mdb/i86pc/modules/apix/amd64/Makefile

1

\*\*\*\*\*

1345 Fri Jan 17 07:29:15 2014

new/usr/src/cmd/mdb/i86pc/modules/apix/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) 2010, Oracle and/or its affiliates. All rights reserved.
23 #
```

```
25 MODULE = apix.so
26 MDBTGT = kvm
```

```
28 MODSRCS = apix.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
```

new/usr/src/cmd/mdb/i86pc/modules/apix/ia32/Makefile

1

\*\*\*\*\*

1305 Fri Jan 17 07:29:16 2014

new/usr/src/cmd/mdb/i86pc/modules/apix/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) 2010, Oracle and/or its affiliates. All rights reserved.
23 #
```

25 MODULE = apix.so

26 MDBTGT = kvm

28 MODSRCS = apix.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

new/usr/src/cmd/mdb/i86pc/modules/pcplusmp/amd64/Makefile

1

\*\*\*\*\*

1359 Fri Jan 17 07:29:16 2014

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
```

new/usr/src/cmd/mdb/i86pc/modules/pcplusmp/ia32/Makefile

1

\*\*\*\*\*

1319 Fri Jan 17 07:29:17 2014

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
```



new/usr/src/cmd/mdb/i86pc/modules/unix/amd64/Makefile

1

\*\*\*\*\*

1443 Fri Jan 17 07:29:17 2014

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
```

new/usr/src/cmd/mdb/i86pc/modules/unix/ia32/Makefile

1

\*\*\*\*\*

1403 Fri Jan 17 07:29:17 2014

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
```

new/usr/src/cmd/mdb/i86pc/modules/uppc/amd64/Makefile

1

\*\*\*\*\*

1355 Fri Jan 17 07:29:18 2014

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
```

new/usr/src/cmd/mdb/i86pc/modules/uppc/ia32/Makefile

1

\*\*\*\*\*

1315 Fri Jan 17 07:29:18 2014

new/usr/src/cmd/mdb/i86pc/modules/uppc/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 = uppc.so
27 MDBTGT = kvm

29 MODSRCS = uppc.c intr_common.c

31 include ../../../../Makefile.cmd
32 include ../../../../intel/Makefile.ia32
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
```

new/usr/src/cmd/mdb/i86xpv/modules/unix/amd64/Makefile

1

\*\*\*\*\*

1563 Fri Jan 17 07:29:18 2014

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
```

new/usr/src/cmd/mdb/i86xpv/modules/unix/ia32/Makefile

1

\*\*\*\*\*

1522 Fri Jan 17 07:29:19 2014

new/usr/src/cmd/mdb/i86xpv/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.i86xpv
34 include ../../../../Makefile.module

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

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

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

new/usr/src/cmd/mdb/i86xpv/modules/xpv\_psm/amd64/Makefile

1

\*\*\*\*\*

1474 Fri Jan 17 07:29:19 2014

new/usr/src/cmd/mdb/i86xpv/modules/xpv\_psm/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 = xpv_psm.so
27 MDBTGT = kvm

29 MODSRCS = xpv_psm.c intr_common.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/common

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

new/usr/src/cmd/mdb/i86xpv/modules/xpv\_psm/ia32/Makefile

1

\*\*\*\*\*

1434 Fri Jan 17 07:29:19 2014

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
```



new/usr/src/cmd/mdb/i86xpv/modules/xpv\_uppc/amd64/Makefile

1

\*\*\*\*\*

1476 Fri Jan 17 07:29:20 2014

new/usr/src/cmd/mdb/i86xpv/modules/xpv\_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 2008 Sun Microsystems, Inc. All rights reserved.
23 # Use is subject to license terms.
24 #
25 #ident "%Z%M% %I% %E% SMI"

26 MODULE = xpv_uppc.so
27 MDBTGT = kvm

29 MODSRCS = xpv_uppc.c intr_common.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/common

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

new/usr/src/cmd/mdb/i86xpv/modules/xpv\_uppc/ia32/Makefile

1

\*\*\*\*\*

1436 Fri Jan 17 07:29:20 2014

new/usr/src/cmd/mdb/i86xpv/modules/xpv\_uppc/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 2008 Sun Microsystems, Inc. All rights reserved.
23 # Use is subject to license terms.
24 #
25 #ident "%Z%M% %I% %E% SMI"

26 MODULE = xpv_uppc.so
27 MDBTGT = kvm

29 MODSRCS = xpv_uppc.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
```

new/usr/src/common/acpica/README.txt

1

\*\*\*\*\*

168 Fri Jan 17 07:29:20 2014

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/>.

new/usr/src/common/acpica/THIRDPARTYLICENSE

1

\*\*\*\*\*

1872 Fri Jan 17 07:29:21 2014

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 MERCHANTABILITY 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.
```

new/usr/src/common/acpica/changes.txt

1

\*\*\*\*\*

565811 Fri Jan 17 07:29:21 2014  
new/usr/src/common/acpica/changes.txt  
update to acpica-unix2-20140114  
update to acpica-unix2-20131218  
update to acpica-unix2-20131115  
update to acpica-unix2-20130927  
acpica-unix2-20130823  
PANKOVs restructure

\*\*\*\*\*

1 -----  
2 14 January 2014. Summary of changes for version 20140114:  
2 27 May 2011. Summary of changes for version 20110527:

4 1) ACPICA kernel-resident subsystem:  
4 This release is available at [www.acpica.org/downloads](http://www.acpica.org/downloads)

6 Updated all ACPICA copyrights and signons to 2014. Added the 2014  
7 copyright to all module headers and signons, including the standard Linux  
8 header. This affects virtually every file in the ACPICA core subsystem,  
9 iASL compiler, all ACPICA utilities, and the test suites.

11 Improved parameter validation for AcpiInstallGpeBlock. Added the  
12 following checks:  
13 1) The incoming device handle refers to type ACPI\_TYPE\_DEVICE.  
14 2) There is not already a GPE block attached to the device.  
15 Likewise, with AcpiRemoveGpeBlock, ensure that the incoming object is a  
16 device.

18 Correctly support "references" in the ACPI\_OBJECT. This change fixes the  
19 support to allow references (namespace nodes) to be passed as arguments  
20 to control methods via the evaluate object interface. This is probably  
21 most useful for testing purposes, however.

23 Improved support for 32/64 bit physical addresses in printf()-like  
24 output. This change improves the support for physical addresses in printf  
25 debug statements and other output on both 32-bit and 64-bit hosts. It  
26 consistently outputs the appropriate number of bytes for each host. The  
27 %p specifier is unsatisfactory since it does not emit uniform output on  
28 all hosts/clib implementations (on some, leading zeros are not supported,  
29 leading to difficult-to-read output).

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

36 Current Release:  
37 Non-Debug Version: 96.2K Code, 27.0K Data, 123.2K Total  
38 Debug Version: 187.5K Code, 78.3K Data, 265.8K Total  
39 Previous Release:  
40 Non-Debug Version: 96.1K Code, 27.0K Data, 123.1K Total  
41 Debug Version: 185.6K Code, 77.3K Data, 262.9K Total

44 2) iASL Compiler/Disassembler and Tools:

46 iASL: Fix a possible fault when using the Connection() operator. Fixes a  
47 problem if the parent Field definition for the Connection operator refers  
48 to an operation region that does not exist. ACPICA BZ 1064.

50 AcpiExec: Load of local test tables is now optional. The utility has the  
51 capability to load some various tables to test features of ACPICA.  
52 However, there are enough of them that the output of the utility became  
53 confusing. With this change, only the required local tables are displayed  
54 (RSDP, XSDT, etc.) along with the actual tables loaded via the command

new/usr/src/common/acpica/changes.txt

2

55 line specification. This makes the default output simpler and easier to  
56 understand. The -el command line option restores the original behavior  
57 for testing purposes.

59 AcpiExec: Added support for overlapping operation regions. This change  
60 expands the simulation of operation regions by supporting regions that  
61 overlap within the given address space. Supports SystemMemory and  
62 SystemIO. ASLTS test suite updated also. David Box. ACPICA BZ 1031.

64 AcpiExec: Added region handler support for PCI\_Config and EC spaces. This  
65 allows AcpiExec to simulate these address spaces, similar to the current  
66 support for SystemMemory and SystemIO.

68 Debugger: Added new command to read/write/compare all namespace objects.  
69 The command "test objects" will exercise the entire namespace by writing  
70 new values to each data object, and ensuring that the write was  
71 successful. The original value is then restored and verified.

73 Debugger: Added the "test predefined" command. This change makes this  
74 test public and puts it under the new "test" command. The test executes  
75 each and every predefined name within the current namespace.

77 -----  
78 18 December 2013. Summary of changes for version 20131218:

80 Global note: The ACPI 5.0A specification was released this month. There  
81 are no changes needed for ACPICA since this release of ACPI is an  
82 errata/clarification release. The specification is available at  
83 [acpi.info](http://acpi.info).

86 1) ACPICA kernel-resident subsystem:

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

96 Added a runtime option to ignore the XSDT and force the use of the RSDT.  
97 This change adds a runtime option that will force ACPICA to use the RSDT  
98 instead of the XSDT (AcpiGbl\_DoNotUseXsdt). Although the ACPI spec  
99 requires that an XSDT be used instead of the RSDT, the XSDT has been  
100 found to be corrupt or ill-formed on some machines. Lv Zheng.

102 Added a runtime option to favor 32-bit FADT register addresses over the  
103 64-bit addresses. This change adds an option to favor 32-bit FADT  
104 addresses when there is a conflict between the 32-bit and 64-bit versions  
105 of the same register. The default behavior is to use the 64-bit version  
106 in accordance with the ACPI specification. This can now be overridden via  
107 the AcpiGbl\_Use32BitFadtAddresses flag. ACPICA BZ 885. Lv Zheng.

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

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

117 Added helper macros to extract bus/segment numbers from the HEST table.  
118 This change adds two macros to extract the encoded bus and segment  
119 numbers from the HEST Bus field - ACPI\_HEST\_BUS and ACPI\_HEST\_SEGMENT.  
120 Betty Dall <[betty.dall@hp.com](mailto:betty.dall@hp.com)>

122 Removed the unused ACPI\_FREE\_BUFFER macro. This macro is no longer used  
 123 by ACPICA. It is not a public macro, so it should have no effect on  
 124 existing OSV code. Lv Zheng.

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

131 Current Release:  
 132 Non-Debug Version: 96.1K Code, 27.0K Data, 123.1K Total  
 133 Debug Version: 185.6K Code, 77.3K Data, 262.9K Total  
 134 Previous Release:  
 135 Non-Debug Version: 95.9K Code, 27.0K Data, 122.9K Total  
 136 Debug Version: 185.1K Code, 77.2K Data, 262.3K Total

## 139 2) iASL Compiler/Disassembler and Tools:

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

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

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

160 iASL: Added full support for the DBG2 table. Adds full disassembler,  
 161 table compiler, and template generator support for the DBG2 table (Debug  
 162 Port 2 table).

164 iASL: Added full support for the PCCT table, update the table definition.  
 165 Updates the PCCT table definition in the actbl3.h header and adds table  
 166 compiler and template generator support.

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

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

179 -----  
 180 15 November 2013. Summary of changes for version 20131115:

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

## 185 1) ACPICA kernel-resident subsystem:

187 Resource Manager: Fixed loop termination for the "get AML length"  
 188 function. The loop previously had an error termination on a NULL resource  
 189 pointer, which can never happen since the loop simply increments a valid  
 190 resource pointer. This fix changes the loop to terminate with an error on  
 191 an invalid end-of-buffer condition. The problem can be seen as an  
 192 infinite loop by callers to AcpiSetCurrentResources with an invalid or  
 193 corrupted resource descriptor, or a resource descriptor that is missing  
 194 an END\_TAG descriptor. Reported by Dan Carpenter  
 195 <dan.carpenter@oracle.com>. Lv Zheng, Bob Moore.

197 Table unload and ACPICA termination: Delete all attached data objects  
 198 during namespace node deletion. This fix updates namespace node deletion  
 199 to delete the entire list of attached objects (attached via  
 200 AcpiAttachObject) instead of just one of the attached items. ACPICA BZ  
 201 1024. Tomasz Nowicki (tomasz.nowicki@linaro.org).

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

208 Debug output: Do not emit the function nesting level for the in-kernel  
 209 build. The nesting level is really only useful during a single-thread  
 210 execution. Therefore, only enable this output for the AcpiExec utility.  
 211 Also, only emit the thread ID when executing under AcpiExec (Context  
 212 switches are still always detected and a message is emitted). ACPICA BZ  
 213 972.

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

220 Current Release:  
 221 Non-Debug Version: 95.9K Code, 27.0K Data, 122.9K Total  
 222 Debug Version: 185.1K Code, 77.2K Data, 262.3K Total  
 223 Previous Release:  
 224 Non-Debug Version: 95.8K Code, 27.0K Data, 122.8K Total  
 225 Debug Version: 185.2K Code, 77.2K Data, 262.4K Total

## 228 2) iASL Compiler/Disassembler and Tools:

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

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

241 Unix Makefiles: Added an option to disable compiler optimizations and the  
 242 \_FORTIFY\_SOURCE flag. Some older compilers have problems compiling ACPICA  
 243 with optimizations (reportedly, gcc 4.4 for example). This change adds a  
 244 command line option for make (NOOPT) that disables all compiler  
 245 optimizations and the \_FORTIFY\_SOURCE compiler flag. The default  
 246 optimization is -O2 with the \_FORTIFY\_SOURCE flag specified. ACPICA BZ  
 247 1034. Lv Zheng, Bob Moore.

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

```

254 -----
255 27 September 2013. Summary of changes for version 20130927:

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

260 1) ACPICA kernel-resident subsystem:

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

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

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

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

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

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

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

317 Replaced __FUNCTION__ with __func__ in the gcc-specific header.

```

```

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

324 Current Release:
325 Non-Debug Version: 95.8K Code, 27.0K Data, 122.8K Total
326 Debug Version: 185.2K Code, 77.2K Data, 262.4K Total
327 Previous Release:
328 Non-Debug Version: 96.7K Code, 27.1K Data, 123.9K Total
329 Debug Version: 184.4K Code, 76.8K Data, 261.2K Total

332 2) iASL Compiler/Disassembler and Tools:

334 iASL: Implemented wildcard support for the -e option. This simplifies use
335 when there are many SSDTs that must be included to resolve external
336 method
337 declarations. ACPICA BZ 1041. Example:
338 iasl -e ssdt*.dat -d dsdt.dat

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

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

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

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

361 -----
362 23 August 2013. Summary of changes for version 20130823:

364 1) ACPICA kernel-resident subsystem:

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

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

384 AcpiRead: On error, do not modify the return value target location. If an

```

385 error happens in the middle of a split 32/32 64-bit I/O operation, do not  
 386 modify the target of the return value pointer. Makes the code consistent  
 387 with the rest of ACPICA. Bjorn Helgaas.

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

394 Current Release:  
 395 Non-Debug Version: 96.7K Code, 27.1K Data, 123.9K Total  
 396 Debug Version: 184.4K Code, 76.8K Data, 261.2K Total  
 397 Previous Release:  
 398 Non-Debug Version: 96.2K Code, 27.1K Data, 123.3K Total  
 399 Debug Version: 185.4K Code, 77.1K Data, 262.5K Total

#### 402 2) iASL Compiler/Disassembler and Tools:

404 AcpiDump: Implemented several new features and fixed some problems:

- 405 1) Added support to dump the RSDP, RSDT, and XSDT tables.
- 406 2) Added support for multiple table instances (SSDT, UEFI).
- 407 3) Added option to dump "customized" (overridden) tables (-c).
- 408 4) Fixed a problem where some table filenames were improperly  
 409 constructed.
- 410 5) Improved some error messages, removed some unnecessary messages.

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

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

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

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

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

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

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

445 -----

446 25 December 2013. Summary of changes for version 20130725:

#### 448 1) ACPICA kernel-resident subsystem:

450 Fixed a problem with the DerefOf operator where references to FieldUnits

451 and BufferFields incorrectly returned the parent object, not the actual  
 452 value of the object. After this change, a dereference of a FieldUnit  
 453 reference results in a read operation on the field to get the value, and  
 454 likewise, the appropriate BufferField value is extracted from the target  
 455 buffer.

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

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

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

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

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

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

489 Current Release:  
 490 Non-Debug Version: 96.2K Code, 27.1K Data, 123.3K Total  
 491 Debug Version: 184.4K Code, 76.8K Data, 261.2K Total  
 492 Previous Release:  
 493 Non-Debug Version: 95.9K Code, 26.9K Data, 122.8K Total  
 494 Debug Version: 184.1K Code, 76.7K Data, 260.8K Total

#### 497 2) iASL Compiler/Disassembler and Tools:

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

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

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



```

517 -----
518 26 June 2013. Summary of changes for version 20130626:

520 1) ACPICA kernel-resident subsystem:

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

525 Removed an arbitrary restriction of 256 GPEs per GPE block (such as the
526 FADT-defined GPE0 and GPE1). For GPE0, GPE1, and each GPE Block Device,
527 the maximum number of GPEs is 1016. Use of multiple GPE block devices
528 makes the system-wide number of GPEs essentially unlimited.

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

535 Current Release:
536 Non-Debug Version: 95.9K Code, 26.9K Data, 122.8K Total
537 Debug Version: 184.1K Code, 76.7K Data, 260.8K Total
538 Previous Release:
539 Non-Debug Version: 96.0K Code, 27.0K Data, 123.0K Total
540 Debug Version: 184.1K Code, 76.8K Data, 260.9K Total

543 2) iASL Compiler/Disassembler and Tools:

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

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

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

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

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

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

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

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

573 -----
574 17 May 2013. Summary of changes for version 20130517:

576 1) ACPICA kernel-resident subsystem:

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

```

```

584 Fixed a possible problem with the new extended sleep registers in the
585 ACPI
586 5.0 FADT. Do not use these registers (even if populated) unless the HW-
587 reduced bit is set in the FADT (as per the ACPI specification). ACPICA BZ
588 1020. Lv Zheng.

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

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

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

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

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

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

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

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

637 2) iASL Compiler/Disassembler and Tools:

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

```

```

649 3) Other hosts - If an OS-dependent module is submitted, it will be
650 distributed with ACPICA.

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

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

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

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

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

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

681 -----
682 18 April 2013. Summary of changes for version 20130418:

684 1) ACPICA kernel-resident subsystem:

686 Fixed a possible buffer overrun during some rare but specific field unit
687 read operations. This overrun can only happen if the DSDT version is 1 --
688 meaning that all AML integers are 32 bits -- and the field length is
689 between 33 and 55 bits long. During the read, an internal buffer object
690 is
691 created for the field unit because the field is larger than an integer
692 (32
693 bits). However, in this case, the buffer will be incorrectly written
694 beyond the end because the buffer length is less than the internal
695 minimum
696 of 64 bits (8 bytes) long. The buffer will be either 5, 6, or 7 bytes
697 long, but a full 8 bytes will be written.

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

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

```

```

715 nsarguments.c

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

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

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

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

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

744 2) iASL Compiler/Disassembler and Tools:

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

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

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

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

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

770 -----
771 28 March 2013. Summary of changes for version 20130328:

773 1) ACPICA kernel-resident subsystem:

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

```

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

788 \_OSI Support: handle any errors from AcpiOsAcquireMutex. Check for error  
 789 return from this interface. Handles a possible timeout case if  
 790 ACPI\_WAIT\_FOREVER is modified by the host to be a value less than  
 791 "forever". Jung-uk Kim.

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

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

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

#### 815 2) iASL Compiler/Disassembler and Tools:

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

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

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

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

848 -----  
 849 14 February 2013. Summary of changes for version 20130214:

#### 851 1) ACPICA Kernel-resident Subsystem:

853 Fixed a possible regression on some hosts: Reinstated the safe return  
 854 macros (return ACPI\_STATUS, etc.) that ensure that the argument is  
 855 evaluated only once. Although these macros are not needed for the ACPICA  
 856 code itself, they are often used by ACPI-related host device drivers  
 857 where  
 858 the safe feature may be necessary.

860 Fixed several issues related to the ACPI 5.0 reduced hardware support  
 861 (SOC): Now ensure that if the platform declares itself as hardware-  
 862 reduced  
 863 via the FADT, the following functions become NOOPs (and always return  
 864 AE\_OK) because ACPI is always enabled by definition on these machines:  
 865 AcpiEnable  
 866 AcpiDisable  
 867 AcpiHwGetMode  
 868 AcpiHwSetMode

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

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

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

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

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

#### 908 2) iASL Compiler/Disassembler and Tools:

910 iASL/Disassembler: Fixed several issues with the definition of the ACPI  
 911 5.0 RASF table (RAS Feature Table). This change incorporates late changes  
 912 that were made to the ACPI 5.0 specification.

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

920 iASL: Implemented compile-time validation of package objects returned by  
 921 predefined names. This new feature validates static package objects  
 922 returned by the various predefined names defined to return packages. Both  
 923 object types and package lengths are validated, for both parent packages  
 924 and sub-packages, if any. The code is similar in structure and behavior  
 925 to  
 926 the runtime repair mechanism within the AML interpreter and uses the  
 927 existing predefined name information table. Adds a new file, aslprepkg.c.  
 928 ACPICA BZ 938.

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

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

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

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

949 -----  
 950 17 January 2013. Summary of changes for version 20130117:

#### 952 1) ACPICA Kernel-resident Subsystem:

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

963 Fixed two issues with the ACPI\_DEBUG\_PRINT macros:  
 964 1) Added the ACPI\_DO\_WHILE macro to the main DEBUG\_PRINT helper macro for  
 965 C compilers that require this support.  
 966 2) Renamed the internal ACPI\_DEBUG macro to ACPI\_DO\_DEBUG\_PRINT since  
 967 ACPI\_DEBUG is already used by many of the various hosts.

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

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

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

#### 987 2) iASL Compiler/Disassembler and Tools:

989 Generic Unix OSL: Use a buffer to eliminate multiple vfprintf()s and  
 990 prevent a possible fault on some hosts. Some C libraries modify the arg  
 991 pointer parameter to vfprintf making it difficult to call it twice in the  
 992 AcpiOsVprintf function. Use a local buffer to workaround this issue. This  
 993 does not affect the Windows OSL since the Win C library does not modify  
 994 the arg pointer. Chao Guan, Bob Moore.

996 iASL: Fixed a possible infinite loop when the maximum error count is  
 997 reached. If an output file other than the .AML file is specified (such as  
 998 a listing file), and the maximum number of errors is reached, do not  
 999 attempt to flush data to the output file(s) as the compiler is aborting.  
 1000 This can cause an infinite loop as the max error count code essentially  
 1001 keeps calling itself.

1003 iASL/Disassembler: Added an option (-in) to ignore NOOP  
 1004 opcodes/operators.

1005 Implemented for both the compiler and the disassembler. Often, the NOOP  
 1006 opcode is used as padding for packages that are changed dynamically by  
 1007 the  
 1008 BIOS. When disassembled and recompiled, these NOOPs will cause syntax  
 1009 errors. This option causes the disassembler to ignore all NOOP opcodes  
 1010 (0xA3), and it also causes the compiler to ignore all ASL source code  
 1011 NOOP  
 1012 statements as well.

1014 Debugger: Enhanced the Sleep command to execute all sleep states. This  
 1015 change allows Sleep to be invoked with no arguments and causes the  
 1016 debugger to execute all of the sleep states, 0-5, automatically.

1018 -----  
 1019 20 December 2012. Summary of changes for version 20121220:

#### 1021 1) ACPICA Kernel-resident Subsystem:

1023 Implemented a new interface, AcpiWalkResourceBuffer. This interface is an  
 1024 alternate entry point for AcpiWalkResources and improves the usability of  
 1025 the resource manager by accepting as input a buffer containing the output  
 1026 of either a \_CRS, \_PRS, or \_AEI method. The key functionality is that the  
 1027 input buffer is not deleted by this interface so that it can be used by  
 1028 the host later. See the ACPICA reference for details.

1030 Interpreter: Add a warning if a 64-bit constant appears in a 32-bit table  
 1031 (DSDT version < 2). The constant will be truncated and this warning  
 1032 reflects that behavior.

1034 Resource Manager: Add support for the new ACPI 5.0 wake bit in the IRQ,  
 1035 ExtendedInterrupt, and GpioInt descriptors. This change adds support to  
 1036 both get and set the new wake bit in these descriptors, separately from  
 1037 the existing share bit. Reported by Aaron Lu.

1039 Interpreter: Fix Store() when an implicit conversion is not possible. For  
 1040 example, in the cases such as a store of a string to an existing package  
 1041 object, implement the store as a CopyObject(). This is a small departure  
 1042 from the ACPI specification which states that the control method should  
 1043 be  
 1044 aborted in this case. However, the ASLTS suite depends on this behavior.

1046 Performance improvement for the various FUNCTION\_TRACE and DEBUG\_PRINT  
 1047 macros: check if debug output is currently enabled as soon as possible to  
 1048 minimize performance impact if debug is in fact not enabled.

1050 Source code restructuring: Cleanup to improve modularity. The following  
 1051 new files have been added: dbconvert.c, evhandler.c, nsprepkg.c,  
 1052 psopinfo.c, psoobject.c, rsdumpinfo.c, utstring.c, and utownid.c.  
 1053 Associated makefiles and project files have been updated.

1055 Changed an exception code for LoadTable operator. For the case where one  
 1056 of the input strings is too long, change the returned exception code from  
 1057 AE\_BAD\_PARAMETER to AE\_AML\_STRING\_LIMIT.

1059 Fixed a possible memory leak in dispatcher error path. On error, delete  
 1060 the mutex object created during method mutex creation. Reported by  
 1061 tim.gardner@canonical.com.

1063 Example Code and Data Size: These are the sizes for the OS-independent  
 1064 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
 1065 debug version of the code includes the debug output trace mechanism and  
 1066 has a much larger code and data size.

1068	Previous Release:
1069	Non-Debug Version: 94.3K Code, 25.3K Data, 119.6K Total
1070	Debug Version: 175.5K Code, 74.5K Data, 250.0K Total
1071	Current Release:
1072	Non-Debug Version: 94.5K Code, 25.5K Data, 120.0K Total
1073	Debug Version: 182.2K Code, 74.9K Data, 257.1K Total

#### 1076 2) iASL Compiler/Disassembler and Tools:

1078 iASL: Disallow a method call as argument to the ObjectType ASL operator.  
 1079 This change tracks an errata to the ACPI 5.0 document. The AML grammar  
 1080 will not allow the interpreter to differentiate between a method and a  
 1081 method invocation when these are used as an argument to the ObjectType  
 1082 operator. The ACPI specification change is to disallow a method  
 1083 invocation  
 1084 (UserTerm) for the ObjectType operator.

1086 Finish support for the TPM2 and CSRT tables in the headers, table  
 1087 compiler, and disassembler.

1089 Unix user-space OSL: Fix a problem with WaitSemaphore where the timeout  
 1090 always expires immediately if the semaphore is not available. The  
 1091 original  
 1092 code was using a relative-time timeout, but sem\_timedwait requires the  
 1093 use  
 1094 of an absolute time.

1096 iASL: Added a remark if the Timer() operator is used within a 32-bit  
 1097 table. This operator returns a 64-bit time value that will be truncated  
 1098 within a 32-bit table.

1100 iASL Source code restructuring: Cleanup to improve modularity. The  
 1101 following new files have been added: aslhex.c, aslxref.c, aslnamesp.c,  
 1102 aslmethod.c, and aslfileio.c. Associated makefiles and project files have  
 1103 been updated.

1106 -----  
 1107 14 November 2012. Summary of changes for version 20121114:

#### 1109 1) ACPICA Kernel-resident Subsystem:

1111 Implemented a performance enhancement for ACPI/AML Package objects. This  
 1112 change greatly increases the performance of Package objects within the  
 1113 interpreter. It changes the processing of reference counts for packages  
 1114 by  
 1115 optimizing for the most common case where the package sub-objects are  
 1116 either Integers, Strings, or Buffers. Increases the overall performance  
 1117 of  
 1118 the ASLTS test suite by 1.5X (Increases the Slack Mode performance by  
 1119 2X.)  
 1120 Chao Guan. ACPICA BZ 943.

1122 Implemented and deployed common macros to extract flag bits from resource  
 1123 descriptors. Improves readability and maintainability of the code. Fixes  
 1124 a  
 1125 problem with the UART serial bus descriptor for the number of data bits  
 1126 flags (was incorrectly 2 bits, should be 3).

1128 Enhanced the ACPI\_GETx and ACPI\_SETx macros. Improved the implementation  
 1129 of the macros and changed the SETx macros to the style of (destination,  
 1130 source). Also added ACPI\_CASTx companion macros. Lv Zheng.

1132 Example Code and Data Size: These are the sizes for the OS-independent  
 1133 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
 1134 debug version of the code includes the debug output trace mechanism and  
 1135 has a much larger code and data size.

1137	Previous Release:
1138	Non-Debug Version: 93.9K Code, 25.2K Data, 119.1K Total
1139	Debug Version: 175.5K Code, 74.5K Data, 250.0K Total
1140	Current Release:
1141	Non-Debug Version: 94.3K Code, 25.3K Data, 119.6K Total
1142	Debug Version: 175.5K Code, 74.5K Data, 250.0K Total

#### 1145 2) iASL Compiler/Disassembler and Tools:

1147 Disassembler: Added the new ACPI 5.0 interrupt sharing flags. This change  
 1148 adds the ShareAndWake and ExclusiveAndWake flags which were added to the  
 1149 Irq, Interrupt, and Gpio resource descriptors in ACPI 5.0. ACPICA BZ 986.

1151 Disassembler: Fixed a problem with external declaration generation. Fixes  
 1152 a problem where an incorrect pathname could be generated for an external  
 1153 declaration if the original reference to the object includes leading  
 1154 carats (^). ACPICA BZ 984.

1156 Debugger: Completed a major update for the Disassemble<method> command.  
 1157 This command was out-of-date and did not properly disassemble control  
 1158 methods that had any reasonable complexity. This fix brings the command  
 1159 up  
 1160 to the same level as the rest of the disassembler. Adds one new file,  
 1161 dmdeferred.c, which is existing code that is now common with the main  
 1162 disassembler and the debugger disassemble command. ACPICA MZ 978.

1164 iASL: Moved the parser entry prototype to avoid a duplicate declaration.  
 1165 Newer versions of Bison emit this prototype, so moved the prototype out  
 1166 of  
 1167 the iASL header to where it is actually used in order to avoid a  
 1168 duplicate  
 1169 declaration.

#### 1171 iASL/Tools: Standardized use of the stream I/O functions:

1172 1) Ensure check for I/O error after every fopen/fread/fwrite  
 1173 2) Ensure proper order of size/count arguments for fread/fwrite  
 1174 3) Use test of (Actual != Requested) after all fwrite, and most fread  
 1175 4) Standardize I/O error messages  
 1176 Improves reliability and maintainability of the code. Bob Moore, Lv

1177 Zheng.  
 1178 ACPICA BZ 981.

1180 Disassembler: Prevent duplicate External() statements. During generation  
 1181 of external statements, detect similar pathnames that are actually  
 1182 duplicates such as these:  
 1183     External (\ABCD)  
 1184     External (ABCD)  
 1185 Remove all leading '\' characters from pathnames during the external  
 1186 statement generation so that duplicates will be detected and tossed.  
 1187 ACPICA BZ 985.

1189 Tools: Replace low-level I/O with stream I/O functions. Replace  
 1190 open/read/write/close with the stream I/O equivalents  
 1191 fopen/fread/fwrite/fclose for portability and performance. Lv Zheng, Bob  
 1192 Moore.

1194 AcpiBin: Fix for the dump-to-hex function. Now correctly output the table  
 1195 name header so that AcpiXtract recognizes the output file/table.

1197 iASL: Remove obsolete -2 option flag. Originally intended to force the  
 1198 compiler/disassembler into an ACPI 2.0 mode, this was never implemented  
 1199 and the entire concept is now obsolete.

1201 -----  
 1202 18 October 2012. Summary of changes for version 20121018:

1205 1) ACPICA Kernel-resident Subsystem:

1207 Updated support for the ACPI 5.0 MPST table. Fixes some problems  
 1208 introduced by late changes to the table as it was added to the ACPI 5.0  
 1209 specification. Includes header, disassembler, and data table compiler  
 1210 support as well as a new version of the MPST template.

1212 AcpiGetObjectInfo: Enhanced the device object support to include the ACPI  
 1213 5.0 \_SUB method. Now calls \_SUB in addition to the other PNP-related ID  
 1214 methods: \_HID, \_CID, and \_UID.

1216 Changed ACPI\_DEVICE\_ID to ACPI\_PNP\_DEVICE\_ID. Also changed  
 1217 ACPI\_DEVICE\_ID\_LIST to ACPI\_PNP\_DEVICE\_ID\_LIST. These changes prevent  
 1218 name collisions on hosts that reserve the \*\_DEVICE\_ID (or \*DeviceId)  
 1219 names for their various drivers. Affects the AcpiGetObjectInfo external  
 1220 interface, and other internal interfaces as well.

1222 Added and deployed a new macro for ACPI\_NAME management: ACPI\_MOVE\_NAME.  
 1223 This macro resolves to a simple 32-bit move of the 4-character ACPI\_NAME  
 1224 on machines that support non-aligned transfers. Optimizes for this case  
 1225 rather than using a strncpy. With assistance from Zheng Lv.

1227 Resource Manager: Small fix for buffer size calculation. Fixed a one byte  
 1228 error in the output buffer calculation. Feng Tang. ACPICA BZ 849.

1230 Added a new debug print message for AML mutex objects that are force-  
 1231 released. At control method termination, any currently acquired mutex  
 1232 objects are force-released. Adds a new debug-only message for each one  
 1233 that is released.

1235 Audited/updated all ACPICA return macros and the function debug depth  
 1236 counter: 1) Ensure that all functions that use the various TRACE macros  
 1237 also use the appropriate ACPICA return macros. 2) Ensure that all normal  
 1238 return statements surround the return expression (value) with parens to  
 1239 ensure consistency across the ACPICA code base. Guan Chao, Tang Feng,  
 1240 Zheng Lv, Bob Moore. ACPICA Bugzilla 972.

1242 Global source code changes/maintenance: All extra lines at the start and

1243 end of each source file have been removed for consistency. Also, within  
 1244 comments, all new sentences start with a single space instead of a double  
 1245 space, again for consistency across the code base.

1247 Example Code and Data Size: These are the sizes for the OS-independent  
 1248 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
 1249 debug version of the code includes the debug output trace mechanism and  
 1250 has a much larger code and data size.

1252 Previous Release:  
 1253 Non-Debug Version: 93.7K Code, 25.3K Data, 119.0K Total  
 1254 Debug Version: 175.0K Code, 74.4K Data, 249.4K Total  
 1255 Current Release:  
 1256 Non-Debug Version: 93.9K Code, 25.2K Data, 119.1K Total  
 1257 Debug Version: 175.5K Code, 74.5K Data, 250.0K Total

1260 2) iASL Compiler/Disassembler and Tools:

1262 AcpiExec: Improved the algorithm used for memory leak/corruption  
 1263 detection. Added some intelligence to the code that maintains the global  
 1264 list of allocated memory. The list is now ordered by allocated memory  
 1265 address, significantly improving performance. When running AcpiExec on  
 1266 the ASLTS test suite, speed improvements of 3X to 5X are seen, depending  
 1267 on the platform and/or the environment. Note, this performance  
 1268 enhancement affects the AcpiExec utility only, not the kernel-resident  
 1269 ACPICA code.

1271 Enhanced error reporting for invalid AML opcodes and bad ACPI\_NAMES. For  
 1272 the disassembler, dump the 48 bytes surrounding the invalid opcode. Fix  
 1273 incorrect table offset reported for invalid opcodes. Report the original  
 1274 32-bit value for bad ACPI\_NAMES (as well as the repaired name.)

1276 Disassembler: Enhanced the -vt option to emit the binary table data in  
 1277 hex format to assist with debugging.

1279 Fixed a potential filename buffer overflow in osunixdir.c. Increased the  
 1280 size of file structure. Colin Ian King.

1282 -----  
 1283 13 September 2012. Summary of changes for version 20120913:

1286 1) ACPICA Kernel-resident Subsystem:

1288 ACPI 5.0: Added two new notify types for the Hardware Error Notification  
 1289 Structure within the Hardware Error Source Table (HEST) table -- CMCI(5)  
 1290 and  
 1291 MCE(6).  
 1292  
 1293 Table Manager: Merged/removed duplicate code in the root table resize  
 1294 functions. One function is external, the other is internal. Lv Zheng,  
 1295 ACPICA  
 1296 BZ 846.

1298 Makefiles: Completely removed the obsolete "Linux" makefiles under  
 1299 acpica/generate/linux. These makefiles are obsolete and have been  
 1300 replaced  
 1301 by  
 1302 the generic unix makefiles under acpica/generate/unix.

1304 Makefiles: Ensure that binary files always copied properly. Minor rule  
 1305 change  
 1306 to ensure that the final binary output files are always copied up to the  
 1307 appropriate binary directory (bin32 or bin64.)

1309 Example Code and Data Size: These are the sizes for the OS-independent  
 1310 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
 1311 debug  
 1312 version of the code includes the debug output trace mechanism and has a  
 1313 much  
 1314 larger code and data size.

1316 Previous Release:  
 1317 Non-Debug Version: 93.8K Code, 25.3K Data, 119.1K Total  
 1318 Debug Version: 175.7K Code, 74.8K Data, 250.5K Total  
 1319 Current Release:  
 1320 Non-Debug Version: 93.7K Code, 25.3K Data, 119.0K Total  
 1321 Debug Version: 175.0K Code, 74.4K Data, 249.4K Total

#### 1324 2) iASL Compiler/Disassembler and Tools:

1326 Disassembler: Fixed a possible fault during the disassembly of resource  
 1327 descriptors when a second parse is required because of the invocation of  
 1328 external control methods within the table. With assistance from  
 1329 adq@lidskialf.net. ACPICA BZ 976.

1331 iASL: Fixed a namepath optimization problem. An error can occur if the  
 1332 parse  
 1333 node that contains the namepath to be optimized does not have a parent  
 1334 node  
 1335 that is a named object. This change fixes the problem.

1337 iASL: Fixed a regression where the AML file is not deleted on errors. The  
 1338 AML  
 1339 output file should be deleted if there are any errors during the  
 1340 compiler.  
 1341 The  
 1342 only exception is if the -f (force output) option is used. ACPICA BZ 974.

1344 iASL: Added a feature to automatically increase internal line buffer  
 1345 sizes.  
 1346 Via realloc(), automatically increase the internal line buffer sizes as  
 1347 necessary to support very long source code lines. The current version of  
 1348 the  
 1349 preprocessor requires a buffer long enough to contain full source code  
 1350 lines.  
 1351 This change increases the line buffer(s) if the input lines go beyond the  
 1352 current buffer size. This eliminates errors that occurred when a source  
 1353 code  
 1354 line was longer than the buffer.

1356 iASL: Fixed a problem with constant folding in method declarations. The  
 1357 SyncLevel term is a ByteConstExpr, and incorrect code would be generated  
 1358 if a  
 1359 Type3 opcode was used.

1361 Debugger: Improved command help support. For incorrect argument count,  
 1362 display  
 1363 full help for the command. For help command itself, allow an argument to  
 1364 specify a command.

1366 Test Suites: Several bug fixes for the ASLTS suite reduces the number of  
 1367 errors during execution of the suite. Guan Chao.

1369 -----  
 1370 16 August 2012. Summary of changes for version 20120816:

#### 1373 1) ACPICA Kernel-resident Subsystem:

1375 Removed all use of the deprecated \_GTS and \_BFS predefined methods. The  
 1376 \_GTS  
 1377 (Going To Sleep) and \_BFS (Back From Sleep) methods are essentially  
 1378 deprecated and will probably be removed from the ACPI specification.  
 1379 Windows  
 1380 does not invoke them, and reportedly never will. The final nail in the  
 1381 coffin  
 1382 is that the ACPI specification states that these methods must be run with  
 1383 interrupts off, which is not going to happen in a kernel interpreter.  
 1384 Note:  
 1385 Linux has removed all use of the methods also. It was discovered that  
 1386 invoking these functions caused failures on some machines, probably  
 1387 because  
 1388 they were never tested since Windows does not call them. Affects two  
 1389 external  
 1390 interfaces, AcpiEnterSleepState and AcpiLeaveSleepStatePrep. Tang Feng.  
 1391 ACPICA BZ 969.

1393 Implemented support for complex bit-packed buffers returned from the \_PLD  
 1394 (Physical Location of Device) predefined method. Adds a new external  
 1395 interface, AcpiDecodePldBuffer that parses the buffer into a more usable  
 1396 C  
 1397 structure. Note: C Bitfields cannot be used for this type of predefined  
 1398 structure since the memory layout of individual bitfields is not defined  
 1399 by  
 1400 the C language. In addition, there are endian concerns where a compiler  
 1401 will  
 1402 change the bitfield ordering based on the machine type. The new ACPICA  
 1403 interface eliminates these issues, and should be called after \_PLD is  
 1404 executed. ACPICA BZ 954.

1406 Implemented a change to allow a scope change to root (via "Scope (\)")  
 1407 during  
 1408 execution of module-level ASL code (code that is executed at table load  
 1409 time.) Lin Ming.

1411 Added the Windows8/Server2012 string for the \_OSI method. This change  
 1412 adds  
 1413 a  
 1414 new \_OSI string, "Windows 2012" for both Windows 8 and Windows Server  
 1415 2012.

1417 Added header support for the new ACPI tables DBG2 (Debug Port Table Type  
 1418 2)  
 1419 and CSRT (Core System Resource Table).

1421 Added struct header support for the \_FDE, \_GRT, \_GTM, and \_SRT predefined  
 1422 names. This simplifies access to the buffers returned by these predefined  
 1423 names. Adds a new file, include/acbuffer.h. ACPICA BZ 956.

1425 GPE support: Removed an extraneous parameter from the various low-level  
 1426 internal GPE functions. Tang Feng.

1428 Removed the linux makefiles from the unix packages. The generate/linux  
 1429 makefiles are obsolete and have been removed from the unix tarball  
 1430 release  
 1431 packages. The replacement makefiles are under generate/unix, and there is  
 1432 a  
 1433 top-level makefile under the main acpica directory. ACPICA BZ 967, 912.

#### 1435 Updates for Unix makefiles:

1436 1) Add -D FORTIFY\_SOURCE=2 for gcc generation. Arjan van de Ven.  
 1437 2) Update linker flags (move to end of command line) for AcpiExec  
 1438 utility.  
 1439 Guan Chao.

1441 Split ACPICA initialization functions to new file, utxfinit.c. Split from  
1442 utxface.c to improve modularity and reduce file size.

1444 Example Code and Data Size: These are the sizes for the OS-independent  
1445 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
1446 debug version of the code includes the debug output trace mechanism and  
1447 has a  
1448 much larger code and data size.

1450 Previous Release:  
1451 Non-Debug Version: 93.5K Code, 25.3K Data, 118.8K Total  
1452 Debug Version: 173.7K Code, 74.0K Data, 247.7K Total  
1453 Current Release:  
1454 Non-Debug Version: 93.8K Code, 25.3K Data, 119.1K Total  
1455 Debug Version: 175.7K Code, 74.8K Data, 250.5K Total

#### 1458 2) iASL Compiler/Disassembler and Tools:

1460 iASL: Fixed a problem with constant folding for fixed-length constant  
1461 expressions. The constant-folding code was not being invoked for constant  
1462 expressions that allow the use of type 3/4/5 opcodes to generate  
1463 constants  
1464 for expressions such as ByteConstExpr, WordConstExpr, etc. This could  
1465 result  
1466 in the generation of invalid AML bytecode. ACPICA BZ 970.

1468 iASL: Fixed a generation issue on newer versions of Bison. Newer versions  
1469 apparently automatically emit some of the necessary externals. This  
1470 change  
1471 handles these versions in order to eliminate generation warnings.

1473 Disassembler: Added support to decode the DBG2 and CSRT ACPI tables.

1475 Disassembler: Add support to decode \_PLD buffers. The decoded buffer  
1476 appears  
1477 within comments in the output file.

1479 Debugger: Fixed a regression with the "Threads" command where  
1480 AE\_BAD\_PARAMETER was always returned.

1482 -----  
1483 11 July 2012. Summary of changes for version 20120711:

#### 1485 1) ACPICA Kernel-resident Subsystem:

1487 Fixed a possible fault in the return package object repair code. Fixes a  
1488 problem that can occur when a lone package object is wrapped with an  
1489 outer  
1490 package object in order to force conformance to the ACPI specification.  
1491 Can  
1492 affect these predefined names: \_ALR, \_MLS, \_PSS, \_TRT, \_TSS, \_PRT, \_HPX,  
1493 \_DLM,  
1494 \_CSD, \_PSD, \_TSD.

1496 Removed code to disable/enable bus master arbitration (ARB\_DIS bit in the  
1497 PM2\_CNT register) in the ACPICA sleep/wake interfaces. Management of the  
1498 ARB\_DIS bit must be implemented in the host-dependent C3 processor power  
1499 state  
1500 support. Note, ARB\_DIS is obsolete and only applies to older chipsets,  
1501 both  
1502 Intel and other vendors. (for Intel: ICH4-M and earlier)

1504 This change removes the code to disable/enable bus master arbitration  
1505 during  
1506 suspend/resume. Use of the ARB\_DIS bit in the optional PM2\_CNT register

1507 causes  
1508 resume problems on some machines. The change has been in use for over  
1509 seven  
1510 years within Linux.

1512 Implemented two new external interfaces to support host-directed dynamic  
1513 ACPI  
1514 table load and unload. They are intended to simplify the host  
1515 implementation  
1516 of hot-plug support:  
1517 AcpiLoadTable: Load an SSDT from a buffer into the namespace.  
1518 AcpiUnloadParentTable: Unload an SSDT via a named object owned by the  
1519 table.  
1520 See the ACPICA reference for additional details. Adds one new file,  
1521 components/tables/tbxload.c

1523 Implemented and deployed two new interfaces for errors and warnings that  
1524 are  
1525 known to be caused by BIOS/firmware issues:  
1526 AcpiBiosError: Prints "ACPI Firmware Error" message.  
1527 AcpiBiosWarning: Prints "ACPI Firmware Warning" message.  
1528 Deployed these new interfaces in the ACPICA Table Manager code for ACPI  
1529 table  
1530 and FADT errors. Additional deployment to be completed as appropriate in  
1531 the  
1532 future. The associated conditional macros are ACPI\_BIOS\_ERROR and  
1533 ACPI\_BIOS\_WARNING. See the ACPICA reference for additional details.  
1534 ACPICA  
1535 BZ  
1536 843.

1538 Implicit notify support: ensure that no memory allocation occurs within a  
1539 critical region. This fix moves a memory allocation outside of the time  
1540 that a  
1541 spinlock is held. Fixes issues on systems that do not allow this  
1542 behavior.  
1543 Jung-uk Kim.

1545 Split exception code utilities and tables into a new file,  
1546 utilities/utexcept.c

1548 Example Code and Data Size: These are the sizes for the OS-independent  
1549 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
1550 debug  
1551 version of the code includes the debug output trace mechanism and has a  
1552 much  
1553 larger code and data size.

1555 Previous Release:  
1556 Non-Debug Version: 93.1K Code, 25.1K Data, 118.2K Total  
1557 Debug Version: 172.9K Code, 73.6K Data, 246.5K Total  
1558 Current Release:  
1559 Non-Debug Version: 93.5K Code, 25.3K Data, 118.8K Total  
1560 Debug Version: 173.7K Code, 74.0K Data, 247.7K Total

#### 1563 2) iASL Compiler/Disassembler and Tools:

1565 iASL: Fixed a parser problem for hosts where EOF is defined as -1 instead  
1566 of  
1567 0. Jung-uk Kim.

1569 Debugger: Enhanced the "tables" command to emit additional information  
1570 about  
1571 the current set of ACPI tables, including the owner ID and flags decode.



1573 Debugger: Reimplemented the "unload" command to use the new  
 1574 AcpiUnloadParentTable external interface. This command was disable  
 1575 previously  
 1576 due to need for an unload interface.

1578 AcpiHelp: Added a new option to decode ACPICA exception codes. The -e  
 1579 option  
 1580 will decode 16-bit hex status codes (ACPI\_STATUS) to name strings.

1582 -----  
 1583 20 June 2012. Summary of changes for version 20120620:

1586 1) ACPICA Kernel-resident Subsystem:

1588 Implemented support to expand the "implicit notify" feature to allow  
 1589 multiple  
 1590 devices to be notified by a single GPE. This feature automatically  
 1591 generates a  
 1592 runtime device notification in the absence of a BIOS-provided GPE control  
 1593 method (\_Lxx/\_Exx) or a host-installed handler for the GPE. Implicit  
 1594 notify is  
 1595 provided by ACPICA for Windows compatibility, and is a workaround for  
 1596 BIOS  
 1597 AML  
 1598 code errors. See the description of the AcpiSetupGpeForWake interface in  
 1599 the  
 1600 APCICA reference. Bob Moore, Rafael Wysocki. ACPICA BZ 918.

1602 Changed some comments and internal function names to simplify and ensure  
 1603 correctness of the Linux code translation. No functional changes.

1605 Example Code and Data Size: These are the sizes for the OS-independent  
 1606 apica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
 1607 debug  
 1608 version of the code includes the debug output trace mechanism and has a  
 1609 much  
 1610 larger code and data size.

1612 Previous Release:  
 1613 Non-Debug Version: 93.0K Code, 25.1K Data, 118.1K Total  
 1614 Debug Version: 172.7K Code, 73.6K Data, 246.3K Total  
 1615 Current Release:  
 1616 Non-Debug Version: 93.1K Code, 25.1K Data, 118.2K Total  
 1617 Debug Version: 172.9K Code, 73.6K Data, 246.5K Total

1620 2) iASL Compiler/Disassembler and Tools:

1622 Disassembler: Added support to emit short, commented descriptions for the  
 1623 ACPI  
 1624 predefined names in order to improve the readability of the disassembled  
 1625 output. ACPICA BZ 959. Changes include:  
 1626 1) Emit descriptions for all standard predefined names (\_INI, \_STA,  
 1627 \_PRW,  
 1628 etc.)  
 1629 2) Emit generic descriptions for the special names (\_Exx, \_Qxx, etc.)  
 1630 3) Emit descriptions for the resource descriptor names (\_MIN, \_LEN,  
 1631 etc.)

1633 AcpiSrc: Fixed several long-standing Linux code translation issues.  
 1634 Argument  
 1635 descriptions in function headers are now translated properly to lower  
 1636 case  
 1637 and  
 1638 underscores. ACPICA BZ 961. Also fixes translation problems such as

1639 these:  
 1640 (old -> new)  
 1641 i\_aSL -> iASL  
 1642 00-7\_f -> 00-7F  
 1643 16\_k -> 16K  
 1644 local\_fADT -> local\_FADT  
 1645 execute\_oSI -> execute\_OSI

1647 iASL: Fixed a problem where null bytes were inadvertently emitted into  
 1648 some  
 1649 listing files.

1651 iASL: Added the existing debug options to the standard help screen. There  
 1652 are  
 1653 no longer two different help screens. ACPICA BZ 957.

1655 AcpiHelp: Fixed some typos in the various predefined name descriptions.  
 1656 Also  
 1657 expand some of the descriptions where appropriate.

1659 iASL: Fixed the -ot option (display compile times/statistics). Was not  
 1660 working  
 1661 properly for standard output; only worked for the debug file case.

1663 -----  
 1664 18 May 2012. Summary of changes for version 20120518:

1667 1) ACPICA Core Subsystem:

1669 Added a new OSL interface, AcpiOsWaitEventsComplete. This interface is  
 1670 defined  
 1671 to block until asynchronous events such as notifies and GPEs have  
 1672 completed.  
 1673 Within ACPICA, it is only called before a notify or GPE handler is  
 1674 removed/uninstalled. It also may be useful for the host OS within related  
 1675 drivers such as the Embedded Controller driver. See the ACPICA reference  
 1676 for  
 1677 additional information. ACPICA BZ 868.

1679 ACPI Tables: Added a new error message for a possible overflow failure  
 1680 during  
 1681 the conversion of FADT 32-bit legacy register addresses to internal  
 1682 common  
 1683 64-  
 1684 bit GAS structure representation. The GAS has a one-byte "bit length"  
 1685 field,  
 1686 thus limiting the register length to 255 bits. ACPICA BZ 953.

1688 Example Code and Data Size: These are the sizes for the OS-independent  
 1689 apica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
 1690 debug  
 1691 version of the code includes the debug output trace mechanism and has a  
 1692 much  
 1693 larger code and data size.

1695 Previous Release:  
 1696 Non-Debug Version: 92.9K Code, 25.0K Data, 117.9K Total  
 1697 Debug Version: 172.6K Code, 73.4K Data, 246.0K Total  
 1698 Current Release:  
 1699 Non-Debug Version: 93.0K Code, 25.1K Data, 118.1K Total  
 1700 Debug Version: 172.7K Code, 73.6K Data, 246.3K Total

1703 2) iASL Compiler/Disassembler and Tools:

```

1705 iASL: Added the ACPI 5.0 "PCC" keyword for use in the Register() ASL
1706 macro.
1707 This keyword was added late in the ACPI 5.0 release cycle and was not
1708 implemented until now.

1710 Disassembler: Added support for Operation Region externals. Adds missing
1711 support for operation regions that are defined in another table, and
1712 referenced locally via a Field or BankField ASL operator. Now generates
1713 the
1714 correct External statement.

1716 Disassembler: Several additional fixes for the External() statement
1717 generation
1718 related to some ASL operators. Also, order the External() statements
1719 alphabetically in the disassembler output. Fixes the External()
1720 generation
1721 for
1722 the Create* field, Alias, and Scope operators:
1723 1) Create* buffer field operators - fix type mismatch warning on
1724 disassembly
1725 2) Alias - implement missing External support
1726 3) Scope - fix to make sure all necessary externals are emitted.

1728 iASL: Improved pathname support. For include files, merge the prefix
1729 pathname
1730 with the file pathname and eliminate unnecessary components. Convert
1731 backslashes in all pathnames to forward slashes, for readability. Include
1732 file
1733 pathname changes affect both #include and Include() type operators.

1735 iASL/DTC/Preprocessor: Gracefully handle early EOF. Handle an EOF at the
1736 end
1737 of a valid line by inserting a newline and then returning the EOF during
1738 the
1739 next call to GetNextLine. Prevents the line from being ignored due to EOF
1740 condition.

1742 iASL: Implemented some changes to enhance the IDE support (-vi option.)
1743 Error
1744 and Warning messages are now correctly recognized for both the source
1745 code
1746 browser and the global error and warning counts.

1748 -----
1749 20 April 2012. Summary of changes for version 20120420:

1752 1) ACPICA Core Subsystem:

1754 Implemented support for multiple notify handlers. This change adds
1755 support
1756 to
1757 allow multiple system and device notify handlers on Device, Thermal Zone,
1758 and
1759 Processor objects. This can simplify the host OS notification
1760 implementation.
1761 Also re-worked and restructured the entire notify support code to
1762 simplify
1763 handler installation, handler removal, notify event queuing, and notify
1764 dispatch to handler(s). Note: there can still only be two global notify
1765 handlers - one for system notifies and one for device notifies. There are
1766 no
1767 changes to the existing handler install/remove interfaces. Lin Ming, Bob
1768 Moore, Rafael Wysocki.

1770 Fixed a regression in the package repair code where the object reference

```

```

1771 count was calculated incorrectly. Regression was introduced in the commit
1772 "Support to add Package wrappers".

1774 Fixed a couple possible memory leaks in the AML parser, in the error
1775 recovery
1776 path. Jesper Juhl, Lin Ming.

1778 Example Code and Data Size: These are the sizes for the OS-independent
1779 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The
1780 debug version of the code includes the debug output trace mechanism and
1781 has a
1782 much larger code and data size.

1784 Previous Release:
1785 Non-Debug Version: 92.9K Code, 25.0K Data, 117.9K Total
1786 Debug Version: 172.5K Code, 73.2K Data, 245.7K Total
1787 Current Release:
1788 Non-Debug Version: 92.9K Code, 25.0K Data, 117.9K Total
1789 Debug Version: 172.6K Code, 73.4K Data, 246.0K Total

1792 2) iASL Compiler/Disassembler and Tools:

1794 iASL: Fixed a problem with the resource descriptor support where the
1795 length
1796 of the StartDependentFn and StartDependentFnNoPrio descriptors were not
1797 included in cumulative descriptor offset, resulting in incorrect values
1798 for
1799 resource tags within resource descriptors appearing after a
1800 StartDependent*
1801 descriptor. Reported by Petr Vandrovec. ACPICA BZ 949.

1803 iASL and Preprocessor: Implemented full support for the #line directive
1804 to
1805 correctly track original source file line numbers through the .i
1806 preprocessor
1807 output file - for error and warning messages.

1809 iASL: Expand the allowable byte constants for address space IDs.
1810 Previously,
1811 the allowable range was 0x80-0xFF (user-defined spaces), now the range is
1812 0x0A-0xFF to allow for custom and new IDs without changing the compiler.

1814 iASL: Add option to treat all warnings as errors (-we). ACPICA BZ 948.

1816 iASL: Add option to completely disable the preprocessor (-Pn).

1818 iASL: Now emit all error/warning messages to standard error (stderr) by
1819 default (instead of the previous stdout).

1821 ASL Test Suite (ASLTS): Reduce iASL warnings due to use of Switch().
1822 Update
1823 for resource descriptor offset fix above. Update/cleanup error output
1824 routines. Enable and send iASL errors/warnings to an error logfile
1825 (error.txt). Send all other iASL output to a logfile (compiler.txt).
1826 Fixed
1827 several extraneous "unrecognized operator" messages.

1829 -----
1830 20 March 2012. Summary of changes for version 20120320:

1833 1) ACPICA Core Subsystem:

1835 Enhanced the sleep/wake interfaces to optionally execute the _GTS method
1836 (Going To Sleep) and the _BFS method (Back From Sleep). Windows

```

1837 apparently  
 1838 does not execute these methods, and therefore these methods are often  
 1839 untested. It has been seen on some systems where the execution of these  
 1840 methods causes errors and also prevents the machine from entering S5. It  
 1841 is  
 1842 therefore suggested that host operating systems do not execute these  
 1843 methods  
 1844 by default. In the future, perhaps these methods can be optionally  
 1845 executed  
 1846 based on the age of the system and/or what is the newest version of  
 1847 Windows  
 1848 that the BIOS asks for via \_OSI. Changed interfaces: AcpiEnterSleepState  
 1849 and  
 1850 AcpiLeaveSleepStatePrep. See the ACPICA reference and Linux BZ 13041. Lin  
 1851 Ming.

1853 Fixed a problem where the length of the local/common FADT was set too  
 1854 early.  
 1855 The local FADT table length cannot be set to the common length until the  
 1856 original length has been examined. There is code that checks the table  
 1857 length  
 1858 and sets various fields appropriately. This can affect older machines  
 1859 with  
 1860 early FADT versions. For example, this can cause inadvertent writes to  
 1861 the  
 1862 CST\_CNT register. Julian Anastasov.

1864 Fixed a mapping issue related to a physical table override. Use the  
 1865 deferred  
 1866 mapping mechanism for tables loaded via the physical override OSL  
 1867 interface.  
 1868 This allows for early mapping before the virtual memory manager is  
 1869 available.  
 1870 Thomas Renninger, Bob Moore.

1872 Enhanced the automatic return-object repair code: Repair a common problem  
 1873 with  
 1874 predefined methods that are defined to return a variable-length Package  
 1875 of  
 1876 sub-objects. If there is only one sub-object, some BIOS ASL code  
 1877 mistakenly  
 1878 simply returns the single object instead of a Package with one sub-  
 1879 object.  
 1880 This new support will repair this error by wrapping a Package object  
 1881 around  
 1882 the original object, creating the correct and expected Package with one  
 1883 sub-  
 1884 object. Names that can be repaired in this manner include: \_ALR, \_CSD,  
 1885 \_HPX,  
 1886 \_MLS, \_PLD, \_PRT, \_PSS, \_TRT, \_TSS, \_BCL, \_DOD, \_FIX, and \_Sx. ACPICA BZ  
 1887 939.

1889 Changed the exception code returned for invalid ACPI paths passed as  
 1890 parameters to external interfaces such as AcpiEvaluateObject. Was  
 1891 AE\_BAD\_PARAMETER, now is the more sensible AE\_BAD\_PATHNAME.

1893 Example Code and Data Size: These are the sizes for the OS-independent  
 1894 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
 1895 debug  
 1896 version of the code includes the debug output trace mechanism and has a  
 1897 much  
 1898 larger code and data size.

1900 Previous Release:  
 1901 Non-Debug Version: 93.0K Code, 25.0K Data, 118.0K Total  
 1902 Debug Version: 172.5K Code, 73.2K Data, 245.7K Total

1903 Current Release:  
 1904 Non-Debug Version: 92.9K Code, 25.0K Data, 117.9K Total  
 1905 Debug Version: 172.5K Code, 73.2K Data, 245.7K Total

## 1908 2) iASL Compiler/Disassembler and Tools:

1910 iASL: Added the infrastructure and initial implementation of a integrated  
 1911 C-  
 1912 like preprocessor. This will simplify BIOS development process by  
 1913 eliminating  
 1914 the need for a separate preprocessing step during builds. On Windows, it  
 1915 also  
 1916 eliminates the need to install a separate C compiler. ACPICA BZ 761. Some  
 1917 features including full #define() macro support are still under  
 1918 development.

1919 These preprocessor directives are supported:

```
1920 #define
1921 #elif
1922 #else
1923 #endif
1924 #error
1925 #if
1926 #ifdef
1927 #ifndef
1928 #include
1929 #pragma message
1930 #undef
1931 #warning
1932 In addition, these new command line options are supported:
1933 -D <symbol> Define symbol for preprocessor use
1934 -li Create preprocessed output file (*.i)
1935 -P Preprocess only and create preprocessor output file (*.i)
```

1937 Table Compiler: Fixed a problem where the equals operator within an  
 1938 expression  
 1939 did not work properly.

1941 Updated iASL to use the current versions of Bison/Flex. Updated the  
 1942 Windows  
 1943 project file to invoke these tools from the standard location. ACPICA BZ  
 1944 904.  
 1945 Versions supported:  
 1946 Flex for Windows: V2.5.4  
 1947 Bison for Windows: V2.4.1

1949 -----  
 1950 15 February 2012. Summary of changes for version 20120215:

## 1953 1) ACPICA Core Subsystem:

1955 There have been some major changes to the sleep/wake support code, as  
 1956 described below (a - e).

1958 a) The AcpiLeaveSleepState has been split into two interfaces, similar to  
 1959 AcpiEnterSleepStatePrep and AcpiEnterSleepState. The new interface is  
 1960 AcpiLeaveSleepStatePrep. This allows the host to perform actions between  
 1961 the  
 1962 time the \_BFS method is called and the \_WAK method is called. NOTE: all  
 1963 hosts  
 1964 must update their wake/resume code or else sleep/wake will not work  
 1965 properly.  
 1966 Rafael Wysocki.

1968 b) In AcpiLeaveSleepState, now enable all runtime GPEs before calling the

1969 `_WAK`  
 1970 method. Some machines require that the GPEs are enabled before the `_WAK`  
 1971 method  
 1972 is executed. Thomas Renninger.

1974 c) In `AcpiLeaveSleepState`, now always clear the `WAK_STS` (wake status)  
 1975 bit.  
 1976 Some BIOS code assumes that `WAK_STS` will be cleared on resume and use it  
 1977 to  
 1978 determine whether the system is rebooting or resuming. Matthew Garrett.

1980 d) Move the invocations of `_GTS` (Going To Sleep) and `_BFS` (Back From  
 1981 Sleep) to  
 1982 match the ACPI specification requirement. Rafael Wysocki.

1984 e) Implemented full support for the ACPI 5.0 `SleepStatus` and `SleepControl`  
 1985 registers within the V5 FADT. This support adds two new files:  
 1986 `hardware/hwesleep.c` implements the support for the new registers. Moved  
 1987 all  
 1988 `sleep/wake` external interfaces to `hardware/hwxfsleep.c`.

1991 Added a new OSL interface for ACPI table overrides,  
 1992 `AcpiOsPhysicalTableOverride`. This interface allows the host to override a  
 1993 table via a physical address, instead of the logical address required by  
 1994 `AcpiOsTableOverride`. This simplifies the host implementation. Initial  
 1995 implementation by Thomas Renninger. The ACPICA implementation creates a  
 1996 single  
 1997 shared function for table overrides that attempts both a logical and a  
 1998 physical override.

2000 Expanded the OSL memory read/write interfaces to 64-bit data  
 2001 (`AcpiOsReadMemory`, `AcpiOsWriteMemory`.) This enables full 64-bit memory  
 2002 transfer support for GAS register structures passed to `AcpiRead` and  
 2003 `AcpiWrite`.

2005 Implemented the `ACPI_REduced_HARdWARE` option to allow the creation of a  
 2006 custom  
 2007 build of ACPICA that supports only the ACPI 5.0 reduced hardware (SoC)  
 2008 model.  
 2009 See the ACPICA reference for details. ACPICA BZ 942. This option removes  
 2010 about  
 2011 10% of the code and 5% of the static data, and the following hardware  
 2012 ACPI  
 2013 features become unavailable:  
 2014 PM Event and Control registers  
 2015 SCI interrupt (and handler)  
 2016 Fixed Events  
 2017 General Purpose Events (GPEs)  
 2018 Global Lock  
 2019 ACPI PM timer  
 2020 FACS table (Waking vectors and Global Lock)

2022 Updated the unix tarball directory structure to match the ACPICA git  
 2023 source  
 2024 tree. This ensures that the generic unix makefiles work properly (in  
 2025 `generate/unix`). Also updated the Linux makefiles to match. ACPICA BZ  
 2026 867.

2028 Updated the return value of the `_REV` predefined method to integer value 5  
 2029 to  
 2030 reflect ACPI 5.0 support.

2032 Moved the external ACPI PM timer interface prototypes to the public  
 2033 `acpixf.h`  
 2034 file where they belong.

2036 Example Code and Data Size: These are the sizes for the OS-independent  
 2037 `acpica.lib` produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
 2038 debug  
 2039 version of the code includes the debug output trace mechanism and has a  
 2040 much  
 2041 larger code and data size.

2043 Previous Release:  
 2044 Non-Debug Version: 92.8K Code, 24.9K Data, 117.7K Total  
 2045 Debug Version: 171.7K Code, 72.9K Data, 244.5K Total  
 2046 Current Release:  
 2047 Non-Debug Version: 93.0K Code, 25.0K Data, 118.0K Total  
 2048 Debug Version: 172.5K Code, 73.2K Data, 245.7K Total

2051 2) iASL Compiler/Disassembler and Tools:

2053 Disassembler: Fixed a problem with the new ACPI 5.0 serial resource  
 2054 descriptors (I2C, SPI, UART) where the resource produce/consumer bit was  
 2055 incorrectly displayed.

2057 `AcpiHelp`: Add display of ACPI/PNP device IDs that are defined in the ACPI  
 2058 specification.

2060 -----  
 2061 11 January 2012. Summary of changes for version 20120111:

2064 1) ACPICA Core Subsystem:

2066 Implemented a new mechanism to allow host device drivers to check for  
 2067 address  
 2068 range conflicts with ACPI Operation Regions. Both `SystemMemory` and  
 2069 `SystemIO`  
 2070 address spaces are supported. A new external interface,  
 2071 `AcpiCheckAddressRange`,  
 2072 allows drivers to check an address range against the ACPI namespace. See  
 2073 the  
 2074 ACPICA reference for additional details. Adds one new file,  
 2075 `utilities/utaddress.c`. Lin Ming, Bob Moore.

2077 Fixed several issues with the ACPI 5.0 FADT support: Add the sleep  
 2078 Control  
 2079 and  
 2080 Status registers, update the ACPI 5.0 flags, and update internal data  
 2081 structures to handle an FADT larger than 256 bytes. The size of the ACPI  
 2082 5.0  
 2083 FADT is 268 bytes.

2085 Updated all ACPICA copyrights and signons to 2012. Added the 2012  
 2086 copyright to  
 2087 all module headers and signons, including the standard Linux header. This  
 2088 affects virtually every file in the ACPICA core subsystem, iASL compiler,  
 2089 and  
 2090 all ACPICA utilities.

2092 Example Code and Data Size: These are the sizes for the OS-independent  
 2093 `acpica.lib` produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
 2094 debug  
 2095 version of the code includes the debug output trace mechanism and has a  
 2096 much  
 2097 larger code and data size.

2099 Previous Release:  
 2100 Non-Debug Version: 92.3K Code, 24.9K Data, 117.2K Total

```

2101   Debug Version:      170.8K Code, 72.6K Data, 243.4K Total
2102   Current Release:
2103   Non-Debug Version:  92.8K Code, 24.9K Data, 117.7K Total
2104   Debug Version:      171.7K Code, 72.9K Data, 244.5K Total

2107 2) iASL Compiler/Disassembler and Tools:

2109 Disassembler: fixed a problem with the automatic resource tag generation
2110 support. Fixes a problem where the resource tags are inadvertently not
2111 constructed if the table being disassembled contains external references
2112 to
2113 control methods. Moved the actual construction of the tags to after the
2114 final
2115 namespace is constructed (after 2nd parse is invoked due to external
2116 control
2117 method references.) ACPICA BZ 941.

2119 Table Compiler: Make all "generic" operators caseless. These are the
2120 operators
2121 like UINT8, String, etc. Making these caseless improves ease-of-use.
2122 ACPICA BZ
2123 934.

2125 -----
2126 23 November 2011. Summary of changes for version 20111123:

2128 0) ACPI 5.0 Support:

2130 This release contains full support for the ACPI 5.0 specification, as
2131 summarized below.

2133 Reduced Hardware Support:
2134 -----

2136 This support allows for ACPI systems without the usual ACPI hardware.
2137 This
2138 support is enabled by a flag in the revision 5 FADT. If it is set, ACPICA
2139 will
2140 not attempt to initialize or use any of the usual ACPI hardware. Note,
2141 when
2142 this flag is set, all of the following ACPI hardware is assumed to be not
2143 present and is not initialized or accessed:

2145   General Purpose Events (GPEs)
2146   Fixed Events (PM1a/PM1b and PM Control)
2147   Power Management Timer and Console Buttons (power/sleep)
2148   Real-time Clock Alarm
2149   Global Lock
2150   System Control Interrupt (SCI)
2151   The FACS is assumed to be non-existent

2153 ACPI Tables:
2154 -----

2156 All new tables and updates to existing tables are fully supported in the
2157 ACPICA headers (for use by device drivers), the disassembler, and the
2158 iASL
2159 Data Table Compiler. ACPI 5.0 defines these new tables:

2161   BGRT          /* Boot Graphics Resource Table */
2162   DRPM          /* Dynamic Root of Trust for Measurement table */
2163   FPDPT         /* Firmware Performance Data Table */
2164   GTDT          /* Generic Timer Description Table */
2165   MPST          /* Memory Power State Table */
2166   PCCT          /* Platform Communications Channel Table */

```

```

2167   PMTT          /* Platform Memory Topology Table */
2168   RASF          /* RAS Feature table */

2170 Operation Regions/SpaceIDs:
2171 -----

2173 All new operation regions are fully supported by the iASL compiler, the
2174 disassembler, and the ACPICA runtime code (for dispatch to region
2175 handlers.)
2176 The new operation region Space IDs are:

2178   GeneralPurposeIo
2179   GenericSerialBus

2181 Resource Descriptors:
2182 -----

2184 All new ASL resource descriptors are fully supported by the iASL
2185 compiler,
2186 the
2187 ASL/AML disassembler, and the ACPICA runtime Resource Manager code
2188 (including
2189 all new predefined resource tags). New descriptors are:

2191   FixedDma
2192   GpioIo
2193   GpioInt
2194   I2cSerialBus
2195   SpiSerialBus
2196   UartSerialBus

2198 ASL/AML Operators, New and Modified:
2199 -----

2201 One new operator is added, the Connection operator, which is used to
2202 associate
2203 a GeneralPurposeIo or GenericSerialBus resource descriptor with
2204 individual
2205 field objects within an operation region. Several new protocols are
2206 associated
2207 with the AccessAs operator. All are fully supported by the iASL compiler,
2208 disassembler, and runtime ACPICA AML interpreter:

2210   Connection          // Declare Field Connection
2211   attributes
2212   AccessAs: AttribBytes (n)          // Read/Write N-Bytes Protocol
2213   AccessAs: AttribRawBytes (n)       // Raw Read/Write N-Bytes
2214   Protocol
2215   AccessAs: AttribRawProcessBytes (n) // Raw Process Call Protocol
2216   RawDataBuffer        // Data type for Vendor Data
2217   fields

2219 Predefined ASL/AML Objects:
2220 -----

2222 All new predefined objects/control-methods are supported by the iASL
2223 compiler
2224 and the ACPICA runtime validation/repair (arguments and return values.)
2225 New
2226 predefined names include the following:

2228 Standard Predefined Names (Objects or Control Methods):
2229   _AEI, _CLS, _CPC, _CWS, _DEP,
2230   _DLM, _EVT, _GCP, _CRT, _GWS,
2231   _HRV, _PRE, _PSE, _SRT, _SUB.

```

2233 Resource Tags (Names used to access individual fields within resource  
 2234 descriptors):  
 2235     \_DBT, \_DPL, \_DRS, \_END, \_FLC,  
 2236     \_IOR, \_LIN, \_MOD, \_PAR, \_PHA,  
 2237     \_PIN, \_PPI, \_POL, \_RXL, \_SLV,  
 2238     \_SPE, \_STB, \_TXL, \_VEN.

2240 ACPICA External Interfaces:  
 2241 -----

2243 Several new interfaces have been defined for use by ACPI-related device  
 2244 drivers and other host OS services:

2246 AcpiAcquireMutex and AcpiReleaseMutex: These interfaces allow the host OS  
 2247 to  
 2248 acquire and release AML mutexes that are defined in the DSDT/SSDT tables  
 2249 provided by the BIOS. They are intended to be used in conjunction with  
 2250 the  
 2251 ACPI 5.0 \_DLM (Device Lock Method) in order to provide transaction-level  
 2252 mutual exclusion with the AML code/interpreter.

2254 AcpiGetEventResources: Returns the (formatted) resource descriptors as  
 2255 defined  
 2256 by the ACPI 5.0 \_AEI object (ACPI Event Information). This object  
 2257 provides  
 2258 resource descriptors associated with hardware-reduced platform events,  
 2259 similar  
 2260 to the AcpiGetCurrentResources interface.

2262 Operation Region Handlers: For General Purpose IO and Generic Serial Bus  
 2263 operation regions, information about the Connection() object and any  
 2264 optional  
 2265 length information is passed to the region handler within the Context  
 2266 parameter.

2268 AcpiBufferToResource: This interface converts a raw AML buffer containing  
 2269 a  
 2270 resource template or resource descriptor to the ACPI\_RESOURCE internal  
 2271 format  
 2272 suitable for use by device drivers. Can be used by an operation region  
 2273 handler  
 2274 to convert the Connection() buffer object into a ACPI\_RESOURCE.

2276 Miscellaneous/Tools/TestSuites:  
 2277 -----

2279 Support for extended \_HID names (Four alpha characters instead of three).  
 2280 Support for ACPI 5.0 features in the AcpiExec and AcpiHelp utilities.  
 2281 Support for ACPI 5.0 features in the ASLTS test suite.  
 2282 Fully updated documentation (ACPICA and iASL reference documents.)

2284 ACPI Table Definition Language:  
 2285 -----

2287 Support for this language was implemented and released as a subsystem of  
 2288 the  
 2289 iASL compiler in 2010. (See the iASL compiler User Guide.)

2292 Non-ACPI 5.0 changes for this release:  
 2293 -----

2295 1) ACPICA Core Subsystem:

2297 Fix a problem with operation region declarations where a failure can  
 2298 occur

2299 if  
 2300 the region name and an argument that evaluates to an object (such as the  
 2301 region address) are in different namespace scopes. Lin Ming, ACPICA BZ  
 2302 937.

2304 Do not abort an ACPI table load if an invalid space ID is found within.  
 2305 This  
 2306 will be caught later if the offending method is executed. ACPICA BZ 925.

2308 Fixed an issue with the FFixedHW space ID where the ID was not always  
 2309 recognized properly (Both ACPICA and iASL). ACPICA BZ 926.

2311 Fixed a problem with the 32-bit generation of the unix-specific OSL  
 2312 (osunixxf.c). Lin Ming, ACPICA BZ 936.

2314 Several changes made to enable generation with the GCC 4.6 compiler.  
 2315 ACPICA BZ  
 2316 935.

2318 New error messages: Unsupported I/O requests (not 8/16/32 bit), and  
 2319 Index/Bank  
 2320 field registers out-of-range.

2322 2) iASL Compiler/Disassembler and Tools:

2324 iASL: Implemented the \_\_PATH\_\_ operator, which returns the full pathname  
 2325 of  
 2326 the current source file.

2328 AcpiHelp: Automatically display expanded keyword information for all ASL  
 2329 operators.

2331 Debugger: Add "Template" command to disassemble/dump resource template  
 2332 buffers.

2334 Added a new master script to generate and execute the ASLTS test suite.  
 2335 Automatically handles 32- and 64-bit generation. See tests/aslts.sh

2337 iASL: Fix problem with listing generation during processing of the  
 2338 Switch()  
 2339 operator where AML listing was disabled until the entire Switch block was  
 2340 completed.

2342 iASL: Improve support for semicolon statement terminators. Fix "invalid  
 2343 character" message for some cases when the semicolon is used. Semicolons  
 2344 are  
 2345 now allowed after every <Term> grammar element. ACPICA BZ 927.

2347 iASL: Fixed some possible aliasing warnings during generation. ACPICA BZ  
 2348 923.

2350 Disassembler: Fix problem with disassembly of the DataTableRegion  
 2351 operator  
 2352 where an inadvertent "Unhandled deferred opcode" message could be  
 2353 generated.

2355 3) Example Code and Data Size

2357 These are the sizes for the OS-independent acpica.lib produced by the  
 2358 Microsoft Visual C++ 9.0 32-bit compiler. The debug version of the code  
 2359 includes the debug output trace mechanism and has a much larger code and  
 2360 data  
 2361 size.

2363 Previous Release:  
 2364 Non-Debug Version: 90.2K Code, 23.9K Data, 114.1K Total

```

2365     Debug Version:      165.6K Code, 68.4K Data, 234.0K Total
2366     Current Release:
2367     Non-Debug Version:   92.3K Code, 24.9K Data, 117.2K Total
2368     Debug Version:      170.8K Code, 72.6K Data, 243.4K Total

2370 -----
2371 22 September 2011. Summary of changes for version 20110922:

2373 0) ACPI 5.0 News:

2375 Support for ACPI 5.0 in ACPICA has been underway for several months and
2376 will
2377 be released at the same time that ACPI 5.0 is officially released.

2379 The ACPI 5.0 specification is on track for release in the next few
2380 months.
2381
2382 1) ACPICA Core Subsystem:

2384 Fixed a problem where the maximum sleep time for the Sleep() operator was
2385 intended to be limited to two seconds, but was inadvertently limited to
2386 20
2387 seconds instead.

2389 Linux and Unix makefiles: Added header file dependencies to ensure
2390 correct
2391 generation of ACPICA core code and utilities. Also simplified the
2392 makefiles
2393 considerably through the use of the vpath variable to specify search
2394 paths.
2395 ACPICA BZ 924.

2397 2) iASL Compiler/Disassembler and Tools:

2399 iASL: Implemented support to check the access length for all fields
2400 created to
2401 access named Resource Descriptor fields. For example, if a resource field
2402 is
2403 defined to be two bits, a warning is issued if a CreateXxxxField() is
2404 used
2405 with an incorrect bit length. This is implemented for all current
2406 resource
2407 descriptor names. ACPICA BZ 930.
2408
2409 Disassembler: Fixed a byte ordering problem with the output of 24-bit and
2410 56-
2411 bit integers.

2413 iASL: Fixed a couple of issues associated with variable-length package
2414 objects. 1) properly handle constants like One, Ones, Zero -- do not make
2415 a
2416 VAR_PACKAGE when these are used as a package length. 2) Allow the
2417 VAR_PACKAGE
2418 opcode (in addition to PACKAGE) when validating object types for
2419 predefined
2420 names.

2422 iASL: Emit statistics for all output files (instead of just the ASL input
2423 and
2424 AML output). Includes listings, hex files, etc.

2426 iASL: Added -G option to the table compiler to allow the compilation of
2427 custom
2428 ACPI tables. The only part of a table that is required is the standard
2429 36-
2430 byte

```

```

2431 ACPI header.

2433 AcpiXtract: Ported to the standard ACPICA environment (with ACPICA
2434 headers),
2435 which also adds correct 64-bit support. Also, now all output filenames
2436 are
2437 completely lower case.

2439 AcpiExec: Ignore any non-AML tables (tables other than DSDT or SSDT) when
2440 loading table files. A warning is issued for any such tables. The only
2441 exception is an FADT. This also fixes a possible fault when attempting to
2442 load
2443 non-AML tables. ACPICA BZ 932.

2445 AcpiHelp: Added the AccessAs and Offset operators. Fixed a problem where
2446 a
2447 missing table terminator could cause a fault when using the -p option.

2449 AcpiSrc: Fixed a possible divide-by-zero fault when generating file
2450 statistics.

2452 3) Example Code and Data Size

2454 These are the sizes for the OS-independent acpica.lib produced by the
2455 Microsoft Visual C++ 9.0 32-bit compiler. The debug version of the code
2456 includes the debug output trace mechanism and has a much larger code and
2457 data
2458 size.

2460 Previous Release (VC 9.0):
2461 Non-Debug Version:   90.2K Code, 23.9K Data, 114.1K Total
2462 Debug Version:      165.6K Code, 68.4K Data, 234.0K Total
2463 Current Release (VC 9.0):
2464 Non-Debug Version:   90.2K Code, 23.9K Data, 114.1K Total
2465 Debug Version:      165.6K Code, 68.4K Data, 234.0K Total

2468 -----
2469 23 June 2011. Summary of changes for version 20110623:

2471 1) ACPI CA Core Subsystem:

2473 Updated the predefined name repair mechanism to not attempt repair of a
2474 _TSS
2475 return object if a _PSS object is present. We can only sort the _TSS
2476 return
2477 package if there is no _PSS within the same scope. This is because if
2478 _PSS
2479 is
2480 present, the ACPI specification dictates that the _TSS Power Dissipation
2481 field
2482 is to be ignored, and therefore some BIOSs leave garbage values in the
2483 _TSS
2484 Power field(s). In this case, it is best to just return the _TSS package
2485 as-
2486 is. Reported by, and fixed with assistance from Fenghua Yu.

2488 Added an option to globally disable the control method return value
2489 validation
2490 and repair. This runtime option can be used to disable return value
2491 repair
2492 if
2493 this is causing a problem on a particular machine. Also added an option
2494 to
2495 AcpiExec (-dr) to set this disable flag.

```

2497 All makefiles and project files: Major changes to improve generation of  
 2498 ACPICA  
 2499 tools. ACPICA BZ 912:  
 2500     Reduce default optimization levels to improve compatibility  
 2501     For Linux, add strict-aliasing=0 for gcc 4  
 2502     Cleanup and simplify use of command line defines  
 2503     Cleanup multithread library support  
 2504     Improve usage messages

2506 Linux-specific header: update handling of THREAD\_ID and pthread. For the  
 2507 32-  
 2508 bit case, improve casting to eliminate possible warnings, especially with  
 2509 the  
 2510 acpica tools.

2512 Example Code and Data Size: These are the sizes for the OS-independent  
 2513 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
 2514 debug  
 2515 version of the code includes the debug output trace mechanism and has a  
 2516 much  
 2517 larger code and data size.

2519 Previous Release (VC 9.0):  
 2520     Non-Debug Version: 90.1K Code, 23.9K Data, 114.0K Total  
 2521     Debug Version: 165.6K Code, 68.4K Data, 234.0K Total  
 2522 Current Release (VC 9.0):  
 2523     Non-Debug Version: 90.2K Code, 23.9K Data, 114.1K Total  
 2524     Debug Version: 165.6K Code, 68.4K Data, 234.0K Total

#### 2526 2) iASL Compiler/Disassembler and Tools:

2528 With this release, a new utility named "acpihelp" has been added to the  
 2529 ACPICA  
 2530 package. This utility summarizes the ACPI specification chapters for the  
 2531 ASL  
 2532 and AML languages. It generates under Linux/Unix as well as Windows, and  
 2533 provides the following functionality:  
 2534     Find/display ASL operator(s) -- with description and syntax.  
 2535     Find/display ASL keyword(s) -- with exact spelling and descriptions.  
 2536     Find/display ACPI predefined name(s) -- with description, number  
 2537     of arguments, and the return value data type.  
 2538     Find/display AML opcode name(s) -- with opcode, arguments, and  
 2539 grammar.  
 2540     Decode/display AML opcode -- with opcode name, arguments, and  
 2541 grammar.

2543 Service Layers: Make multi-thread support configurable. Conditionally  
 2544 compile  
 2545 the multi-thread support so that threading libraries will not be linked  
 2546 if  
 2547 not  
 2548 necessary. The only tool that requires multi-thread support is AcpiExec.

2550 iASL: Update yyerror/AslCompilerError for "const" errors. Newer versions  
 2551 of  
 2552 Bison appear to want the interface to yyerror to be a const char \* (or at  
 2553 least this is a problem when generating iASL on some systems.) ACPICA BZ  
 2554 923  
 2555 Pierre Lejeune.

2557 Tools: Fix for systems where O\_BINARY is not defined. Only used for  
 2558 Windows  
 2559 versions of the tools.

2561 -----  
 2562 27 May 2011. Summary of changes for version 20110527:

#### 2564 1) ACPI CA Core Subsystem:

2566 ASL Load() operator: Reinstate most restrictions on the incoming ACPI  
 2567 table  
 2568     8 ASL Load() operator: Reinstate most restrictions on the incoming ACPI table  
 2569 signature. Now, only allow SSDT, OEMx, and a null signature. History:  
 2570     1) Originally, we checked the table signature for "SSDT" or "PSDT".  
 2571         (PSDT is now obsolete.)  
 2572     2) We added support for OEMx tables, signature "OEM" plus a fourth  
 2573         "don't care" character.  
 2574     3) Valid tables were encountered with a null signature, so we just  
 2575         gave up on validating the signature, (05/2008).  
 2576     4) We encountered non-AML tables such as the MADT, which caused  
 2577         interpreter errors and kernel faults. So now, we once again allow  
 2578         only SSDT, OEMx, and now, also a null signature. (05/2011).

2579 Added the missing \_TDL predefined name to the global name list in order  
 2580 to  
 2581 enable validation. Affects both the core ACPICA code and the iASL  
 2582 compiler.  
 2583     20 Added the missing \_TDL predefined name to the global name list in order to  
 2584     21 enable validation. Affects both the core ACPICA code and the iASL compiler.

2584 Example Code and Data Size: These are the sizes for the OS-independent  
 2585 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
 2586 debug  
 2587 version of the code includes the debug output trace mechanism and has a  
 2588 much  
 2589     24 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug  
 2590     25 version of the code includes the debug output trace mechanism and has a much  
 2591     26 larger code and data size.

2591 Previous Release (VC 9.0):  
 2592     Non-Debug Version: 90.0K Code, 23.8K Data, 113.8K Total  
 2593     Debug Version: 164.5K Code, 68.0K Data, 232.5K Total  
 2594 Current Release (VC 9.0):  
 2595     Non-Debug Version: 90.1K Code, 23.9K Data, 114.0K Total  
 2596     Debug Version: 165.6K Code, 68.4K Data, 234.0K Total

#### 2598 2) iASL Compiler/Disassembler and Tools:

2600 Debugger/AcpiExec: Implemented support for "complex" method arguments on  
 2601 the  
 2602 debugger command line. This adds support beyond simple integers --  
 2603 including  
 2604     37 Debugger/AcpiExec: Implemented support for "complex" method arguments on the  
 2605     38 debugger command line. This adds support beyond simple integers -- including  
 2606     39 Strings, Buffers, and Packages. Includes support for nested packages.  
 2607     40 Increased the default command line buffer size to accommodate these  
 2608     41 arguments.  
 2609     42 Increased the default command line buffer size to accommodate these arguments.  
 2610 See the ACPICA reference for details and syntax. ACPICA BZ 917.  
 2611 Debugger/AcpiExec: Implemented support for "default" method arguments for  
 2612 the  
 2613 Execute/Debug command. Now, the debugger will always invoke a control  
 2614 method  
 2615 with the required number of arguments -- even if the command line  
 2616 specifies  
 2617 none or insufficient arguments. It uses default integer values for any  
 2618 missing  
 2619     43 Debugger/AcpiExec: Implemented support for "default" method arguments for the  
 2620     44 Execute/Debug command. Now, the debugger will always invoke a control method  
 2621     45 with the required number of arguments -- even if the command line specifies  
 2622     46 none or insufficient arguments. It uses default integer values for any missing



2617 arguments. Also fixes a bug where only six method arguments maximum were  
2618 supported instead of the required seven.

2620 **Debugger/AcpiExec: Add a maximum buffer length parameter to AcpiOsGetLine**  
2621 **and**  
2622 *50 Debugger/AcpiExec: Add a maximum buffer length parameter to AcpiOsGetLine and*  
2623 *also return status in order to prevent buffer overruns. See the ACPICA*  
2623 *reference for details and syntax. ACPICA BZ 921*

2625 iASL: Cleaned up support for Berkeley yacc. A general cleanup of code and  
2626 makefiles to simplify support for the two different but similar parser  
2627 generators, bison and yacc.

2629 **Updated the generic unix makefile for gcc 4. The default gcc version is**  
2630 **now**  
2631 *58 Updated the generic unix makefile for gcc 4. The default gcc version is now*  
2631 *expected to be 4 or greater, since options specific to gcc 4 are used.*

2633 -----  
2634 13 April 2011. Summary of changes for version 20110413:

2636 1) ACPI CA Core Subsystem:

2638 **Implemented support to execute a so-called "orphan" \_REG method under the**  
2639 *66 Implemented support to execute a so-called "orphan" \_REG method under the EC*  
2640 *device. This change will force the execution of a \_REG method underneath the*  
2641 **EC**  
2642 **device. This change will force the execution of a \_REG method underneath**  
2643 **the**  
2644 **EC**  
2645 device even if there is no corresponding operation region of type  
2646 EmbeddedControl. Fixes a problem seen on some machines and apparently is  
2647 compatible with Windows behavior. ACPICA BZ 875.

2647 **Added more predefined methods that are eligible for automatic NULL**  
2648 **package**  
2649 **element removal. This change adds another group of predefined names to**  
2650 **the**  
2651 *73 Added more predefined methods that are eligible for automatic NULL package*  
2652 *element removal. This change adds another group of predefined names to the*  
2653 **list**  
2654 **of names that can be repaired by having NULL package elements dynamically**  
2655 **removed. This group are those methods that return a single variable-**  
2656 **length**  
2657 **package containing simple data types such as integers, buffers, strings.**  
2658 **This**  
2659 **includes: \_ALx, \_BCL, \_CID, \_DOD, \_EDL, \_FIX, \_PCL, \_PLD, \_PMD, \_PRx,**  
2660 **\_PSL,**  
2661 *77 removed. This group are those methods that return a single variable-length*  
2662 *package containing simple data types such as integers, buffers, strings. This*  
2663 *includes: \_ALx, \_BCL, \_CID, \_DOD, \_EDL, \_FIX, \_PCL, \_PLD, \_PMD, \_PRx, \_PSL,*  
2664 *\_Sx,*  
2665 **and \_TZD. ACPICA BZ 914.**

2662 Split and segregated all internal global lock functions to a new file,  
2663 evglock.c.

2665 **Updated internal address SpaceID for DataTable regions. Moved this**  
2666 **internal**  
2667 *86 Updated internal address SpaceID for DataTable regions. Moved this internal*  
2668 **space**  
2669 **id in preparation for ACPI 5.0 changes that will include some new space**  
2670 **IDs.**  
2671 *88 id in preparation for ACPI 5.0 changes that will include some new space IDs.*  
2672 **This**  
2673 **change should not affect user/host code.**

2673 Example Code and Data Size: These are the sizes for the OS-independent  
2674 acpica.lib  
2675 **produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug**  
2676 **version of**  
2677 **the code includes the debug output trace mechanism and has a much larger**  
2678 **code**  
2679 *94 produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug version of*  
2680 *95 the code includes the debug output trace mechanism and has a much larger code*  
2681 **and**  
2682 **data size.**

2682 Previous Release (VC 9.0):  
2683 Non-Debug Version: 89.8K Code, 23.8K Data, 113.6K Total  
2684 Debug Version: 164.2K Code, 67.9K Data, 232.1K Total  
2685 Current Release (VC 9.0):  
2686 Non-Debug Version: 90.0K Code, 23.8K Data, 113.8K Total  
2687 Debug Version: 164.5K Code, 68.0K Data, 232.5K Total

2689 2) iASL Compiler/Disassembler and Tools:

2691 **iASL/DTC: Major update for new grammar features. Allow generic data types**  
2692 **in**  
2693 **custom ACPI tables. Field names are now optional. Any line can be split**  
2694 **to**  
2695 **multiple lines using the continuation char (\). Large buffers now use**  
2696 **line-**  
2697 *108 iASL/DTC: Major update for new grammar features. Allow generic data types in*  
2698 *109 custom ACPI tables. Field names are now optional. Any line can be split to*  
2699 *110 multiple lines using the continuation char (\). Large buffers now use line-*  
2700 *111 continuation character(s) and no colon on the continuation lines. See the*  
2701 **grammar**  
2702 **update in the iASL compiler reference. ACPI BZ 910,911. Lin Ming, Bob**  
2703 **Moore.**  
2704 *113 update in the iASL compiler reference. ACPI BZ 910,911. Lin Ming, Bob Moore.*

2702 **iASL: Mark ASL "Return()" and the simple "Return" as "Null" return**  
2703 **statements.**  
2704 **Since the parser stuffs a "zero" as the return value for these statements**  
2705 **(due**  
2706 *115 iASL: Mark ASL "Return()" and the simple "Return" as "Null" return statements.*  
2707 *116 Since the parser stuffs a "zero" as the return value for these statements (due*  
2708 **to**  
2709 **the underlying AML grammar), they were seen as "return with value" by the**  
2710 **iASL**  
2711 **semantic checking. They are now seen correctly as "null" return**  
2712 **statements.**  
2713 *118 the underlying AML grammar), they were seen as "return with value" by the iASL*  
2714 *119 semantic checking. They are now seen correctly as "null" return statements.*

2712 **iASL: Check if a \_REG declaration has a corresponding Operation Region.**  
2713 **Adds a**  
2714 **check for each \_REG to ensure that there is in fact a corresponding**  
2715 **operation**  
2716 *121 iASL: Check if a \_REG declaration has a corresponding Operation Region. Adds a*  
2717 *122 check for each \_REG to ensure that there is in fact a corresponding operation*  
2718 *123 region declaration in the same scope. If not, the \_REG method is not very*  
2719 **useful**  
2720 **since it probably won't be executed. ACPICA BZ 915.**

2720 **iASL/DTC: Finish support for expression evaluation. Added a new**  
2721 **expression**  
2722 *127 iASL/DTC: Finish support for expression evaluation. Added a new expression*  
2723 **parser**  
2724 **that implements c-style operator precedence and parenthesization. ACPICA**  
2725 **bugzilla**

2725 908.

2727 **Disassembler/DTC: Remove support for () and <> style comments in data**  
 2728 **tables.**  
 133 *Disassembler/DTC: Remove support for () and <> style comments in data tables.*  
 2729 Now  
 2730 **that DTC has full expression support, we don't want to have comment**  
 2731 **strings**  
 135 *that DTC has full expression support, we don't want to have comment strings*  
 2732 **that**  
 2733 **start with a parentheses or a less-than symbol. Now, only the standard /\***  
 2734 **and**  
 137 *start with a parentheses or a less-than symbol. Now, only the standard /\* and*  
 2735 *//*  
 2736 **comments are supported, as well as the bracket [] comments.**

2738 **AcpiXtract: Fix for RSDP and dynamic SSDT extraction. These tables have**  
 2739 **"unusual"**  
 2740 **headers in the acpidump file. Update the header validation to support**  
 2741 **these**  
 2742 **tables. Problem introduced in previous AcpiXtract version in the change**  
 2743 **to**  
 143 *headers in the acpidump file. Update the header validation to support these*  
 144 *tables. Problem introduced in previous AcpiXtract version in the change to*  
 2744 **support "wrong checksum" error messages emitted by acpidump utility.**

2746 **iASL: Add a \* option to generate all template files (as a synonym for**  
 2747 **ALL)**  
 2748 **as**  
 147 *iASL: Add a \* option to generate all template files (as a synonym for ALL) as*  
 2749 *in*  
 2750 **"iasl -T \*" or "iasl -T ALL".**

2752 **iASL/DTC: Do not abort compiler on fatal errors. We do not want to**  
 2753 **completely**  
 2754 **abort the compiler on "fatal" errors, simply should abort the current**  
 2755 **compile.**  
 151 *iASL/DTC: Do not abort compiler on fatal errors. We do not want to completely*  
 152 *abort the compiler on "fatal" errors, simply should abort the current compile.*  
 2756 **This allows multiple compiles with a single (possibly wildcard) compiler**  
 2757 **invocation.**

2759 -----

2760 16 March 2011. Summary of changes for version 20110316:

2762 1) ACPI CA Core Subsystem:

2764 **Fixed a problem caused by a \_PRW method appearing at the namespace root**  
 2765 **scope**  
 2766 **during the setup of wake GPEs. A fault could occur if a \_PRW directly**  
 2767 **under**  
 161 *Fixed a problem caused by a \_PRW method appearing at the namespace root scope*  
 162 *during the setup of wake GPEs. A fault could occur if a \_PRW directly under*  
 2768 *the*  
 2769 **root object was passed to the AcpiSetupGpeForWake interface. Lin Ming.**

2771 **Implemented support for "spurious" Global Lock interrupts. On some**  
 2772 **systems, a**  
 2773 **global lock interrupt can occur without the pending flag being set. Upon**  
 2774 **a**  
 2775 **GL**  
 2776 **interrupt, we now ensure that a thread is actually waiting for the lock**  
 2777 **before**  
 166 *Implemented support for "spurious" Global Lock interrupts. On some systems, a*  
 167 *global lock interrupt can occur without the pending flag being set. Upon a GL*  
 168 *interrupt, we now ensure that a thread is actually waiting for the lock before*

2778 signaling GL availability. Rafael Wysocki, Bob Moore.

2780 Example Code and Data Size: These are the sizes for the OS-independent  
 2781 acpica.lib  
 2782 **produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug**  
 2783 **version of**  
 2784 **the code includes the debug output trace mechanism and has a much larger**  
 2785 **code**  
 173 *produced by the Microsoft Visual C++ 9.0 32-bit compiler. The debug version of*  
 174 *the code includes the debug output trace mechanism and has a much larger code*  
 2786 *and*  
 2787 **data size.**

2789 Previous Release (VC 9.0):  
 2790 Non-Debug Version: 89.7K Code, 23.7K Data, 113.4K Total  
 2791 Debug Version: 163.9K Code, 67.5K Data, 231.4K Total  
 2792 Current Release (VC 9.0):  
 2793 Non-Debug Version: 89.8K Code, 23.8K Data, 113.6K Total  
 2794 Debug Version: 164.2K Code, 67.9K Data, 232.1K Total

2796 2) iASL Compiler/Disassembler and Tools:

2798 **Implemented full support for the "SLIC" ACPI table. Includes support in**  
 2799 **the**  
 2800 **header files, disassembler, table compiler, and template generator. Bob**  
 2801 **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,*  
 2802 **Lin Ming.**

2804 **AcpiXtract: Correctly handle embedded comments and messages from**  
 2805 **AcpiDump.**  
 2806 **Apparently some or all versions of acpidump will occasionally emit a**  
 2807 **comment**  
 191 *AcpiXtract: Correctly handle embedded comments and messages from AcpiDump.*  
 192 *Apparently some or all versions of acpidump will occasionally emit a comment*  
 2808 *like*  
 2809 **"Wrong checksum", etc., into the dump file. This was causing problems for**  
 2810 **AcpiXtract. ACPICA BZ 905.**

2812 **iASL: Fix the Linux makefile by removing an inadvertent double file**  
 2813 **inclusion.**  
 197 *iASL: Fix the Linux makefile by removing an inadvertent double file inclusion.*  
 2814 **ACPICA BZ 913.**

2816 **AcpiExec: Update installation of operation region handlers. Install one**  
 2817 **handler**  
 2818 **for a user-defined address space. This is used by the ASL test suite**  
 2819 **(ASLTS).**  
 202 *for a user-defined address space. This is used by the ASL test suite (ASLTS).*

2821 -----

2822 11 February 2011. Summary of changes for version 20110211:

2824 1) ACPI CA Core Subsystem:

2826 **Added a mechanism to defer \_REG methods for some early-installed**  
 2827 **handlers.**  
 2828 **Most user handlers should be installed before call to**  
 2829 **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.*  
 2830 **However, Event handlers and region handlers should be installed after**  
 2831 **AcpiInitializeObjects. Override handlers for the "default" regions should**  
 2832 **be**  
 212 *AcpiInitializeObjects. Override handlers for the "default" regions should be*

2833 installed early, however. This change executes all \_REG methods for the  
 2834 default regions (Memory/IO/PCI/DataTable) simultaneously to prevent any  
 2835 chicken/egg issues between them. ACPICA BZ 848.

2837 **Implemented an optimization for GPE detection. This optimization will**  
 2838 **simply**  
 2839 *217 Implemented an optimization for GPE detection. This optimization will simply*  
 2839 ignore GPE registers that contain no enabled GPEs -- there is no need to  
 2840 read the register since this information is available internally. This  
 2841 **becomes more important on machines with a large GPE space. ACPICA**  
 2842 **bugzilla**  
 2843 *220 becomes more important on machines with a large GPE space. ACPICA bugzilla*  
 2843 884. Lin Ming. Suggestion from Joe Liu.

2845 **Removed all use of the highly unreliable FADT revision field. The**  
 2846 **revision**  
 2847 **number in the FADT has been found to be completely unreliable and cannot**  
 2848 **be**  
 2849 **trusted. Only the actual table length can be used to infer the version.**  
 2850 **This**  
 2851 **change updates the ACPICA core and the disassembler so that both no**  
 2852 **longer**  
 2823 *223 Removed all use of the highly unreliable FADT revision field. The revision*  
 2824 *224 number in the FADT has been found to be completely unreliable and cannot be*  
 2825 *225 trusted. Only the actual table length can be used to infer the version. This*  
 2826 *226 change updates the ACPICA core and the disassembler so that both no longer*  
 2853 even look at the FADT version and instead depend solely upon the FADT  
 2854 length.

2856 **Fix an unresolved name issue for the no-debug and no-error-message source**  
 2857 **generation cases. The \_AcpiModuleName was left undefined in these cases,**  
 2858 **but**  
 2859 *231 generation cases. The \_AcpiModuleName was left undefined in these cases, but*  
 2859 it is actually needed as a parameter to some interfaces. Define  
 2860 \_AcpiModuleName as a null string in these cases. ACPICA Bugzilla 888.

2862 Split several large files (makefiles and project files updated)  
 2863 utglobal.c -> utdecode.c  
 2864 dbcomds.c -> dbmethod.c dbnames.c  
 2865 dsopcode.c -> dsargs.c dscontrol.c  
 2866 dsload.c -> dsload2.c  
 2867 aslanalyze.c -> asltypes.c aslwalks.c

2869 Example Code and Data Size: These are the sizes for the OS-independent  
 2870 acpica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
 2871 **debug version of the code includes the debug output trace mechanism and**  
 2872 **has**  
 2873 *244 debug version of the code includes the debug output trace mechanism and has*  
 2873 a much larger code and data size.

2875 Previous Release (VC 9.0):  
 2876 Non-Debug Version: 89.7K Code, 23.7K Data, 113.4K Total  
 2877 Debug Version: 163.9K Code, 67.5K Data, 231.4K Total  
 2878 Current Release (VC 9.0):  
 2879 Non-Debug Version: 89.7K Code, 23.7K Data, 113.4K Total  
 2880 Debug Version: 163.9K Code, 67.5K Data, 231.4K Total

2882 2) iASL Compiler/Disassembler and Tools:

2884 iASL: Implemented the predefined macros \_\_LINE\_\_, \_\_FILE\_\_, and \_\_DATE\_\_.  
 2885 These are useful C-style macros with the standard definitions. ACPICA  
 2886 bugzilla 898.

2888 **iASL/DTC: Added support for integer expressions and labels. Support for**  
 2889 **full**  
 2890 **expressions for all integer fields in all ACPI tables. Support for labels**

2891 **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*  
 2892 "generic" portions of tables such as UEFI. See the iASL reference manual.

2894 Debugger: Added a command to display the status of global handlers. The  
 2895 "handlers" command will display op region, fixed event, and miscellaneous  
 2896 **global handlers. installation status -- and for op regions, whether**  
 2897 **default**  
 266 *global handlers. installation status -- and for op regions, whether default*  
 2898 or user-installed handler will be used.

2900 **iASL: Warn if reserved method incorrectly returns a value. Many**  
 2901 **predefined**  
 2902 **names are defined such that they do not return a value. If implemented as**  
 2903 **a**  
 269 *iASL: Warn if reserved method incorrectly returns a value. Many predefined*  
 270 *names are defined such that they do not return a value. If implemented as a*  
 2904 method, issue a warning if such a name explicitly returns a value. ACPICA  
 2905 Bugzilla 855.

2907 **iASL: Added detection of GPE method name conflicts. Detects a conflict**  
 2908 **where**  
 2909 **there are two GPE methods of the form \_Lxy and \_Exy in the same scope.**  
 2910 **(For**  
 274 *iASL: Added detection of GPE method name conflicts. Detects a conflict where*  
 275 *there are two GPE methods of the form \_Lxy and \_Exy in the same scope. (For*  
 2911 *example, \_L1d and \_E1d in the same scope.) ACPICA bugzilla 848.*

2913 iASL/DTC: Fixed a couple input scanner issues with comments and line  
 2914 **numbers. Comment remover could get confused and miss a comment ending.**  
 2915 **Fixed**  
 279 *numbers. Comment remover could get confused and miss a comment ending. Fixed*  
 2916 a problem with line counter maintenance.

2918 **iASL/DTC: Reduced the severity of some errors from fatal to error. There**  
 2919 **is**  
 282 *iASL/DTC: Reduced the severity of some errors from fatal to error. There is*  
 2920 no need to abort on simple errors within a field definition.

2922 **Debugger: Simplified the output of the help command. All help output now**  
 2923 **in**  
 285 *Debugger: Simplified the output of the help command. All help output now in*  
 2924 a single screen, instead of help subcommands. ACPICA Bugzilla 897.

2926 -----  
 2927 12 January 2011. Summary of changes for version 20110112:

2929 1) ACPI CA Core Subsystem:

2931 **Fixed a race condition between method execution and namespace walks that**  
 2932 **can**  
 293 *Fixed a race condition between method execution and namespace walks that can*  
 2933 possibly cause a fault. The problem was apparently introduced in version  
 2934 **20100528 as a result of a performance optimization that reduces the**  
 2935 **number**  
 2936 **of**  
 295 *20100528 as a result of a performance optimization that reduces the number of*  
 2937 namespace walks upon method exit by using the delete\_namespace\_subtree  
 2938 **function instead of the delete\_namespace\_by\_owner function used**  
 2939 **previously.**  
 297 *function instead of the delete\_namespace\_by\_owner function used previously.*  
 2940 Bug is a missing namespace lock in the delete\_namespace\_subtree function.  
 2941 dana.myers@oracle.com

2943 Fixed several issues and a possible fault with the automatic "serialized"

2944 **method support. History: This support changes a method to "serialized" on**  
 2945 **the**  
 302 *method support. History: This support changes a method to "serialized" on the*  
 2946 *fly if the method generates an AE\_ALREADY\_EXISTS error, indicating the*  
 2947 **possibility that it cannot handle reentrancy. This fix repairs a couple**  
 2948 **of**  
 304 *possibility that it cannot handle reentrancy. This fix repairs a couple of*  
 2949 *issues seen in the field, especially on machines with many cores:*

- 2951 1) Delete method children only upon the exit of the last thread,  
 2952 so as to not delete objects out from under other running threads  
 2953 (and possibly causing a fault.)
- 2954 2) Set the "serialized" bit for the method only upon the exit of the  
 2955 Last thread, so as to not cause deadlock when running threads  
 2956 attempt to exit.
- 2957 3) Cleanup the use of the AML "MethodFlags" and internal method flags  
 2958 so that there is no longer any confusion between the two.

2960 Lin Ming, Bob Moore. Reported by dana.myers@oracle.com.

2962 **Debugger: Now lock the namespace for duration of a namespace dump.**  
 2963 **Prevents**

318 *Debugger: Now lock the namespace for duration of a namespace dump. Prevents*  
 2964 *issues if the namespace is changing dynamically underneath the debugger.*  
 2965 *Especially affects temporary namespace nodes, since the debugger displays*  
 2966 *these also.*

2968 Updated the ordering of include files. The ACPICA headers should appear  
 2969 **before any compiler-specific headers (stdio.h, etc.) so that acenv.h can**  
 2970 **set**  
 2971 **any necessary compiler-specific defines, etc. Affects the ACPI-related**  
 2972 **tools**  
 324 *before any compiler-specific headers (stdio.h, etc.) so that acenv.h can set*  
 325 *any necessary compiler-specific defines, etc. Affects the ACPI-related tools*  
 2973 *and utilities.*

2975 **Updated all ACPICA copyrights and signons to 2011. Added the 2011**  
 2976 **copyright**  
 2977 **to all module headers and signons, including the Linux header. This**  
 2978 **affects**

328 *Updated all ACPICA copyrights and signons to 2011. Added the 2011 copyright*  
 329 *to all module headers and signons, including the Linux header. This affects*  
 2979 *virtually every file in the ACPICA core subsystem, iASL compiler, and all*  
 2980 *utilities.*

2982 Added project files for MS Visual Studio 2008 (VC++ 9.0). The original  
 2983 **project files for VC++ 6.0 are now obsolete. New project files can be**  
 2984 **found**

334 *project files for VC++ 6.0 are now obsolete. New project files can be found*  
 2985 *under acpica/generate/msvc9. See acpica/generate/msvc9/readme.txt for*  
 2986 *details.*

2988 Example Code and Data Size: These are the sizes for the OS-independent  
 2989 apcica.lib produced by the Microsoft Visual C++ 9.0 32-bit compiler. The  
 2990 **debug version of the code includes the debug output trace mechanism and**  
 2991 **has a**

340 *debug version of the code includes the debug output trace mechanism and has a*  
 2992 *much larger code and data size.*

2994 Previous Release (VC 6.0):  
 2995 Non-Debug Version: 89.8K Code, 18.9K Data, 108.7K Total  
 2996 Debug Version: 166.6K Code, 52.1K Data, 218.7K Total  
 2997 Current Release (VC 9.0):  
 2998 Non-Debug Version: 89.7K Code, 23.7K Data, 113.4K Total  
 2999 Debug Version: 163.9K Code, 67.5K Data, 231.4K Total

3001 2) iASL Compiler/Disassembler and Tools:

3003 **iASL: Added generic data types to the Data Table compiler. Add "generic"**  
 3004 **data**  
 3005 **types such as UINT32, String, Unicode, etc., to simplify the generation**  
 3006 **of**  
 352 *iASL: Added generic data types to the Data Table compiler. Add "generic" data*  
 353 *types such as UINT32, String, Unicode, etc., to simplify the generation of*  
 3007 *platform-defined tables such as UEFI. Lin Ming.*

3009 **iASL: Added listing support for the Data Table Compiler. Adds listing**  
 3010 **support**

356 *iASL: Added listing support for the Data Table Compiler. Adds listing support*  
 3011 *(-l) to display actual binary output for each line of input code.*

3013 -----  
 3014 09 December 2010. Summary of changes for version 20101209:

3016 1) ACPI CA Core Subsystem:

3018 **Completed the major overhaul of the GPE support code that was begun in**  
 3019 **July**

364 *Completed the major overhaul of the GPE support code that was begun in July*  
 3020 *2010. Major features include: removal of \_PRW execution in ACPICA (host*  
 3021 *executes \_PRWs anyway), cleanup of "wake" GPE interfaces and processing,*  
 3022 **changes to existing interfaces, simplification of GPE handler operation,**  
 3023 **and**  
 367 *changes to existing interfaces, simplification of GPE handler operation, and*  
 3024 *a handful of new interfaces:*

3026 AcpiUpdateAllGpes  
 3027 AcpiFinishGpe  
 3028 AcpiSetupGpeForWake  
 3029 AcpiSetGpeWakeMask  
 3030 One new file, evxfpe.c to consolidate all external GPE interfaces.

3032 See the ACPICA Programmer Reference for full details and programming  
 3033 **information. See the new section 4.4 "General Purpose Event (GPE)**  
 3034 **Support"**

3035 **for a full overview, and section 8.7 "ACPI General Purpose Event**  
 3036 **Management"**  
 3037 **for programming details. ACPICA BZ 858,870,877. Matthew Garrett, Lin**  
 3038 **Ming,**

377 *information. See the new section 4.4 "General Purpose Event (GPE) Support"*  
 378 *for a full overview, and section 8.7 "ACPI General Purpose Event Management"*  
 379 *for programming details. ACPICA BZ 858,870,877. Matthew Garrett, Lin Ming,*  
 3039 *Bob Moore, Rafael Wysocki.*

3041 **Implemented a new GPE feature for Windows compatibility, the "Implicit**  
 3042 **Wake**

3043 **GPE Notify". This feature will automatically issue a Notify(2) on a**  
 3044 **device**

382 *Implemented a new GPE feature for Windows compatibility, the "Implicit Wake*  
 383 *GPE Notify". This feature will automatically issue a Notify(2) on a device*  
 3045 *when a Wake GPE is received if there is no corresponding GPE method or*  
 3046 *handler. ACPICA BZ 870.*

3048 **Fixed a problem with the Scope() operator during table parse and load**  
 3049 **phase.**

3050 **During load phase (table load or method execution), the scope operator**  
 3051 **should**

3052 **not enter the target into the namespace. Instead, it should open a new**  
 3053 **scope**

387 *Fixed a problem with the Scope() operator during table parse and load phase.*  
 388 *During load phase (table load or method execution), the scope operator should*  
 389 *not enter the target into the namespace. Instead, it should open a new scope*

3054 at the target location. Linux BZ 19462, ACPICA BZ 882.

3056 Example Code and Data Size: These are the sizes for the OS-independent  
3057 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
3058 **debug version of the code includes the debug output trace mechanism and**  
3059 **has a**  
3060 *394 debug version of the code includes the debug output trace mechanism and has a*  
3060 much larger code and data size.

3062 Previous Release:  
3063 Non-Debug Version: 89.8K Code, 18.9K Data, 108.7K Total  
3064 Debug Version: 166.6K Code, 52.1K Data, 218.7K Total  
3065 Current Release:  
3066 Non-Debug Version: 89.9K Code, 19.0K Data, 108.9K Total  
3067 Debug Version: 166.3K Code, 52.1K Data, 218.4K Total

3069 2) iASL Compiler/Disassembler and Tools:

3071 **iASL: Relax the alphanumeric restriction on \_CID strings. These strings**  
3072 **are**  
3073 **"bus-specific" per the ACPI specification, and therefore any characters**  
3074 **are**  
3075 **acceptable. The only checks that can be performed are for a null string**  
3076 **and**  
3077 *406 iASL: Relax the alphanumeric restriction on \_CID strings. These strings are*  
3077 *407 "bus-specific" per the ACPI specification, and therefore any characters are*  
3077 *408 acceptable. The only checks that can be performed are for a null string and*  
3077 *409 perhaps for a leading asterisk. ACPICA BZ 886.*

3079 iASL: Fixed a problem where a syntax error that caused a premature EOF  
3080 condition on the source file emitted a very confusing error message. The  
3081 premature EOF is now detected correctly. ACPICA BZ 891.

3083 **Disassembler: Decode the AccessSize within a Generic Address Structure**  
3084 **(byte**  
3085 *415 Disassembler: Decode the AccessSize within a Generic Address Structure (byte*  
3085 *access, word access, etc.) Note, this field does not allow arbitrary bit*  
3086 *access, the size is encoded as 1=byte, 2=word, 3=dword, and 4=qword.*

3088 **New: AcpiNames utility - Example namespace dump utility. Shows an example**  
3089 **of**  
3090 *419 New: AcpiNames utility - Example namespace dump utility. Shows an example of*  
3090 *ACPICA configuration for a minimal namespace dump utility. Uses table and*  
3091 *namespace managers, but no AML interpreter. Does not add any*  
3092 *functionality*  
3093 *421 namespace managers, but no AML interpreter. Does not add any functionality*  
3093 *over AcpiExec, it is a subset of AcpiExec. The purpose is to show how to*  
3094 *partition and configure ACPICA. ACPICA BZ 883.*

3096 **AML Debugger: Increased the debugger buffer size for method return**  
3097 **objects.**  
3098 **Was 4K, increased to 16K. Also enhanced error messages for debugger**  
3099 **method**  
3100 *425 AML Debugger: Increased the debugger buffer size for method return objects.*  
3100 *426 Was 4K, increased to 16K. Also enhanced error messages for debugger method*  
3100 *execution, including the buffer overflow case.*

3102 -----  
3103 13 October 2010. Summary of changes for version 20101013:

3105 1) ACPI CA Core Subsystem:

3107 **Added support to clear the PCIEXP\_WAKE event. When clearing ACPI events,**  
3108 **now**  
3109 *434 Added support to clear the PCIEXP\_WAKE event. When clearing ACPI events, now*  
3109 *clear the PCIEXP\_WAKE\_STS bit in the ACPI PM1 Status Register, via*

3110 HwClearAcpiStatus. Original change from Colin King. ACPICA BZ 880.

3112 **Changed the type of the predefined namespace object \_TZ from ThermalZone**  
3113 **to**  
3114 **Device. This was found to be confusing to the host software that**  
3115 **processes**  
3116 **the various thermal zones, since \_TZ is not really a ThermalZone.**  
3117 **However,**  
3118 **a**  
3119 *438 Changed the type of the predefined namespace object \_TZ from ThermalZone to*  
3119 *439 Device. This was found to be confusing to the host software that processes*  
3119 *440 the various thermal zones, since \_TZ is not really a ThermalZone. However, a*  
3119 *Notify() can still be performed on it. ACPICA BZ 876. Suggestion from Rui*  
3120 *Zhang.*

3122 Added Windows Vista SP2 to the list of supported \_OSI strings. The actual  
3123 string is "Windows 2006 SP2".

3125 **Eliminated duplicate code in AcpiUtExecute\* functions. Now that the**  
3126 **nsrepair**  
3127 *447 Eliminated duplicate code in AcpiUtExecute\* functions. Now that the nsrepair*  
3127 *code automatically repairs \_HID-related strings, this type of code is no*  
3128 *longer needed in Execute\_HID, Execute\_CID, and Execute\_UID. ACPICA BZ*  
3129 *878.*  
3129 *449 longer needed in Execute\_HID, Execute\_CID, and Execute\_UID. ACPICA BZ 878.*

3131 Example Code and Data Size: These are the sizes for the OS-independent  
3132 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
3133 **debug version of the code includes the debug output trace mechanism and**  
3134 **has a**  
3135 *453 debug version of the code includes the debug output trace mechanism and has a*  
3135 much larger code and data size.

3137 Previous Release:  
3138 Non-Debug Version: 89.9K Code, 19.0K Data, 108.9K Total  
3139 Debug Version: 166.3K Code, 52.1K Data, 218.4K Total  
3140 Current Release:  
3141 Non-Debug Version: 89.9K Code, 19.0K Data, 108.9K Total  
3142 Debug Version: 166.3K Code, 52.1K Data, 218.4K Total

3144 2) iASL Compiler/Disassembler and Tools:

3146 **iASL: Implemented additional compile-time validation for \_HID strings.**  
3147 **The**  
3148 **non-hex prefix (such as "PNP" or "ACPI") must be uppercase, and the**  
3149 **length**  
3150 **of**  
3151 **the string must be exactly seven or eight characters. For both \_HID and**  
3152 **\_CID**  
3153 *465 iASL: Implemented additional compile-time validation for \_HID strings. The*  
3153 *466 non-hex prefix (such as "PNP" or "ACPI") must be uppercase, and the length of*  
3153 *467 the string must be exactly seven or eight characters. For both \_HID and \_CID*  
3153 *strings, all characters must be alphanumeric. ACPICA BZ 874.*

3155 iASL: Allow certain "null" resource descriptors. Some BIOS code creates  
3156 **descriptors that are mostly or all zeros, with the expectation that they**  
3157 **will**  
3158 **be filled in at runtime. iASL now allows this as long as there is a**  
3159 **"resource**  
3160 *471 descriptors that are mostly or all zeros, with the expectation that they will*  
3160 *472 be filled in at runtime. iASL now allows this as long as there is a "resource*  
3160 *tag" (name) associated with the descriptor, which gives the ASL a handle*  
3161 *needed to modify the descriptor. ACPICA BZ 873.*

3163 **Added single-thread support to the generic Unix application OSL.**  
3164 **Primarily**

3165 for iASL support, this change removes the use of semaphores in the  
 3166 single-  
 476 Added single-thread support to the generic Unix application OSL. Primarily  
 477 for iASL support, this change removes the use of semaphores in the single-  
 3167 threaded ACPICA tools/applications - increasing performance. The  
 3168 \_MULTI\_THREADED option was replaced by the (reverse) ACPI\_SINGLE\_THREADED  
 3169 option. ACPICA BZ 879.

3171 **AcpiExec: several fixes for the 64-bit version. Adds XSDT support and**  
 3172 **support**  
 482 AcpiExec: several fixes for the 64-bit version. Adds XSDT support and support  
 3173 for 64-bit DSDT/FACS addresses in the FADT. Lin Ming.

3175 iASL: Moved all compiler messages to a new file, aslmessages.h.

3177 -----  
 3178 15 September 2010. Summary of changes for version 20100915:

3180 1) ACPI CA Core Subsystem:

3182 **Removed the AcpiOsDerivePciId OSL interface. The various host**  
 3183 **implementations**  
 492 Removed the AcpiOsDerivePciId OSL interface. The various host implementations  
 3184 of this function were not OS-dependent and are now obsolete and can be  
 3185 removed from all host OSLs. This function has been replaced by  
 3186 AcpiHwDerivePciId, which is now part of the ACPICA core code.  
 3187 AcpiHwDerivePciId has been implemented without recursion. Adds one new  
 3188 module, hwpci.c. ACPICA BZ 857.

3190 Implemented a dynamic repair for \_HID and \_CID strings. The following  
 3191 problems are now repaired at runtime: 1) Remove a leading asterisk in the  
 3192 string, and 2) the entire string is uppercased. Both repairs are in  
 3193 accordance with the ACPI specification and will simplify host driver  
 3194 code.  
 502 accordance with the ACPI specification and will simplify host driver code.  
 3195 ACPICA BZ 871.

3197 The ACPI\_THREAD\_ID type is no longer configurable, internally it is now  
 3198 always UINT64. This simplifies the ACPICA code, especially any printf  
 3199 output.  
 506 always UINT64. This simplifies the ACPICA code, especially any printf output.  
 3200 UINT64 is the only common data type for all thread\_id types across all  
 3201 operating systems. It is now up to the host OSL to cast the native  
 3202 thread\_id  
 3203 type to UINT64 before returning the value to ACPICA (via  
 3204 AcpiOsGetThreadId).  
 508 operating systems. It is now up to the host OSL to cast the native thread\_id  
 509 type to UINT64 before returning the value to ACPICA (via AcpiOsGetThreadId).  
 3205 Lin Ming, Bob Moore.

3207 **Added the ACPI\_INLINE type to enhance the ACPICA configuration. The**  
 3208 **"inline"**  
 3209 keyword is not standard across compilers, and this type allows inline to  
 3210 be  
 512 Added the ACPI\_INLINE type to enhance the ACPICA configuration. The "inline"  
 513 keyword is not standard across compilers, and this type allows inline to be  
 3211 configured on a per-compiler basis. Lin Ming.

3213 **Made the system global AcpiGbl\_SystemAwakeAndRunning publically**  
 3214 **available.**  
 3215 Added an extern for this boolean in acpixf.h. Some hosts utilize this  
 3216 value  
 516 Made the system global AcpiGbl\_SystemAwakeAndRunning publically available.  
 517 Added an extern for this boolean in acpixf.h. Some hosts utilize this value  
 3217 during suspend/restore operations. ACPICA BZ 869.

3219 All code that implements error/warning messages with the "ACPI:" prefix  
 3220 has  
 520 All code that implements error/warning messages with the "ACPI:" prefix has  
 3221 been moved to a new module, utxferror.c.

3223 **The UINT64\_OVERLAY was moved to utmath.c, which is the only module where**  
 3224 **it**  
 523 The UINT64\_OVERLAY was moved to utmath.c, which is the only module where it  
 3225 is used. ACPICA BZ 829. Lin Ming, Bob Moore.

3227 Example Code and Data Size: These are the sizes for the OS-independent  
 3228 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 3229 debug version of the code includes the debug output trace mechanism and  
 3230 has a  
 528 debug version of the code includes the debug output trace mechanism and has a  
 3231 much larger code and data size.

3233 Previous Release:  
 3234 Non-Debug Version: 89.1K Code, 19.0K Data, 108.1K Total  
 3235 Debug Version: 165.1K Code, 51.9K Data, 217.0K Total  
 3236 Current Release:  
 3237 Non-Debug Version: 89.9K Code, 19.0K Data, 108.9K Total  
 3238 Debug Version: 166.3K Code, 52.1K Data, 218.4K Total

3240 2) iASL Compiler/Disassembler and Tools:

3242 **iASL/Disassembler: Write ACPI errors to stderr instead of the output**  
 3243 **file.**  
 3244 This keeps the output files free of random error messages that may  
 3245 originate  
 3246 from within the namespace/interpreter code. Used this opportunity to  
 3247 merge  
 540 iASL/Disassembler: Write ACPI errors to stderr instead of the output file.  
 541 This keeps the output files free of random error messages that may originate  
 542 from within the namespace/interpreter code. Used this opportunity to merge  
 3248 all ACPI:-style messages into a single new module, utxferror.c. ACPICA BZ  
 3249 866. Lin Ming, Bob Moore.

3251 **Tools: update some printf's for ansi warnings on size\_t. Handle width**  
 3252 **change**  
 546 Tools: update some printf's for ansi warnings on size\_t. Handle width change  
 3253 of size\_t on 32-bit versus 64-bit generations. Lin Ming.

3255 -----  
 3256 06 August 2010. Summary of changes for version 20100806:

3258 1) ACPI CA Core Subsystem:

3260 **Designed and implemented a new host interface to the \_OSI support code.**  
 3261 **This**  
 3262 will allow the host to dynamically add or remove multiple \_OSI strings,  
 3263 as  
 3264 well as install an optional handler that is called for each \_OSI  
 3265 invocation.  
 3266 Also added a new AML debugger command, 'osi' to display and modify the  
 3267 global  
 3268 \_OSI string table, and test support in the AcpiExec utility. See the  
 3269 ACPICA  
 554 Designed and implemented a new host interface to the \_OSI support code. This  
 555 will allow the host to dynamically add or remove multiple \_OSI strings, as  
 556 well as install an optional handler that is called for each \_OSI invocation.  
 557 Also added a new AML debugger command, 'osi' to display and modify the global  
 558 \_OSI string table, and test support in the AcpiExec utility. See the ACPICA  
 3270 reference manual for full details. Lin Ming, Bob Moore. ACPICA BZ 836.

3271 New Functions:  
 3272 AcpiInstallInterface - Add an \_OSI string.

```

3273     AcpiRemoveInterface - Delete an _OSI string.
3274     AcpiInstallInterfaceHandler - Install optional _OSI handler.
3275     Obsolete Functions:
3276     AcpiOsValidateInterface - no longer used.
3277     New Files:
3278     source/components/utilities/utosi.c

```

```

3280 Re-introduced the support to enable multi-byte transfers for Embedded
3281 Controller (EC) operation regions. A reported problem was found to be a
3282 bug
3283 in the host OS, not in the multi-byte support. Previously, the maximum
3284 data
3285 size passed to the EC operation region handler was a single byte. There
3286 are
3287 often EC Fields larger than one byte that need to be transferred, and it
3288 is
3289 useful for the EC driver to lock these as a single transaction. This
3290 change
3291 Controller (EC) operation regions. A reported problem was found to be a bug
3292 in the host OS, not in the multi-byte support. Previously, the maximum data
3293 size passed to the EC operation region handler was a single byte. There are
3294 often EC Fields larger than one byte that need to be transferred, and it is
3295 useful for the EC driver to lock these as a single transaction. This change
3296 enables single transfers larger than 8 bits. This effectively changes the
3297 access to the EC space from ByteAcc to AnyAcc, and will probably require
3298 changes to the host OS Embedded Controller driver to enable 16/32/64/256-bit
3299 bit
3300 changes to the host OS Embedded Controller driver to enable 16/32/64/256-bit
3301 transfers in addition to 8-bit transfers. Alexey Starikovskiy, Lin Ming.

```

```

3297 Fixed a problem with the prototype for AcpiOsReadPciConfiguration. The
3298 prototype in acpiosxf.h had the output value pointer as a (void *).
3299 It should be a (UINT64 *). This may affect some host OSL code.

```

```

3301 Fixed a couple problems with the recently modified Linux makefiles for
3302 iASL
3303 584 Fixed a couple problems with the recently modified Linux makefiles for iASL
3304 and AcpiExec. These new makefiles place the generated object files in the
3305 local directory so that there can be no collisions between the files that
3306 are
3307 586 local directory so that there can be no collisions between the files that are
3308 shared between them that are compiled with different options.

```

```

3308 Example Code and Data Size: These are the sizes for the OS-independent
3309 apica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The
3310 debug version of the code includes the debug output trace mechanism and
3311 has a
3312 591 debug version of the code includes the debug output trace mechanism and has a
3313 much larger code and data size.

```

```

3314 Previous Release:
3315 Non-Debug Version: 88.3K Code, 18.8K Data, 107.1K Total
3316 Debug Version: 164.0K Code, 51.5K Data, 215.5K Total
3317 Current Release:
3318 Non-Debug Version: 89.1K Code, 19.0K Data, 108.1K Total
3319 Debug Version: 165.1K Code, 51.9K Data, 217.0K Total

```

```

3321 2) iASL Compiler/Disassembler and Tools:

```

```

3323 iASL/Disassembler: Added a new option (-da, "disassemble all") to load
3324 the
3325 iASL/Disassembler: Added a new option (-da, "disassemble all") to load the
3326 namespace from and disassemble an entire group of AML files. Useful for
3327 loading all of the AML tables for a given machine (DSDT, SSDT1...SSDTn)
3328 and
3329 605 loading all of the AML tables for a given machine (DSDT, SSDT1...SSDTn) and

```

```

3328 disassembling with one simple command. ACPICA BZ 865. Lin Ming.

```

```

3330 iASL: Allow multiple invocations of -e option. This change allows
3331 multiple
3332 uses of -e on the command line: "-e ssdt1.dat -e ssdt2.dat". ACPICA BZ
3333 834.
3334 608 iASL: Allow multiple invocations of -e option. This change allows multiple
3335 609 uses of -e on the command line: "-e ssdt1.dat -e ssdt2.dat". ACPICA BZ 834.
3336 Lin Ming.

```

```

3336 -----
3337 02 July 2010. Summary of changes for version 20100702:

```

```

3339 1) ACPI CA Core Subsystem:

```

```

3341 Implemented several updates to the recently added GPE reference count
3342 support. The model for "wake" GPEs is changing to give the host OS
3343 complete
3344 control of these GPEs. Eventually, the ACPICA core will not execute any
3345 _PRW
3346 methods, since the host already must execute them. Also, additional
3347 changes
3348 618 support. The model for "wake" GPEs is changing to give the host OS complete
3349 619 control of these GPEs. Eventually, the ACPICA core will not execute any _PRW
3350 620 methods, since the host already must execute them. Also, additional changes
3351 were made to help ensure that the reference counts are kept in proper
3352 synchronization with reality. Rafael J. Wysocki.

```

```

3351 1) Ensure that GPEs are not enabled twice during initialization.
3352 2) Ensure that GPE enable masks stay in sync with the reference count.
3353 3) Do not inadvertently enable GPEs when writing GPE registers.
3354 4) Remove the internal wake reference counter and add new AcpiGpeWakeUp
3355 interface. This interface will set or clear individual GPEs for wakeup.
3356 5) Remove GpeType argument from AcpiEnable and AcpiDisable. These
3357 interfaces
3358 629 5) Remove GpeType argument from AcpiEnable and AcpiDisable. These interfaces
3359 are now used for "runtime" GPEs only.

```

```

3360 Changed the behavior of the GPE install/remove handler interfaces. The
3361 GPE
3362 is
3363 no longer disabled during this process, as it was found to cause problems
3364 on
3365 632 Changed the behavior of the GPE install/remove handler interfaces. The GPE is
3366 633 no longer disabled during this process, as it was found to cause problems on
3367 some machines. Rafael J. Wysocki.

```

```

3367 Reverted a change introduced in version 20100528 to enable Embedded
3368 Controller multi-byte transfers. This change was found to cause problems
3369 with
3370 Controller multi-byte transfers. This change was found to cause problems with
3371 Index Fields and possibly Bank Fields. It will be reintroduced when these
3372 problems have been resolved.

```

```

3373 Fixed a problem with references to Alias objects within Package Objects.
3374 A
3375 641 Fixed a problem with references to Alias objects within Package Objects. A
3376 reference to an Alias within the definition of a Package was not always
3377 resolved properly. Aliases to objects like Processors, Thermal zones,
3378 etc.
3379 were resolved to the actual object instead of a reference to the object
3380 as
3381 643 resolved properly. Aliases to objects like Processors, Thermal zones, etc.
3382 644 were resolved to the actual object instead of a reference to the object as it
3383 should be. Package objects are only allowed to contain integer, string,

```

3382 buffer, package, and reference objects. Redhat bugzilla 608648.

3384 Example Code and Data Size: These are the sizes for the OS-independent  
 3385 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 3386 **debug version of the code includes the debug output trace mechanism and**  
 3387 **has a**  
 650 *debug version of the code includes the debug output trace mechanism and has a*  
 3388 much larger code and data size.

3390 Previous Release:  
 3391 Non-Debug Version: 88.3K Code, 18.8K Data, 107.1K Total  
 3392 Debug Version: 164.1K Code, 51.5K Data, 215.6K Total  
 3393 Current Release:  
 3394 Non-Debug Version: 88.3K Code, 18.8K Data, 107.1K Total  
 3395 Debug Version: 164.0K Code, 51.5K Data, 215.5K Total

3397 2) iASL Compiler/Disassembler and Tools:

3399 iASL: Implemented a new compiler subsystem to allow definition and  
 3400 **compilation of the non-AML ACPI tables such as FADT, MADT, SRAT, etc.**  
 3401 **These**  
 663 *compilation of the non-AML ACPI tables such as FADT, MADT, SRAT, etc. These*  
 3402 are called "ACPI Data Tables", and the new compiler is the "Data Table  
 3403 Compiler". This compiler is intended to simplify the existing error-prone  
 3404 process of creating these tables for the BIOS, as well as allowing the  
 3405 **disassembly, modification, recompilation, and override of existing ACPI**  
 3406 **data**  
 667 *disassembly, modification, recompilation, and override of existing ACPI data*  
 3407 tables. See the iASL User Guide for detailed information.

3409 iASL: Implemented a new Template Generator option in support of the new  
 3410 **Data**  
 670 *iASL: Implemented a new Template Generator option in support of the new Data*  
 3411 Table Compiler. This option will create examples of all known ACPI tables  
 3412 that can be used as the basis for table development. See the iASL  
 3413 documentation and the -T option.

3415 Disassembler and headers: Added support for the WDDT ACPI table (Watchdog  
 3416 Descriptor Table).

3418 Updated the Linux makefiles for iASL and AcpiExec to place the generated  
 3419 object files in the local directory so that there can be no collisions  
 3420 between the shared files between them that are generated with different  
 3421 options.

3423 **Added support for Mac OS X in the Unix OSL used for iASL and AcpiExec.**  
 3424 **Use**  
 683 *Added support for Mac OS X in the Unix OSL used for iASL and AcpiExec. Use*  
 3425 the #define \_\_APPLE\_\_ to enable this support.

3427 -----  
 3428 28 May 2010. Summary of changes for version 20100528:

3430 Note: The ACPI 4.0a specification was released on April 5, 2010 and is  
 3431 available at [www.acpi.info](http://www.acpi.info). This is primarily an errata release.

3433 1) ACPI CA Core Subsystem:

3435 **Undefined ACPI tables: We are looking for the definitions for the**  
 3436 **following**  
 694 *Undefined ACPI tables: We are looking for the definitions for the following*  
 3437 ACPI tables that have been seen in the field: ATKG, IEIT, GSCI.

3439 **Implemented support to enable multi-byte transfers for Embedded**  
 3440 **Controller**  
 3441 **(EC) operation regions. Previously, the maximum data size passed to the**

3442 **EC**  
 3443 **operation region handler was a single byte. There are often EC Fields**  
 3444 **larger**  
 3445 **than one byte that need to be transferred, and it is useful for the EC**  
 3446 **driver**  
 3447 **to lock these as a single transaction. This change enables single**  
 3448 **transfers**  
 3449 **larger than 8 bits. This effectively changes the access to the EC space**  
 3450 **from**  
 3451 **ByteAcc to AnyAcc, and will probably require changes to the host OS**  
 3452 **Embedded**  
 3453 **Controller driver to enable 16/32/64/256-bit transfers in addition to 8-**  
 3454 **bit**  
 697 *Implemented support to enable multi-byte transfers for Embedded Controller*  
 698 *(EC) operation regions. Previously, the maximum data size passed to the EC*  
 699 *operation region handler was a single byte. There are often EC Fields larger*  
 700 *than one byte that need to be transferred, and it is useful for the EC driver*  
 701 *to lock these as a single transaction. This change enables single transfers*  
 702 *larger than 8 bits. This effectively changes the access to the EC space from*  
 703 *ByteAcc to AnyAcc, and will probably require changes to the host OS Embedded*  
 704 *Controller driver to enable 16/32/64/256-bit transfers in addition to 8-bit*  
 3455 transfers. Alexey Starikovskiy, Lin Ming

3457 **Implemented a performance enhancement for namespace search and access.**  
 3458 **This**  
 3459 **change enhances the performance of namespace searches and walks by adding**  
 3460 **a**  
 3461 **backpointer to the parent in each namespace node. On large namespaces,**  
 3462 **this**  
 3463 **change can improve overall ACPI performance by up to 9X. Adding a pointer**  
 3464 **to**  
 3465 **each namespace node increases the overall size of the internal namespace**  
 3466 **by**  
 707 *Implemented a performance enhancement for namespace search and access. This*  
 708 *change enhances the performance of namespace searches and walks by adding a*  
 709 *backpointer to the parent in each namespace node. On large namespaces, this*  
 710 *change can improve overall ACPI performance by up to 9X. Adding a pointer to*  
 711 *each namespace node increases the overall size of the internal namespace by*  
 3467 *about 5%, since each namespace entry usually consists of both a namespace*  
 3468 *node and an ACPI operand object. However, this is the first growth of the*  
 3469 namespace in ten years. ACPICA bugzilla 817. Alexey Starikovskiy.

3471 **Implemented a performance optimization that reduces the number of**  
 3472 **namespace**  
 3473 **walks. On control method exit, only walk the namespace if the method is**  
 3474 **known**  
 3475 **to have created namespace objects outside of its local scope. Previously,**  
 3476 **the**  
 3477 **entire namespace was traversed on each control method exit. This change**  
 3478 **can**  
 3479 **improve overall ACPI performance by up to 3X. Alexey Starikovskiy, Bob**  
 3480 **Moore.**  
 716 *Implemented a performance optimization that reduces the number of namespace*  
 717 *walks. On control method exit, only walk the namespace if the method is known*  
 718 *to have created namespace objects outside of its local scope. Previously, the*  
 719 *entire namespace was traversed on each control method exit. This change can*  
 720 *improve overall ACPI performance by up to 3X. Alexey Starikovskiy, Bob Moore.*

3482 **Added support to truncate I/O addresses to 16 bits for Windows**  
 3483 **compatibility.**  
 722 *Added support to truncate I/O addresses to 16 bits for Windows compatibility.*  
 3484 Some ASL code has been seen in the field that inadvertently has bits set  
 3485 **above bit 15. This feature is optional and is enabled if the BIOS**  
 3486 **requests**  
 724 *above bit 15. This feature is optional and is enabled if the BIOS requests*  
 3487 any Windows OSI strings. It can also be enabled by the host OS. Matthew



3488 Garrett, Bob Moore.

3490 Added support to limit the maximum time for the ASL Sleep() operator. To  
3491 prevent accidental deep sleeps, limit the maximum time that Sleep() will  
3492 actually sleep. Configurable, the default maximum is two seconds. ACPICA  
3493 bugzilla 854.

3495 **Added run-time validation support for the \_WDG and \_WED Microsoft**  
3496 **predefined**  
3497 **methods. These objects are defined by "Windows Instrumentation", and are**  
3498 **not**  
733 *Added run-time validation support for the \_WDG and \_WED Microsoft predefined*  
734 *methods. These objects are defined by "Windows Instrumentation", and are not*  
3499 *part of the ACPI spec. ACPICA BZ 860.*

3501 Expanded all statistic counters used during namespace and device  
3502 **initialization from 16 to 32 bits in order to support very large**  
3503 **namespaces.**  
738 *initialization from 16 to 32 bits in order to support very large namespaces.*

3505 **Replaced all instances of %d in printf format specifiers with %u since**  
3506 **nearly**  
740 *Replaced all instances of %d in printf format specifiers with %u since nearly*  
3507 *all integers in ACPICA are unsigned.*

3509 **Fixed the exception namestring for AE\_WAKE\_ONLY\_GPE. Was incorrectly**  
3510 **returned**  
743 *Fixed the exception namestring for AE\_WAKE\_ONLY\_GPE. Was incorrectly returned*  
3511 *as AE\_NO\_HANDLER.*

3513 Example Code and Data Size: These are the sizes for the OS-independent  
3514 apica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
3515 **debug version of the code includes the debug output trace mechanism and**  
3516 **has a**  
748 *debug version of the code includes the debug output trace mechanism and has a*  
3517 *much larger code and data size.*

3519	Previous Release:
3520	Non-Debug Version: 88.4K Code, 18.8K Data, 107.2K Total
3521	Debug Version: 164.2K Code, 51.5K Data, 215.7K Total
3522	Current Release:
3523	Non-Debug Version: 88.3K Code, 18.8K Data, 107.1K Total
3524	Debug Version: 164.1K Code, 51.5K Data, 215.6K Total

3526 2) iASL Compiler/Disassembler and Tools:

3528 iASL: Added compiler support for the \_WDG and \_WED Microsoft predefined  
3529 **methods. These objects are defined by "Windows Instrumentation", and are**  
3530 **not**  
761 *methods. These objects are defined by "Windows Instrumentation", and are not*  
3531 *part of the ACPI spec. ACPICA BZ 860.*

3533 AcpiExec: added option to disable the memory tracking mechanism. The -dt  
3534 option will disable the tracking mechanism, which improves performance  
3535 considerably.

3537 AcpiExec: Restructured the command line options into -d (disable) and -e  
3538 (enable) options.

3540 -----  
3541 28 April 2010. Summary of changes for version 20100428:

3543 1) ACPI CA Core Subsystem:

3545 Implemented GPE support for dynamically loaded ACPI tables. For all GPEs,  
3546 **including FADT-based and GPE Block Devices, execute any \_PRW methods in**

3547 **the**  
777 *including FADT-based and GPE Block Devices, execute any \_PRW methods in the*  
3548 *new table, and process any \_Lxx/\_Exxx GPE methods in the new table. Any*  
3549 *runtime GPE that is referenced by an \_Lxx/\_Exxx method in the new table is*  
3550 *immediately enabled. Handles the FADT-defined GPEs as well as GPE Block*  
3551 *Devices. Provides compatibility with other ACPI implementations. Two new*  
3552 **files added, evgpeinit.c and evgpeutil.c. ACPICA BZ 833. Lin Ming, Bob**  
3553 **Moore.**  
782 *files added, evgpeinit.c and evgpeutil.c. ACPICA BZ 833. Lin Ming, Bob Moore.*

3555 **Fixed a regression introduced in version 20100331 within the table**  
3556 **manager**  
3557 **where initial table loading could fail. This was introduced in the fix**  
3558 **for**  
3559 **AcpiReallocateRootTable. Also, renamed some of fields in the table**  
3560 **manager**  
784 *Fixed a regression introduced in version 20100331 within the table manager*  
785 *where initial table loading could fail. This was introduced in the fix for*  
786 *AcpiReallocateRootTable. Also, renamed some of fields in the table manager*  
3561 *data structures to clarify their meaning and use.*

3563 Fixed a possible allocation overrun during internal object copy in  
3564 **AcpiUtCopySimpleObject. The original code did not correctly handle the**  
3565 **case**  
3566 **where the object to be copied was a namespace node. Lin Ming. ACPICA BZ**  
3567 **847.**  
790 *AcpiUtCopySimpleObject. The original code did not correctly handle the case*  
791 *where the object to be copied was a namespace node. Lin Ming. ACPICA BZ 847.*

3569 Updated the allocation dump routine, AcpiUtDumpAllocation and fixed a  
3570 **possible access beyond end-of-allocation. Also, now fully validate**  
3571 **descriptor**  
794 *possible access beyond end-of-allocation. Also, now fully validate descriptor*  
3572 *(size and type) before output. Lin Ming, Bob Moore. ACPICA BZ 847*

3574 Example Code and Data Size: These are the sizes for the OS-independent  
3575 apica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
3576 **debug version of the code includes the debug output trace mechanism and**  
3577 **has a**  
799 *debug version of the code includes the debug output trace mechanism and has a*  
3578 *much larger code and data size.*

3580	Previous Release:
3581	Non-Debug Version: 87.9K Code, 18.6K Data, 106.5K Total
3582	Debug Version: 163.5K Code, 51.3K Data, 214.8K Total
3583	Current Release:
3584	Non-Debug Version: 88.4K Code, 18.8K Data, 107.2K Total
3585	Debug Version: 164.2K Code, 51.5K Data, 215.7K Total

3587 2) iASL Compiler/Disassembler and Tools:

3589 iASL: Implemented Min/Max/Len/Gran validation for address resource  
3590 **descriptors. This change implements validation for the address fields**  
3591 **that**  
812 *descriptors. This change implements validation for the address fields that*  
3592 *are common to all address-type resource descriptors. These checks are*  
3593 *implemented: Checks for valid Min/Max, length within the Min/Max window,*  
3594 **valid granularity, Min/Max a multiple of granularity, and \_MIF/\_MAF as**  
3595 **per**  
3596 **table 6-40 in the ACPI 4.0a specification. Also split the large**  
3597 **aslstypel.c**  
815 *valid granularity, Min/Max a multiple of granularity, and \_MIF/\_MAF as per*  
816 *table 6-40 in the ACPI 4.0a specification. Also split the large aslstypel.c*  
3598 *and aslstype2.c files into five new files. ACPICA BZ 840.*

3600 iASL: Added support for the \_Wxxx predefined names. This support was

3601 **missing**  
 819 *iASL: Added support for the \_Wxx predefined names. This support was missing*  
 3602 and these names were not recognized by the compiler as valid predefined  
 3603 names. ACPICA BZ 851.

3605 **iASL: Added an error for all predefined names that are defined to return**  
 3606 **no**  
 3607 **value and thus must be implemented as Control Methods. These include all**  
 3608 **of**  
 823 *iASL: Added an error for all predefined names that are defined to return no*  
 824 *value and thus must be implemented as Control Methods. These include all of*  
 3609 *the \_Lxx, \_Exx, \_Wxx, and \_Qxx names, as well as some other miscellaneous*  
 3610 *names such as \_DIS, \_INI, \_IRC, \_OFF, \_ON, and \_PSx. ACPICA BZ 850, 856.*

3612 **iASL: Implemented the -ts option to emit hex AML data in ASL format, as**  
 3613 **an**  
 3614 **ASL Buffer. Allows ACPI tables to be easily included within ASL files, to**  
 3615 **be**  
 3616 **dynamically loaded via the Load() operator. Also cleaned up output for**  
 3617 **the**  
 3618 **-**  
 828 *iASL: Implemented the -ts option to emit hex AML data in ASL format, as an*  
 829 *ASL Buffer. Allows ACPI tables to be easily included within ASL files, to be*  
 830 *dynamically loaded via the Load() operator. Also cleaned up output for the -*  
 3619 *ta and -tc options. ACPICA BZ 853.*

3621 **Tests: Added a new file with examples of extended iASL error checking.**  
 3622 **Demonstrates the advanced error checking ability of the iASL compiler.**  
 3623 **Available at tests/misc/badcode.asl.**

3625 -----  
 3626 31 March 2010. Summary of changes for version 20100331:

3628 1) ACPI CA Core Subsystem:

3630 **Completed a major update for the GPE support in order to improve support**  
 3631 **for**  
 3632 **shared GPEs and to simplify both host OS and ACPICA code. Added a**  
 3633 **reference**  
 3634 **count mechanism to support shared GPEs that require multiple device**  
 3635 **drivers.**  
 842 *Completed a major update for the GPE support in order to improve support for*  
 843 *shared GPEs and to simplify both host OS and ACPICA code. Added a reference*  
 844 *count mechanism to support shared GPEs that require multiple device drivers.*  
 3636 *Several external interfaces have changed. One external interface has been*  
 3637 *removed. One new external interface was added. Most of the GPE external*  
 3638 *interfaces now use the GPE spinlock instead of the events mutex (and the*  
 3639 *Flags parameter for many GPE interfaces has been removed.) See the*  
 3640 *updated*  
 3641 *ACPICA Programmer Reference for details. Matthew Garrett, Bob Moore,*  
 3642 *Rafael*  
 848 *Flags parameter for many GPE interfaces has been removed.) See the updated*  
 849 *ACPICA Programmer Reference for details. Matthew Garrett, Bob Moore, Rafael*  
 3643 *Wysocki. ACPICA BZ 831.*

3645 **Changed:**  
 3646 **AcpiEnableGpe, AcpiDisableGpe, AcpiClearGpe, AcpiGetGpeStatus**  
 3647 **Removed:**  
 3648 **AcpiSetGpeType**  
 3649 **New:**  
 3650 **AcpiSetGpe**

3652 **Implemented write support for DataTable operation regions. These regions**  
 3653 **are**  
 3654 **defined via the DataTableRegion() operator. Previously, only read support**  
 3655 **was**

3656 **implemented. The ACPI specification allows DataTableRegions to be**  
 3657 **read/write,**  
 859 *Implemented write support for DataTable operation regions. These regions are*  
 860 *defined via the DataTableRegion() operator. Previously, only read support was*  
 861 *implemented. The ACPI specification allows DataTableRegions to be read/write,*  
 3658 *however.*

3660 **Implemented a new subsystem option to force a copy of the DSDT to local**  
 3661 **memory. Optionally copy the entire DSDT to local memory (instead of**  
 3662 **simply**  
 3663 **mapping it.) There are some (albeit very rare) BIOSs that corrupt or**  
 3664 **replace**  
 3665 **the original DSDT, creating the need for this option. Default is FALSE,**  
 3666 **do**  
 865 *memory. Optionally copy the entire DSDT to local memory (instead of simply*  
 866 *mapping it.) There are some (albeit very rare) BIOSs that corrupt or replace*  
 867 *the original DSDT, creating the need for this option. Default is FALSE, do*  
 3667 *not copy the DSDT.*

3669 **Implemented detection of a corrupted or replaced DSDT. This change adds**  
 3670 **support to detect a DSDT that has been corrupted and/or replaced from**  
 3671 **outside**  
 3672 **the OS (by firmware). This is typically catastrophic for the system, but**  
 3673 **has**  
 871 *support to detect a DSDT that has been corrupted and/or replaced from outside*  
 872 *the OS (by firmware). This is typically catastrophic for the system, but has*  
 3674 *been seen on some machines. Once this problem has been detected, the DSDT*  
 3675 *copy option can be enabled via system configuration. Lin Ming, Bob Moore.*

3677 **Fixed two problems with AcpiReallocateRootTable during the root table**  
 3678 **copy.**  
 876 *Fixed two problems with AcpiReallocateRootTable during the root table copy.*  
 3679 *When copying the root table to the new allocation, the length used was*  
 3680 *incorrect. The new size was used instead of the current table size,*  
 3681 *meaning*  
 3682 *too much data was copied. Also, the count of available slots for ACPI*  
 3683 *tables*  
 878 *incorrect. The new size was used instead of the current table size, meaning*  
 879 *too much data was copied. Also, the count of available slots for ACPI tables*  
 3684 *was not set correctly. Alexey Starikovskiy, Bob Moore.*

3686 **Example Code and Data Size: These are the sizes for the OS-independent**  
 3687 **apica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The**  
 3688 **debug version of the code includes the debug output trace mechanism and**  
 3689 **has a**  
 884 *debug version of the code includes the debug output trace mechanism and has a*  
 3690 *much larger code and data size.*

3692 **Previous Release:**  
 3693 **Non-Debug Version: 87.5K Code, 18.4K Data, 105.9K Total**  
 3694 **Debug Version: 163.4K Code, 51.1K Data, 214.5K Total**  
 3695 **Current Release:**  
 3696 **Non-Debug Version: 87.9K Code, 18.6K Data, 106.5K Total**  
 3697 **Debug Version: 163.5K Code, 51.3K Data, 214.8K Total**

3699 2) iASL Compiler/Disassembler and Tools:

3701 **iASL: Implement limited typechecking for values returned from predefined**  
 3702 **control methods. The type of any returned static (unnamed) object is now**  
 3703 **validated. For example, Return(1). ACPICA BZ 786.**

3705 **iASL: Fixed a predefined name object verification regression. Fixes a**  
 3706 **problem**  
 900 *iASL: Fixed a predefined name object verification regression. Fixes a problem*  
 3707 *introduced in version 20100304. An error is incorrectly generated if a*  
 3708 *predefined name is declared as a static named object with a value defined*

3709 using the keywords "Zero", "One", or "Ones". Lin Ming.

3711 **iASL: Added Windows 7 support for the -g option (get local ACPI tables)**  
 3712 **by**  
 905 *iASL: Added Windows 7 support for the -g option (get local ACPI tables) by*  
 3713 *reducing the requested registry access rights. ACPICA BZ 842.*

3715 **Disassembler: fixed a possible fault when generating External()**  
 3716 **statements.**  
 3717 **Introduced in commit ae7d6fd: Properly handle externals with parent-**  
 3718 **prefix**  
 908 *Disassembler: fixed a possible fault when generating External() statements.*  
 909 *Introduced in commit ae7d6fd: Properly handle externals with parent-prefix*  
 3719 *(carat). Fixes a string length allocation calculation. Lin Ming.*

3721 -----  
 3722 04 March 2010. Summary of changes for version 20100304:

3724 1) ACPI CA Core Subsystem:

3726 Fixed a possible problem with the AML Mutex handling function  
 3727 AcpiExReleaseMutex where the function could fault under the very rare  
 3728 **condition when the interpreter has blocked, the interpreter lock is**  
 3729 **released,**  
 919 *condition when the interpreter has blocked, the interpreter lock is released,*  
 3730 *the interpreter is then reentered via the same thread, and attempts to*  
 3731 **acquire an AML mutex that was previously acquired. FreeBSD report 140979.**  
 3732 **Lin**  
 921 *acquire an AML mutex that was previously acquired. FreeBSD report 140979. Lin*  
 3733 *Ming.*

3735 Implemented additional configuration support for the AML "Debug Object".  
 3736 Output from the debug object can now be enabled via a global variable,  
 3737 **AcpiGbl\_EnableAmlDebugObject. This will assist with remote machine**  
 3738 **debugging.**  
 3739 **This debug output is now available in the release version of ACPICA**  
 3740 **instead**  
 3741 **of just the debug version. Also, the entire debug output module can now**  
 3742 **be**  
 926 *AcpiGbl\_EnableAmlDebugObject. This will assist with remote machine debugging.*  
 927 *This debug output is now available in the release version of ACPICA instead*  
 928 *of just the debug version. Also, the entire debug output module can now be*  
 3743 *configured out of the ACPICA build if desired. One new file added,*  
 3744 *executer/exdebug.c. Lin Ming, Bob Moore.*

3746 Added header support for the ACPI MCHI table (Management Controller Host  
 3747 **Interface Table). This table was added in ACPI 4.0, but the defining**  
 3748 **document**  
 933 *Interface Table). This table was added in ACPI 4.0, but the defining document*  
 3749 *has only recently become available.*

3751 **Standardized output of integer values for ACPICA warnings/errors. Always**  
 3752 **use**  
 3753 **0x prefix for hex output, always use %u for unsigned integer decimal**  
 3754 **output.**  
 3755 **Affects ACPI\_INFO, ACPI\_ERROR, ACPI\_EXCEPTION, and ACPI\_WARNING (about**  
 3756 **400**  
 936 *Standardized output of integer values for ACPICA warnings/errors. Always use*  
 937 *0x prefix for hex output, always use %u for unsigned integer decimal output.*  
 938 *Affects ACPI\_INFO, ACPI\_ERROR, ACPI\_EXCEPTION, and ACPI\_WARNING (about 400*  
 3757 *invocations.) These invocations were converted from the original*  
 3758 *ACPI\_DEBUG\_PRINT invocations and were not consistent. ACPICA BZ 835.*

3760 Example Code and Data Size: These are the sizes for the OS-independent  
 3761 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 3762 **debug version of the code includes the debug output trace mechanism and**

3763 **has a**  
 944 *debug version of the code includes the debug output trace mechanism and has a*  
 3764 *much larger code and data size.*

3766 Previous Release:  
 3767 Non-Debug Version: 87.1K Code, 18.0K Data, 105.1K Total  
 3768 Debug Version: 163.5K Code, 50.9K Data, 214.4K Total  
 3769 Current Release:  
 3770 Non-Debug Version: 87.5K Code, 18.4K Data, 105.9K Total  
 3771 Debug Version: 163.4K Code, 51.1K Data, 214.5K Total

3773 2) iASL Compiler/Disassembler and Tools:

3775 iASL: Implemented typechecking support for static (non-control method)  
 3776 predefined named objects that are declared with the Name() operator. For  
 3777 example, the type of this object is now validated to be of type Integer:  
 3778 **Name(\_BBN, 1). This change migrates the compiler to using the core**  
 3779 **predefined**  
 959 *Name(\_BBN, 1). This change migrates the compiler to using the core predefined*  
 3780 *name table instead of maintaining a local version. Added a new file,*  
 3781 *aslpredef.c. ACPICA BZ 832.*

3783 Disassembler: Added support for the ACPI 4.0 MCHI table.

3785 -----  
 3786 21 January 2010. Summary of changes for version 20100121:

3788 1) ACPI CA Core Subsystem:

3790 Added the 2010 copyright to all module headers and signons. This affects  
 3791 virtually every file in the ACPICA core subsystem, the iASL compiler, the  
 3792 tools/utilities, and the test suites.

3794 **Implemented a change to the AcpiGetDevices interface to eliminate**  
 3795 **unnecessary**  
 974 *Implemented a change to the AcpiGetDevices interface to eliminate unnecessary*  
 3796 *invocations of the \_STA method. In the case where a specific \_HID is*  
 3797 *requested, do not run \_STA until a \_HID match is found. This eliminates*  
 3798 **potentially dozens of \_STA calls during a search for a particular**  
 3799 **device/HID,**  
 977 *potentially dozens of \_STA calls during a search for a particular device/HID,*  
 3800 *which in turn can improve boot times. ACPICA BZ 828. Lin Ming.*

3802 **Implemented an additional repair for predefined method return values.**  
 3803 **Attempt**  
 3804 **to repair unexpected NULL elements within returned Package objects.**  
 3805 **Create**  
 3806 **an**  
 3807 **Integer of value zero, a NULL String, or a zero-length Buffer as**  
 3808 **appropriate.**  
 980 *Implemented an additional repair for predefined method return values. Attempt*  
 981 *to repair unexpected NULL elements within returned Package objects. Create an*  
 982 *Integer of value zero, a NULL String, or a zero-length Buffer as appropriate.*  
 3809 *ACPICA BZ 818. Lin Ming, Bob Moore.*

3811 **Removed the obsolete ACPI\_INTEGER data type. This type was introduced as**  
 3812 **the**  
 3813 **code was migrated from ACPI 1.0 (with 32-bit AML integers) to ACPI 2.0**  
 3814 **(with**  
 3815 **64-bit AML integers). It is now obsolete and this change removes it from**  
 3816 **the**  
 3817 **ACPICA code base, replaced by UINT64. The original typedef has been**  
 3818 **retained**  
 3819 **for now for compatibility with existing device driver code. ACPICA BZ**  
 3820 **824.**  
 985 *Removed the obsolete ACPI\_INTEGER data type. This type was introduced as the*

986 code was migrated from ACPI 1.0 (with 32-bit AML integers) to ACPI 2.0 (with  
987 64-bit AML integers). It is now obsolete and this change removes it from the  
988 ACPICA code base, replaced by UINT64. The original typedef has been retained  
989 for now for compatibility with existing device driver code. ACPICA BZ 824.

3822 Removed the unused UINT32\_STRUCT type, and the obsolete Integer64 field  
3823 in  
991 Removed the unused UINT32\_STRUCT type, and the obsolete Integer64 field in  
3824 the parse tree object.

3826 Added additional warning options for the gcc-4 generation. Updated the  
3827 source  
3828 accordingly. This includes some code restructuring to eliminate  
3829 unreachable  
3830 code, elimination of some gotos, elimination of unused return values,  
3831 some  
994 Added additional warning options for the gcc-4 generation. Updated the source  
995 accordingly. This includes some code restructuring to eliminate unreachable  
996 code, elimination of some gotos, elimination of unused return values, some  
3832 additional casting, and removal of redundant declarations.

3834 Example Code and Data Size: These are the sizes for the OS-independent  
3835 apica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
3836 debug version of the code includes the debug output trace mechanism and  
3837 has a  
1001 debug version of the code includes the debug output trace mechanism and has a  
3838 much larger code and data size.

3840 Previous Release:  
3841 Non-Debug Version: 87.0K Code, 18.0K Data, 105.0K Total  
3842 Debug Version: 163.4K Code, 50.8K Data, 214.2K Total  
3843 Current Release:  
3844 Non-Debug Version: 87.1K Code, 18.0K Data, 105.1K Total  
3845 Debug Version: 163.5K Code, 50.9K Data, 214.4K Total

3847 2) iASL Compiler/Disassembler and Tools:

3849 No functional changes for this release.

3851 -----  
3852 14 December 2009. Summary of changes for version 20091214:

3854 1) ACPI CA Core Subsystem:

3856 Enhanced automatic data type conversions for predefined name repairs.  
3857 This  
3858 change expands the automatic repairs/conversions for predefined name  
3859 return  
3860 values to make Integers, Strings, and Buffers fully interchangeable.  
3861 Also,  
3862 a  
3863 Buffer can be converted to a Package of Integers if necessary. The  
3864 nsrepair.c  
1020 Enhanced automatic data type conversions for predefined name repairs. This  
1021 change expands the automatic repairs/conversions for predefined name return  
1022 values to make Integers, Strings, and Buffers fully interchangeable. Also, a  
1023 Buffer can be converted to a Package of Integers if necessary. The nsrepair.c  
3865 module was completely restructured. Lin Ming, Bob Moore.

3867 Implemented automatic removal of null package elements during predefined  
3868 name  
1026 Implemented automatic removal of null package elements during predefined name  
3869 repairs. This change will automatically remove embedded and trailing NULL  
3870 package elements from returned package objects that are defined to  
3871 contain  
3872 a

3873 variable number of sub-packages. The driver is then presented with a  
3874 package  
1028 package elements from returned package objects that are defined to contain a  
1029 variable number of sub-packages. The driver is then presented with a package  
3875 with no null elements to deal with. ACPICA BZ 819.

3877 Implemented a repair for the predefined \_FDE and \_GTM names. The expected  
3878 return value for both names is a Buffer of 5 DWORDs. This repair fixes  
3879 two  
3880 possible problems (both seen in the field), where a package of integers  
3881 is  
3882 returned, or a buffer of BYTES is returned. With assistance from Jung-uk  
3883 Kim.  
1033 return value for both names is a Buffer of 5 DWORDs. This repair fixes two  
1034 possible problems (both seen in the field), where a package of integers is  
1035 returned, or a buffer of BYTES is returned. With assistance from Jung-uk Kim.

3885 Implemented additional module-level code support. This change will  
3886 properly  
3887 execute module-level code that is not at the root of the namespace (under  
3888 a  
3889 Device object, etc.). Now executes the code within the current scope  
3890 instead  
1037 Implemented additional module-level code support. This change will properly  
1038 execute module-level code that is not at the root of the namespace (under a  
1039 Device object, etc.). Now executes the code within the current scope instead  
3891 of the root. ACPICA BZ 762. Lin Ming.

3893 Fixed possible mutex acquisition errors when running \_REG methods. Fixes  
3894 a  
3895 problem where mutex errors can occur when running a \_REG method that is  
3896 in  
3897 the same scope as a method-defined operation region or an operation  
3898 region  
3899 under a module-level IF block. This type of code is rare, so the problem  
3900 has  
1042 Fixed possible mutex acquisition errors when running \_REG methods. Fixes a  
1043 problem where mutex errors can occur when running a \_REG method that is in  
1044 the same scope as a method-defined operation region or an operation region  
1045 under a module-level IF block. This type of code is rare, so the problem has  
3901 not been seen before. ACPICA BZ 826. Lin Ming, Bob Moore.

3903 Fixed a possible memory leak during module-level code execution. An  
3904 object  
1048 Fixed a possible memory leak during module-level code execution. An object  
3905 could be leaked for each block of executed module-level code if the  
3906 interpreter slack mode is enabled This change deletes any implicitly  
3907 returned  
1050 interpreter slack mode is enabled This change deletes any implicitly returned  
3908 object from the module-level code block. Lin Ming.

3910 Removed messages for successful predefined repair(s). The repair  
3911 mechanism  
3912 was considered too wordy. Now, messages are only unconditionally emitted  
3913 if  
1053 Removed messages for successful predefined repair(s). The repair mechanism  
1054 was considered too wordy. Now, messages are only unconditionally emitted if  
3914 the return object cannot be repaired. Existing messages for successful  
3915 repairs were converted to ACPI\_DEBUG\_PRINT messages for now. ACPICA BZ  
3916 827.  
1056 repairs were converted to ACPI\_DEBUG\_PRINT messages for now. ACPICA BZ 827.

3918 Example Code and Data Size: These are the sizes for the OS-independent  
3919 apica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
3920 debug version of the code includes the debug output trace mechanism and  
3921 has a

1060 *debug version of the code includes the debug output trace mechanism and has a*  
 3922 *much larger code and data size.*

3924 Previous Release:  
 3925 Non-Debug Version: 86.6K Code, 18.2K Data, 104.8K Total  
 3926 Debug Version: 162.7K Code, 50.8K Data, 213.5K Total  
 3927 Current Release:  
 3928 Non-Debug Version: 87.0K Code, 18.0K Data, 105.0K Total  
 3929 Debug Version: 163.4K Code, 50.8K Data, 214.2K Total

3931 2) iASL Compiler/Disassembler and Tools:

3933 **iASL: Fixed a regression introduced in 20091112 where intermediate .SRC**  
 3934 **files**  
 1072 *iASL: Fixed a regression introduced in 20091112 where intermediate .SRC files*  
 3935 *were no longer automatically removed at the termination of the compile.*

3937 acpiexec: Implemented the -f option to specify default region fill value.  
 3938 This option specifies the value used to initialize buffers that simulate  
 3939 **operation regions. Default value is zero. Useful for debugging problems**  
 3940 **that**  
 1077 *operation regions. Default value is zero. Useful for debugging problems that*  
 3941 *depend on a specific initial value for a region or field.*

3943 -----  
 3944 12 November 2009. Summary of changes for version 20091112:

3946 1) ACPI CA Core Subsystem:

3948 Implemented a post-order callback to AcpiWalkNamespace. The existing  
 3949 interface only has a pre-order callback. This change adds an additional  
 3950 **parameter for a post-order callback which will be more useful for bus**  
 3951 **scans.**  
 1087 *parameter for a post-order callback which will be more useful for bus scans.*  
 3952 ACPICA BZ 779. Lin Ming. Updated the ACPICA Programmer Reference.

3954 Modified the behavior of the operation region memory mapping cache for  
 3955 **SystemMemory. Ensure that the memory mappings created for operation**  
 3956 **regions**  
 1091 *SystemMemory. Ensure that the memory mappings created for operation regions*  
 3957 *do not cross 4K page boundaries. Crossing a page boundary while mapping*  
 3958 **regions can cause kernel warnings on some hosts if the pages have**  
 3959 **different**  
 3960 **attributes. Such regions are probably BIOS bugs, and this is the**  
 3961 **workaround.**  
 1093 *regions can cause kernel warnings on some hosts if the pages have different*  
 1094 *attributes. Such regions are probably BIOS bugs, and this is the workaround.*  
 3962 Linux BZ 14445. Lin Ming.

3964 Implemented an automatic repair for predefined methods that must return  
 3965 **sorted lists. This change will repair (by sorting) packages returned by**  
 3966 **\_ALR,**  
 3967 **\_PSS, and \_TSS. Drivers can now assume that the packages are correctly**  
 3968 **sorted**  
 1098 *sorted lists. This change will repair (by sorting) packages returned by \_ALR,*  
 1099 *\_PSS, and \_TSS. Drivers can now assume that the packages are correctly sorted*  
 3969 *and do not contain NULL package elements. Adds one new file,*  
 3970 *namespace/nsrepair2.c. ACPICA BZ 784. Lin Ming, Bob Moore.*

3972 **Fixed a possible fault during predefined name validation if a return**  
 3973 **Package**  
 1103 *Fixed a possible fault during predefined name validation if a return Package*  
 3974 *object contains NULL elements. Also adds a warning if a NULL element is*  
 3975 **followed by any non-null elements. ACPICA BZ 813, 814. Future enhancement**  
 3976 **may**  
 1105 *followed by any non-null elements. ACPICA BZ 813, 814. Future enhancement may*

3977 include repair or removal of all such NULL elements where possible.

3979 **Implemented additional module-level executable AML code support. This**  
 3980 **change**  
 1108 *Implemented additional module-level executable AML code support. This change*  
 3981 *will execute module-level code that is not at the root of the namespace*  
 3982 **(under a Device object, etc.) at table load time. Module-level executable**  
 3983 **AML**  
 1110 *(under a Device object, etc.) at table load time. Module-level executable AML*  
 3984 *code has been illegal since ACPI 2.0. ACPICA BZ 762. Lin Ming.*

3986 **Implemented a new internal function to create Integer objects. This**  
 3987 **function**  
 1113 *Implemented a new internal function to create Integer objects. This function*  
 3988 *simplifies miscellaneous object creation code. ACPICA BZ 823.*

3990 **Reduced the severity of predefined repair messages, Warning to Info.**  
 3991 **Since**  
 3992 **the object was successfully repaired, a warning is too severe. Reduced to**  
 3993 **an**  
 3994 **info message for now. These messages may eventually be changed to debug-**  
 3995 **only.**  
 1116 *Reduced the severity of predefined repair messages, Warning to Info. Since*  
 1117 *the object was successfully repaired, a warning is too severe. Reduced to an*  
 1118 *info message for now. These messages may eventually be changed to debug-only.*  
 3996 ACPICA BZ 812.

3998 Example Code and Data Size: These are the sizes for the OS-independent  
 3999 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 4000 **debug version of the code includes the debug output trace mechanism and**  
 4001 **has a**  
 1123 *debug version of the code includes the debug output trace mechanism and has a*  
 4002 *much larger code and data size.*

4004 Previous Release:  
 4005 Non-Debug Version: 85.8K Code, 18.0K Data, 103.8K Total  
 4006 Debug Version: 161.8K Code, 50.6K Data, 212.4K Total  
 4007 Current Release:  
 4008 Non-Debug Version: 86.6K Code, 18.2K Data, 104.8K Total  
 4009 Debug Version: 162.7K Code, 50.8K Data, 213.5K Total

4011 2) iASL Compiler/Disassembler and Tools:

4013 **iASL: Implemented Switch() with While(1) so that Break works correctly.**  
 4014 **This**  
 4015 **change correctly implements the Switch operator with a surrounding**  
 4016 **While(1)**  
 1135 *iASL: Implemented Switch() with While(1) so that Break works correctly. This*  
 1136 *change correctly implements the Switch operator with a surrounding While(1)*  
 4017 *so that the Break operator works as expected. ACPICA BZ 461. Lin Ming.*

4019 **iASL: Added a message if a package initializer list is shorter than**  
 4020 **package**  
 4021 **length. Adds a new remark for a Package() declaration if an initializer**  
 4022 **list**  
 1139 *iASL: Added a message if a package initializer list is shorter than package*  
 1140 *length. Adds a new remark for a Package() declaration if an initializer list*  
 4023 *exists, but is shorter than the declared length of the package. Although*  
 4024 *technically legal, this is probably a coding error and it is seen in the*  
 4025 *field. ACPICA BZ 815. Lin Ming, Bob Moore.*

4027 **iASL: Fixed a problem where the compiler could fault after the maximum**  
 4028 **number**  
 1145 *iASL: Fixed a problem where the compiler could fault after the maximum number*  
 4029 *of errors was reached (200).*

4031 **acpixtract: Fixed a possible warning for pointer cast if the compiler**  
 4032 **warning**  
 1148 *acpixtract: Fixed a possible warning for pointer cast if the compiler warning*  
 4033 level set very high.

4035 -----  
 4036 13 October 2009. Summary of changes for version 20091013:

4038 1) ACPI CA Core Subsystem:

4040 **Fixed a problem where an Operation Region \_REG method could be executed**  
 4041 **more**  
 4042 **than once. If a custom address space handler is installed by the host**  
 4043 **before**  
 4044 **the "initialize operation regions" phase of the ACPICA initialization,**  
 4045 **any**  
 1156 *Fixed a problem where an Operation Region \_REG method could be executed more*  
 1157 *than once. If a custom address space handler is installed by the host before*  
 1158 *the "initialize operation regions" phase of the ACPICA initialization, any*  
 4046 *\_REG methods for that address space could be executed twice. This change*  
 4047 *fixes the problem. ACPICA BZ 427. Lin Ming.*

4049 Fixed a possible memory leak for the Scope() ASL operator. When the exact  
 4050 invocation of "Scope(\)" is executed (change scope to root), one internal  
 4051 operand object was leaked. Lin Ming.

4053 Implemented a run-time repair for the \_MAT predefined method. If the \_MAT  
 4054 return value is defined as a Field object in the AML, and the field  
 4055 **size is less than or equal to the default width of an integer (32 or**  
 4056 **64), \_MAT**  
 1168 *size is less than or equal to the default width of an integer (32 or 64), \_MAT*  
 4057 *can incorrectly return an Integer instead of a Buffer. ACPICA now*  
 4058 *automatically repairs this problem. ACPICA BZ 810.*

4060 **Implemented a run-time repair for the \_BIF and \_BIX predefined methods.**  
 4061 **The**  
 1172 *Implemented a run-time repair for the \_BIF and \_BIX predefined methods. The*  
 4062 *"OEM Information" field is often incorrectly returned as an Integer with*  
 4063 *value zero if the field is not supported by the platform. This is due to*  
 4064 *an*  
 1174 *value zero if the field is not supported by the platform. This is due to an*  
 4065 *ambiguity in the ACPI specification. The field should always be a string.*  
 4066 *ACPICA now automatically repairs this problem by returning a NULL string*  
 4067 *within the returned Package. ACPICA BZ 807.*

4069 Example Code and Data Size: These are the sizes for the OS-independent  
 4070 apica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 4071 **debug version of the code includes the debug output trace mechanism and**  
 4072 **has a**  
 1181 *debug version of the code includes the debug output trace mechanism and has a*  
 4073 *much larger code and data size.*

4075 Previous Release:  
 4076 Non-Debug Version: 85.6K Code, 18.0K Data, 103.6K Total  
 4077 Debug Version: 161.7K Code, 50.9K Data, 212.6K Total  
 4078 Current Release:  
 4079 Non-Debug Version: 85.8K Code, 18.0K Data, 103.8K Total  
 4080 Debug Version: 161.8K Code, 50.6K Data, 212.4K Total

4082 2) iASL Compiler/Disassembler and Tools:

4084 Disassembler: Fixed a problem where references to external symbols that  
 4085 **contained one or more parent-prefixes (carats) were not handled**  
 4086 **correctly,**  
 1194 *contained one or more parent-prefixes (carats) were not handled correctly,*  
 4087 *possibly causing a fault. ACPICA BZ 806. Lin Ming.*

4089 Disassembler: Restructured the code so that all functions that handle  
 4090 external symbols are in a single module. One new file is added,  
 4091 common/dmextern.c.

4093 AML Debugger: Added a max count argument for the Batch command (which  
 4094 executes multiple predefined methods within the namespace.)

4096 iASL: Updated the compiler documentation (User Reference.) Available at  
 4097 <http://www.acpica.org/documentation/>. ACPICA BZ 750.

4099 AcpiXtract: Updated for Lint and other formatting changes. Close all open  
 4100 files.

4102 -----  
 4103 03 September 2009. Summary of changes for version 20090903:

4105 1) ACPI CA Core Subsystem:

4107 For Windows Vista compatibility, added the automatic execution of an \_INI  
 4108 method located at the namespace root (\\_INI). This method is executed at  
 4109 **table load time. This support is in addition to the automatic execution**  
 4110 **of**  
 1217 *table load time. This support is in addition to the automatic execution of*  
 4111 *\\_SB.\_INI. Lin Ming.*

4113 **Fixed a possible memory leak in the interpreter for AML package objects**  
 4114 **if**  
 4115 **the package initializer list is longer than the defined size of the**  
 4116 **package.**  
 4117 **This apparently can only happen if the BIOS changes the package size on**  
 4118 **the**  
 1220 *Fixed a possible memory leak in the interpreter for AML package objects if*  
 1221 *the package initializer list is longer than the defined size of the package.*  
 1222 *This apparently can only happen if the BIOS changes the package size on the*  
 4119 *fly (seen in a \_PSS object), as ASL compilers do not allow this. The*  
 4120 *interpreter will truncate the package to the defined size (and issue an*  
 4121 *error*  
 4122 *message), but previously could leave the extra objects undeleted if they*  
 4123 *were*  
 4124 *pre-created during the argument processing (such is the case if the*  
 4125 *package*  
 1224 *interpreter will truncate the package to the defined size (and issue an error*  
 1225 *message), but previously could leave the extra objects undeleted if they were*  
 1226 *pre-created during the argument processing (such is the case if the package*  
 4126 *consists of a number of sub-packages as in the \_PSS.) ACPICA BZ 805.*

4128 Fixed a problem seen when a Buffer or String is stored to itself via ASL.  
 4129 **This has been reported in the field. Previously, ACPICA would zero out**  
 4130 **the**  
 1230 *This has been reported in the field. Previously, ACPICA would zero out the*  
 4131 *buffer/string. Now, the operation is treated as a noop. Provides Windows*  
 4132 *compatibility. ACPICA BZ 803. Lin Ming.*

4134 Removed an extraneous error message for ASL constructs of the form  
 4135 **Store(LocalX,LocalX) when LocalX is uninitialized. These curious**  
 4136 **statements**  
 4137 **are seen in many BIOSs and are once again treated as NOOPs and no error**  
 4138 **is**  
 1235 *Store(LocalX,LocalX) when LocalX is uninitialized. These curious statements*  
 1236 *are seen in many BIOSs and are once again treated as NOOPs and no error is*  
 4139 *emitted when they are encountered. ACPICA BZ 785.*

4141 Fixed an extraneous warning message if a \_DSM reserved method returns a  
 4142 Package object. \_DSM can return any type of object, so validation on the  
 4143 return type cannot be performed. ACPICA BZ 802.

4145 Example Code and Data Size: These are the sizes for the OS-independent  
 4146 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 4147 **debug version of the code includes the debug output trace mechanism and**  
 4148 **has a**  
 1245 *debug version of the code includes the debug output trace mechanism and has a*  
 4149 much larger code and data size.

4151 Previous Release:  
 4152 Non-Debug Version: 85.5K Code, 18.0K Data, 103.5K Total  
 4153 Debug Version: 161.6K Code, 50.9K Data, 212.5K Total  
 4154 Current Release:  
 4155 Non-Debug Version: 85.6K Code, 18.0K Data, 103.6K Total  
 4156 Debug Version: 161.7K Code, 50.9K Data, 212.6K Total

4158 2) iASL Compiler/Disassembler and Tools:

4160 iASL: Fixed a problem with the use of the Alias operator and Resource  
 4161 Templates. The correct alias is now constructed and no error is emitted.  
 4162 ACPICA BZ 738.

4164 **iASL: Implemented the -I option to specify additional search directories**  
 4165 **for**  
 1261 *iASL: Implemented the -I option to specify additional search directories for*  
 4166 include files. Allows multiple additional search paths for include files.  
 4167 **Directories are searched in the order specified on the command line**  
 4168 **(after**  
 1263 *Directories are searched in the order specified on the command line (after*  
 4169 the local directory is searched.) ACPICA BZ 800.

4171 iASL: Fixed a problem where the full pathname for include files was not  
 4172 emitted for warnings/errors. This caused the IDE support to not work  
 4173 properly. ACPICA BZ 765.

4175 iASL: Implemented the -@ option to specify a Windows-style response file  
 4176 containing additional command line options. ACPICA BZ 801.

4178 **AcpiExec: Added support to load multiple AML files simultaneously (such**  
 4179 **as**  
 4180 **a**  
 1273 *AcpiExec: Added support to load multiple AML files simultaneously (such as a*  
 4181 DSDT and multiple SSDTs). Also added support for wildcards within the AML  
 4182 pathname. These features allow all machine tables to be easily loaded and  
 4183 debugged together. ACPICA BZ 804.

4185 **Disassembler: Added missing support for disassembly of HEST table Error**  
 4186 **Bank**  
 1278 *Disassembler: Added missing support for disassembly of HEST table Error Bank*  
 4187 subtables.

4189 -----  
 4190 30 July 2009. Summary of changes for version 20090730:

4192 The ACPI 4.0 implementation for ACPICA is complete with this release.

4194 1) ACPI CA Core Subsystem:

4196 ACPI 4.0: Added header file support for all new and changed ACPI tables.  
 4197 **Completely new tables are: IBFT, IVRS, MSCT, and WAET. Tables that are**  
 4198 **new**  
 4199 **for ACPI 4.0, but have previously been supported in ACPICA are: CPEP,**  
 4200 **BERT,**  
 4201 **EINJ, ERST, and HEST. Other newly supported tables are: UEFI and WDAT.**  
 4202 **There**  
 1289 *Completely new tables are: IBFT, IVRS, MSCT, and WAET. Tables that are new*  
 1290 *for ACPI 4.0, but have previously been supported in ACPICA are: CPEP, BERT,*

1291 *EINJ, ERST, and HEST. Other newly supported tables are: UEFI and WDAT. There*  
 4203 *have been some ACPI 4.0 changes to other existing tables. Split the large*  
 4204 *actbl1.h header into the existing actbl2.h header. ACPICA BZ 774.*

4206 **ACPI 4.0: Implemented predefined name validation for all new names. There**  
 4207 **are**  
 4208 **31 new names in ACPI 4.0. The predefined validation module was split into**  
 4209 **two**  
 1295 *ACPI 4.0: Implemented predefined name validation for all new names. There are*  
 1296 *31 new names in ACPI 4.0. The predefined validation module was split into two*  
 4210 *files. The new file is namespace/nsrepair.c. ACPICA BZ 770.*

4212 Implemented support for so-called "module-level executable code". This is  
 4213 **executable AML code that exists outside of any control method and is**  
 4214 **intended**  
 4215 **to be executed at table load time. Although illegal since ACPI 2.0, this**  
 4216 **type**  
 4217 **of code still exists and is apparently still being created. Blocks of**  
 4218 **this**  
 4219 **code are now detected and executed as intended. Currently, the code**  
 4220 **blocks**  
 1300 *executable AML code that exists outside of any control method and is intended*  
 1301 *to be executed at table load time. Although illegal since ACPI 2.0, this type*  
 1302 *of code still exists and is apparently still being created. Blocks of this*  
 1303 *code are now detected and executed as intended. Currently, the code blocks*  
 4221 *must exist under either an If, Else, or While construct; these are the*  
 4222 *typical cases seen in the field. ACPICA BZ 762. Lin Ming.*

4224 Implemented an automatic dynamic repair for predefined names that return  
 4225 **nested Package objects. This applies to predefined names that are defined**  
 4226 **to**  
 1308 *nested Package objects. This applies to predefined names that are defined to*  
 4227 *return a variable-length Package of sub-packages. If the number of sub-*  
 4228 **packages is one, BIOS code is occasionally seen that creates a simple**  
 4229 **single**  
 1310 *packages is one, BIOS code is occasionally seen that creates a simple single*  
 4230 *package with no sub-packages. This code attempts to fix the problem by*  
 4231 **wrapping a new package object around the existing package. These methods**  
 4232 **can**  
 4233 **be repaired: \_ALR, \_CSD, \_HPX, \_MLS, \_PRT, \_PSS, \_TRT, and \_TSS. ACPICA**  
 4234 **BZ**  
 1312 *wrapping a new package object around the existing package. These methods can*  
 1313 *be repaired: \_ALR, \_CSD, \_HPX, \_MLS, \_PRT, \_PSS, \_TRT, and \_TSS. ACPICA BZ*  
 4235 *790.*

4237 **Fixed a regression introduced in 20090625 for the AcpiGetDevices**  
 4238 **interface.**  
 4239 **The \_HID/\_CID matching was broken and no longer matched IDs correctly.**  
 4240 **ACPICA**  
 1316 *Fixed a regression introduced in 20090625 for the AcpiGetDevices interface.*  
 1317 *The \_HID/\_CID matching was broken and no longer matched IDs correctly. ACPICA*  
 4241 *BZ 793.*

4243 Fixed a problem with AcpiReset where the reset would silently fail if the  
 4244 **register was one of the protected I/O ports. AcpiReset now bypasses the**  
 4245 **port**  
 4246 **validation mechanism. This may eventually be driven into the**  
 4247 **AcpiRead/Write**  
 1321 *register was one of the protected I/O ports. AcpiReset now bypasses the port*  
 1322 *validation mechanism. This may eventually be driven into the AcpiRead/Write*  
 4248 *interfaces.*

4250 Fixed a regression related to the recent update of the AcpiRead/Write  
 4251 **interfaces. A sleep/suspend could fail if the optional PM2 Control**  
 4252 **register**  
 1326 *interfaces. A sleep/suspend could fail if the optional PM2 Control register*

4253 does not exist during an attempt to write the Bus Master Arbitration bit.  
 4254 **(However, some hosts already delete the code that writes this bit, and**  
 4255 **the**  
 1328 *(However, some hosts already delete the code that writes this bit, and the*  
 4256 *code may in fact be obsolete at this date.)* ACPICA BZ 799.

4258 **Fixed a problem where AcpiTerminate could fault if inadvertently called**  
 4259 **twice**  
 1331 *Fixed a problem where AcpiTerminate could fault if inadvertently called twice*  
 4260 *in succession.* ACPICA BZ 795.

4262 Example Code and Data Size: These are the sizes for the OS-independent  
 4263 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 4264 **debug version of the code includes the debug output trace mechanism and**  
 4265 **has a**  
 1336 *debug version of the code includes the debug output trace mechanism and has a*  
 4266 *much larger code and data size.*

4268 Previous Release:  
 4269 Non-Debug Version: 84.7K Code, 17.8K Data, 102.5K Total  
 4270 Debug Version: 160.5K Code, 50.6K Data, 211.1K Total  
 4271 Current Release:  
 4272 Non-Debug Version: 85.5K Code, 18.0K Data, 103.5K Total  
 4273 Debug Version: 161.6K Code, 50.9K Data, 212.5K Total

4275 2) iASL Compiler/Disassembler and Tools:

4277 ACPI 4.0: Implemented disassembler support for all new ACPI tables and  
 4278 changes to existing tables. ACPICA BZ 775.

4280 -----  
 4281 25 June 2009. Summary of changes for version 20090625:

4283 The ACPI 4.0 Specification was released on June 16 and is available at  
 4284 www.acpi.info. ACPICA implementation of ACPI 4.0 is underway and will  
 4285 continue for the next few releases.

4287 1) ACPI CA Core Subsystem:

4289 ACPI 4.0: Implemented interpreter support for the IPMI operation region  
 4290 **address space. Includes support for bi-directional data buffers and an**  
 4291 **IPMI**  
 4292 **address space handler (to be installed by an IPMI device driver.)** ACPICA  
 4293 **BZ**  
 1361 *address space. Includes support for bi-directional data buffers and an IPMI*  
 1362 *address space handler (to be installed by an IPMI device driver.)* ACPICA BZ  
 4294 773. Lin Ming.

4296 **ACPI 4.0: Added changes for existing ACPI tables - FACS and SRAT.**  
 4297 **Includes**  
 1365 *ACPI 4.0: Added changes for existing ACPI tables - FACS and SRAT. Includes*  
 4298 *support in both the header files and the disassembler.*

4300 Completed a major update for the AcpiGetObjectInfo external interface.  
 4301 Changes include:  
 4302 - Support for variable, unlimited length HID, UID, and CID strings.  
 4303 - **Support Processor objects the same as Devices (HID,UID,CID,ADR,STA,**  
 4304 **etc.)**  
 1371 *- Support Processor objects the same as Devices (HID,UID,CID,ADR,STA, etc.)*  
 4305 - Call the \_SxW power methods on behalf of a device object.  
 4306 - Determine if a device is a PCI root bridge.  
 4307 - Change the ACPI\_BUFFER parameter to ACPI\_DEVICE\_INFO.  
 4308 **These changes will require an update to all callers of this interface.**  
 4309 **See**  
 4310 **the updated ACPICA Programmer Reference for details. One new source file**  
 4311 **has**

1375 *These changes will require an update to all callers of this interface. See*  
 1376 *the updated ACPICA Programmer Reference for details. One new source file has*  
 4312 *been added - utilities/utids.c. ACPICA BZ 368, 780.*

4314 Updated the AcpiRead and AcpiWrite external interfaces to support 64-bit  
 4315 **transfers. The Value parameter has been extended from 32 bits to 64 bits**  
 4316 **in**  
 4317 **order to support new ACPI 4.0 tables. These changes will require an**  
 4318 **update**  
 4319 **to**  
 1380 *transfers. The Value parameter has been extended from 32 bits to 64 bits in*  
 1381 *order to support new ACPI 4.0 tables. These changes will require an update to*  
 4320 *all callers of these interfaces. See the ACPICA Programmer Reference for*  
 4321 *details.* ACPICA BZ 768.

4323 **Fixed several problems with AcpiAttachData. The handler was not invoked**  
 4324 **when**  
 4325 **the host node was deleted. The data sub-object was not automatically**  
 4326 **deleted**  
 4327 **when the host node was deleted. The interface to the handler had an**  
 4328 **unused**  
 1385 *Fixed several problems with AcpiAttachData. The handler was not invoked when*  
 1386 *the host node was deleted. The data sub-object was not automatically deleted*  
 1387 *when the host node was deleted. The interface to the handler had an unused*  
 4329 *parameter, this was removed.* ACPICA BZ 778.

4331 Enhanced the function that dumps ACPI table headers. All non-printable  
 4332 **characters in the string fields are now replaced with '?' (Signature,**  
 4333 **OemId,**  
 1391 *characters in the string fields are now replaced with '?' (Signature, OemId,*  
 4334 *OemTableId, and CompilerId.)* ACPI tables with non-printable characters in  
 4335 these fields are occasionally seen in the field. ACPICA BZ 788.

4337 Fixed a problem with predefined method repair code where the code that  
 4338 **attempts to repair/convert an object of incorrect type is only executed**  
 4339 **on**  
 4340 **the first time the predefined method is called. The mechanism that**  
 4341 **disables**  
 1396 *attempts to repair/convert an object of incorrect type is only executed on*  
 1397 *the first time the predefined method is called. The mechanism that disables*  
 4342 *warnings on subsequent calls was interfering with the repair mechanism.*  
 4343 *ACPICA BZ 781.*

4345 **Fixed a possible memory leak in the predefined validation/repair code**  
 4346 **when**  
 4347 **a**  
 1401 *Fixed a possible memory leak in the predefined validation/repair code when a*  
 4348 *buffer is automatically converted to an expected string object.*

4350 **Removed obsolete 16-bit files from the distribution and from the current**  
 4351 **git**  
 1404 *Removed obsolete 16-bit files from the distribution and from the current git*  
 4352 *tree head.* ACPICA BZ 776.

4354 Example Code and Data Size: These are the sizes for the OS-independent  
 4355 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 4356 **debug version of the code includes the debug output trace mechanism and**  
 4357 **has a**  
 1409 *debug version of the code includes the debug output trace mechanism and has a*  
 4358 *much larger code and data size.*

4360 Previous Release:  
 4361 Non-Debug Version: 83.4K Code, 17.5K Data, 100.9K Total  
 4362 Debug Version: 158.9K Code, 50.0K Data, 208.9K Total  
 4363 Current Release:  
 4364 Non-Debug Version: 84.7K Code, 17.8K Data, 102.5K Total



4365        Debug Version:        160.5K Code, 50.6K Data, 211.1K Total

4367 2) iASL Compiler/Disassembler and Tools:

4369 ACPI 4.0: iASL and Disassembler - implemented support for the new IPMI  
4370 operation region keyword. ACPICA BZ 771, 772. Lin Ming.

4372 ACPI 4.0: iASL - implemented compile-time validation support for all new  
4373 predefined names and control methods (31 total). ACPICA BZ 769.

4375 -----

4376 21 May 2009. Summary of changes for version 20090521:

4378 1) ACPI CA Core Subsystem:

4380 **Disabled the preservation of the SCI enable bit in the PM1 control**  
4381 **register.**  
4382 **The SCI enable bit (bit 0, SCI\_EN) is defined by the ACPI specification**  
4383 **to**  
4384 **be**  
4385 *Disabled the preservation of the SCI enable bit in the PM1 control register.*  
4386 *The SCI enable bit (bit 0, SCI\_EN) is defined by the ACPI specification to be*  
4387 *a "preserved" bit - "OSPM always preserves this bit position", section*  
4388 *4.7.3.2.1. However, some machines fail if this bit is in fact preserved*  
4389 *because the bit needs to be explicitly set by the OS as a workaround. No*  
4390 *machines fail if the bit is not preserved. Therefore, ACPICA no longer*  
4391 *attempts to preserve this bit.*

4391 Fixed a problem in AcpiRsGetPciRoutingTableLength where an invalid or  
4392 incorrectly formed \_PRT package could cause a fault. Added validation to  
4393 ensure that each package element is actually a sub-package.

4395 **Implemented a new interface to install or override a single control**  
4396 **method,**  
4397 **AcpiInstallMethod. This interface is useful when debugging in order to**  
4398 **repair**  
4399 **an existing method or to install a missing method without having to**  
4400 **override**  
4401 *Implemented a new interface to install or override a single control method,*  
4402 *AcpiInstallMethod. This interface is useful when debugging in order to repair*  
4403 *an existing method or to install a missing method without having to override*  
4404 *the entire ACPI table. See the ACPICA Programmer Reference for use and*  
4405 *examples. Lin Ming, Bob Moore.*

4404 Fixed several reference count issues with the DdbHandle object that is  
4405 **created from a Load or LoadTable operator. Prevent premature deletion of**  
4406 **the**  
4407 **object. Also, mark the object as invalid once the table has been**  
4408 **unloaded.**  
4409 **This is needed because the handle itself may not be deleted after the**  
4410 **table**  
4411 *created from a Load or LoadTable operator. Prevent premature deletion of the*  
4412 *object. Also, mark the object as invalid once the table has been unloaded.*  
4413 *This is needed because the handle itself may not be deleted after the table*  
4414 *unload, depending on whether it has been stored in a named object by the*  
4415 *caller. Lin Ming.*

4414 Fixed a problem with Mutex Sync Levels. Fixed a problem where if multiple  
4415 **mutexes of the same sync level are acquired but then not released in**  
4416 **strict**  
4417 **opposite order, the internally maintained Current Sync Level becomes**  
4418 **confused**  
4419 *mutexes of the same sync level are acquired but then not released in strict*  
4420 *opposite order, the internally maintained Current Sync Level becomes confused*  
4421 *and can cause subsequent execution errors. ACPICA BZ 471.*

4421 Changed the allowable release order for ASL mutex objects. The ACPI 4.0  
4422 **specification has been changed to make the SyncLevel for mutex objects**  
4423 **more**  
4424 **useful. When releasing a mutex, the SyncLevel of the mutex must now be**  
4425 **the**  
4426 **same as the current sync level. This makes more sense than the previous**  
4427 **rule**  
4428 *specification has been changed to make the SyncLevel for mutex objects more*  
4429 *useful. When releasing a mutex, the SyncLevel of the mutex must now be the*  
4430 *same as the current sync level. This makes more sense than the previous rule*  
4431 *(SyncLevel less than or equal). This change updates the code to match the*  
4432 *specification.*

4431 **Fixed a problem with the local version of the AcpiOsPurgeCache function.**  
4432 **The**  
4433 *Fixed a problem with the local version of the AcpiOsPurgeCache function. The*  
4434 *(local) cache must be locked during all cache object deletions. Andrew*  
4435 *Baumann.*

4436 **Updated the Load operator to use operation region interfaces. This**  
4437 **replaces**  
4438 **direct memory mapping with region access calls. Now, all region accesses**  
4439 **go**  
4440 *Updated the Load operator to use operation region interfaces. This replaces*  
4441 *direct memory mapping with region access calls. Now, all region accesses go*  
4442 *through the installed region handler as they should.*

4442 **Simplified and optimized the NsGetNextNode function. Reduced parameter**  
4443 **count**  
4444 *Simplified and optimized the NsGetNextNode function. Reduced parameter count*  
4445 *and reduced code for this frequently used function.*

4446 Example Code and Data Size: These are the sizes for the OS-independent  
4447 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
4448 **debug version of the code includes the debug output trace mechanism and**  
4449 **has a**  
4450 *debug version of the code includes the debug output trace mechanism and has a*  
4451 *much larger code and data size.*

4452 Previous Release:  
4453 Non-Debug Version: 82.8K Code, 17.5K Data, 100.3K Total  
4454 Debug Version: 158.0K Code, 49.9K Data, 207.9K Total  
4455 Current Release:  
4456 Non-Debug Version: 83.4K Code, 17.5K Data, 100.9K Total  
4457 Debug Version: 158.9K Code, 50.0K Data, 208.9K Total

4459 2) iASL Compiler/Disassembler and Tools:

4461 **Disassembler: Fixed some issues with DMAR, HEST, MADT tables. Some**  
4462 **problems**  
4463 **with sub-table disassembly and handling invalid sub-tables. Attempt**  
4464 **recovery**  
4465 *Disassembler: Fixed some issues with DMAR, HEST, MADT tables. Some problems*  
4466 *with sub-table disassembly and handling invalid sub-tables. Attempt recovery*  
4467 *after an invalid sub-table ID.*

4467 -----

4468 22 April 2009. Summary of changes for version 20090422:

4470 1) ACPI CA Core Subsystem:

4472 **Fixed a compatibility issue with the recently released I/O port**  
4473 **protection**  
4474 *Fixed a compatibility issue with the recently released I/O port protection*  
4475 *mechanism. For windows compatibility, 1) On a port protection violation,*  
4476 *simply ignore the request and do not return an exception (allow the*

4476 **control**  
 1505 *simply ignore the request and do not return an exception (allow the control*  
 4477 *method to continue execution.)* 2) If only part of the request overlaps a  
 4478 **protected port, read/write the individual ports that are not protected.**  
 4479 **Linux**  
 1507 *protected port, read/write the individual ports that are not protected. Linux*  
 4480 BZ 13036. Lin Ming

4482 **Enhanced the execution of the ASL/AML BreakPoint operator so that it**  
 4483 **actually**  
 1510 *Enhanced the execution of the ASL/AML BreakPoint operator so that it actually*  
 4484 *breaks into the AML debugger if the debugger is present. This matches the*  
 4485 *ACPI-defined behavior.*

4487 Fixed several possible warnings related to the use of the configurable  
 4488 ACPI\_THREAD\_ID. This type can now be configured as either an integer or a  
 4489 pointer with no warnings. Also fixes several warnings in printf-like  
 4490 statements for the 64-bit build when the type is configured as a pointer.  
 4491 ACPICA BZ 766, 767.

4493 **Fixed a number of possible warnings when compiling with gcc 4+ (depending**  
 4494 **on**  
 4495 **warning options.) Examples include printf formats, aliasing, unused**  
 4496 **globals,**  
 1520 *Fixed a number of possible warnings when compiling with gcc 4+ (depending on*  
 1521 *warning options.) Examples include printf formats, aliasing, unused globals,*  
 4497 *missing prototypes, missing switch default statements, use of non-ANSI*  
 4498 **library functions, use of non-ANSI constructs. See generate/unix/Makefile**  
 4499 **for**  
 1523 *library functions, use of non-ANSI constructs. See generate/unix/Makefile for*  
 4500 *a list of warning options used with gcc 3 and 4. ACPICA BZ 735.*

4502 Example Code and Data Size: These are the sizes for the OS-independent  
 4503 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 4504 **debug version of the code includes the debug output trace mechanism and**  
 4505 **has a**  
 1528 *debug version of the code includes the debug output trace mechanism and has a*  
 4506 *much larger code and data size.*

4508 Previous Release:  
 4509 Non-Debug Version: 82.6K Code, 17.6K Data, 100.2K Total  
 4510 Debug Version: 157.7K Code, 49.9K Data, 207.6K Total  
 4511 Current Release:  
 4512 Non-Debug Version: 82.8K Code, 17.5K Data, 100.3K Total  
 4513 Debug Version: 158.0K Code, 49.9K Data, 207.9K Total

4515 2) iASL Compiler/Disassembler and Tools:

4517 **iASL: Fixed a generation warning from Bison 2.3 and fixed several**  
 4518 **warnings**  
 4519 **on**  
 1540 *iASL: Fixed a generation warning from Bison 2.3 and fixed several warnings on*  
 4520 *the 64-bit build.*

4522 **iASL: Fixed a problem where the Unix/Linux versions of the compiler could**  
 4523 **not**  
 1543 *iASL: Fixed a problem where the Unix/Linux versions of the compiler could not*  
 4524 *correctly digest Windows/DOS formatted files (with CR/LF).*

4526 iASL: Added a new option for "quiet mode" (-va) that produces only the  
 4527 compilation summary, not individual errors and warnings. Useful for large  
 4528 batch compilations.

4530 **AcpiExec: Implemented a new option (-z) to enable a forced**  
 4531 **semaphore/mutex**  
 4532 **timeout that can be used to detect hang conditions during execution of**

4533 **AML**  
 4534 **code (includes both internal semaphores and AML-defined mutexes and**  
 4535 **events.)**  
 1550 *AcpiExec: Implemented a new option (-z) to enable a forced semaphore/mutex*  
 1551 *timeout that can be used to detect hang conditions during execution of AML*  
 1552 *code (includes both internal semaphores and AML-defined mutexes and events.)*

4537 Added new makefiles for the generation of acpica in a generic unix-like  
 4538 **environment. These makefiles are intended to generate the acpica tools**  
 4539 **and**  
 1555 *environment. These makefiles are intended to generate the acpica tools and*  
 4540 *utilities from the original acpica git source tree structure.*

4542 Test Suites: Updated and cleaned up the documentation files. Updated the  
 4543 **copyrights to 2009, affecting all source files. Use the new version of**  
 4544 **iASL**  
 4545 **with quiet mode. Increased the number of available semaphores in the**  
 4546 **Windows**  
 4547 **OSL, allowing the aslts to execute fully on Windows. For the Unix OSL,**  
 4548 **added**  
 1559 *copyrights to 2009, affecting all source files. Use the new version of iASL*  
 1560 *with quiet mode. Increased the number of available semaphores in the Windows*  
 1561 *OSL, allowing the aslts to execute fully on Windows. For the Unix OSL, added*  
 4549 *an alternate implementation of the semaphore timeout to allow aslts to*  
 4550 *execute fully on Cygwin.*

4552 -----  
 4553 20 March 2009. Summary of changes for version 20090320:

4555 1) ACPI CA Core Subsystem:

4557 **Fixed a possible race condition between AcpiWalkNamespace and dynamic**  
 4558 **table**  
 4559 **unloads. Added a reader/writer locking mechanism to allow multiple**  
 4560 **concurrent**  
 4561 **namespace walks (readers), but block a dynamic table unload until it can**  
 4562 **gain**  
 4563 **exclusive write access to the namespace. This fixes a problem where a**  
 4564 **table**  
 4565 **unload could (possibly catastrophically) delete the portion of the**  
 4566 **namespace**  
 4567 **that is currently being examined by a walk. Adds a new file, utlock.c,**  
 4568 **that**  
 1570 *Fixed a possible race condition between AcpiWalkNamespace and dynamic table*  
 1571 *unloads. Added a reader/writer locking mechanism to allow multiple concurrent*  
 1572 *namespace walks (readers), but block a dynamic table unload until it can gain*  
 1573 *exclusive write access to the namespace. This fixes a problem where a table*  
 1574 *unload could (possibly catastrophically) delete the portion of the namespace*  
 1575 *that is currently being examined by a walk. Adds a new file, utlock.c, that*  
 4569 *implements the reader/writer lock mechanism. ACPICA BZ 749.*

4571 **Fixed a regression introduced in version 20090220 where a change to the**  
 4572 **FADT**  
 4573 **handling could cause the ACPICA subsystem to access non-existent I/O**  
 4574 **ports.**  
 1578 *Fixed a regression introduced in version 20090220 where a change to the FADT*  
 1579 *handling could cause the ACPICA subsystem to access non-existent I/O ports.*

4576 **Modified the handling of FADT register and table (FACS/DSDT) addresses.**  
 4577 **The**  
 1581 *Modified the handling of FADT register and table (FACS/DSDT) addresses. The*  
 4578 *FADT can contain both 32-bit and 64-bit versions of these addresses.*  
 4579 *Previously, the 64-bit versions were favored, meaning that if both 32 and*  
 4580 *64*  
 1583 *Previously, the 64-bit versions were favored, meaning that if both 32 and 64*  
 4581 *versions were valid, but not equal, the 64-bit version was used. This was*

4582 found to cause some machines to fail. Now, in this case, the 32-bit  
4583 version  
1585 found to cause some machines to fail. Now, in this case, the 32-bit version  
4584 is used instead. This now matches the Windows behavior.

4586 Implemented a new mechanism to protect certain I/O ports. Provides  
4587 Microsoft  
1588 Implemented a new mechanism to protect certain I/O ports. Provides Microsoft  
4588 compatibility and protects the standard PC I/O ports from access via AML  
4589 code. Adds a new file, hwvalid.c

4591 Fixed a possible extraneous warning message from the FADT support. The  
4592 message warns of a 32/64 length mismatch between the legacy and GAS  
4593 definitions for a register.

4595 Removed the obsolete AcpiOsValidateAddress OSL interface. This interface  
4596 is  
4597 made obsolete by the port protection mechanism above. It was previously  
4598 used  
4599 to validate the entire address range of an operation region, which could  
4600 be  
1596 Removed the obsolete AcpiOsValidateAddress OSL interface. This interface is  
1597 made obsolete by the port protection mechanism above. It was previously used  
1598 to validate the entire address range of an operation region, which could be  
4601 incorrect if the range included illegal ports, but fields within the  
4602 operation region did not actually access those ports. Validation is now  
4603 performed on a per-field basis instead of the entire region.

4605 Modified the handling of the PM1 Status Register ignored bit (bit 11.)  
4606 Ignored bits must be "preserved" according to the ACPI spec. Usually,  
4607 this  
4608 means a read/modify/write when writing to the register. However, for  
4609 status  
4610 registers, writing a one means clear the event. Writing a zero means  
4611 preserve  
4612 the event (do not clear.) This behavior is clarified in the ACPI 4.0  
4613 spec,  
1604 Ignored bits must be "preserved" according to the ACPI spec. Usually, this  
1605 means a read/modify/write when writing to the register. However, for status  
1606 registers, writing a one means clear the event. Writing a zero means preserve  
1607 the event (do not clear.) This behavior is clarified in the ACPI 4.0 spec,  
4614 and the ACPICA code now simply always writes a zero to the ignored bit.

4616 Modified the handling of ignored bits for the PM1 A/B Control Registers.  
4617 As  
1610 Modified the handling of ignored bits for the PM1 A/B Control Registers. As  
4618 per the ACPI specification, for the control registers, preserve  
4619 (read/modify/write) all bits that are defined as either reserved or  
4620 ignored.  
1612 (read/modify/write) all bits that are defined as either reserved or ignored.

4622 Updated the handling of write-only bits in the PM1 A/B Control Registers.  
4623 When reading the register, zero the write-only bits as per the ACPI spec.  
4624 ACPICA BZ 443. Lin Ming.

4626 Removed "Linux" from the list of supported \_OSI strings. Linux no longer  
4627 wants to reply true to this request. The Windows strings are the only  
4628 paths  
1619 wants to reply true to this request. The Windows strings are the only paths  
4629 through the AML that are tested and known to work properly.

4631 Previous Release:  
4632 Non-Debug Version: 82.0K Code, 17.5K Data, 99.5K Total  
4633 Debug Version: 156.9K Code, 49.8K Data, 206.7K Total  
4634 Current Release:  
4635 Non-Debug Version: 82.6K Code, 17.6K Data, 100.2K Total

4636 Debug Version: 157.7K Code, 49.9K Data, 207.6K Total

4638 2) iASL Compiler/Disassembler and Tools:

4640 Acpiexec: Split the large aeexec.c file into two new files, aehandlers.c  
4641 and  
1631 Acpiexec: Split the large aeexec.c file into two new files, aehandlers.c and  
4642 aetables.c

4644 -----  
4645 20 February 2009. Summary of changes for version 20090220:

4647 1) ACPI CA Core Subsystem:

4649 Optimized the ACPI register locking. Removed locking for reads from the  
4650 ACPI  
4651 bit registers in PM1 Status, Enable, Control, and PM2 Control. The lock  
4652 is  
1639 Optimized the ACPI register locking. Removed locking for reads from the ACPI  
1640 bit registers in PM1 Status, Enable, Control, and PM2 Control. The lock is  
4653 not required when reading the single-bit registers. The  
4654 AcpiGetRegisterUnlocked function is no longer needed and has been  
4655 removed.  
4656 This will improve performance for reads on these registers. ACPICA BZ  
4657 760.  
1642 AcpiGetRegisterUnlocked function is no longer needed and has been removed.  
1643 This will improve performance for reads on these registers. ACPICA BZ 760.

4659 Fixed the parameter validation for AcpiRead/Write. Now return  
4660 AE\_BAD\_PARAMETER if the input register pointer is null, and  
4661 AE\_BAD\_ADDRESS  
4662 if  
4663 the register has an address of zero. Previously, these cases simply  
4664 returned  
4665 AE\_OK. For optional registers such as PM1B status/enable/control, the  
4666 caller  
1646 AE\_BAD\_PARAMETER if the input register pointer is null, and AE\_BAD\_ADDRESS if  
1647 the register has an address of zero. Previously, these cases simply returned  
1648 AE\_OK. For optional registers such as PM1B status/enable/control, the caller  
4667 should check for a valid register address before calling. ACPICA BZ 748.

4669 Renamed the external ACPI bit register access functions. Renamed  
4670 AcpiGetRegister and AcpiSetRegister to clarify the purpose of these  
4671 functions. The new names are AcpiReadBitRegister and  
4672 AcpiWriteBitRegister.  
4673 Also, restructured the code for these functions by simplifying the code  
4674 path  
1653 functions. The new names are AcpiReadBitRegister and AcpiWriteBitRegister.  
1654 Also, restructured the code for these functions by simplifying the code path  
4675 and condensing duplicate code to reduce code size.

4677 Added new functions to transparently handle the possibly split PM1 A/B  
4678 registers. AcpiHwReadMultiple and AcpiHwWriteMultiple. These two  
4679 functions  
4680 now handle the split registers for PM1 Status, Enable, and Control.  
4681 ACPICA  
4682 BZ  
1658 registers. AcpiHwReadMultiple and AcpiHwWriteMultiple. These two functions  
1659 now handle the split registers for PM1 Status, Enable, and Control. ACPICA BZ  
4683 746.

4685 Added a function to handle the PM1 control registers,  
4686 AcpiHwWritePm1Control.  
4687 This function writes both of the PM1 control registers (A/B). These  
4688 registers  
4689 are different than the PM1 A/B status and enable registers in that

4690 different  
 4691 values can be written to the A/B registers. Most notably, the SLP\_TYP  
 4692 bits  
 4693 can be different, as per the values returned from the \_Sx predefined  
 4694 methods.  
 1662 Added a function to handle the PM1 control registers, AcpiHwWritePmlControl.  
 1663 This function writes both of the PM1 control registers (A/B). These registers  
 1664 are different than the PM1 A/B status and enable registers in that different  
 1665 values can be written to the A/B registers. Most notably, the SLP\_TYP bits  
 1666 can be different, as per the values returned from the \_Sx predefined methods.

4696 Removed an extra register write within AcpiHwClearAcpiStatus. This  
 4697 function  
 4698 was writing an optional PM1B status register twice. The existing call to  
 4699 the  
 4700 low-level AcpiHwRegisterWrite automatically handles a possibly split PM1  
 4701 A/B  
 1668 Removed an extra register write within AcpiHwClearAcpiStatus. This function  
 1669 was writing an optional PM1B status register twice. The existing call to the  
 1670 low-level AcpiHwRegisterWrite automatically handles a possibly split PM1 A/B  
 4702 register. ACPICA BZ 751.

4704 Split out the PM1 Status registers from the FADT. Added new globals for  
 4705 these  
 1673 Split out the PM1 Status registers from the FADT. Added new globals for these  
 4706 registers (A/B), similar to the way the PM1 Enable registers are handled.  
 4707 Instead of overloading the FADT Event Register blocks. This makes the  
 4708 code  
 1675 Instead of overloading the FADT Event Register blocks. This makes the code  
 4709 clearer and less prone to error.

4711 Fixed the warning message for when the platform contains too many ACPI  
 4712 tables  
 4713 for the default size of the global root table data structure. The  
 4714 calculation  
 1678 Fixed the warning message for when the platform contains too many ACPI tables  
 1679 for the default size of the global root table data structure. The calculation  
 4715 for the truncation value was incorrect.

4717 Removed the ACPI\_GET\_OBJECT\_TYPE macro. Removed all instances of this  
 4718 obsolete macro, since it is now a simple reference to ->common.type.  
 4719 There  
 1683 obsolete macro, since it is now a simple reference to ->common.type. There  
 4720 were about 150 invocations of the macro across 41 files. ACPICA BZ 755.

4722 Removed the redundant ACPI\_BITREG\_SLEEP\_TYPE\_B. This type is the same as  
 4723 TYPE\_A. Removed this and all related instances. Renamed SLEEP\_TYPE\_A to  
 4724 simply SLEEP\_TYPE. ACPICA BZ 754.

4726 Conditionally compile the AcpiSetFirmwareWakingVector64 function. This  
 4727 function is only needed on 64-bit host operating systems and is thus not  
 4728 included for 32-bit hosts.

4730 Debug output: print the input and result for invocations of the \_OSI  
 4731 reserved  
 4732 control method via the ACPI\_LV\_INFO debug level. Also, reduced some of  
 4733 the  
 1694 Debug output: print the input and result for invocations of the \_OSI reserved  
 1695 control method via the ACPI\_LV\_INFO debug level. Also, reduced some of the  
 4734 verbosity of this debug level. Len Brown.

4736 Example Code and Data Size: These are the sizes for the OS-independent  
 4737 apica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 4738 debug version of the code includes the debug output trace mechanism and  
 4739 has a  
 1700 debug version of the code includes the debug output trace mechanism and has a

4740 much larger code and data size.

4742 Previous Release:  
 4743 Non-Debug Version: 82.3K Code, 17.5K Data, 99.8K Total  
 4744 Debug Version: 157.3K Code, 49.8K Data, 207.1K Total  
 4745 Current Release:  
 4746 Non-Debug Version: 82.0K Code, 17.5K Data, 99.5K Total  
 4747 Debug Version: 156.9K Code, 49.8K Data, 206.7K Total

4749 2) iASL Compiler/Disassembler and Tools:

4751 Disassembler: Decode the FADT PM\_Profile field. Emit ascii names for the  
 4752 various legal performance profiles.

4754 -----  
 4755 23 January 2009. Summary of changes for version 20090123:

4757 1) ACPI CA Core Subsystem:

4759 Added the 2009 copyright to all module headers and signons. This affects  
 4760 virtually every file in the ACPICA core subsystem, the iASL compiler, and  
 4761 the tools/utilities.

4763 Implemented a change to allow the host to override any ACPI table,  
 4764 including  
 4765 dynamically loaded tables. Previously, only the DSDT could be replaced by  
 4766 the  
 4767 host. With this change, the AcpiOsTableOverride interface is called for  
 4768 each  
 4769 table found in the RSDT/XSDT during ACPICA initialization, and also  
 4770 whenever  
 1724 Implemented a change to allow the host to override any ACPI table, including  
 1725 dynamically loaded tables. Previously, only the DSDT could be replaced by the  
 1726 host. With this change, the AcpiOsTableOverride interface is called for each  
 1727 table found in the RSDT/XSDT during ACPICA initialization, and also whenever  
 4771 a table is dynamically loaded via the AML Load operator.

4773 Updated FADT flag definitions, especially the Boot Architecture flags.

4775 Debugger: For the Find command, automatically pad the input ACPI name  
 4776 with  
 4777 underscores if the name is shorter than 4 characters. This enables a  
 4778 match  
 1732 Debugger: For the Find command, automatically pad the input ACPI name with  
 1733 underscores if the name is shorter than 4 characters. This enables a match  
 4779 with the actual namespace entry which is itself padded with underscores.

4781 Example Code and Data Size: These are the sizes for the OS-independent  
 4782 apica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 4783 debug version of the code includes the debug output trace mechanism and  
 4784 has a  
 1738 debug version of the code includes the debug output trace mechanism and has a  
 4785 much larger code and data size.

4787 Previous Release:  
 4788 Non-Debug Version: 82.3K Code, 17.4K Data, 99.7K Total  
 4789 Debug Version: 157.1K Code, 49.7K Data, 206.8K Total  
 4790 Current Release:  
 4791 Non-Debug Version: 82.3K Code, 17.5K Data, 99.8K Total  
 4792 Debug Version: 157.3K Code, 49.8K Data, 207.1K Total

4794 2) iASL Compiler/Disassembler and Tools:

4796 Fix build error under Bison-2.4.

4798 Disassembler: Enhanced FADT support. Added decoding of the Boot

4799 **Architecture**

1752 *Dissassembler: Enhanced FADT support. Added decoding of the Boot Architecture*  
 4800 flags. Now decode all flags, regardless of the FADT version. Flag output  
 4801 includes the FADT version which first defined each flag.

4803 **The iASL -g option now dumps the RSDT to a file (in addition to the FADT**  
 4804 **and**  
 1756 *The iASL -g option now dumps the RSDT to a file (in addition to the FADT and*  
 4805 DSDT). Windows only.

4807 -----  
 4808 04 December 2008. Summary of changes for version 20081204:

4810 1) ACPI CA Core Subsystem:

4812 **The ACPICA Programmer Reference has been completely updated and revamped**  
 4813 **for**  
 1764 *The ACPICA Programmer Reference has been completely updated and revamped for*  
 4814 this release. This includes updates to the external interfaces, OSL  
 4815 interfaces, the overview sections, and the debugger reference.

4817 Several new ACPICA interfaces have been implemented and documented in the  
 4818 programmer reference:  
 4819 AcpiReset - Writes the reset value to the FADT-defined reset register.  
 4820 AcpiDisableAllGpes - Disable all available GPEs.  
 4821 AcpiEnableAllRuntimeGpes - Enable all available runtime GPEs.  
 4822 AcpiGetGpeDevice - Get the GPE block device associated with a GPE.  
 4823 AcpiGbl\_CurrentGpeCount - Tracks the current number of available GPEs.  
 4824 AcpiRead - Low-level read ACPI register (was HwLowLevelRead.)  
 4825 AcpiWrite - Low-level write ACPI register (was HwLowLevelWrite.)

4827 **Most of the public ACPI hardware-related interfaces have been moved to a**  
 4828 **new**  
 1778 *Most of the public ACPI hardware-related interfaces have been moved to a new*  
 4829 file, components/hardware/hwxface.c

4831 Enhanced the FADT parsing and low-level ACPI register access: The ACPI  
 4832 register lengths within the FADT are now used, and the low level ACPI  
 4833 register access no longer hardcodes the ACPI register lengths. Given that  
 4834 **there may be some risk in actually trusting the FADT register lengths, a**  
 4835 **run-**  
 4836 **time option was added to fall back to the default hardcoded lengths if**  
 4837 **the**  
 1784 *there may be some risk in actually trusting the FADT register lengths, a run-*  
 1785 *time option was added to fall back to the default hardcoded lengths if the*  
 4838 FADT proves to contain incorrect values - UseDefaultRegisterWidths. This  
 4839 **option is set to true for now, and a warning is issued if a suspicious**  
 4840 **FADT**  
 1787 *option is set to true for now, and a warning is issued if a suspicious FADT*  
 4841 register length is overridden with the default value.

4843 **Fixed a reference count issue in NsRepairObject. This problem was**  
 4844 **introduced**  
 1790 *Fixed a reference count issue in NsRepairObject. This problem was introduced*  
 4845 in version 20081031 as part of a fix to repair Buffer objects within  
 4846 Packages. Lin Ming.

4848 Added semaphore support to the Linux/Unix application OS-services layer  
 4849 (OSL). ACPICA BZ 448. Lin Ming.

4851 **Added the ACPI\_MUTEX\_TYPE configuration option to select whether mutexes**  
 4852 **will**  
 1797 *Added the ACPI\_MUTEX\_TYPE configuration option to select whether mutexes will*  
 4853 be implemented in the OSL, or will binary semaphores be used instead.

4855 Example Code and Data Size: These are the sizes for the OS-independent

4856 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 4857 **debug version of the code includes the debug output trace mechanism and**  
 4858 **has a**  
 1802 *debug version of the code includes the debug output trace mechanism and has a*  
 4859 much larger code and data size.

4861 Previous Release:  
 4862 Non-Debug Version: 81.7K Code, 17.3K Data, 99.0K Total  
 4863 Debug Version: 156.4K Code, 49.4K Data, 205.8K Total  
 4864 Current Release:  
 4865 Non-Debug Version: 82.3K Code, 17.4K Data, 99.7K Total  
 4866 Debug Version: 157.1K Code, 49.7K Data, 206.8K Total

4868 2) iASL Compiler/Disassembler and Tools:

4870 **iASL: Completed the '-e' option to include additional ACPI tables in**  
 4871 **order**  
 4872 **to**  
 4873 **aid with disassembly and External statement generation. ACPICA BZ 742.**  
 4874 **Lin**  
 1814 *iASL: Completed the '-e' option to include additional ACPI tables in order to*  
 1815 *aid with disassembly and External statement generation. ACPICA BZ 742. Lin*  
 4875 Ming.

4877 iASL: Removed the "named object in while loop" error. The compiler cannot  
 4878 determine how many times a loop will execute. ACPICA BZ 730.

4880 **Disassembler: Implemented support for FADT revision 2 (MS extension).**  
 4881 **ACPICA**  
 1821 *Disassembler: Implemented support for FADT revision 2 (MS extension). ACPICA*  
 4882 BZ 743.

4884 **Disassembler: Updates for several ACPI data tables (HEST, EINJ, and**  
 4885 **MCFG).**  
 1824 *Disassembler: Updates for several ACPI data tables (HEST, EINJ, and MCFG).*

4887 -----  
 4888 31 October 2008. Summary of changes for version 20081031:

4890 1) ACPI CA Core Subsystem:

4892 **Restructured the ACPICA header files into public/private. acpi.h now**  
 4893 **includes**  
 4894 **only the "public" acpica headers. All other acpica headers are "private"**  
 4895 **and**  
 4896 **should not be included by acpica users. One new file, accommon.h is used**  
 4897 **to**  
 4898 **include the commonly used private headers for acpica code generation.**  
 4899 **Future**  
 1831 *Restructured the ACPICA header files into public/private. acpi.h now includes*  
 1832 *only the "public" acpica headers. All other acpica headers are "private" and*  
 1833 *should not be included by acpica users. One new file, accommon.h is used to*  
 1834 *include the commonly used private headers for acpica code generation. Future*  
 4900 plans include moving all private headers to a new subdirectory.

4902 Implemented an automatic Buffer->String return value conversion for  
 4903 **predefined ACPI methods. For these methods (such as \_BIF), added**  
 4904 **automatic**  
 4905 **conversion for return objects that are required to be a String, but a**  
 4906 **Buffer**  
 4907 **was found instead. This can happen when reading string battery data from**  
 4908 **an**  
 4909 **operation region, because it used to be difficult to convert the data**  
 4910 **from**  
 4911 **buffer to string from within the ASL. Ensures that the host OS is**  
 4912 **provided**

1838 predefined ACPI methods. For these methods (such as \_BIF), added automatic  
 1839 conversion for return objects that are required to be a String, but a Buffer  
 1840 was found instead. This can happen when reading string battery data from an  
 1841 operation region, because it used to be difficult to convert the data from  
 1842 buffer to string from within the ASL. Ensures that the host OS is provided  
 1913 with a valid null-terminated string. Linux BZ 11822.

4915 Updated the FACS waking vector interfaces. Split  
 4916 AcpiSetFirmwareWakingVector  
 4917 into two: one for the 32-bit vector, another for the 64-bit vector. This  
 4918 is  
 4919 required because the host OS must setup the wake much differently for  
 4920 each  
 4921 vector (real vs. protected mode, etc.) and the interface itself should  
 4922 not  
 4923 be  
 4924 deciding which vector to use. Also, eliminated the  
 4925 GetFirmwareWakingVector  
 4926 interface, as it served no purpose (only the firmware reads the vector,  
 4927 OS  
 1845 Updated the FACS waking vector interfaces. Split AcpiSetFirmwareWakingVector  
 1846 into two: one for the 32-bit vector, another for the 64-bit vector. This is  
 1847 required because the host OS must setup the wake much differently for each  
 1848 vector (real vs. protected mode, etc.) and the interface itself should not be  
 1849 deciding which vector to use. Also, eliminated the GetFirmwareWakingVector  
 1850 interface, as it served no purpose (only the firmware reads the vector, OS  
 4928 only writes the vector.) ACPICA BZ 731.

4930 Implemented a mechanism to escape infinite AML While() loops. Added a  
 4931 loop  
 4932 counter to force exit from AML While loops if the count becomes too  
 4933 large.  
 1853 Implemented a mechanism to escape infinite AML While() loops. Added a loop  
 1854 counter to force exit from AML While loops if the count becomes too large.  
 4934 This can occur in poorly written AML when the hardware does not respond  
 4935 within a while loop and the loop does not implement a timeout. The  
 4936 maximum  
 4937 loop count is configurable. A new exception code is returned when a loop  
 4938 is  
 1856 within a while loop and the loop does not implement a timeout. The maximum  
 1857 loop count is configurable. A new exception code is returned when a loop is  
 4939 broken, AE\_AML\_INFINITE\_LOOP. Alexey Starikovskiy, Bob Moore.

4941 Optimized the execution of AML While loops. Previously, a control state  
 4942 object was allocated and freed for each execution of the loop. The  
 4943 optimization is to simply reuse the control state for each iteration.  
 4944 This  
 1862 optimization is to simply reuse the control state for each iteration. This  
 4945 speeds up the raw loop execution time by about 5%.

4947 Enhanced the implicit return mechanism. For Windows compatibility, return  
 4948 an  
 4949 implicit integer of value zero for methods that contain no executable  
 4950 code.  
 1865 Enhanced the implicit return mechanism. For Windows compatibility, return an  
 1866 implicit integer of value zero for methods that contain no executable code.  
 4951 Such methods are seen in the field as stubs (presumably), and can cause  
 4952 drivers to fail if they expect a return value. Lin Ming.

4954 Allow multiple backslashes as root prefixes in namepaths. In a fully  
 4955 qualified namepath, allow multiple backslash prefixes. This can happen  
 4956 (and  
 1871 qualified namepath, allow multiple backslash prefixes. This can happen (and  
 4957 is seen in the field) because of the use of a double-backslash in strings  
 4958 (since backslash is the escape character) causing confusion. ACPICA BZ  
 4959 739

1873 (since backslash is the escape character) causing confusion. ACPICA BZ 739  
 4960 Lin Ming.

4962 Emit a warning if two different FACS or DSDT tables are discovered in the  
 4963 FADT. Checks if there are two valid but different addresses for the FACS  
 4964 and  
 1877 FADT. Checks if there are two valid but different addresses for the FACS and  
 4965 DSDT within the FADT (mismatch between the 32-bit and 64-bit fields.)

4967 Consolidated the method argument count validation code. Merged the code  
 4968 that  
 1880 Consolidated the method argument count validation code. Merged the code that  
 4969 validates control method argument counts into the predefined validation  
 4970 module. Eliminates possible multiple warnings for incorrect argument  
 4971 counts.  
 1882 module. Eliminates possible multiple warnings for incorrect argument counts.

4973 Implemented ACPICA example code. Includes code for ACPICA initialization,  
 4974 handler installation, and calling a control method. Available at  
 4975 source/tools/examples.

4977 Added a global pointer for FACS table to simplify internal FACS access.  
 4978 Use  
 4979 the global pointer instead of using AcpiGetTableByIndex for each FACS  
 4980 access.  
 1888 Added a global pointer for FACS table to simplify internal FACS access. Use  
 1889 the global pointer instead of using AcpiGetTableByIndex for each FACS access.  
 4981 This simplifies the code for the Global Lock and the Firmware Waking  
 4982 Vector(s).

4984 Example Code and Data Size: These are the sizes for the OS-independent  
 4985 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 4986 debug version of the code includes the debug output trace mechanism and  
 4987 has a  
 1895 debug version of the code includes the debug output trace mechanism and has a  
 4988 much larger code and data size.

4990 Previous Release:  
 4991 Non-Debug Version: 81.2K Code, 17.0K Data, 98.2K Total  
 4992 Debug Version: 155.8K Code, 49.1K Data, 204.9K Total  
 4993 Current Release:  
 4994 Non-Debug Version: 81.7K Code, 17.3K Data, 99.0K Total  
 4995 Debug Version: 156.4K Code, 49.4K Data, 205.8K Total

4997 2) iASL Compiler/Disassembler and Tools:

4999 iASL: Improved disassembly of external method calls. Added the -e option  
 5000 to  
 5001 allow the inclusion of additional ACPI tables to help with the  
 5002 disassembly  
 5003 of  
 1907 iASL: Improved disassembly of external method calls. Added the -e option to  
 1908 allow the inclusion of additional ACPI tables to help with the disassembly of  
 5004 method invocations and the generation of external declarations during the  
 5005 disassembly. Certain external method invocations cannot be disassembled  
 5006 properly without the actual declaration of the method. Use the -e option  
 5007 to  
 5008 include the table where the external method(s) are actually declared.  
 5009 Most  
 1911 properly without the actual declaration of the method. Use the -e option to  
 1912 include the table where the external method(s) are actually declared. Most  
 5010 useful for disassembling SSDTs that make method calls back to the master  
 5011 DSDT. Lin Ming. Example: To disassemble an SSDT with calls to DSDT: iasl  
 5012 -d  
 1914 DSDT. Lin Ming. Example: To disassemble an SSDT with calls to DSDT: iasl -d  
 5013 -e dsdt.aml ssdt1.aml

5015 iASL: Fix to allow references to aliases within ASL namepaths. Fixes a  
 5016 problem where the use of an alias within a namepath would result in a not  
 5017 found error or cause the compiler to fault. Also now allows forward  
 5018 references from the Alias operator itself. ACPICA BZ 738.

5020 -----  
 5021 26 September 2008. Summary of changes for version 20080926:

5023 1) ACPI CA Core Subsystem:

5025 **Designed and implemented a mechanism to validate predefined ACPI methods**  
 5026 **and**  
 5027 **objects. This code validates the predefined ACPI objects (objects whose**  
 5028 **names**  
 1927 *Designed and implemented a mechanism to validate predefined ACPI methods and*  
 1928 *objects. This code validates the predefined ACPI objects (objects whose names*  
 5029 *start with underscore) that appear in the namespace, at the time they are*  
 5030 *evaluated. The argument count and the type of the returned object are*  
 5031 **validated against the ACPI specification. The purpose of this validation**  
 5032 **is**  
 5033 **to detect problems with the BIOS-implemented predefined ACPI objects**  
 5034 **before**  
 5035 **the results are returned to the ACPI-related drivers. Future enhancements**  
 5036 **may**  
 1931 *validated against the ACPI specification. The purpose of this validation is*  
 1932 *to detect problems with the BIOS-implemented predefined ACPI objects before*  
 1933 *the results are returned to the ACPI-related drivers. Future enhancements may*  
 5037 *include actual repair of incorrect return objects where possible. Two new*  
 5038 *files are nspredef.c and acpredef.h.*

5040 **Fixed a fault in the AML parser if a memory allocation fails during the**  
 5041 **Op**  
 1937 *Fixed a fault in the AML parser if a memory allocation fails during the Op*  
 5042 *completion routine AcpiPsCompleteThisOp. Lin Ming. ACPICA BZ 492.*

5044 **Fixed an issue with implicit return compatibility. This change improves**  
 5045 **the**  
 5046 **implicit return mechanism to be more compatible with the MS interpreter.**  
 5047 **Lin**  
 1940 *Fixed an issue with implicit return compatibility. This change improves the*  
 1941 *implicit return mechanism to be more compatible with the MS interpreter. Lin*  
 5048 *Ming, ACPICA BZ 349.*

5050 **Implemented support for zero-length buffer-to-string conversions. Allow**  
 5051 **zero**  
 5052 **length strings during interpreter buffer-to-string conversions. For**  
 5053 **example,**  
 1944 *Implemented support for zero-length buffer-to-string conversions. Allow zero*  
 1945 *length strings during interpreter buffer-to-string conversions. For example,*  
 5054 *during the ToDecimalString and ToHexString operators, as well as implicit*  
 5055 *conversions. Fiodor Suietov, ACPICA BZ 585.*

5057 **Fixed two possible memory leaks in the error exit paths of**  
 5058 **AcpiUtUpdateObjectReference and AcpiUtWalkPackageTree. These functions**  
 5059 **are**  
 1950 *AcpiUtUpdateObjectReference and AcpiUtWalkPackageTree. These functions are*  
 5060 *similar in that they use a stack of state objects in order to eliminate*  
 5061 *recursion. The stack must be fully unwound and deallocated if an error*  
 5062 *occurs. Lin Ming. ACPICA BZ 383.*

5064 **Removed the unused ACPI\_BITREG\_WAKE\_ENABLE definition and entry in the**  
 5065 **global**  
 1955 *Removed the unused ACPI\_BITREG\_WAKE\_ENABLE definition and entry in the global*  
 5066 *ACPI register table. This bit does not exist and is unused. Lin Ming, Bob*  
 5067 *Moore ACPICA BZ 442.*

5069 **Removed the obsolete version number in module headers. Removed the**  
 5070 **"\$Revision" number that appeared in each module header. This version**  
 5071 **number**  
 5072 **was useful under SourceSafe and CVS, but has no meaning under git. It is**  
 5073 **not**  
 1960 *"\$Revision" number that appeared in each module header. This version number*  
 1961 *was useful under SourceSafe and CVS, but has no meaning under git. It is not*  
 5074 *only incorrect, it could also be misleading.*

5076 **Example Code and Data Size: These are the sizes for the OS-independent**  
 5077 **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The**  
 5078 **debug version of the code includes the debug output trace mechanism and**  
 5079 **has a**  
 1966 *debug version of the code includes the debug output trace mechanism and has a*  
 5080 *much larger code and data size.*

5082 Previous Release:  
 5083 Non-Debug Version: 79.7K Code, 16.4K Data, 96.1K Total  
 5084 Debug Version: 153.7K Code, 48.2K Data, 201.9K Total  
 5085 Current Release:  
 5086 Non-Debug Version: 81.2K Code, 17.0K Data, 98.2K Total  
 5087 Debug Version: 155.8K Code, 49.1K Data, 204.9K Total

5089 -----  
 5090 29 August 2008. Summary of changes for version 20080829:

5092 1) ACPI CA Core Subsystem:

5094 **Completed a major cleanup of the internal ACPI\_OPERAND\_OBJECT of type**  
 5095 **Reference. Changes include the elimination of cheating on the Object**  
 5096 **field**  
 1982 *Reference. Changes include the elimination of cheating on the Object field*  
 5097 *for the DdbHandle subtype, addition of a reference class field to*  
 5098 **differentiate the various reference types (instead of an AML opcode), and**  
 5099 **the**  
 1984 *differentiate the various reference types (instead of an AML opcode), and the*  
 5100 *cleanup of debug output for this object. Lin Ming, Bob Moore. BZ 723*

5102 **Reduce an error to a warning for an incorrect method argument count.**  
 5103 **Previously aborted with an error if too few arguments were passed to a**  
 5104 **control method via the external ACPICA interface. Now issue a warning**  
 5105 **instead**  
 5106 **and continue. Handles the case where the method inadvertently declares**  
 5107 **too**  
 5108 **many arguments, but does not actually use the extra ones. Applies mainly**  
 5109 **to**  
 1989 *control method via the external ACPICA interface. Now issue a warning instead*  
 1990 *and continue. Handles the case where the method inadvertently declares too*  
 1991 *many arguments, but does not actually use the extra ones. Applies mainly to*  
 5110 *the predefined methods. Lin Ming. Linux BZ 11032.*

5112 **Disallow the evaluation of named object types with no intrinsic value.**  
 5113 **Return**  
 5114 **AE\_TYPE for objects that have no value and therefore evaluation is**  
 5115 **undefined:**  
 5116 **Device, Event, Mutex, Region, Thermal, and Scope. Previously, evaluation**  
 5117 **of**  
 5118 **these types were allowed, but an exception would be generated at some**  
 5119 **point**  
 1994 *Disallow the evaluation of named object types with no intrinsic value. Return*  
 1995 *AE\_TYPE for objects that have no value and therefore evaluation is undefined:*  
 1996 *Device, Event, Mutex, Region, Thermal, and Scope. Previously, evaluation of*  
 1997 *these types were allowed, but an exception would be generated at some point*  
 5120 *during the evaluation. Now, the error is generated up front.*

5122 Fixed a possible memory leak in the AcpiNsGetExternalPathname function  
 5123 (nsnames.c). Fixes a leak in the error exit path.

5125 **Removed the obsolete debug levels ACPI\_DB\_WARN and ACPI\_DB\_ERROR. These**  
 5126 **debug**  
 5127 **levels were made obsolete by the ACPI\_WARNING, ACPI\_ERROR, and**  
 5128 **ACPI\_EXCEPTION**  
 2003 *Removed the obsolete debug levels ACPI\_DB\_WARN and ACPI\_DB\_ERROR. These debug*  
 2004 *levels were made obsolete by the ACPI\_WARNING, ACPI\_ERROR, and ACPI\_EXCEPTION*  
 5129 *interfaces. Also added ACPI\_DB\_EVENTS to correspond with the existing*  
 5130 *ACPI\_LV\_EVENTS.*

5132 Removed obsolete and/or unused exception codes from the acexcep.h header.  
 5133 **There is the possibility that certain device drivers may be affected if**  
 5134 **they**  
 2009 *There is the possibility that certain device drivers may be affected if they*  
 5135 *use any of these exceptions.*

5137 **The ACPICA documentation has been added to the public git source tree,**  
 5138 **under**  
 2012 *The ACPICA documentation has been added to the public git source tree, under*  
 5139 *acpica/documents. Included are the ACPICA programmer reference, the iASL*  
 5140 *compiler reference, and the changes.txt release logfile.*

5142 Example Code and Data Size: These are the sizes for the OS-independent  
 5143 apica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 5144 **debug version of the code includes the debug output trace mechanism and**  
 5145 **has a**  
 2018 *debug version of the code includes the debug output trace mechanism and has a*  
 5146 *much larger code and data size.*

5148 Previous Release:  
 5149 Non-Debug Version: 79.7K Code, 16.4K Data, 96.1K Total  
 5150 Debug Version: 153.9K Code, 48.4K Data, 202.3K Total  
 5151 Current Release:  
 5152 Non-Debug Version: 79.7K Code, 16.4K Data, 96.1K Total  
 5153 Debug Version: 153.7K Code, 48.2K Data, 201.9K Total

5155 2) iASL Compiler/Disassembler and Tools:

5157 Allow multiple argument counts for the predefined \_SCP method. ACPI 3.0  
 5158 defines \_SCP with 3 arguments. Previous versions defined it with only 1  
 5159 argument. iASL now allows both definitions.

5161 **iASL/disassembler: avoid infinite loop on bad ACPI tables. Check for**  
 5162 **zero-**  
 2034 *iASL/disassembler: avoid infinite loop on bad ACPI tables. Check for zero-*  
 5163 *length subtables when disassembling ACPI tables. Also fixed a couple of*  
 5164 *errors where a full 16-bit table type field was not extracted from the*  
 5165 *input*  
 2036 *errors where a full 16-bit table type field was not extracted from the input*  
 5166 *properly.*

5168 acpisrc: Improve comment counting mechanism for generating source code  
 5169 **statistics. Count first and last lines of multi-line comments as**  
 5170 **whitespace,**  
 5171 **not comment lines. Handle Linux legal header in addition to standard**  
 5172 **acpica**  
 2040 *statistics. Count first and last lines of multi-line comments as whitespace,*  
 2041 *not comment lines. Handle Linux legal header in addition to standard acpica*  
 5173 *header.*

5175 -----

5177 29 July 2008. Summary of changes for version 20080729:

5179 1) ACPI CA Core Subsystem:

5181 Fix a possible deadlock in the GPE dispatch. Remove call to  
 5182 **AcpiHwDisableAllGpes** during wake in **AcpiEvGpeDispatch**. This call will  
 5183 **attempt**  
 5184 **to acquire the GPE lock but can deadlock since the GPE lock is already**  
 5185 **held**  
 5186 **at dispatch time. This code was introduced in version 20060831 as a**  
 5187 **response**  
 2051 *AcpiHwDisableAllGpes* during wake in *AcpiEvGpeDispatch*. This call will attempt  
 2052 *to acquire the GPE lock but can deadlock since the GPE lock is already held*  
 2053 *at dispatch time. This code was introduced in version 20060831 as a response*  
 5188 *to Linux BZ 6881 and has since been removed from Linux.*

5190 **Add a function to dereference returned reference objects. Examines the**  
 5191 **return**  
 5192 **object from a call to AcpiEvaluateObject. Any Index or RefOf references**  
 5193 **are**  
 5194 **automatically dereferenced in an attempt to return something useful**  
 5195 **(these**  
 5196 **reference types cannot be converted into an external ACPI\_OBJECT.)**  
 5197 **Provides**  
 2056 *Add a function to dereference returned reference objects. Examines the return*  
 2057 *object from a call to AcpiEvaluateObject. Any Index or RefOf references are*  
 2058 *automatically dereferenced in an attempt to return something useful (these*  
 2059 *reference types cannot be converted into an external ACPI\_OBJECT.) Provides*  
 5198 *MS compatibility. Lin Ming, Bob Moore. Linux BZ 11105*

5200 x2APIC support: changes for MADT and SRAT ACPI tables. There are 2 new  
 5201 subtables for the MADT and one new subtable for the SRAT. Includes  
 5202 **disassembler and AcpiSrc support. Data from the Intel 64 Architecture**  
 5203 **x2APIC**  
 2064 *disassembler and AcpiSrc support. Data from the Intel 64 Architecture x2APIC*  
 5204 *Specification, June 2008.*

5206 **Additional error checking for pathname utilities. Add error check after**  
 5207 **all**  
 2067 *Additional error checking for pathname utilities. Add error check after all*  
 5208 *calls to AcpiNsGetPathnameLength. Add status return from*  
 5209 **AcpiNsBuildExternalPath and check after all calls. Add parameter**  
 5210 **validation**  
 2069 *AcpiNsBuildExternalPath and check after all calls. Add parameter validation*  
 5211 *to AcpiUtInitializeBuffer. Reported by and initial patch by Ingo Molnar.*

5213 **Return status from the global init function AcpiUtGlobalInitialize. This**  
 5214 **is**  
 5215 **used by both the kernel subsystem and the utilities such as iASL**  
 5216 **compiler.**  
 5217 **The function could possibly fail when the caches are initialized. Yang**  
 5218 **Yi.**  
 2072 *Return status from the global init function AcpiUtGlobalInitialize. This is*  
 2073 *used by both the kernel subsystem and the utilities such as iASL compiler.*  
 2074 *The function could possibly fail when the caches are initialized. Yang Yi.*

5220 Add a function to decode reference object types to strings. Created for  
 5221 improved error messages.

5223 **Improve object conversion error messages. Better error messages during**  
 5224 **object**  
 5225 **conversion from internal to the external ACPI\_OBJECT. Used for external**  
 5226 **calls**  
 2079 *Improve object conversion error messages. Better error messages during object*  
 2080 *conversion from internal to the external ACPI\_OBJECT. Used for external calls*  
 5227 *to AcpiEvaluateObject.*

5229 Example Code and Data Size: These are the sizes for the OS-independent



5230 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 5231 **debug version of the code includes the debug output trace mechanism and**  
 5232 **has a**  
 5233 *debug version of the code includes the debug output trace mechanism and has a*  
 much larger code and data size.

5235 Previous Release:  
 5236 Non-Debug Version: 79.6K Code, 16.2K Data, 95.8K Total  
 5237 Debug Version: 153.5K Code, 48.2K Data, 201.7K Total  
 5238 Current Release:  
 5239 Non-Debug Version: 79.7K Code, 16.4K Data, 96.1K Total  
 5240 Debug Version: 153.9K Code, 48.4K Data, 202.3K Total

5242 2) iASL Compiler/Disassembler and Tools:

5244 **Debugger: fix a possible hang when evaluating non-methods. Fixes a**  
 5245 **problem**  
 5246 **introduced in version 20080701. If the object being evaluated (via**  
 5247 **execute**  
 5248 **command) is not a method, the debugger can hang while trying to obtain**  
 5249 **non-**  
 5297 *Debugger: fix a possible hang when evaluating non-methods. Fixes a problem*  
 5298 *introduced in version 20080701. If the object being evaluated (via execute*  
 5299 *command) is not a method, the debugger can hang while trying to obtain non-*  
 5250 *existent parameters.*

5252 iASL: relax error for using reserved "\_T\_x" identifiers. These names can  
 5253 appear in a disassembled ASL file if they were emitted by the original  
 5254 compiler. Instead of issuing an error or warning and forcing the user to  
 5255 manually change these names, issue a remark instead.

5257 **iASL: error if named object created in while loop. Emit an error if any**  
 5258 **named**  
 5259 **object is created within a While loop. If allowed, this code will**  
 5260 **generate**  
 5261 **a**  
 5262 **run-time error on the second iteration of the loop when an attempt is**  
 5263 **made**  
 5264 **to**  
 5297 *iASL: error if named object created in while loop. Emit an error if any named*  
 5298 *object is created within a While loop. If allowed, this code will generate a*  
 5299 *run-time error on the second iteration of the loop when an attempt is made to*  
 5265 *create the same named object twice. ACPICA bugzilla 730.*

5267 **iASL: Support absolute pathnames for include files. Add support for**  
 5268 **absolute**  
 5269 **pathnames within the Include operator. previously, only relative**  
 5270 **pathnames**  
 5297 *iASL: Support absolute pathnames for include files. Add support for absolute*  
 5298 *pathnames within the Include operator. previously, only relative pathnames*  
 5271 *were supported.*

5273 **iASL: Enforce minimum 1 interrupt in interrupt macro and Resource**  
 5274 **Descriptor.**  
 5297 *iASL: Enforce minimum 1 interrupt in interrupt macro and Resource Descriptor.*  
 5275 *The ACPI spec requires one interrupt minimum. BZ 423*

5277 iASL: Handle a missing ResourceSource arg, with a present SourceIndex.  
 5278 Handles the case for the Interrupt Resource Descriptor where  
 5279 the ResourceSource argument is omitted but ResourceSourceIndex  
 5280 is present. Now leave room for the Index. BZ 426

5282 **iASL: Prevent error message if CondRefOf target does not exist. Fixes**  
 5283 **cases**  
 5297 *iASL: Prevent error message if CondRefOf target does not exist. Fixes cases*  
 5284 *where an error message is emitted if the target does not exist. BZ 516*

5286 iASL: Fix broken -g option (get Windows ACPI tables). Fixes the -g option  
 5287 (get ACPI tables on Windows). This was apparently broken in version  
 5288 20070919.  
 5297 *(get ACPI tables on Windows). This was apparently broken in version 20070919.*

5290 **AcpiXtract: Handle EOF while extracting data. Correctly handle the case**  
 5291 **where**  
 5292 *AcpiXtract: Handle EOF while extracting data. Correctly handle the case where*  
 5292 *the EOF happens immediately after the last table in the input file. Print*  
 5293 *completion message. Previously, no message was displayed in this case.*

5295 -----  
 5296 01 July 2008. Summary of changes for version 20080701:

5298 0) Git source tree / acpica.org

5300 Fixed a problem where a git-clone from http would not transfer the entire  
 5301 source tree.

5303 1) ACPI CA Core Subsystem:

5305 Implemented a "careful" GPE disable in AcpiEvDisableGpe, only modify one  
 5306 **enable bit. Now performs a read-change-write of the enable register**  
 5307 **instead**  
 5308 **of simply writing out the cached enable mask. This will prevent**  
 5309 **inadvertent**  
 5310 **enabling of GPEs if a rogue GPE is received during initialization (before**  
 5311 **GPE**  
 5312 *enable bit. Now performs a read-change-write of the enable register instead*  
 5313 *of simply writing out the cached enable mask. This will prevent inadvertent*  
 5314 *enabling of GPEs if a rogue GPE is received during initialization (before GPE*  
 5312 *handlers are installed.)*

5314 Implemented a copy for dynamically loaded tables. Previously, dynamically  
 5315 loaded tables were simply mapped - but on some machines this memory is  
 5316 corrupted after suspend. Now copy the table to a local buffer. For the  
 5317 OpRegion case, added checksum verify. Use the table length from the table  
 5318 header, not the region length. For the Buffer case, use the table length  
 5319 also. Dennis Noordsij, Bob Moore. BZ 10734

5321 **Fixed a problem where the same ACPI table could not be dynamically loaded**  
 5322 **and**  
 5323 **unloaded more than once. Without this change, a table cannot be loaded**  
 5324 **again**  
 5325 *Fixed a problem where the same ACPI table could not be dynamically loaded and*  
 5326 *unloaded more than once. Without this change, a table cannot be loaded again*  
 5325 *once it has been loaded/unloaded one time. The current mechanism does not*  
 5326 **unregister a table upon an unload. During a load, if the same table is**  
 5327 **found,**  
 5328 *unregister a table upon an unload. During a load, if the same table is found,*  
 5328 *this no longer returns an exception. BZ 722*

5330 Fixed a problem where the wrong descriptor length was calculated for the  
 5331 **EndTag descriptor in 64-bit mode. The "minimal" descriptors such as**  
 5332 **EndTag**  
 5333 *EndTag descriptor in 64-bit mode. The "minimal" descriptors such as EndTag*  
 5333 *are calculated as 12 bytes long, but the actual length in the internal*  
 5334 **descriptor is 16 because of the round-up to 8 on the 64-bit build.**  
 5335 **Reported**  
 5336 *descriptor is 16 because of the round-up to 8 on the 64-bit build. Reported*  
 5336 *by Linn Crosetto. BZ 728*

5338 **Fixed a possible memory leak in the Unload operator. The DdbHandle**  
 5339 **returned**  
 5340 **by Load() did not have its reference count decremented during unload,**

5341 **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*  
 5342 *to a memory leak. Lin Ming. BZ 727*

5344 Fixed a possible memory leak when deleting thermal/processor objects. Any  
 5345 associated notify handlers (and objects) were not being deleted. Fiodor  
 5346 Suietov. BZ 506

5348 **Fixed the ordering of the ASCII names in the global mutex table to match**  
 5349 **the**  
 5350 **actual mutex IDs. Used by AcpiUtGetMutexName, a function used for debug**  
 5351 **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.*  
 5352 Vegard Nossun. BZ 726

5354 Enhanced the AcpiGetObjectInfo interface to return the number of required  
 5355 **arguments if the object is a control method. Added this call to the**  
 5356 **debugger**  
 2182 *arguments if the object is a control method. Added this call to the debugger*  
 5357 *so the proper number of default arguments are passed to a method. This*  
 5358 *prevents a warning when executing methods from AcpiExec.*

5360 Added a check for an invalid handle in AcpiGetObjectInfo. Return  
 5361 AE\_BAD\_PARAMETER if input handle is invalid. BZ 474

5363 Fixed an extraneous warning from exconfig.c on the 64-bit build.

5365 Example Code and Data Size: These are the sizes for the OS-independent  
 5366 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 5367 **debug version of the code includes the debug output trace mechanism and**  
 5368 **has a**  
 2193 *debug version of the code includes the debug output trace mechanism and has a*  
 5369 *much larger code and data size.*

5371	Previous Release:	
5372	Non-Debug Version:	79.3K Code, 16.2K Data, 95.5K Total
5373	Debug Version:	153.0K Code, 48.2K Data, 201.2K Total
5374	Current Release:	
5375	Non-Debug Version:	79.6K Code, 16.2K Data, 95.8K Total
5376	Debug Version:	153.5K Code, 48.2K Data, 201.7K Total

5378 2) iASL Compiler/Disassembler and Tools:

5380 iASL: Added two missing ACPI reserved names. Added \_MTP and \_ASZ, both  
 5381 resource descriptor names.

5383 **iASL: Detect invalid ASCII characters in input (windows version). Removed**  
 5384 **the**  
 2208 *iASL: Detect invalid ASCII characters in input (windows version). Removed the*  
 5385 *"-CF" flag from the flex compile, enables correct detection of non-ASCII*  
 5386 *characters in the input. BZ 441*

5388 **iASL: Eliminate warning when result of LoadTable is not used. Eliminate**  
 5389 **the**  
 2212 *iASL: Eliminate warning when result of LoadTable is not used. Eliminate the*  
 5390 *"result of operation not used" warning when the DDB handle returned from*  
 5391 *LoadTable is not used. The warning is not needed. BZ 590*

5393 **AcpiExec: Add support for dynamic table load/unload. Now calls \_CFG**  
 5394 **method**  
 5395 **to**  
 5396 **pass address of table to the AML. Added option to disable OpRegion**  
 5397 **simulation**  
 5398 **to allow creation of an OpRegion with a real address that was passed to**

5399 **\_CFG.**  
 5400 **All of this allows testing of the Load and Unload operators from**  
 5401 **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.*

5403 **Debugger: update tables command for unloaded tables. Handle unloaded**  
 5404 **tables**  
 2221 *Debugger: update tables command for unloaded tables. Handle unloaded tables*  
 5405 *and use the standard table header output routine.*

5407 -----  
 5408 09 June 2008. Summary of changes for version 20080609:

5410 1) ACPI CA Core Subsystem:

5412 **Implemented a workaround for reversed \_PRT entries. A significant number**  
 5413 **of**  
 2229 *Implemented a workaround for reversed \_PRT entries. A significant number of*  
 5414 *BIOSs erroneously reverse the \_PRT SourceName and the SourceIndex. This*  
 5415 **change dynamically detects and repairs this problem. Provides**  
 5416 **compatibility**  
 2231 *change dynamically detects and repairs this problem. Provides compatibility*  
 5417 *with MS ACPI. BZ 6859*

5419 Simplified the internal ACPI hardware interfaces to eliminate the locking  
 5420 flag parameter from Register Read/Write. Added a new external interface,  
 5421 AcpiGetRegisterUnlocked.

5423 **Fixed a problem where the invocation of a GPE control method could hang.**  
 5424 **This**  
 2238 *Fixed a problem where the invocation of a GPE control method could hang. This*  
 5425 *was a regression introduced in 20080514. The new method argument count*  
 5426 *validation mechanism can enter an infinite loop when a GPE method is*  
 5427 **dispatched. Problem fixed by removing the obsolete code that passed GPE**  
 5428 **block**  
 5429 **information to the notify handler via the control method parameter**  
 5430 **pointer.**  
 2241 *dispatched. Problem fixed by removing the obsolete code that passed GPE block*  
 2242 *information to the notify handler via the control method parameter pointer.*

5432 **Fixed a problem where the \_SST execution status was incorrectly returned**  
 5433 **to**  
 5434 **the caller of AcpiEnterSleepStatePrep. This was a regression introduced**  
 5435 **in**  
 2244 *Fixed a problem where the \_SST execution status was incorrectly returned to*  
 2245 *the caller of AcpiEnterSleepStatePrep. This was a regression introduced in*  
 5436 *20080514. \_SST is optional and a NOT\_FOUND exception should never be*  
 5437 *returned. BZ 716*

5439 **Fixed a problem where a deleted object could be accessed from within the**  
 5440 **AML**  
 5441 **parser. This was a regression introduced in version 20080123 as a fix for**  
 5442 **the**  
 2249 *Fixed a problem where a deleted object could be accessed from within the AML*  
 2250 *parser. This was a regression introduced in version 20080123 as a fix for the*  
 5443 *Unload operator. Lin Ming. BZ 10669*

5445 **Cleaned up the debug operand dump mechanism. Eliminated unnecessary**  
 5446 **operands**  
 2253 *Cleaned up the debug operand dump mechanism. Eliminated unnecessary operands*  
 5447 *and eliminated the use of a negative index in a loop. Operands are now*  
 5448 **displayed in the correct order, not backwards. This also fixes a**  
 5449 **regression**

2255 displayed in the correct order, not backwards. This also fixes a regression  
5450 introduced in 20080514 on 64-bit systems where the elimination of  
5451 **ACPI\_NATIVE\_UINT caused the negative index to go large and positive. BZ**  
5452 **715**  
2257 *ACPI\_NATIVE\_UINT caused the negative index to go large and positive. BZ 715*

5454 **Fixed a possible memory leak in EvPciConfigRegionSetup where the error**  
5455 **exit**  
2259 *Fixed a possible memory leak in EvPciConfigRegionSetup where the error exit*  
5456 *path did not delete a locally allocated structure.*

5458 Updated definitions for the DMAR and SRAT tables to synchronize with the  
5459 current specifications. Includes disassembler support.

5461 Fixed a problem in the mutex debug code (in utmutex.c) where an incorrect  
5462 **loop termination value was used. Loop terminated on iteration early,**  
5463 **missing**  
2266 *loop termination value was used. Loop terminated on iteration early, missing*  
5464 *one mutex. Linn Crosetto*

5466 Example Code and Data Size: These are the sizes for the OS-independent  
5467 apica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
5468 **debug version of the code includes the debug output trace mechanism and**  
5469 **has a**  
2271 *debug version of the code includes the debug output trace mechanism and has a*  
5470 *much larger code and data size.*

5472 Previous Release:  
5473 Non-Debug Version: 79.5K Code, 16.2K Data, 95.7K Total  
5474 Debug Version: 153.3K Code, 48.3K Data, 201.6K Total  
5475 Current Release:  
5476 Non-Debug Version: 79.3K Code, 16.2K Data, 95.5K Total  
5477 Debug Version: 153.0K Code, 48.2K Data, 201.2K Total

5479 2) iASL Compiler/Disassembler and Tools:

5481 Disassembler: Implemented support for EisaId() within \_CID objects. Now  
5482 disassemble integer \_CID objects back to EisaId invocations, including  
5483 multiple integers within \_CID packages. Includes single-step support for  
5484 debugger also.

5486 Disassembler: Added support for DMAR and SRAT table definition changes.

5488 -----  
5489 14 May 2008. Summary of changes for version 20080514:

5491 1) ACPI CA Core Subsystem:

5493 Fixed a problem where GPEs were enabled too early during the ACPICA  
5494 initialization. This could lead to "handler not installed" errors on some  
5495 **machines. Moved GPE enable until after \_REG/\_STA/\_INI methods are run.**  
5496 **This**  
5497 **ensures that all operation regions and devices throughout the namespace**  
5498 **have**  
2297 *machines. Moved GPE enable until after \_REG/\_STA/\_INI methods are run. This*  
2298 *ensures that all operation regions and devices throughout the namespace have*  
5499 *been initialized before GPEs are enabled. Alexey Starikovskiy, BZ 9916.*

5501 Implemented a change to the enter sleep code. Moved execution of the \_GTS  
5502 **method to just before setting sleep enable bit. The execution was moved**  
5503 **from**  
2302 *method to just before setting sleep enable bit. The execution was moved from*  
5504 *AcpiEnterSleepStatePrep to AcpiEnterSleepState. \_GTS is now executed*  
5505 *immediately before the SLP\_EN bit is set, as per the ACPI specification.*  
5506 *Luming Yu, BZ 1653.*

5508 **Implemented a fix to disable unknown GPEs (2nd version). Now always**  
5509 **disable**  
2307 *Implemented a fix to disable unknown GPEs (2nd version). Now always disable*  
5510 *the GPE, even if ACPICA thinks that that it is already disabled. It is*  
5511 **possible that the AML or some other code has enabled the GPE unbeknownst**  
5512 **to**  
2309 *possible that the AML or some other code has enabled the GPE unbeknownst to*  
5513 *the ACPICA code.*

5515 **Fixed a problem with the Field operator where zero-length fields would**  
5516 **return**  
5517 **an AE\_AML\_NO\_OPERAND exception during table load. Fix enables zero-length**  
5518 **ASL**  
2312 *Fixed a problem with the Field operator where zero-length fields would return*  
2313 *an AE\_AML\_NO\_OPERAND exception during table load. Fix enables zero-length ASL*  
5519 *field declarations in Field(), BankField(), and IndexField(). BZ 10606.*

5521 **Implemented a fix for the Load operator, now load the table at the**  
5522 **namespace**  
5523 **root. This reverts a change introduced in version 20071019. The table is**  
5524 **now**  
2316 *Implemented a fix for the Load operator, now load the table at the namespace*  
2317 *root. This reverts a change introduced in version 20071019. The table is now*  
5525 *loaded at the namespace root even though this goes against the ACPI*  
5526 **specification. This provides compatibility with other ACPI**  
5527 **implementations.**  
5528 **The ACPI specification will be updated to reflect this in ACPI 4.0. Lin**  
5529 **Ming.**  
2319 *specification. This provides compatibility with other ACPI implementations.*  
2320 *The ACPI specification will be updated to reflect this in ACPI 4.0. Lin Ming.*

5531 **Fixed a problem where ACPICA would not Load() tables with unusual**  
5532 **signatures.**  
2322 *Fixed a problem where ACPICA would not Load() tables with unusual signatures.*  
5533 *Now ignore ACPI table signature for Load() operator. Only "SSDT" is*  
5534 *acceptable to the ACPI spec, but tables are seen with OEMx and null sigs.*  
5535 **Therefore, signature validation is worthless. Apparently MS ACPI accepts**  
5536 **such**  
2325 *Therefore, signature validation is worthless. Apparently MS ACPI accepts such*  
5537 *signatures, ACPICA must be compatible. BZ 10454.*

5539 **Fixed a possible negative array index in AcpiUtValidateException. Added**  
5540 **NULL**  
5541 **fields to the exception string arrays to eliminate a -1 subtraction on**  
5542 **the**  
2328 *Fixed a possible negative array index in AcpiUtValidateException. Added NULL*  
2329 *fields to the exception string arrays to eliminate a -1 subtraction on the*  
5543 *SubStatus field.*

5545 Updated the debug tracking macros to reduce overall code and data size.  
5546 Changed ACPI\_MODULE\_NAME and ACPI\_FUNCTION\_NAME to use arrays of strings  
5547 instead of pointers to static strings. Jan Beulich and Bob Moore.

5549 Implemented argument count checking in control method invocation via  
5550 **AcpiEvaluateObject. Now emit an error if too few arguments, warning if**  
5551 **too**  
5552 **many. This applies only to extern programmatic control method execution,**  
5553 **not**  
2337 *AcpiEvaluateObject. Now emit an error if too few arguments, warning if too*  
2338 *many. This applies only to extern programmatic control method execution, not*  
5554 *method-to-method calls within the AML. Lin Ming.*

5556 **Eliminated the ACPI\_NATIVE\_UINT type across all ACPICA code. This type is**  
5557 **no**  
5558 **longer needed, especially with the removal of 16-bit support. It was**  
5559 **replaced**

5560 mostly with UINT32, but also ACPI\_SIZE where a type that changes 32/64  
 5561 bit  
 5562 on  
 2341 Eliminated the ACPI\_NATIVE\_UINT type across all ACPICA code. This type is no  
 2342 longer needed, especially with the removal of 16-bit support. It was replaced  
 2343 mostly with UINT32, but also ACPI\_SIZE where a type that changes 32/64 bit on  
 5563 32/64-bit platforms is required.

5565 Added the C const qualifier for appropriate string constants -- mostly  
 5566 MODULE\_NAME and printf format strings. Jan Beulich.

5568 Example Code and Data Size: These are the sizes for the OS-independent  
 5569 apica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 5570 debug version of the code includes the debug output trace mechanism and  
 5571 has a  
 2351 debug version of the code includes the debug output trace mechanism and has a  
 5572 much larger code and data size.

5574 Previous Release:  
 5575 Non-Debug Version: 80.0K Code, 17.4K Data, 97.4K Total  
 5576 Debug Version: 159.4K Code, 64.4K Data, 223.8K Total  
 5577 Current Release:  
 5578 Non-Debug Version: 79.5K Code, 16.2K Data, 95.7K Total  
 5579 Debug Version: 153.3K Code, 48.3K Data, 201.6K Total

5581 2) iASL Compiler/Disassembler and Tools:

5583 Implemented ACPI table revision ID validation in the disassembler. Zero  
 5584 is  
 5585 always invalid. For DSDTs, the ID controls the interpreter integer width.  
 5586 1  
 2363 Implemented ACPI table revision ID validation in the disassembler. Zero is  
 2364 always invalid. For DSDTs, the ID controls the interpreter integer width. 1  
 5587 means 32-bit and this is unusual. 2 or greater is 64-bit.

5589 -----  
 5590 21 March 2008. Summary of changes for version 20080321:

5592 1) ACPI CA Core Subsystem:

5594 Implemented an additional change to the GPE support in order to suppress  
 5595 spurious or stray GPEs. The AcpiEvDisableGpe function will now  
 5596 permanently  
 5597 disable incoming GPEs that are neither enabled nor disabled -- meaning  
 5598 that  
 5599 the GPE is unknown to the system. This should prevent future interrupt  
 5600 floods  
 2373 spurious or stray GPEs. The AcpiEvDisableGpe function will now permanently  
 2374 disable incoming GPEs that are neither enabled nor disabled -- meaning that  
 2375 the GPE is unknown to the system. This should prevent future interrupt floods  
 5601 from that GPE. BZ 6217 (Zhang Rui)

5603 Fixed a problem where NULL package elements were not returned to the  
 5604 AcpiEvaluateObject interface correctly. The element was simply ignored  
 5605 instead of returning a NULL ACPI\_OBJECT package element, potentially  
 5606 causing  
 5607 a buffer overflow and/or confusing the caller who expected a fixed number  
 5608 of  
 2380 instead of returning a NULL ACPI\_OBJECT package element, potentially causing  
 2381 a buffer overflow and/or confusing the caller who expected a fixed number of  
 5609 elements. BZ 10132 (Lin Ming, Bob Moore)

5611 Fixed a problem with the CreateField, CreateXXXXField (Bit, Byte, Word,  
 5612 Dword,  
 5613 Qword), Field, BankField, and IndexField operators when invoked from  
 5614 inside

5615 an executing control method. In this case, these operators created  
 5616 namespace  
 2384 Fixed a problem with the CreateField, CreateXXXXField (Bit, Byte, Word, Dword,  
 2385 Qword), Field, BankField, and IndexField operators when invoked from inside  
 2386 an executing control method. In this case, these operators created namespace  
 5617 nodes that were incorrectly left marked as permanent nodes instead of  
 5618 temporary nodes. This could cause a problem if there is race condition  
 5619 between an exiting control method and a running namespace walk. (Reported  
 5620 by  
 2389 between an exiting control method and a running namespace walk. (Reported by  
 5621 Linn Crosetto)

5623 Fixed a problem where the CreateField and CreateXXXXField operators would  
 5624 incorrectly allow duplicate names (the name of the field) with no  
 5625 exception  
 2393 incorrectly allow duplicate names (the name of the field) with no exception  
 5626 generated.

5628 Implemented several changes for Notify handling. Added support for new  
 5629 Notify  
 2396 Implemented several changes for Notify handling. Added support for new Notify  
 5630 values (ACPI 2.0+) and improved the Notify debug output. Notify on  
 5631 PowerResource objects is no longer allowed, as per the ACPI  
 5632 specification.  
 2398 PowerResource objects is no longer allowed, as per the ACPI specification.  
 5633 (Bob Moore, Zhang Rui)

5635 All Reference Objects returned via the AcpiEvaluateObject interface are  
 5636 now  
 5637 marked as type "REFERENCE" instead of "ANY". The type ANY is now reserved  
 5638 for  
 5639 NULL objects - either NULL package elements or unresolved named  
 5640 references.  
 2401 All Reference Objects returned via the AcpiEvaluateObject interface are now  
 2402 marked as type "REFERENCE" instead of "ANY". The type ANY is now reserved for  
 2403 NULL objects - either NULL package elements or unresolved named references.

5642 Fixed a problem where an extraneous debug message was produced for  
 5643 package  
 2405 Fixed a problem where an extraneous debug message was produced for package  
 5644 objects (when debugging enabled). The message "Package List length larger  
 5645 than NumElements count" is now produced in the correct case, and is now  
 5646 an  
 2407 than NumElements count" is now produced in the correct case, and is now an  
 5647 error message rather than a debug message. Added a debug message for the  
 5648 opposite case, where NumElements is larger than the Package List (the  
 5649 package  
 2409 opposite case, where NumElements is larger than the Package List (the package  
 5650 will be padded out with NULL elements as per the ACPI spec.)

5652 Implemented several improvements for the output of the ASL "Debug" object  
 5653 to  
 2412 Implemented several improvements for the output of the ASL "Debug" object to  
 5654 clarify and keep all data for a given object on one output line.

5656 Fixed two size calculation issues with the variable-length Start  
 5657 Dependent  
 2415 Fixed two size calculation issues with the variable-length Start Dependent  
 5658 resource descriptor.

5660 Example Code and Data Size: These are the sizes for the OS-independent  
 5661 apica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 5662 debug version of the code includes the debug output trace mechanism and  
 5663 has  
 2420 debug version of the code includes the debug output trace mechanism and has  
 5664 a much larger code and data size.

5666 Previous Release:  
 5667 Non-Debug Version: 79.7K Code, 17.3K Data, 97.0K Total  
 5668 Debug Version: 158.9K Code, 64.0K Data, 222.9K Total  
 5669 Current Release:  
 5670 Non-Debug Version: 80.0K Code, 17.4K Data, 97.4K Total  
 5671 Debug Version: 159.4K Code, 64.4K Data, 223.8K Total

5673 2) iASL Compiler/Disassembler and Tools:

5675 **Fixed a problem with the use of the Switch operator where execution of**  
 5676 **the**  
 2432 *Fixed a problem with the use of the Switch operator where execution of the*  
 5677 *containing method by multiple concurrent threads could cause an*  
 5678 *AE\_ALREADY\_EXISTS exception. This is caused by the fact that there is no*  
 5679 *actual Switch opcode, it must be simulated with local named temporary*  
 5680 **variables and if/else pairs. The solution chosen was to mark any method**  
 5681 **that**  
 5682 **uses Switch as Serialized, thus preventing multiple thread entries. BZ**  
 5683 **469.**  
 2436 *variables and if/else pairs. The solution chosen was to mark any method that*  
 2437 *uses Switch as Serialized, thus preventing multiple thread entries. BZ 469.*

5685 -----  
 5686 13 February 2008. Summary of changes for version 20080213:

5688 1) ACPI CA Core Subsystem:

5690 **Implemented another MS compatibility design change for GPE/Notify**  
 5691 **handling.**  
 5692 **GPEs are now cleared/enabled asynchronously to allow all pending notifies**  
 5693 **to**  
 2444 *Implemented another MS compatibility design change for GPE/Notify handling.*  
 2445 *GPEs are now cleared/enabled asynchronously to allow all pending notifies to*  
 5694 *complete first. It is expected that the OSL will queue the enable request*  
 5695 **behind all pending notify requests (may require changes to the local host**  
 5696 **OSL**  
 2447 *behind all pending notify requests (may require changes to the local host OSL*  
 5697 *in AcpiOsExecute). Alexey Starikovskiy.*

5699 Fixed a problem where buffer and package objects passed as arguments to a  
 5700 **control method via the external AcpiEvaluateObject interface could cause**  
 5701 **an**  
 2451 *control method via the external AcpiEvaluateObject interface could cause an*  
 5702 *AE\_AML\_INTERNAL exception depending on the order and type of operators*  
 5703 *executed by the target control method.*

5705 Fixed a problem where resource descriptor size optimization could cause a  
 5706 **problem when a \_CRS resource template is passed to a \_SRS method. The**  
 5707 **\_SRS**  
 2456 *problem when a \_CRS resource template is passed to a \_SRS method. The \_SRS*  
 5708 *resource template must use the same descriptors (with the same size) as*  
 5709 **returned from \_CRS. This change affects the following resource**  
 5710 **descriptors:**  
 5711 **IRQ / IRQNoFlags and StartDependentFn / StartDependentFnNoPri. (BZ**  
 5712 **9487)**  
 2458 *returned from \_CRS. This change affects the following resource descriptors:*  
 2459 *IRQ / IRQNoFlags and StartDependentFn / StartDependentFnNoPri. (BZ 9487)*

5714 **Fixed a problem where a CopyObject to RegionField, BankField, and**  
 5715 **IndexField**  
 5716 **objects did not perform an implicit conversion as it should. These types**  
 5717 **must**  
 5718 **retain their initial type permanently as per the ACPI specification.**  
 5719 **However,**  
 2461 *Fixed a problem where a CopyObject to RegionField, BankField, and IndexField*

2462 *objects did not perform an implicit conversion as it should. These types must*  
 2463 *retain their initial type permanently as per the ACPI specification. However,*  
 5720 *a CopyObject to all other object types should not perform an implicit*  
 5721 *conversion, as per the ACPI specification. (Lin Ming, Bob Moore) BZ 388*

5723 Fixed a problem with the AcpiGetDevices interface where the mechanism to  
 5724 match device CIDs did not examine the entire list of available CIDs, but  
 5725 instead aborted on the first non-matching CID. Andrew Patterson.

5727 **Fixed a regression introduced in version 20071114. The ACPI\_HIDWORD macro**  
 5728 **was**  
 2471 *Fixed a regression introduced in version 20071114. The ACPI\_HIDWORD macro was*  
 5729 *inadvertently changed to return a 16-bit value instead of a 32-bit value,*  
 5730 *truncating the upper dword of a 64-bit value. This macro is only used to*  
 5731 *display debug output, so no incorrect calculations were made. Also,*  
 5732 *reimplemented the macro so that a 64-bit shift is not performed by*  
 5733 *inefficient compilers.*

5735 Added missing va\_end statements that should correspond with each va\_start  
 5736 statement.

5738 Example Code and Data Size: These are the sizes for the OS-independent  
 5739 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 5740 **debug version of the code includes the debug output trace mechanism and**  
 5741 **has**  
 2483 *debug version of the code includes the debug output trace mechanism and has*  
 5742 *a much larger code and data size.*

5744 Previous Release:  
 5745 Non-Debug Version: 79.5K Code, 17.2K Data, 96.7K Total  
 5746 Debug Version: 159.0K Code, 63.8K Data, 222.8K Total  
 5747 Current Release:  
 5748 Non-Debug Version: 79.7K Code, 17.3K Data, 97.0K Total  
 5749 Debug Version: 158.9K Code, 64.0K Data, 222.9K Total

5751 2) iASL Compiler/Disassembler and Tools:

5753 Implemented full disassembler support for the following new ACPI tables:  
 5754 BERT, EINJ, and ERST. Implemented partial disassembler support for the  
 5755 complicated HEST table. These tables support the Windows Hardware Error  
 5756 Architecture (WHEA).

5758 -----  
 5759 23 January 2008. Summary of changes for version 20080123:

5761 1) ACPI CA Core Subsystem:

5763 Added the 2008 copyright to all module headers and signons. This affects  
 5764 virtually every file in the ACPICA core subsystem, the iASL compiler, and  
 5765 the tools/utilities.

5767 **Fixed a problem with the SizeOf operator when used with Package and**  
 5768 **Buffer**  
 5769 **objects. These objects have deferred execution for some arguments, and**  
 5770 **the**  
 5771 **execution is now completed before the SizeOf is executed. This problem**  
 5772 **caused**  
 5773 **unexpected AE\_PACKAGE\_LIMIT errors on some systems (Lin Ming, Bob Moore)**  
 5774 **BZ**  
 2509 *Fixed a problem with the SizeOf operator when used with Package and Buffer*  
 2510 *objects. These objects have deferred execution for some arguments, and the*  
 2511 *execution is now completed before the SizeOf is executed. This problem caused*  
 2512 *unexpected AE\_PACKAGE\_LIMIT errors on some systems (Lin Ming, Bob Moore) BZ*  
 5775 *9558*

5777 **Implemented an enhancement to the interpreter "slack mode". In the**

5778 **absence**  
 5779 **of**  
 5780 **an explicit return or an implicitly returned object from the last**  
 5781 **executed**  
 5782 **opcode, a control method will now implicitly return an integer of value 0**  
 5783 **for**  
 2515 *Implemented an enhancement to the interpreter "slack mode". In the absence of*  
 2516 *an explicit return or an implicitly returned object from the last executed*  
 2517 *opcode, a control method will now implicitly return an integer of value 0 for*  
 5784 *Microsoft compatibility. (Lin Ming) BZ 392*

5786 **Fixed a problem with the Load operator where an exception was not**  
 5787 **returned**  
 5788 **in**  
 2520 *Fixed a problem with the Load operator where an exception was not returned in*  
 5789 *the case where the table is already loaded. (Lin Ming) BZ 463*

5791 **Implemented support for the use of DDBHandles as an Indexed Reference, as**  
 5792 **per**  
 2523 *Implemented support for the use of DDBHandles as an Indexed Reference, as per*  
 5793 *the ACPI spec. (Lin Ming) BZ 486*

5795 **Implemented support for UserTerm (Method invocation) for the Unload**  
 5796 **operator**  
 2526 *Implemented support for UserTerm (Method invocation) for the Unload operator*  
 5797 *as per the ACPI spec. (Lin Ming) BZ 580*

5799 **Fixed a problem with the LoadTable operator where the OemId and**  
 5800 **OemTableId**  
 5801 **input strings could cause unexpected failures if they were shorter than**  
 5802 **the**  
 2529 *Fixed a problem with the LoadTable operator where the OemId and OemTableId*  
 2530 *input strings could cause unexpected failures if they were shorter than the*  
 5803 *maximum lengths allowed. (Lin Ming, Bob Moore) BZ 576*

5805 **Implemented support for UserTerm (Method invocation) for the Unload**  
 5806 **operator**  
 2533 *Implemented support for UserTerm (Method invocation) for the Unload operator*  
 5807 *as per the ACPI spec. (Lin Ming) BZ 580*

5809 **Implemented header file support for new ACPI tables - BERT, ERST, EINJ,**  
 5810 **HEST,**  
 2536 *Implemented header file support for new ACPI tables - BERT, ERST, EINJ, HEST,*  
 5811 *IBFT, UEFI, WDAT. Disassembler support is forthcoming.*

5813 **Example Code and Data Size: These are the sizes for the OS-independent**  
 5814 **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The**  
 5815 **debug version of the code includes the debug output trace mechanism and**  
 5816 **has**  
 2541 *debug version of the code includes the debug output trace mechanism and has*  
 5817 *a much larger code and data size.*

5819 **Previous Release:**  
 5820 **Non-Debug Version: 79.3K Code, 17.2K Data, 96.5K Total**  
 5821 **Debug Version: 158.6K Code, 63.8K Data, 222.4K Total**  
 5822 **Current Release:**  
 5823 **Non-Debug Version: 79.5K Code, 17.2K Data, 96.7K Total**  
 5824 **Debug Version: 159.0K Code, 63.8K Data, 222.8K Total**

5826 **2) iASL Compiler/Disassembler and Tools:**

5828 **Implemented support in the disassembler for checksum validation on**  
 5829 **incoming**  
 5830 **binary DSDTs and SSDTs. If incorrect, a message is displayed within the**  
 5831 **table**  
 2553 *Implemented support in the disassembler for checksum validation on incoming*

2554 *binary DSDTs and SSDTs. If incorrect, a message is displayed within the table*  
 5832 *header dump at the start of the disassembly.*

5834 **Implemented additional debugging information in the namespace listing**  
 5835 **file**  
 5836 **created during compilation. In addition to the namespace hierarchy, the**  
 5837 **full**  
 2557 *Implemented additional debugging information in the namespace listing file*  
 2558 *created during compilation. In addition to the namespace hierarchy, the full*  
 5838 *pathname to each namespace object is displayed.*

5840 **Fixed a problem with the disassembler where invalid ACPI tables could**  
 5841 **cause**  
 2561 *Fixed a problem with the disassembler where invalid ACPI tables could cause*  
 5842 *faults or infinite loops.*

5844 **Fixed an unexpected parse error when using the optional "parameter types"**  
 5845 **list in a control method declaration. (Lin Ming) BZ 397**

5847 **Fixed a problem where two External declarations with the same name did**  
 5848 **not**  
 2567 *Fixed a problem where two External declarations with the same name did not*  
 5849 *cause an error (Lin Ming) BZ 509*

5851 **Implemented support for full TermArgs (adding Argx, Localx and method**  
 5852 **invocation) for the ParameterData parameter to the LoadTable operator.**  
 5853 **(Lin**  
 2571 *invocation) for the ParameterData parameter to the LoadTable operator. (Lin*  
 5854 *Ming) BZ 583,587*

5856 -----  
 5857 19 December 2007. Summary of changes for version 20071219:

5859 **1) ACPI CA Core Subsystem:**

5861 **Implemented full support for deferred execution for the TermArg string**  
 5862 **arguments for DataTableRegion. This enables forward references and full**  
 5863 **operand resolution for the three string arguments. Similar to**  
 5864 **OperationRegion**  
 2581 *operand resolution for the three string arguments. Similar to OperationRegion*  
 5865 *deferred argument execution.) Lin Ming. BZ 430*

5867 **Implemented full argument resolution support for the BankValue argument**  
 5868 **to**  
 5869 **BankField. Previously, only constants were supported, now any TermArg may**  
 5870 **be**  
 2584 *Implemented full argument resolution support for the BankValue argument to*  
 2585 *BankField. Previously, only constants were supported, now any TermArg may be*  
 5871 *used. Lin Ming BZ 387, 393*

5873 **Fixed a problem with AcpiGetDevices where the search of a branch of the**  
 5874 **device tree could be terminated prematurely. In accordance with the ACPI**  
 5875 **specification, the search down the current branch is terminated if a**  
 5876 **device**  
 5877 **is both not present and not functional (instead of just not present.)**  
 5878 **Yakui**  
 2590 *specification, the search down the current branch is terminated if a device*  
 2591 *is both not present and not functional (instead of just not present.) Yakui*  
 5879 *Zhao.*

5881 **Fixed a problem where "unknown" GPEs could be allowed to fire repeatedly**  
 5882 **if**  
 5883 **the underlying AML code changed the GPE enable registers. Now, any**  
 5884 **unknown**  
 5885 **incoming GPE (no \_Lxx/\_Exx method and not the EC GPE) is immediately**  
 5886 **disabled**

2594 Fixed a problem where "unknown" GPEs could be allowed to fire repeatedly if  
 2595 the underlying AML code changed the GPE enable registers. Now, any unknown  
 2596 incoming GPE (no `_Lxx/_Exx` method and not the EC GPE) is immediately disabled  
 5887 instead of simply ignored. Rui Zhang.

5889 Fixed a problem with Index Fields where the Index register was  
 5890 incorrectly  
 2599 Fixed a problem with Index Fields where the Index register was incorrectly  
 5891 limited to a maximum of 32 bits. Now any size may be used.

5893 Fixed a couple memory leaks associated with "implicit return" objects  
 5894 when  
 2602 Fixed a couple memory leaks associated with "implicit return" objects when  
 5895 the AML Interpreter slack mode is enabled. Lin Ming BZ 349

5897 Example Code and Data Size: These are the sizes for the OS-independent  
 5898 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 5899 debug version of the code includes the debug output trace mechanism and  
 5900 has  
 2607 debug version of the code includes the debug output trace mechanism and has  
 5901 a much larger code and data size.

5903 Previous Release:  
 5904 Non-Debug Version: 79.0K Code, 17.2K Data, 96.2K Total  
 5905 Debug Version: 157.9K Code, 63.6K Data, 221.5K Total  
 5906 Current Release:  
 5907 Non-Debug Version: 79.3K Code, 17.2K Data, 96.5K Total  
 5908 Debug Version: 158.6K Code, 63.8K Data, 222.4K Total

5910 -----  
 5911 14 November 2007. Summary of changes for version 20071114:

5913 1) ACPI CA Core Subsystem:

5915 Implemented event counters for each of the Fixed Events, the ACPI SCI  
 5916 (interrupt) itself, and control methods executed. Named  
 5917 `AcpiFixedEventCount[]`, `AcpiSciCount`, and `AcpiMethodCount` respectively.  
 5918 These  
 2624 `AcpiFixedEventCount[]`, `AcpiSciCount`, and `AcpiMethodCount` respectively. These  
 5919 should be useful for debugging and statistics.

5921 Implemented a new external interface, `AcpiGetStatistics`, to retrieve the  
 5922 contents of the various event counters. Returns the current values for  
 5923 `AcpiSciCount`, `AcpiGpeCount`, the `AcpiFixedEventCount` array, and  
 5924 `AcpiMethodCount`. The interface can be expanded in the future if new  
 5925 counters  
 5926 are added. Device drivers should use this interface rather than access  
 5927 the  
 2630 `AcpiMethodCount`. The interface can be expanded in the future if new counters  
 2631 are added. Device drivers should use this interface rather than access the  
 5928 counters directly.

5930 Fixed a problem with the `FromBCD` and `ToBCD` operators. With some  
 5931 compilers,  
 5932 the `ShortDivide` function worked incorrectly, causing problems with the  
 5933 BCD  
 2634 Fixed a problem with the `FromBCD` and `ToBCD` operators. With some compilers,  
 2635 the `ShortDivide` function worked incorrectly, causing problems with the BCD  
 5934 functions with large input values. A truncation from 64-bit to 32-bit  
 5935 inadvertently occurred. Internal BZ 435. Lin Ming

5937 Fixed a problem with Index references passed as method arguments.  
 5938 References  
 5939 passed as arguments to control methods were dereferenced immediately  
 5940 (before  
 5941 control was passed to the called method). The references are now

5942 correctly  
 2639 Fixed a problem with Index references passed as method arguments. References  
 2640 passed as arguments to control methods were dereferenced immediately (before  
 2641 control was passed to the called method). The references are now correctly  
 5943 passed directly to the called method. BZ 5389. Lin Ming

5945 Fixed a problem with `CopyObject` used in conjunction with the Index  
 5946 operator.  
 5947 The reference was incorrectly dereferenced before the copy. The reference  
 5948 is  
 2644 Fixed a problem with `CopyObject` used in conjunction with the Index operator.  
 2645 The reference was incorrectly dereferenced before the copy. The reference is  
 5949 now correctly copied. BZ 5391. Lin Ming

5951 Fixed a problem with Control Method references within Package objects.  
 5952 These  
 2648 Fixed a problem with Control Method references within Package objects. These  
 5953 references are now correctly generated. This completes the package  
 5954 construction overhaul that began in version 20071019.

5956 Example Code and Data Size: These are the sizes for the OS-independent  
 5957 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 5958 debug version of the code includes the debug output trace mechanism and  
 5959 has  
 2654 debug version of the code includes the debug output trace mechanism and has  
 5960 a much larger code and data size.

5962 Previous Release:  
 5963 Non-Debug Version: 78.8K Code, 17.2K Data, 96.0K Total  
 5964 Debug Version: 157.2K Code, 63.4K Data, 220.6K Total  
 5965 Current Release:  
 5966 Non-Debug Version: 79.0K Code, 17.2K Data, 96.2K Total  
 5967 Debug Version: 157.9K Code, 63.6K Data, 221.5K Total

5970 2) iASL Compiler/Disassembler and Tools:

5972 The `AcpiExec` utility now installs handlers for all of the predefined  
 5973 Operation Region types. New types supported are: `PCI_Config`, `CMOS`, and  
 5974 `PCIBARTarget`.

5976 Fixed a problem with the 64-bit version of `AcpiExec` where the extended  
 5977 (64-  
 2671 Fixed a problem with the 64-bit version of `AcpiExec` where the extended (64-  
 5978 bit) address fields for the DSDT and FACS within the FADT were not being  
 5979 used, causing truncation of the upper 32-bits of these addresses. Lin  
 5980 Ming  
 2673 used, causing truncation of the upper 32-bits of these addresses. Lin Ming  
 5981 and Bob Moore

5983 -----  
 5984 19 October 2007. Summary of changes for version 20071019:

5986 1) ACPI CA Core Subsystem:

5988 Fixed a problem with the Alias operator when the target of the alias is a  
 5989 named ASL operator that opens a new scope -- `Scope, Device`,  
 5990 `PowerResource`,  
 2682 named ASL operator that opens a new scope -- `Scope, Device, PowerResource`,  
 5991 Processor, and ThermalZone. In these cases, any children of the original  
 5992 operator could not be accessed via the alias, potentially causing  
 5993 unexpected  
 2684 operator could not be accessed via the alias, potentially causing unexpected  
 5994 `AE_NOT_FOUND` exceptions. (BZ 9067)

5996 Fixed a problem with the Package operator where all named references were

5997 **created as object references and left otherwise unresolved. According to**  
5998 **the**  
5999 **ACPI specification, a Package can only contain Data Objects or references**  
6000 **to**  
2688 *created as object references and left otherwise unresolved. According to the*  
2689 *ACPI specification, a Package can only contain Data Objects or references to*  
6001 *control methods. The implication is that named references to Data Objects*  
6002 *(Integer, Buffer, String, Package, BufferField, Field) should be resolved*  
6003 *immediately upon package creation. This is the approach taken with this*  
6004 *change. References to all other named objects (Methods, Devices, Scopes,*  
6005 *etc.) are all now properly created as reference objects. (BZ 5328)*

6007 Reverted a change to Notify handling that was introduced in version  
6008 20070508. This version changed the Notify handling from asynchronous to  
6009 **fully synchronous (Device driver Notify handling with respect to the**  
6010 **Notify**  
2698 *fully synchronous (Device driver Notify handling with respect to the Notify*  
6011 *ASL operator). It was found that this change caused more problems than it*  
6012 *solved and was removed by most users.*

6014 **Fixed a problem with the Increment and Decrement operators where the type**  
6015 **of**  
2702 *Fixed a problem with the Increment and Decrement operators where the type of*  
6016 *the target object could be unexpectedly and incorrectly changed. (BZ 353)*  
6017 *Lin Ming.*

6019 Fixed a problem with the Load and LoadTable operators where the table  
6020 location within the namespace was ignored. Instead, the table was always  
6021 loaded into the root or current scope. Lin Ming.

6023 Fixed a problem with the Load operator when loading a table from a buffer  
6024 object. The input buffer was prematurely zeroed and/or deleted. (BZ 577)

6026 **Fixed a problem with the Debug object where a store of a DdbHandle**  
6027 **reference**  
2713 *Fixed a problem with the Debug object where a store of a DdbHandle reference*  
6028 *object to the Debug object could cause a fault.*

6030 **Added a table checksum verification for the Load operator, in the case**  
6031 **where**  
2716 *Added a table checksum verification for the Load operator, in the case where*  
6032 *the load is from a buffer. (BZ 578).*

6034 **Implemented additional parameter validation for the LoadTable operator.**  
6035 **The**  
6036 **length of the input strings SignatureString, OemIdString, and OemTableId**  
6037 **are**  
2719 *Implemented additional parameter validation for the LoadTable operator. The*  
2720 *length of the input strings SignatureString, OemIdString, and OemTableId are*  
6038 *now checked for maximum lengths. (BZ 582) Lin Ming.*

6040 Example Code and Data Size: These are the sizes for the OS-independent  
6041 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
6042 **debug version of the code includes the debug output trace mechanism and**  
6043 **has**  
2725 *debug version of the code includes the debug output trace mechanism and has*  
6044 *a much larger code and data size.*

6046 Previous Release:  
6047 Non-Debug Version: 78.5K Code, 17.1K Data, 95.6K Total  
6048 Debug Version: 156.7K Code, 63.2K Data, 219.9K Total  
6049 Current Release:  
6050 Non-Debug Version: 78.8K Code, 17.2K Data, 96.0K Total  
6051 Debug Version: 157.2K Code, 63.4K Data, 220.6K Total

6054 2) iASL Compiler/Disassembler:

6056 Fixed a problem where if a single file was specified and the file did not  
6057 exist, no error message was emitted. (Introduced with wildcard support in  
6058 version 20070917.)

6060 -----  
6061 19 September 2007. Summary of changes for version 20070919:

6063 1) ACPI CA Core Subsystem:

6065 Designed and implemented new external interfaces to install and remove  
6066 **handlers for ACPI table-related events. Current events that are defined**  
6067 **are**  
2748 *handlers for ACPI table-related events. Current events that are defined are*  
6068 *LOAD and UNLOAD. These interfaces allow the host to track ACPI tables as*  
6069 *they are dynamically loaded and unloaded. See AcpiInstallTableHandler and*  
6070 *AcpiRemoveTableHandler. (Lin Ming and Bob Moore)*

6072 Fixed a problem where the use of the AcpiGbl\_AllMethodsSerialized flag  
6073 (acpi\_serialized option on Linux) could cause some systems to hang during  
6074 initialization. (Bob Moore) BZ 8171

6076 Fixed a problem where objects of certain types (Device, ThermalZone,  
6077 Processor, PowerResource) can be not found if they are declared and  
6078 referenced from within the same control method (Lin Ming) BZ 341

6080 Example Code and Data Size: These are the sizes for the OS-independent  
6081 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
6082 **debug version of the code includes the debug output trace mechanism and**  
6083 **has**  
2763 *debug version of the code includes the debug output trace mechanism and has*  
6084 *a much larger code and data size.*

6086 Previous Release:  
6087 Non-Debug Version: 78.3K Code, 17.0K Data, 95.3K Total  
6088 Debug Version: 156.3K Code, 63.1K Data, 219.4K Total  
6089 Current Release:  
6090 Non-Debug Version: 78.5K Code, 17.1K Data, 95.6K Total  
6091 Debug Version: 156.7K Code, 63.2K Data, 219.9K Total

6094 2) iASL Compiler/Disassembler:

6096 **Implemented support to allow multiple files to be compiled/disassembled**  
6097 **in**  
6098 **a**  
6099 **single invocation. This includes command line wildcard support for both**  
6100 **the**  
2776 *Implemented support to allow multiple files to be compiled/disassembled in a*  
2777 *single invocation. This includes command line wildcard support for both the*  
6101 *Windows and Unix versions of the compiler. This feature simplifies the*  
6102 **disassembly and compilation of multiple ACPI tables in a single**  
6103 **directory.**  
2779 *disassembly and compilation of multiple ACPI tables in a single directory.*

6105 -----  
6106 08 May 2007. Summary of changes for version 20070508:

6108 1) ACPI CA Core Subsystem:

6110 **Implemented a Microsoft compatibility design change for the handling of**  
6111 **the**  
2786 *Implemented a Microsoft compatibility design change for the handling of the*  
6112 *Notify AML operator. Previously, notify handlers were dispatched and*  
6113 *executed completely asynchronously in a deferred thread. The new design*



6114 still executes the notify handlers in a different thread, but the  
 6115 original  
 6116 thread that executed the Notify() now waits at a synchronization point  
 6117 for  
 6118 the notify handler to complete. Some machines depend on a synchronous  
 6119 Notify  
 6120 still executes the notify handlers in a different thread, but the original  
 6121 thread that executed the Notify() now waits at a synchronization point for  
 6122 the notify handler to complete. Some machines depend on a synchronous Notify  
 6123 operator in order to operate correctly.

6122 Implemented support to allow Package objects to be passed as method  
 6123 arguments to the external AcpiEvaluateObject interface. Previously, this  
 6124 would return the AE\_NOT\_IMPLEMENTED exception. This feature had not been  
 6125 implemented since there were no reserved control methods that required it  
 6126 until recently.

6128 Fixed a problem with the internal FADT conversion where ACPI 1.0 FADTs  
 6129 that  
 6130 Fixed a problem with the internal FADT conversion where ACPI 1.0 FADTs that  
 6131 contained invalid non-zero values in reserved fields could cause later  
 6132 failures because these fields have meaning in later revisions of the  
 6133 FADT.  
 6134 For incoming ACPI 1.0 FADTs, these fields are now always zeroed. (The  
 6135 fields  
 6136 failures because these fields have meaning in later revisions of the FADT.  
 6137 For incoming ACPI 1.0 FADTs, these fields are now always zeroed. (The fields  
 6138 are: Preferred\_PM\_Profile, PSTATE\_CNT, CST\_CNT, and IAPC\_BOOT\_FLAGS.)

6137 Fixed a problem where the Global Lock handle was not properly updated if  
 6138 a  
 6139 thread that acquired the Global Lock via executing AML code then  
 6140 attempted  
 6141 to acquire the lock via the AcpiAcquireGlobalLock interface. Reported by  
 6142 Joe  
 6143 Fixed a problem where the Global Lock handle was not properly updated if a  
 6144 thread that acquired the Global Lock via executing AML code then attempted  
 6145 to acquire the lock via the AcpiAcquireGlobalLock interface. Reported by Joe  
 6146 Liu.

6145 Fixed a problem in AcpiEvDeleteGpeXrupt where the global interrupt list  
 6146 could be corrupted if the interrupt being removed was at the head of the  
 6147 list. Reported by Linn Crosetto.

6149 Example Code and Data Size: These are the sizes for the OS-independent  
 6150 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 6151 debug version of the code includes the debug output trace mechanism and  
 6152 has  
 6153 debug version of the code includes the debug output trace mechanism and has  
 6154 a much larger code and data size.

6155 Previous Release:  
 6156 Non-Debug Version: 78.0K Code, 17.1K Data, 95.1K Total  
 6157 Debug Version: 155.9K Code, 63.1K Data, 219.0K Total  
 6158 Current Release:  
 6159 Non-Debug Version: 78.3K Code, 17.0K Data, 95.3K Total  
 6160 Debug Version: 156.3K Code, 63.1K Data, 219.4K Total

6162 -----  
 6163 20 March 2007. Summary of changes for version 20070320:

6165 1) ACPI CA Core Subsystem:

6167 Implemented a change to the order of interpretation and evaluation of AML  
 6168 operand objects within the AML interpreter. The interpreter now evaluates  
 6169 operands in the order that they appear in the AML stream (and the

6170 corresponding ASL code), instead of in the reverse order (after the  
 6171 entire  
 6172 operand list has been parsed). The previous behavior caused several  
 6173 subtle  
 6174 corresponding ASL code), instead of in the reverse order (after the entire  
 6175 operand list has been parsed). The previous behavior caused several subtle  
 6176 incompatibilities with the Microsoft AML interpreter as well as being  
 6177 somewhat non-intuitive. BZ 7871, local BZ 263. Valery Podrezov.

6177 Implemented a change to the ACPI Global Lock support. All interfaces to  
 6178 the  
 6179 Implemented a change to the ACPI Global Lock support. All interfaces to the  
 6180 global lock now allow the same thread to acquire the lock multiple times.  
 6181 This affects the AcpiAcquireGlobalLock external interface to the global  
 6182 lock  
 6183 This affects the AcpiAcquireGlobalLock external interface to the global lock  
 6184 as well as the internal use of the global lock to support AML fields -- a  
 6185 control method that is holding the global lock can now simultaneously  
 6186 access  
 6187 AML fields that require global lock protection. Previously, in both  
 6188 cases,  
 6189 this would have resulted in an AE\_ALREADY\_ACQUIRED exception. The change  
 6190 to  
 6191 control method that is holding the global lock can now simultaneously access  
 6192 AML fields that require global lock protection. Previously, in both cases,  
 6193 this would have resulted in an AE\_ALREADY\_ACQUIRED exception. The change to  
 6194 AcpiAcquireGlobalLock is of special interest to drivers for the Embedded  
 6195 Controller. There is no change to the behavior of the AML Acquire  
 6196 operator,  
 6197 Controller. There is no change to the behavior of the AML Acquire operator,  
 6198 as this can already be used to acquire a mutex multiple times by the same  
 6199 thread. BZ 8066. With assistance from Alexey Starikovskiy.

6195 Fixed a problem where invalid objects could be referenced in the AML  
 6196 Interpreter after error conditions. During operand evaluation, ensure  
 6197 that  
 6198 Interpreter after error conditions. During operand evaluation, ensure that  
 6199 the internal "Return Object" field is cleared on error and only valid  
 6200 pointers are stored there. Caused occasional access to deleted objects  
 6201 that  
 6202 pointers are stored there. Caused occasional access to deleted objects that  
 6203 resulted in "large reference count" warning messages. Valery Podrezov.

6203 Fixed a problem where an AE\_STACK\_OVERFLOW internal exception could occur  
 6204 on  
 6205 Fixed a problem where an AE\_STACK\_OVERFLOW internal exception could occur on  
 6206 deeply nested control method invocations. BZ 7873, local BZ 487. Valery  
 6207 Podrezov.

6208 Fixed an internal problem with the handling of result objects on the  
 6209 interpreter result stack. BZ 7872. Valery Podrezov.

6211 Removed obsolete code that handled the case where AML\_NAME\_OP is the  
 6212 target  
 6213 Removed obsolete code that handled the case where AML\_NAME\_OP is the target  
 6214 of a reference (Reference.Opcode). This code was no longer necessary. BZ  
 6215 7874. Valery Podrezov.

6216 Removed obsolete ACPI\_NO\_INTEGER64\_SUPPORT from two header files. This  
 6217 was  
 6218 a  
 6219 Removed obsolete ACPI\_NO\_INTEGER64\_SUPPORT from two header files. This was a  
 6220 remnant from the previously discontinued 16-bit support.

6221 Example Code and Data Size: These are the sizes for the OS-independent  
 6222 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The

6223 **debug version of the code includes the debug output trace mechanism and**  
 6224 **has**  
 2874 *debug version of the code includes the debug output trace mechanism and has*  
 6225 a much larger code and data size.

6227 Previous Release:  
 6228 Non-Debug Version: 78.0K Code, 17.1K Data, 95.1K Total  
 6229 Debug Version: 155.8K Code, 63.3K Data, 219.1K Total  
 6230 Current Release:  
 6231 Non-Debug Version: 78.0K Code, 17.1K Data, 95.1K Total  
 6232 Debug Version: 155.9K Code, 63.1K Data, 219.0K Total

6234 -----  
 6235 26 January 2007. Summary of changes for version 20070126:

6237 1) ACPI CA Core Subsystem:

6239 Added the 2007 copyright to all module headers and signons. This affects  
 6240 virtually every file in the ACPICA core subsystem, the iASL compiler, and  
 6241 the utilities.

6243 Implemented a fix for an incorrect parameter passed to AcpiTbDeleteTable  
 6244 **during a table load. A bad pointer was passed in the case where the DSDT**  
 6245 **is**  
 2894 *during a table load. A bad pointer was passed in the case where the DSDT is*  
 6246 overridden, causing a fault in this case.

6248 Example Code and Data Size: These are the sizes for the OS-independent  
 6249 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 6250 **debug version of the code includes the debug output trace mechanism and**  
 6251 **has**  
 2899 *debug version of the code includes the debug output trace mechanism and has*  
 6252 a much larger code and data size.

6254 Previous Release:  
 6255 Non-Debug Version: 78.0K Code, 17.1K Data, 95.1K Total  
 6256 Debug Version: 155.8K Code, 63.3K Data, 219.1K Total  
 6257 Current Release:  
 6258 Non-Debug Version: 78.0K Code, 17.1K Data, 95.1K Total  
 6259 Debug Version: 155.8K Code, 63.3K Data, 219.1K Total

6261 -----  
 6262 15 December 2006. Summary of changes for version 20061215:

6264 1) ACPI CA Core Subsystem:

6266 **Support for 16-bit ACPICA has been completely removed since it is no**  
 6267 **longer**  
 2914 *Support for 16-bit ACPICA has been completely removed since it is no longer*  
 6268 necessary and it clutters the code. All 16-bit macros, types, and  
 6269 **conditional compiles have been removed, cleaning up and simplifying the**  
 6270 **code**  
 2916 *conditional compiles have been removed, cleaning up and simplifying the code*  
 6271 across the entire subsystem. DOS support is no longer needed since the  
 6272 bootable Linux firmware kit is now available.

6274 The handler for the Global Lock is now removed during AcpiTerminate to  
 6275 enable a clean subsystem restart, via the implementation of the  
 6276 AcpiEvRemoveGlobalLockHandler function. (With assistance from Joel Bretz,  
 6277 HP)

6279 **Implemented enhancements to the multithreading support within the**  
 6280 **debugger**  
 6281 **to enable improved multithreading debugging and evaluation of the**  
 6282 **subsystem.**  
 2925 *Implemented enhancements to the multithreading support within the debugger*

2926 *to enable improved multithreading debugging and evaluation of the subsystem.*  
 6283 (Valery Podrezov)

6285 **Debugger: Enhanced the Statistics/Memory command to emit the total**  
 6286 **(maximum)**  
 6287 **memory used during the execution, as well as the maximum memory consumed**  
 6288 **by**  
 2929 *Debugger: Enhanced the Statistics/Memory command to emit the total (maximum)*  
 2930 *memory used during the execution, as well as the maximum memory consumed by*  
 6289 each of the various object types. (Valery Podrezov)

6291 Example Code and Data Size: These are the sizes for the OS-independent  
 6292 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 6293 **debug version of the code includes the debug output trace mechanism and**  
 6294 **has**  
 2935 *debug version of the code includes the debug output trace mechanism and has*  
 6295 a much larger code and data size.

6297 Previous Release:  
 6298 Non-Debug Version: 77.9K Code, 17.0K Data, 94.9K Total  
 6299 Debug Version: 155.2K Code, 63.1K Data, 218.3K Total  
 6300 Current Release:  
 6301 Non-Debug Version: 78.0K Code, 17.1K Data, 95.1K Total  
 6302 Debug Version: 155.8K Code, 63.3K Data, 219.1K Total

6305 2) iASL Compiler/Disassembler and Tools:

6307 AcpiExec: Implemented a new option (-m) to display full memory use  
 6308 statistics upon subsystem/program termination. (Valery Podrezov)

6310 -----  
 6311 09 November 2006. Summary of changes for version 20061109:

6313 1) ACPI CA Core Subsystem:

6315 **Optimized the Load ASL operator in the case where the source operand is**  
 6316 **an**  
 2956 *Optimized the Load ASL operator in the case where the source operand is an*  
 6317 operation region. Simply map the operation region memory, instead of  
 6318 performing a bitwise read. (Region must be of type SystemMemory, see  
 6319 below.)

6321 Fixed the Load ASL operator for the case where the source operand is a  
 6322 **region field. A buffer object is also allowed as the source operand. BZ**  
 6323 **480**  
 2962 *region field. A buffer object is also allowed as the source operand. BZ 480*

6325 **Fixed a problem where the Load ASL operator allowed the source operand to**  
 6326 **be**  
 2964 *Fixed a problem where the Load ASL operator allowed the source operand to be*  
 6327 an operation region of any type. It is now restricted to regions of type  
 6328 SystemMemory, as per the ACPI specification. BZ 481

6330 Additional cleanup and optimizations for the new Table Manager code.

6332 **AcpiEnable will now fail if all of the required ACPI tables are not**  
 6333 **loaded**  
 2970 *AcpiEnable will now fail if all of the required ACPI tables are not loaded*  
 6334 (FADT, FACS, DSDT). BZ 477

6336 **Added #pragma pack(8/4) to aobject.h to ensure that the structures in**  
 6337 **this**  
 2973 *Added #pragma pack(8/4) to aobject.h to ensure that the structures in this*  
 6338 header are always compiled as aligned. The ACPI\_OPERAND\_OBJECT has been  
 6339 manually optimized to be aligned and will not work if it is byte-packed.

6341 Example Code and Data Size: These are the sizes for the OS-independent  
 6342 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 6343 **debug version of the code includes the debug output trace mechanism and**  
 6344 **has**  
 2979 *debug version of the code includes the debug output trace mechanism and has*  
 6345 a much larger code and data size.

6347 Previous Release:  
 6348 Non-Debug Version: 78.1K Code, 17.1K Data, 95.2K Total  
 6349 Debug Version: 155.4K Code, 63.1K Data, 218.5K Total  
 6350 Current Release:  
 6351 Non-Debug Version: 77.9K Code, 17.0K Data, 94.9K Total  
 6352 Debug Version: 155.2K Code, 63.1K Data, 218.3K Total

6355 2) iASL Compiler/Disassembler and Tools:

6357 Fixed a problem where the presence of the \_OSI predefined control method  
 6358 within complex expressions could cause an internal compiler error.

6360 AcpiExec: Implemented full region support for multiple address spaces.  
 6361 SpaceId is now part of the REGION object. BZ 429

6363 -----  
 6364 11 October 2006. Summary of changes for version 20061011:

6366 1) ACPI CA Core Subsystem:

6368 Completed an AML interpreter performance enhancement for control method  
 6369 execution. Previously a 2-pass parse/execution, control methods are now  
 6370 completely parsed and executed in a single pass. This improves overall  
 6371 interpreter performance by ~25%, reduces code size, and reduces CPU stack  
 6372 use. (Valery Podrezov + interpreter changes in version 20051202 that  
 6373 eliminated namespace loading during the pass one parse.)

6375 **Implemented \_CID support for PCI Root Bridge detection. If the \_HID does**  
 6376 **not**  
 6377 **match the predefined PCI Root Bridge IDs, the \_CID list (if present) is**  
 6378 **now**  
 3010 *Implemented \_CID support for PCI Root Bridge detection. If the \_HID does not*  
 3011 *match the predefined PCI Root Bridge IDs, the \_CID list (if present) is now*  
 6379 obtained and also checked for an ID match.

6381 **Implemented additional support for the PCI \_ADR execution: upsearch until**  
 6382 **a**  
 3014 *Implemented additional support for the PCI \_ADR execution: upsearch until a*  
 6383 *device scope is found before executing \_ADR. This allows PCI\_Config*  
 6384 **operation regions to be declared locally within control methods**  
 6385 **underneath**  
 3016 *operation regions to be declared locally within control methods underneath*  
 6386 PCI device objects.

6388 Fixed a problem with a possible race condition between threads executing  
 6389 AcpiWalkNamespace and the AML interpreter. This condition was removed by  
 6390 **modifying AcpiWalkNamespace to (by default) ignore all temporary**  
 6391 **namespace**  
 3021 *modifying AcpiWalkNamespace to (by default) ignore all temporary namespace*  
 6392 *entries created during any concurrent control method execution. An*  
 6393 *additional namespace race condition is known to exist between*  
 6394 *AcpiWalkNamespace and the Load/Unload ASL operators and is still under*  
 6395 *investigation.*

6397 Restructured the AML ParseLoop function, breaking it into several  
 6398 **subfunctions in order to reduce CPU stack use and improve**  
 6399 **maintainability.**

3028 *subfunctions in order to reduce CPU stack use and improve maintainability.*  
 6400 (Mikhail Kouzmich)

6402 **AcpiGetHandle: Fix for parameter validation to detect invalid**  
 6403 **combinations**  
 3031 *AcpiGetHandle: Fix for parameter validation to detect invalid combinations*  
 6404 of prefix handle and pathname. BZ 478

6406 Example Code and Data Size: These are the sizes for the OS-independent  
 6407 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 6408 **debug version of the code includes the debug output trace mechanism and**  
 6409 **has**  
 3036 *debug version of the code includes the debug output trace mechanism and has*  
 6410 a much larger code and data size.

6412 Previous Release:  
 6413 Non-Debug Version: 77.9K Code, 17.1K Data, 95.0K Total  
 6414 Debug Version: 154.6K Code, 63.0K Data, 217.6K Total  
 6415 Current Release:  
 6416 Non-Debug Version: 78.1K Code, 17.1K Data, 95.2K Total  
 6417 Debug Version: 155.4K Code, 63.1K Data, 218.5K Total

6419 2) iASL Compiler/Disassembler and Tools:

6421 **Ported the -g option (get local ACPI tables) to the new ACPICA Table**  
 6422 **Manager**  
 3048 *Ported the -g option (get local ACPI tables) to the new ACPICA Table Manager*  
 6423 to restore original behavior.

6425 -----  
 6426 27 September 2006. Summary of changes for version 20060927:

6428 1) ACPI CA Core Subsystem:

6430 Removed the "Flags" parameter from AcpiGetRegister and AcpiSetRegister.  
 6431 These functions now use a spinlock for mutual exclusion and the interrupt  
 6432 level indication flag is not needed.

6434 Fixed a problem with the Global Lock where the lock could appear to be  
 6435 obtained before it is actually obtained. The global lock semaphore was  
 6436 **inadvertently created with one unit instead of zero units. (BZ 464)**  
 6437 **Fiodor**  
 3062 *inadvertently created with one unit instead of zero units. (BZ 464) Fiodor*  
 6438 Suietov.

6440 **Fixed a possible memory leak and fault in AcpiExResolveObjectToValue**  
 6441 **during**  
 3065 *Fixed a possible memory leak and fault in AcpiExResolveObjectToValue during*  
 6442 a read from a buffer or region field. (BZ 458) Fiodor Suietov.

6444 Example Code and Data Size: These are the sizes for the OS-independent  
 6445 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 6446 **debug version of the code includes the debug output trace mechanism and**  
 6447 **has**  
 3070 *debug version of the code includes the debug output trace mechanism and has*  
 6448 a much larger code and data size.

6450 Previous Release:  
 6451 Non-Debug Version: 77.9K Code, 17.1K Data, 95.0K Total  
 6452 Debug Version: 154.7K Code, 63.0K Data, 217.7K Total  
 6453 Current Release:  
 6454 Non-Debug Version: 77.9K Code, 17.1K Data, 95.0K Total  
 6455 Debug Version: 154.6K Code, 63.0K Data, 217.6K Total

6458 2) iASL Compiler/Disassembler and Tools:

6460 **Fixed a compilation problem with the pre-defined Resource Descriptor**  
 6461 **field**  
 6462 **names where an "object does not exist" error could be incorrectly**  
 6463 **generated**  
 3083 *Fixed a compilation problem with the pre-defined Resource Descriptor field*  
 3084 *names where an "object does not exist" error could be incorrectly generated*  
 6464 *if the parent ResourceTemplate pathname places the template within a*  
 6465 *different namespace scope than the current scope. (BZ 7212)*

6467 **Fixed a problem where the compiler could hang after syntax errors**  
 6468 **detected**  
 3088 *Fixed a problem where the compiler could hang after syntax errors detected*  
 6469 *in an Elseif construct. (BZ 453)*

6471 *Fixed a problem with the AmlFilename parameter to the DefinitionBlock()*  
 6472 **operator. An incorrect output filename was produced when this parameter**  
 6473 **was**  
 3092 *operator. An incorrect output filename was produced when this parameter was*  
 6474 *a null string (""). Now, the original input filename is used as the AML*  
 6475 *output filename, with an ".aml" extension.*

6477 **Implemented a generic batch command mode for the AcpiExec utility**  
 6478 **(execute**  
 3096 *Implemented a generic batch command mode for the AcpiExec utility (execute*  
 6479 *any AML debugger command) (Valery Podrezov).*

6481 -----  
 6482 12 September 2006. Summary of changes for version 20060912:

6484 1) ACPI CA Core Subsystem:

6486 *Enhanced the implementation of the "serialized mode" of the interpreter*  
 6487 *(enabled via the AcpiGbl\_AllMethodsSerialized flag.) When this mode is*  
 6488 **specified, instead of creating a serialization semaphore per control**  
 6489 **method,**  
 3106 *specified, instead of creating a serialization semaphore per control method,*  
 6490 *the interpreter lock is simply no longer released before a blocking*  
 6491 *operation during control method execution. This effectively makes the AML*  
 6492 *Interpreter single-threaded. The overhead of a semaphore per-method is*  
 6493 *eliminated.*

6495 **Fixed a regression where an error was no longer emitted if a control**  
 6496 **method**  
 3112 *Fixed a regression where an error was no longer emitted if a control method*  
 6497 *attempts to create 2 objects of the same name. This once again returns*  
 6498 **AE\_ALREADY\_EXISTS. When this exception occurs, it invokes the mechanism**  
 6499 **that**  
 3114 *AE\_ALREADY\_EXISTS. When this exception occurs, it invokes the mechanism that*  
 6500 *will dynamically serialize the control method to possible prevent future*  
 6501 *errors. (BZ 440)*

6503 *Integrated a fix for a problem with PCI Express HID detection in the PCI*  
 6504 *Config Space setup procedure. (BZ 7145)*

6506 *Moved all FADT-related functions to a new file, tbfdt.c. Eliminated the*  
 6507 *AcpiHwInitialize function - the FADT registers are now validated when the*  
 6508 *table is loaded.*

6510 **Added two new warnings during FADT verification - 1) if the FADT is**  
 6511 **larger**  
 6512 **than the largest known FADT version, and 2) if there is a mismatch**  
 6513 **between**  
 6514 **a**  
 6515 **32-bit block address and the 64-bit X counterpart (when both are non-**  
 6516 **zero.)**

3125 *Added two new warnings during FADT verification - 1) if the FADT is larger*  
 3126 *than the largest known FADT version, and 2) if there is a mismatch between a*  
 3127 *32-bit block address and the 64-bit X counterpart (when both are non-zero.)*

6518 *Example Code and Data Size: These are the sizes for the OS-independent*  
 6519 *acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The*  
 6520 **debug version of the code includes the debug output trace mechanism and**  
 6521 **has**  
 3131 *debug version of the code includes the debug output trace mechanism and has*  
 6522 *a much larger code and data size.*

6524 Previous Release:  
 6525 Non-Debug Version: 77.9K Code, 16.7K Data, 94.6K Total  
 6526 Debug Version: 154.9K Code, 62.6K Data, 217.5K Total  
 6527 Current Release:  
 6528 Non-Debug Version: 77.9K Code, 17.1K Data, 95.0K Total  
 6529 Debug Version: 154.7K Code, 63.0K Data, 217.7K Total

6532 2) iASL Compiler/Disassembler and Tools:

6534 **Fixed a problem with the implementation of the Switch() operator where**  
 6535 **the**  
 6536 **temporary variable was declared too close to the actual Switch, instead**  
 6537 **of**  
 3144 *Fixed a problem with the implementation of the Switch() operator where the*  
 3145 *temporary variable was declared too close to the actual Switch, instead of*  
 6538 *at method level. This could cause a problem if the Switch() operator is*  
 6539 *within a while loop, causing an error on the second iteration. (BZ 460)*

6541 *Disassembler - fix for error emitted for unknown type for target of scope*  
 6542 *operator. Now, ignore it and continue.*

6544 *Disassembly of an FADT now verifies the input FADT and reports any errors*  
 6545 *found. Fix for proper disassembly of full-sized (ACPI 2.0) FADTs.*

6547 **Disassembly of raw data buffers with byte initialization data now**  
 6548 **prefixes**  
 3155 *Disassembly of raw data buffers with byte initialization data now prefixes*  
 6549 *each output line with the current buffer offset.*

6551 *Disassembly of ASF! table now includes all variable-length data fields at*  
 6552 *the end of some of the subtables.*

6554 *The disassembler now emits a comment if a buffer appears to be a*  
 6555 **ResourceTemplate, but cannot be disassembled as such because the EndTag**  
 6556 **does**  
 3162 *ResourceTemplate, but cannot be disassembled as such because the EndTag does*  
 6557 *not appear at the very end of the buffer.*

6559 **AcpiExec - Added the "-t" command line option to enable the serialized**  
 6560 **mode**  
 3165 *AcpiExec - Added the "-t" command line option to enable the serialized mode*  
 6561 *of the AML interpreter.*

6563 -----  
 6564 31 August 2006. Summary of changes for version 20060831:

6566 1) ACPI CA Core Subsystem:

6568 *Miscellaneous fixes for the Table Manager:*  
 6569 *- Correctly initialize internal common FADT for all 64-bit "X" fields*  
 6570 *- Fixed a couple table mapping issues during table load*  
 6571 *- Fixed a couple alignment issues for IA64*  
 6572 *- Initialize input array to zero in AcpiInitializeTables*  
 6573 *- Additional parameter validation for AcpiGetTable, AcpiGetTableHeader,*

6574 AcpiGetTableByIndex

6576 **Change for GPE support: when a "wake" GPE is received, all wake GPEs are**  
 6577 **now**  
 3181 *Change for GPE support: when a "wake" GPE is received, all wake GPEs are now*  
 6578 immediately disabled to prevent the waking GPE from firing again and to  
 6579 prevent other wake GPEs from interrupting the wake process.

6581 **Added the AcpiGpeCount global that tracks the number of processed GPEs,**  
 6582 **to**  
 3185 *Added the AcpiGpeCount global that tracks the number of processed GPEs, to*  
 6583 be used for debugging systems with a large number of ACPI interrupts.

6585 Implemented support for the "DMAR" ACPI table (DMA Redirection Table) in  
 6586 both the ACPICA headers and the disassembler.

6588 **Example Code and Data Size: These are the sizes for the OS-independent**  
 6589 **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The**  
 6590 **debug version of the code includes the debug output trace mechanism and**  
 6591 **has**  
 3193 *debug version of the code includes the debug output trace mechanism and has*  
 6592 a much larger code and data size.

6594	Previous Release:			
6595	Non-Debug Version:	77.8K Code,	16.5K Data,	94.3K Total
6596	Debug Version:	154.6K Code,	62.3K Data,	216.9K Total
6597	Current Release:			
6598	Non-Debug Version:	77.9K Code,	16.7K Data,	94.6K Total
6599	Debug Version:	154.9K Code,	62.6K Data,	217.5K Total

6602 2) iASL Compiler/Disassembler and Tools:

6604 Disassembler support for the DMAR ACPI table.

6606 -----  
 6607 23 August 2006. Summary of changes for version 20060823:

6609 1) ACPI CA Core Subsystem:

6611 The Table Manager component has been completely redesigned and  
 6612 **reimplemented. The new design is much simpler, and reduces the overall**  
 6613 **code**  
 6614 **and data size of the kernel-resident ACPICA by approximately 5%. Also, it**  
 6615 **is**  
 3214 *reimplemented. The new design is much simpler, and reduces the overall code*  
 3215 *and data size of the kernel-resident ACPICA by approximately 5%. Also, it is*  
 6616 now possible to obtain the ACPI tables very early during kernel  
 6617 initialization, even before dynamic memory management is initialized.  
 6618 (Alexey Starikovskiy, Fiodor Suietov, Bob Moore)

6620 Obsolete ACPICA interfaces:

6622 - **AcpiGetFirmwareTable: Use AcpiGetTable instead (works at early kernel**  
 6623 **init**  
 3222 *- AcpiGetFirmwareTable: Use AcpiGetTable instead (works at early kernel init*  
 6624 *time).*  
 6625 - AcpiLoadTable: Not needed.  
 6626 - AcpiUnloadTable: Not needed.

6628 New ACPICA interfaces:

6630 - **AcpiInitializeTables: Must be called before the table manager can be**  
 6631 **used.**  
 3229 *- AcpiInitializeTables: Must be called before the table manager can be used.*  
 6632 - AcpiReallocateRootTable: Used to transfer the root table to dynamically

6633 allocated memory after it becomes available.  
 6634 - **AcpiGetTableByIndex: Allows the host to easily enumerate all ACPI**  
 6635 **tables**  
 3232 *- AcpiGetTableByIndex: Allows the host to easily enumerate all ACPI tables*  
 6636 in the RSDT/XSDT.

6638 Other ACPICA changes:

6640 - **AcpiGetTableHeader returns the actual mapped table header, not a copy.**  
 6641 **Use**  
 3237 *- AcpiGetTableHeader returns the actual mapped table header, not a copy. Use*  
 6642 *AcpiOsUnmapMemory to free this mapping.*  
 6643 - AcpiGetTable returns the actual mapped table. The mapping is managed  
 6644 internally and must not be deleted by the caller. Use of this interface  
 6645 causes no additional dynamic memory allocation.  
 6646 - **AcpiFindRootPointer: Support for physical addressing has been**  
 6647 **eliminated,**  
 3242 *- AcpiFindRootPointer: Support for physical addressing has been eliminated,*  
 6648 *it appeared to be unused.*  
 6649 - The interface to AcpiOsMapMemory has changed to be consistent with the  
 6650 other allocation interfaces.  
 6651 - **The interface to AcpiOsGetRootPointer has changed to eliminate**  
 6652 **unnecessary**  
 3246 *- The interface to AcpiOsGetRootPointer has changed to eliminate unnecessary*  
 6653 *parameters.*  
 6654 - **ACPI\_PHYSICAL\_ADDRESS is now 32 bits on 32-bit platforms, 64 bits on**  
 6655 **64-**  
 3248 *- ACPI\_PHYSICAL\_ADDRESS is now 32 bits on 32-bit platforms, 64 bits on 64-*  
 6656 *bit platforms. Was previously 64 bits on all platforms.*  
 6657 - **The interface to the ACPI Global Lock acquire/release macros have**  
 6658 **changed**  
 3250 *- The interface to the ACPI Global Lock acquire/release macros have changed*  
 6659 slightly since ACPICA no longer keeps a local copy of the FACS with a  
 6660 constructed pointer to the actual global lock.

6662 Porting to the new table manager:

6664 - **AcpiInitializeTables: Must be called once, and can be called anytime**  
 6665 **during the OS initialization process. It allows the host to specify an**  
 6666 **area**  
 3257 *during the OS initialization process. It allows the host to specify an area*  
 6667 *of memory to be used to store the internal version of the RSDT/XSDT (root*  
 6668 *table). This allows the host to access ACPI tables before memory*  
 6669 *management*  
 3259 *table). This allows the host to access ACPI tables before memory management*  
 6670 *is initialized and running.*  
 6671 - **AcpiReallocateRootTable: Can be called after memory management is**  
 6672 **running**  
 3261 *- AcpiReallocateRootTable: Can be called after memory management is running*  
 6673 *to copy the root table to a dynamically allocated array, freeing up the*  
 6674 *scratch memory specified in the call to AcpiInitializeTables.*  
 6675 - **AcpiSubsystemInitialize: This existing interface is independent of the**  
 6676 **Table Manager, and does not have to be called before the Table Manager**  
 6677 **can**  
 3265 *Table Manager, and does not have to be called before the Table Manager can*  
 6678 *be used, it only must be called before the rest of ACPICA can be used.*  
 6679 - **ACPI Tables: Some changes have been made to the names and structure of**  
 6680 **the**  
 6681 **actbl.h and actbl1.h header files and may require changes to existing**  
 6682 **code.**  
 6683 **For example, bitfields have been completely removed because of their lack**  
 6684 **of**  
 3267 *- ACPI Tables: Some changes have been made to the names and structure of the*  
 3268 *actbl.h and actbl1.h header files and may require changes to existing code.*  
 3269 *For example, bitfields have been completely removed because of their lack of*  
 6685 *portability across C compilers.*

6686 - Update interfaces to the Global Lock acquire/release macros if local  
6687 versions are used. (see acwin.h)

6689 Obsolete files: tbconvrt.c, tbget.c, tbgetall.c, tbrsdt.c

6691 New files: tbfind.c

6693 Example Code and Data Size: These are the sizes for the OS-independent  
6694 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
6695 **debug version of the code includes the debug output trace mechanism and**  
6696 **has**  
3280 *debug version of the code includes the debug output trace mechanism and has*  
6697 a much larger code and data size.

6699 Previous Release:  
6700 Non-Debug Version: 80.7K Code, 17.9K Data, 98.6K Total  
6701 Debug Version: 161.0K Code, 65.1K Data, 226.1K Total  
6702 Current Release:  
6703 Non-Debug Version: 77.8K Code, 16.5K Data, 94.3K Total  
6704 Debug Version: 154.6K Code, 62.3K Data, 216.9K Total

6707 2) iASL Compiler/Disassembler and Tools:

6709 No changes for this release.

6711 -----  
6712 21 July 2006. Summary of changes for version 20060721:

6714 1) ACPI CA Core Subsystem:

6716 The full source code for the ASL test suite used to validate the iASL  
6717 compiler and the ACPICA core subsystem is being released with the ACPICA  
6718 **source for the first time. The source is contained in a separate package**  
6719 **and**  
6720 **consists of over 1100 files that exercise all ASL/AML operators. The**  
6721 **package**  
6722 **should appear on the Intel/ACPI web site shortly. (Valery Podrezov,**  
6723 **Fiodor**  
3302 *source for the first time. The source is contained in a separate package and*  
3303 *consists of over 1100 files that exercise all ASL/AML operators. The package*  
3304 *should appear on the Intel/ACPI web site shortly. (Valery Podrezov, Fiodor*  
6724 *Suietov)*

6726 Completed a new design and implementation for support of the ACPI Global  
6727 Lock. On the OS side, the global lock is now treated as a standard AML  
6728 mutex. Previously, multiple OS threads could "acquire" the global lock  
6729 **simultaneously. However, this could cause the BIOS to be starved out of**  
6730 **the**  
3310 *simultaneously. However, this could cause the BIOS to be starved out of the*  
6731 lock - especially in cases such as the Embedded Controller driver where  
6732 there is a tight coupling between the OS and the BIOS.

6734 Implemented an optimization for the ACPI Global Lock interrupt mechanism.  
6735 The Global Lock interrupt handler no longer queues the execution of a  
6736 **separate thread to signal the global lock semaphore. Instead, the**  
6737 **semaphore**  
3316 *separate thread to signal the global lock semaphore. Instead, the semaphore*  
6738 is signaled directly from the interrupt handler.

6740 Implemented support within the AML interpreter for package objects that  
6741 **contain a larger AML length (package list length) than the package**  
6742 **element**  
3320 *contain a larger AML length (package list length) than the package element*  
6743 count. In this case, the length of the package is truncated to match the  
6744 **package element count. Some BIOS code apparently modifies the package**

6745 **length**  
6746 **on the fly, and this change supports this behavior. Provides**  
6747 **compatibility**  
3322 *package element count. Some BIOS code apparently modifies the package length*  
3323 *on the fly, and this change supports this behavior. Provides compatibility*  
6748 with the MS AML interpreter. (With assistance from Fiodor Suietov)

6750 **Implemented a temporary fix for the BankValue parameter of a Bank Field**  
6751 **to**  
3326 *Implemented a temporary fix for the BankValue parameter of a Bank Field to*  
6752 support all constant values, now including the Zero and One opcodes.  
6753 **Evaluation of this parameter must eventually be converted to a full**  
6754 **TermArg**  
6755 **evaluation. A not-implemented error is now returned (temporarily) for**  
6756 **non-**  
3328 *Evaluation of this parameter must eventually be converted to a full TermArg*  
3329 *evaluation. A not-implemented error is now returned (temporarily) for non-*  
6757 constant values for this parameter.

6759 Fixed problem reports (Fiodor Suietov) integrated:  
6760 **- Fix for premature object deletion after CopyObject on Operation Region**  
6761 **(BZ**  
3333 *- Fix for premature object deletion after CopyObject on Operation Region (BZ*  
6762 350)

6764 Example Code and Data Size: These are the sizes for the OS-independent  
6765 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
6766 **debug version of the code includes the debug output trace mechanism and**  
6767 **has**  
3338 *debug version of the code includes the debug output trace mechanism and has*  
6768 a much larger code and data size.

6770 Previous Release:  
6771 Non-Debug Version: 80.7K Code, 18.0K Data, 98.7K Total  
6772 Debug Version: 160.9K Code, 65.1K Data, 226.0K Total  
6773 Current Release:  
6774 Non-Debug Version: 80.7K Code, 17.9K Data, 98.6K Total  
6775 Debug Version: 161.0K Code, 65.1K Data, 226.1K Total

6778 2) iASL Compiler/Disassembler and Tools:

6780 No changes for this release.

6782 -----  
6783 07 July 2006. Summary of changes for version 20060707:

6785 1) ACPI CA Core Subsystem:

6787 Added the ACPI\_PACKED\_POINTERS\_NOT\_SUPPORTED macro to support C compilers  
6788 that do not allow the initialization of address pointers within packed  
6789 structures - even though the hardware itself may support misaligned  
6790 transfers. Some of the debug data structures are packed by default to  
6791 minimize size.

6793 **Added an error message for the case where AcpiOsGetThreadId() returns**  
6794 **zero.**  
3364 *Added an error message for the case where AcpiOsGetThreadId() returns zero.*  
6795 A non-zero value is required by the core ACPICA code to ensure the proper  
6796 operation of AML mutexes and recursive control methods.

6798 The DSDT is now the only ACPI table that determines whether the AML  
6799 **interpreter is in 32-bit or 64-bit mode. Not really a functional change,**  
6800 **but**  
6801 **the hooks for per-table 32/64 switching have been removed from the code.**  
6802 **A**

3369 interpreter is in 32-bit or 64-bit mode. Not really a functional change, but  
 3370 the hooks for per-table 32/64 switching have been removed from the code. A  
 6803 clarification to the ACPI specification is forthcoming in ACPI 3.0B.

6805 Fixed a possible leak of an OwnerID in the error path of  
 6806 AcpiTbInitTableDescriptor (tbinstal.c), and migrated all table OwnerID  
 6807 deletion to a single place in AcpiTbUninstallTable to correct possible  
 6808 leaks  
 3375 deletion to a single place in AcpiTbUninstallTable to correct possible leaks  
 6809 when using the AcpiTbDeleteTablesByType interface (with assistance from  
 6810 Lance Ortiz.)

6812 Fixed a problem with Serialized control methods where the semaphore  
 6813 associated with the method could be over-sigaled after multiple method  
 6814 invocations.

6816 Fixed two issues with the locking of the internal namespace data  
 6817 structure.  
 3383 Fixed two issues with the locking of the internal namespace data structure.  
 6818 Both the Unload() operator and AcpiUnloadTable interface now lock the  
 6819 namespace during the namespace deletion associated with the table unload  
 6820 (with assistance from Linn Crosetto.)

6822 Fixed problem reports (Valery Podrezov) integrated:  
 6823 - Eliminate unnecessary memory allocation for CreateXxxxField (BZ 5426)

6825 Fixed problem reports (Fiodor Suietov) integrated:  
 6826 - Incomplete cleanup branches in AcpiTbGetTableRsdt (BZ 369)  
 6827 - On Address Space handler deletion, needless deactivation call (BZ 374)  
 6828 - AcpiRemoveAddressSpaceHandler: validate Device handle parameter (BZ  
 6829 375)  
 6830 - Possible memory leak, Notify sub-objects of Processor, Power,  
 6831 ThermalZone  
 3394 - AcpiRemoveAddressSpaceHandler: validate Device handle parameter (BZ 375)  
 3395 - Possible memory leak, Notify sub-objects of Processor, Power, ThermalZone  
 6832 (BZ 376)  
 6833 - AcpiRemoveAddressSpaceHandler: validate Handler parameter (BZ 378)  
 6834 - Minimum Length of RSMT should be validated (BZ 379)  
 6835 - AcpiRemoveNotifyHandler: return AE\_NOT\_EXIST if Processor Obj has no  
 6836 Handler (BZ 380)  
 6837 - AcpiUnloadTable: return AE\_NOT\_EXIST if no table of specified type  
 6838 loaded  
 3401 - AcpiUnloadTable: return AE\_NOT\_EXIST if no table of specified type loaded  
 6839 (BZ 381)

6841 Example Code and Data Size: These are the sizes for the OS-independent  
 6842 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 6843 debug version of the code includes the debug output trace mechanism and  
 6844 has  
 3406 debug version of the code includes the debug output trace mechanism and has  
 6845 a much larger code and data size.

6847	Previous Release:	
6848	Non-Debug Version:	80.5K Code, 17.8K Data, 98.3K Total
6849	Debug Version:	160.8K Code, 64.8K Data, 225.6K Total
6850	Current Release:	
6851	Non-Debug Version:	80.7K Code, 17.9K Data, 98.6K Total
6852	Debug Version:	161.0K Code, 65.1K Data, 226.1K Total

6855 2) iASL Compiler/Disassembler and Tools:

6857 Fixed problem reports:  
 6858 Compiler segfault when ASL contains a long (>1024) String declaration (BZ  
 6859 436)

6861 -----  
 6862 23 June 2006. Summary of changes for version 20060623:

6864 1) ACPI CA Core Subsystem:

6866 Implemented a new ACPI\_SPINLOCK type for the OSL lock interfaces. This  
 6867 allows the type to be customized to the host OS for improved efficiency  
 6868 (since a spinlock is usually a very small object.)

6870 Implemented support for "ignored" bits in the ACPI registers. According  
 6871 to  
 3432 Implemented support for "ignored" bits in the ACPI registers. According to  
 6872 the ACPI specification, these bits should be preserved when writing the  
 6873 registers via a read/modify/write cycle. There are 3 bits preserved in  
 6874 this  
 3434 registers via a read/modify/write cycle. There are 3 bits preserved in this  
 6875 manner: PM1\_CONTROL[0] (SCI\_EN), PM1\_CONTROL[9], and PM1\_STATUS[11].

6877 Implemented the initial deployment of new OSL mutex interfaces. Since  
 6878 some  
 3437 Implemented the initial deployment of new OSL mutex interfaces. Since some  
 6879 host operating systems have separate mutex and semaphore objects, this  
 6880 feature was requested. The base code now uses mutexes (and the new mutex  
 6881 interfaces) wherever a binary semaphore was used previously. However, for  
 6882 the current release, the mutex interfaces are defined as macros to map  
 6883 them  
 6884 to the existing semaphore interfaces. Therefore, no OSL changes are  
 6885 required  
 3441 the current release, the mutex interfaces are defined as macros to map them  
 3442 to the existing semaphore interfaces. Therefore, no OSL changes are required  
 6886 at this time. (See acpiosxf.h)

6888 Fixed several problems with the support for the control method SyncLevel  
 6889 parameter. The SyncLevel now works according to the ACPI specification  
 6890 and  
 6891 in concert with the Mutex SyncLevel parameter, since the current  
 6892 SyncLevel  
 6893 is a property of the executing thread. Mutual exclusion for control  
 6894 methods  
 3446 parameter. The SyncLevel now works according to the ACPI specification and  
 3447 in concert with the Mutex SyncLevel parameter, since the current SyncLevel  
 3448 is a property of the executing thread. Mutual exclusion for control methods  
 6895 is now implemented with a mutex instead of a semaphore.

6897 Fixed three instances of the use of the C shift operator in the bitfield  
 6898 support code (exfldio.c) to avoid the use of a shift value larger than  
 6899 the  
 6900 target data width. The behavior of C compilers is undefined in this case  
 6901 and  
 6902 can cause unpredictable results, and therefore the case must be detected  
 6903 and  
 3452 support code (exfldio.c) to avoid the use of a shift value larger than the  
 3453 target data width. The behavior of C compilers is undefined in this case and  
 3454 can cause unpredictable results, and therefore the case must be detected and  
 6904 avoided. (Fiodor Suietov)

6906 Added an info message whenever an SSMT or OEM table is loaded dynamically  
 6907 via the Load() or LoadTable() ASL operators. This should improve  
 6908 debugging  
 6909 capability since it will show exactly what tables have been loaded  
 6910 (beyond  
 3458 via the Load() or LoadTable() ASL operators. This should improve debugging  
 3459 capability since it will show exactly what tables have been loaded (beyond  
 6911 the tables present in the RSMT/XSMT.)

6913 Example Code and Data Size: These are the sizes for the OS-independent

6914 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 6915 **debug version of the code includes the debug output trace mechanism and**  
 6916 **has**  
 3464 *debug version of the code includes the debug output trace mechanism and has*  
 6917 a much larger code and data size.

6919 Previous Release:  
 6920 Non-Debug Version: 80.0K Code, 17.6K Data, 97.6K Total  
 6921 Debug Version: 160.2K Code, 64.7K Data, 224.9K Total  
 6922 Current Release:  
 6923 Non-Debug Version: 80.5K Code, 17.8K Data, 98.3K Total  
 6924 Debug Version: 160.8K Code, 64.8K Data, 225.6K Total

6927 2) iASL Compiler/Disassembler and Tools:

6929 No changes for this release.

6931 -----  
 6932 08 June 2006. Summary of changes for version 20060608:

6934 1) ACPI CA Core Subsystem:

6936 **Converted the locking mutex used for the ACPI hardware to a spinlock.**  
 6937 **This**  
 3484 *Converted the locking mutex used for the ACPI hardware to a spinlock. This*  
 6938 change should eliminate all problems caused by attempting to acquire a  
 6939 semaphore at interrupt level, and it means that all ACPICA external  
 6940 **interfaces that directly access the ACPI hardware can be safely called**  
 6941 **from**  
 6942 **interrupt level. OSL code that implements the semaphore interfaces should**  
 6943 **be**  
 3487 *interfaces that directly access the ACPI hardware can be safely called from*  
 3488 *interrupt level. OSL code that implements the semaphore interfaces should be*  
 6944 able to eliminate any workarounds for being called at interrupt level.

6946 Fixed a regression introduced in 20060526 where the ACPI device  
 6947 **initialization could be prematurely aborted with an AE\_NOT\_FOUND if a**  
 6948 **device**  
 3492 *initialization could be prematurely aborted with an AE\_NOT\_FOUND if a device*  
 6949 did not have an optional \_INI method.

6951 Fixed an IndexField issue where a write to the Data Register should be  
 6952 **limited in size to the AccessSize (width) of the IndexField itself. (BZ**  
 6953 **433,**  
 3496 *limited in size to the AccessSize (width) of the IndexField itself. (BZ 433,*  
 6954 Fiodor Suietov)

6956 Fixed problem reports (Valery Podrezov) integrated:  
 6957 - Allow store of ThermalZone objects to Debug object (BZ 5369/5370)

6959 Fixed problem reports (Fiodor Suietov) integrated:  
 6960 - AcpiGetTableHeader doesn't handle multiple instances correctly (BZ 364)

6962 Removed four global mutexes that were obsolete and were no longer being  
 6963 used.

6965 Example Code and Data Size: These are the sizes for the OS-independent  
 6966 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 6967 **debug version of the code includes the debug output trace mechanism and**  
 6968 **has**  
 3510 *debug version of the code includes the debug output trace mechanism and has*  
 6969 a much larger code and data size.

6971 Previous Release:  
 6972 Non-Debug Version: 80.0K Code, 17.7K Data, 97.7K Total

6973 Debug Version: 160.3K Code, 64.9K Data, 225.2K Total  
 6974 Current Release:  
 6975 Non-Debug Version: 80.0K Code, 17.6K Data, 97.6K Total  
 6976 Debug Version: 160.2K Code, 64.7K Data, 224.9K Total

6979 2) iASL Compiler/Disassembler and Tools:

6981 Fixed a fault when using -g option (get tables from registry) on Windows  
 6982 machines.

6984 Fixed problem reports integrated:  
 6985 - Generate error if CreateField NumBits parameter is zero. (BZ 405)  
 6986 - Fault if Offset/Length in Field unit is very large (BZ 432, Fiodor  
 6987 Suietov)  
 6988 - Global table revision override (-r) is ignored (BZ 413)

6990 -----  
 6991 26 May 2006. Summary of changes for version 20060526:

6993 1) ACPI CA Core Subsystem:

6995 Restructured, flattened, and simplified the internal interfaces for  
 6996 **namespace object evaluation - resulting in smaller code, less CPU stack**  
 6997 **use,**  
 3538 *namespace object evaluation - resulting in smaller code, less CPU stack use,*  
 6998 and fewer interfaces. (With assistance from Mikhail Kouzmich)

7000 **Fixed a problem with the CopyObject operator where the first parameter**  
 7001 **was**  
 7002 **not typed correctly for the parser, interpreter, compiler, and**  
 7003 **disassembler.**  
 3541 *Fixed a problem with the CopyObject operator where the first parameter was*  
 3542 *not typed correctly for the parser, interpreter, compiler, and disassembler.*  
 7004 Caused various errors and unexpected behavior.

7006 Fixed a problem where a ShiftLeft or ShiftRight of more than 64 bits  
 7007 produced incorrect results with some C compilers. Since the behavior of C  
 7008 compilers when the shift value is larger than the datatype width is  
 7009 **apparently not well defined, the interpreter now detects this condition**  
 7010 **and**  
 3548 *apparently not well defined, the interpreter now detects this condition and*  
 7011 simply returns zero as expected in all such cases. (BZ 395)

7013 Fixed problem reports (Valery Podrezov) integrated:  
 7014 - Update String-to-Integer conversion to match ACPI 3.0A spec (BZ 5329)  
 7015 - Allow interpreter to handle nested method declarations (BZ 5361)

7017 Fixed problem reports (Fiodor Suietov) integrated:  
 7018 - **AcpiTerminate doesn't free debug memory allocation list objects (BZ**  
 7019 **355)**  
 7020 - **After Core Subsystem shutdown, AcpiSubsystemStatus returns AE\_OK (BZ**  
 7021 **356)**  
 3556 *AcpiTerminate doesn't free debug memory allocation list objects (BZ 355)*  
 3557 *After Core Subsystem shutdown, AcpiSubsystemStatus returns AE\_OK (BZ 356)*  
 7022 - AcpiOsUnmapMemory for RSDP can be invoked inconsistently (BZ 357)  
 7023 - Resource Manager should return AE\_TYPE for non-device objects (BZ 358)  
 7024 - Incomplete cleanup branch in AcpiNsEvaluateRelative (BZ 359)  
 7025 - Use AcpiOsFree instead of ACPI\_FREE in AcpiRsSetSrsMethodData (BZ 360)  
 7026 - Incomplete cleanup branch in AcpiPsParseAml (BZ 361)  
 7027 - Incomplete cleanup branch in AcpiDsDeleteWalkState (BZ 362)  
 7028 - **AcpiGetTableHeader returns AE\_NO\_ACPI\_TABLES until DSDT is loaded (BZ**  
 7029 **365)**  
 3564 *AcpiGetTableHeader returns AE\_NO\_ACPI\_TABLES until DSDT is loaded (BZ 365)*  
 7030 - Status of the Global Initialization Handler call not used (BZ 366)  
 7031 - Incorrect object parameter to Global Initialization Handler (BZ 367)



7033 Example Code and Data Size: These are the sizes for the OS-independent  
 7034 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 7035 **debug version of the code includes the debug output trace mechanism and**  
 7036 **has**  
 3570 *debug version of the code includes the debug output trace mechanism and has*  
 7037 a much larger code and data size.

7039 Previous Release:  
 7040 Non-Debug Version: 79.8K Code, 17.7K Data, 97.5K Total  
 7041 Debug Version: 160.5K Code, 65.1K Data, 225.6K Total  
 7042 Current Release:  
 7043 Non-Debug Version: 80.0K Code, 17.7K Data, 97.7K Total  
 7044 Debug Version: 160.3K Code, 64.9K Data, 225.2K Total

#### 7047 2) iASL Compiler/Disassembler and Tools:

7049 Modified the parser to allow the names IO, DMA, and IRQ to be used as  
 7050 namespace identifiers with no collision with existing resource descriptor  
 7051 macro names. This provides compatibility with other ASL compilers and is  
 7052 **most useful for disassembly/recompilation of existing tables without**  
 7053 **parse**  
 3586 *most useful for disassembly/recompilation of existing tables without parse*  
 7054 errors. (With assistance from Thomas Renninger)

7056 Disassembler: fixed an incorrect disassembly problem with the  
 7057 DataTableRegion and CopyObject operators. Fixed a possible fault during  
 7058 disassembly of some Alias operators.

7060 -----  
 7061 12 May 2006. Summary of changes for version 20060512:

#### 7063 1) ACPI CA Core Subsystem:

7065 Replaced the AcpiOsQueueForExecution interface with a new interface named  
 7066 **AcpiOsExecute. The major difference is that the new interface does not**  
 7067 **have**  
 7068 **a Priority parameter, this appeared to be useless and has been replaced**  
 7069 **by**  
 7070 **a**  
 3599 *AcpiOsExecute. The major difference is that the new interface does not have*  
 3600 *a Priority parameter, this appeared to be useless and has been replaced by a*  
 7071 *Type parameter. The Type tells the host what type of execution is being*  
 7072 *requested, such as global lock handler, notify handler, GPE handler, etc.*  
 7073 **This allows the host to queue and execute the request as appropriate for**  
 7074 **the**  
 7075 **request type, possibly using different work queues and different**  
 7076 **priorities**  
 3603 *This allows the host to queue and execute the request as appropriate for the*  
 3604 *request type, possibly using different work queues and different priorities*  
 7077 *for the various request types. This enables fixes for multithreading*  
 7078 **deadlock problems such as BZ #5534, and will require changes to all**  
 7079 **existing**  
 3606 *deadlock problems such as BZ #5534, and will require changes to all existing*  
 7080 *OS interface layers. (Alexey Starikovskiy and Bob Moore)*

7082 **Fixed a possible memory leak associated with the support for the so-**  
 7083 **called**  
 3609 *Fixed a possible memory leak associated with the support for the so-called*  
 7084 *"implicit return" ACPI extension. Reported by FreeBSD, BZ #6514. (Fiodor*  
 7085 *Suietov)*

7087 Fixed a problem with the Load() operator where a table load from an  
 7088 **operation region could overwrite an internal table buffer by up to 7**  
 7089 **bytes**

7090 **and cause alignment faults on IPF systems. (With assistance from Luming**  
 7091 **Yu)**  
 3614 *operation region could overwrite an internal table buffer by up to 7 bytes*  
 3615 *and cause alignment faults on IPF systems. (With assistance from Luming Yu)*

7093 Example Code and Data Size: These are the sizes for the OS-independent  
 7094 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 7095 **debug version of the code includes the debug output trace mechanism and**  
 7096 **has**  
 3619 *debug version of the code includes the debug output trace mechanism and has*  
 7097 a much larger code and data size.

7099 Previous Release:  
 7100 Non-Debug Version: 79.7K Code, 17.7K Data, 97.4K Total  
 7101 Debug Version: 160.1K Code, 65.2K Data, 225.3K Total  
 7102 Current Release:  
 7103 Non-Debug Version: 79.8K Code, 17.7K Data, 97.5K Total  
 7104 Debug Version: 160.5K Code, 65.1K Data, 225.6K Total

#### 7108 2) iASL Compiler/Disassembler and Tools:

7110 **Disassembler: Implemented support to cross reference the internal**  
 7111 **namespace**  
 7112 **and automatically generate ASL External() statements for symbols not**  
 7113 **defined**  
 3633 *Disassembler: Implemented support to cross reference the internal namespace*  
 3634 *and automatically generate ASL External() statements for symbols not defined*  
 7114 *within the current table being disassembled. This will simplify the*  
 7115 **disassembly and recompilation of interdependent tables such as SSDTs**  
 7116 **since**  
 3636 *disassembly and recompilation of interdependent tables such as SSDTs since*  
 7117 these statements will no longer have to be added manually.

7119 Disassembler: Implemented experimental support to automatically detect  
 7120 **invocations of external control methods and generate appropriate**  
 7121 **External()**  
 7122 **statements. This is problematic because the AML cannot be correctly**  
 7123 **parsed**  
 7124 **until the number of arguments for each control method is known.**  
 7125 **Currently,**  
 3640 *invocations of external control methods and generate appropriate External()*  
 3641 *statements. This is problematic because the AML cannot be correctly parsed*  
 3642 *until the number of arguments for each control method is known. Currently,*  
 7126 *standalone method invocations and invocations as the source operand of a*  
 7127 *Store() statement are supported.*

7129 Disassembler: Implemented support for the ASL pseudo-operators LNotEqual,  
 7130 LLessEqual, and LGreaterEqual. Previously disassembled as LNot(LEqual()),  
 7131 LNot(LGreater()), and LNot(LLess()), this makes the disassembled ASL code  
 7132 more readable and likely closer to the original ASL source.

7134 -----  
 7135 21 April 2006. Summary of changes for version 20060421:

#### 7137 1) ACPI CA Core Subsystem:

7139 Removed a device initialization optimization introduced in 20051216 where  
 7140 the \_STA method was not run unless an \_INI was also present for the same  
 7141 **device. This optimization could cause problems because it could allow**  
 7142 **\_INI**  
 3658 *device. This optimization could cause problems because it could allow \_INI*  
 7143 *methods to be run within a not-present device subtree. (If a not-present*  
 7144 *device had no \_INI, \_STA would not be run, the not-present status would*  
 7145 *not*

3660 device had no `_INI`, `_STA` would not be run, the not-present status would not  
 7146 be discovered, and the children of the device would be incorrectly  
 7147 traversed.)

7149 Implemented a new `_STA` optimization where namespace subtrees that do not  
 7150 contain `_INI` are identified and ignored during device initialization.  
 7151 Selectively running `_STA` can significantly improve boot time on large  
 7152 machines (with assistance from Len Brown.)

7154 Implemented support for the device initialization case where the returned  
 7155 `_STA` flags indicate a device not-present but functioning. In this case,  
 7156 `_INI`  
 3670 `_STA` flags indicate a device not-present but functioning. In this case, `_INI`  
 7157 is not run, but the device children are examined for presence, as per the  
 7158 ACPI specification.

7160 Implemented an additional change to the `IndexField` support in order to  
 7161 conform to MS behavior. The value written to the `Index Register` is not  
 7162 simply a byte offset, it is a byte offset in units of the access width of  
 7163 the parent `Index Field`. (Fiodor Suietov)

7165 Defined and deployed a new OSL interface, `AcpiOsValidateAddress`. This  
 7166 interface is called during the creation of all AML operation regions, and  
 7167 allows the host OS to exert control over what addresses it will allow the  
 7168 AML code to access. Operation Regions whose addresses are disallowed will  
 7169 cause a runtime exception when they are actually accessed (will not  
 7170 affect  
 3683 cause a runtime exception when they are actually accessed (will not affect  
 7171 or abort table loading.) See `oswinxf` or `osunixxf` for an example  
 7172 implementation.

7174 Defined and deployed a new OSL interface, `AcpiOsValidateInterface`. This  
 7175 interface allows the host OS to match the various "optional"  
 7176 interface/behavior strings for the `_OSI` predefined control method as  
 7177 appropriate (with assistance from Bjorn Helgaas.) See `oswinxf` or `osunixxf`  
 7178 for an example implementation.

7180 Restructured and corrected various problems in the exception handling  
 7181 code  
 3693 Restructured and corrected various problems in the exception handling code  
 7182 paths within `DsCallControlMethod` and `DsTerminateControlMethod` in `dsmethod`  
 7183 (with assistance from Takayoshi Kochi.)

7185 Modified the Linux source converter to ignore quoted string literals  
 7186 while  
 7187 converting identifiers from mixed to lower case. This will correct  
 7188 problems  
 3697 Modified the Linux source converter to ignore quoted string literals while  
 3698 converting identifiers from mixed to lower case. This will correct problems  
 7189 with the disassembler and other areas where such strings must not be  
 7190 modified.

7192 The `ACPI_FUNCTION_*` macros no longer require quotes around the function  
 7193 name. This allows the Linux source converter to convert the names, now  
 7194 that  
 3703 name. This allows the Linux source converter to convert the names, now that  
 7195 the converter ignores quoted strings.

7197 Example Code and Data Size: These are the sizes for the OS-independent  
 7198 `acpica.lib` produced by the Microsoft Visual C++ 6.0 32-bit compiler. The  
 7199 debug version of the code includes the debug output trace mechanism and  
 7200 has  
 3708 debug version of the code includes the debug output trace mechanism and has  
 7201 a much larger code and data size.

7203 Previous Release:

7205 Non-Debug Version: 81.1K Code, 17.7K Data, 98.8K Total  
 7206 Debug Version: 158.9K Code, 64.9K Data, 223.8K Total  
 7207 Current Release:  
 7208 Non-Debug Version: 79.7K Code, 17.7K Data, 97.4K Total  
 7209 Debug Version: 160.1K Code, 65.2K Data, 225.3K Total

7212 2) iASL Compiler/Disassembler and Tools:

7214 Implemented 3 new warnings for iASL, and implemented multiple warning  
 7215 levels  
 3722 Implemented 3 new warnings for iASL, and implemented multiple warning levels  
 7216 (w2 flag).

7218 1) Ignored timeouts: If the `TimeoutValue` parameter to `Wait` or `Acquire` is  
 7219 not  
 3725 1) Ignored timeouts: If the `TimeoutValue` parameter to `Wait` or `Acquire` is not  
 7220 `WAIT_FOREVER` (0xFFFF) and the code does not examine the return value to  
 7221 check for the possible timeout, a warning is issued.

7223 2) Useless operators: If an ASL operator does not specify an optional  
 7224 target  
 3729 2) Useless operators: If an ASL operator does not specify an optional target  
 7225 operand and it also does not use the function return value from the  
 7226 operator, a warning is issued since the operator effectively does  
 7227 nothing.  
 3731 operator, a warning is issued since the operator effectively does nothing.

7229 3) Unreferenced objects: If a namespace object is created, but never  
 7230 referenced, a warning is issued. This is a warning level 2 since there  
 7231 are  
 7232 cases where this is ok, such as when a secondary table is loaded that  
 7233 uses  
 7234 the unreferenced objects. Even so, care is taken to only flag objects  
 7235 that  
 3734 referenced, a warning is issued. This is a warning level 2 since there are  
 3735 cases where this is ok, such as when a secondary table is loaded that uses  
 3736 the unreferenced objects. Even so, care is taken to only flag objects that  
 7236 don't look like they will ever be used. For example, the reserved methods  
 7237 (starting with an underscore) are usually not referenced because it is  
 7238 expected that the OS will invoke them.

7240 -----

7241 31 March 2006. Summary of changes for version 20060331:

7243 1) ACPI CA Core Subsystem:

7245 Implemented header file support for the following additional ACPI tables:  
 7246 `ASF!`, `BOOT`, `CPEP`, `DBGP`, `MCFG`, `SPCR`, `SPMI`, `TCPA`, and `WRT`. With this  
 7247 support,  
 7248 all current and known ACPI tables are now defined in the ACPICA headers  
 7249 and  
 3747 `ASF!`, `BOOT`, `CPEP`, `DBGP`, `MCFG`, `SPCR`, `SPMI`, `TCPA`, and `WRT`. With this support,  
 7248 all current and known ACPI tables are now defined in the ACPICA headers and  
 7250 are available for use by device drivers and other software.

7252 Implemented support to allow tables that contain ACPI names with invalid  
 7253 characters to be loaded. Previously, this would cause the table load to  
 7254 fail, but since there are several known cases of such tables on existing  
 7255 machines, this change was made to enable ACPI support for them. Also,  
 7256 this  
 3754 machines, this change was made to enable ACPI support for them. Also, this  
 7257 matches the behavior of the Microsoft ACPI implementation.

7259 Fixed a couple regressions introduced during the memory optimization in

7260 **the**  
 7357 *Fixed a couple regressions introduced during the memory optimization in the*  
 7261 *20060317 release. The namespace node definition required additional*  
 7262 **reorganization and an internal datatype that had been changed to 8-bit**  
 7263 **was**  
 7359 *reorganization and an internal datatype that had been changed to 8-bit was*  
 7264 *restored to 32-bit. (Valery Podrezov)*

7266 *Fixed a problem where a null pointer passed to AcpiUtDeleteGenericState*  
 7267 *could be passed through to AcpiOsReleaseObject which is unexpected. Such*  
 7268 *null pointers are now trapped and ignored, matching the behavior of the*  
 7269 *previous implementation before the deployment of AcpiOsReleaseObject.*  
 7270 *(Valery Podrezov, Fiodor Suietov)*

7272 **Fixed a memory mapping leak during the deletion of a SystemMemory**  
 7273 **operation**  
 7368 *Fixed a memory mapping leak during the deletion of a SystemMemory operation*  
 7274 *region where a cached memory mapping was not deleted. This became a*  
 7275 **noticeable problem for operation regions that are defined within**  
 7276 **frequently**  
 7370 *noticeable problem for operation regions that are defined within frequently*  
 7277 *used control methods. (Dana Meyers)*

7279 *Reorganized the ACPI table header files into two main files: one for the*  
 7280 **ACPI tables consumed by the ACPICA core, and another for the**  
 7281 **miscellaneous**  
 7282 **ACPI tables that are consumed by the drivers and other software. The**  
 7283 **various**  
 7374 *ACPI tables consumed by the ACPICA core, and another for the miscellaneous*  
 7375 *ACPI tables that are consumed by the drivers and other software. The various*  
 7284 *FADT definitions were merged into one common section and three different*  
 7285 *tables (ACPI 1.0, 1.0+, and 2.0)*

7287 *Example Code and Data Size: These are the sizes for the OS-independent*  
 7288 *acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. The*  
 7289 **debug version of the code includes the debug output trace mechanism and**  
 7290 **has**  
 7381 *debug version of the code includes the debug output trace mechanism and has*  
 7291 *a much larger code and data size.*

7293	Previous Release:	
7294	Non-Debug Version:	80.9K Code, 17.7K Data, 98.6K Total
7295	Debug Version:	158.7K Code, 64.8K Data, 223.5K Total
7296	Current Release:	
7297	Non-Debug Version:	81.1K Code, 17.7K Data, 98.8K Total
7298	Debug Version:	158.9K Code, 64.9K Data, 223.8K Total

7301 2) iASL Compiler/Disassembler and Tools:

7303 *Disassembler: Implemented support to decode and format all non-AML ACPI*  
 7304 *tables (tables other than DSDTs and SSDTs.) This includes the new tables*  
 7305 **added to the ACPICA headers, therefore all current and known ACPI tables**  
 7306 **are**  
 7396 *added to the ACPICA headers, therefore all current and known ACPI tables are*  
 7307 *supported.*

7309 *Disassembler: The change to allow ACPI names with invalid characters also*  
 7310 **enables the disassembly of such tables. Invalid characters within names**  
 7311 **are**  
 7380 *enables the disassembly of such tables. Invalid characters within names are*  
 7312 *changed to '\*' to make the name printable; the iASL compiler will still*  
 7313 *generate an error for such names, however, since this is an invalid ACPI*  
 7314 *character.*

7316 **Implemented an option for AcpiXtract (-a) to extract all tables found in**

7317 **the**  
 3805 *Implemented an option for AcpiXtract (-a) to extract all tables found in the*  
 7318 *input file. The default invocation extracts only the DSDTs and SSDTs.*

7320 *Fixed a couple of gcc generation issues for iASL and AcpiExec and added a*  
 7321 *makefile for the AcpiXtract utility.*

7323 -----  
 7324 17 March 2006. Summary of changes for version 20060317:

7326 1) ACPI CA Core Subsystem:

7328 *Implemented the use of a cache object for all internal namespace nodes.*  
 7329 *Since there are about 1000 static nodes in a typical system, this will*  
 7330 **decrease memory use for cache implementations that minimize per-**  
 7331 **allocation**  
 3818 *decrease memory use for cache implementations that minimize per-allocation*  
 7332 *overhead (such as a slab allocator.)*

7334 **Removed the reference count mechanism for internal namespace nodes, since**  
 7335 **it**  
 3821 *Removed the reference count mechanism for internal namespace nodes, since it*  
 7336 *was deemed unnecessary. This reduces the size of each namespace node by*  
 7337 **about 5%-10% on all platforms. Nodes are now 20 bytes for the 32-bit**  
 7338 **case,**  
 3823 *about 5%-10% on all platforms. Nodes are now 20 bytes for the 32-bit case,*  
 7339 *and 32 bytes for the 64-bit case.*

7341 **Optimized several internal data structures to reduce object size on 64-**  
 7342 **bit**  
 3826 *Optimized several internal data structures to reduce object size on 64-bit*  
 7343 *platforms by packing data within the 64-bit alignment. This includes the*  
 7344 *frequently used ACPI\_OPERAND\_OBJECT, of which there can be ~1000 static*  
 7345 *instances corresponding to the namespace objects.*

7347 **Added two new strings for the predefined \_OSI method: "Windows 2001.1**  
 7348 **SP1"**  
 3831 *Added two new strings for the predefined \_OSI method: "Windows 2001.1 SP1"*  
 7349 *and "Windows 2006".*

7351 *Split the allocation tracking mechanism out to a separate file, from*  
 7352 *utalloc.c to uttrack.c. This mechanism appears to be only useful for*  
 7353 *application-level code. Kernels may wish to not include uttrack.c in*  
 7354 *distributions.*

7356 **Removed all remnants of the obsolete ACPI\_REPORT\_\* macros and the**  
 7357 **associated**  
 3839 *Removed all remnants of the obsolete ACPI\_REPORT\_\* macros and the associated*  
 7358 *code. (These macros have been replaced by the ACPI\_ERROR and ACPI\_WARNING*  
 7359 *macros.)*

7361 **Code and Data Size: These are the sizes for the acpica.lib produced by**  
 7362 **the**  
 7363 **Microsoft Visual C++ 6.0 32-bit compiler. The values do not include any**  
 7364 **ACPI**  
 7365 **driver or OSPM code. The debug version of the code includes the debug**  
 7366 **output**  
 3843 *Code and Data Size: These are the sizes for the acpica.lib produced by the*  
 3844 *Microsoft Visual C++ 6.0 32-bit compiler. The values do not include any ACPI*  
 3845 *driver or OSPM code. The debug version of the code includes the debug output*  
 7367 *trace mechanism and has a much larger code and data size. Note that these*  
 7368 *values will vary depending on the efficiency of the compiler and the*  
 7369 *compiler options used during generation.*

7371	Previous Release:	
7372	Non-Debug Version:	81.1K Code, 17.8K Data, 98.9K Total

7373      Debug Version:       161.6K Code, 65.7K Data, 227.3K Total  
 7374      Current Release:  
 7375      Non-Debug Version:   80.9K Code, 17.7K Data, 98.6K Total  
 7376      Debug Version:       158.7K Code, 64.8K Data, 223.5K Total

7379 2) iASL Compiler/Disassembler and Tools:

7381 **Implemented an ANSI C version of the acpixtract utility. This version**  
 7382 **will**  
 7386 *Implemented an ANSI C version of the acpixtract utility. This version will*  
 7383 automatically extract the DSDT and all SSDTs from the input acpidump text  
 7384 file and dump the binary output to separate files. It can also display a  
 7385 summary of the input file including the headers for each table found and  
 7386 will extract any single ACPI table, with any signature. (See  
 7387 source/tools/acpixtract)

7389 -----  
 7390 10 March 2006. Summary of changes for version 20060310:

7392 1) ACPI CA Core Subsystem:

7394 Tagged all external interfaces to the subsystem with the new  
 7395 **ACPI\_EXPORT\_SYMBOL** macro. This macro can be defined as necessary to  
 7396 **assist**  
 7373 *ACPI\_EXPORT\_SYMBOL* macro. This macro can be defined as necessary to assist  
 7397 kernel integration. For Linux, the macro resolves to the EXPORT\_SYMBOL  
 7398 macro. The default definition is NULL.

7400 **Added the ACPI\_THREAD\_ID type for the return value from**  
 7401 **AcpiOsGetThreadId.**  
 7377 *Added the ACPI\_THREAD\_ID type for the return value from AcpiOsGetThreadId.*  
 7402 This allows the host to define this as necessary to simplify kernel  
 7403 integration. The default definition is ACPI\_NATIVE\_UINT.

7405 **Fixed two interpreter problems related to error processing, the deletion**  
 7406 **of**  
 7381 *Fixed two interpreter problems related to error processing, the deletion of*  
 7407 objects, and placing invalid pointers onto the internal operator result  
 7408 stack. BZ 6028, 6151 (Valery Podrezov)

7410 **Increased the reference count threshold where a warning is emitted for**  
 7411 **large**  
 7412 **reference counts in order to eliminate unnecessary warnings on systems**  
 7413 **with**  
 7385 *Increased the reference count threshold where a warning is emitted for large*  
 7386 *reference counts in order to eliminate unnecessary warnings on systems with*  
 7414 *large namespaces (especially 64-bit.) Increased the value from 0x400 to*  
 7415 *0x800.*

7417 **Due to universal disagreement as to the meaning of the 'c' in the**  
 7418 **calloc()**  
 7390 *Due to universal disagreement as to the meaning of the 'c' in the calloc()*  
 7419 *function, the ACPI\_MEM\_CALLOCATE macro has been renamed to*  
 7420 *ACPI\_ALLOCATE\_ZEROED so that the purpose of the interface is 'clear'.*  
 7421 *ACPI\_MEM\_ALLOCATE and ACPI\_MEM\_FREE are renamed to ACPI\_ALLOCATE and*  
 7422 *ACPI\_FREE.*

7424 **Code and Data Size: These are the sizes for the acpica.lib produced by**  
 7425 **the**  
 7426 **Microsoft Visual C++ 6.0 32-bit compiler. The values do not include any**  
 7427 **ACPI**  
 7428 **driver or OSPM code. The debug version of the code includes the debug**  
 7429 **output**  
 7396 *Code and Data Size: These are the sizes for the acpica.lib produced by the*  
 7397 *Microsoft Visual C++ 6.0 32-bit compiler. The values do not include any ACPI*

3898 *driver or OSPM code. The debug version of the code includes the debug output*  
 7430 *trace mechanism and has a much larger code and data size. Note that these*  
 7431 *values will vary depending on the efficiency of the compiler and the*  
 7432 *compiler options used during generation.*

7434      Previous Release:  
 7435      Non-Debug Version:   81.0K Code, 17.8K Data, 98.8K Total  
 7436      Debug Version:       161.4K Code, 65.7K Data, 227.1K Total  
 7437      Current Release:  
 7438      Non-Debug Version:   81.1K Code, 17.8K Data, 98.9K Total  
 7439      Debug Version:       161.6K Code, 65.7K Data, 227.3K Total

7442 2) iASL Compiler/Disassembler:

7444 Disassembler: implemented support for symbolic resource descriptor  
 7445 **references. If a CreateXxxxField operator references a fixed offset**  
 7446 **within**  
 7447 **a**  
 7448 **resource descriptor, a name is assigned to the descriptor and the offset**  
 7449 **is**  
 7394 *references. If a CreateXxxxField operator references a fixed offset within a*  
 7395 *resource descriptor, a name is assigned to the descriptor and the offset is*  
 7450 *translated to the appropriate resource tag and pathname. The addition of*  
 7451 *this support brings the disassembled code very close to the original ASL*  
 7452 **source code and helps eliminate run-time errors when the disassembled**  
 7453 **code**  
 7398 *source code and helps eliminate run-time errors when the disassembled code*  
 7454 *is modified (and recompiled) in such a way as to invalidate the original*  
 7455 *fixed offsets.*

7457 **Implemented support for a Descriptor Name as the last parameter to the**  
 7458 **ASL**  
 7392 *Implemented support for a Descriptor Name as the last parameter to the ASL*  
 7459 *Register() macro. This parameter was inadvertently left out of the ACPI*  
 7460 *specification, and will be added for ACPI 3.0b.*

7462 Fixed a problem where the use of the "\_OSI" string (versus the full path  
 7463 "\\\_OSI") caused an internal compiler error. ("No back ptr to op")

7465 **Fixed a problem with the error message that occurs when an invalid string**  
 7466 **is**  
 7467 **used for a \_HID object (such as one with an embedded asterisk:**  
 7468 **"\*PNP010A".)**  
 7329 *Fixed a problem with the error message that occurs when an invalid string is*  
 7390 *used for a \_HID object (such as one with an embedded asterisk: "\*PNP010A".)*  
 7469 *The correct message is now displayed.*

7471 -----  
 7472 17 February 2006. Summary of changes for version 20060217:

7474 1) ACPI CA Core Subsystem:

7476 **Implemented a change to the IndexField support to match the behavior of**  
 7477 **the**  
 7478 **Microsoft AML interpreter. The value written to the Index register is now**  
 7479 **a**  
 7480 **byte offset, no longer an index based upon the width of the Data**  
 7481 **register.**  
 7398 *Implemented a change to the IndexField support to match the behavior of the*  
 7399 *Microsoft AML interpreter. The value written to the Index register is now a*  
 7400 *byte offset, no longer an index based upon the width of the Data register.*  
 7482 *This should fix IndexField problems seen on some machines where the Data*  
 7483 *register is not exactly one byte wide. The ACPI specification will be*  
 7484 *clarified on this point.*

7486 Fixed a problem where several resource descriptor types could overrun the  
 7487 internal descriptor buffer due to size miscalculation: VendorShort,  
 7488 VendorLong, and Interrupt. This was noticed on IA64 machines, but could  
 7489 affect all platforms.

7491 **Fixed a problem where individual resource descriptors were misaligned  
 7492 within**  
 3950 *Fixed a problem where individual resource descriptors were misaligned within*  
 7493 the internal buffer, causing alignment faults on IA64 platforms.

7495 **Code and Data Size: These are the sizes for the acpica.lib produced by  
 7496 the**  
 7497 **Microsoft Visual C++ 6.0 32-bit compiler. The values do not include any  
 7498 ACPI**  
 7499 **driver or OSPM code. The debug version of the code includes the debug  
 7500 output**  
 3953 *Code and Data Size: These are the sizes for the acpica.lib produced by the*  
 3954 *Microsoft Visual C++ 6.0 32-bit compiler. The values do not include any ACPI*  
 3955 *driver or OSPM code. The debug version of the code includes the debug output*  
 7501 trace mechanism and has a much larger code and data size. Note that these  
 7502 values will vary depending on the efficiency of the compiler and the  
 7503 compiler options used during generation.

7505	Previous Release:	
7506	Non-Debug Version:	81.1K Code, 17.8K Data, 98.9K Total
7507	Debug Version:	161.3K Code, 65.6K Data, 226.9K Total
7508	Current Release:	
7509	Non-Debug Version:	81.0K Code, 17.8K Data, 98.8K Total
7510	Debug Version:	161.4K Code, 65.7K Data, 227.1K Total

7513 2) iASL Compiler/Disassembler:

7515 Implemented support for new reserved names: \_WDG and \_WED are Microsoft  
 7516 extensions for Windows Instrumentation Management, \_TDL is a new ACPI-  
 7517 defined method (Throttling Depth Limit.)

7519 Fixed a problem where a zero-length VendorShort or VendorLong resource  
 7520 descriptor was incorrectly emitted as a descriptor of length one.

7522 -----  
 7523 10 February 2006. Summary of changes for version 20060210:

7525 1) ACPI CA Core Subsystem:

7527 Removed a couple of extraneous ACPI\_ERROR messages that appeared during  
 7528 normal execution. These became apparent after the conversion from  
 7529 ACPI\_DEBUG\_PRINT.

7531 **Fixed a problem where the CreateField operator could hang if the BitIndex  
 7532 or**  
 3986 *Fixed a problem where the CreateField operator could hang if the BitIndex or*  
 7533 NumBits parameter referred to a named object. (Valery Podrezov, BZ 5359)

7535 Fixed a problem where a DeRefOf operation on a buffer object incorrectly  
 7536 failed with an exception. This also fixes a couple of related RefOf and  
 7537 DeRefOf issues. (Valery Podrezov, BZ 5360/5392/5387)

7539 **Fixed a problem where the AE\_BUFFER\_LIMIT exception was returned instead  
 7540 of**  
 7541 **AE\_STRING\_LIMIT on an out-of-bounds Index() operation. (Valery Podrezov,  
 7542 BZ**  
 3993 *Fixed a problem where the AE\_BUFFER\_LIMIT exception was returned instead of*  
 3994 *AE\_STRING\_LIMIT on an out-of-bounds Index() operation. (Valery Podrezov, BZ*  
 7543 5480)

7545 **Implemented a memory cleanup at the end of the execution of each  
 7546 iteration**  
 7547 **of an AML While() loop, preventing the accumulation of outstanding  
 7548 objects.**  
 3997 *Implemented a memory cleanup at the end of the execution of each iteration*  
 3998 *of an AML While() loop, preventing the accumulation of outstanding objects.*  
 7549 (Valery Podrezov, BZ 5427)

7551 **Eliminated a chunk of duplicate code in the object resolution code.  
 7552 (Valery**  
 4001 *Eliminated a chunk of duplicate code in the object resolution code. (Valery*  
 7553 Podrezov, BZ 5336)

7555 Fixed several warnings during the 64-bit code generation.

7557 **The AcpiSrc source code conversion tool now inserts one line of  
 7558 whitespace**  
 7559 **after an if() statement that is followed immediately by a comment,  
 7560 improving**  
 4006 *The AcpiSrc source code conversion tool now inserts one line of whitespace*  
 4007 *after an if() statement that is followed immediately by a comment, improving*  
 7561 readability of the Linux code.

7563 Code and Data Size: The current and previous library sizes for the core  
 7564 subsystem are shown below. These are the code and data sizes for the  
 7565 **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler.**  
 7566 **These**  
 7567 **values do not include any ACPI driver or OSPM code. The debug version of  
 7568 the**  
 4012 *acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These*  
 4013 *values do not include any ACPI driver or OSPM code. The debug version of the*  
 7569 code includes the debug output trace mechanism and has a much larger code  
 7570 **and data size. Note that these values will vary depending on the  
 7571 efficiency**  
 4015 *and data size. Note that these values will vary depending on the efficiency*  
 7572 of the compiler and the compiler options used during generation.

7574	Previous Release:	
7575	Non-Debug Version:	81.0K Code, 17.9K Data, 98.9K Total
7576	Debug Version:	161.3K Code, 65.7K Data, 227.0K Total
7577	Current Release:	
7578	Non-Debug Version:	81.1K Code, 17.8K Data, 98.9K Total
7579	Debug Version:	161.3K Code, 65.6K Data, 226.9K Total

7582 2) iASL Compiler/Disassembler:

7584 **Fixed a problem with the disassembly of a BankField operator with a  
 7585 complex**  
 4028 *Fixed a problem with the disassembly of a BankField operator with a complex*  
 7586 expression for the BankValue parameter.

7588 -----  
 7589 27 January 2006. Summary of changes for version 20060127:

7591 1) ACPI CA Core Subsystem:

7593 **Implemented support in the Resource Manager to allow unresolved  
 7594 namestring**  
 7595 **references within resource package objects for the \_PRT method. This  
 7596 support**  
 4036 *Implemented support in the Resource Manager to allow unresolved namestring*  
 4037 *references within resource package objects for the \_PRT method. This support*  
 7597 is in addition to the previously implemented unresolved reference support  
 7598 within the AML parser. If the interpreter slack mode is enabled, these  
 7599 **unresolved references will be passed through to the caller as a NULL**

7600 **package**  
4040 *unresolved references will be passed through to the caller as a NULL package*  
7601 entry.

7603 Implemented and deployed new macros and functions for error and warning  
7604 messages across the subsystem. These macros are simpler and generate less  
7605 code than their predecessors. The new macros ACPI\_ERROR, ACPI\_EXCEPTION,  
7606 ACPI\_WARNING, and ACPI\_INFO replace the ACPI\_REPORT\_\* macros. The older  
7607 macros remain defined to allow ACPI drivers time to migrate to the new  
7608 macros.

7610 **Implemented the ACPI\_CPU\_FLAGS type to simplify host OS integration of**  
7611 **the**  
4050 *Implemented the ACPI\_CPU\_FLAGS type to simplify host OS integration of the*  
7612 *Acquire/Release Lock OSL interfaces.*

7614 Fixed a problem where Alias ASL operators are sometimes not correctly  
7615 resolved, in both the interpreter and the iASL compiler.

7617 **Fixed several problems with the implementation of the**  
7618 **ConcatenateResTemplate**  
4056 *Fixed several problems with the implementation of the ConcatenateResTemplate*  
7619 ASL operator. As per the ACPI specification, zero length buffers are now  
7620 treated as a single EndTag. One-length buffers always cause a fatal  
7621 **exception. Non-zero length buffers that do not end with a full 2-byte**  
7622 **EndTag**  
4059 *exception. Non-zero length buffers that do not end with a full 2-byte EndTag*  
7623 cause a fatal exception.

7625 Fixed a possible structure overwrite in the AcpiGetObjectInfo external  
7626 interface. (With assistance from Thomas Renninger)

7628 Code and Data Size: The current and previous library sizes for the core  
7629 subsystem are shown below. These are the code and data sizes for the  
7630 **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler.**  
7631 **These**  
7632 **values do not include any ACPI driver or OSPM code. The debug version of**  
7633 **the**  
4067 *acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These*  
4068 *values do not include any ACPI driver or OSPM code. The debug version of the*  
7634 code includes the debug output trace mechanism and has a much larger code  
7635 **and data size. Note that these values will vary depending on the**  
7636 **efficiency**  
4070 *and data size. Note that these values will vary depending on the efficiency*  
7637 of the compiler and the compiler options used during generation.

7639	Previous Release:	
7640	Non-Debug Version:	83.1K Code, 18.4K Data, 101.5K Total
7641	Debug Version:	163.2K Code, 66.2K Data, 229.4K Total
7642	Current Release:	
7643	Non-Debug Version:	81.0K Code, 17.9K Data, 98.9K Total
7644	Debug Version:	161.3K Code, 65.7K Data, 227.0K Total

7647 2) iASL Compiler/Disassembler:

7649 **Fixed an internal error that was generated for any forward references to**  
7650 **ASL**  
4083 *Fixed an internal error that was generated for any forward references to ASL*  
7651 *Alias objects.*

7653 -----  
7654 13 January 2006. Summary of changes for version 20060113:

7656 1) ACPI CA Core Subsystem:

7658 Added 2006 copyright to all module headers and signons. This affects  
7659 virtually every file in the ACPICA core subsystem, iASL compiler, and the  
7660 utilities.

7661  
7662 **Enhanced the ACPICA error reporting in order to simplify user migration**  
7663 **to**  
4095 *Enhanced the ACPICA error reporting in order to simplify user migration to*  
7664 the non-debug version of ACPICA. Replaced all instances of the  
7665 **ACPI\_DEBUG\_PRINT macro invoked at the ACPI\_DB\_ERROR and ACPI\_DB\_WARN**  
7666 **debug**  
4097 *ACPI\_DEBUG\_PRINT macro invoked at the ACPI\_DB\_ERROR and ACPI\_DB\_WARN debug*  
7667 levels with the ACPI\_REPORT\_ERROR and ACPI\_REPORT\_WARNING macros,  
7668 **respectively. This preserves all error and warning messages in the non-**  
7669 **debug**  
4099 *respectively. This preserves all error and warning messages in the non-debug*  
7670 version of the ACPICA code (this has been referred to as the "debug lite"  
7671 option.) Over 200 cases were converted to create a total of over 380  
7672 **error/warning messages across the ACPICA code. This increases the code**  
7673 **and**  
7674 **data size of the default non-debug version of the code somewhat (about**  
7675 **13K),**  
4102 *error/warning messages across the ACPICA code. This increases the code and*  
4103 *data size of the default non-debug version of the code somewhat (about 13K),*  
7676 but all error/warning reporting may be disabled if desired (and code  
7677 eliminated) by specifying the ACPI\_NO\_ERROR\_MESSAGES compile-time  
7678 **configuration option. The size of the debug version of ACPICA remains**  
7679 **about**  
4106 *configuration option. The size of the debug version of ACPICA remains about*  
7680 the same.

7682 Fixed a memory leak within the AML Debugger "Set" command. One object was  
7683 not properly deleted for every successful invocation of the command.

7685 Code and Data Size: The current and previous library sizes for the core  
7686 subsystem are shown below. These are the code and data sizes for the  
7687 **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler.**  
7688 **These**  
7689 **values do not include any ACPI driver or OSPM code. The debug version of**  
7690 **the**  
4114 *acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These*  
4115 *values do not include any ACPI driver or OSPM code. The debug version of the*  
7691 code includes the debug output trace mechanism and has a much larger code  
7692 **and data size. Note that these values will vary depending on the**  
7693 **efficiency**  
4117 *and data size. Note that these values will vary depending on the efficiency*  
7694 of the compiler and the compiler options used during generation.

7696	Previous Release:	
7697	Non-Debug Version:	76.6K Code, 12.3K Data, 88.9K Total
7698	Debug Version:	163.7K Code, 67.5K Data, 231.2K Total
7699	Current Release:	
7700	Non-Debug Version:	83.1K Code, 18.4K Data, 101.5K Total
7701	Debug Version:	163.2K Code, 66.2K Data, 229.4K Total

7704 2) iASL Compiler/Disassembler:

7706 The compiler now officially supports the ACPI 3.0a specification that was  
7707 **released on December 30, 2005. (Specification is available at**  
7708 **www.acpi.info)**  
4131 *released on December 30, 2005. (Specification is available at www.acpi.info)*

7710 -----  
7711 16 December 2005. Summary of changes for version 20051216:

7713 1) ACPI CA Core Subsystem:

7715 Implemented optional support to allow unresolved names within ASL Package  
 7716 objects. A null object is inserted in the package when a named reference  
 7717 cannot be located in the current namespace. Enabled via the interpreter  
 7718 **slack flag, this should eliminate AE\_NOT\_FOUND exceptions seen on**  
 7719 **machines**  
 4141 *slack flag, this should eliminate AE\_NOT\_FOUND exceptions seen on machines*  
 7720 that contain such code.

7722 Implemented an optimization to the initialization sequence that can  
 7723 improve  
 7724 boot time. During ACPI device initialization, the `_STA` method is now run  
 7725 if  
 7726 and only if the `_INI` method exists. The `_STA` method is used to determine  
 7727 if  
 7728 the device is present; An `_INI` can only be run if `_STA` returns present,  
 7729 but  
 4144 Implemented an optimization to the initialization sequence that can improve  
 4145 boot time. During ACPI device initialization, the `_STA` method is now run if  
 4146 and only if the `_INI` method exists. The `_STA` method is used to determine if  
 4147 the device is present; An `_INI` can only be run if `_STA` returns present, but  
 7730 it is a waste of time to run the `_STA` method if the `_INI` does not exist.  
 7731 (Prototype and assistance from Dong Wei)

7733 Implemented use of the C99 `uintptr_t` for the pointer casting macros if it  
 7734 is  
 7735 available in the current compiler. Otherwise, the default (void \*) cast  
 7736 is  
 4151 Implemented use of the C99 `uintptr_t` for the pointer casting macros if it is  
 4152 available in the current compiler. Otherwise, the default (void \*) cast is  
 7737 used as before.

7739 Fixed some possible memory leaks found within the execution path of the  
 7740 Break, Continue, If, and CreateField operators. (Valery Podrezov)

7742 Fixed a problem introduced in the 20051202 release where an exception is  
 7743 generated during method execution if a control method attempts to declare  
 7744 another method.

7746 Moved resource descriptor string constants that are used by both the AML  
 7747 disassembler and AML debugger to the common utilities directory so that  
 7748 these components are independent.

7750 Implemented support in the AcpiExec utility (`-e` switch) to globally  
 7751 ignore  
 4166 Implemented support in the AcpiExec utility (`-e` switch) to globally ignore  
 7752 exceptions during control method execution (method is not aborted.)

7754 Added the rsinfo.c source file to the AcpiExec makefile for Linux/Unix  
 7755 generation.

7757 Code and Data Size: The current and previous library sizes for the core  
 7758 subsystem are shown below. These are the code and data sizes for the  
 7759 **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler.**  
 7760 **These**  
 7761 **values do not include any ACPI driver or OSPM code. The debug version of**  
 7762 **the**  
 4174 *acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These*  
 4175 *values do not include any ACPI driver or OSPM code. The debug version of the*  
 7763 *code includes the debug output trace mechanism and has a much larger code*  
 7764 *and data size. Note that these values will vary depending on the*  
 7765 *efficiency*  
 4177 *and data size. Note that these values will vary depending on the efficiency*  
 7766 of the compiler and the compiler options used during generation.

7768 Previous Release:

7769 Non-Debug Version: 76.3K Code, 12.3K Data, 88.6K Total  
 7770 Debug Version: 163.2K Code, 67.4K Data, 230.6K Total  
 7771 Current Release:  
 7772 Non-Debug Version: 76.6K Code, 12.3K Data, 88.9K Total  
 7773 Debug Version: 163.7K Code, 67.5K Data, 231.2K Total

7776 2) iASL Compiler/Disassembler:

7778 Fixed a problem where a CPU stack overflow fault could occur if a  
 7779 recursive  
 4190 Fixed a problem where a CPU stack overflow fault could occur if a recursive  
 7780 method call was made from within a Return statement.

7782 -----

7783 02 December 2005. Summary of changes for version 20051202:

7785 1) ACPI CA Core Subsystem:

7787 Modified the parsing of control methods to no longer create namespace  
 7788 objects during the first pass of the parse. Objects are now created only  
 7789 during the execute phase, at the moment the namespace creation operator  
 7790 is  
 7791 encountered in the AML (Name, OperationRegion, CreateByteField, etc.)  
 7792 This  
 4200 during the execute phase, at the moment the namespace creation operator is  
 4201 encountered in the AML (Name, OperationRegion, CreateByteField, etc.) This  
 7793 should eliminate ALREADY\_EXISTS exceptions seen on some machines where  
 7794 reentrant control methods are protected by an AML mutex. The mutex will  
 7795 now  
 7796 correctly block multiple threads from attempting to create the same  
 7797 object  
 4203 reentrant control methods are protected by an AML mutex. The mutex will now  
 4204 correctly block multiple threads from attempting to create the same object  
 7798 more than once.

7800 Increased the number of available Owner Ids for namespace object tracking  
 7801 from 32 to 255. This should eliminate the OWNER\_ID\_LIMIT exceptions seen  
 7802 on  
 7803 some machines with a large number of ACPI tables (either static or  
 7804 dynamic).  
 4208 from 32 to 255. This should eliminate the OWNER\_ID\_LIMIT exceptions seen on  
 4209 some machines with a large number of ACPI tables (either static or dynamic).

7806 Fixed a problem with the AcpiExec utility where a fault could occur when  
 7807 the  
 4211 Fixed a problem with the AcpiExec utility where a fault could occur when the  
 7808 `-b` switch (batch mode) is used.

7810 Enhanced the namespace dump routine to output the owner ID for each  
 7811 namespace object.

7813 Code and Data Size: The current and previous library sizes for the core  
 7814 subsystem are shown below. These are the code and data sizes for the  
 7815 **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler.**  
 7816 **These**  
 7817 **values do not include any ACPI driver or OSPM code. The debug version of**  
 7818 **the**  
 4219 *acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These*  
 4220 *values do not include any ACPI driver or OSPM code. The debug version of the*  
 7819 *code includes the debug output trace mechanism and has a much larger code*  
 7820 *and data size. Note that these values will vary depending on the*  
 7821 *efficiency*  
 4222 *and data size. Note that these values will vary depending on the efficiency*  
 7822 of the compiler and the compiler options used during generation.

7824 Previous Release:  
 7825 Non-Debug Version: 76.3K Code, 12.3K Data, 88.6K Total  
 7826 Debug Version: 163.0K Code, 67.4K Data, 230.4K Total  
 7827 Current Release:  
 7828 Non-Debug Version: 76.3K Code, 12.3K Data, 88.6K Total  
 7829 Debug Version: 163.2K Code, 67.4K Data, 230.6K Total

7832 2) iASL Compiler/Disassembler:

7834 **Fixed a parse error during compilation of certain Switch/Case constructs.**  
 7835 **To**  
 7836 **simplify the parse, the grammar now allows for multiple Default**  
 7837 **statements**  
 4235 *Fixed a parse error during compilation of certain Switch/Case constructs. To*  
 4236 *simplify the parse, the grammar now allows for multiple Default statements*  
 7838 *and this error is now detected and flagged during the analysis phase.*

7840 Disassembler: The disassembly now includes the contents of the original  
 7841 table header within a comment at the start of the file. This includes the  
 7842 name and version of the original ASL compiler.

7844 -----  
 7845 17 November 2005. Summary of changes for version 20051117:

7847 1) ACPI CA Core Subsystem:

7849 **Fixed a problem in the AML parser where the method thread count could be**  
 7850 **decremented below zero if any errors occurred during the method parse**  
 7851 **phase.**  
 7852 **This should eliminate AE\_AML\_METHOD\_LIMIT exceptions seen on some**  
 7853 **machines.**  
 4249 *decremented below zero if any errors occurred during the method parse phase.*  
 4250 *This should eliminate AE\_AML\_METHOD\_LIMIT exceptions seen on some machines.*  
 7854 *This also fixed a related regression with the mechanism that detects and*  
 7855 *corrects methods that cannot properly handle reentrancy (related to the*  
 7856 *deployment of the new OwnerId mechanism.)*

7858 **Eliminated the pre-parsing of control methods (to detect errors) during**  
 7859 **table load. Related to the problem above, this was causing unwind issues**  
 7860 **if**  
 7861 **any errors occurred during the parse, and it seemed to be overkill. A**  
 7862 **table**  
 4256 *table load. Related to the problem above, this was causing unwind issues if*  
 4257 *any errors occurred during the parse, and it seemed to be overkill. A table*  
 7863 *load should not be aborted if there are problems with any single control*  
 7864 *method, thus rendering this feature rather pointless.*

7866 **Fixed a problem with the new table-driven resource manager where an**  
 7867 **internal**  
 4261 *Fixed a problem with the new table-driven resource manager where an internal*  
 7868 *buffer overflow could occur for small resource templates.*

7870 **Implemented a new external interface, AcpiGetVendorResource. This**  
 7871 **interface**  
 7872 **will find and return a vendor-defined resource descriptor within a \_CRS**  
 7873 **or**  
 7874 **\_PRS method via an ACPI 3.0 UUID match. With assistance from Bjorn**  
 7875 **Helgaas.**  
 4264 *Implemented a new external interface, AcpiGetVendorResource. This interface*  
 4265 *will find and return a vendor-defined resource descriptor within a \_CRS or*  
 4266 *\_PRS method via an ACPI 3.0 UUID match. With assistance from Bjorn Helgaas.*

7877 **Removed the length limit (200) on string objects as per the upcoming ACPI**  
 7878 **3.0A specification. This affects the following areas of the interpreter:**  
 7879 **1)**

7880 **any implicit conversion of a Buffer to a String, 2) a String object**  
 7881 **result**  
 4269 *3.0A specification. This affects the following areas of the interpreter: 1)*  
 4270 *any implicit conversion of a Buffer to a String, 2) a String object result*  
 7882 *of the ASL Concatenate operator, 3) the String object result of the ASL*  
 7883 *ToString operator.*

7885 **Fixed a problem in the Windows OS interface layer (OSL) where a**  
 7886 **WAIT FOREVER**  
 4274 *Fixed a problem in the Windows OS interface layer (OSL) where a WAIT\_FOREVER*  
 7887 *on a semaphore object would incorrectly timeout. This allows the*  
 7888 *multithreading features of the AcpiExec utility to work properly under*  
 7889 *Windows.*

7891 Updated the Linux makefiles for the iASL compiler and AcpiExec to include  
 7892 the recently added file named "utresrc.c".

7894 **Code and Data Size: The current and previous library sizes for the core**  
 7895 **subsystem are shown below. These are the code and data sizes for the**  
 7896 **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler.**  
 7897 **These**  
 7898 **values do not include any ACPI driver or OSPM code. The debug version of**  
 7899 **the**  
 4284 *acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These*  
 4285 *values do not include any ACPI driver or OSPM code. The debug version of the*  
 7900 *code includes the debug output trace mechanism and has a much larger code*  
 7901 *and data size. Note that these values will vary depending on the*  
 7902 *efficiency*  
 4287 *and data size. Note that these values will vary depending on the efficiency*  
 7903 *of the compiler and the compiler options used during generation.*

7905 Previous Release:  
 7906 Non-Debug Version: 76.2K Code, 12.3K Data, 88.5K Total  
 7907 Debug Version: 163.0K Code, 67.4K Data, 230.4K Total  
 7908 Current Release:  
 7909 Non-Debug Version: 76.3K Code, 12.3K Data, 88.6K Total  
 7910 Debug Version: 163.0K Code, 67.4K Data, 230.4K Total

7913 2) iASL Compiler/Disassembler:

7915 **Removed the limit (200) on string objects as per the upcoming ACPI 3.0A**  
 7916 **specification. For the iASL compiler, this means that string literals**  
 7917 **within**  
 4301 *specification. For the iASL compiler, this means that string literals within*  
 7918 *the source ASL can be of any length.*

7920 **Enhanced the listing output to dump the AML code for resource descriptors**  
 7921 **immediately after the ASL code for each descriptor, instead of in a block**  
 7922 **at**  
 4305 *immediately after the ASL code for each descriptor, instead of in a block at*  
 7923 *the end of the entire resource template.*

7925 **Enhanced the compiler debug output to dump the entire original parse tree**  
 7926 **constructed during the parse phase, before any transforms are applied to**  
 7927 **the**  
 4309 *constructed during the parse phase, before any transforms are applied to the*  
 7928 *tree. The transformed tree is dumped also.*

7930 -----  
 7931 02 November 2005. Summary of changes for version 20051102:

7933 1) ACPI CA Core Subsystem:

7935 **Modified the subsystem initialization sequence to improve GPE support.**  
 7936 **The**



7937 GPE initialization has been split into two parts in order to defer  
 7938 execution  
 7939 of the \_PRW methods (Power Resources for Wake) until after the hardware  
 7940 is  
 4317 Modified the subsystem initialization sequence to improve GPE support. The  
 4318 GPE initialization has been split into two parts in order to defer execution  
 4319 of the \_PRW methods (Power Resources for Wake) until after the hardware is  
 7941 fully initialized and the SCI handler is installed. This allows the \_PRW  
 7942 methods to access fields protected by the Global Lock. This will fix  
 7943 systems  
 4321 methods to access fields protected by the Global Lock. This will fix systems  
 7944 where a NO\_GLOBAL\_LOCK exception has been seen during initialization.

7946 Converted the ACPI internal object disassemble and display code within  
 7947 the  
 4324 Converted the ACPI internal object disassemble and display code within the  
 7948 AML debugger to fully table-driven operation, reducing code size and  
 7949 increasing maintainability.

7951 Fixed a regression with the ConcatenateResTemplate() ASL operator  
 7952 introduced  
 4328 Fixed a regression with the ConcatenateResTemplate() ASL operator introduced  
 7953 in the 20051021 release.

7955 Implemented support for "local" internal ACPI object types within the  
 7956 debugger "Object" command and the AcpiWalkNamespace external interfaces.  
 7957 These local types include RegionFields, BankFields, IndexFields, Alias,  
 7958 and  
 4333 These local types include RegionFields, BankFields, IndexFields, Alias, and  
 7959 reference objects.

7961 Moved common AML resource handling code into a new file, "utresrc.c".  
 7962 This  
 4336 Moved common AML resource handling code into a new file, "utresrc.c". This  
 7963 code is shared by both the Resource Manager and the AML Debugger.

7965 Code and Data Size: The current and previous library sizes for the core  
 7966 subsystem are shown below. These are the code and data sizes for the  
 7967 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler.  
 7968 These  
 7969 values do not include any ACPI driver or OSPM code. The debug version of  
 7970 the  
 4341 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These  
 4342 values do not include any ACPI driver or OSPM code. The debug version of the  
 7971 code includes the debug output trace mechanism and has a much larger code  
 7972 and data size. Note that these values will vary depending on the  
 7973 efficiency  
 4344 and data size. Note that these values will vary depending on the efficiency  
 7974 of the compiler and the compiler options used during generation.

7976	Previous Release:	
7977	Non-Debug Version:	76.1K Code, 12.2K Data, 88.3K Total
7978	Debug Version:	163.5K Code, 67.0K Data, 230.5K Total
7979	Current Release:	
7980	Non-Debug Version:	76.2K Code, 12.3K Data, 88.5K Total
7981	Debug Version:	163.0K Code, 67.4K Data, 230.4K Total

7984 2) iASL Compiler/Disassembler:

7986 Fixed a problem with very large initializer lists (more than 4000  
 7987 elements)  
 4357 Fixed a problem with very large initializer lists (more than 4000 elements)  
 7988 for both Buffer and Package objects where the parse stack could overflow.

7990 Enhanced the pre-compile source code scan for non-ASCII characters to

7991 ignore  
 7992 characters within comment fields. The scan is now always performed and is  
 7993 no  
 4360 Enhanced the pre-compile source code scan for non-ASCII characters to ignore  
 4361 characters within comment fields. The scan is now always performed and is no  
 7994 longer optional, detecting invalid characters within a source file  
 7995 immediately rather than during the parse phase or later.

7997 Enhanced the ASL grammar definition to force early reductions on all  
 7998 list-  
 4365 Enhanced the ASL grammar definition to force early reductions on all list-  
 7999 style grammar elements so that the overall parse stack usage is greatly  
 8000 reduced. This should improve performance and reduce the possibility of  
 8001 parse  
 4367 reduced. This should improve performance and reduce the possibility of parse  
 8002 stack overflow.

8004 Eliminated all reduce/reduce conflicts in the iASL parser generation.  
 8005 Also,  
 4370 Eliminated all reduce/reduce conflicts in the iASL parser generation. Also,  
 8006 with the addition of a %expected statement, the compiler generates from  
 8007 source with no warnings.

8009 Fixed a possible segment fault in the disassembler if the input filename  
 8010 does not contain a "dot" extension (Thomas Renninger).

8012 -----  
 8013 21 October 2005. Summary of changes for version 20051021:

8015 1) ACPI CA Core Subsystem:

8017 Implemented support for the EM64T and other x86-64 processors. This  
 8018 essentially entails recognizing that these processors support non-aligned  
 8019 memory transfers. Previously, all 64-bit processors were assumed to lack  
 8020 hardware support for non-aligned transfers.

8022 Completed conversion of the Resource Manager to nearly full table-driven  
 8023 operation. Specifically, the resource conversion code (convert AML to  
 8024 internal format and the reverse) and the debug code to dump internal  
 8025 resource descriptors are fully table-driven, reducing code and data size  
 8026 and  
 4390 resource descriptors are fully table-driven, reducing code and data size and  
 8027 improving maintainability.

8029 The OSL interfaces for Acquire and Release Lock now use a 64-bit flag  
 8030 word  
 8031 on 64-bit processors instead of a fixed 32-bit word. (With assistance  
 8032 from  
 4393 The OSL interfaces for Acquire and Release Lock now use a 64-bit flag word  
 4394 on 64-bit processors instead of a fixed 32-bit word. (With assistance from  
 8033 Alexey Starikovskiy)

8035 Implemented support within the resource conversion code for the Type-  
 8036 Specific byte within the various ACPI 3.0 \*WordSpace macros.

8038 Fixed some issues within the resource conversion code for the type-  
 8039 specific  
 4400 Fixed some issues within the resource conversion code for the type-specific  
 8040 flags for both Memory and I/O address resource descriptors. For Memory,  
 8041 implemented support for the MTP and TTP flags. For I/O, split the TRS and  
 8042 TTP flags into two separate fields.

8044 Code and Data Size: The current and previous library sizes for the core  
 8045 subsystem are shown below. These are the code and data sizes for the  
 8046 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler.  
 8047 These

8048 values do not include any ACPI driver or OSPM code. The debug version of  
 8049 the  
 4407 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These  
 4408 values do not include any ACPI driver or OSPM code. The debug version of the  
 8050 code includes the debug output trace mechanism and has a much larger code  
 8051 and data size. Note that these values will vary depending on the  
 8052 efficiency  
 4410 and data size. Note that these values will vary depending on the efficiency  
 8053 of the compiler and the compiler options used during generation.

8055 Previous Release:  
 8056 Non-Debug Version: 77.1K Code, 12.1K Data, 89.2K Total  
 8057 Debug Version: 168.0K Code, 68.3K Data, 236.3K Total  
 8058 Current Release:  
 8059 Non-Debug Version: 76.1K Code, 12.2K Data, 88.3K Total  
 8060 Debug Version: 163.5K Code, 67.0K Data, 230.5K Total

#### 8064 2) iASL Compiler/Disassembler:

8066 Relaxed a compiler restriction that disallowed a ResourceIndex byte if  
 8067 the  
 4424 Relaxed a compiler restriction that disallowed a ResourceIndex byte if the  
 8068 corresponding ResourceSource string was not also present in a resource  
 8069 descriptor declaration. This restriction caused problems with existing  
 8070 AML/ASL code that includes the Index byte without the string. When such  
 8071 AML  
 4427 AML/ASL code that includes the Index byte without the string. When such AML  
 8072 was disassembled, it could not be compiled without modification. Further,  
 8073 the modified code created a resource template with a different size than  
 8074 the  
 8075 original, breaking code that used fixed offsets into the resource  
 8076 template  
 4429 the modified code created a resource template with a different size than the  
 4430 original, breaking code that used fixed offsets into the resource template  
 8077 buffer.

8079 Removed a recent feature of the disassembler to ignore a lone  
 8080 ResourceIndex  
 4433 Removed a recent feature of the disassembler to ignore a lone ResourceIndex  
 8081 byte. This byte is now emitted if present so that the exact AML can be  
 8082 reproduced when the disassembled code is recompiled.

8084 Improved comments and text alignment for the resource descriptor code  
 8085 emitted by the disassembler.

8087 Implemented disassembler support for the ACPI 3.0 AccessSize field within  
 8088 a  
 4440 Implemented disassembler support for the ACPI 3.0 AccessSize field within a  
 8089 Register() resource descriptor.

8091 -----

8092 30 September 2005. Summary of changes for version 20050930:

#### 8094 1) ACPI CA Core Subsystem:

8096 Completed a major overhaul of the Resource Manager code - specifically,  
 8097 optimizations in the area of the AML/internal resource conversion code.  
 8098 The  
 8099 code has been optimized to simplify and eliminate duplicated code, CPU  
 8100 stack  
 4449 optimizations in the area of the AML/internal resource conversion code. The  
 4450 code has been optimized to simplify and eliminate duplicated code, CPU stack  
 8101 use has been decreased by optimizing function parameters and local  
 8102 variables, and naming conventions across the manager have been

8103 standardized  
 4452 variables, and naming conventions across the manager have been standardized  
 8104 for clarity and ease of maintenance (this includes function, parameter,  
 8105 variable, and struct/typedef names.) The update may force changes in some  
 8106 driver code, depending on how resources are handled by the host OS.

8108 All Resource Manager dispatch and information tables have been moved to a  
 8109 single location for clarity and ease of maintenance. One new file was  
 8110 created, named "rsinfo.c".

8112 The ACPI return macros (return\_ACPI\_STATUS, etc.) have been modified to  
 8113 guarantee that the argument is not evaluated twice, making them less  
 8114 prone  
 4462 guarantee that the argument is not evaluated twice, making them less prone  
 8115 to macro side-effects. However, since there exists the possibility of  
 8116 additional stack use if a particular compiler cannot optimize them (such  
 8117 as  
 8118 in the debug generation case), the original macros are optionally  
 8119 available.  
 4464 additional stack use if a particular compiler cannot optimize them (such as  
 4465 in the debug generation case), the original macros are optionally available.  
 8120 Note that some invocations of the return\_VALUE macro may now cause size  
 8121 mismatch warnings; the return\_UINT8 and return\_UINT32 macros are provided  
 8122 to  
 4467 mismatch warnings; the return\_UINT8 and return\_UINT32 macros are provided to  
 8123 eliminate these. (From Randy Dunlap)

8125 Implemented a new mechanism to enable debug tracing for individual  
 8126 control  
 4470 Implemented a new mechanism to enable debug tracing for individual control  
 8127 methods. A new external interface, AcpiDebugTrace, is provided to enable  
 8128 this mechanism. The intent is to allow the host OS to easily enable and  
 8129 disable tracing for problematic control methods. This interface can be  
 8130 easily exposed to a user or debugger interface if desired. See the file  
 8131 psxfac.c for details.

8133 AcpiUtCalloc will now return a valid pointer if a length of zero is  
 8134 specified - a length of one is used and a warning is issued. This matches  
 8135 the behavior of AcpiUtAllocate.

8137 Code and Data Size: The current and previous library sizes for the core  
 8138 subsystem are shown below. These are the code and data sizes for the  
 8139 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler.  
 8140 These  
 8141 values do not include any ACPI driver or OSPM code. The debug version of  
 8142 the  
 4483 acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These  
 4484 values do not include any ACPI driver or OSPM code. The debug version of the  
 8143 code includes the debug output trace mechanism and has a much larger code  
 8144 and data size. Note that these values will vary depending on the  
 8145 efficiency  
 4486 and data size. Note that these values will vary depending on the efficiency  
 8146 of the compiler and the compiler options used during generation.

8148 Previous Release:  
 8149 Non-Debug Version: 77.5K Code, 12.0K Data, 89.5K Total  
 8150 Debug Version: 168.1K Code, 68.4K Data, 236.5K Total  
 8151 Current Release:  
 8152 Non-Debug Version: 77.1K Code, 12.1K Data, 89.2K Total  
 8153 Debug Version: 168.0K Code, 68.3K Data, 236.3K Total

#### 8156 2) iASL Compiler/Disassembler:

8158 A remark is issued if the effective compile-time length of a package or  
 8159 buffer is zero. Previously, this was a warning.

8161 -----  
 8162 16 September 2005. Summary of changes for version 20050916:

8164 1) ACPI CA Core Subsystem:

8166 Fixed a problem within the Resource Manager where support for the Generic  
 8167 **Register descriptor was not fully implemented. This descriptor is now**  
 8168 **fully**  
 4508 *Register descriptor was not fully implemented. This descriptor is now fully*  
 8169 recognized, parsed, disassembled, and displayed.

8171 Completely restructured the Resource Manager code to utilize table-driven  
 8172 **dispatch and lookup, eliminating many of the large switch() statements.**  
 8173 **This**  
 4512 *dispatch and lookup, eliminating many of the large switch() statements. This*  
 8174 reduces overall subsystem code size and code complexity. Affects the  
 8175 resource parsing and construction, disassembly, and debug dump output.

8177 Cleaned up and restructured the debug dump output for all resource  
 8178 descriptors. Improved readability of the output and reduced code size.

8180 Fixed a problem where changes to internal data structures caused the  
 8181 optional ACPI\_MUTEX\_DEBUG code to fail compilation if specified.

8183 Code and Data Size: The current and previous library sizes for the core  
 8184 subsystem are shown below. These are the code and data sizes for the  
 8185 **acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler.**  
 8186 **These**  
 8187 **values do not include any ACPI driver or OSPM code. The debug version of**  
 8188 **the**  
 4524 *acpica.lib produced by the Microsoft Visual C++ 6.0 32-bit compiler. These*  
 4525 *values do not include any ACPI driver or OSPM code. The debug version of the*  
 8189 code includes the debug output trace mechanism and has a much larger code  
 8190 **and data size. Note that these values will vary depending on the**  
 8191 **efficiency**  
 4527 *and data size. Note that these values will vary depending on the efficiency*  
 8192 of the compiler and the compiler options used during generation.

8194	Previous Release:	
8195	Non-Debug Version:	78.4K Code, 11.8K Data, 90.2K Total
8196	Debug Version:	169.6K Code, 69.9K Data, 239.5K Total
8197	Current Release:	
8198	Non-Debug Version:	77.5K Code, 12.0K Data, 89.5K Total
8199	Debug Version:	168.1K Code, 68.4K Data, 236.5K Total

8202 2) iASL Compiler/Disassembler:

8204 **Updated the disassembler to automatically insert an EndDependentFn()**  
 8205 **macro**  
 4540 *Updated the disassembler to automatically insert an EndDependentFn() macro*  
 8206 into the ASL stream if this macro is missing in the original AML code,  
 8207 simplifying compilation of the resulting ASL module.

8209 Fixed a problem in the disassembler where a disassembled ResourceSource  
 8210 **string (within a large resource descriptor) was not surrounded by quotes**  
 8211 **and**  
 4545 *string (within a large resource descriptor) was not surrounded by quotes and*  
 8212 not followed by a comma, causing errors when the resulting ASL module was  
 8213 compiled. Also, escape sequences within a ResourceSource string are now  
 8214 handled correctly (especially "\\")

8216 -----  
 8217 02 September 2005. Summary of changes for version 20050902:

8219 1) ACPI CA Core Subsystem:

8221 Fixed a problem with the internal Owner ID allocation and deallocation  
 8222 mechanisms for control method execution and recursive method invocation.  
 8223 This should eliminate the OWNER\_ID\_LIMIT exceptions and "Invalid OwnerId"  
 8224 messages seen on some systems. Recursive method invocation depth is  
 8225 currently limited to 255. (Alexey Starikovskiy)

8227 Completely eliminated all vestiges of support for the "module-level  
 8228 **executable code" until this support is fully implemented and debugged.**  
 8229 **This**  
 4562 *executable code" until this support is fully implemented and debugged. This*  
 8230 should eliminate the NO\_RETURN\_VALUE exceptions seen during table load on  
 8231 some systems that invoke this support.

8233 **Fixed a problem within the resource manager code where the transaction**  
 8234 **flags**  
 4566 *Fixed a problem within the resource manager code where the transaction flags*  
 8235 for a 64-bit address descriptor were handled incorrectly in the type-  
 8236 specific flag byte.

8238 **Consolidated duplicate code within the address descriptor resource**  
 8239 **manager**  
 4570 *Consolidated duplicate code within the address descriptor resource manager*  
 8240 code, reducing overall subsystem code size.

8242 Fixed a fault when using the AML debugger "disassemble" command to  
 8243 disassemble individual control methods.

8245 Removed references to the "release\_current" directory within the Unix  
 8246 release package.

8248 Code and Data Size: The current and previous core subsystem library sizes  
 8249 are shown below. These are the code and data sizes for the acpica.lib  
 8250 produced by the Microsoft Visual C++ 6.0 compiler. These values do not  
 8251 **include any ACPI driver or OSPM code. The debug version of the code**  
 8252 **includes**  
 8253 **the debug output trace mechanism and has a much larger code and data**  
 8254 **size.**  
 8255 **Note that these values will vary depending on the efficiency of the**  
 8256 **compiler**  
 4582 *include any ACPI driver or OSPM code. The debug version of the code includes*  
 4583 *the debug output trace mechanism and has a much larger code and data size.*  
 4584 *Note that these values will vary depending on the efficiency of the compiler*  
 8257 and the compiler options used during generation.

8259	Previous Release:	
8260	Non-Debug Version:	78.6K Code, 11.7K Data, 90.3K Total
8261	Debug Version:	170.0K Code, 69.9K Data, 239.9K Total
8262	Current Release:	
8263	Non-Debug Version:	78.4K Code, 11.8K Data, 90.2K Total
8264	Debug Version:	169.6K Code, 69.9K Data, 239.5K Total

8267 2) iASL Compiler/Disassembler:

8269 **Implemented an error check for illegal duplicate values in the interrupt**  
 8270 **and**  
 4597 *Implemented an error check for illegal duplicate values in the interrupt and*  
 8271 dma lists for the following ASL macros: Dma(), Irq(), IrqNoFlags(), and  
 8272 Interrupt().

8274 **Implemented error checking for the Irq() and IrqNoFlags() macros to**  
 8275 **detect**  
 4601 *Implemented error checking for the Irq() and IrqNoFlags() macros to detect*  
 8276 too many values in the interrupt list (16 max) and invalid values in the

8277 list (range 0 - 15)

8279 The maximum length string literal within an ASL file is now restricted to  
8280 200 characters as per the ACPI specification.

8282 Fixed a fault when using the -ln option (generate namespace listing).

8284 Implemented an error check to determine if a DescriptorName within a  
8285 resource descriptor has already been used within the current scope.

8287 -----  
8288 15 August 2005. Summary of changes for version 20050815:  
8289  
8290 1) ACPI CA Core Subsystem:  
8291  
8292 **Implemented a full bitwise compare to determine if a table load request**  
8293 **is attempting to load a duplicate table. The compare is performed if the**  
8294 **table**  
8295 *Implemented a full bitwise compare to determine if a table load request is*  
8296 *attempting to load a duplicate table. The compare is performed if the table*  
8297 *signatures and table lengths match. This will allow different tables with*  
8298 *the same OEM Table ID and revision to be loaded - probably against the*  
8299 *ACPI*  
8300 *the same OEM Table ID and revision to be loaded - probably against the ACPI*  
8301 *specification, but discovered in the field nonetheless.*  
8302  
8303 Added the changes.txt logfile to each of the zipped release packages.  
8304  
8305 Code and Data Size: Current and previous core subsystem library sizes are  
8306 **shown below. These are the code and data sizes for the acpica.lib**  
8307 **produced**  
8308 *shown below. These are the code and data sizes for the acpica.lib produced*  
8309 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
8310 *any ACPI driver or OSPM code. The debug version of the code includes the*  
8311 *debug output trace mechanism and has a much larger code and data size.*  
8312 **Note**  
8313 **that these values will vary depending on the efficiency of the compiler**  
8314 **and**  
8315 *debug output trace mechanism and has a much larger code and data size. Note*  
8316 *that these values will vary depending on the efficiency of the compiler and*  
8317 *the compiler options used during generation.*  
8318  
8319 Previous Release:  
8320 Non-Debug Version: 78.6K Code, 11.7K Data, 90.3K Total  
8321 Debug Version: 167.0K Code, 69.9K Data, 236.9K Total  
8322 Current Release:  
8323 Non-Debug Version: 78.6K Code, 11.7K Data, 90.3K Total  
8324 Debug Version: 170.0K Code, 69.9K Data, 239.9K Total  
8325  
8326 2) iASL Compiler/Disassembler:  
8327  
8328 Fixed a problem where incorrect AML code could be generated for Package  
8329 objects if optimization is disabled (via the -oa switch).  
8330  
8331 Fixed a problem with where incorrect AML code is generated for variable-  
8332 **length packages when the package length is not specified and the number**  
8333 **of**  
8334 *length packages when the package length is not specified and the number of*  
8335 *initializer values is greater than 255.*  
8336  
8337 -----  
8338 29 July 2005. Summary of changes for version 20050729:

8336 1) ACPI CA Core Subsystem:

8338 **Implemented support to ignore an attempt to install/load a particular**  
8339 **ACPI**  
8340 *Implemented support to ignore an attempt to install/load a particular ACPI*  
8341 *table more than once. Apparently there exists BIOS code that repeatedly*  
8342 *attempts to load the same SSDT upon certain events. With assistance from*  
8343 *Venkatesh Pallipadi.*  
8344  
8345 Restructured the main interface to the AML parser in order to correctly  
8346 **handle all exceptional conditions. This will prevent leakage of the**  
8347 **OwnerId**  
8348 **resource and should eliminate the AE\_OWNER\_ID\_LIMIT exceptions seen on**  
8349 **some**  
8350 *handle all exceptional conditions. This will prevent leakage of the OwnerId*  
8351 *resource and should eliminate the AE\_OWNER\_ID\_LIMIT exceptions seen on some*  
8352 *machines. With assistance from Alexey Starikovskiy.*  
8353  
8354 Support for "module level code" has been disabled in this version due to  
8355 **a**  
8356 **number of issues that have appeared on various machines. The support can**  
8357 **be**  
8358 *Support for "module level code" has been disabled in this version due to a*  
8359 *number of issues that have appeared on various machines. The support can be*  
8360 *enabled by defining ACPI\_ENABLE\_MODULE\_LEVEL\_CODE during subsystem*  
8361 *compilation. When the issues are fully resolved, the code will be enabled*  
8362 *by*  
8363 *compilation. When the issues are fully resolved, the code will be enabled by*  
8364 *default again.*  
8365  
8366 Modified the internal functions for debug print support to define the  
8367 **FunctionName** parameter as a (const char \*) for compatibility with  
8368 **compiler**  
8369 *FunctionName* parameter as a (const char \*) for compatibility with compiler  
8370 built-in macros such as \_\_FUNCTION\_\_, etc.  
8371  
8372 Linted the entire ACPICA source tree for both 32-bit and 64-bit.  
8373  
8374 **Implemented support to display an object count summary for the AML**  
8375 **Debugger**  
8376 *Implemented support to display an object count summary for the AML Debugger*  
8377 *commands Object and Methods.*  
8378  
8379 Code and Data Size: Current and previous core subsystem library sizes are  
8380 **shown below. These are the code and data sizes for the acpica.lib**  
8381 **produced**  
8382 *shown below. These are the code and data sizes for the acpica.lib produced*  
8383 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
8384 *any ACPI driver or OSPM code. The debug version of the code includes the*  
8385 *debug output trace mechanism and has a much larger code and data size.*  
8386 **Note**  
8387 **that these values will vary depending on the efficiency of the compiler**  
8388 **and**  
8389 *debug output trace mechanism and has a much larger code and data size. Note*  
8390 *that these values will vary depending on the efficiency of the compiler and*  
8391 *the compiler options used during generation.*  
8392  
8393 Previous Release:  
8394 Non-Debug Version: 78.6K Code, 11.6K Data, 90.2K Total  
8395 Debug Version: 170.0K Code, 69.7K Data, 239.7K Total  
8396 Current Release:  
8397 Non-Debug Version: 78.6K Code, 11.7K Data, 90.3K Total  
8398 Debug Version: 167.0K Code, 69.9K Data, 236.9K Total

8390 2) iASL Compiler/Disassembler:

8392 Fixed a regression that appeared in the 20050708 version of the compiler  
 8393 **where an error message was inadvertently emitted for invocations of the**  
 8394 **\_OSI**  
 4701 *where an error message was inadvertently emitted for invocations of the \_OSI*  
 8395 reserved control method.

8397 -----  
 8398 08 July 2005. Summary of changes for version 20050708:

8400 1) ACPI CA Core Subsystem:

8402 The use of the CPU stack in the debug version of the subsystem has been  
 8403 considerably reduced. Previously, a debug structure was declared in every  
 8404 function that used the debug macros. This structure has been removed in  
 8405 favor of declaring the individual elements as parameters to the debug  
 8406 **functions. This reduces the cumulative stack use during nested execution**  
 8407 **of**  
 8408 **ACPI function calls at the cost of a small increase in the code size of**  
 8409 **the**  
 8410 **debug version of the subsystem. With assistance from Alexey Starikovskiy**  
 8411 **and**  
 4713 *functions. This reduces the cumulative stack use during nested execution of*  
 4714 *ACPI function calls at the cost of a small increase in the code size of the*  
 4715 *debug version of the subsystem. With assistance from Alexey Starikovskiy and*  
 8412 Len Brown.

8414 Added the ACPI\_GET\_FUNCTION\_NAME macro to enable the compiler-dependent  
 8415 headers to define a macro that will return the current function name at  
 8416 **runtime (such as \_\_FUNCTION\_\_ or \_\_func\_\_, etc.) The function name is used**  
 8417 **by**  
 4720 *runtime (such as \_\_FUNCTION\_\_ or \_\_func\_\_, etc.) The function name is used by*  
 8418 *the debug trace output. If ACPI\_GET\_FUNCTION\_NAME is not defined in the*  
 8419 **compiler-dependent header, the function name is saved on the CPU stack**  
 8420 **(one**  
 4722 *compiler-dependent header, the function name is saved on the CPU stack (one*  
 8421 *pointer per function.) This mechanism is used because apparently there*  
 8422 **exists no standard ANSI-C defined macro that that returns the function**  
 8423 **name.**  
 4724 *exists no standard ANSI-C defined macro that that returns the function name.*

8425 Redesigned and reimplemented the "Owner ID" mechanism used to track  
 8426 namespace objects created/deleted by ACPI tables and control method  
 8427 **execution. A bitmap is now used to allocate and free the IDs, thus**  
 8428 **solving**  
 8429 **the wraparound problem present in the previous implementation. The size**  
 8430 **of**  
 4728 *execution. A bitmap is now used to allocate and free the IDs, thus solving*  
 4729 *the wraparound problem present in the previous implementation. The size of*  
 8431 *the namespace node descriptor was reduced by 2 bytes as a result (Alexey*  
 8432 *Starikovskiy).*

8434 **Removed the UINT32\_BIT and UINT16\_BIT types that were used for the**  
 8435 **bitfield**  
 4733 *Removed the UINT32\_BIT and UINT16\_BIT types that were used for the bitfield*  
 8436 *flag definitions within the headers for the predefined ACPI tables. These*  
 8437 **have been replaced by UINT8\_BIT in order to increase the code portability**  
 8438 **of**  
 4735 *have been replaced by UINT8\_BIT in order to increase the code portability of*  
 8439 *the subsystem. If the use of UINT8 remains a problem, we may be forced to*  
 8440 *eliminate bitfields entirely because of a lack of portability.*

8442 **Enhanced the performance of the AcpiUtUpdateObjectReference procedure.**  
 8443 **This**  
 8444 **is a frequently used function and this improvement increases the**  
 8445 **performance**

4739 *Enhanced the performance of the AcpiUtUpdateObjectReference procedure. This*  
 4740 *is a frequently used function and this improvement increases the performance*  
 8446 *of the entire subsystem (Alexey Starikovskiy).*

8448 Fixed several possible memory leaks and the inverse - premature object  
 8449 deletion (Alexey Starikovskiy).

8451 Code and Data Size: Current and previous core subsystem library sizes are  
 8452 **shown below. These are the code and data sizes for the acpica.lib**  
 8453 **produced**  
 4747 *shown below. These are the code and data sizes for the acpica.lib produced*  
 8454 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 8455 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 8456 **debug output trace mechanism and has a much larger code and data size.**  
 8457 **Note**  
 8458 **that these values will vary depending on the efficiency of the compiler**  
 8459 **and**  
 4750 *debug output trace mechanism and has a much larger code and data size. Note*  
 4751 *that these values will vary depending on the efficiency of the compiler and*  
 8460 *the compiler options used during generation.*

8462 Previous Release:  
 8463 Non-Debug Version: 78.6K Code, 11.5K Data, 90.1K Total  
 8464 Debug Version: 165.2K Code, 69.6K Data, 234.8K Total  
 8465 Current Release:  
 8466 Non-Debug Version: 78.6K Code, 11.6K Data, 90.2K Total  
 8467 Debug Version: 170.0K Code, 69.7K Data, 239.7K Total

8469 -----  
 8470 24 June 2005. Summary of changes for version 20050624:

8472 1) ACPI CA Core Subsystem:

8474 Modified the new OSL cache interfaces to use ACPI\_CACHE\_T as the type for  
 8475 **the host-defined cache object. This allows the OSL implementation to**  
 8476 **define**  
 4767 *the host-defined cache object. This allows the OSL implementation to define*  
 8477 *and type this object in any manner desired, simplifying the OSL*  
 8478 *implementation. For example, ACPI\_CACHE\_T is defined as kmem\_cache\_t for*  
 8479 *Linux, and should be defined in the OS-specific header file for other*  
 8480 *operating systems as required.*

8482 Changed the interface to AcpiOsAcquireObject to directly return the  
 8483 requested object as the function return (instead of ACPI\_STATUS.) This  
 8484 change was made for performance reasons, since this is the purpose of the  
 8485 interface in the first place. AcpiOsAcquireObject is now similar to the  
 8486 AcpiOsAllocate interface.

8488 **Implemented a new AML debugger command named Businfo. This command**  
 8489 **displays**  
 8490 **information about all devices that have an associate \_PRT object. The**  
 8491 **\_ADR,**  
 4779 *Implemented a new AML debugger command named Businfo. This command displays*  
 4780 *information about all devices that have an associate \_PRT object. The \_ADR,*  
 8492 *\_HID, \_UID, and \_CID are displayed for these devices.*

8494 **Modified the initialization sequence in AcpiInitializeSubsystem to call**  
 8495 **the**  
 8496 **OSL interface AcpiOsInitialize first, before any local initialization.**  
 8497 **This**  
 4783 *Modified the initialization sequence in AcpiInitializeSubsystem to call the*  
 4784 *OSL interface AcpiOsInitialize first, before any local initialization. This*  
 8498 *change was required because the global initialization now calls OSL*  
 8499 *interfaces.*

8501 **Enhanced the Dump command to display the entire contents of Package**

8502 **objects**  
 4788 *Enhanced the Dump command to display the entire contents of Package objects*  
 8503 (including all sub-objects and their values.)

8505 Restructured the code base to split some files because of size and/or  
 8506 **because the code logically belonged in a separate file. New files are**  
 8507 **listed**  
 8508 **below. All makefiles and project files included in the ACPI CA release**  
 8509 **have**  
 4792 *because the code logically belonged in a separate file. New files are listed*  
 4793 *below. All makefiles and project files included in the ACPI CA release have*  
 8510 been updated.

8511 utilities/utcache.c /\* Local cache interfaces \*/  
 8512 utilities/utmutex.c /\* Local mutex support \*/  
 8513 utilities/utstate.c /\* State object support \*/  
 8514 interpreter/parser/psloop.c /\* Main AML parse loop \*/

8516 Code and Data Size: Current and previous core subsystem library sizes are  
 8517 **shown below. These are the code and data sizes for the acpica.lib**  
 8518 **produced**  
 4801 *shown below. These are the code and data sizes for the acpica.lib produced*  
 8519 by the Microsoft Visual C++ 6.0 compiler, and these values do not include  
 8520 any ACPI driver or OSPM code. The debug version of the code includes the  
 8521 debug output trace mechanism and has a much larger code and data size.  
 8522 **Note**  
 8523 **that these values will vary depending on the efficiency of the compiler**  
 8524 **and**  
 4804 *debug output trace mechanism and has a much larger code and data size. Note*  
 4805 *that these values will vary depending on the efficiency of the compiler and*  
 8525 the compiler options used during generation.

8527 Previous Release:  
 8528 Non-Debug Version: 78.3K Code, 11.6K Data, 89.9K Total  
 8529 Debug Version: 164.0K Code, 69.1K Data, 233.1K Total  
 8530 Current Release:  
 8531 Non-Debug Version: 78.6K Code, 11.5K Data, 90.1K Total  
 8532 Debug Version: 165.2K Code, 69.6K Data, 234.8K Total

8535 2) iASL Compiler/Disassembler:

8537 **Fixed a regression introduced in version 20050513 where the use of a**  
 8538 **Package**  
 4818 *Fixed a regression introduced in version 20050513 where the use of a Package*  
 8539 object within a Case() statement caused a compile time exception. The  
 8540 original behavior has been restored (a Match() operator is emitted.)

8542 -----  
 8543 17 June 2005. Summary of changes for version 20050617:

8545 1) ACPI CA Core Subsystem:

8547 **Moved the object cache operations into the OS interface layer (OSL) to**  
 8548 **allow**  
 4827 *Moved the object cache operations into the OS interface layer (OSL) to allow*  
 8549 the host OS to handle these operations if desired (for example, the Linux  
 8550 **OSL will invoke the slab allocator). This support is optional; the**  
 8551 **compile**  
 8552 **time define ACPI\_USE\_LOCAL\_CACHE may be used to utilize the original**  
 8553 **cache**  
 4829 *OSL will invoke the slab allocator). This support is optional; the compile*  
 4830 *time define ACPI\_USE\_LOCAL\_CACHE may be used to utilize the original cache*  
 8554 code in the ACPI CA core. The new OSL interfaces are shown below. See  
 8555 utalloc.c for an example implementation, and acpiosxf.h for the exact  
 8556 interface definitions. With assistance from Alexey Starikovskiy.  
 8557 AcpiOsCreateCache

8558 AcpiOsDeleteCache  
 8559 AcpiOsPurgeCache  
 8560 AcpiOsAcquireObject  
 8561 AcpiOsReleaseObject

8563 **Modified the interfaces to AcpiOsAcquireLock and AcpiOsReleaseLock to**  
 8564 **return**  
 4840 *Modified the interfaces to AcpiOsAcquireLock and AcpiOsReleaseLock to return*  
 8565 and restore a flags parameter. This fits better with many OS lock models.  
 8566 Note: the current execution state (interrupt handler or not) is no longer  
 8567 **passed to these interfaces. If necessary, the OSL must determine this**  
 8568 **state**  
 4843 *passed to these interfaces. If necessary, the OSL must determine this state*  
 8569 by itself, a simple and fast operation. With assistance from Alexey  
 8570 Starikovskiy.

8572 Fixed a problem in the ACPI table handling where a valid XSDT was assumed  
 8573 **present if the revision of the RSDP was 2 or greater. According to the**  
 8574 **ACPI**  
 4848 *present if the revision of the RSDP was 2 or greater. According to the ACPI*  
 8575 specification, the XSDT is optional in all cases, and the table manager  
 8576 therefore now checks for both an RSDP >=2 and a valid XSDT pointer.  
 8577 **Otherwise, the RSDT pointer is used. Some ACPI 2.0 compliant BIOSs**  
 8578 **contain**  
 4851 *Otherwise, the RSDT pointer is used. Some ACPI 2.0 compliant BIOSs contain*  
 8579 only the RSDT.

8581 **Fixed an interpreter problem with the Mid() operator in the case of an**  
 8582 **input**  
 8583 **string where the resulting output string is of zero length. It now**  
 8584 **correctly**  
 4854 *Fixed an interpreter problem with the Mid() operator in the case of an input*  
 4855 *string where the resulting output string is of zero length. It now correctly*  
 8585 returns a valid, null terminated string object instead of a string object  
 8586 with a null pointer.

8588 **Fixed a problem with the control method argument handling to allow a**  
 8589 **store**  
 8590 **to an Arg object that already contains an object of type Device. The**  
 8591 **Device**  
 4859 *Fixed a problem with the control method argument handling to allow a store*  
 4860 *to an Arg object that already contains an object of type Device. The Device*  
 8592 object is now correctly overwritten. Previously, an error was returned.

8595 **Enhanced the debugger Find command to emit object values in addition to**  
 8596 **the**  
 8597 **found object pathnames. The output format is the same as the dump**  
 8598 **namespace**  
 4864 *Enhanced the debugger Find command to emit object values in addition to the*  
 4865 *found object pathnames. The output format is the same as the dump namespace*  
 8599 command.

8601 **Enhanced the debugger Set command. It now has the ability to set the**  
 8602 **value**  
 8603 **of any Named integer object in the namespace (Previously, only method**  
 8604 **locals**  
 4868 *Enhanced the debugger Set command. It now has the ability to set the value*  
 4869 *of any Named integer object in the namespace (Previously, only method locals*  
 8605 and args could be set.)

8607 Code and Data Size: Current and previous core subsystem library sizes are  
 8608 **shown below. These are the code and data sizes for the acpica.lib**  
 8609 **produced**  
 4873 *shown below. These are the code and data sizes for the acpica.lib produced*  
 8610 by the Microsoft Visual C++ 6.0 compiler, and these values do not include

8611 any ACPI driver or OSPM code. The debug version of the code includes the  
 8612 **debug output trace mechanism and has a much larger code and data size.**  
 8613 **Note**  
 8614 **that these values will vary depending on the efficiency of the compiler**  
 8615 **and**  
 8616 **debug output trace mechanism and has a much larger code and data size. Note**  
 8617 **that these values will vary depending on the efficiency of the compiler and**  
 8618 **the compiler options used during generation.**

8618	Previous Release:	
8619	Non-Debug Version:	78.1K Code, 11.6K Data, 89.7K Total
8620	Debug Version:	164.0K Code, 69.3K Data, 233.3K Total
8621	Current Release:	
8622	Non-Debug Version:	78.3K Code, 11.6K Data, 89.9K Total
8623	Debug Version:	164.0K Code, 69.1K Data, 233.1K Total

8626 2) iASL Compiler/Disassembler:

8628 **Fixed a regression in the disassembler where if/else/while constructs**  
 8629 **were**  
 8630 **Fixed a regression in the disassembler where if/else/while constructs were**  
 8631 **output incorrectly. This problem was introduced in the previous release**  
 8632 **(20050526). This problem also affected the single-step disassembly in the**  
 8633 **debugger.**

8634 **Fixed a problem where compiling the reserved \_OSI method would randomly**  
 8635 **(but**  
 8636 **Fixed a problem where compiling the reserved \_OSI method would randomly (but**  
 8637 **rarely) produce compile errors.**

8638 **Enhanced the disassembler to emit compilable code in the face of**  
 8639 **incorrect**  
 8640 **Enhanced the disassembler to emit compilable code in the face of incorrect**  
 8641 **AML resource descriptors. If the optional ResourceSourceIndex is present,**  
 8642 **but the ResourceSource is not, do not emit the ResourceSourceIndex in the**  
 8643 **disassembly. Otherwise, the resulting code cannot be compiled without**  
 8644 **errors.**

8645 -----  
 8646 26 May 2005. Summary of changes for version 20050526:

8648 1) ACPI CA Core Subsystem:

8650 Implemented support to execute Type 1 and Type 2 AML opcodes appearing at  
 8651 **the module level (not within a control method.) These opcodes are**  
 8652 **executed**  
 8653 **exactly once at the time the table is loaded. This type of code was legal**  
 8654 **up**  
 8655 **until the release of ACPI 2.0B (2002) and is now supported within ACPI CA**  
 8656 **in**  
 8657 **order to provide backwards compatibility with earlier BIOS**  
 8658 **implementations.**  
 8659 **the module level (not within a control method.) These opcodes are executed**  
 8660 **exactly once at the time the table is loaded. This type of code was legal up**  
 8661 **until the release of ACPI 2.0B (2002) and is now supported within ACPI CA in**  
 8662 **order to provide backwards compatibility with earlier BIOS implementations.**  
 8663 **This eliminates the "Encountered executable code at module level" warning**  
 8664 **that was previously generated upon detection of such code.**

8662 Fixed a problem in the interpreter where an AE\_NOT\_FOUND exception could  
 8663 inadvertently be generated during the lookup of namespace objects in the  
 8664 **second pass parse of ACPI tables and control methods. It appears that**  
 8665 **this**  
 8666 **problem could occur during the resolution of forward references to**  
 8667 **namespace**

4919 **second pass parse of ACPI tables and control methods. It appears that this**  
 4920 **problem could occur during the resolution of forward references to namespace**  
 8668 **objects.**

8670 Added the ACPI\_MUTEX\_DEBUG #ifdef to the AcpiUtReleaseMutex function,  
 8671 corresponding to the same #ifdef in the AcpiUtAcquireMutex function. This  
 8672 allows the deadlock detection debug code to be compiled out in the normal  
 8673 case, improving mutex performance (and overall subsystem performance)  
 8674 considerably.

8676 Implemented a handful of miscellaneous fixes for possible memory leaks on  
 8677 error conditions and error handling control paths. These fixes were  
 8678 suggested by FreeBSD and the Coverity Prevent source code analysis tool.

8680 **Added a check for a null RSDT pointer in AcpiGetFirmwareTable**  
 8681 **(tbxfroot.c)**  
 4933 **Added a check for a null RSDT pointer in AcpiGetFirmwareTable (tbxfroot.c)**  
 8682 **to prevent a fault in this error case.**

8684 Code and Data Size: Current and previous core subsystem library sizes are  
 8685 **shown below. These are the code and data sizes for the acpica.lib**  
 8686 **produced**  
 4937 **shown below. These are the code and data sizes for the acpica.lib produced**  
 8687 **by the Microsoft Visual C++ 6.0 compiler, and these values do not include**  
 8688 **any ACPI driver or OSPM code. The debug version of the code includes the**  
 8689 **debug output trace mechanism and has a much larger code and data size.**  
 8690 **Note**  
 8691 **that these values will vary depending on the efficiency of the compiler**  
 8692 **and**  
 4940 **debug output trace mechanism and has a much larger code and data size. Note**  
 4941 **that these values will vary depending on the efficiency of the compiler and**  
 8693 **the compiler options used during generation.**

8695	Previous Release:	
8696	Non-Debug Version:	78.2K Code, 11.6K Data, 89.8K Total
8697	Debug Version:	163.7K Code, 69.3K Data, 233.0K Total
8698	Current Release:	
8699	Non-Debug Version:	78.1K Code, 11.6K Data, 89.7K Total
8700	Debug Version:	164.0K Code, 69.3K Data, 233.3K Total

8703 2) iASL Compiler/Disassembler:

8705 Implemented support to allow Type 1 and Type 2 ASL operators to appear at  
 8706 the module level (not within a control method.) These operators will be  
 8707 **executed once at the time the table is loaded. This type of code was**  
 8708 **legal**  
 4956 **executed once at the time the table is loaded. This type of code was legal**  
 8709 **up until the release of ACPI 2.0B (2002) and is now supported by the iASL**  
 8710 **compiler in order to provide backwards compatibility with earlier BIOS**  
 8711 **ASL**  
 4958 **compiler in order to provide backwards compatibility with earlier BIOS ASL**  
 8712 **code.**

8714 The ACPI integer width (specified via the table revision ID or the -r  
 8715 **override, 32 or 64 bits) is now used internally during compile-time**  
 8716 **constant**  
 4962 **override, 32 or 64 bits) is now used internally during compile-time constant**  
 8717 **folding to ensure that constants are truncated to 32 bits if necessary.**  
 8718 **Previously, the revision ID value was only emitted in the AML table**  
 8719 **header.**  
 4964 **Previously, the revision ID value was only emitted in the AML table header.**

8721 **An error message is now generated for the Mutex and Method operators if**  
 8722 **the**  
 4966 **An error message is now generated for the Mutex and Method operators if the**

8723 SyncLevel parameter is outside the legal range of 0 through 15.

8725 **Fixed a problem with the Method operator ParameterTypes list handling**  
 8726 **(ACPI**  
 8727 **3.0). Previously, more than 2 types or 2 arguments generated a syntax**  
 8728 **error.**  
 4969 *Fixed a problem with the Method operator ParameterTypes list handling (ACPI*  
 4970 *3.0). Previously, more than 2 types or 2 arguments generated a syntax error.*  
 8729 The actual underlying implementation of method argument typechecking is  
 8730 still under development, however.

8732 -----  
 8733 13 May 2005. Summary of changes for version 20050513:

8735 1) ACPI CA Core Subsystem:

8737 **Implemented support for PCI Express root bridges -- added support for**  
 8738 **device**  
 4979 *Implemented support for PCI Express root bridges -- added support for device*  
 8739 *PNP0A08 in the root bridge search within AcpiEvPciConfigRegionSetup.*

8741 **The interpreter now automatically truncates incoming 64-bit constants to**  
 8742 **32**  
 8743 **bits if currently executing out of a 32-bit ACPI table (Revision < 2).**  
 8744 **This**  
 4982 *The interpreter now automatically truncates incoming 64-bit constants to 32*  
 4983 *bits if currently executing out of a 32-bit ACPI table (Revision < 2). This*  
 8745 *also affects the iASL compiler constant folding. (Note: as per below, the*  
 8746 *iASL compiler no longer allows 64-bit constants within 32-bit tables.)*

8748 Fixed a problem where string and buffer objects with "static" pointers  
 8749 (pointers to initialization data within an ACPI table) were not handled  
 8750 **consistently. The internal object copy operation now always copies the**  
 8751 **data**  
 4989 *consistently. The internal object copy operation now always copies the data*  
 8752 *to a newly allocated buffer, regardless of whether the source object is*  
 8753 *static or not.*

8755 Fixed a problem with the FromBCD operator where an implicit result  
 8756 **conversion was improperly performed while storing the result to the**  
 8757 **target**  
 4994 *conversion was improperly performed while storing the result to the target*  
 8758 *operand. Since this is an "explicit conversion" operator, the implicit*  
 8759 *conversion should never be performed on the output.*

8761 Fixed a problem with the CopyObject operator where a copy to an existing  
 8762 **named object did not always completely overwrite the existing object**  
 8763 **stored**  
 8764 **at name. Specifically, a buffer-to-buffer copy did not delete the**  
 8765 **existing**  
 4999 *named object did not always completely overwrite the existing object stored*  
 5000 *at name. Specifically, a buffer-to-buffer copy did not delete the existing*  
 8766 *buffer.*

8768 **Replaced "InterruptLevel" with "InterruptNumber" in all GPE interfaces**  
 8769 **and**  
 5003 *Replaced "InterruptLevel" with "InterruptNumber" in all GPE interfaces and*  
 8770 *structs for consistency.*

8772 Code and Data Size: Current and previous core subsystem library sizes are  
 8773 **shown below. These are the code and data sizes for the acpica.lib**  
 8774 **produced**  
 5007 *shown below. These are the code and data sizes for the acpica.lib produced*  
 8775 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 8776 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 8777 *debug output trace mechanism and has a much larger code and data size.*

8778 **Note**  
 8779 **that these values will vary depending on the efficiency of the compiler**  
 8780 **and**  
 5010 *debug output trace mechanism and has a much larger code and data size. Note*  
 5011 *that these values will vary depending on the efficiency of the compiler and*  
 8781 *the compiler options used during generation.*

8783 Previous 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  
 8786 Current Release: (Same sizes)  
 8787 Non-Debug Version: 78.2K Code, 11.6K Data, 89.8K Total  
 8788 Debug Version: 163.7K Code, 69.3K Data, 233.0K Total

8791 2) iASL Compiler/Disassembler:

8793 **The compiler now emits a warning if an attempt is made to generate a 64-**  
 8794 **bit**  
 8795 **integer constant from within a 32-bit ACPI table (Revision < 2). The**  
 8796 **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*  
 8797 *is truncated to 32 bits.*

8799 Fixed a problem with large package objects: if the static length of the  
 8800 package is greater than 255, the "variable length package" opcode is  
 8801 emitted. Previously, this caused an error. This requires an update to the  
 8802 **ACPI spec, since it currently (incorrectly) states that packages larger**  
 8803 **than**  
 5031 *ACPI spec, since it currently (incorrectly) states that packages larger than*  
 8804 *255 elements are not allowed.*

8806 **The disassembler now correctly handles variable length packages and**  
 8807 **packages**  
 5034 *The disassembler now correctly handles variable length packages and packages*  
 8808 *larger than 255 elements.*

8810 -----  
 8811 08 April 2005. Summary of changes for version 20050408:

8813 1) ACPI CA Core Subsystem:

8815 Fixed three cases in the interpreter where an "index" argument to an ASL  
 8816 function was still (internally) 32 bits instead of the required 64 bits.  
 8817 This was the Index argument to the Index, Mid, and Match operators.

8819 **The "strupr" function is now permanently local (AcpiUtStrupr), since this**  
 8820 **is**  
 5046 *The "strupr" function is now permanently local (AcpiUtStrupr), since this is*  
 8821 *not a POSIX-defined function and not present in most kernel-level C*  
 8822 *libraries. All references to the C library strupr function have been*  
 8823 **removed**  
 5048 *libraries. All references to the C library strupr function have been removed*  
 8824 *from the headers.*

8826 **Completed the deployment of static functions/prototypes. All prototypes**  
 8827 **with**  
 8828 **the static attribute have been moved from the headers to the owning C**  
 8829 **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.*

8831 Implemented an extract option (-e) for the AcpiBin utility (AML binary  
 8832 **utility). This option allows the utility to extract individual ACPI**  
 8833 **tables**



5055 utility). This option allows the utility to extract individual ACPI tables  
8834 from the output of AcpiDmp. It provides the same functionality of the  
8835 acpixtract.pl perl script without the worry of setting the correct perl  
8836 options. AcpiBin runs on Windows and has not yet been generated/validated  
8837 in  
5058 options. AcpiBin runs on Windows and has not yet been generated/validated in  
8838 the Linux/Unix environment (but should be soon).  
8839  
8840 Updated and fixed the table dump option for AcpiBin (-d). This option  
8841 converts a single ACPI table to a hex/ascii file, similar to the output  
8842 of  
5062 converts a single ACPI table to a hex/ascii file, similar to the output of  
8843 AcpiDmp.

8845 Code and Data Size: Current and previous core subsystem library sizes are  
8846 shown below. These are the code and data sizes for the acpica.lib  
8847 produced  
5066 shown below. These are the code and data sizes for the acpica.lib produced  
8848 by the Microsoft Visual C++ 6.0 compiler, and these values do not include  
8849 any ACPI driver or OSPM code. The debug version of the code includes the  
8850 debug output trace mechanism and has a much larger code and data size.  
8851 Note  
8852 that these values will vary depending on the efficiency of the compiler  
8853 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  
8854 the compiler options used during generation.

8856	Previous Release:			
8857	Non-Debug Version:	78.0K Code,	11.6K Data,	89.6K Total
8858	Debug Version:	163.5K Code,	69.3K Data,	232.8K Total
8859	Current Release:			
8860	Non-Debug Version:	78.2K Code,	11.6K Data,	89.8K Total
8861	Debug Version:	163.7K Code,	69.3K Data,	233.0K Total

8864 2) iASL Compiler/Disassembler:

8866 Disassembler fix: Added a check to ensure that the table length found in  
8867 the  
8868 ACPI table header within the input file is not longer than the actual  
8869 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  
8870 file size. This indicates some kind of file or table corruption.

8872 -----  
8873 29 March 2005. Summary of changes for version 20050329:

8875 1) ACPI CA Core Subsystem:

8877 An error is now generated if an attempt is made to create a Buffer Field  
8878 of  
5092 An error is now generated if an attempt is made to create a Buffer Field of  
8879 length zero (A CreateField with a length operand of zero.)

8881 The interpreter now issues a warning whenever executable code at the  
8882 module  
5095 The interpreter now issues a warning whenever executable code at the module  
8883 level is detected during ACPI table load. This will give some idea of the  
8884 prevalence of this type of code.

8886 Implemented support for references to named objects (other than control  
8887 methods) within package objects.

8889 Enhanced package object output for the debug object. Package objects are

8890 now  
5102 Enhanced package object output for the debug object. Package objects are now  
8891 completely dumped, showing all elements.

8893 Enhanced miscellaneous object output for the debug object. Any object can  
8894 now be written to the debug object (for example, a device object can be  
8895 written, and the type of the object will be displayed.)

8897 The "static" qualifier has been added to all local functions across both  
8898 the  
5109 The "static" qualifier has been added to all local functions across both the  
8899 core subsystem and the iASL compiler.

8901 The number of "long" lines (> 80 chars) within the source has been  
8902 significantly reduced, by about 1/3.

8904 Cleaned up all header files to ensure that all CA/iASL functions are  
8905 prototyped (even static functions) and the formatting is consistent.

8907 Two new header files have been added, acopcode.h and acnames.h.

8909 Removed several obsolete functions that were no longer used.

8911 Code and Data Size: Current and previous core subsystem library sizes are  
8912 shown below. These are the code and data sizes for the acpica.lib  
8913 produced  
5123 shown below. These are the code and data sizes for the acpica.lib produced  
8914 by the Microsoft Visual C++ 6.0 compiler, and these values do not include  
8915 any ACPI driver or OSPM code. The debug version of the code includes the  
8916 debug output trace mechanism and has a much larger code and data size.  
8917 Note  
8918 that these values will vary depending on the efficiency of the compiler  
8919 and  
5126 debug output trace mechanism and has a much larger code and data size. Note  
5127 that these values will vary depending on the efficiency of the compiler and  
8920 the compiler options used during generation.

8922	Previous Release:			
8923	Non-Debug Version:	78.3K Code,	11.5K Data,	89.8K Total
8924	Debug Version:	165.4K Code,	69.7K Data,	236.1K Total
8925	Current Release:			
8926	Non-Debug Version:	78.0K Code,	11.6K Data,	89.6K Total
8927	Debug Version:	163.5K Code,	69.3K Data,	232.8K Total

8931 2) iASL Compiler/Disassembler:

8933 Fixed a problem with the resource descriptor generation/support. For the  
8934 ResourceSourceIndex and the ResourceSource fields, both must be present,  
8935 or  
5142 ResourceSourceIndex and the ResourceSource fields, both must be present, or  
8936 both must be not present - can't have one without the other.

8938 The compiler now returns non-zero from the main procedure if any errors  
8939 have  
5145 The compiler now returns non-zero from the main procedure if any errors have  
8940 occurred during the compilation.

8943 -----  
8944 09 March 2005. Summary of changes for version 20050309:

8946 1) ACPI CA Core Subsystem:

8948 The string-to-buffer implicit conversion code has been modified again

8949 after  
 8950 a change to the ACPI specification. In order to match the behavior of  
 8951 the  
 8952 other major ACPI implementation, the target buffer is no longer truncated  
 8953 if  
 8954 The string-to-buffer implicit conversion code has been modified again after  
 8955 a change to the ACPI specification. In order to match the behavior of the  
 8956 other major ACPI implementation, the target buffer is no longer truncated if  
 8957 the source string is smaller than an existing target buffer. This change  
 8958 requires an update to the ACPI spec, and should eliminate the recent  
 8959 AE\_AML\_BUFFER\_LIMIT issues.

8958 The "implicit return" support was rewritten to a new algorithm that  
 8959 solves  
 8960 the general case. Rather than attempt to determine when a method is about  
 8961 to  
 8962 exit, the result of every ASL operator is saved momentarily until the  
 8963 very  
 8964 The "implicit return" support was rewritten to a new algorithm that solves  
 8965 the general case. Rather than attempt to determine when a method is about to  
 8966 exit, the result of every ASL operator is saved momentarily until the very  
 8967 next ASL operator is executed. Therefore, no matter how the method exits,  
 8968 there will always be a saved implicit return value. This feature is only  
 8969 enabled with the AcpiGbl\_EnableInterpreterSlack flag, and should  
 8970 eliminate  
 8971 enabled with the AcpiGbl\_EnableInterpreterSlack flag, and should eliminate  
 8972 AE\_AML\_NO\_RETURN\_VALUE errors when enabled.

8970 Implemented implicit conversion support for the predicate (operand) of  
 8971 the  
 8972 If, Else, and While operators. String and Buffer arguments are  
 8973 automatically  
 8974 Implemented implicit conversion support for the predicate (operand) of the  
 8975 If, Else, and While operators. String and Buffer arguments are automatically  
 8976 converted to Integers.

8976 Changed the string-to-integer conversion behavior to match the new ACPI  
 8977 errata: "If no integer object exists, a new integer is created. The ASCII  
 8978 string is interpreted as a hexadecimal constant. Each string character is  
 8979 interpreted as a hexadecimal value ('0'-'9', 'A'-'F', 'a', 'f'), starting  
 8980 with the first character as the most significant digit, and ending with  
 8981 the  
 8982 first non-hexadecimal character or end-of-string." This means that the  
 8983 first  
 8984 with the first character as the most significant digit, and ending with the  
 8985 first non-hexadecimal character or end-of-string." This means that the first  
 8986 non-hex character terminates the conversion and this is the code that was  
 8987 changed.

8987 Fixed a problem where the ObjectType operator would fail (fault) when  
 8988 used  
 8989 Fixed a problem where the ObjectType operator would fail (fault) when used  
 8990 on an Index of a Package which pointed to a null package element. The  
 8991 operator now properly returns zero (Uninitialized) in this case.

8992 Fixed a problem where the While operator used excessive memory by not  
 8993 properly popping the result stack during execution. There was no memory  
 8994 leak  
 8995 properly popping the result stack during execution. There was no memory leak  
 8996 after execution, however. (Code provided by Valery Podrezov.)

8997 Fixed a problem where references to control methods within Package  
 8998 objects  
 8999 Fixed a problem where references to control methods within Package objects  
 9000 caused the method to be invoked, instead of producing a reference object  
 9001 pointing to the method.

9002 Restructured and simplified the pswalk.c module (AcpiPsDeleteParseTree)  
 9003 to  
 9004 Restructured and simplified the pswalk.c module (AcpiPsDeleteParseTree) to  
 9005 improve performance and reduce code size. (Code provided by Alexey  
 9006 Starikovskiy.)

9007 Code and Data Size: Current and previous core subsystem library sizes are  
 9008 shown below. These are the code and data sizes for the acpica.lib  
 9009 produced  
 9010 shown below. These are the code and data sizes for the acpica.lib produced  
 9011 by the Microsoft Visual C++ 6.0 compiler, and these values do not include  
 9012 any ACPI driver or OSPM code. The debug version of the code includes the  
 9013 debug output trace mechanism and has a much larger code and data size.  
 9014 Note  
 9015 that these values will vary depending on the efficiency of the compiler  
 9016 and  
 9017 debug output trace mechanism and has a much larger code and data size. Note  
 9018 that these values will vary depending on the efficiency of the compiler and  
 9019 the compiler options used during generation.

9018	Previous Release:	
9019	Non-Debug Version:	78.3K Code, 11.5K Data, 89.8K Total
9020	Debug Version:	165.4K Code, 69.6K Data, 236.0K Total
9021	Current Release:	
9022	Non-Debug Version:	78.3K Code, 11.5K Data, 89.8K Total
9023	Debug Version:	165.4K Code, 69.7K Data, 236.1K Total

9026 2) iASL Compiler/Disassembler:

9028 Fixed a problem with the Return operator with no arguments. Since the AML  
 9029 grammar for the byte encoding requires an operand for the Return opcode,  
 9030 the  
 9031 grammar for the byte encoding requires an operand for the Return opcode, the  
 9032 compiler now emits a Return(Zero) for this case. An ACPI specification  
 9033 update has been written for this case.

9034 For tables other than the DSDT, namespace optimization is automatically  
 9035 disabled. This is because SSDTs can be loaded anywhere in the namespace,  
 9036 the  
 9037 disabled. This is because SSDTs can be loaded anywhere in the namespace, the  
 9038 compiler has no knowledge of where, and thus cannot optimize namepaths.

9039 Added "ProcessorObj" to the ObjectTypeKeyword list. This object type was  
 9040 inadvertently omitted from the ACPI specification, and will require an  
 9041 update to the spec.

9043 The source file scan for ASCII characters is now optional (-a). This  
 9044 change  
 9045 The source file scan for ASCII characters is now optional (-a). This change  
 9046 was made because some vendors place non-ascii characters within comments.  
 9047 However, the scan is simply a brute-force byte compare to ensure all  
 9048 characters in the file are in the range 0x00 to 0x7F.

9049 Fixed a problem with the CondRefOf operator where the compiler was  
 9050 inappropriately checking for the existence of the target. Since the point  
 9051 of  
 9052 inappropriately checking for the existence of the target. Since the point of  
 9053 the operator is to check for the existence of the target at run-time, the  
 9054 compiler no longer checks for the target existence.

9055 Fixed a problem where errors generated from the internal AML interpreter  
 9056 during constant folding were not handled properly, causing a fault.

9058 Fixed a problem with overly aggressive range checking for the Stall

9059 **operator. The valid range (max 255) is now only checked if the operand is**  
 9060 **of**  
 5243 *operator. The valid range (max 255) is now only checked if the operand is of*  
 9061 *type Integer. All other operand types cannot be statically checked.*

9063 **Fixed a problem where control method references within the RefOf,**  
 9064 **DeRefOf,**  
 9065 **and ObjectType operators were not treated properly. They are now treated**  
 9066 **as**  
 5246 *Fixed a problem where control method references within the RefOf, DeRefOf,*  
 5247 *and ObjectType operators were not treated properly. They are now treated as*  
 9067 *actual references, not method invocations.*

9069 **Fixed and enhanced the "list namespace" option (-ln). This option was**  
 9070 **broken**  
 5250 *Fixed and enhanced the "list namespace" option (-ln). This option was broken*  
 9071 *a number of releases ago.*

9073 **Improved error handling for the Field, IndexField, and BankField**  
 9074 **operators.**  
 5253 *Improved error handling for the Field, IndexField, and BankField operators.*  
 9075 *The compiler now cleanly reports and recovers from errors in the field*  
 9076 *component (FieldUnit) list.*

9078 **Fixed a disassembler problem where the optional ResourceDescriptor fields**  
 9079 **TRS and TTP were not always handled correctly.**

9081 **Disassembler - Comments in output now use "/" instead of "/\*"**

9083 -----  
 9084 28 February 2005. Summary of changes for version 20050228:

9086 1) ACPI CA Core Subsystem:

9088 **Fixed a problem where the result of an Index() operator (an object**  
 9089 **reference) must increment the reference count on the target object for**  
 9090 **the**  
 5268 *reference) must increment the reference count on the target object for the*  
 9091 *life of the object reference.*

9093 **Implemented AML Interpreter and Debugger support for the new ACPI 3.0**  
 9094 **Extended Address (IO, Memory, Space), QwordSpace, DwordSpace, and**  
 9095 **WordSpace**  
 5272 *Extended Address (IO, Memory, Space), QwordSpace, DwordSpace, and WordSpace*  
 9096 *resource descriptors.*

9098 **Implemented support in the \_OSI method for the ACPI 3.0 "Extended Address**  
 9099 **Space Descriptor" string, indicating interpreter support for the**  
 9100 **descriptors**  
 5276 *Space Descriptor" string, indicating interpreter support for the descriptors*  
 9101 *above.*

9103 **Implemented header support for the new ACPI 3.0 FADT flag bits.**

9105 **Implemented header support for the new ACPI 3.0 PCI Express bits for the**  
 9106 **PM1**  
 5281 *Implemented header support for the new ACPI 3.0 PCI Express bits for the PM1*  
 9107 *status/enable registers.*

9109 **Updated header support for the MADT processor local Apic struct and MADT**  
 9110 **platform interrupt source struct for new ACPI 3.0 fields.**

9112 **Implemented header support for the SRAT and SLIT ACPI tables.**

9114 **Implemented the -s switch in AcpiExec to enable the "InterpreterSlack"**  
 9115 **flag**

5289 **Implemented the -s switch in AcpiExec to enable the "InterpreterSlack" flag**  
 9116 **at runtime.**

9118 **Code and Data Size: Current and previous core subsystem library sizes are**  
 9119 **shown below. These are the code and data sizes for the acpica.lib**  
 9120 **produced**  
 5293 *shown below. These are the code and data sizes for the acpica.lib produced*  
 9121 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 9122 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 9123 *debug output trace mechanism and has a much larger code and data size.*  
 9124 **Note**  
 9125 **that these values will vary depending on the efficiency of the compiler**  
 9126 **and**  
 5296 *debug output trace mechanism and has a much larger code and data size. Note*  
 5297 *that these values will vary depending on the efficiency of the compiler and*  
 9127 *the compiler options used during generation.*

9129	Previous Release:	
9130	Non-Debug Version:	78.2K Code, 11.5K Data, 89.7K Total
9131	Debug Version:	164.9K Code, 69.2K Data, 234.1K Total
9132	Current Release:	
9133	Non-Debug Version:	78.3K Code, 11.5K Data, 89.8K Total
9134	Debug Version:	165.4K Code, 69.6K Data, 236.0K Total

9137 2) iASL Compiler/Disassembler:

9139 **Fixed a problem with the internal 64-bit String-to-integer conversion**  
 9140 **with**  
 5310 *Fixed a problem with the internal 64-bit String-to-integer conversion with*  
 9141 *strings less than two characters long.*

9143 **Fixed a problem with constant folding where the result of the Index()**  
 9144 **operator can not be considered a constant. This means that Index() cannot**  
 9145 **be**  
 5314 *operator can not be considered a constant. This means that Index() cannot be*  
 9146 *a type3 opcode and this will require an update to the ACPI specification.*

9148 **Disassembler: Implemented support for the TTP, MTP, and TRS resource**  
 9149 **descriptor fields. These fields were inadvertently ignored and not output**  
 9150 **in**  
 5318 *descriptor fields. These fields were inadvertently ignored and not output in*  
 9151 *the disassembly of the resource descriptor.*

9154 -----  
 9155 11 February 2005. Summary of changes for version 20050211:

9157 1) ACPI CA Core Subsystem:

9159 **Implemented ACPI 3.0 support for implicit conversion within the Match()**  
 9160 **operator. MatchObjects can now be of type integer, buffer, or string**  
 9161 **instead**  
 9162 **of just type integer. Package elements are implicitly converted to the**  
 9163 **type**  
 5328 *operator. MatchObjects can now be of type integer, buffer, or string instead*  
 5329 *of just type integer. Package elements are implicitly converted to the type*  
 9164 *of the MatchObject. This change aligns the behavior of Match() with the*  
 9165 *behavior of the other logical operators (LLess(), etc.) It also requires*  
 9166 **an**  
 5331 *behavior of the other logical operators (LLess(), etc.) It also requires an*  
 9167 *errata change to the ACPI specification as this support was intended for*  
 9168 *ACPI 3.0, but was inadvertently omitted.*

9170 **Fixed a problem with the internal implicit "to buffer" conversion.**  
 9171 **Strings**

9172 that are converted to buffers will cause buffer truncation if the string  
 9173 is  
 9174 smaller than the target buffer. Integers that are converted to buffers  
 9175 will  
 5335 Fixed a problem with the internal implicit "to buffer" conversion. Strings  
 5336 that are converted to buffers will cause buffer truncation if the string is  
 5337 smaller than the target buffer. Integers that are converted to buffers will  
 9176 not cause buffer truncation, only zero extension (both as per the ACPI  
 9177 spec.) The problem was introduced when code was added to truncate the  
 9178 buffer, but this should not be performed in all cases, only the string  
 9179 case.  
 5340 buffer, but this should not be performed in all cases, only the string case.

9181 Fixed a problem with the Buffer and Package operators where the  
 9182 interpreter  
 5342 Fixed a problem with the Buffer and Package operators where the interpreter  
 9183 would get confused if two such operators were used as operands to an ASL  
 9184 operator (such as Lless(Buffer(1){0},Buffer(1){1}). The internal result  
 9185 stack was not being popped after the execution of these operators,  
 9186 resulting  
 5345 stack was not being popped after the execution of these operators, resulting  
 9187 in an AE\_NO\_RETURN\_VALUE exception.

9189 Fixed a problem with constructs of the form Store(Index(...),...). The  
 9190 reference object returned from Index was inadvertently resolved to an  
 9191 actual  
 9192 value. This problem was introduced in version 20050114 when the behavior  
 9193 of  
 5349 reference object returned from Index was inadvertently resolved to an actual  
 5350 value. This problem was introduced in version 20050114 when the behavior of  
 9194 Store() was modified to restrict the object types that can be used as the  
 9195 source operand (to match the ACPI specification.)

9197 Reduced excessive stack use within the AcpiGetObjectInfo procedure.

9199 Added a fix to aclinux.h to allow generation of AcpiExec on Linux.

9201 Updated the AcpiSrc utility to add the FADT\_DESCRIPTOR\_REV2\_MINUS struct.

9203 Code and Data Size: Current and previous core subsystem library sizes are  
 9204 shown below. These are the code and data sizes for the acpica.lib  
 9205 produced  
 5361 shown below. These are the code and data sizes for the acpica.lib produced  
 9206 by the Microsoft Visual C++ 6.0 compiler, and these values do not include  
 9207 any ACPI driver or OSPM code. The debug version of the code includes the  
 9208 debug output trace mechanism and has a much larger code and data size.  
 9209 Note  
 9210 that these values will vary depending on the efficiency of the compiler  
 9211 and  
 5364 debug output trace mechanism and has a much larger code and data size. Note  
 5365 that these values will vary depending on the efficiency of the compiler and  
 9212 the compiler options used during generation.

9214	Previous Release:	
9215	Non-Debug Version:	78.1K Code, 11.5K Data, 89.6K Total
9216	Debug Version:	164.8K Code, 69.2K Data, 234.0K Total
9217	Current Release:	
9218	Non-Debug Version:	78.2K Code, 11.5K Data, 89.7K Total
9219	Debug Version:	164.9K Code, 69.2K Data, 234.1K Total

9222 2) iASL Compiler/Disassembler:

9224 Fixed a code generation problem in the constant folding optimization code  
 9225 where incorrect code was generated if a constant was reduced to a buffer  
 9226 object (i.e., a reduced type 5 opcode.)

9228 Fixed a typechecking problem for the ToBuffer operator. Caused by an  
 9229 incorrect return type in the internal opcode information table.

9231 -----  
 9232 25 January 2005. Summary of changes for version 20050125:

9234 1) ACPI CA Core Subsystem:

9236 Fixed a recently introduced problem with the Global Lock where the  
 9237 underlying semaphore was not created. This problem was introduced in  
 9238 version 20050114, and caused an AE\_AML\_NO\_OPERAND exception during an  
 9239 Acquire() operation on \_GL.

9241 The local object cache is now optional, and is disabled by default. Both  
 9242 AcpiExec and the iASL compiler enable the cache because they run in user  
 9243 mode and this enhances their performance. #define  
 9244 ACPI\_ENABLE\_OBJECT\_CACHE  
 5397 mode and this enhances their performance. #define ACPI\_ENABLE\_OBJECT\_CACHE  
 9245 to enable the local cache.

9247 Fixed an issue in the internal function AcpiUtEvaluateObject concerning  
 9248 the  
 9249 optional "implicit return" support where an error was returned if no  
 9250 return  
 9251 object was expected, but one was implicitly returned. AE\_OK is now  
 9252 returned  
 5400 Fixed an issue in the internal function AcpiUtEvaluateObject concerning the  
 5401 optional "implicit return" support where an error was returned if no return  
 5402 object was expected, but one was implicitly returned. AE\_OK is now returned  
 9253 in this case and the implicitly returned object is deleted.  
 9254 AcpiUtEvaluateObject is only occasionally used, and only to execute  
 9255 reserved  
 5404 AcpiUtEvaluateObject is only occasionally used, and only to execute reserved  
 9256 methods such as \_STA and \_INI where the return type is known up front.

9258 Fixed a few issues with the internal convert-to-integer code. It now  
 9259 returns  
 5407 Fixed a few issues with the internal convert-to-integer code. It now returns  
 9260 an error if an attempt is made to convert a null string, a string of only  
 9261 blanks/tabs, or a zero-length buffer. This affects both implicit  
 9262 conversion  
 5409 blanks/tabs, or a zero-length buffer. This affects both implicit conversion  
 9263 and explicit conversion via the ToInteger() operator.

9265 The internal debug code in AcpiUtAcquireMutex has been commented out. It  
 9266 is  
 9267 not needed for normal operation and should increase the performance of  
 9268 the  
 9269 entire subsystem. The code remains in case it is needed for debug  
 9270 purposes  
 5412 The internal debug code in AcpiUtAcquireMutex has been commented out. It is  
 5413 not needed for normal operation and should increase the performance of the  
 5414 entire subsystem. The code remains in case it is needed for debug purposes  
 9271 again.

9273 The AcpiExec source and makefile are included in the Unix/Linux package  
 9274 for  
 5417 The AcpiExec source and makefile are included in the Unix/Linux package for  
 9275 the first time.

9277 Code and Data Size: Current and previous core subsystem library sizes are  
 9278 shown below. These are the code and data sizes for the acpica.lib  
 9279 produced  
 5421 shown below. These are the code and data sizes for the acpica.lib produced  
 9280 by the Microsoft Visual C++ 6.0 compiler, and these values do not include

9281 any ACPI driver or OSPM code. The debug version of the code includes the  
 9282 **debug output trace mechanism and has a much larger code and data size.**  
 9283 **Note**  
 9284 **that these values will vary depending on the efficiency of the compiler**  
 9285 **and**  
 9286 **debug output trace mechanism and has a much larger code and data size. Note**  
 9287 **that these values will vary depending on the efficiency of the compiler and**  
 9288 **the compiler options used during generation.**

9288	Previous Release:			
9289	Non-Debug Version:	78.4K Code,	11.5K Data,	89.9K Total
9290	Debug Version:	165.4K Code,	69.4K Data,	234.8K Total
9291	Current Release:			
9292	Non-Debug Version:	78.1K Code,	11.5K Data,	89.6K Total
9293	Debug Version:	164.8K Code,	69.2K Data,	234.0K Total

9295 2) iASL Compiler/Disassembler:

9297 **Switch/Case support: A warning is now issued if the type of the Switch**  
 9298 **value**  
 9299 *Switch/Case support: A warning is now issued if the type of the Switch value*  
 9300 *cannot be determined at compile time. For example, Switch(Arg0) will*  
 9301 **generate the warning, and the type is assumed to be an integer. As per**  
 9302 **the**  
 9303 **ACPI spec, use a construct such as Switch(ToInteger(Arg0)) to eliminate**  
 9304 **the**  
 9305 *generate the warning, and the type is assumed to be an integer. As per the*  
 9306 *ACPI spec, use a construct such as Switch(ToInteger(Arg0)) to eliminate the*  
 9307 *warning.*

9306 Switch/Case support: Implemented support for buffer and string objects as  
 9307 the switch value. This is an ACPI 3.0 feature, now that LEqual supports  
 9308 buffers and strings.

9310 **Switch/Case support: The emitted code for the LEqual() comparisons now**  
 9311 **uses**  
 9312 **the switch value as the first operand, not the second. The case value is**  
 9313 **now**  
 9314 *Switch/Case support: The emitted code for the LEqual() comparisons now uses*  
 9315 *the switch value as the first operand, not the second. The case value is now*  
 9316 *the second operand, and this allows the case value to be implicitly*  
 9317 *converted to the type of the switch value, not the other way around.*

9317 **Switch/Case support: Temporary variables are now emitted immediately**  
 9318 **within**  
 9319 **the control method, not at the global level. This means that there are**  
 9320 **now**  
 9321 **36 temps available per-method, not 36 temps per-module as was the case**  
 9322 **with**  
 9323 *Switch/Case support: Temporary variables are now emitted immediately within*  
 9324 *the control method, not at the global level. This means that there are now*  
 9325 *36 temps available per-method, not 36 temps per-module as was the case with*  
 9326 *the earlier implementation (\_T\_0 through \_T\_9 and \_T\_A through \_T\_Z.)*

9325 -----  
 9326 14 January 2005. Summary of changes for version 20050114:

9328 Added 2005 copyright to all module headers. This affects every module in  
 9329 the core subsystem, iASL compiler, and the utilities.

9331 1) ACPI CA Core Subsystem:

9333 Fixed an issue with the String-to-Buffer conversion code where the string  
 9334 **null terminator was not included in the buffer after conversion, but**  
 9335 **there**  
 9336 **is existing ASL that assumes the string null terminator is included. This**

9337 **is**  
 9338 *null terminator was not included in the buffer after conversion, but there*  
 9339 *is existing ASL that assumes the string null terminator is included. This is*  
 9340 *the root of the ACPI\_AML\_BUFFER\_LIMIT regression. This problem was*  
 9341 *introduced in the previous version when the code was updated to correctly*  
 9342 **set the converted buffer size as per the ACPI specification. The ACPI**  
 9343 **spec**  
 9344 **is ambiguous and will be updated to specify that the null terminator must**  
 9345 **be**  
 9346 *set the converted buffer size as per the ACPI specification. The ACPI spec*  
 9347 *is ambiguous and will be updated to specify that the null terminator must be*  
 9348 *included in the converted buffer. This also affects the ToBuffer() ASL*  
 9349 *operator.*

9347 Fixed a problem with the Mid() ASL/AML operator where it did not work  
 9348 **correctly on Buffer objects. Newly created sub-buffers were not being**  
 9349 **marked**  
 9350 *correctly on Buffer objects. Newly created sub-buffers were not being marked*  
 9351 *as initialized.*

9353 Fixed a problem in AcpiTbFindTable where incorrect string compares were  
 9354 **performed on the OemId and OemTableId table header fields. These fields**  
 9355 **are**  
 9356 *performed on the OemId and OemTableId table header fields. These fields are*  
 9357 *not null terminated, so strncmp is now used instead of strcmp.*

9358 Implemented a restriction on the Store() ASL/AML operator to align the  
 9359 **behavior with the ACPI specification. Previously, any object could be**  
 9360 **used**  
 9361 **as the source operand. Now, the only objects that may be used are**  
 9362 **Integers,**  
 9363 *behavior with the ACPI specification. Previously, any object could be used*  
 9364 *as the source operand. Now, the only objects that may be used are Integers,*  
 9365 *Buffers, Strings, Packages, Object References, and DDB Handles. If*  
 9366 *necessary, the original behavior can be restored by enabling the*  
 9367 *EnableInterpreterSlack flag.*

9367 **Enhanced the optional "implicit return" support to allow an implicit**  
 9368 **return**  
 9369 *Enhanced the optional "implicit return" support to allow an implicit return*  
 9370 *value from methods that are invoked externally via the AcpiEvaluateObject*  
 9371 *interface. This enables implicit returns from the \_STA and \_INI methods,*  
 9372 *for example.*

9373 **Changed the Revision() ASL/AML operator to return the current version of**  
 9374 **the**  
 9375 **AML interpreter, in the YYYYMMDD format. Previously, it incorrectly**  
 9376 **returned**  
 9377 *Changed the Revision() ASL/AML operator to return the current version of the*  
 9378 *AML interpreter, in the YYYYMMDD format. Previously, it incorrectly returned*  
 9379 *the supported ACPI version (This is the function of the \_REV method).*

9379 **Updated the \_REV predefined method to return the currently supported**  
 9380 **version**  
 9381 *Updated the \_REV predefined method to return the currently supported version*  
 9382 *of ACPI, now 3.*

9383 Implemented batch mode option for the AcpiExec utility (-b).

9385 Code and Data Size: Current and previous core subsystem library sizes are  
 9386 **shown below. These are the code and data sizes for the apica.lib**  
 9387 **produced**  
 9388 *shown below. These are the code and data sizes for the apica.lib produced*  
 9389 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 9390 *any ACPI driver or OSPM code. The debug version of the code includes the*

9390 debug output trace mechanism and has a much larger code and data size.  
 9391 **Note**  
 9392 **that these values will vary depending on the efficiency of the compiler**  
 9393 **and**  
 5509 debug output trace mechanism and has a much larger code and data size. Note  
 5510 that these values will vary depending on the efficiency of the compiler and  
 9394 the compiler options used during generation.

9396	Previous Release:			
9397	Non-Debug Version:	78.3K Code,	11.5K Data,	89.8K Total
9398	Debug Version:	165.3K Code,	69.4K Data,	234.7K Total
9399	Current Release:			
9400	Non-Debug Version:	78.4K Code,	11.5K Data,	89.9K Total
9401	Debug Version:	165.4K Code,	69.4K Data,	234.8K Total

9403 -----  
 9404 10 December 2004. Summary of changes for version 20041210:

9406 ACPI 3.0 support is nearing completion in both the iASL compiler and the  
 9407 ACPI CA core subsystem.

9409 1) ACPI CA Core Subsystem:

9411 **Fixed a problem in the ToDecimalString operator where the resulting**  
 9412 **string**  
 5528 *Fixed a problem in the ToDecimalString operator where the resulting string*  
 9413 *length was incorrectly calculated. The length is now calculated exactly,*  
 9414 *eliminating incorrect AE\_STRING\_LIMIT exceptions.*

9416 **Fixed a problem in the ToHexString operator to allow a maximum 200**  
 9417 **character**  
 5532 *Fixed a problem in the ToHexString operator to allow a maximum 200 character*  
 9418 *string to be produced.*

9420 **Fixed a problem in the internal string-to-buffer and buffer-to-buffer**  
 9421 **copy**  
 5535 *Fixed a problem in the internal string-to-buffer and buffer-to-buffer copy*  
 9422 *routine where the length of the resulting buffer was not truncated to the*  
 9423 *new size (if the target buffer already existed).*

9425 Code and Data Size: Current and previous core subsystem library sizes are  
 9426 **shown below. These are the code and data sizes for the acpica.lib**  
 9427 **produced**  
 5540 *shown below. These are the code and data sizes for the acpica.lib produced*  
 9428 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 9429 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 9430 *debug output trace mechanism and has a much larger code and data size.*  
 9431 **Note**  
 9432 **that these values will vary depending on the efficiency of the compiler**  
 9433 **and**  
 5543 *debug output trace mechanism and has a much larger code and data size. Note*  
 5544 *that these values will vary depending on the efficiency of the compiler and*  
 9434 *the compiler options used during generation.*

9436	Previous Release:			
9437	Non-Debug Version:	78.3K Code,	11.5K Data,	89.8K Total
9438	Debug Version:	164.7K Code,	68.5K Data,	233.2K Total
9439	Current Release:			
9440	Non-Debug Version:	78.3K Code,	11.5K Data,	89.8K Total
9441	Debug Version:	165.3K Code,	69.4K Data,	234.7K Total

9444 2) iASL Compiler/Disassembler:

9446 Implemented the new ACPI 3.0 resource template macros - DWordSpace,  
 9447 ExtendedIO, ExtendedMemory, ExtendedSpace, QWordSpace, and WordSpace.

9448 Includes support in the disassembler.

9450 **Implemented support for the new (ACPI 3.0) parameter to the Register**  
 9451 **macro,**  
 5561 *Implemented support for the new (ACPI 3.0) parameter to the Register macro,*  
 9452 *AccessSize.*

9454 Fixed a problem where the \_HE resource name for the Interrupt macro was  
 9455 referencing bit 0 instead of bit 1.

9457 Implemented check for maximum 255 interrupts in the Interrupt macro.

9459 Fixed a problem with the predefined resource descriptor names where  
 9460 incorrect AML code was generated if the offset within the resource buffer  
 9461 was 0 or 1. The optimizer shortened the AML code to a single byte opcode  
 9462 but did not update the surrounding package lengths.

9464 **Changes to the Dma macro: All channels within the channel list must be**  
 9465 **in**  
 5574 *Changes to the Dma macro: All channels within the channel list must be in*  
 9466 *the range 0-7. Maximum 8 channels can be specified. BusMaster operand is*  
 9467 *optional (default is BusMaster).*

9469 Implemented check for maximum 7 data bytes for the VendorShort macro.

9471 **The ReadWrite parameter is now optional for the Memory32 and similar**  
 9472 **macros.**  
 5580 *The ReadWrite parameter is now optional for the Memory32 and similar macros.*

9474 -----  
 9475 03 December 2004. Summary of changes for version 20041203:

9477 1) ACPI CA Core Subsystem:

9479 **The low-level field insertion/extraction code (exfldio) has been**  
 9480 **completely**  
 5587 *The low-level field insertion/extraction code (exfldio) has been completely*  
 9481 *rewritten to eliminate unnecessary complexity, bugs, and boundary*  
 9482 *conditions.*

9484 **Fixed a problem in the ToInteger, ToBuffer, ToHexString, and**  
 9485 **ToDecimalString**  
 5591 *Fixed a problem in the ToInteger, ToBuffer, ToHexString, and ToDecimalString*  
 9486 *operators where the input operand could be inadvertently deleted if no*  
 9487 *conversion was necessary (e.g., if the input to ToInteger was an Integer*  
 9488 *object.)*

9490 **Fixed a problem with the ToDecimalString and ToHexString where an**  
 9491 **incorrect**  
 5596 *Fixed a problem with the ToDecimalString and ToHexString where an incorrect*  
 9492 *exception code was returned if the resulting string would be > 200 chars.*  
 9493 *AE\_STRING\_LIMIT is now returned.*

9495 Fixed a problem with the Concatenate operator where AE\_OK was always  
 9496 returned, even if the operation failed.

9498 Fixed a problem in oswinxf (used by AcpiExec and iASL) to allow > 128  
 9499 semaphores to be allocated.

9501 Code and Data Size: Current and previous core subsystem library sizes are  
 9502 **shown below. These are the code and data sizes for the acpica.lib**  
 9503 **produced**  
 5607 *shown below. These are the code and data sizes for the acpica.lib produced*  
 9504 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 9505 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 9506 *debug output trace mechanism and has a much larger code and data size.*

9507 **Note**  
 9508 **that these values will vary depending on the efficiency of the compiler**  
 9509 **and**  
 9510 *debug output trace mechanism and has a much larger code and data size. Note*  
 9511 *that these values will vary depending on the efficiency of the compiler and*  
 9512 *the compiler options used during generation.*

9512 Previous Release:  
 9513 Non-Debug Version: 78.5K Code, 11.5K Data, 90.0K Total  
 9514 Debug Version: 165.2K Code, 68.6K Data, 233.8K Total  
 9515 Current Release:  
 9516 Non-Debug Version: 78.3K Code, 11.5K Data, 89.8K Total  
 9517 Debug Version: 164.7K Code, 68.5K Data, 233.2K Total

9520 2) iASL Compiler/Disassembler:

9522 Fixed typechecking for the ObjectType and SizeOf operators. Problem was  
 9523 recently introduced in 20041119.

9525 **Fixed a problem with the ToUUID macro where the upper nybble of each**  
 9526 **buffer**  
 9527 *Fixed a problem with the ToUUID macro where the upper nybble of each buffer*  
 9528 *byte was inadvertently set to zero.*

9529 -----  
 9530 19 November 2004. Summary of changes for version 20041119:

9532 1) ACPI CA Core Subsystem:

9534 **Fixed a problem in the internal ConvertToInteger routine where new**  
 9535 **integers**  
 9536 **were not truncated to 32 bits for 32-bit ACPI tables. This routine**  
 9537 **converts**  
 9538 *Fixed a problem in the internal ConvertToInteger routine where new integers*  
 9539 *were not truncated to 32 bits for 32-bit ACPI tables. This routine converts*  
 9540 *buffers and strings to integers.*

9540 **Implemented support to store a value to an Index() on a String object.**  
 9541 **This**  
 9542 *Implemented support to store a value to an Index() on a String object. This*  
 9543 *is an ACPI 2.0 feature that had not yet been implemented.*

9544 **Implemented new behavior for storing objects to individual package**  
 9545 **elements**  
 9546 **(via the Index() operator). The previous behavior was to invoke the**  
 9547 **implicit**  
 9548 *Implemented new behavior for storing objects to individual package elements*  
 9549 *(via the Index() operator). The previous behavior was to invoke the implicit*  
 9550 *conversion rules if an object was already present at the index. The new*  
 9551 *behavior is to simply delete any existing object and directly store the*  
 9552 *new*  
 9553 *object. Although the ACPI specification seems unclear on this subject,*  
 9554 *other*  
 9555 *behavior is to simply delete any existing object and directly store the new*  
 9556 *object. Although the ACPI specification seems unclear on this subject, other*  
 9557 *ACPI implementations behave in this manner. (This is the root of the*  
 9558 *AE\_BAD\_HEX\_CONSTANT issue.)*

9556 **Modified the RSDP memory scan mechanism to support the extended checksum**  
 9557 **for**  
 9558 *Modified the RSDP memory scan mechanism to support the extended checksum for*  
 9559 *ACPI 2.0 (and above) RSDPs. Note that the search continues until a valid*  
 9560 *RSDP signature is found with a valid checksum.*

9561 Code and Data Size: Current and previous core subsystem library sizes are

9562 **shown below. These are the code and data sizes for the acpica.lib**  
 9563 **produced**  
 9564 *shown below. These are the code and data sizes for the acpica.lib produced*  
 9565 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 9566 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 9567 *debug output trace mechanism and has a much larger code and data size.*  
 9568 **Note**  
 9569 **that these values will vary depending on the efficiency of the compiler**  
 9570 **and**  
 9571 *debug output trace mechanism and has a much larger code and data size. Note*  
 9572 *that these values will vary depending on the efficiency of the compiler and*  
 9573 *the compiler options used during generation.*

9572 Previous Release:  
 9573 Non-Debug Version: 78.5K Code, 11.5K Data, 90.0K Total  
 9574 Debug Version: 165.2K Code, 68.6K Data, 233.8K Total  
 9575 Current Release:  
 9576 Non-Debug Version: 78.5K Code, 11.5K Data, 90.0K Total  
 9577 Debug Version: 165.2K Code, 68.6K Data, 233.8K Total

9580 2) iASL Compiler/Disassembler:

9582 Fixed a missing semicolon in the aslcompiler.y file.

9584 -----  
 9585 05 November 2004. Summary of changes for version 20041105:

9587 1) ACPI CA Core Subsystem:

9589 **Implemented support for FADT revision 2. This was an interim table**  
 9590 **(between**  
 9591 *Implemented support for FADT revision 2. This was an interim table (between*  
 9592 *ACPI 1.0 and ACPI 2.0) that adds support for the FADT reset register.*

9593 Implemented optional support to allow uninitialized LocalX and ArgX  
 9594 **variables in a control method. The variables are initialized to an**  
 9595 **Integer**  
 9596 *variables in a control method. The variables are initialized to an Integer*  
 9597 *object with a value of zero. This support is enabled by setting the*  
 9598 *AcpiGbl\_EnableInterpreterSlack flag to TRUE.*

9599 **Implemented support for Integer objects for the SizeOf operator. Either**  
 9600 **4**  
 9601 **or 8 is returned, depending on the current integer size (32-bit or 64-**  
 9602 **bit,**  
 9603 *Implemented support for Integer objects for the SizeOf operator. Either 4*  
 9604 *or 8 is returned, depending on the current integer size (32-bit or 64-bit,*  
 9605 *depending on the parent table revision).*

9605 **Fixed a problem in the implementation of the SizeOf and ObjectType**  
 9606 **operators**  
 9607 *Fixed a problem in the implementation of the SizeOf and ObjectType operators*  
 9608 *where the operand was resolved to a value too early, causing incorrect*  
 9609 *return values for some objects.*

9610 Fixed some possible memory leaks during exceptional conditions.

9612 Code and Data Size: Current and previous core subsystem library sizes are  
 9613 **shown below. These are the code and data sizes for the acpica.lib**  
 9614 **produced**  
 9615 *shown below. These are the code and data sizes for the acpica.lib produced*  
 9616 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 9617 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 9618 *debug output trace mechanism and has a much larger code and data size.*  
 9619 **Note**

9619 **that these values will vary depending on the efficiency of the compiler**  
 9620 **and**  
 5701 *debug output trace mechanism and has a much larger code and data size. Note*  
 5702 *that these values will vary depending on the efficiency of the compiler and*  
 9621 the compiler options used during generation.

9623 Previous Release:  
 9624 Non-Debug Version: 78.0K Code, 11.5K Data, 89.5K Total  
 9625 Debug Version: 164.8K Code, 68.6K Data, 233.4K Total  
 9626 Current Release:  
 9627 Non-Debug Version: 78.5K Code, 11.5K Data, 90.0K Total  
 9628 Debug Version: 165.2K Code, 68.6K Data, 233.8K Total

9631 2) iASL Compiler/Disassembler:

9633 Implemented support for all ACPI 3.0 reserved names and methods.

9635 Implemented all ACPI 3.0 grammar elements in the front-end, including  
 9636 support for semicolons.

9638 Implemented the ACPI 3.0 Function() and ToUUID() macros

9640 Fixed a problem in the disassembler where a Scope() operator would not be  
 9641 emitted properly if the target of the scope was in another table.

9643 -----  
 9644 15 October 2004. Summary of changes for version 20041015:

9646 Note: ACPI CA is currently undergoing an in-depth and complete formal  
 9647 evaluation to test/verify the following areas. Other suggestions are  
 9648 welcome. This will result in an increase in the frequency of releases and  
 9649 the number of bug fixes in the next few months.  
 9650 - Functional tests for all ASL/AML operators  
 9651 - All implicit/explicit type conversions  
 9652 - Bit fields and operation regions  
 9653 - 64-bit math support and 32-bit-only "truncated" math support  
 9654 - Exceptional conditions, both compiler and interpreter  
 9655 - Dynamic object deletion and memory leaks  
 9656 - ACPI 3.0 support when implemented  
 9657 - External interfaces to the ACPI subsystem

9660 1) ACPI CA Core Subsystem:

9662 **Fixed two alignment issues on 64-bit platforms - within debug statements**  
 9663 **in**  
 9664 **AcpiEvGpeDetect and AcpiEvCreateGpeBlock. Removed references to the**  
 9665 **Address**  
 5744 *Fixed two alignment issues on 64-bit platforms - within debug statements in*  
 5745 *AcpiEvGpeDetect and AcpiEvCreateGpeBlock. Removed references to the Address*  
 9666 field within the non-aligned ACPI generic address structure.

9668 Fixed a problem in the Increment and Decrement operators where incorrect  
 9669 operand resolution could result in the inadvertent modification of the  
 9670 original integer when the integer is passed into another method as an  
 9671 argument and the arg is then incremented/decremented.

9673 **Fixed a problem in the FromBCD operator where the upper 32-bits of a 64-**  
 9674 **bit**  
 5753 *Fixed a problem in the FromBCD operator where the upper 32-bits of a 64-bit*  
 9675 BCD number were truncated during conversion.

9677 **Fixed a problem in the ToDecimal operator where the length of the**  
 9678 **resulting**  
 9679 **string could be set incorrectly too long if the input operand was a**

9680 **Buffer**  
 5756 *Fixed a problem in the ToDecimal operator where the length of the resulting*  
 5757 *string could be set incorrectly too long if the input operand was a Buffer*  
 9681 object.

9683 **Fixed a problem in the Logical operators (LLess, etc.) where a NULL byte**  
 9684 **(0)**  
 5760 *Fixed a problem in the Logical operators (LLess, etc.) where a NULL byte (0)*  
 9685 within a buffer would prematurely terminate a compare between buffer  
 9686 objects.

9688 Added a check for string overflow (>200 characters as per the ACPI  
 9689 specification) during the Concatenate operator with two string operands.

9691 Code and Data Size: Current and previous core subsystem library sizes are  
 9692 **shown below. These are the code and data sizes for the acpica.lib**  
 9693 **produced**  
 5768 *shown below. These are the code and data sizes for the acpica.lib produced*  
 9694 by the Microsoft Visual C++ 6.0 compiler, and these values do not include  
 9695 any ACPI driver or OSPM code. The debug version of the code includes the  
 9696 **debug output trace mechanism and has a much larger code and data size.**  
 9697 **Note**  
 9698 **that these values will vary depending on the efficiency of the compiler**  
 9699 **and**  
 5771 *debug output trace mechanism and has a much larger code and data size. Note*  
 5772 *that these values will vary depending on the efficiency of the compiler and*  
 9700 the compiler options used during generation.

9702 Previous Release:  
 9703 Non-Debug Version: 77.8K Code, 11.5K Data, 89.3K Total  
 9704 Debug Version: 164.6K Code, 68.5K Data, 233.1K Total  
 9705 Current Release:  
 9706 Non-Debug Version: 78.0K Code, 11.5K Data, 89.5K Total  
 9707 Debug Version: 164.8K Code, 68.6K Data, 233.4K Total

9711 2) iASL Compiler/Disassembler:

9713 Allow the use of the ObjectType operator on uninitialized Locals and Args  
 9714 (returns 0 as per the ACPI specification).

9716 **Fixed a problem where the compiler would fault if there was a syntax**  
 9717 **error**  
 5789 *Fixed a problem where the compiler would fault if there was a syntax error*  
 9718 in the FieldName of all of the various CreateXXXField operators.

9720 **Disallow the use of lower case letters within the EISAID macro, as per**  
 9721 **the**  
 9722 **ACPI specification. All EISAID strings must be of the form "UUUNNNN"**  
 9723 **Where**  
 5792 *Disallow the use of lower case letters within the EISAID macro, as per the*  
 5793 *ACPI specification. All EISAID strings must be of the form "UUUNNNN" Where*  
 9724 U is an uppercase letter and N is a hex digit.

9727 -----  
 9728 06 October 2004. Summary of changes for version 20041006:

9730 1) ACPI CA Core Subsystem:

9732 Implemented support for the ACPI 3.0 Timer operator. This ASL function  
 9733 implements a 64-bit timer with 100 nanosecond granularity.

9735 Defined a new OSL interface, AcpiOsGetTimer. This interface is used to  
 9736 **implement the ACPI 3.0 Timer operator. This allows the host OS to**



9737 **implement**  
 9738 **the timer with the best clock available. Also, it keeps the core**  
 9739 **subsystem**  
 5806 *implement the ACPI 3.0 Timer operator. This allows the host OS to implement*  
 5807 *the timer with the best clock available. Also, it keeps the core subsystem*  
 9740 *out of the clock handling business, since the host OS (usually) performs*  
 9741 *this function.*

9743 *Fixed an alignment issue on 64-bit platforms. The HwLowLevelRead(Write)*  
 9744 *functions use a 64-bit address which is part of the packed ACPI Generic*  
 9745 **Address Structure. Since the structure is non-aligned, the alignment**  
 9746 **macros**  
 5813 *Address Structure. Since the structure is non-aligned, the alignment macros*  
 9747 *are now used to extract the address to a local variable before use.*

9749 **Fixed a problem where the ToInteger operator assumed all input strings**  
 9750 **were**  
 9751 **hexadecimal. The operator now handles both decimal strings and hex**  
 9752 **strings**  
 5816 *Fixed a problem where the ToInteger operator assumed all input strings were*  
 5817 *hexadecimal. The operator now handles both decimal strings and hex strings*  
 9753 *(prefixed with "0x").*

9755 *Fixed a problem where the string length in the string object created as a*  
 9756 *result of the internal ConvertToString procedure could be incorrect. This*  
 9757 **potentially affected all implicit conversions and also the**  
 9758 **ToDecimalString**  
 5822 *potentially affected all implicit conversions and also the ToDecimalString*  
 9759 *and ToHexString operators.*

9761 *Fixed two problems in the ToString operator. If the length parameter was*  
 9762 *zero, an incorrect string object was created and the value of the input*  
 9763 *length parameter was inadvertently changed from zero to Ones.*

9765 **Fixed a problem where the optional ResourceSource string in the**  
 9766 **ExtendedIRQ**  
 5829 *Fixed a problem where the optional ResourceSource string in the ExtendedIRQ*  
 9767 *resource macro was ignored.*

9769 **Simplified the interfaces to the internal division functions, reducing**  
 9770 **code**  
 5832 *Simplified the interfaces to the internal division functions, reducing code*  
 9771 *size and complexity.*

9773 *Code and Data Size: Current and previous core subsystem library sizes are*  
 9774 **shown below. These are the code and data sizes for the acpica.lib**  
 9775 **produced**  
 5836 *shown below. These are the code and data sizes for the acpica.lib produced*  
 9776 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 9777 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 9778 **debug output trace mechanism and has a much larger code and data size.**  
 9779 **Note**  
 9780 **that these values will vary depending on the efficiency of the compiler**  
 9781 **and**  
 5839 *debug output trace mechanism and has a much larger code and data size. Note*  
 5840 *that these values will vary depending on the efficiency of the compiler and*  
 9782 *the compiler options used during generation.*

9784	Previous Release:			
9785	Non-Debug Version:	77.9K Code,	11.4K Data,	89.3K Total
9786	Debug Version:	164.5K Code,	68.3K Data,	232.8K Total
9787	Current Release:			
9788	Non-Debug Version:	77.8K Code,	11.5K Data,	89.3K Total
9789	Debug Version:	164.6K Code,	68.5K Data,	233.1K Total

9792 2) iASL Compiler/Disassembler:

9794 Implemented support for the ACPI 3.0 Timer operator.

9796 **Fixed a problem where the Default() operator was inadvertently ignored in**  
 9797 **a**  
 5855 *Fixed a problem where the Default() operator was inadvertently ignored in a*  
 9798 *Switch/Case block. This was a problem in the translation of the Switch*  
 9799 *statement to If...Else pairs.*

9801 **Added support to allow a standalone Return operator, with no parentheses**  
 9802 **(or**  
 5859 *Added support to allow a standalone Return operator, with no parentheses (or*  
 9803 *operands).*

9805 *Fixed a problem with code generation for the ElseIf operator where the*  
 9806 *translated Else...If parse tree was improperly constructed leading to the*  
 9807 *loss of some code.*

9809 -----

9810 22 September 2004. Summary of changes for version 20040922:

9812 1) ACPI CA Core Subsystem:

9814 **Fixed a problem with the implementation of the LNot() operator where**  
 9815 **"Ones"**  
 9816 **was not returned for the TRUE case. Changed the code to return Ones**  
 9817 **instead**  
 9818 **of (!Arg) which was usually 1. This change affects iASL constant folding**  
 9819 **for**  
 5871 *Fixed a problem with the implementation of the LNot() operator where "Ones"*  
 5872 *was not returned for the TRUE case. Changed the code to return Ones instead*  
 5873 *of (!Arg) which was usually 1. This change affects iASL constant folding for*  
 9820 *this operator also.*

9822 **Fixed a problem in AcpiUtInitializeBuffer where an existing buffer was**  
 9823 **not**  
 5876 *Fixed a problem in AcpiUtInitializeBuffer where an existing buffer was not*  
 9824 *initialized properly -- Now zero the entire buffer in this case where the*  
 9825 *buffer already exists.*

9827 *Changed the interface to AcpiOsSleep from (UINT32 Seconds, UINT32*  
 9828 *Milliseconds) to simply (ACPI\_INTEGER Milliseconds). This simplifies all*  
 9829 *related code considerably. This will require changes/updates to all OS*  
 9830 *interface layers (OSLs.)*

9832 **Implemented a new external interface, AcpiInstallExceptionHandler, to**  
 9833 **allow**  
 9834 **a system exception handler to be installed. This handler is invoked upon**  
 9835 **any**  
 5885 *Implemented a new external interface, AcpiInstallExceptionHandler, to allow*  
 5886 *a system exception handler to be installed. This handler is invoked upon any*  
 9836 *run-time exception that occurs during control method execution.*

9838 *Added support for the DSDT in AcpiTbFindTable. This allows the*  
 9839 *DataTableRegion() operator to access the local copy of the DSDT.*

9841 *Code and Data Size: Current and previous core subsystem library sizes are*  
 9842 **shown below. These are the code and data sizes for the acpica.lib**  
 9843 **produced**  
 5893 *shown below. These are the code and data sizes for the acpica.lib produced*  
 9844 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 9845 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 9846 **debug output trace mechanism and has a much larger code and data size.**  
 9847 **Note**  
 9848 **that these values will vary depending on the efficiency of the compiler**

9849 **and**  
 5896 *debug output trace mechanism and has a much larger code and data size. Note*  
 5897 *that these values will vary depending on the efficiency of the compiler and*  
 9850 *the compiler options used during generation.*

9852	Previous Release:			
9853	Non-Debug Version:	77.8K Code,	11.4K Data,	89.2K Total
9854	Debug Version:	164.2K Code,	68.2K Data,	232.4K Total
9855	Current Release:			
9856	Non-Debug Version:	77.9K Code,	11.4K Data,	89.3K Total
9857	Debug Version:	164.5K Code,	68.3K Data,	232.8K Total

9860 2) iASL Compiler/Disassembler:

9862 Fixed a problem with constant folding and the LNot operator. LNot was  
 9863 **returning 1 in the TRUE case, not Ones as per the ACPI specification.**  
 9864 **This**  
 5911 *returning 1 in the TRUE case, not Ones as per the ACPI specification. This*  
 9865 *could result in the generation of an incorrect folded/reduced constant.*

9867 End-Of-File is now allowed within a "/\*"-style comment. A parse error no  
 9868 **longer occurs if such a comment is at the very end of the input ASL**  
 9869 **source**  
 5915 *longer occurs if such a comment is at the very end of the input ASL source*  
 9870 *file.*

9872 Implemented the "-r" option to override the Revision in the table header.  
 9873 **The initial use of this option will be to simplify the evaluation of the**  
 9874 **AML**  
 9875 **interpreter by allowing a single ASL source module to be compiled for**  
 9876 **either**  
 5919 *The initial use of this option will be to simplify the evaluation of the AML*  
 5920 *interpreter by allowing a single ASL source module to be compiled for either*  
 9877 *32-bit or 64-bit integers.*

9880 -----  
 9881 27 August 2004. Summary of changes for version 20040827:

9883 1) ACPI CA Core Subsystem:

9885 - Implemented support for implicit object conversion in the non-numeric  
 9886 **logical operators (LEqual, LGreater, LGreaterEqual, LLess, LLessEqual,**  
 9887 **and**  
 5930 *logical operators (LEqual, LGreater, LGreaterEqual, LLess, LLessEqual, and*  
 9888 *LNotEqual.) Any combination of Integers/Strings/Buffers may now be used;*  
 9889 **the second operand is implicitly converted on the fly to match the type**  
 9890 **of**  
 5932 *the second operand is implicitly converted on the fly to match the type of*  
 9891 *the first operand. For example:*

9893     LEqual (Source1, Source2)

9895 **Source1 and Source2 must each evaluate to an integer, a string, or a**  
 9896 **buffer.**  
 9897 **The data type of Source1 dictates the required type of Source2. Source2**  
 9898 **is**  
 5937 *Source1 and Source2 must each evaluate to an integer, a string, or a buffer.*  
 5938 *The data type of Source1 dictates the required type of Source2. Source2 is*  
 9899 *implicitly converted if necessary to match the type of Source1.*

9901 - **Updated and corrected the behavior of the string conversion support.**  
 9902 **The**  
 5941 *- Updated and corrected the behavior of the string conversion support. The*  
 9903 *rules concerning conversion of buffers to strings (according to the ACPI*

9904 specification) are as follows:

9906 ToDecimalString - explicit byte-wise conversion of buffer to string of  
 9907 **decimal values (0-255) separated by commas. ToHexString - explicit byte-**  
 9908 **wise**  
 5946 *decimal values (0-255) separated by commas. ToHexString - explicit byte-wise*  
 9909 *conversion of buffer to string of hex values (0-FF) separated by commas.*  
 9910 **ToString - explicit byte-wise conversion of buffer to string. Byte-by-**  
 9911 **byte**  
 9912 **copy with no transform except NULL terminated. Any other implicit buffer-**  
 9913 **to-**  
 9914 **string conversion - byte-wise conversion of buffer to string of hex**  
 9915 **values**  
 5948 *ToString - explicit byte-wise conversion of buffer to string. Byte-by-byte*  
 5949 *copy with no transform except NULL terminated. Any other implicit buffer-to-*  
 5950 *string conversion - byte-wise conversion of buffer to string of hex values*  
 9916 *(0-FF) separated by spaces.*

9918 - Fixed typo in definition of AcpiGbl\_EnableInterpreterSlack.

9920 - **Fixed a problem in AcpiNsGetPathnameLength where the returned length**  
 9921 **was**  
 5955 *- Fixed a problem in AcpiNsGetPathnameLength where the returned length was*  
 9922 *one byte too short in the case of a node in the root scope. This could*  
 9923 *cause a fault during debug output.*

9925 - **Code and Data Size: Current and previous core subsystem library sizes**  
 9926 **are**  
 9927 **shown below. These are the code and data sizes for the acpica.lib**  
 9928 **produced**  
 5959 *- Code and Data Size: Current and previous core subsystem library sizes are*  
 5960 *shown below. These are the code and data sizes for the acpica.lib produced*  
 9929 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 9930 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 9931 **debug output trace mechanism and has a much larger code and data size.**  
 9932 **Note**  
 9933 **that these values will vary depending on the efficiency of the compiler**  
 9934 **and**  
 5963 *debug output trace mechanism and has a much larger code and data size. Note*  
 5964 *that these values will vary depending on the efficiency of the compiler and*  
 9935 *the compiler options used during generation.*

9937	Previous Release:			
9938	Non-Debug Version:	77.9K Code,	11.5K Data,	89.4K Total
9939	Debug Version:	164.1K Code,	68.3K Data,	232.4K Total
9940	Current Release:			
9941	Non-Debug Version:	77.8K Code,	11.4K Data,	89.2K Total
9942	Debug Version:	164.2K Code,	68.2K Data,	232.4K Total

9945 2) iASL Compiler/Disassembler:

9947 - Fixed a Linux generation error.

9950 -----  
 9951 16 August 2004. Summary of changes for version 20040816:

9953 1) ACPI CA Core Subsystem:

9955 Designed and implemented support within the AML interpreter for the so-  
 9956 **called "implicit return". This support returns the result of the last**  
 9957 **ASL**  
 5986 *called "implicit return". This support returns the result of the last ASL*  
 9958 *operation within a control method, in the absence of an explicit Return()*  
 9959 *operator. A few machines depend on this behavior, even though it is not*

9960 explicitly supported by the ASL language. It is optional support that  
 9961 can  
 9989 explicitly supported by the ASL language. It is optional support that can  
 9962 be enabled at runtime via the AcpiGbl\_EnableInterpreterSlack flag.

9964 Removed support for the PCI\_Config address space from the internal low  
 9965 level  
 5992 Removed support for the PCI\_Config address space from the internal low level  
 9966 hardware interfaces (AcpiHwLowLevelRead and AcpiHwLowLevelWrite). This  
 9967 support was not used internally, and would not work correctly anyway  
 9968 because  
 5994 support was not used internally, and would not work correctly anyway because  
 9969 the PCI bus number and segment number were not supported. There are  
 9970 separate interfaces for PCI configuration space access because of the  
 9971 unique  
 5996 separate interfaces for PCI configuration space access because of the unique  
 9972 interface.

9974 Code and Data Size: Current and previous core subsystem library sizes are  
 9975 shown below. These are the code and data sizes for the acpica.lib  
 9976 produced  
 6000 shown below. These are the code and data sizes for the acpica.lib produced  
 9977 by the Microsoft Visual C++ 6.0 compiler, and these values do not include  
 9978 any ACPI driver or OSPM code. The debug version of the code includes the  
 9979 debug output trace mechanism and has a much larger code and data size.  
 9980 Note  
 9981 that these values will vary depending on the efficiency of the compiler  
 9982 and  
 6003 debug output trace mechanism and has a much larger code and data size. Note  
 6004 that these values will vary depending on the efficiency of the compiler and  
 9983 the compiler options used during generation.

9985	Previous Release:			
9986	Non-Debug Version:	78.0K Code,	11.5K Data,	89.5K Total
9987	Debug Version:	164.1K Code,	68.2K Data,	232.3K Total
9988	Current Release:			
9989	Non-Debug Version:	77.9K Code,	11.5K Data,	89.4K Total
9990	Debug Version:	164.1K Code,	68.3K Data,	232.4K Total

9993 2) iASL Compiler/Disassembler:

9995 Fixed a problem where constants in ASL expressions at the root level (not  
 9996 within a control method) could be inadvertently truncated during code  
 9997 generation. This problem was introduced in the 20040715 release.

10000 -----  
 10001 15 July 2004. Summary of changes for version 20040715:

10003 1) ACPI CA Core Subsystem:

10005 Restructured the internal HW GPE interfaces to pass/track the current  
 10006 state  
 6027 Restructured the internal HW GPE interfaces to pass/track the current state  
 10007 of interrupts (enabled/disabled) in order to avoid possible deadlock and  
 10008 increase flexibility of the interfaces.

10010 Implemented a "lexicographical compare" for String and Buffer objects  
 10011 within  
 10012 the logical operators -- LGreater, LLess, LGreaterEqual, and LLessEqual -  
 10013 -  
 10014 as per further clarification to the ACPI specification. Behavior is  
 10015 similar  
 6031 Implemented a "lexicographical compare" for String and Buffer objects within  
 6032 the logical operators -- LGreater, LLess, LGreaterEqual, and LLessEqual --

6033 as per further clarification to the ACPI specification. Behavior is similar  
 10016 to C library "strcmp".

10018 Completed a major reduction in CPU stack use for the AcpiGetFirmwareTable  
 10019 external function. In the 32-bit non-debug case, the stack use has been  
 10020 reduced from 168 bytes to 32 bytes.

10022 Deployed a new run-time configuration flag,  
 10023 AcpiGbl\_EnableInterpreterSlack,  
 6040 Deployed a new run-time configuration flag, AcpiGbl\_EnableInterpreterSlack,  
 10024 whose purpose is to allow the AML interpreter to forgive certain bad AML  
 10025 constructs. Default setting is FALSE.

10027 Implemented the first use of AcpiGbl\_EnableInterpreterSlack in the Field  
 10028 IO  
 10029 support code. If enabled, it allows field access to go beyond the end of  
 10030 a  
 10031 region definition if the field is within the region length rounded up to  
 10032 the  
 6044 Implemented the first use of AcpiGbl\_EnableInterpreterSlack in the Field IO  
 6045 support code. If enabled, it allows field access to go beyond the end of a  
 6046 region definition if the field is within the region length rounded up to the  
 10033 next access width boundary (a common coding error.)

10035 Renamed OSD\_HANDLER to ACPI\_OSD\_HANDLER, and OSD\_EXECUTION\_CALLBACK to  
 10036 ACPI\_OSD\_EXEC\_CALLBACK for consistency with other ACPI symbols. Also,  
 10037 these  
 6050 ACPI\_OSD\_EXEC\_CALLBACK for consistency with other ACPI symbols. Also, these  
 10038 symbols are lowercased by the latest version of the AcpiSrc tool.

10040 The prototypes for the PCI interfaces in acpiosxf.h have been updated to  
 10041 rename "Register" to simply "Reg" to prevent certain compilers from  
 10042 complaining.

10044 Code and Data Size: Current and previous core subsystem library sizes are  
 10045 shown below. These are the code and data sizes for the acpica.lib  
 10046 produced  
 6058 shown below. These are the code and data sizes for the acpica.lib produced  
 10047 by the Microsoft Visual C++ 6.0 compiler, and these values do not include  
 10048 any ACPI driver or OSPM code. The debug version of the code includes the  
 10049 debug output trace mechanism and has a much larger code and data size.  
 10050 Note  
 10051 that these values will vary depending on the efficiency of the compiler  
 10052 and  
 6061 debug output trace mechanism and has a much larger code and data size. Note  
 6062 that these values will vary depending on the efficiency of the compiler and  
 10053 the compiler options used during generation.

10055	Previous Release:			
10056	Non-Debug Version:	77.8K Code,	11.5K Data,	89.3K Total
10057	Debug Version:	163.8K Code,	68.2K Data,	232.0K Total
10058	Current Release:			
10059	Non-Debug Version:	78.0K Code,	11.5K Data,	89.5K Total
10060	Debug Version:	164.1K Code,	68.2K Data,	232.3K Total

10063 2) iASL Compiler/Disassembler:

10065 Implemented full support for Package objects within the Case() operator.  
 10066 Note: The Break() operator is currently not supported within Case blocks  
 10067 (TermLists) as there is some question about backward compatibility with  
 10068 ACPI  
 6077 (TermLists) as there is some question about backward compatibility with ACPI  
 10069 1.0 interpreters.

10072 **Fixed a problem where complex terms were not supported properly within**  
 10073 **the**  
 6081 *Fixed a problem where complex terms were not supported properly within the*  
 10074 *Switch() operator.*

10076 Eliminated extraneous warning for compiler-emitted reserved names of the  
 10077 form "\_T\_x". (Used in Switch/Case operators.)

10079 Eliminated optimization messages for "\_T\_x" objects and small constants  
 10080 within the DefinitionBlock operator.

10083 -----  
 10084 15 June 2004. Summary of changes for version 20040615:

10086 1) ACPI CA Core Subsystem:

10088 **Implemented support for Buffer and String objects (as per ACPI 2.0) for**  
 10089 **the**  
 6096 *Implemented support for Buffer and String objects (as per ACPI 2.0) for the*  
 10090 *following ASL operators: LEqual, LGreater, LLess, LGreaterEqual, and*  
 10091 *LLessEqual.*

10093 All directory names in the entire source package are lower case, as they  
 10094 were in earlier releases.

10096 **Implemented "Disassemble" command in the AML debugger that will**  
 10097 **disassemble**  
 6103 *Implemented "Disassemble" command in the AML debugger that will disassemble*  
 10098 *a single control method.*

10100 Code and Data Size: Current and previous core subsystem library sizes are  
 10101 **shown below. These are the code and data sizes for the acpica.lib**  
 10102 **produced**  
 6107 *shown below. These are the code and data sizes for the acpica.lib produced*  
 10103 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 10104 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 10105 *debug output trace mechanism and has a much larger code and data size.*  
 10106 **Note**  
 10107 **that these values will vary depending on the efficiency of the compiler**  
 10108 **and**  
 6110 *debug output trace mechanism and has a much larger code and data size. Note*  
 6111 *that these values will vary depending on the efficiency of the compiler and*  
 10109 *the compiler options used during generation.*

10111 Previous Release:  
 10112 Non-Debug Version: 77.7K Code, 11.5K Data, 89.2K Total  
 10113 Debug Version: 163.3K Code, 67.2K Data, 230.5K Total

10115 Current Release:  
 10116 Non-Debug Version: 77.8K Code, 11.5K Data, 89.3K Total  
 10117 Debug Version: 163.8K Code, 68.2K Data, 232.0K Total

10120 2) iASL Compiler/Disassembler:

10122 **Implemented support for Buffer and String objects (as per ACPI 2.0) for**  
 10123 **the**  
 6125 *Implemented support for Buffer and String objects (as per ACPI 2.0) for the*  
 10124 *following ASL operators: LEqual, LGreater, LLess, LGreaterEqual, and*  
 10125 *LLessEqual.*

10127 All directory names in the entire source package are lower case, as they  
 10128 were in earlier releases.

10130 Fixed a fault when using the -g or -d<nofilename> options if the FADT was

10131 not found.

10133 **Fixed an issue with the Windows version of the compiler where later**  
 10134 **versions**  
 6135 *Fixed an issue with the Windows version of the compiler where later versions*  
 10135 *of Windows place the FADT in the registry under the name "FADT" and not*  
 10136 *"FACP" as earlier versions did. This applies when using the -g or -*  
 10137 *d<nofilename> options. The compiler now looks for both strings as*  
 10138 *necessary.*

10140 **Fixed a problem with compiler namepath optimization where a namepath**  
 10141 **within**  
 10142 **the Scope() operator could not be optimized if the namepath was a subpath**  
 10143 **of**  
 6141 *Fixed a problem with compiler namepath optimization where a namepath within*  
 6142 *the Scope() operator could not be optimized if the namepath was a subpath of*  
 10144 *the current scope path.*

10146 -----  
 10147 27 May 2004. Summary of changes for version 20040527:

10149 1) ACPI CA Core Subsystem:

10151 **Completed a new design and implementation for EBDA (Extended BIOS Data**  
 10152 **Area)**  
 10153 **support in the RSDP scan code. The original code improperly scanned for**  
 10154 **the**  
 10155 **EBDA by simply scanning from memory location 0 to 0x400. The correct**  
 10156 **method**  
 6150 *Completed a new design and implementation for EBDA (Extended BIOS Data Area)*  
 6151 *support in the RSDP scan code. The original code improperly scanned for the*  
 6152 *EBDA by simply scanning from memory location 0 to 0x400. The correct method*  
 10157 *is to first obtain the EBDA pointer from within the BIOS data area, then*  
 10158 *scan 1K of memory starting at the EBDA pointer. There appear to be few*  
 10159 *if*  
 6154 *scan 1K of memory starting at the EBDA pointer. There appear to be few if*  
 10160 *any machines that place the RSDP in the EBDA, however.*

10162 Integrated a fix for a possible fault during evaluation of BufferField  
 10163 arguments. Obsolete code that was causing the problem was removed.

10165 Found and fixed a problem in the Field Support Code where data could be  
 10166 corrupted on a bit field read that starts on an aligned boundary but does  
 10167 not end on an aligned boundary. Merged the read/write "datum length"  
 10168 calculation code into a common procedure.

10170 Rolled in a couple of changes to the FreeBSD-specific header.

10173 Code and Data Size: Current and previous core subsystem library sizes are  
 10174 **shown below. These are the code and data sizes for the acpica.lib**  
 10175 **produced**  
 6169 *shown below. These are the code and data sizes for the acpica.lib produced*  
 10176 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 10177 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 10178 *debug output trace mechanism and has a much larger code and data size.*  
 10179 **Note**  
 10180 **that these values will vary depending on the efficiency of the compiler**  
 10181 **and**  
 6172 *debug output trace mechanism and has a much larger code and data size. Note*  
 6173 *that these values will vary depending on the efficiency of the compiler and*  
 10182 *the compiler options used during generation.*

10184 Previous Release:  
 10185 Non-Debug Version: 77.6K Code, 11.5K Data, 89.1K Total  
 10186 Debug Version: 163.2K Code, 67.2K Data, 230.4K Total

10187 Current Release:  
 10188 Non-Debug Version: 77.7K Code, 11.5K Data, 89.2K Total  
 10189 Debug Version: 163.3K Code, 67.2K Data, 230.5K Total

10192 2) iASL Compiler/Disassembler:

10194 **Fixed a generation warning produced by some overly-verbose compilers for**  
 10195 **a**  
 6186 *Fixed a generation warning produced by some overly-verbose compilers for a*  
 10196 *64-bit constant.*

10198 -----  
 10199 14 May 2004. Summary of changes for version 20040514:

10201 1) ACPI CA Core Subsystem:

10203 Fixed a problem where hardware GPE enable bits sometimes not set properly  
 10204 during and after GPE method execution. Result of 04/27 changes.

10206 Removed extra "clear all GPEs" when sleeping/waking.

10208 Removed AcpiHwEnableGpe and AcpiHwDisableGpe, replaced by the single  
 10209 **AcpiHwWriteGpeEnableReg. Changed a couple of calls to the functions above**  
 10210 **to**  
 6200 *AcpiHwWriteGpeEnableReg. Changed a couple of calls to the functions above to*  
 10211 *the new AcpiEv\* calls as appropriate.*

10213 **ACPI\_OS\_NAME was removed from the OS-specific headers. The default name**  
 10214 **is**  
 10215 **now "Microsoft Windows NT" for maximum compatibility. However this can**  
 10216 **be**  
 6203 *ACPI\_OS\_NAME was removed from the OS-specific headers. The default name is*  
 6204 *now "Microsoft Windows NT" for maximum compatibility. However this can be*  
 10217 *changed by modifying the acconfig.h file.*

10219 Allow a single invocation of AcpiInstallNotifyHandler for a handler that  
 10220 traps both types of notifies (System, Device). Use ACPI\_ALL\_NOTIFY flag.

10222 Run \_INI methods on ThermalZone objects. This is against the ACPI  
 10223 **specification, but there is apparently ASL code in the field that has**  
 10224 **these**  
 6211 *specification, but there is apparently ASL code in the field that has these*  
 10225 *\_INI methods, and apparently "other" AML interpreters execute them.*

10227 Performed a full 16/32/64 bit lint that resulted in some small changes.

10229 Added a sleep simulation command to the AML debugger to test sleep code.

10231 Code and Data Size: Current and previous core subsystem library sizes are  
 10232 **shown below. These are the code and data sizes for the acpica.lib**  
 10233 **produced**  
 6219 *shown below. These are the code and data sizes for the acpica.lib produced*  
 10234 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 10235 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 10236 *debug output trace mechanism and has a much larger code and data size.*  
 10237 **Note**  
 10238 **that these values will vary depending on the efficiency of the compiler**  
 10239 **and**  
 6222 *debug output trace mechanism and has a much larger code and data size. Note*  
 6223 *that these values will vary depending on the efficiency of the compiler and*  
 10240 *the compiler options used during generation.*

10242 Previous Release:  
 10243 Non-Debug Version: 77.6K Code, 11.5K Data, 89.1K Total  
 10244 Debug Version: 162.9K Code, 67.0K Data, 229.9K Total

10245 Current Release:  
 10246 Non-Debug Version: 77.6K Code, 11.5K Data, 89.1K Total  
 10247 Debug Version: 163.2K Code, 67.2K Data, 230.4K Total

10249 -----  
 10250 27 April 2004. Summary of changes for version 20040427:

10252 1) ACPI CA Core Subsystem:

10254 Completed a major overhaul of the GPE handling within ACPI CA. There are  
 10255 **now three types of GPEs: wake-only, runtime-only, and combination**  
 10256 **wake/run.**  
 6239 *now three types of GPEs: wake-only, runtime-only, and combination wake/run.*  
 10257 *The only GPEs allowed to be combination wake/run are for button-style*  
 10258 *devices such as a control-method power button, control-method sleep*  
 10259 *button,*  
 10260 *or a notebook lid switch. GPEs that have an \_Lxx or \_Exx method and are*  
 10261 *not*  
 6241 *devices such as a control-method power button, control-method sleep button,*  
 6242 *or a notebook lid switch. GPEs that have an \_Lxx or \_Exx method and are not*  
 10262 *referenced by any \_PRW methods are marked for "runtime" and hardware*  
 10263 *enabled. Any GPE that is referenced by a \_PRW method is marked for*  
 10264 *"wake"*  
 6244 *enabled. Any GPE that is referenced by a \_PRW method is marked for "wake"*  
 10265 *(and disabled at runtime). However, at sleep time, only those GPEs that*  
 10266 *have been specifically enabled for wake via the AcpiEnableGpe interface*  
 10267 *will*  
 6246 *have been specifically enabled for wake via the AcpiEnableGpe interface will*  
 10268 *actually be hardware enabled.*

10270 **A new external interface has been added, AcpiSetGpeType(), that is meant**  
 10271 **to**  
 10272 **be used by device drivers to force a GPE to a particular type. It will**  
 10273 **be**  
 6249 *A new external interface has been added, AcpiSetGpeType(), that is meant to*  
 6250 *be used by device drivers to force a GPE to a particular type. It will be*  
 10274 *especially useful for the drivers for the button devices mentioned above.*

10276 Completed restructuring of the ACPI CA initialization sequence so that  
 10277 **default operation region handlers are installed before GPEs are**  
 10278 **initialized**  
 10279 **and the \_PRW methods are executed. This will prevent errors when the**  
 10280 **\_PRW**  
 6254 *default operation region handlers are installed before GPEs are initialized*  
 6255 *and the \_PRW methods are executed. This will prevent errors when the \_PRW*  
 10281 *methods attempt to access system memory or I/O space.*

10283 **GPE enable/disable no longer reads the GPE enable register. We now keep**  
 10284 **the**  
 6258 *GPE enable/disable no longer reads the GPE enable register. We now keep the*  
 10285 *enable info for runtime and wake separate and in the GPE\_EVENT\_INFO. We*  
 10286 *thus no longer depend on the hardware to maintain these bits.*

10288 Always clear the wake status and fixed/GPE status bits before sleep, even  
 10289 for state S5.

10291 Improved the AML debugger output for displaying the GPE blocks and their  
 10292 current status.

10294 **Added new strings for the \_OSI method, of the form "Windows 2001 SPx"**  
 10295 **where**  
 6268 *Added new strings for the \_OSI method, of the form "Windows 2001 SPx" where*  
 10296 *x = 0,1,2,3,4.*

10298 **Fixed a problem where the physical address was incorrectly calculated**  
 10299 **when**

10300 the Load() operator was used to directly load from an Operation Region  
 10301 (vs.  
 10302 loading from a Field object.) Also added check for minimum table length  
 10303 for  
 6271 Fixed a problem where the physical address was incorrectly calculated when  
 6272 the Load() operator was used to directly load from an Operation Region (vs.  
 6273 loading from a Field object.) Also added check for minimum table length for  
 10304 this case.

10306 Fix for multiple mutex acquisition. Restore original thread SyncLevel on  
 10307 mutex release.

10309 Added ACPI\_VALID\_SXDS flag to the AcpiGetObjectInfo interface for  
 10310 consistency with the other fields returned.

10312 Shrunk the ACPI\_GPE\_EVENT\_INFO structure by 40%. There is one such  
 10313 structure for each GPE in the system, so the size of this structure is  
 10314 important.

10316 CPU stack requirement reduction: Cleaned up the method execution and  
 10317 object  
 6286 CPU stack requirement reduction: Cleaned up the method execution and object  
 10318 evaluation paths so that now a parameter structure is passed, instead of  
 10319 copying the various method parameters over and over again.

10321 In evregion.c: Correctly exit and reenter the interpreter region if and  
 10322 only if dispatching an operation region request to a user-installed  
 10323 handler.  
 6291 only if dispatching an operation region request to a user-installed handler.  
 10324 Do not exit/reenter when dispatching to a default handler (e.g., default  
 10325 system memory or I/O handlers)

10328 Notes for updating drivers for the new GPE support. The following  
 10329 changes  
 10330 must be made to ACPI-related device drivers that are attached to one or  
 10331 more  
 10332 GPEs: (This information will be added to the ACPI CA Programmer  
 10333 Reference.)  
 6296 Notes for updating drivers for the new GPE support. The following changes  
 6297 must be made to ACPI-related device drivers that are attached to one or more  
 6298 GPEs: (This information will be added to the ACPI CA Programmer Reference.)

10335 1) AcpiInstallGpeHandler no longer automatically enables the GPE, you  
 10336 must  
 6300 1) AcpiInstallGpeHandler no longer automatically enables the GPE, you must  
 10337 explicitly call AcpiEnableGpe.  
 10338 2) There is a new interface called AcpiSetGpeType. This should be called  
 10339 before enabling the GPE. Also, this interface will automatically disable  
 10340 the GPE if it is currently enabled.  
 10341 3) AcpiEnableGpe no longer supports a GPE type flag.

10343 Specific drivers that must be changed:

10344 1) EC driver:  
 10345 AcpiInstallGpeHandler (NULL, GpeNum, ACPI\_GPE\_EDGE\_TRIGGERED,  
 10346 AeGpeHandler, NULL);  
 10347 AcpiSetGpeType (NULL, GpeNum, ACPI\_GPE\_TYPE\_RUNTIME);  
 10348 AcpiEnableGpe (NULL, GpeNum, ACPI\_NOT\_ISR);

10350 2) Button Drivers (Power, Lid, Sleep):

10351 Run \_PRW method under parent device  
 10352 If \_PRW exists: /\* This is a control-method button \*/  
 10353 Extract GPE number and possibly GpeDevice  
 10354 AcpiSetGpeType (GpeDevice, GpeNum, ACPI\_GPE\_TYPE\_WAKE\_RUN);  
 10355 AcpiEnableGpe (GpeDevice, GpeNum, ACPI\_NOT\_ISR);

10357 For all other devices that have \_PRWs, we automatically set the GPE type  
 10358 to  
 10359 ACPI\_GPE\_TYPE\_WAKE, but the GPE is NOT automatically (wake) enabled.  
 10360 This  
 10361 must be done on a selective basis, usually requiring some kind of user  
 10362 app  
 6321 For all other devices that have \_PRWs, we automatically set the GPE type to  
 6322 ACPI\_GPE\_TYPE\_WAKE, but the GPE is NOT automatically (wake) enabled. This  
 6323 must be done on a selective basis, usually requiring some kind of user app  
 10363 to allow the user to pick the wake devices.

10366 Code and Data Size: Current and previous core subsystem library sizes are  
 10367 shown below. These are the code and data sizes for the acpica.lib  
 10368 produced  
 6328 shown below. These are the code and data sizes for the acpica.lib produced  
 10369 by the Microsoft Visual C++ 6.0 compiler, and these values do not include  
 10370 any ACPI driver or OSPM code. The debug version of the code includes the  
 10371 debug output trace mechanism and has a much larger code and data size.  
 10372 Note  
 10373 that these values will vary depending on the efficiency of the compiler  
 10374 and  
 6331 debug output trace mechanism and has a much larger code and data size. Note  
 6332 that these values will vary depending on the efficiency of the compiler and  
 10375 the compiler options used during generation.

10377 Previous Release:  
 10378 Non-Debug Version: 77.0K Code, 11.4K Data, 88.4K Total  
 10379 Debug Version: 161.0K Code, 66.3K Data, 227.3K Total  
 10380 Current Release:

10382 Non-Debug Version: 77.6K Code, 11.5K Data, 89.1K Total  
 10383 Debug Version: 162.9K Code, 67.0K Data, 229.9K Total

10387 -----  
 10388 02 April 2004. Summary of changes for version 20040402:

10390 1) ACPI CA Core Subsystem:

10392 Fixed an interpreter problem where an indirect store through an ArgX  
 10393 parameter was incorrectly applying the "implicit conversion rules" during  
 10394 the store. From the ACPI specification: "If the target is a method local  
 10395 or  
 6352 the store. From the ACPI specification: "If the target is a method local or  
 10396 argument (LocalX or ArgX), no conversion is performed and the result is  
 10397 stored directly to the target". The new behavior is to disable implicit  
 10398 conversion during ALL stores to an ArgX.

10400 Changed the behavior of the \_PRW method scan to ignore any and all errors  
 10401 returned by a given \_PRW. This prevents the scan from aborting from the  
 10402 failure of any single \_PRW.

10404 Moved the runtime configuration parameters from the global init procedure  
 10405 to  
 6361 Moved the runtime configuration parameters from the global init procedure to  
 10406 static variables in acglobal.h. This will allow the host to override the  
 10407 default values easily.

10409 Code and Data Size: Current and previous core subsystem library sizes are  
 10410 shown below. These are the code and data sizes for the acpica.lib  
 10411 produced  
 6366 shown below. These are the code and data sizes for the acpica.lib produced  
 10412 by the Microsoft Visual C++ 6.0 compiler, and these values do not include  
 10413 any ACPI driver or OSPM code. The debug version of the code includes the

10414 debug output trace mechanism and has a much larger code and data size.  
 10415 **Note**  
 10416 **that these values will vary depending on the efficiency of the compiler**  
 10417 **and**  
 6369 debug output trace mechanism and has a much larger code and data size. *Note*  
 6370 *that these values will vary depending on the efficiency of the compiler and*  
 10418 the compiler options used during generation.

10420 Previous Release:  
 10421 Non-Debug Version: 76.9K Code, 11.4K Data, 88.3K Total  
 10422 Debug Version: 160.8K Code, 66.1K Data, 226.9K Total  
 10423 Current Release:  
 10424 Non-Debug Version: 77.0K Code, 11.4K Data, 88.4K Total  
 10425 Debug Version: 161.0K Code, 66.3K Data, 227.3K Total

10428 2) iASL Compiler/Disassembler:

10430 **iASL now fully disassembles SSDTs. However, External() statements are**  
 10431 **not**  
 6383 *iASL now fully disassembles SSDTs. However, External() statements are not*  
 10432 generated automatically for unresolved symbols at this time. This is a  
 10433 planned feature for future implementation.

10435 **Fixed a scoping problem in the disassembler that occurs when the type of**  
 10436 **the**  
 6387 *Fixed a scoping problem in the disassembler that occurs when the type of the*  
 10437 target of a Scope() operator is overridden. This problem caused an  
 10438 incorrectly nested internal namespace to be constructed.

10440 **Any warnings or errors that are emitted during disassembly are now**  
 10441 **commented**  
 10442 **out automatically so that the resulting file can be recompiled without**  
 10443 **any**  
 6391 *Any warnings or errors that are emitted during disassembly are now commented*  
 6392 *out automatically so that the resulting file can be recompiled without any*  
 10444 hand editing.

10446 -----  
 10447 26 March 2004. Summary of changes for version 20040326:

10449 1) ACPI CA Core Subsystem:

10451 Implemented support for "wake" GPEs via interaction between GPEs and the  
 10452 \_PRW methods. Every GPE that is pointed to by one or more \_PRWs is  
 10453 identified as a WAKE GPE and by default will no longer be enabled at  
 10454 runtime. **Previously, we were blindly enabling all GPEs with a**  
 10455 **corresponding**  
 10456 **\_Lxx or \_Exx method - but most of these turn out to be WAKE GPEs anyway.**  
 10457 **We**  
 6403 *runtime. Previously, we were blindly enabling all GPEs with a corresponding*  
 6404 *\_Lxx or \_Exx method - but most of these turn out to be WAKE GPEs anyway. We*  
 10458 believe this has been the cause of thousands of "spurious" GPEs on some  
 10459 systems.

10461 This new GPE behavior is can be reverted to the original behavior (enable  
 10462 ALL GPEs at runtime) via a runtime flag.

10464 Fixed a problem where aliased control methods could not access objects  
 10465 properly. The proper scope within the namespace was not initialized  
 10466 (transferred to the target of the aliased method) before executing the  
 10467 target method.

10469 **Fixed a potential race condition on internal object deletion on the**  
 10470 **return**  
 6416 *Fixed a potential race condition on internal object deletion on the return*

10471 object in AcpiEvaluateObject.

10473 Integrated a fix for resource descriptors where both \_MEM and \_MTP were  
 10474 being extracted instead of just \_MEM. (i.e. bitmask was incorrectly too  
 10475 wide, 0x0F instead of 0x03.)

10477 **Added a special case for ACPI\_ROOT\_OBJECT in AcpiUtGetNodeName,**  
 10478 **preventing**  
 10479 **a**  
 6423 *Added a special case for ACPI\_ROOT\_OBJECT in AcpiUtGetNodeName, preventing a*  
 10480 fault in some cases.

10482 Updated Notify() values for debug statements in evmisc.c

10484 Return proper status from AcpiUtMutexInitialize, not just simply AE\_OK.

10486 Code and Data Size: Current and previous core subsystem library sizes are  
 10487 **shown below. These are the code and data sizes for the acpica.lib**  
 10488 **produced**  
 6431 *shown below. These are the code and data sizes for the acpica.lib produced*  
 10489 by the Microsoft Visual C++ 6.0 compiler, and these values do not include  
 10490 any ACPI driver or OSPM code. The debug version of the code includes the  
 10491 **debug output trace mechanism and has a much larger code and data size.**  
 10492 **Note**  
 10493 **that these values will vary depending on the efficiency of the compiler**  
 10494 **and**  
 6434 *debug output trace mechanism and has a much larger code and data size. Note*  
 6435 *that these values will vary depending on the efficiency of the compiler and*  
 10495 the compiler options used during generation.

10497 Previous Release:

10499 Non-Debug Version: 76.5K Code, 11.3K Data, 87.8K Total  
 10500 Debug Version: 160.3K Code, 66.0K Data, 226.3K Total  
 10501 Current Release:  
 10502 Non-Debug Version: 76.9K Code, 11.4K Data, 88.3K Total  
 10503 Debug Version: 160.8K Code, 66.1K Data, 226.9K Total

10505 -----  
 10506 11 March 2004. Summary of changes for version 20040311:

10508 1) ACPI CA Core Subsystem:

10510 Fixed a problem where errors occurring during the parse phase of control  
 10511 method execution did not abort cleanly. For example, objects created and  
 10512 installed in the namespace were not deleted. This caused all subsequent  
 10513 invocations of the method to return the AE\_ALREADY\_EXISTS exception.

10515 **Implemented a mechanism to force a control method to "Serialized"**  
 10516 **execution**  
 6456 *Implemented a mechanism to force a control method to "Serialized" execution*  
 10517 if the method attempts to create namespace objects. (The root of the  
 10518 AE\_ALREADY\_EXISTS problem.)

10520 Implemented support for the predefined \_OSI "internal" control method.  
 10521 **Initial supported strings are "Linux", "Windows 2000", "Windows 2001",**  
 10522 **and**  
 10523 **"Windows 2001.1", and can be easily upgraded for new strings as**  
 10524 **necessary.**  
 6461 *Initial supported strings are "Linux", "Windows 2000", "Windows 2001", and*  
 6462 *"Windows 2001.1", and can be easily upgraded for new strings as necessary.*  
 10525 This feature will allow "other" operating systems to execute the fully  
 10526 tested, "Windows" code path through the ASL code

10528 Global Lock Support: Now allows multiple acquires and releases with any  
 10529 **internal thread. Removed concept of "owning thread" for this special**

10530 **mutex.**  
 6467 *internal thread. Removed concept of "owning thread" for this special mutex.*

10532 **Fixed two functions that were inappropriately declaring large objects on the**  
 10533 **the**  
 10534 **CPU stack: PsParseLoop, NsEvaluateRelative. Reduces the stack usage**  
 10535 **during**  
 6469 *Fixed two functions that were inappropriately declaring large objects on the*  
 6470 *CPU stack: PsParseLoop, NsEvaluateRelative. Reduces the stack usage during*  
 10536 *method execution considerably.*

10538 Fixed a problem in the ACPI 2.0 FACS descriptor (actbl2.h) where the  
 10539 S4Bios\_f field was incorrectly defined as UINT32 instead of UINT32\_BIT.

10541 Fixed a problem where AcpiEvGpeDetect would fault if there were no GPEs  
 10542 defined on the machine.

10544 **Implemented two runtime options: One to force all control method**  
 10545 **execution**  
 10546 **to "Serialized" to mimic Windows behavior, another to disable \_OSI**  
 10547 **support**  
 6479 *Implemented two runtime options: One to force all control method execution*  
 6480 *to "Serialized" to mimic Windows behavior, another to disable \_OSI support*  
 10548 *if it causes problems on a given machine.*

10550 Code and Data Size: Current and previous core subsystem library sizes are  
 10551 **shown below. These are the code and data sizes for the acpica.lib**  
 10552 **produced**  
 6484 *shown below. These are the code and data sizes for the acpica.lib produced*  
 10553 *by the Microsoft Visual C++ 6.0 compiler, and these values do not include*  
 10554 *any ACPI driver or OSPM code. The debug version of the code includes the*  
 10555 *debug output trace mechanism and has a much larger code and data size.*  
 10556 **Note**  
 10557 **that these values will vary depending on the efficiency of the compiler**  
 10558 **and**  
 6487 *debug output trace mechanism and has a much larger code and data size. Note*  
 6488 *that these values will vary depending on the efficiency of the compiler and*  
 10559 *the compiler options used during generation.*

10561	Previous Release:			
10562	Non-Debug Version:	74.8K Code,	10.1K Data,	84.9K Total
10563	Debug Version:	158.7K Code,	65.1K Data,	223.8K Total
10564	Current Release:			
10565	Non-Debug Version:	76.5K Code,	11.3K Data,	87.8K Total
10566	Debug Version:	160.3K Code,	66.0K Data,	226.3K Total

10568 2) iASL Compiler/Disassembler:

10570 Fixed an array size problem for FreeBSD that would cause the compiler to  
 10571 fault.

10573 -----  
 10574 20 February 2004. Summary of changes for version 20040220:

10577 1) ACPI CA Core Subsystem:

10579 Implemented execution of \_SxD methods for Device objects in the  
 10580 GetObjectInfo interface.

10582 Fixed calls to \_SST method to pass the correct arguments.

10584 Added a call to \_SST on wake to restore to "working" state.

10586 Check for End-Of-Buffer failure case in the WalkResources interface.

10588 Integrated fix for 64-bit alignment issue in acglobal.h by moving two  
 10589 structures to the beginning of the file.

10591 After wake, clear GPE status register(s) before enabling GPEs.

10593 **After wake, clear/enable power button. (Perhaps we should clear/enable**  
 10594 **all**  
 6523 *After wake, clear/enable power button. (Perhaps we should clear/enable all*  
 10595 *fixed events upon wake.)*

10597 Fixed a couple of possible memory leaks in the Namespace manager.

10599 Integrated latest acnetbsd.h file.

10601 -----  
 10602 11 February 2004. Summary of changes for version 20040211:

10605 1) ACPI CA Core Subsystem:

10607 Completed investigation and implementation of the call-by-reference  
 10608 mechanism for control method arguments.

10610 Fixed a problem where a store of an object into an indexed package could  
 10611 fail if the store occurs within a different method than the method that  
 10612 created the package.

10614 **Fixed a problem where the ToDecimal operator could return incorrect**  
 10615 **results.**  
 6543 *Fixed a problem where the ToDecimal operator could return incorrect results.*

10617 **Fixed a problem where the CopyObject operator could fail on some of the**  
 10618 **more**  
 6545 *Fixed a problem where the CopyObject operator could fail on some of the more*  
 10619 *obscure objects (e.g., Reference objects.)*

10621 Improved the output of the Debug object to display buffer, package, and  
 10622 index objects.

10624 **Fixed a problem where constructs of the form "RefOf (ArgX)" did not**  
 10625 **return**  
 6551 *Fixed a problem where constructs of the form "RefOf (ArgX)" did not return*  
 10626 *the expected result.*

10628 Added permanent ACPI\_REPORT\_ERROR macros for all instances of the  
 10629 ACPI\_AML\_INTERNAL exception.

10631 Integrated latest version of acfreebsd.h

10633 -----  
 10634 16 January 2004. Summary of changes for version 20040116:

10636 The purpose of this release is primarily to update the copyright years in  
 10637 each module, thus causing a huge number of diffs. There are a few small  
 10638 functional changes, however.

10640 1) ACPI CA Core Subsystem:

10642 **Improved error messages when there is a problem finding one or more of**  
 10643 **the**  
 6568 *Improved error messages when there is a problem finding one or more of the*  
 10644 *required base ACPI tables*

10646 Reintroduced the definition of APIC\_HEADER in actbl.h

10648 Changed definition of MADT\_ADDRESS\_OVERRIDE to 64 bits (actbl.h)



10650 Removed extraneous reference to NewObj in dsmthdat.c

10652 2) iASL compiler

10654 **Fixed a problem introduced in December that disabled the correct**  
 10655 **disassembly**  
 10656 *Fixed a problem introduced in December that disabled the correct disassembly*  
 10656 of Resource Templates

10659 -----  
 10660 03 December 2003. Summary of changes for version 20031203:

10662 1) ACPI CA Core Subsystem:

10664 Changed the initialization of Operation Regions during subsystem  
 10665 init to perform two entire walks of the ACPI namespace; The first  
 10666 to initialize the regions themselves, the second to execute the  
 10667 \_REG methods. This fixed some interdependencies across \_REG  
 10668 methods found on some machines.

10670 Fixed a problem where a Store(Local0, Local1) could simply update  
 10671 the object reference count, and not create a new copy of the  
 10672 object if the Local1 is uninitialized.

10674 Implemented support for the \_SST reserved method during sleep  
 10675 transitions.

10677 Implemented support to clear the SLP\_TYP and SLP\_EN bits when  
 10678 waking up, this is apparently required by some machines.

10680 When sleeping, clear the wake status only if SleepState is not S5.

10682 Fixed a problem in AcpiRsExtendedIrqResource() where an incorrect  
 10683 pointer arithmetic advanced a string pointer too far.

10685 Fixed a problem in AcpiTbGetTablePtr() where a garbage pointer  
 10686 could be returned if the requested table has not been loaded.

10688 Within the support for IRQ resources, restructured the handling of  
 10689 the active and edge/level bits.

10691 Fixed a few problems in AcpiPsxExecute() where memory could be  
 10692 leaked under certain error conditions.

10694 Improved error messages for the cases where the ACPI mode could  
 10695 not be entered.

10697 Code and Data Size: Current and previous core subsystem library  
 10698 sizes are shown below. These are the code and data sizes for the  
 10699 apica.lib produced by the Microsoft Visual C++ 6.0 compiler, and  
 10700 these values do not include any ACPI driver or OSPM code. The  
 10701 debug version of the code includes the debug output trace  
 10702 mechanism and has a much larger code and data size. Note that  
 10703 these values will vary depending on the efficiency of the compiler  
 10704 and the compiler options used during generation.

10706 Previous Release (20031029):  
 10707 Non-Debug Version: 74.4K Code, 10.1K Data, 84.5K Total  
 10708 Debug Version: 158.3K Code, 65.0K Data, 223.3K Total  
 10709 Current Release:  
 10710 Non-Debug Version: 74.8K Code, 10.1K Data, 84.9K Total  
 10711 Debug Version: 158.7K Code, 65.1K Data, 223.8K Total

10713 2) iASL Compiler/Disassembler:

10715 Implemented a fix for the iASL disassembler where a bad index was  
 10716 generated. This was most noticeable on 64-bit platforms

10719 -----  
 10720 29 October 2003. Summary of changes for version 20031029:

10722 1) ACPI CA Core Subsystem:

10725 Fixed a problem where a level-triggered GPE with an associated  
 10726 \_Lxx control method was incorrectly cleared twice.

10728 Fixed a problem with the Field support code where an access can  
 10729 occur beyond the end-of-region if the field is non-aligned but  
 10730 extends to the very end of the parent region (resulted in an  
 10731 AE\_AML\_REGION\_LIMIT exception.)

10733 Fixed a problem with ACPI Fixed Events where an RT Clock handler  
 10734 would not get invoked on an RTC event. The RTC event bitmasks for  
 10735 the PM1 registers were not being initialized properly.

10737 Implemented support for executing \_STA and \_INI methods for  
 10738 Processor objects. Although this is currently not part of the  
 10739 ACPI specification, there is existing ASL code that depends on the  
 10740 init-time execution of these methods.

10742 Implemented and deployed a GetDescriptorName function to decode  
 10743 the various types of internal descriptors. Guards against null  
 10744 descriptors during debug output also.

10746 Implemented and deployed a GetNodeName function to extract the 4-  
 10747 character namespace node name. This function simplifies the debug  
 10748 and error output, as well as guarding against null pointers during  
 10749 output.

10751 Implemented and deployed the ACPI\_FORMAT\_UINT64 helper macro to  
 10752 simplify the debug and error output of 64-bit integers. This  
 10753 macro replaces the HIDWORD and LODWORD macros for dumping these  
 10754 integers.

10756 Updated the implementation of the Stall() operator to only call  
 10757 AcpiOsStall(), and also return an error if the operand is larger  
 10758 than 255. This preserves the required behavior of not  
 10759 relinquishing the processor, as would happen if AcpiOsSleep() was  
 10760 called for "long stalls".

10762 Constructs of the form "Store(LocalX,LocalX)" where LocalX is not  
 10763 initialized are now treated as NOOPs.

10765 Cleaned up a handful of warnings during 64-bit generation.

10767 Fixed a reported error where and incorrect GPE number was passed  
 10768 to the GPE dispatch handler. This value is only used for error  
 10769 output, however. Used this opportunity to clean up and streamline  
 10770 the GPE dispatch code.

10772 Code and Data Size: Current and previous core subsystem library  
 10773 sizes are shown below. These are the code and data sizes for the  
 10774 apica.lib produced by the Microsoft Visual C++ 6.0 compiler, and  
 10775 these values do not include any ACPI driver or OSPM code. The

10777 debug version of the code includes the debug output trace  
 10778 mechanism and has a much larger code and data size. Note that  
 10779 these values will vary depending on the efficiency of the compiler

10780 and the compiler options used during generation.

```
10782 Previous Release (20031002):
10783   Non-Debug Version:  74.1K Code,   9.7K Data,   83.8K Total
10784   Debug Version:     157.9K Code,  64.8K Data,  222.7K Total
10785 Current Release:
10786   Non-Debug Version:  74.4K Code,  10.1K Data,   84.5K Total
10787   Debug Version:     158.3K Code,  65.0K Data,  223.3K Total
```

10790 2) iASL Compiler/Disassembler:

10792 Updated the iASL compiler to return an error if the operand to the  
10793 Stall() operator is larger than 255.

```
10796 -----
10797 02 October 2003.  Summary of changes for version 20031002:
```

10800 1) ACPI CA Core Subsystem:

10802 Fixed a problem with Index Fields where the index was not  
10803 incremented for fields that require multiple writes to the  
10804 index/data registers (Fields that are wider than the data  
10805 register.)

10807 Fixed a problem with all Field objects where a write could go  
10808 beyond the end-of-field if the field was larger than the access  
10809 granularity and therefore required multiple writes to complete the  
10810 request. An extra write beyond the end of the field could happen  
10811 inadvertently.

10813 Fixed a problem with Index Fields where a BUFFER\_OVERFLOW error  
10814 would incorrectly be returned if the width of the Data Register  
10815 was larger than the specified field access width.

10817 Completed fixes for LoadTable() and Unload() and verified their  
10818 operation. Implemented full support for the "DdbHandle" object  
10819 throughout the ACPI CA subsystem.

10821 Implemented full support for the MADT and ECDT tables in the ACPI  
10822 CA header files. Even though these tables are not directly  
10823 consumed by ACPI CA, the header definitions are useful for ACPI  
10824 device drivers.

10826 Integrated resource descriptor fixes posted to the Linux ACPI  
10827 list. This included checks for minimum descriptor length, and  
10828 support for trailing NULL strings within descriptors that have  
10829 optional string elements.

10831 Code and Data Size: Current and previous core subsystem library  
10832 sizes are shown below. These are the code and data sizes for the  
10833 apica.lib produced by the Microsoft Visual C++ 6.0 compiler, and  
10834 these values do not include any ACPI driver or OSPM code. The  
10835 debug version of the code includes the debug output trace  
10836 mechanism and has a much larger code and data size. Note that  
10837 these values will vary depending on the efficiency of the compiler  
10838 and the compiler options used during generation.

```
10840 Previous Release (20030918):
10841   Non-Debug Version:  73.9K Code,   9.7K Data,   83.6K Total
10842   Debug Version:     157.3K Code,  64.5K Data,  221.8K Total
10843 Current Release:
10844   Non-Debug Version:  74.1K Code,   9.7K Data,   83.8K Total
10845   Debug Version:     157.9K Code,  64.8K Data,  222.7K Total
```

10848 2) iASL Compiler:

10850 Implemented detection of non-ASCII characters within the input  
10851 source ASL file. This catches attempts to compile binary (AML)  
10852 files early in the compile, with an informative error message.

10854 Fixed a problem where the disassembler would fault if the output  
10855 filename could not be generated or if the output file could not be  
10856 opened.

```
10858 -----
10859 18 September 2003.  Summary of changes for version 20030918:
```

10862 1) ACPI CA Core Subsystem:

10864 Found and fixed a longstanding problem with the late execution of  
10865 the various deferred AML opcodes (such as Operation Regions,  
10866 Buffer Fields, Buffers, and Packages). If the name string  
10867 specified for the name of the new object placed the object in a  
10868 scope other than the current scope, the initialization/execution  
10869 of the opcode failed. The solution to this problem was to  
10870 implement a mechanism where the late execution of such opcodes  
10871 does not attempt to lookup/create the name a second time in an  
10872 incorrect scope. This fixes the "region size computed  
10873 incorrectly" problem.

10875 Fixed a call to AcpiHwRegisterWrite in hwregs.c that was causing a  
10876 Global Lock AE\_BAD\_PARAMETER error.

10878 Fixed several 64-bit issues with prototypes, casting and data  
10879 types.

10881 Removed duplicate prototype from acdisasm.h

10883 Fixed an issue involving EC Operation Region Detach (Shaohua Li)

10885 Code and Data Size: Current and previous core subsystem library  
10886 sizes are shown below. These are the code and data sizes for the  
10887 apica.lib produced by the Microsoft Visual C++ 6.0 compiler, and  
10888 these values do not include any ACPI driver or OSPM code. The  
10889 debug version of the code includes the debug output trace  
10890 mechanism and has a much larger code and data size. Note that  
10891 these values will vary depending on the efficiency of the compiler  
10892 and the compiler options used during generation.

10894 Previous Release:

```
10896   Non-Debug Version:  73.7K Code,   9.7K Data,   83.4K Total
10897   Debug Version:     156.9K Code,  64.2K Data,  221.1K Total
10898 Current Release:
10899   Non-Debug Version:  73.9K Code,   9.7K Data,   83.6K Total
10900   Debug Version:     157.3K Code,  64.5K Data,  221.8K Total
```

10903 2) Linux:

10905 Fixed the AcpiOsSleep implementation in osunixxf.c to pass the  
10906 correct sleep time in seconds.

```
10908 -----
10909 14 July 2003.  Summary of changes for version 20030619:
```

10911 1) ACPI CA Core Subsystem:

10913 Parse SSDTs in order discovered, as opposed to reverse order  
 10914 (Hrvoje Habjanic)

10916 Fixes from FreeBSD and NetBSD. (Frank van der Linden, Thomas  
 10917 Klausner,  
 10918 Nate Lawson)

10921 2) Linux:

10923 Dynamically allocate SDT list (suggested by Andi Kleen)

10925 proc function return value cleanups (Andi Kleen)

10927 Correctly handle NMI watchdog during long stalls (Andrew Morton)

10929 Make it so acpimp=force works (reported by Andrew Morton)

10932 -----  
 10933 19 June 2003. Summary of changes for version 20030619:

10935 1) ACPI CA Core Subsystem:

10937 Fix To/FromBCD, eliminating the need for an arch-specific #define.

10939 Do not acquire a semaphore in the S5 shutdown path.

10941 Fix ex\_digits\_needed for 0. (Takayoshi Kochi)

10943 Fix sleep/stall code reversal. (Andi Kleen)

10945 Revert a change having to do with control method calling  
 10946 semantics.

10948 2) Linux:

10950 acpiphp update (Takayoshi Kochi)

10952 Export acpi\_disabled for sonypi (Stelian Pop)

10954 Mention acpimp=force in config help

10956 Re-add acpitables.c and acpimp=force. This improves backwards  
 10958 compatibility and also cleans up the code to a significant degree.

10960 Add ASUS Value-add driver (Karol Kozimor and Julien Lerouge)

10962 -----  
 10963 22 May 2003. Summary of changes for version 20030522:

10965 1) ACPI CA Core Subsystem:

10967 Found and fixed a reported problem where an AE\_NOT\_FOUND error  
 10968 occurred occasionally during \_BST evaluation. This turned out to  
 10969 be an Owner ID allocation issue where a called method did not get  
 10970 a new ID assigned to it. Eventually, (after 64k calls), the Owner  
 10971 ID UINT16 would wraparound so that the ID would be the same as the  
 10972 caller's and the called method would delete the caller's  
 10973 namespace.

10975 Implemented extended error reporting for control methods that are  
 10976 aborted due to a run-time exception. Output includes the exact  
 10977 AML instruction that caused the method abort, a dump of the method

10978 locals and arguments at the time of the abort, and a trace of all  
 10979 nested control method calls.

10981 Modified the interpreter to allow the creation of buffers of zero  
 10982 length from the AML code. Implemented new code to ensure that no  
 10983 attempt is made to actually allocate a memory buffer (of length  
 10984 zero) - instead, a simple buffer object with a NULL buffer pointer  
 10985 and length zero is created. A warning is no longer issued when  
 10986 the AML attempts to create a zero-length buffer.

10988 Implemented a workaround for the "leading asterisk issue" in  
 10989 \_HIDs, \_UIDs, and \_CIDs in the AML interpreter. One leading  
 10990 asterisk is automatically removed if present in any HID, UID, or  
 10991 CID strings. The iASL compiler will still flag this asterisk as  
 10992 an error, however.

10994 Implemented full support for \_CID methods that return a package of  
 10995 multiple CIDs (Compatible IDs). The AcpiGetObjectInfo() interface  
 10996 now additionally returns a device \_CID list if present. This  
 10997 required a change to the external interface in order to pass an  
 10998 ACPI\_BUFFER object as a parameter since the \_CID list is of  
 10999 variable length.

11001 Fixed a problem with the new AE\_SAME\_HANDLER exception where  
 11002 handler initialization code did not know about this exception.

11004 Code and Data Size: Current and previous core subsystem library  
 11005 sizes are shown below. These are the code and data sizes for the  
 11006 acpica.lib produced by the Microsoft Visual C++ 6.0 compiler, and  
 11007 these values do not include any ACPI driver or OSPM code. The  
 11008 debug version of the code includes the debug output trace  
 11009 mechanism and has a much larger code and data size. Note that  
 11010 these values will vary depending on the efficiency of the compiler  
 11011 and the compiler options used during generation.

11013 Previous Release (20030509):  
 11014 Non-Debug Version: 73.4K Code, 9.7K Data, 83.1K Total  
 11015 Debug Version: 156.1K Code, 63.9K Data, 220.0K Total  
 11016 Current Release:  
 11017 Non-Debug Version: 73.7K Code, 9.7K Data, 83.4K Total  
 11018 Debug Version: 156.9K Code, 64.2K Data, 221.1K Total

11021 2) Linux:

11023 Fixed a bug in which we would reinitialize the ACPI interrupt  
 11024 after it was already working, thus disabling all ACPI and the IRQs  
 11025 for any other device sharing the interrupt. (Thanks to Stian  
 11026 Jordet)

11028 Toshiba driver update (John Belmonte)

11030 Return only 0 or 1 for our interrupt handler status (Andrew  
 11031 Morton)

11034 3) iASL Compiler:

11036 Fixed a reported problem where multiple (nested) ElseIf()  
 11037 statements were not handled correctly by the compiler, resulting  
 11038 in incorrect warnings and incorrect AML code. This was a problem  
 11039 in both the ASL parser and the code generator.

11042 4) Documentation:

11044 Added changes to existing interfaces, new exception codes, and new  
 11045 text concerning reference count object management versus garbage  
 11046 collection.

11048 -----

11049 09 May 2003. Summary of changes for version 20030509.

#### 11052 1) ACPI CA Core Subsystem:

11054 Changed the subsystem initialization sequence to hold off  
 11055 installation of address space handlers until the hardware has been  
 11056 initialized and the system has entered ACPI mode. This is because  
 11057 the installation of space handlers can cause \_REG methods to be  
 11058 run. Previously, the \_REG methods could potentially be run before  
 11059 ACPI mode was enabled.

11061 Fixed some memory leak issues related to address space handler and  
 11062 notify handler installation. There were some problems with the  
 11063 reference count mechanism caused by the fact that the handler  
 11064 objects are shared across several namespace objects.

11066 Fixed a reported problem where reference counts within the  
 11067 namespace were not properly updated when named objects created by  
 11068 method execution were deleted.

11070 Fixed a reported problem where multiple SSDTs caused a deletion  
 11071 issue during subsystem termination. Restructured the table data  
 11072 structures to simplify the linked lists and the related code.

11074 Fixed a problem where the table ID associated with secondary  
 11075 tables (SSDTs) was not being propagated into the namespace objects  
 11076 created by those tables. This would only present a problem for  
 11077 tables that are unloaded at run-time, however.

11079 Updated AcpiOsReadable and AcpiOsWritable to use the ACPI\_SIZE  
 11080 type as the length parameter (instead of UINT32).

11082 Solved a long-standing problem where an ALREADY\_EXISTS error  
 11083 appears on various systems. This problem could happen when there  
 11084 are multiple PCI\_Config operation regions under a single PCI root  
 11085 bus. This doesn't happen very frequently, but there are some  
 11086 systems that do this in the ASL.

11088 Fixed a reported problem where the internal DeleteNode function  
 11089 was incorrectly handling the case where a namespace node was the  
 11090 first in the parent's child list, and had additional peers (not  
 11091 the only child, but first in the list of children.)

11093 Code and Data Size: Current core subsystem library sizes are shown  
 11094 below. These are the code and data sizes for the acpica.lib  
 11095 produced by the Microsoft Visual C++ 6.0 compiler, and these  
 11096 values do not include any ACPI driver or OSPM code. The debug  
 11097 version of the code includes the debug output trace mechanism and  
 11098 has a much larger code and data size. Note that these values will  
 11099 vary depending on the efficiency of the compiler and the compiler  
 11100 options used during generation.

11102	Previous Release			
11103	Non-Debug Version:	73.7K Code,	9.5K Data,	83.2K Total
11104	Debug Version:	156.1K Code,	63.6K Data,	219.7K Total
11105	Current Release:			
11106	Non-Debug Version:	73.4K Code,	9.7K Data,	83.1K Total
11107	Debug Version:	156.1K Code,	63.9K Data,	220.0K Total

11110 2) Linux:

11112 Allow ":" in OS override string (Ducrot Bruno)

11114 Kobject fix (Greg KH)

#### 11117 3 iASL Compiler/Disassembler:

11119 Fixed a problem in the generation of the C source code files (AML  
 11120 is emitted in C source statements for BIOS inclusion) where the  
 11121 Ascii dump that appears within a C comment at the end of each line  
 11122 could cause a compile time error if the AML sequence happens to  
 11123 have an open comment or close comment sequence embedded.

11126 -----

11127 24 April 2003. Summary of changes for version 20030424.

#### 11130 1) ACPI CA Core Subsystem:

11132 Support for big-endian systems has been implemented. Most of the  
 11133 support has been invisibly added behind big-endian versions of the  
 11134 ACPI\_MOVE\_\* macros.

11136 Fixed a problem in AcpiHwDisableGpeBlock() and  
 11137 AcpiHwClearGpeBlock() where an incorrect offset was passed to the  
 11138 low level hardware write routine. The offset parameter was  
 11139 actually eliminated from the low level read/write routines because  
 11140 they had become obsolete.

11142 Fixed a problem where a handler object was deleted twice during  
 11143 the removal of a fixed event handler.

11146 2) Linux:

11148 A fix for SMP systems with link devices was contributed by

11150 Compaq's Dan Zink.

11152 (2.5) Return whether we handled the interrupt in our IRQ handler.  
 11153 (Linux ISRs no longer return void, so we can propagate the handler  
 11154 return value from the ACPI CA core back to the OS.)

#### 11158 3) Documentation:

11160 The ACPI CA Programmer Reference has been updated to reflect new  
 11161 interfaces and changes to existing interfaces.

11163 -----

11164 28 March 2003. Summary of changes for version 20030328.

#### 11166 1) ACPI CA Core Subsystem:

11168 The GPE Block Device support has been completed. New interfaces  
 11169 are AcpiInstallGpeBlock and AcpiRemoveGpeBlock. The Event  
 11170 interfaces (enable, disable, clear, getstatus) have been split  
 11171 into separate interfaces for Fixed Events and General Purpose  
 11172 Events (GPEs) in order to support GPE Block Devices properly.

11174 Fixed a problem where the error message "Failed to acquire  
 11175 semaphore" would appear during operations on the embedded

11176 controller (EC).

11178 Code and Data Size: Current core subsystem library sizes are shown  
11179 below. These are the code and data sizes for the acpica.lib  
11180 produced by the Microsoft Visual C++ 6.0 compiler, and these  
11181 values do not include any ACPI driver or OSPM code. The debug  
11182 version of the code includes the debug output trace mechanism and  
11183 has a much larger code and data size. Note that these values will  
11184 vary depending on the efficiency of the compiler and the compiler  
11185 options used during generation.

11187	Previous Release			
11188	Non-Debug Version:	72.3K Code,	9.5K Data,	81.8K Total
11189	Debug Version:	154.0K Code,	63.4K Data,	217.4K Total
11190	Current Release:			
11191	Non-Debug Version:	73.7K Code,	9.5K Data,	83.2K Total
11192	Debug Version:	156.1K Code,	63.6K Data,	219.7K Total

11195 -----  
11196 28 February 2003. Summary of changes for version 20030228.

11199 1) ACPI CA Core Subsystem:

11201 The GPE handling and dispatch code has been completely overhauled  
11202 in preparation for support of GPE Block Devices (ID ACPI0006).  
11203 This affects internal data structures and code only; there should  
11204 be no differences visible externally. One new file has been  
11205 added, evgpeblk.c

11207 The FADT fields GPE0\_BLK\_LEN and GPE1\_BLK\_LEN are now the only  
11208 fields that are used to determine the GPE block lengths. The  
11209 REGISTER\_BIT\_WIDTH field of the X\_GPEX\_BLK extended address  
11210 structures are ignored. This is per the ACPI specification but it  
11211 isn't very clear. The full 256 Block 0/1 GPEs are now supported  
11212 (the use of REGISTER\_BIT\_WIDTH limited the number of GPEs to 128).

11214 In the SCI interrupt handler, removed the read of the PM1\_CONTROL  
11215 register to look at the SCI\_EN bit. On some machines, this read  
11216 causes an SMI event and greatly slows down SCI events. (This may  
11217 in fact be the cause of slow battery status response on some  
11218 systems.)

11220 Fixed a problem where a store of a NULL string to a package object  
11221 could cause the premature deletion of the object. This was seen  
11222 during execution of the battery\_BIF method on some systems,  
11223 resulting in no battery data being returned.

11225 Added AcpiWalkResources interface to simplify parsing of resource  
11226 lists.

11228 Code and Data Size: Current core subsystem library sizes are shown  
11229 below. These are the code and data sizes for the acpica.lib  
11230 produced by the Microsoft Visual C++ 6.0 compiler, and these  
11231 values do not include any ACPI driver or OSPM code. The debug  
11232 version of the code includes the debug output trace mechanism and  
11233 has a much larger code and data size. Note that these values will  
11234 vary depending on the efficiency of the compiler and the compiler  
11235 options used during generation.

11237	Previous Release			
11238	Non-Debug Version:	72.0K Code,	9.5K Data,	81.5K Total
11239	Debug Version:	153.0K Code,	62.9K Data,	215.9K Total
11240	Current Release:			
11241	Non-Debug Version:	72.3K Code,	9.5K Data,	81.8K Total

11242 Debug Version: 154.0K Code, 63.4K Data, 217.4K Total

11245 2) Linux

11247 S3 fixes (Ole Rohne)

11249 Update ACPI PHP driver with to use new acpi\_walk\_resource API  
11250 (Bjorn Helgaas)

11252 Add S4BIOS support (Pavel Machek)

11254 Map in entire table before performing checksum (John Stultz)

11256 Expand the mem= cmdline to allow the specification of reserved and  
11257 ACPI DATA blocks (Pavel Machek)

11259 Never use ACPI on VISWS

11261 Fix derive\_pci\_id (Ducrot Bruno, Alvaro Lopez)

11263 Revert a change that allowed P\_BLK lengths to be 4 or 5. This is  
11264 causing us to think that some systems support C2 when they really  
11265 don't.

11267 Do not count processor objects for non-present CPUs (Thanks to  
11268 Dominik Brodowski)

11271 3) iASL Compiler:

11273 Fixed a problem where ASL include files could not be found and  
11274 opened.

11276 Added support for the \_PDC reserved name.

11279 -----  
11280 22 January 2003. Summary of changes for version 20030122.

11283 1) ACPI CA Core Subsystem:

11285 Added a check for constructs of the form: Store (Local0, Local0)  
11286 where Local0 is not initialized. Apparently, some BIOS  
11287 programmers believe that this is a NOOP. Since this store doesn't  
11288 do anything anyway, the new prototype behavior will ignore this  
11289 error. This is a case where we can relax the strict checking in  
11290 the interpreter in the name of compatibility.

11293 2) Linux

11295 The AcpiSrc Source Conversion Utility has been released with the  
11296 Linux package for the first time. This is the utility that is  
11297 used to convert the ACPI CA base source code to the Linux version.

11299 (Both) Handle P\_BLK lengths shorter than 6 more gracefully

11301 (Both) Move more headers to include/acpi, and delete an unused  
11302 header.

11304 (Both) Move drivers/acpi/include directory to include/acpi

11306 (Both) Boot functions don't use cmdline, so don't pass it around

11308 (Both) Remove include of unused header (Adrian Bunk)

11310 (Both) acpiphp.h includes both linux/acpi.h and acpi\_bus.h. Since  
11311 the  
11312 former now also includes the latter, acpiphp.h only needs the one,  
11313 now.

11315 (2.5) Make it possible to select method of bios restoring after S3  
11316 resume. [=> no more ugly ifdefs] (Pavel Machek)

11318 (2.5) Make proc write interfaces work (Pavel Machek)

11320 (2.5) Properly init/clean up in cpufreq/acpi (Dominik Brodowski)

11322 (2.5) Break out ACPI Perf code into its own module, under cpufreq  
11323 (Dominik Brodowski)

11325 (2.4) S4BIOS support (Ducrot Bruno)

11327 (2.4) Fix acpiphp\_glue.c for latest ACPI struct changes (Sergio  
11328 Visinoni)

11331 3) iASL Compiler:

11333 Added support to disassemble SSDT and PSDTs.

11335 Implemented support to obtain SSDTs from the Windows registry if  
11336 available.

11339 -----  
11340 09 January 2003. Summary of changes for version 20030109.

11342 1) ACPI CA Core Subsystem:

11344 Changed the behavior of the internal Buffer-to-String conversion  
11345 function. The current ACPI specification states that the contents  
11346 of the buffer are "converted to a string of two-character  
11347 hexadecimal numbers, each separated by a space". Unfortunately,  
11348 this definition is not backwards compatible with existing ACPI 1.0  
11349 implementations (although the behavior was not defined in the ACPI  
11350 1.0 specification). The new behavior simply copies data from the  
11351 buffer to the string until a null character is found or the end of  
11352 the buffer is reached. The new String object is always null  
11353 terminated. This problem was seen during the generation of \_BIF  
11354 battery data where incorrect strings were returned for battery  
11355 type, etc. This will also require an errata to the ACPI  
11356 specification.

11358 Renamed all instances of NATIVE\_UINT and NATIVE\_INT to  
11359 ACPI\_NATIVE\_UINT and ACPI\_NATIVE\_INT, respectively.

11361 Copyright in all module headers (both Linux and non-Linux) has be  
11362 updated to 2003.

11364 Code and Data Size: Current core subsystem library sizes are shown  
11365 below. These are the code and data sizes for the acpica.lib  
11366 produced by the Microsoft Visual C++ 6.0 compiler, and these  
11367 values do not include any ACPI driver or OSPM code. The debug  
11368 version of the code includes the debug output trace mechanism and  
11369 has a much larger code and data size. Note that these values will  
11370 vary depending on the efficiency of the compiler and the compiler  
11371 options used during generation.

11373 Previous Release

11374 Non-Debug Version: 72.0K Code, 9.5K Data, 81.5K Total  
11375 Debug Version: 153.0K Code, 62.9K Data, 215.9K Total  
11376 Current Release:  
11377 Non-Debug Version: 72.0K Code, 9.5K Data, 81.5K Total  
11378 Debug Version: 153.0K Code, 62.9K Data, 215.9K Total

11381 2) Linux

11383 Fixed an oops on module insertion/removal (Matthew Tippet)

11385 (2.4) Fix to handle dynamic size of mp\_irqs (Joerg Prante)

11387 (2.5) Replace pr\_debug (Randy Dunlap)

11389 (2.5) Remove usage of CPUFREQ\_ALL\_CPUS (Dominik Brodowski)

11391 (Both) Eliminate spawning of thread from timer callback, in favor  
11392 of schedule\_work()

11394 (Both) Show Lid status in /proc (Zdenek OGAR Skalak)

11396 (Both) Added define for Fixed Function HW region (Matthew Wilcox)

11398 (Both) Add missing statics to button.c (Pavel Machek)

11400 Several changes have been made to the source code translation  
11401 utility that generates the Linux Code in order to make the code  
11402 more "Linux-like":

11404 All typedefs on structs and unions have been removed in keeping  
11405 with the Linux coding style.

11407 Removed the non-Linux SourceSafe module revision number from each  
11408 module header.

11410 Completed major overhaul of symbols to be lowercased for linux.  
11411 Doubled the number of symbols that are lowercased.

11413 Fixed a problem where identifiers within procedure headers and  
11414 within quotes were not fully lower cased (they were left with a  
11415 starting capital.)

11417 Some C macros whose only purpose is to allow the generation of 16-  
11418 bit code are now completely removed in the Linux code, increasing  
11419 readability and maintainability.

11421 -----

11423 12 December 2002. Summary of changes for version 20021212.

11426 1) ACPI CA Core Subsystem:

11428 Fixed a problem where the creation of a zero-length AML Buffer  
11429 would cause a fault.

11431 Fixed a problem where a Buffer object that pointed to a static AML  
11432 buffer (in an ACPI table) could inadvertently be deleted, causing  
11433 memory corruption.

11435 Fixed a problem where a user buffer (passed in to the external  
11436 ACPI CA interfaces) could be overwritten if the buffer was too  
11437 small to complete the operation, causing memory corruption.

11439 Fixed a problem in the Buffer-to-String conversion code where a

11440 string of length one was always returned, regardless of the size  
11441 of the input Buffer object.

11443 Removed the NATIVE\_CHAR data type across the entire source due to  
11444 lack of need and lack of consistent use.

11446 Code and Data Size: Current core subsystem library sizes are shown  
11447 below. These are the code and data sizes for the acpica.lib  
11448 produced by the Microsoft Visual C++ 6.0 compiler, and these  
11449 values do not include any ACPI driver or OSPM code. The debug  
11450 version of the code includes the debug output trace mechanism and  
11451 has a much larger code and data size. Note that these values will  
11452 vary depending on the efficiency of the compiler and the compiler  
11453 options used during generation.

11455	Previous Release			
11456	Non-Debug Version:	72.1K Code,	9.5K Data,	81.6K Total
11457	Debug Version:	152.7K Code,	62.7K Data,	215.4K Total
11458	Current Release:			
11459	Non-Debug Version:	72.0K Code,	9.5K Data,	81.5K Total
11460	Debug Version:	153.0K Code,	62.9K Data,	215.9K Total

11463 -----  
11464 05 December 2002. Summary of changes for version 20021205.

#### 11466 1) ACPI CA Core Subsystem:

11468 Fixed a problem where a store to a String or Buffer object could  
11469 cause corruption of the DSDT if the object type being stored was  
11470 the same as the target object type and the length of the object  
11471 being stored was equal to or smaller than the original (existing)  
11472 target object. This was seen to cause corruption of battery \_BIF  
11473 buffers if the \_BIF method modified the buffer on the fly.

11475 Fixed a problem where an internal error was generated if a control  
11476 method invocation was used in an OperationRegion, Buffer, or  
11477 Package declaration. This was caused by the deferred parsing of  
11478 the control method and thus the deferred creation of the internal  
11479 method object. The solution to this problem was to create the  
11480 internal method object at the moment the method is encountered in  
11481 the first pass - so that subsequent references to the method will  
11482 be able to obtain the required parameter count and thus properly  
11483 parse the method invocation. This problem presented itself as an  
11484 AE\_AML\_INTERNAL during the pass 1 parse phase during table load.

11486 Fixed a problem where the internal String object copy routine did  
11487 not always allocate sufficient memory for the target String object  
11488 and caused memory corruption. This problem was seen to cause  
11489 "Allocation already present in list!" errors as memory allocation  
11490 became corrupted.

11492 Implemented a new function for the evaluation of namespace objects  
11493 that allows the specification of the allowable return object  
11494 types. This simplifies a lot of code that checks for a return  
11495 object of one or more specific objects returned from the  
11496 evaluation (such as \_STA, etc.) This may become an external  
11497 function if it would be useful to ACPI-related drivers.

11499 Completed another round of prefixing #defines with "ACPI\_" for  
11500 clarity.

11502 Completed additional code restructuring to allow more modular  
11503 linking for iASL compiler and AcpiExec. Several files were split  
11504 creating new files. New files: nsparse.c dsinit.c evgpe.c

11506 Implemented an abort mechanism to terminate an executing control  
11507 method via the AML debugger. This feature is useful for debugging  
11508 control methods that depend (wait) for specific hardware  
11509 responses.

11511 Code and Data Size: Current core subsystem library sizes are shown  
11512 below. These are the code and data sizes for the acpica.lib  
11513 produced by the Microsoft Visual C++ 6.0 compiler, and these  
11514 values do not include any ACPI driver or OSPM code. The debug  
11515 version of the code includes the debug output trace mechanism and  
11516 has a much larger code and data size. Note that these values will  
11517 vary depending on the efficiency of the compiler and the compiler  
11518 options used during generation.

11520	Previous Release			
11521	Non-Debug Version:	71.4K Code,	9.0K Data,	80.4K Total
11522	Debug Version:	152.9K Code,	63.3K Data,	216.2K Total
11523	Current Release:			
11524	Non-Debug Version:	72.1K Code,	9.5K Data,	81.6K Total
11525	Debug Version:	152.7K Code,	62.7K Data,	215.4K Total

#### 11528 2) iASL Compiler/Disassembler

11530 Fixed a compiler code generation problem for "Interrupt" Resource  
11531 Descriptors. If specified in the ASL, the optional "Resource  
11532 Source Index" and "Resource Source" fields were not inserted into  
11533 the correct location within the AML resource descriptor, creating  
11534 an invalid descriptor.

11536 Fixed a disassembler problem for "Interrupt" resource descriptors.  
11537 The optional "Resource Source Index" and "Resource Source" fields  
11538 were ignored.

11541 -----  
11542 22 November 2002. Summary of changes for version 20021122.

#### 11545 1) ACPI CA Core Subsystem:

11547 Fixed a reported problem where an object stored to a Method Local  
11548 or Arg was not copied to a new object during the store - the  
11549 object pointer was simply copied to the Local/Arg. This caused  
11550 all subsequent operations on the Local/Arg to also affect the  
11551 original source of the store operation.

11553 Fixed a problem where a store operation to a Method Local or Arg  
11554 was not completed properly if the Local/Arg contained a reference  
11555 (from RefOf) to a named field. The general-purpose store-to-  
11556 namespace-node code is now used so that this case is handled  
11557 automatically.

11559 Fixed a problem where the internal object copy routine would cause  
11560 a protection fault if the object being copied was a Package and  
11561 contained either 1) a NULL package element or 2) a nested sub-  
11562 package.

11564 Fixed a problem with the GPE initialization that resulted from an  
11565 ambiguity in the ACPI specification. One section of the  
11566 specification states that both the address and length of the GPE  
11567 block must be zero if the block is not supported. Another section  
11568 implies that only the address need be zero if the block is not  
11569 supported. The code has been changed so that both the address and  
11570 the length must be non-zero to indicate a valid GPE block (i.e.,  
11571 if either the address or the length is zero, the GPE block is

11572 invalid.)

11574 Code and Data Size: Current core subsystem library sizes are shown  
11575 below. These are the code and data sizes for the acpica.lib  
11576 produced by the Microsoft Visual C++ 6.0 compiler, and these  
11577 values do not include any ACPI driver or OSPM code. The debug  
11578 version of the code includes the debug output trace mechanism and  
11579 has a much larger code and data size. Note that these values will  
11580 vary depending on the efficiency of the compiler and the compiler  
11581 options used during generation.

11583	Previous Release			
11584	Non-Debug Version:	71.3K Code,	9.0K Data,	80.3K Total
11585	Debug Version:	152.7K Code,	63.2K Data,	215.5K Total
11586	Current Release:			
11587	Non-Debug Version:	71.4K Code,	9.0K Data,	80.4K Total
11588	Debug Version:	152.9K Code,	63.3K Data,	216.2K Total

11591 2) Linux

11593 Cleaned up EC driver. Exported an external EC read/write  
11594 interface. By going through this, other drivers (most notably  
11595 sonypi) will be able to serialize access to the EC.

11598 3) iASL Compiler/Disassembler

11600 Implemented support to optionally generate include files for both  
11601 ASM and C (the -i switch). This simplifies BIOS development by  
11602 automatically creating include files that contain external  
11603 declarations for the symbols that are created within the

11605 (optionally generated) ASM and C AML source files.

11608 -----  
11609 15 November 2002. Summary of changes for version 20021115.

11611 1) ACPI CA Core Subsystem:

11613 Fixed a memory leak problem where an error during resolution of

11615 method arguments during a method invocation from another method  
11616 failed to cleanup properly by deleting all successfully resolved  
11617 argument objects.

11619 Fixed a problem where the target of the Index() operator was not  
11620 correctly constructed if the source object was a package. This  
11621 problem has not been detected because the use of a target operand  
11622 with Index() is very rare.

11624 Fixed a problem with the Index() operator where an attempt was  
11625 made to delete the operand objects twice.

11627 Fixed a problem where an attempt was made to delete an operand  
11628 twice during execution of the CondRefOf() operator if the target  
11629 did not exist.

11631 Implemented the first of perhaps several internal create object  
11632 functions that create and initialize a specific object type. This  
11633 consolidates duplicated code wherever the object is created, thus  
11634 shrinking the size of the subsystem.

11636 Implemented improved debug/error messages for errors that occur  
11637 during nested method invocations. All executing method pathnames

11638 are displayed (with the error) as the call stack is unwound - thus  
11639 simplifying debug.

11641 Fixed a problem introduced in the 10/02 release that caused  
11642 premature deletion of a buffer object if a buffer was used as an  
11643 ASL operand where an integer operand is required (Thus causing an  
11644 implicit object conversion from Buffer to Integer.) The change in  
11645 the 10/02 release was attempting to fix a memory leak (albeit  
11646 incorrectly.)

11648 Code and Data Size: Current core subsystem library sizes are shown  
11649 below. These are the code and data sizes for the acpica.lib  
11650 produced by the Microsoft Visual C++ 6.0 compiler, and these  
11651 values do not include any ACPI driver or OSPM code. The debug  
11652 version of the code includes the debug output trace mechanism and  
11653 has a much larger code and data size. Note that these values will  
11654 vary depending on the efficiency of the compiler and the compiler  
11655 options used during generation.

11657	Previous Release			
11658	Non-Debug Version:	71.9K Code,	9.1K Data,	81.0K Total
11659	Debug Version:	153.1K Code,	63.3K Data,	216.4K Total
11660	Current Release:			
11661	Non-Debug Version:	71.3K Code,	9.0K Data,	80.3K Total
11662	Debug Version:	152.7K Code,	63.2K Data,	215.5K Total

11665 2) Linux

11667 Changed the implementation of the ACPI semaphores to use down()  
11668 instead of down\_interruptable(). It is important that the  
11669 execution of ACPI control methods not be interrupted by signals.  
11670 Methods must run to completion, or the system may be left in an  
11671 unknown/unstable state.

11673 Fixed a compilation error when CONFIG\_SOFTWARE\_SUSPEND is not set.  
11674 (Shawn Starr)

11677 3) iASL Compiler/Disassembler

11680 Changed the default location of output files. All output files  
11681 are now placed in the current directory by default instead of in  
11682 the directory of the source file. This change may affect some  
11683 existing makefiles, but it brings the behavior of the compiler in  
11684 line with other similar tools. The location of the output files  
11685 can be overridden with the -p command line switch.

11688 -----  
11689 11 November 2002. Summary of changes for version 20021111.

11692 0) ACPI Specification 2.0B is released and is now available at:  
11693 <http://www.acpi.info/index.html>

11696 1) ACPI CA Core Subsystem:

11698 Implemented support for the ACPI 2.0 SMBus Operation Regions.  
11699 This includes the early detection and handoff of the request to  
11700 the SMBus region handler (avoiding all of the complex field  
11701 support code), and support for the bidirectional return packet  
11702 from an SMBus write operation. This paves the way for the  
11703 development of SMBus drivers in each host operating system.



11705 Fixed a problem where the semaphore WAIT\_FOREVER constant was  
 11706 defined as 32 bits, but must be 16 bits according to the ACPI  
 11707 specification. This had the side effect of causing ASL  
 11708 Mutex/Event timeouts even though the ASL code requested a wait  
 11709 forever. Changed all internal references to the ACPI timeout  
 11710 parameter to 16 bits to prevent future problems. Changed the name  
 11711 of WAIT\_FOREVER to ACPI\_WAIT\_FOREVER.

11713 Code and Data Size: Current core subsystem library sizes are shown  
 11714 below. These are the code and data sizes for the acpica.lib  
 11715 produced by the Microsoft Visual C++ 6.0 compiler, and these  
 11716 values do not include any ACPI driver or OSPM code. The debug  
 11717 version of the code includes the debug output trace mechanism and  
 11718 has a much larger code and data size. Note that these values will  
 11719 vary depending on the efficiency of the compiler and the compiler  
 11720 options used during generation.

11722	Previous Release			
11723	Non-Debug Version:	71.4K Code,	9.0K Data,	80.4K Total
11724	Debug Version:	152.3K Code,	63.0K Data,	215.3K Total
11725	Current Release:			
11726	Non-Debug Version:	71.9K Code,	9.1K Data,	81.0K Total
11727	Debug Version:	153.1K Code,	63.3K Data,	216.4K Total

11730 2) Linux

11732 Module loading/unloading fixes (John Cagle)

11735 3) iASL Compiler/Disassembler

11737 Added support for the SMBBlockProcessCall keyword (ACPI 2.0)

11739 Implemented support for the disassembly of all SMBus protocol  
 11740 keywords (SMBQuick, SMBWord, etc.)

11742 -----

11743 01 November 2002. Summary of changes for version 20021101.

11746 1) ACPI CA Core Subsystem:

11748 Fixed a problem where platforms that have a GPE1 block but no GPE0  
 11749 block were not handled correctly. This resulted in a "GPE  
 11750 overlap" error message. GPE0 is no longer required.

11752 Removed code added in the previous release that inserted nodes  
 11753 into the namespace in alphabetical order. This caused some side-  
 11754 effects on various machines. The root cause of the problem is  
 11755 still under investigation since in theory, the internal ordering  
 11756 of the namespace nodes should not matter.

11759 Enhanced error reporting for the case where a named object is not  
 11760 found during control method execution. The full ACPI namepath  
 11761 (name reference) of the object that was not found is displayed in  
 11762 this case.

11764 Note: as a result of the overhaul of the namespace object types in  
 11765 the previous release, the namespace nodes for the predefined  
 11766 scopes (\_TZ, \_PR, etc.) are now of the type ACPI\_TYPE\_LOCAL\_SCOPE  
 11767 instead of ACPI\_TYPE\_ANY. This simplifies the namespace  
 11768 management code but may affect code that walks the namespace tree  
 11769 looking for specific object types.

11771 Code and Data Size: Current core subsystem library sizes are shown  
 11772 below. These are the code and data sizes for the acpica.lib  
 11773 produced by the Microsoft Visual C++ 6.0 compiler, and these  
 11774 values do not include any ACPI driver or OSPM code. The debug  
 11775 version of the code includes the debug output trace mechanism and  
 11776 has a much larger code and data size. Note that these values will  
 11777 vary depending on the efficiency of the compiler and the compiler  
 11778 options used during generation.

11780	Previous Release			
11781	Non-Debug Version:	70.7K Code,	8.6K Data,	79.3K Total
11782	Debug Version:	151.7K Code,	62.4K Data,	214.1K Total
11783	Current Release:			
11784	Non-Debug Version:	71.4K Code,	9.0K Data,	80.4K Total
11785	Debug Version:	152.3K Code,	63.0K Data,	215.3K Total

11788 2) Linux

11790 Fixed a problem introduced in the previous release where the  
 11791 Processor and Thermal objects were not recognized and installed in  
 11792 /proc. This was related to the scope type change described above.

11795 3) iASL Compiler/Disassembler

11797 Implemented the -g option to get all of the required ACPI tables  
 11798 from the registry and save them to files (Windows version of the  
 11799 compiler only.) The required tables are the FADT, FACS, and DSDT.

11801 Added ACPI table checksum validation during table disassembly in  
 11802 order to catch corrupted tables.

11805 -----

11806 22 October 2002. Summary of changes for version 20021022.

11808 1) ACPI CA Core Subsystem:

11810 Implemented a restriction on the Scope operator that the target  
 11811 must already exist in the namespace at the time the operator is  
 11812 encountered (during table load or method execution). In other  
 11813 words, forward references are not allowed and Scope() cannot  
 11814 create a new object. This changes the previous behavior where the  
 11815 interpreter would create the name if not found. This new behavior  
 11816 correctly enables the search-to-root algorithm during namespace  
 11817 lookup of the target name. Because of this upsearch, this fixes  
 11818 the known Compaq \_SB\_.OKEC problem and makes both the AML  
 11819 interpreter and iASL compiler compatible with other ACPI  
 11820 implementations.

11822 Completed a major overhaul of the internal ACPI object types for  
 11823 the ACPI Namespace and the associated operand objects. Many of  
 11824 these types had become obsolete with the introduction of the two-  
 11825 pass namespace load. This cleanup simplifies the code and makes  
 11826 the entire namespace load mechanism much clearer and easier to  
 11827 understand.

11829 Improved debug output for tracking scope opening/closing to help  
 11830 diagnose scoping issues. The old scope name as well as the new  
 11831 scope name are displayed. Also improved error messages for  
 11832 problems with ASL Mutex objects and error messages for GPE  
 11833 problems.

11835 Cleaned up the namespace dump code, removed obsolete code.

11837 All string output (for all namespace/object dumps) now uses the  
 11838 common ACPI string output procedure which handles escapes properly  
 11839 and does not emit non-printable characters.

11841 Fixed some issues with constants in the 64-bit version of the  
 11842 local C library (utclib.c)

11845 2) Linux

11847 EC Driver: No longer attempts to acquire the Global Lock at  
 11848 interrupt level.

11851 3) iASL Compiler/Disassembler

11853 Implemented ACPI 2.0B grammar change that disallows all Type 1 and  
 11854 2 opcodes outside of a control method. This means that the  
 11855 "executable" operators (versus the "namespace" operators) cannot  
 11856 be used at the table level; they can only be used within a control  
 11857 method.

11859 Implemented the restriction on the Scope() operator where the  
 11860 target must already exist in the namespace at the time the  
 11861 operator is encountered (during ASL compilation). In other words,  
 11862 forward references are not allowed and Scope() cannot create a new  
 11863 object. This makes the iASL compiler compatible with other ACPI  
 11864 implementations and makes the Scope() implementation adhere to the  
 11865 ACPI specification.

11867 Fixed a problem where namepath optimization for the Alias operator  
 11868 was optimizing the wrong path (of the two namepaths.) This caused  
 11869 a "Missing alias link" error message.

11871 Fixed a problem where an "unknown reserved name" warning could be  
 11872 incorrectly generated for names like "\_SB" when the trailing  
 11873 underscore is not used in the original ASL.

11875 Fixed a problem where the reserved name check did not handle  
 11876 NamePaths with multiple NameSegs correctly. The first nameseg of  
 11877 the NamePath was examined instead of the last NameSeg.

11880 -----

11882 02 October 2002. Summary of changes for this release.

11885 1) ACPI CA Core Subsystem version 20021002:

11887 Fixed a problem where a store/copy of a string to an existing  
 11888 string did not always set the string length properly in the String  
 11889 object.

11891 Fixed a reported problem with the ToString operator where the  
 11892 behavior was identical to the ToHexString operator instead of just  
 11893 simply converting a raw buffer to a string data type.

11895 Fixed a problem where CopyObject and the other "explicit"  
 11896 conversion operators were not updating the internal namespace node  
 11897 type as part of the store operation.

11899 Fixed a memory leak during implicit source operand conversion  
 11900 where the original object was not deleted if it was converted to a  
 11901 new object of a different type.

11903 Enhanced error messages for all problems associated with namespace  
 11904 lookups. Common procedure generates and prints the lookup name as  
 11905 well as the formatted status.

11907 Completed implementation of a new design for the Alias support  
 11908 within the namespace. The existing design did not handle the case  
 11909 where a new object was assigned to one of the two names due to the  
 11910 use of an explicit conversion operator, resulting in the two names  
 11911 pointing to two different objects. The new design simply points  
 11912 the Alias name to the original name node - not to the object.  
 11913 This results in a level of indirection that must be handled in the  
 11914 name resolution mechanism.

11916 Code and Data Size: Current core subsystem library sizes are shown  
 11917 below. These are the code and data sizes for the acpica.lib  
 11918 produced by the Microsoft Visual C++ 6.0 compiler, and these  
 11919 values do not include any ACPI driver or OSPM code. The debug  
 11920 version of the code includes the debug output trace mechanism and  
 11921 has a larger code and data size. Note that these values will vary  
 11922 depending on the efficiency of the compiler and the compiler  
 11923 options used during generation.

11925	Previous Release			
11926	Non-Debug Version:	69.6K Code,	8.3K Data,	77.9K Total
11927	Debug Version:	150.0K Code,	61.7K Data,	211.7K Total
11928	Current Release:			
11929	Non-Debug Version:	70.7K Code,	8.6K Data,	79.3K Total
11930	Debug Version:	151.7K Code,	62.4K Data,	214.1K Total

11933 2) Linux

11935 Initialize thermal driver's timer before it is used. (Knut  
 11936 Neumann)

11938 Allow handling negative celsius values. (Kochi Takayoshi)

11940 Fix thermal management and make trip points. R/W (Pavel Machek)

11942 Fix /proc/acpi/sleep. (P. Christeas)

11944 IA64 fixes. (David Mosberger)

11946 Fix reversed logic in blacklist code. (Sergio Monteiro Basto)

11948 Replace ACPI\_DEBUG define with ACPI\_DEBUG\_OUTPUT. (Dominik  
 11949 Brodowski)

11952 3) iASL Compiler/Disassembler

11954 Clarified some warning/error messages.

11957 -----

11958 18 September 2002. Summary of changes for this release.

11961 1) ACPI CA Core Subsystem version 20020918:

11963 Fixed a reported problem with reference chaining (via the Index()  
 11964 and RefOf() operators) in the ObjectType() and SizeOf() operators.  
 11965 The definition of these operators includes the dereferencing of  
 11966 all chained references to return information on the base object.

11968 Fixed a problem with stores to indexed package elements - the  
 11969 existing code would not complete the store if an "implicit  
 11970 conversion" was not performed. In other words, if the existing  
 11971 object (package element) was to be replaced completely, the code  
 11972 didn't handle this case.

11974 Relaxed typechecking on the ASL "Scope" operator to allow the  
 11975 target name to refer to an object of type Integer, String, or  
 11976 Buffer, in addition to the scoping object types (Device,  
 11977 predefined Scopes, Processor, PowerResource, and ThermalZone.)  
 11978 This allows existing AML code that has workarounds for a bug in  
 11979 Windows to function properly. A warning is issued, however. This  
 11980 affects both the AML interpreter and the iASL compiler. Below is  
 11981 an example of this type of ASL code:

```
11983      Name(DEV,0x00)
11984      Scope(DEV)
11985      {
```

11987 Fixed some reported problems with 64-bit integer support in the  
 11988 local implementation of C library functions (clib.c)

11991 2) Linux

11993 Use ACPI fix map region instead of IOAPIC region, since it is  
 11994 undefined in non-SMP.

11996 Ensure that the SCI has the proper polarity and trigger, even on  
 11997 systems that do not have an interrupt override entry in the MADT.

11999 2.5 big driver reorganization (Pat Mochel)

12001 Use early table mapping code from acpitab.c (Andi Kleen)

12003 New blacklist entries (Andi Kleen)

12005 Blacklist improvements. Split blacklist code out into a separate  
 12006 file. Move checking the blacklist to very early. Previously, we  
 12007 would use ACPI tables, and then halfway through init, check the  
 12008 blacklist -- too late. Now, it's early enough to completely fall-  
 12009 back to non-ACPI.

12012 3) iASL Compiler/Disassembler version 20020918:

12014 Fixed a problem where the typechecking code didn't know that an  
 12015 alias could point to a method. In other words, aliases were not  
 12016 being dereferenced during typechecking.

12019 -----

12020 29 August 2002. Summary of changes for this release.

12022 1) ACPI CA Core Subsystem Version 20020829:

12024 If the target of a Scope() operator already exists, it must be an  
 12025 object type that actually opens a scope -- such as a Device,  
 12026 Method, Scope, etc. This is a fatal runtime error. Similar error  
 12027 check has been added to the iASL compiler also.

12029 Tightened up the namespace load to disallow multiple names in the  
 12030 same scope. This previously was allowed if both objects were of  
 12031 the same type. (i.e., a lookup was the same as entering a new  
 12032 name).

12035 2) Linux

12037 Ensure that the ACPI interrupt has the proper trigger and  
 12038 polarity.

12040 local\_irq\_disable is extraneous. (Matthew Wilcox)

12042 Make "acpi=off" actually do what it says, and not use the ACPI  
 12043 interpreter \*or\* the tables.

12045 Added arch-neutral support for parsing SLIT and SRAT tables (Kochi  
 12046 Takayoshi)

12049 3) iASL Compiler/Disassembler Version 20020829:

12051 Implemented namepath optimization for name declarations. For  
 12052 example, a declaration like "Method (\\_SB\_.ABCD)" would get  
 12053 optimized to "Method (ABCD)" if the declaration is within the  
 12054 \\_SB\_ scope. This optimization is in addition to the named  
 12055 reference path optimization first released in the previous  
 12056 version. This would seem to complete all possible optimizations  
 12057 for namepaths within the ASL/AML.

12059 If the target of a Scope() operator already exists, it must be an  
 12060 object type that actually opens a scope -- such as a Device,  
 12061 Method, Scope, etc.

12063 Implemented a check and warning for unreachable code in the same  
 12064 block below a Return() statement.

12066 Fixed a problem where the listing file was not generated if the  
 12067 compiler aborted if the maximum error count was exceeded (200).

12069 Fixed a problem where the typechecking of method return values was  
 12070 broken. This includes the check for a return value when the  
 12071 method is invoked as a TermArg (a return value is expected.)

12073 Fixed a reported problem where EOF conditions during a quoted  
 12074 string or comment caused a fault.

12077 -----

12078 15 August 2002. Summary of changes for this release.

12080 1) ACPI CA Core Subsystem Version 20020815:

12082 Fixed a reported problem where a Store to a method argument that  
 12083 contains a reference did not perform the indirect store correctly.  
 12084 This problem was created during the conversion to the new  
 12085 reference object model - the indirect store to a method argument  
 12086 code was not updated to reflect the new model.

12088 Reworked the ACPI mode change code to better conform to ACPI 2.0,  
 12089 handle corner cases, and improve code legibility (Kochi Takayoshi)

12091 Fixed a problem with the pathname parsing for the carat (^)  
 12092 prefix. The heavy use of the carat operator by the new namepath  
 12093 optimization in the iASL compiler uncovered a problem with the AML  
 12094 interpreter handling of this prefix. In the case where one or  
 12095 more carats precede a single nameseg, the nameseg was treated as  
 12096 standalone and the search rule (to root) was inadvertently  
 12097 applied. This could cause both the iASL compiler and the  
 12098 interpreter to find the wrong object or to miss the error that  
 12099 should occur if the object does not exist at that exact pathname.

12101 Found and fixed the problem where the HP Pavilion DSDT would not  
 12102 load. This was a relatively minor tweak to the table loading code  
 12103 (a problem caused by the unexpected encounter with a method  
 12104 invocation not within a control method), but it does not solve the  
 12105 overall issue of the execution of AML code at the table level.  
 12106 This investigation is still ongoing.

12108 Code and Data Size: Current core subsystem library sizes are shown  
 12109 below. These are the code and data sizes for the acpica.lib  
 12110 produced by the Microsoft Visual C++ 6.0 compiler, and these  
 12111 values do not include any ACPI driver or OSPM code. The debug  
 12112 version of the code includes the debug output trace mechanism and  
 12113 has a larger code and data size. Note that these values will vary  
 12114 depending on the efficiency of the compiler and the compiler  
 12115 options used during generation.

12117	Previous Release			
12118	Non-Debug Version:	69.1K Code,	8.2K Data,	77.3K Total
12119	Debug Version:	149.4K Code,	61.6K Data,	211.0K Total
12120	Current Release:			
12121	Non-Debug Version:	69.6K Code,	8.3K Data,	77.9K Total
12122	Debug Version:	150.0K Code,	61.7K Data,	211.7K Total

12125 2) Linux

12127 Remove redundant slab.h include (Brad Hards)

12129 Fix several bugs in thermal.c (Herbert Nachtnebel)

12131 Make CONFIG\_ACPI\_BOOT work properly (Pavel Machek)

12133 Change acpi\_system\_suspend to use updated irq functions (Pavel  
 12134 Machek)

12136 Export acpi\_get\_firmware\_table (Matthew Wilcox)

12138 Use proper root proc entry for ACPI (Kochi Takayoshi)

12140 Fix early-boot table parsing (Bjorn Helgaas)

12143 3) iASL Compiler/Disassembler

12145 Reworked the compiler options to make them more consistent and to  
 12146 use two-letter options where appropriate. We were running out of  
 12147 sensible letters. This may break some makefiles, so check the  
 12148 current options list by invoking the compiler with no parameters.

12150 Completed the design and implementation of the ASL namespace  
 12151 optimization option for the compiler. This option optimizes all  
 12152 references to named objects to the shortest possible path. The  
 12153 first attempt tries to utilize a single nameseg (4 characters) and  
 12154 the "search-to-root" algorithm used by the interpreter. If that  
 12155 cannot be used (because either the name is not in the search path  
 12156 or there is a conflict with another object with the same name),  
 12157 the pathname is optimized using the carat prefix (usually a  
 12158 shorter string than specifying the entire path from the root.)

12160 Implemented support to obtain the DSDT from the Windows registry  
 12161 (when the disassembly option is specified with no input file).  
 12162 Added this code as the implementation for AcpiOsTableOverride in  
 12163 the Windows OSL. Migrated the 16-bit code (used in the AcpiDump  
 12164 utility) to scan memory for the DSDT to the AcpiOsTableOverride  
 12165 function in the DOS OSL to make the disassembler truly OS

12166 independent.

12168 Implemented a new option to disassemble and compile in one step.  
 12169 When used without an input filename, this option will grab the  
 12170 DSDT from the local machine, disassemble it, and compile it in one  
 12171 step.

12173 Added a warning message for invalid escapes (a backslash followed  
 12174 by any character other than the allowable escapes). This catches  
 12175 the quoted string error "\\_SB\_" (which should be "\\\_SB\_").

12177 Also, there are numerous instances in the ACPI specification where  
 12178 this error occurs.

12180 Added a compiler option to disable all optimizations. This is  
 12181 basically the "compatibility mode" because by using this option,  
 12182 the AML code will come out exactly the same as other ASL  
 12183 compilers.

12185 Added error messages for incorrectly ordered dependent resource  
 12186 functions. This includes: missing EndDependentFn macro at end of  
 12187 dependent resource list, nested dependent function macros (both  
 12188 start and end), and missing StartDependentFn macro. These are  
 12189 common errors that should be caught at compile time.

12191 Implemented \_OSI support for the disassembler and compiler. \_OSI  
 12192 must be included in the namespace for proper disassembly (because  
 12193 the disassembler must know the number of arguments.)

12195 Added an "optimization" message type that is optional (off by  
 12196 default). This message is used for all optimizations - including  
 12197 constant folding, integer optimization, and namepath optimization.

12199 -----  
 12200 25 July 2002. Summary of changes for this release.

12203 1) ACPI CA Core Subsystem Version 20020725:

12205 The AML Disassembler has been enhanced to produce compilable ASL  
 12206 code and has been integrated into the iASL compiler (see below) as  
 12207 well as the single-step disassembly for the AML debugger and the  
 12208 disassembler for the AcpiDump utility. All ACPI 2.0A opcodes,  
 12209 resource templates and macros are fully supported. The  
 12210 disassembler has been tested on over 30 different AML files,  
 12211 producing identical AML when the resulting disassembled ASL file  
 12212 is recompiled with the same ASL compiler.

12214 Modified the Resource Manager to allow zero interrupts and zero  
 12215 dma channels during the GetCurrentResources call. This was  
 12216 causing problems on some platforms.

12218 Added the AcpiOsRedirectOutput interface to the OSL to simplify  
 12219 output redirection for the AcpiOsPrintf and AcpiOsVprintf  
 12220 interfaces.

12222 Code and Data Size: Current core subsystem library sizes are shown  
 12223 below. These are the code and data sizes for the acpica.lib  
 12224 produced by the Microsoft Visual C++ 6.0 compiler, and these  
 12225 values do not include any ACPI driver or OSPM code. The debug  
 12226 version of the code includes the debug output trace mechanism and  
 12227 has a larger code and data size. Note that these values will vary  
 12228 depending on the efficiency of the compiler and the compiler  
 12229 options used during generation.

12231 Previous Release

12232 Non-Debug Version: 68.7K Code, 7.4K Data, 76.1K Total  
 12233 Debug Version: 142.9K Code, 58.7K Data, 201.6K Total  
 12234 Current Release:  
 12235 Non-Debug Version: 69.1K Code, 8.2K Data, 77.3K Total  
 12236 Debug Version: 149.4K Code, 61.6K Data, 211.0K Total

12239 2) Linux

12241 Fixed a panic in the EC driver (Dominik Brodowski)

12243 Implemented checksum of the R/XSDT itself during Linux table scan  
 12244 (Richard Schaal)

12247 3) iASL compiler

12249 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.

12256 Fixed a problem with the VendorShort and VendorLong resource descriptors where an invalid AML sequence was created.

12259 Implemented a fix for BufferData term in the ASL parser. It was inadvertently defined twice, allowing invalid syntax to pass and causing reduction conflicts.

12263 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.

12270 -----

12271 02 July 2002. Summary of changes for this release.

12274 1) ACPI CA Core Subsystem Version 20020702:

12276 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.

12286 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.

12291 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.

12296 Added additional error messages in the Resource Manager to explain AE\_BAD\_DATA errors when they occur during resource parsing.

12299 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.

12306 2) Linux

12308 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:

```
12312 ACPI: PCI Interrupt Routing Table [_SB_.PCI0.PPB._PRT]
12313 acpi_utils-0430 [145] acpi_evaluate_reference:
12314 Invalid element in package (not a device reference)
```

12316 The method evaluation utility uses the new buffer allocation scheme instead of calling AcpiEvaluate Object twice.

12319 Added support for ECDDT. This allows the use of the Embedded

12321 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 ECDDT are still rare, so only limited testing was performed of the added functionality.)

12326 Fixed memory leaks in the EC driver.

12328 Eliminated a brittle code structure in acpi\_bus\_init().

12330 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.

12334 Implemented fix for keyboard hang when getting battery readings on some systems (Stephen White)

12337 PCI IRQ routing update (Dominik Brodowski)

12339 Fix an ifdef to allow compilation on UP with LAPIC but no IOAPIC support

12342 -----

12343 11 June 2002. Summary of changes for this release.

12346 1) ACPI CA Core Subsystem Version 20020611:

12348 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).

12356 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.

12360 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.

12365 Implemented immediate resolution of the AML Constant opcodes  
 12366 (Zero, One, Ones, Revision) to Integer objects upon detection  
 12367 within the AML stream. This has simplified and reduced the  
 12368 generated code size of the subsystem by eliminating about 10  
 12369 switch statements for these constants (which previously were  
 12370 contained in Reference objects.) The complicating issues are that  
 12371 the Zero opcode is used as a "placeholder" for unspecified  
 12372 optional target operands and stores to constants are defined to be  
 12373 no-ops.

12375 Code and Data Size: Current core subsystem library sizes are shown  
 12376 below. These are the code and data sizes for the acpica.lib  
 12377 produced by the Microsoft Visual C++ 6.0 compiler, and these  
 12378 values do not include any ACPI driver or OSPM code. The debug  
 12379 version of the code includes the debug output trace mechanism and  
 12380 has a larger code and data size. Note that these values will vary  
 12381 depending on the efficiency of the compiler and the compiler  
 12382 options used during generation.

12384	Previous Release			
12385	Non-Debug Version:	69.3K Code,	7.4K Data,	76.7K Total
12386	Debug Version:	143.8K Code,	58.8K Data,	202.6K Total
12387	Current Release:			
12388	Non-Debug Version:	68.7K Code,	7.4K Data,	76.1K Total
12389	Debug Version:	142.9K Code,	58.7K Data,	201.6K Total

12392 2) Linux

12395 Added preliminary support for obtaining \_TRA data for PCI root  
 12396 bridges (Bjorn Helgaas).

12399 3) iASL Compiler Version X2046:

12401 Fixed a problem where the "\_DDN" reserved name was defined to be a  
 12402 control method with one argument. There are no arguments, and  
 12403 \_DDN does not have to be a control method.

12405 Fixed a problem with the Linux version of the compiler where the  
 12406 source lines printed with error messages were the wrong lines.  
 12407 This turned out to be the "LF versus CR/LF" difference between  
 12408 Windows and Unix. This appears to be the longstanding issue  
 12409 concerning listing output and error messages.

12411 Fixed a problem with the Linux version of compiler where opcode  
 12412 names within error messages were wrong. This was caused by a  
 12413 slight difference in the output of the Flex tool on Linux versus  
 12414 Windows.

12416 Fixed a problem with the Linux compiler where the hex output files  
 12417 contained some garbage data caused by an internal buffer overrun.

12420 -----  
 12421 17 May 2002. Summary of changes for this release.

12424 1) ACPI CA Core Subsystem Version 20020517:

12426 Implemented a workaround to an BIOS bug discovered on the HP  
 12427 OmniBook where the FADT revision number and the table size are  
 12428 inconsistent (ACPI 2.0 revision vs. ACPI 1.0 table size). The new  
 12429 behavior is to fallback to using only the ACPI 1.0 fields of the

12430 FADT if the table is too small to be a ACPI 2.0 table as claimed  
 12431 by the revision number. Although this is a BIOS bug, this is a  
 12432 case where the workaround is simple enough and with no side  
 12433 effects, so it seemed prudent to add it. A warning message is  
 12434 issued, however.

12436 Implemented minimum size checks for the fixed-length ACPI tables -  
 12437 - the FADT and FACS, as well as consistency checks between the  
 12438 revision number and the table size.

12440 Fixed a reported problem in the table override support where the  
 12441 new table pointer was incorrectly treated as a physical address  
 12442 instead of a logical address.

12444 Eliminated the use of the AE\_AML\_ERROR exception and replaced it  
 12445 with more descriptive codes.

12447 Fixed a problem where an exception would occur if an ASL Field was  
 12448 defined with no named Field Units underneath it (used by some  
 12449 index fields).

12451 Code and Data Size: Current core subsystem library sizes are shown  
 12452 below. These are the code and data sizes for the acpica.lib  
 12453 produced by the Microsoft Visual C++ 6.0 compiler, and these  
 12454 values do not include any ACPI driver or OSPM code. The debug  
 12455 version of the code includes the debug output trace mechanism and  
 12456 has a larger code and data size. Note that these values will vary  
 12457 depending on the efficiency of the compiler and the compiler  
 12458 options used during generation.

12460	Previous Release			
12461	Non-Debug Version:	68.8K Code,	7.1K Data,	75.9K Total
12462	Debug Version:	142.9K Code,	58.4K Data,	201.3K Total
12463	Current Release:			
12464	Non-Debug Version:	69.3K Code,	7.4K Data,	76.7K Total
12465	Debug Version:	143.8K Code,	58.8K Data,	202.6K Total

12469 2) Linux

12471 Much work done on ACPI init (MADT and PCI IRQ routing support).  
 12472 (Paul D. and Dominik Brodowski)

12474 Fix PCI IRQ-related panic on boot (Sam Revitch)

12476 Set BM\_ARB\_DIS when entering a sleep state (Ducrot Bruno)

12478 Fix "MHz" typo (Dominik Brodowski)

12480 Fix RTC year 2000 issue (Dominik Brodowski)

12482 Preclude multiple button proc entries (Eric Brunet)

12484 Moved arch-specific code out of include/platform/aclinux.h

12486 3) iASL Compiler Version X2044:

12488 Implemented error checking for the string used in the EISAID macro  
 12489 (Usually used in the definition of the \_HID object.) The code now  
 12490 strictly enforces the PnP format - exactly 7 characters, 3  
 12491 uppercase letters and 4 hex digits.

12493 If a raw string is used in the definition of the \_HID object  
 12494 (instead of the EISAID macro), the string must contain all  
 12495 alphanumeric characters (e.g., "\*PNP0011" is not allowed because

12496 of the asterisk.)

12498 Implemented checking for invalid use of ACPI reserved names for  
12499 most of the name creation operators (Name, Device, Event, Mutex,  
12500 OperationRegion, PowerResource, Processor, and ThermalZone.)  
12501 Previously, this check was only performed for control methods.

12503 Implemented an additional check on the Name operator to emit an  
12504 error if a reserved name that must be implemented in ASL as a  
12505 control method is used. We know that a reserved name must be a  
12506 method if it is defined with input arguments.

12508 The warning emitted when a namespace object reference is not found  
12509 during the cross reference phase has been changed into an error.  
12510 The "External" directive should be used for names defined in other  
12511 modules.

#### 12514 4) Tools and Utilities

12516 The 16-bit tools (adump16 and aexec16) have been regenerated and  
12517 tested.

12519 Fixed a problem with the output of both acpidump and adump16 where  
12520 the indentation of closing parentheses and brackets was not

12522 aligned properly with the parent block.

12525 -----

12526 03 May 2002. Summary of changes for this release.

#### 12529 1) ACPI CA Core Subsystem Version 20020503:

12531 Added support a new OSL interface that allows the host operating

12533 system software to override the DSDT found in the firmware -  
12534 AcpiOsTableOverride. With this interface, the OSL can examine the  
12535 version of the firmware DSDT and replace it with a different one  
12536 if desired.

12538 Added new external interfaces for accessing ACPI registers from  
12539 device drivers and other system software - AcpiGetRegister and  
12540 AcpiSetRegister. This was simply an externalization of the  
12541 existing AcpiHwBitRegister interfaces.

12543 Fixed a regression introduced in the previous build where the  
12544 ASL/AML CreateField operator always returned an error,  
12545 "destination must be a NS Node".

12547 Extended the maximum time (before failure) to successfully enable  
12548 ACPI mode to 3 seconds.

12550 Code and Data Size: Current core subsystem library sizes are shown  
12551 below. These are the code and data sizes for the acpica.lib  
12552 produced by the Microsoft Visual C++ 6.0 compiler, and these  
12553 values do not include any ACPI driver or OSPM code. The debug  
12554 version of the code includes the debug output trace mechanism and  
12555 has a larger code and data size. Note that these values will vary  
12556 depending on the efficiency of the compiler and the compiler  
12557 options used during generation.

12559 Previous Release  
12560 Non-Debug Version: 68.5K Code, 7.0K Data, 75.5K Total  
12561 Debug Version: 142.4K Code, 58.3K Data, 200.7K Total

12562 Current Release:  
12563 Non-Debug Version: 68.8K Code, 7.1K Data, 75.9K Total  
12564 Debug Version: 142.9K Code, 58.4K Data, 201.3K Total

#### 12567 2) Linux

12569 Enhanced ACPI init code for SMP. We are now fully MPS and \$PIR-  
12570 free. While 3 out of 4 of our in-house systems work fine, the last  
12571 one still hangs when testing the LAPIC timer.

12573 Renamed many files in 2.5 kernel release to omit "acpi\_" from the  
12574 name.

12576 Added warning on boot for Presario 711FR.

12578 Sleep improvements (Pavel Machek)

12580 ACPI can now be built without CONFIG\_PCI enabled.

12582 IA64: Fixed memory map functions (JI Lee)

#### 12585 3) iASL Compiler Version X2043:

12587 Added support to allow the compiler to be integrated into the MS  
12588 VC++ development environment for one-button compilation of single  
12589 files or entire projects -- with error-to-source-line mapping.

12591 Implemented support for compile-time constant folding for the  
12592 Type3, Type4, and Type5 opcodes first defined in the ACPI 2.0  
12593 specification. This allows the ASL writer to use expressions  
12594 instead of Integer/Buffer/String constants in terms that must  
12595 evaluate to constants at compile time and will also simplify the  
12596 emitted AML in any such sub-expressions that can be folded  
12597 (evaluated at compile-time.) This increases the size of the  
12598 compiler significantly because a portion of the ACPI CA AML  
12599 interpreter is included within the compiler in order to pre-  
12600 evaluate constant expressions.

12603 Fixed a problem with the "Unicode" ASL macro that caused the  
12604 compiler to fault. (This macro is used in conjunction with the  
12605 \_STR reserved name.)

12607 Implemented an AML opcode optimization to use the Zero, One, and  
12608 Ones opcodes where possible to further reduce the size of integer  
12609 constants and thus reduce the overall size of the generated AML  
12610 code.

12612 Implemented error checking for new reserved terms for ACPI version  
12613 2.0A.

12615 Implemented the -qr option to display the current list of ACPI  
12616 reserved names known to the compiler.

12618 Implemented the -qc option to display the current list of ASL  
12619 operators that are allowed within constant expressions and can  
12620 therefore be folded at compile time if the operands are constants.

#### 12623 4) Documentation

12625 Updated the Programmer's Reference for new interfaces, data types,  
12626 and memory allocation model options.

12628 Updated the iASL Compiler User Reference to apply new format and  
12629 add information about new features and options.

12631 -----  
12632 19 April 2002. Summary of changes for this release.

12634 1) ACPI CA Core Subsystem Version 20020419:

12636 The source code base for the Core Subsystem has been completely  
12637 cleaned with PC-lint (FlexLint) for both 32-bit and 64-bit  
12638 versions. The Lint option files used are included in the  
12639 /acpi/generate/lint directory.

12641 Implemented enhanced status/error checking across the entire  
12642 Hardware manager subsystem. Any hardware errors (reported from  
12643 the OSL) are now bubbled up and will abort a running control  
12644 method.

12647 Fixed a problem where the per-ACPI-table integer width (32 or 64)  
12648 was stored only with control method nodes, causing a fault when  
12649 non-control method code was executed during table loading. The  
12650 solution implemented uses a global variable to indicate table  
12651 width across the entire ACPI subsystem. Therefore, ACPI CA does  
12652 not support mixed integer widths across different ACPI tables  
12653 (DSDT, SSDT).

12655 Fixed a problem where NULL extended fields (X fields) in an ACPI  
12656 2.0 ACPI FADT caused the table load to fail. Although the  
12657 existing ACPI specification is a bit fuzzy on this topic, the new  
12658 behavior is to fall back on a ACPI 1.0 field if the corresponding  
12659 ACPI 2.0 X field is zero (even though the table revision indicates  
12660 a full ACPI 2.0 table.) The ACPI specification will be updated to  
12661 clarify this issue.

12663 Fixed a problem with the SystemMemory operation region handler  
12664 where memory was always accessed byte-wise even if the AML-  
12665 specified access width was larger than a byte. This caused  
12666 problems on systems with memory-mapped I/O. Memory is now  
12667 accessed with the width specified. On systems that do not support  
12668 non-aligned transfers, a check is made to guarantee proper address  
12669 alignment before proceeding in order to avoid an AML-caused  
12670 alignment fault within the kernel.

12673 Fixed a problem with the ExtendedIrq resource where only one byte  
12674 of the 4-byte Irq field was extracted.

12676 Fixed the AcpiExDigitsNeeded() procedure to support \_UID. This  
12677 function was out of date and required a rewrite.

12679 Code and Data Size: Current core subsystem library sizes are shown  
12680 below. These are the code and data sizes for the acpica.lib  
12681 produced by the Microsoft Visual C++ 6.0 compiler, and these  
12682 values do not include any ACPI driver or OSPM code. The debug  
12683 version of the code includes the debug output trace mechanism and  
12684 has a larger code and data size. Note that these values will vary  
12685 depending on the efficiency of the compiler and the compiler  
12686 options used during generation.

12688	Previous Release			
12689	Non-Debug Version:	66.6K Code,	6.5K Data,	73.1K Total
12690	Debug Version:	139.8K Code,	57.4K Data,	197.2K Total
12691	Current Release:			
12692	Non-Debug Version:	68.5K Code,	7.0K Data,	75.5K Total
12693	Debug Version:	142.4K Code,	58.3K Data,	200.7K Total

12696 2) Linux

12698 PCI IRQ routing fixes (Dominik Brodowski)

12701 3) iASL Compiler Version X2042:

12703 Implemented an additional compile-time error check for a field  
12704 unit whose size + minimum access width would cause a run-time  
12705 access beyond the end-of-region. Previously, only the field size  
12706 itself was checked.

12708 The Core subsystem and iASL compiler now share a common parse  
12709 object in preparation for compile-time evaluation of the type  
12710 3/4/5 ASL operators.

12713 -----  
12714 Summary of changes for this release: 03\_29\_02

12716 1) ACPI CA Core Subsystem Version 20020329:

12718 Implemented support for late evaluation of TermArg operands to  
12719 Buffer and Package objects. This allows complex expressions to be  
12720 used in the declarations of these object types.

12722 Fixed an ACPI 1.0 compatibility issue when reading Fields. In ACPI  
12723 1.0, if the field was larger than 32 bits, it was returned as a  
12724 buffer - otherwise it was returned as an integer. In ACPI 2.0,  
12725 the field is returned as a buffer only if the field is larger than  
12726 64 bits. The TableRevision is now considered when making this  
12727 conversion to avoid incompatibility with existing ASL code.

12729 Implemented logical addressing for AcpiOsGetRootPointer. This  
12730 allows an RSDP with either a logical or physical address. With  
12731 this support, the host OS can now override all ACPI tables with  
12732 one logical RSDP. Includes implementation of "typed" pointer  
12733 support to allow a common data type for both physical and logical  
12734 pointers internally. This required a change to the  
12735 AcpiOsGetRootPointer interface.

12737 Implemented the use of ACPI 2.0 Generic Address Structures for all  
12738 GPE, Fixed Event, and PM Timer I/O. This allows the use of memory  
12739 mapped I/O for these ACPI features.

12741 Initialization now ignores not only non-required tables (All  
12742 tables other than the FADT, FACS, DSDT, and SSDTs), but also does  
12743 not validate the table headers of unrecognized tables.

12745 Fixed a problem where a notify handler could only be  
12746 installed/removed on an object of type Device. All "notify"

12748 objects are now supported -- Devices, Processor, Power, and  
12749 Thermal.

12751 Removed most verbosity from the ACPI\_DB\_INFO debug level. Only  
12752 critical information is returned when this debug level is enabled.

12754 Code and Data Size: Current core subsystem library sizes are shown  
12755 below. These are the code and data sizes for the acpica.lib  
12756 produced by the Microsoft Visual C++ 6.0 compiler, and these  
12757 values do not include any ACPI driver or OSPM code. The debug  
12758 version of the code includes the debug output trace mechanism and  
12759 has a larger code and data size. Note that these values will vary



12760 depending on the efficiency of the compiler and the compiler  
12761 options used during generation.

12763 Previous Release  
12764 Non-Debug Version: 65.4K Code, 6.2K Data, 71.6K Total  
12765 Debug Version: 138.0K Code, 56.6K Data, 194.6K Total  
12766 Current Release:  
12767 Non-Debug Version: 66.6K Code, 6.5K Data, 73.1K Total  
12768 Debug Version: 139.8K Code, 57.4K Data, 197.2K Total

12771 2) Linux:

12773 The processor driver (acpi\_processor.c) now fully supports ACPI  
12774 2.0-based processor performance control (e.g. Intel(R)  
12775 SpeedStep(TM) technology) Note that older laptops that only have  
12776 the Intel "applet" interface are not supported through this. The  
12777 'limit' and 'performance' interface (/proc) are fully functional.  
12778 [Note that basic policy for controlling performance state  
12779 transitions will be included in the next version of ospmd.] The  
12780 idle handler was modified to more aggressively use C2, and PIIX4  
12781 errata handling underwent a complete overhaul (big thanks to  
12782 Dominik Brodowski).

12784 Added support for ACPI-PCI device binding (acpi\_pci\_root.c). \_ADR-  
12785 based devices in the ACPI namespace are now dynamically bound  
12786 (associated) with their PCI counterparts (e.g. PCI1->01:00.0).  
12787 This allows, among other things, ACPI to resolve bus numbers for  
12788 subordinate PCI bridges.

12790 Enhanced PCI IRQ routing to get the proper bus number for \_PRT  
12791 entries defined underneath PCI bridges.

12793 Added IBM 600E to bad bios list due to invalid \_ADR value for  
12794 PIIX4 PCI-ISA bridge, resulting in improper PCI IRQ routing.

12796 In the process of adding full MADT support (e.g. IOAPIC) for IA32  
12797 (acpi.c, mpparse.c) -- stay tuned.

12799 Added back visual differentiation between fixed-feature and  
12800 control-method buttons in dmesg. Buttons are also subtyped (e.g.  
12801 button/power/PWRF) to simplify button identification.

12803 We no longer use -Wno-unused when compiling debug. Please ignore  
12804 any "\_THIS\_MODULE defined but not used" messages.

12806 Can now shut down the system using "magic sysrq" key.

12809 3) iASL Compiler version 2041:

12811 Fixed a problem where conversion errors for hex/octal/decimal  
12812 constants were not reported.

12814 Implemented a fix for the General Register template Address field.  
12815 This field was 8 bits when it should be 64.

12817 Fixed a problem where errors/warnings were no longer being emitted  
12818 within the listing output file.

12820 Implemented the ACPI 2.0A restriction on ACPI Table Signatures to  
12821 exactly 4 characters, alphanumeric only.

12826 -----  
12827 Summary of changes for this release: 03\_08\_02

12830 1) ACPI CA Core Subsystem Version 20020308:

12832 Fixed a problem with AML Fields where the use of the "AccessAny"  
12833 keyword could cause an interpreter error due to attempting to read  
12834 or write beyond the end of the parent Operation Region.

12836 Fixed a problem in the SystemMemory Operation Region handler where  
12837 an attempt was made to map memory beyond the end of the region.  
12838 This was the root cause of the "AE\_ERROR" and "AE\_NO\_MEMORY"  
12839 errors on some Linux systems.

12841 Fixed a problem where the interpreter/namespace "search to root"  
12842 algorithm was not functioning for some object types. Relaxed the  
12843 internal restriction on the search to allow upsearches for all  
12844 external object types as well as most internal types.

12847 2) Linux:

12849 We now use safe\_halt() macro versus individual calls to sti | hlt.

12851 Writing to the processor limit interface should now work. "echo 1"  
12852 will increase the limit, 2 will decrease, and 0 will reset to the

12854 default.

12857 3) ASL compiler:

12859 Fixed segfault on Linux version.

12862 -----  
12863 Summary of changes for this release: 02\_25\_02

12865 1) ACPI CA Core Subsystem:

12868 Fixed a problem where the GPE bit masks were not initialized  
12869 properly, causing erratic GPE behavior.

12871 Implemented limited support for multiple calling conventions. The  
12872 code can be generated with either the VPL (variable parameter  
12873 list, or "C") convention, or the FPL (fixed parameter list, or  
12874 "Pascal") convention. The core subsystem is about 3.4% smaller  
12875 when generated with FPL.

12878 2) Linux

12880 Re-add some /proc/acpi/event functionality that was lost during  
12881 the rewrite

12883 Resolved issue with /proc events for fixed-feature buttons showing  
12884 up as the system device.

12886 Fixed checks on C2/C3 latencies to be inclusive of maximum values.

12888 Replaced AE\_ERRORS in acpi\_osl.c with more specific error codes.

12890 Changed ACPI PRT option from "pci=noacpi-routing" to "pci=noacpi"

12892 Fixed limit interface & usage to fix bugs with passive cooling  
12893 hysteresis.

12895 Restructured PRT support.

12898 -----  
12899 Summary of changes for this label: 02\_14\_02

12902 1) ACPI CA Core Subsystem:

12904 Implemented support in AcpiLoadTable to allow loading of FACS and  
12905 FADT tables.

12907 Support for the now-obsolete interim 0.71 64-bit ACPI tables has  
12908 been removed. All 64-bit platforms should be migrated to the ACPI  
12909 2.0 tables. The actbl71.h header has been removed from the source  
12910 tree.

12912 All C macros defined within the subsystem have been prefixed with  
12913 "ACPI\_" to avoid collision with other system include files.

12915 Removed the return value for the two AcpiOsPrint interfaces, since  
12916 it is never used and causes lint warnings for ignoring the return  
12917 value.

12919 Added error checking to all internal mutex acquire and release  
12920 calls. Although a failure from one of these interfaces is  
12921 probably a fatal system error, these checks will cause the  
12922 immediate abort of the currently executing method or interface.

12924 Fixed a problem where the AcpiSetCurrentResources interface could  
12925 fault. This was a side effect of the deployment of the new memory  
12926 allocation model.

12928 Fixed a couple of problems with the Global Lock support introduced  
12929 in the last major build. The "common" (1.0/2.0) internal FACS was  
12930 being overwritten with the FACS signature and clobbering the  
12931 Global Lock pointer. Also, the actual firmware FACS was being  
12932 unmapped after construction of the "common" FACS, preventing  
12933 access to the actual Global Lock field within it. The "common"  
12934 internal FACS is no longer installed as an actual ACPI table; it  
12935 is used simply as a global.

12937 Code and Data Size: Current core subsystem library sizes are shown  
12938 below. These are the code and data sizes for the acpica.lib  
12939 produced by the Microsoft Visual C++ 6.0 compiler, and these  
12940 values do not include any ACPI driver or OSPM code. The debug  
12941 version of the code includes the debug output trace mechanism and  
12942 has a larger code and data size. Note that these values will vary  
12943 depending on the efficiency of the compiler and the compiler  
12944 options used during generation.

12946	Previous Release (02_07_01)			
12947	Non-Debug Version:	65.2K Code,	6.2K Data,	71.4K Total
12948	Debug Version:	136.9K Code,	56.4K Data,	193.3K Total
12949	Current Release:			
12950	Non-Debug Version:	65.4K Code,	6.2K Data,	71.6K Total
12951	Debug Version:	138.0K Code,	56.6K Data,	194.6K Total

12954 2) Linux

12956 Updated Linux-specific code for core macro and OSL interface  
12957 changes described above.

12959 Improved /proc/acpi/event. It now can be opened only once and has  
12960 proper poll functionality.

12962 Fixed and restructured power management (acpi\_bus).

12964 Only create /proc "view by type" when devices of that class exist.

12966 Fixed "charging/discharging" bug (and others) in acpi\_battery.

12968 Improved thermal zone code.

12971 3) ASL Compiler, version X2039:

12974 Implemented the new compiler restriction on ASL String hex/octal  
12975 escapes to non-null, ASCII values. An error results if an invalid  
12976 value is used. (This will require an ACPI 2.0 specification  
12977 change.)

12979 AML object labels that are output to the optional C and ASM source  
12980 are now prefixed with both the ACPI table signature and table ID  
12981 to help guarantee uniqueness within a large BIOS project.

12984 -----  
12985 Summary of changes for this label: 02\_01\_02

12987 1) ACPI CA Core Subsystem:

12989 ACPI 2.0 support is complete in the entire Core Subsystem and the  
12990 ASL compiler. All new ACPI 2.0 operators are implemented and all  
12991 other changes for ACPI 2.0 support are complete. With  
12992 simultaneous code and data optimizations throughout the subsystem,  
12993 ACPI 2.0 support has been implemented with almost no additional  
12994 cost in terms of code and data size.

12996 Implemented a new mechanism for allocation of return buffers. If  
12997 the buffer length is set to ACPI\_ALLOCATE\_BUFFER, the buffer will  
12998 be allocated on behalf of the caller. Consolidated all return  
12999 buffer validation and allocation to a common procedure. Return  
13000 buffers will be allocated via the primary OSL allocation interface  
13001 since it appears that a separate pool is not needed by most users.  
13002 If a separate pool is required for these buffers, the caller can  
13003 still use the original mechanism and pre-allocate the buffer(s).

13005 Implemented support for string operands within the DereferOf  
13006 operator.

13008 Restructured the Hardware and Event managers to be table driven,  
13009 simplifying the source code and reducing the amount of generated  
13010 code.

13012 Split the common read/write low-level ACPI register bitfield  
13013 procedure into a separate read and write, simplifying the code  
13014 considerably.

13016 Obsolete the AcpiOsCallocate OSL interface. This interface was  
13017 used only a handful of times and didn't have enough critical mass  
13018 for a separate interface. Replaced with a common calloc procedure  
13019 in the core.

13021 Fixed a reported problem with the GPE number mapping mechanism  
13022 that allows GPE1 numbers to be non-contiguous with GPE0.  
13023 Reorganized the GPE information and shrunk a large array that was

13024 originally large enough to hold info for all possible GPEs (256)  
 13025 to simply large enough to hold all GPEs up to the largest GPE  
 13026 number on the machine.

13028 Fixed a reported problem with resource structure alignment on 64-  
 13029 bit platforms.

13031 Changed the AcpiEnableEvent and AcpiDisableEvent external  
 13032 interfaces to not require any flags for the common case of  
 13033 enabling/disabling a GPE.

13035 Implemented support to allow a "Notify" on a Processor object.

13037 Most TBDs in comments within the source code have been resolved  
 13038 and eliminated.

13041 Fixed a problem in the interpreter where a standalone parent  
 13042 prefix (^) was not handled correctly in the interpreter and  
 13043 debugger.

13045 Removed obsolete and unnecessary GPE save/restore code.

13047 Implemented Field support in the ASL Load operator. This allows a  
 13048 table to be loaded from a named field, in addition to loading a  
 13049 table directly from an Operation Region.

13051 Implemented timeout and handle support in the external Global Lock  
 13052 interfaces.

13054 Fixed a problem in the AcpiDump utility where pathnames were no  
 13055 longer being generated correctly during the dump of named objects.

13057 Modified the AML debugger to give a full display of if/while  
 13058 predicates instead of just one AML opcode at a time. (The  
 13059 predicate can have several nested ASL statements.) The old method  
 13060 was confusing during single stepping.

13062 Code and Data Size: Current core subsystem library sizes are shown  
 13063 below. These are the code and data sizes for the acpica.lib  
 13064 produced by the Microsoft Visual C++ 6.0 compiler, and these  
 13065 values do not include any ACPI driver or OSPM code. The debug  
 13066 version of the code includes the debug output trace mechanism and  
 13067 has a larger code and data size. Note that these values will vary  
 13068 depending on the efficiency of the compiler and the compiler  
 13069 options used during generation.

13071	Previous Release (12_18_01)			
13072	Non-Debug Version:	66.1K Code,	5.5K Data,	71.6K Total
13073	Debug Version:	138.3K Code,	55.9K Data,	194.2K Total
13074	Current Release:			
13075	Non-Debug Version:	65.2K Code,	6.2K Data,	71.4K Total
13076	Debug Version:	136.9K Code,	56.4K Data,	193.3K Total

13078 2) Linux

13080 Implemented fix for PIIX reverse throttling errata (Processor  
 13081 driver)

13083 Added new Limit interface (Processor and Thermal drivers)

13085 New thermal policy (Thermal driver)

13087 Many updates to /proc

13089 Battery "low" event support (Battery driver)

13091 Supports ACPI PCI IRQ routing (PCI Link and PCI root drivers)

13093 IA32 - IA64 initialization unification, no longer experimental

13095 Menuconfig options redesigned

13097 3) ASL Compiler, version X2037:

13099 Implemented several new output features to simplify integration of  
 13100 AML code into firmware: 1) Output the AML in C source code with  
 13101 labels for each named ASL object. The original ASL source code  
 13102 is interleaved as C comments. 2) Output the AML in ASM source code  
 13103 with labels and interleaved ASL source. 3) Output the AML in  
 13104 raw hex table form, in either C or ASM.

13106 Implemented support for optional string parameters to the  
 13107 LoadTable operator.

13109 Completed support for embedded escape sequences within string  
 13110 literals. The compiler now supports all single character escapes  
 13111 as well as the Octal and Hex escapes. Note: the insertion of a  
 13112 null byte into a string literal (via the hex/octal escape) causes  
 13113 the string to be immediately terminated. A warning is issued.

13115 Fixed a problem where incorrect AML was generated for the case  
 13116 where an ASL namepath consists of a single parent prefix (  
 13118 ) with no trailing name segments.

13120 The compiler has been successfully generated with a 64-bit C  
 13121 compiler.

13126 -----  
 13127 Summary of changes for this label: 12\_18\_01

13129 1) Linux

13131 Enhanced blacklist with reason and severity fields. Any table's  
 13132 signature may now be used to identify a blacklisted system.

13134 Call \_PIC control method to inform the firmware which interrupt  
 13135 model the OS is using. Turn on any disabled link devices.

13137 Cleaned up busmgr /proc error handling (Andreas Dilger)

13139 2) ACPI CA Core Subsystem:

13141 Implemented ACPI 2.0 semantics for the "Break" operator (Exit from  
 13142 while loop)

13144 Completed implementation of the ACPI 2.0 "Continue",  
 13145 "ConcatenateResTemplate", "DataTableRegion", and "LoadTable"  
 13146 operators. All new ACPI 2.0 operators are now implemented in both  
 13147 the ASL compiler and the AML interpreter. The only remaining ACPI  
 13148 2.0 task is support for the String data type in the Derefof  
 13149 operator. Fixed a problem with AcquireMutex where the status code  
 13150 was lost if the caller had to actually wait for the mutex.

13152 Increased the maximum ASL Field size from 64K bits to 4G bits.

13154 Completed implementation of the external Global Lock interfaces --  
 13155 AcpiAcquireGlobalLock and AcpiReleaseGlobalLock. The Timeout and

13156 Handler parameters were added.

13158 Completed another pass at removing warnings and issues when  
13159 compiling with 64-bit compilers. The code now compiles cleanly  
13160 with the Intel 64-bit C/C++ compiler. Most notably, the pointer  
13161 add and subtract (diff) macros have changed considerably.

13164 Created and deployed a new ACPI\_SIZE type that is 64-bits wide on  
13165 64-bit platforms, 32-bits on all others. This type is used  
13166 wherever memory allocation and/or the C sizeof() operator is used,  
13167 and affects the OSL memory allocation interfaces AcpiOsAllocate  
13168 and AcpiOsCAllocate.

13170 Implemented sticky user breakpoints in the AML debugger.

13172 Code and Data Size: Current core subsystem library sizes are shown  
13173 below. These are the code and data sizes for the acpica.lib  
13174 produced by the Microsoft Visual C++ 6.0 compiler, and these  
13175 values do not include any ACPI driver or OSPM code. The debug  
13176 version of the code includes the debug output trace mechanism and  
13177 has a larger code and data size. Note that these values will vary  
13178 depending on the efficiency of the compiler and the compiler  
13179 options used during generation.

13181	Previous Release (12_05_01)			
13182	Non-Debug Version:	64.7K Code,	5.3K Data,	70.0K Total
13183	Debug Version:	136.2K Code,	55.6K Data,	191.8K Total
13184	Current Release:			
13185	Non-Debug Version:	66.1K Code,	5.5K Data,	71.6K Total
13186	Debug Version:	138.3K Code,	55.9K Data,	194.2K Total

13188 3) ASL Compiler, version X2034:

13190 Now checks for (and generates an error if detected) the use of a  
13191 Break or Continue statement without an enclosing While statement.

13194 Successfully generated the compiler with the Intel 64-bit C  
13195 compiler.

13197 -----  
13198 Summary of changes for this label: 12\_05\_01

13200 1) ACPI CA Core Subsystem:

13202 The ACPI 2.0 CopyObject operator is fully implemented. This  
13203 operator creates a new copy of an object (and is also used to  
13204 bypass the "implicit conversion" mechanism of the Store operator.)

13206 The ACPI 2.0 semantics for the SizeOf operator are fully  
13207 implemented. The change is that performing a SizeOf on a  
13208 reference object causes an automatic dereference of the object to  
13209 the actual value before the size is evaluated. This behavior was  
13210 undefined in ACPI 1.0.

13212 The ACPI 2.0 semantics for the Extended IRQ resource descriptor  
13213 have been implemented. The interrupt polarity and mode are now  
13214 independently set.

13216 Fixed a problem where ASL Constants (Zero, One, Ones, Revision)  
13217 appearing in Package objects were not properly converted to  
13218 integers when the internal Package was converted to an external  
13219 object (via the AcpiEvaluateObject interface.)

13221 Fixed a problem with the namespace object deletion mechanism for

13222 objects created by control methods. There were two parts to this  
13223 problem: 1) Objects created during the initialization phase method  
13224 parse were not being deleted, and 2) The object owner ID mechanism  
13225 to track objects was broken.

13227 Fixed a problem where the use of the ASL Scope operator within a  
13228 control method would result in an invalid opcode exception.

13230 Fixed a problem introduced in the previous label where the buffer  
13231 length required for the \_PRT structure was not being returned  
13232 correctly.

13234 Code and Data Size: Current core subsystem library sizes are shown  
13235 below. These are the code and data sizes for the acpica.lib  
13236 produced by the Microsoft Visual C++ 6.0 compiler, and these  
13237 values do not include any ACPI driver or OSPM code. The debug  
13238 version of the code includes the debug output trace mechanism and  
13239 has a larger code and data size. Note that these values will vary  
13240 depending on the efficiency of the compiler and the compiler  
13241 options used during generation.

13243	Previous Release (11_20_01)			
13244	Non-Debug Version:	64.1K Code,	5.3K Data,	69.4K Total
13245	Debug Version:	135.1K Code,	55.4K Data,	190.5K Total

13247	Current Release:			
13248	Non-Debug Version:	64.7K Code,	5.3K Data,	70.0K Total
13249	Debug Version:	136.2K Code,	55.6K Data,	191.8K Total

13251 2) Linux:

13253 Updated all files to apply cleanly against 2.4.16.

13255 Added basic PCI Interrupt Routing Table (PRT) support for IA32  
13256 (acpi\_pci.c), and unified the PRT code for IA32 and IA64. This  
13257 version supports both static and dynamic PRT entries, but dynamic  
13258 entries are treated as if they were static (not yet  
13259 reconfigurable). Architecture- specific code to use this data is  
13260 absent on IA32 but should be available shortly.

13262 Changed the initialization sequence to start the ACPI interpreter  
13263 (acpi\_init) prior to initialization of the PCI driver (pci\_init)  
13264 in init/main.c. This ordering is required to support PRT and  
13265 facilitate other (future) enhancement. A side effect is that the  
13266 ACPI bus driver and certain device drivers can no longer be loaded  
13267 as modules.

13269 Modified the 'make menuconfig' options to allow PCI Interrupt  
13270 Routing support to be included without the ACPI Bus and other  
13271 device drivers.

13273 3) ASL Compiler, version X2033:

13275 Fixed some issues with the use of the new CopyObject and  
13276 DataTableRegion operators. Both are fully functional.

13278 -----  
13279 Summary of changes for this label: 11\_20\_01

13281 20 November 2001. Summary of changes for this release.

13283 1) ACPI CA Core Subsystem:

13285 Updated Index support to match ACPI 2.0 semantics. Storing a  
13286 Integer, String, or Buffer to an Index of a Buffer will store only  
13287 the least-significant byte of the source to the Indexed buffer

13288 byte. Multiple writes are not performed.

13290 Fixed a problem where the access type used in an AccessAs ASL  
13291 operator was not recorded correctly into the field object.

13293 Fixed a problem where ASL Event objects were created in a  
13294 signalled state. Events are now created in an unsignalled state.

13296 The internal object cache is now purged after table loading and  
13297 initialization to reduce the use of dynamic kernel memory -- on  
13298 the assumption that object use is greatest during the parse phase  
13299 of the entire table (versus the run-time use of individual control  
13300 methods.)

13302 ACPI 2.0 variable-length packages are now fully operational.

13304 Code and Data Size: Code and Data optimizations have permitted new  
13305 feature development with an actual reduction in the library size.  
13306 Current core subsystem library sizes are shown below. These are  
13307 the code and data sizes for the acpica.lib produced by the  
13308 Microsoft Visual C++ 6.0 compiler, and these values do not include  
13309 any ACPI driver or OSPM code. The debug version of the code  
13310 includes the debug output trace mechanism and has a larger code  
13311 and data size. Note that these values will vary depending on the  
13312 efficiency of the compiler and the compiler options used during  
13313 generation.

13315	Previous Release (11_09_01):			
13316	Non-Debug Version:	63.7K Code,	5.2K Data,	68.9K Total
13317	Debug Version:	134.5K Code,	55.4K Data,	189.9K Total

13319	Current Release:			
13320	Non-Debug Version:	64.1K Code,	5.3K Data,	69.4K Total
13321	Debug Version:	135.1K Code,	55.4K Data,	190.5K Total

13323 2) Linux:

13325 Enhanced the ACPI boot-time initialization code to allow the use  
13326 of Local APIC tables for processor enumeration on IA-32, and to  
13327 pave the way for a fully MPS-free boot (on SMP systems) in the  
13328 near future. This functionality replaces  
13329 arch/i386/kernel/acpitables.c, which was introduced in an earlier  
13330 2.4.15-preX release. To enable this feature you must add  
13331 "acpi\_boot=on" to the kernel command line -- see the help entry  
13332 for CONFIG\_ACPI\_BOOT for more information. An IA-64 release is in  
13333 the works...

13335 Restructured the configuration options to allow boot-time table  
13336 parsing support without inclusion of the ACPI Interpreter (and  
13337 other) code.

13339 NOTE: This release does not include fixes for the reported events,  
13340 power-down, and thermal passive cooling issues (coming soon).

13342 3) ASL Compiler:

13344 Added additional typechecking for Fields within restricted access  
13345 Operation Regions. All fields within EC and CMOS regions must be  
13346 declared with ByteAcc. All fields withing SMBus regions must be  
13347 declared with the BufferAcc access type.

13349 Fixed a problem where the listing file output of control methods  
13350 no longer interleaved the actual AML code with the ASL source  
13351 code.

13356 -----  
13357 Summary of changes for this label: 11\_09\_01

13359 1) ACPI CA Core Subsystem:

13361 Implemented ACPI 2.0-defined support for writes to fields with a  
13362 Buffer, String, or Integer source operand that is smaller than the  
13363 target field. In these cases, the source operand is zero-extended  
13364 to fill the target field.

13366 Fixed a problem where a Field starting bit offset (within the  
13367 parent operation region) was calculated incorrectly if the

13369 alignment of the field differed from the access width. This  
13370 affected CreateWordField, CreateDwordField, CreateQwordField, and  
13371 possibly other fields that use the "AccessAny" keyword.

13373 Fixed a problem introduced in the 11\_02\_01 release where indirect  
13374 stores through method arguments did not operate correctly.

13376 2) Linux:

13378 Implemented boot-time ACPI table parsing support  
13379 (CONFIG\_ACPI\_BOOT) for IA32 and IA64 UP/SMP systems. This code  
13380 facilitates the use of ACPI tables (e.g. MADT, SRAT) rather than  
13381 legacy BIOS interfaces (e.g. MPS) for the configuration of system  
13382 processors, memory, and interrupts during setup\_arch(). Note that  
13383 this patch does not include the required architecture-specific  
13384 changes required to apply this information -- subsequent patches  
13385 will be posted for both IA32 and IA64 to achieve this.

13387 Added low-level sleep support for IA32 platforms, courtesy of Pat  
13388 Mochel. This allows IA32 systems to transition to/from various  
13389 sleeping states (e.g. S1, S3), although the lack of a centralized  
13390 driver model and power-manageable drivers will prevent its  
13391 (successful) use on most systems.

13393 Revamped the ACPI 'menuconfig' layout: created new "ACPI Support"  
13394 submenu, unified IA32 and IA64 options, added new "Boot using ACPI  
13395 tables" option, etc.

13397 Increased the default timeout for the EC driver from 1ms to 10ms  
13398 (1000 cycles of 10us) to try to address AE\_TIME errors during EC  
13399 transactions.

13401 -----  
13402 Summary of changes for this label: 11\_02\_01

13404 1) ACPI CA Core Subsystem:

13406 ACPI 2.0 Support: Implemented ACPI 2.0 64-bit Field access  
13407 (QWordAcc keyword). All ACPI 2.0 64-bit support is now  
13408 implemented.

13410 OSL Interfaces: Several of the OSL (AcpiOs\*) interfaces required  
13411 changes to support ACPI 2.0 Qword field access. Read/Write  
13412 PciConfiguration(), Read/Write Memory(), and Read/Write Port() now  
13413 accept an ACPI\_INTEGER (64 bits) as the value parameter. Also,  
13414 the value parameter for the address space handler interface is now  
13415 an ACPI\_INTEGER. OSL implementations of these interfaces must now  
13416 handle the case where the Width parameter is 64.

13418 Index Fields: Fixed a problem where unaligned bit assembly and  
13419 disassembly for IndexFields was not supported correctly.

13421 Index and Bank Fields: Nested Index and Bank Fields are now  
 13422 supported. During field access, a check is performed to ensure  
 13423 that the value written to an Index or Bank register is not out of  
 13424 the range of the register. The Index (or Bank) register is  
 13425 written before each access to the field data. Future support will  
 13426 include allowing individual IndexFields to be wider than the  
 13427 DataRegister width.

13429 Fields: Fixed a problem where the AML interpreter was incorrectly  
 13430 attempting to write beyond the end of a Field/OpRegion. This was  
 13431 a boundary case that occurred when a DWORD field was written to a  
 13432 BYTE access OpRegion, forcing multiple writes and causing the  
 13433 interpreter to write one datum too many.

13435 Fields: Fixed a problem with Field/OpRegion access where the  
 13436 starting bit address of a field was incorrectly calculated if the  
 13437 current access type was wider than a byte (WordAcc, DwordAcc, or  
 13438 QwordAcc).

13440 Fields: Fixed a problem where forward references to individual  
 13441 FieldUnits (individual Field names within a Field definition) were  
 13442 not resolved during the AML table load.

13444 Fields: Fixed a problem where forward references from a Field  
 13445 definition to the parent Operation Region definition were not  
 13446 resolved during the AML table load.

13448 Fields: Duplicate FieldUnit names within a scope are now detected  
 13449 during AML table load.

13451 Acpi Interfaces: Fixed a problem where the AcpiGetName() interface  
 13452 returned an incorrect name for the root node.

13454 Code and Data Size: Code and Data optimizations have permitted new  
 13455 feature development with an actual reduction in the library size.  
 13456 Current core subsystem library sizes are shown below. These are  
 13457 the code and data sizes for the acpica.lib produced by the  
 13458 Microsoft Visual C++ 6.0 compiler, and these values do not include  
 13459 any ACPI driver or OSPM code. The debug version of the code  
 13460 includes the debug output trace mechanism and has a larger code  
 13461 and data size. Note that these values will vary depending on the  
 13462 efficiency of the compiler and the compiler options used during  
 13463 generation.

13465	Previous Release (10_18_01):			
13466	Non-Debug Version:	63.9K Code,	5.1K Data,	69.0K Total
13467	Debug Version:	136.7K Code,	57.4K Data,	194.2K Total

13469	Current Release:			
13470	Non-Debug Version:	63.7K Code,	5.2K Data,	68.9K Total
13471	Debug Version:	134.5K Code,	55.4K Data,	189.9K Total

13473 2) Linux:

13475 Improved /proc processor output (Pavel Machek) Re-added  
 13476 MODULE\_LICENSE("GPL") to all modules.

13478 3) ASL Compiler version X2030:

13480 Duplicate FieldUnit names within a scope are now detected and  
 13481 flagged as errors.

13483 4) Documentation:

13485 Programmer Reference updated to reflect OSL and address space

13486 handler interface changes described above.

13488 -----  
 13489 Summary of changes for this label: 10\_18\_01

13491 ACPI CA Core Subsystem:

13493 Fixed a problem with the internal object reference count mechanism  
 13494 that occasionally caused premature object deletion. This resolves  
 13495 all of the outstanding problem reports where an object is deleted  
 13496 in the middle of an interpreter evaluation. Although this problem  
 13497 only showed up in rather obscure cases, the solution to the  
 13498 problem involved an adjustment of all reference counts involving  
 13499 objects attached to namespace nodes.

13501 Fixed a problem with Field support in the interpreter where  
 13502 writing to an aligned field whose length is an exact multiple (2  
 13503 or greater) of the field access granularity would cause an attempt  
 13504 to write beyond the end of the field.

13506 The top level AML opcode execution functions within the  
 13507 interpreter have been renamed with a more meaningful and  
 13508 consistent naming convention. The modules exmonad.c and  
 13509 exdyadic.c were eliminated. New modules are exoparg1.c,  
 13510 exoparg2.c, exoparg3.c, and exoparg6.c.

13512 Support for the ACPI 2.0 "Mid" ASL operator has been implemented.

13514 Fixed a problem where the AML debugger was causing some internal  
 13515 objects to not be deleted during subsystem termination.

13517 Fixed a problem with the external AcpiEvaluateObject interface  
 13518 where the subsystem would fault if the named object to be  
 13519 evaluated referred to a constant such as Zero, Ones, etc.

13521 Fixed a problem with IndexFields and BankFields where the  
 13522 subsystem would fault if the index, data, or bank registers were  
 13523 not defined in the same scope as the field itself.

13525 Added printf format string checking for compilers that support  
 13526 this feature. Corrected more than 50 instances of issues with  
 13527 format specifiers within invocations of ACPI\_DEBUG\_PRINT  
 13528 throughout the core subsystem code.

13530 The ASL "Revision" operator now returns the ACPI support level  
 13531 implemented in the core - the value "2" since the ACPI 2.0 support  
 13532 is more than 50% implemented.

13534 Enhanced the output of the AML debugger "dump namespace" command  
 13535 to output in a more human-readable form.

13537 Current core subsystem library code sizes are shown below. These

13539 are the code and data sizes for the acpica.lib produced by the  
 13540 Microsoft Visual C++ 6.0 compiler, and these values do not include  
 13541 any ACPI driver or OSPM code. The debug version of the code  
 13542 includes the full debug trace mechanism -- leading to a much

13544 larger code and data size. Note that these values will vary  
 13545 depending on the efficiency of the compiler and the compiler  
 13546 options used during generation.

13548	Previous Label (09_20_01):			
13549	Non-Debug Version:	65K Code,	5K Data,	70K Total
13550	Debug Version:	138K Code,	58K Data,	196K Total

13552        This Label:

13554        Non-Debug Version:   63.9K Code,    5.1K Data,    69.0K Total  
 13555        Debug Version:       136.7K Code,   57.4K Data,   194.2K Total

13557 Linux:

13559 Implemented a "Bad BIOS Blacklist" to track machines that have  
 13560 known ASL/AML problems.

13562 Enhanced the /proc interface for the thermal zone driver and added  
 13563 support for \_HOT (the critical suspend trip point). The 'info'  
 13564 file now includes threshold/policy information, and allows setting  
 13565 of \_SCP (cooling preference) and \_TZP (polling frequency) values  
 13566 to the 'info' file. Examples: "echo tzp=5 > info" sets the polling  
 13567 frequency to 5 seconds, and "echo scp=1 > info" sets the cooling  
 13568 preference to the passive/quiet mode (if supported by the ASL).

13570 Implemented a workaround for a gcc bug that resuted in an OOPs  
 13571 when loading the control method battery driver.

13573 -----  
 13574 Summary of changes for this label: 09\_20\_01

13576 ACPI CA Core Subsystem:

13578 The AcpiEnableEvent and AcpiDisableEvent interfaces have been  
 13579 modified to allow individual GPE levels to be flagged as wake-  
 13580 enabled (i.e., these GPEs are to remain enabled when the platform  
 13581 sleeps.)

13583 The AcpiEnterSleepState and AcpiLeaveSleepState interfaces now  
 13584 support wake-enabled GPEs. This means that upon entering the  
 13585 sleep state, all GPEs that are not wake-enabled are disabled.  
 13586 When leaving the sleep state, these GPEs are reenabled.

13588 A local double-precision divide/modulo module has been added to  
 13589 enhance portability to OS kernels where a 64-bit math library is  
 13590 not available. The new module is "utmath.c".

13592 Several optimizations have been made to reduce the use of CPU  
 13593 stack. Originally over 2K, the maximum stack usage is now below  
 13594 2K at 1860 bytes (1.82k)

13596 Fixed a problem with the AcpiGetFirmwareTable interface where the  
 13597 root table pointer was not mapped into a logical address properly.

13599 Fixed a problem where a NULL pointer was being dereferenced in the  
 13600 interpreter code for the ASL Notify operator.

13602 Fixed a problem where the use of the ASL Revision operator  
 13603 returned an error. This operator now returns the current version  
 13604 of the ACPI CA core subsystem.

13606 Fixed a problem where objects passed as control method parameters  
 13607 to AcpiEvaluateObject were always deleted at method termination.  
 13608 However, these objects may end up being stored into the namespace  
 13609 by the called method. The object reference count mechanism was  
 13610 applied to these objects instead of a force delete.

13612 Fixed a problem where static strings or buffers (contained in the  
 13613 AML code) that are declared as package elements within the ASL  
 13614 code could cause a fault because the interpreter would attempt to  
 13615 delete them. These objects are now marked with the "static  
 13616 object" flag to prevent any attempt to delete them.

13618 Implemented an interpreter optimization to use operands directly  
 13619 from the state object instead of extracting the operands to local  
 13620 variables. This reduces stack use and code size, and improves  
 13621 performance.

13623 The module exxface.c was eliminated as it was an unnecessary extra  
 13624 layer of code.

13626 Current core subsystem library code sizes are shown below. These  
 13627 are the code and data sizes for the acpica.lib produced by the  
 13628 Microsoft Visual C++ 6.0 compiler, and these values do not include  
 13629 any ACPI driver or OSPM code. The debug version of the code  
 13630 includes the full debug trace mechanism -- leading to a much  
 13631 larger code and data size. Note that these values will vary  
 13632 depending on the efficiency of the compiler and the compiler  
 13633 options used during generation.

13635        Non-Debug Version:   65K Code,    5K Data,    70K Total  
 13636        (Previously 69K)    Debug Version:   138K Code, 58K Data, 196K  
 13637        Total        (Previously 195K)

13639 Linux:

13641 Support for ACPI 2.0 64-bit integers has been added.    All ACPI  
 13642 Integer objects are now 64 bits wide

13644 All Acpi data types and structures are now in lower case.   Only  
 13645 Acpi macros are upper case for differentiation.

13647 Documentation:

13649 Changes to the external interfaces as described above.

13651 -----  
 13652 Summary of changes for this label: 08\_31\_01

13654 ACPI CA Core Subsystem:

13656 A bug with interpreter implementation of the ASL Divide operator  
 13657 was found and fixed. The implicit function return value (not the  
 13658 explicit store operands) was returning the remainder instead of  
 13659 the quotient. This was a longstanding bug and it fixes several  
 13660 known outstanding issues on various platforms.

13662 The ACPI\_DEBUG\_PRINT and function trace entry/exit macros have  
 13663 been further optimized for size. There are 700 invocations of the  
 13664 DEBUG\_PRINT macro alone, so each optimization reduces the size of  
 13665 the debug version of the subsystem significantly.

13667 A stack trace mechanism has been implemented. The maximum stack  
 13668 usage is about 2K on 32-bit platforms. The debugger command "stat  
 13669 stack" will display the current maximum stack usage.

13671 All public symbols and global variables within the subsystem are  
 13672 now prefixed with the string "Acpi". This keeps all of the  
 13673 symbols grouped together in a kernel map, and avoids conflicts  
 13674 with other kernel subsystems.

13676 Most of the internal fixed lookup tables have been moved into the  
 13677 code segment via the const operator.

13679 Several enhancements have been made to the interpreter to both  
 13680 reduce the code size and improve performance.

13682 Current core subsystem library code sizes are shown below. These  
 13683 are the code and data sizes for the acpica.lib produced by the

13684 Microsoft Visual C++ 6.0 compiler, and these values do not include  
 13685 any ACPI driver or OSPM code. The debug version of the code  
 13686 includes the full debug trace mechanism which contains over 700  
 13687 invocations of the DEBUG\_PRINT macro, 500 function entry macro  
 13688 invocations, and over 900 function exit macro invocations --  
 13689 leading to a much larger code and data size. Note that these  
 13690 values will vary depending on the efficiency of the compiler and  
 13691 the compiler options used during generation.

13693 Non-Debug Version: 64K Code, 5K Data, 69K Total  
 13694 Debug Version: 137K Code, 58K Data, 195K Total

13696 Linux:

13698 Implemented wbinvd() macro, pending a kernel-wide definition.

13700 Fixed /proc/acpi/event to handle poll() and short reads.

13702 ASL Compiler, version X2026:

13704 Fixed a problem introduced in the previous label where the AML

13706 code emitted for package objects produced packages with zero  
 13707 length.

13709 -----  
 13710 Summary of changes for this label: 08\_16\_01

13712 ACPI CA Core Subsystem:

13714 The following ACPI 2.0 ASL operators have been implemented in the  
 13715 AML interpreter (These are already supported by the Intel ASL  
 13716 compiler): ToDecimalString, ToHexString, ToString, ToInteger, and  
 13717 ToBuffer. Support for 64-bit AML constants is implemented in the  
 13718 AML parser, debugger, and disassembler.

13720 The internal memory tracking mechanism (leak detection code) has  
 13721 been upgraded to reduce the memory overhead (a separate tracking  
 13722 block is no longer allocated for each memory allocation), and now  
 13723 supports all of the internal object caches.

13725 The data structures and code for the internal object caches have  
 13726 been coalesced and optimized so that there is a single cache and  
 13727 memory list data structure and a single group of functions that  
 13728 implement generic cache management. This has reduced the code  
 13729 size in both the debug and release versions of the subsystem.

13731 The DEBUG\_PRINT macro(s) have been optimized for size and replaced  
 13732 by ACPI\_DEBUG\_PRINT. The syntax for this macro is slightly  
 13733 different, because it generates a single call to an internal  
 13734 function. This results in a savings of about 90 bytes per  
 13735 invocation, resulting in an overall code and data savings of about  
 13736 16% in the debug version of the subsystem.

13738 Linux:

13740 Fixed C3 disk corruption problems and re-enabled C3 on supporting  
 13741 machines.

13743 Integrated low-level sleep code by Patrick Mochel.

13745 Further tweaked source code Linuxization.

13747 Other minor fixes.

13749 ASL Compiler:

13751 Support for ACPI 2.0 variable length packages is fixed/completed.

13753 Fixed a problem where the optional length parameter for the ACPI  
 13754 2.0 ToString operator.

13756 Fixed multiple extraneous error messages when a syntax error is  
 13757 detected within the declaration line of a control method.

13759 -----  
 13760 Summary of changes for this label: 07\_17\_01

13762 ACPI CA Core Subsystem:

13764 Added a new interface named AcpiGetFirmwareTable to obtain any  
 13765 ACPI table via the ACPI signature. The interface can be called at  
 13766 any time during kernel initialization, even before the kernel  
 13767 virtual memory manager is initialized and paging is enabled. This  
 13768 allows kernel subsystems to obtain ACPI tables very early, even  
 13769 before the ACPI CA subsystem is initialized.

13771 Fixed a problem where Fields defined with the AnyAcc attribute  
 13772 could be resolved to the incorrect address under the following  
 13773 conditions: 1) the field width is larger than 8 bits and 2) the  
 13774 parent operation region is not defined on a DWORD boundary.

13776 Fixed a problem where the interpreter is not being locked during  
 13777 namespace initialization (during execution of the \_INI control  
 13778 methods), causing an error when an attempt is made to release it  
 13779 later.

13781 ACPI 2.0 support in the AML Interpreter has begun and will be  
 13782 ongoing throughout the rest of this year. In this label, The Mod  
 13783 operator is implemented.

13785 Added a new data type to contain full PCI addresses named  
 13786 ACPI\_PCI\_ID. This structure contains the PCI Segment, Bus, Device,  
 13787 and Function values.

13789 Linux:

13791 Enhanced the Linux version of the source code to change most  
 13792 capitalized ACPI type names to lowercase. For example, all  
 13793 instances of ACPI\_STATUS are changed to acpi\_status. This will  
 13794 result in a large diff, but the change is strictly cosmetic and  
 13795 aligns the CA code closer to the Linux coding standard.

13797 OSL Interfaces:

13799 The interfaces to the PCI configuration space have been changed to  
 13800 add the PCI Segment number and to split the single 32-bit combined  
 13801 DeviceFunction field into two 16-bit fields. This was  
 13802 accomplished by moving the four values that define an address in  
 13803 PCI configuration space (segment, bus, device, and function) to  
 13804 the new ACPI\_PCI\_ID structure.

13806 The changes to the PCI configuration space interfaces led to a  
 13807 reexamination of the complete set of address space access  
 13808 interfaces for PCI, I/O, and Memory. The previously existing 18  
 13809 interfaces have proven difficult to maintain (any small change  
 13810 must be propagated across at least 6 interfaces) and do not easily  
 13811 allow for future expansion to 64 bits if necessary. Also, on some  
 13812 systems, it would not be appropriate to demultiplex the access  
 13813 width (8, 16, 32, or 64) before calling the OSL if the  
 13814 corresponding native OS interfaces contain a similar access width  
 13815 parameter. For these reasons, the 18 address space interfaces



13816 have been replaced by these 6 new ones:

```
13818 AcpiOsReadPciConfiguration
13819 AcpiOsWritePciConfiguration
13820 AcpiOsReadMemory
13821 AcpiOsWriteMemory
13822 AcpiOsReadPort
13823 AcpiOsWritePort
```

13825 Added a new interface named AcpiOsGetRootPointer to allow the OSL  
13826 to perform the platform and/or OS-specific actions necessary to  
13827 obtain the ACPI RSDP table pointer. On IA-32 platforms, this  
13828 interface will simply call down to the CA core to perform the low-  
13829 memory search for the table. On IA-64, the RSDP is obtained from  
13830 EFI. Migrating this interface to the OSL allows the CA core to

13832 remain OS and platform independent.

13834 Added a new interface named AcpiOsSignal to provide a generic  
13835 "function code and pointer" interface for various miscellaneous  
13836 signals and notifications that must be made to the host OS. The  
13837 first such signals are intended to support the ASL Fatal and  
13838 Breakpoint operators. In the latter case, the AcpiOsBreakpoint  
13839 interface has been obsoleted.

13841 The definition of the AcpiFormatException interface has been  
13842 changed to simplify its use. The caller no longer must supply a  
13843 buffer to the call; A pointer to a const string is now returned  
13844 directly. This allows the call to be easily used in printf  
13845 statements, etc. since the caller does not have to manage a local  
13846 buffer.

13849 ASL Compiler, Version X2025:

13851 The ACPI 2.0 Switch/Case/Default operators have been implemented  
13852 and are fully functional. They will work with all ACPI 1.0  
13853 interpreters, since the operators are simply translated to If/Else  
13854 pairs.

13856 The ACPI 2.0 ElseIf operator is implemented and will also work  
13857 with 1.0 interpreters, for the same reason.

13859 Implemented support for ACPI 2.0 variable-length packages. These  
13860 packages have a separate opcode, and their size is determined by  
13861 the interpreter at run-time.

13863 Documentation The ACPI CA Programmer Reference has been updated to  
13864 reflect the new interfaces and changes to existing interfaces.

13866 -----  
13867 Summary of changes for this label: 06\_15\_01

13869 ACPI CA Core Subsystem:

13871 Fixed a problem where a DWORD-accessed field within a Buffer  
13872 object would get its byte address inadvertently rounded down to  
13873 the nearest DWORD. Buffers are always Byte-accessible.

13875 ASL Compiler, version X2024:

13877 Fixed a problem where the Switch() operator would either fault or  
13878 hang the compiler. Note however, that the AML code for this ACPI  
13879 2.0 operator is not yet implemented.

13881 Compiler uses the new AcpiOsGetTimer interface to obtain compile

13882 timings.

13884 Implementation of the CreateField operator automatically converts  
13885 a reference to a named field within a resource descriptor from a  
13886 byte offset to a bit offset if required.

13888 Added some missing named fields from the resource descriptor  
13889 support. These are the names that are automatically created by the  
13890 compiler to reference fields within a descriptor. They are only  
13891 valid at compile time and are not passed through to the AML  
13892 interpreter.

13894 Resource descriptor named fields are now typed as Integers and  
13895 subject to compile-time typechecking when used in expressions.

13897 -----  
13898 Summary of changes for this label: 05\_18\_01

13900 ACPI CA Core Subsystem:

13902 Fixed a couple of problems in the Field support code where bits  
13903 from adjacent fields could be returned along with the proper field  
13904 bits. Restructured the field support code to improve performance,  
13905 readability and maintainability.

13907 New DEBUG\_PRINTP macro automatically inserts the procedure name  
13908 into the output, saving hundreds of copies of procedure name  
13909 strings within the source, shrinking the memory footprint of the  
13910 debug version of the core subsystem.

13912 Source Code Structure:

13914 The source code directory tree was restructured to reflect the  
13915 current organization of the component architecture. Some files  
13916 and directories have been moved and/or renamed.

13918 Linux:

13920 Fixed leaking kacpidpc processes.

13922 Fixed queueing event data even when /proc/acpi/event is not  
13923 opened.

13925 ASL Compiler, version X2020:

13927 Memory allocation performance enhancement - over 24X compile time  
13928 improvement on large ASL files. Parse nodes and namestring  
13929 buffers are now allocated from a large internal compiler buffer.

13931 The temporary .SRC file is deleted unless the "-s" option is  
13932 specified

13934 The "-d" debug output option now sends all output to the .DBG file  
13935 instead of the console.

13937 "External" second parameter is now optional

13939 "ElseIf" syntax now properly allows the predicate

13941 Last operand to "Load" now recognized as a Target operand

13943 Debug object can now be used anywhere as a normal object.

13945 ResourceTemplate now returns an object of type BUFFER

13947 EISAID now returns an object of type INTEGER

13949 "Index" now works with a STRING operand

13951 "LoadTable" now accepts optional parameters

13953 "ToString" length parameter is now optional

13955 "Interrupt (ResourceType," parse error fixed.

13957 "Register" with a user-defined region space parse error fixed

13959 Escaped backslash at the end of a string ("\\") scan/parse error  
13960 fixed

13962 "Revision" is now an object of type INTEGER.

13966 -----  
13967 Summary of changes for this label: 05\_02\_01

13969 Linux:

13971 /proc/acpi/event now blocks properly.

13973 Removed /proc/sys/acpi. You can still dump your DSDT from  
13974 /proc/acpi/dsdt.

13976 ACPI CA Core Subsystem:

13978 Fixed a problem introduced in the previous label where some of the  
13979 "small" resource descriptor types were not recognized.

13981 Improved error messages for the case where an ASL Field is outside  
13982 the range of the parent operation region.

13984 ASL Compiler, version X2018:

13987 Added error detection for ASL Fields that extend beyond the length  
13988 of the parent operation region (only if the length of the region  
13989 is known at compile time.) This includes fields that have a  
13990 minimum access width that is smaller than the parent region, and  
13991 individual field units that are partially or entirely beyond the  
13992 extent of the parent.

13996 -----  
13997 Summary of changes for this label: 04\_27\_01

13999 ACPI CA Core Subsystem:

14001 Fixed a problem where the namespace mutex could be released at the  
14002 wrong time during execution of AcpiRemoveAddressSpaceHandler.

14004 Added optional thread ID output for debug traces, to simplify  
14005 debugging of multiple threads. Added context switch notification  
14006 when the debug code realizes that a different thread is now  
14007 executing ACPI code.

14009 Some additional external data types have been prefixed with the  
14010 string "ACPI\_" for consistency. This may effect existing code.  
14011 The data types affected are the external callback typedefs - e.g.,

14013 WALK\_CALLBACK becomes ACPI\_WALK\_CALLBACK.

14015 Linux:

14017 Fixed an issue with the OSL semaphore implementation where a  
14018 thread was waking up with an error from receiving a SIGCHLD  
14019 signal.

14021 Linux version of ACPI CA now uses the system C library for string  
14022 manipulation routines instead of a local implementation.

14024 Cleaned up comments and removed TBDs.

14026 ASL Compiler, version X2017:

14028 Enhanced error detection and reporting for all file I/O  
14029 operations.

14031 Documentation:

14033 Programmer Reference updated to version 1.06.

14037 -----  
14038 Summary of changes for this label: 04\_13\_01

14040 ACPI CA Core Subsystem:

14042 Restructured support for BufferFields and RegionFields.  
14043 BankFields support is now fully operational. All known 32-bit  
14044 limitations on field sizes have been removed. Both BufferFields  
14045 and (Operation) RegionFields are now supported by the same field  
14046 management code.

14048 Resource support now supports QWORD address and IO resources. The  
14049 16/32/64 bit address structures and the Extended IRQ structure  
14050 have been changed to properly handle Source Resource strings.

14052 A ThreadId of -1 is now used to indicate a "mutex not acquired"  
14053 condition internally and must never be returned by AcpiOsThreadId.  
14054 This reserved value was changed from 0 since Unix systems allow a  
14055 thread ID of 0.

14057 Linux:

14059 Driver code reorganized to enhance portability

14061 Added a kernel configuration option to control ACPI\_DEBUG

14063 Fixed the EC driver to honor \_GLK.

14065 ASL Compiler, version X2016:

14067 Fixed support for the "FixedHw" keyword. Previously, the FixedHw  
14068 address space was set to 0, not 0x7f as it should be.

14070 -----  
14071 Summary of changes for this label: 03\_13\_01

14073 ACPI CA Core Subsystem:

14075 During ACPI initialization, the \_SB\_.INI method is now run if  
14076 present.

14078 Notify handler fix - notifies are deferred until the parent method  
14079 completes execution. This fixes the "mutex already acquired"

14080 issue seen occasionally.

14082 Part of the "implicit conversion" rules in ACPI 2.0 have been  
14083 found to cause compatibility problems with existing ASL/AML. The  
14084 convert "result-to-target-type" implementation has been removed  
14085 for stores to method Args and Locals. Source operand conversion  
14086 is still fully implemented. Possible changes to ACPI 2.0  
14087 specification pending.

14089 Fix to AcpiRsCalculatePciRoutingTableLength to return correct  
14090 length.

14092 Fix for compiler warnings for 64-bit compiles.

14094 Linux:

14096 /proc output aligned for easier parsing.

14098 Release-version compile problem fixed.

14100 New kernel configuration options documented in Configure.help.

14102 IBM 600E - Fixed Sleep button may generate "Invalid <NULL>  
14103 context" message.

14105 OSPM:

14107 Power resource driver integrated with bus manager.

14109 Fixed kernel fault during active cooling for thermal zones.

14111 Source Code:

14113 The source code tree has been restructured.

14117 -----  
14118 Summary of changes for this label: 03\_02\_01

14120 Linux OS Services Layer (OSL):

14122 Major revision of all Linux-specific code.

14124 Modularized all ACPI-specific drivers.

14126 Added new thermal zone and power resource drivers.

14128 Revamped /proc interface (new functionality is under /proc/acpi).

14130 New kernel configuration options.

14132 Linux known issues:

14134 New kernel configuration options not documented in Configure.help  
14135 yet.

14138 Module dependencies not currently implemented. If used, they  
14139 should be loaded in this order: busmgr, power, ec, system,  
14140 processor, battery, ac\_adapter, button, thermal.

14142 Modules will not load if CONFIG\_MODVERSION is set.

14144 IBM 600E - entering S5 may reboot instead of shutting down.

14146 IBM 600E - Sleep button may generate "Invalid <NULL> context"  
14147 message.

14149 Some systems may fail with "execution mutex already acquired"  
14150 message.

14152 ACPI CA Core Subsystem:

14154 Added a new OSL Interface, AcpiOsGetThreadId. This was required  
14155 for the deadlock detection code. Defined to return a non-zero, 32-  
14156 bit thread ID for the currently executing thread. May be a non-  
14157 zero constant integer on single-thread systems.

14159 Implemented deadlock detection for internal subsystem mutexes. We  
14160 may add conditional compilation for this code (debug only) later.

14162 ASL/AML Mutex object semantics are now fully supported. This  
14163 includes multiple acquires/releases by owner and support for the

14165 Mutex SyncLevel parameter.

14167 A new "Force Release" mechanism automatically frees all ASL  
14168 Mutexes that have been acquired but not released when a thread  
14169 exits the interpreter. This forces conformance to the ACPI spec  
14170 ("All mutexes must be released when an invocation exits") and  
14171 prevents deadlocked ASL threads. This mechanism can be expanded  
14172 (later) to monitor other resource acquisitions if OEM ASL code  
14173 continues to misbehave (which it will).

14175 Several new ACPI exception codes have been added for the Mutex  
14176 support.

14178 Recursive method calls are now allowed and supported (the ACPI  
14179 spec does in fact allow recursive method calls.) The number of  
14180 recursive calls is subject to the restrictions imposed by the  
14181 SERIALIZED method keyword and SyncLevel (ACPI 2.0) method  
14182 parameter.

14184 Implemented support for the SyncLevel parameter for control  
14185 methods (ACPI 2.0 feature)

14187 Fixed a deadlock problem when multiple threads attempted to use  
14188 the interpreter.

14190 Fixed a problem where the string length of a String package  
14191 element was not always set in a package returned from  
14192 AcpiEvaluateObject.

14194 Fixed a problem where the length of a String package element was  
14195 not always included in the length of the overall package returned  
14196 from AcpiEvaluateObject.

14198 Added external interfaces (Acpi\*) to the ACPI debug memory  
14199 manager. This manager keeps a list of all outstanding  
14200 allocations, and can therefore detect memory leaks and attempts to  
14201 free memory blocks more than once. Useful for code such as the  
14202 power manager, etc. May not be appropriate for device drivers.  
14203 Performance with the debug code enabled is slow.

14205 The ACPI Global Lock is now an optional hardware element.

14207 ASL Compiler Version X2015:

14209 Integrated changes to allow the compiler to be generated on  
14210 multiple platforms.

14212 Linux makefile added to generate the compiler on Linux

14214 Source Code:

14216 All platform-specific headers have been moved to their own  
14217 subdirectory, Include/Platform.

14219 New source file added, Interpreter/ammutex.c

14221 New header file, Include/acstruct.h

14223 Documentation:

14225 The programmer reference has been updated for the following new  
14226 interfaces: AcpiOsGetThreadId AcpiAllocate AcpiCAllocate AcpiFree

14228 -----  
14229 Summary of changes for this label: 02\_08\_01

14231 Core ACPI CA Subsystem: Fixed a problem where an error was  
14232 incorrectly returned if the return resource buffer was larger than  
14233 the actual data (in the resource interfaces).

14235 References to named objects within packages are resolved to the  
14237 full pathname string before packages are returned directly (via  
14238 the AcpiEvaluateObject interface) or indirectly via the resource  
14239 interfaces.

14241 Linux OS Services Layer (OSL):

14243 Improved /proc battery interface.

14246 Added C-state debugging output and other miscellaneous fixes.

14248 ASL Compiler Version X2014:

14250 All defined method arguments can now be used as local variables,  
14251 including the ones that are not actually passed in as parameters.  
14252 The compiler tracks initialization of the arguments and issues an  
14253 exception if they are used without prior assignment (just like  
14254 locals).

14256 The -o option now specifies a filename prefix that is used for all  
14257 output files, including the AML output file. Otherwise, the  
14258 default behavior is as follows: 1) the AML goes to the file  
14259 specified in the DSDT. 2) all other output files use the input  
14260 source filename as the base.

14262 -----  
14263 Summary of changes for this label: 01\_25\_01

14265 Core ACPI CA Subsystem: Restructured the implementation of object  
14266 store support within the interpreter. This includes support for  
14267 the Store operator as well as any ASL operators that include a  
14268 target operand.

14270 Partially implemented support for Implicit Result-to-Target  
14271 conversion. This is when a result object is converted on the fly  
14272 to the type of an existing target object. Completion of this  
14273 support is pending further analysis of the ACPI specification  
14274 concerning this matter.

14276 CPU-specific code has been removed from the subsystem (hardware  
14277 directory).

14279 New Power Management Timer functions added

14281 Linux OS Services Layer (OSL): Moved system state transition code  
14282 to the core, fixed it, and modified Linux OSL accordingly.

14284 Fixed C2 and C3 latency calculations.

14287 We no longer use the compilation date for the version message on  
14288 initialization, but retrieve the version from AcpiGetSystemInfo().

14290 Incorporated for fix Sony VAIO machines.

14292 Documentation: The Programmer Reference has been updated and  
14293 reformatted.

14296 ASL Compiler: Version X2013: Fixed a problem where the line  
14297 numbering and error reporting could get out of sync in the  
14298 presence of multiple include files.

14300 -----  
14301 Summary of changes for this label: 01\_15\_01

14303 Core ACPI CA Subsystem:

14305 Implemented support for type conversions in the execution of the  
14306 ASL Concatenate operator (The second operand is converted to  
14307 match the type of the first operand before concatenation.)

14309 Support for implicit source operand conversion is partially  
14310 implemented. The ASL source operand types Integer, Buffer, and  
14311 String are freely interchangeable for most ASL operators and are  
14312 converted by the interpreter on the fly as required. Implicit  
14313 Target operand conversion (where the result is converted to the  
14314 target type before storing) is not yet implemented.

14316 Support for 32-bit and 64-bit BCD integers is implemented.

14318 Problem fixed where a field read on an aligned field could cause a  
14319 read past the end of the field.

14321 New exception, AE\_AML\_NO\_RETURN\_VALUE, is returned when a method  
14322 does not return a value, but the caller expects one. (The ASL  
14323 compiler flags this as a warning.)

14325 ASL Compiler:

14327 Version X2011:

14328 1. Static typechecking of all operands is implemented. This  
14329 prevents the use of invalid objects (such as using a Package where  
14330 an Integer is required) at compile time instead of at interpreter  
14331 run-time.

14332 2. The ASL source line is printed with ALL errors and warnings.

14333 3. Bug fix for source EOF without final linefeed.

14334 4. Debug option is split into a parse trace and a namespace trace.

14335 5. Namespace output option (-n) includes initial values for  
14336 integers and strings.

14337 6. Parse-only option added for quick syntax checking.

14338 7. Compiler checks for duplicate ACPI name declarations

14340 Version X2012:

14341 1. Relaxed typechecking to allow interchangeability between  
14342 strings, integers, and buffers. These types are now converted by  
14343 the interpreter at runtime.

14344 2. Compiler reports time taken by each internal subsystem in the  
14345 debug output file.

14348 -----  
14349 Summary of changes for this label: 12\_14\_00

14351 ASL Compiler:

14353 This is the first official release of the compiler. Since the  
14354 compiler requires elements of the Core Subsystem, this label  
14355 synchronizes everything.

14357 -----  
14358 Summary of changes for this label: 12\_08\_00

14361 Fixed a problem where named references within the ASL definition  
14362 of both OperationRegions and CreateXXXFields did not work  
14363 properly. The symptom was an AE\_AML\_OPERAND\_TYPE during  
14364 initialization of the region/field. This is similar (but not  
14365 related internally) to the problem that was fixed in the last  
14366 label.

14368 Implemented both 32-bit and 64-bit support for the BCD ASL  
14369 functions ToBCD and FromBCD.

14371 Updated all legal headers to include "2000" in the copyright  
14372 years.

14374 -----  
14375 Summary of changes for this label: 12\_01\_00

14377 Fixed a problem where method invocations within the ASL definition  
14378 of both OperationRegions and CreateXXXFields did not work  
14379 properly. The symptom was an AE\_AML\_OPERAND\_TYPE during  
14380 initialization of the region/field:

14382 nsinit-0209: AE\_AML\_OPERAND\_TYPE while getting region arguments  
14383 [DEBG] ammonad-0284: Exec\_monadic2\_r/Not: bad operand(s)  
14384 (0x3005)

14386 Fixed a problem where operators with more than one nested  
14387 subexpression would fail. The symptoms were varied, by mostly  
14388 AE\_AML\_OPERAND\_TYPE errors. This was actually a rather serious  
14389 problem that has gone unnoticed until now.

14391 Subtract (Add (1,2), Multiply (3,4))

14393 Fixed a problem where AcpiGetHandle didn't quite get fixed in the  
14394 previous build (The prefix part of a relative path was handled  
14395 incorrectly).

14397 Fixed a problem where Operation Region initialization failed if  
14398 the operation region name was a "namepath" instead of a simple  
14399 "nameseg". Symptom was an AE\_NO\_OPERAND error.

14401 Fixed a problem where an assignment to a local variable via the  
14402 indirect RefOf mechanism only worked for the first such  
14403 assignment. Subsequent assignments were ignored.

14405 -----  
14406 Summary of changes for this label: 11\_15\_00

14408 ACPI 2.0 table support with backwards support for ACPI 1.0 and the  
14409 0.71 extensions. Note: although we can read ACPI 2.0 BIOS tables,

14410 the AML interpreter does NOT have support for the new 2.0 ASL  
14411 grammar terms at this time.

14413 All ACPI hardware access is via the GAS structures in the ACPI 2.0  
14414 FADT.

14416 All physical memory addresses across all platforms are now 64 bits  
14417 wide. Logical address width remains dependent on the platform  
14418 (i.e., "void \*").

14420 AcpiOsMapMemory interface changed to a 64-bit physical address.

14422 The AML interpreter integer size is now 64 bits, as per the ACPI  
14423 2.0 specification.

14425 For backwards compatibility with ACPI 1.0, ACPI tables with a  
14426 revision number less than 2 use 32-bit integers only.

14428 Fixed a problem where the evaluation of OpRegion operands did not  
14429 always resolve them to numbers properly.

14431 -----  
14432 Summary of changes for this label: 10\_20\_00

14434 Fix for CBN\_.\_STA issue. This fix will allow correct access to  
14435 CBN\_ OpRegions when the \_STA returns 0x8.

14437 Support to convert ACPI constants (Ones, Zeros, One) to actual  
14438 values before a package object is returned

14440 Fix for method call as predicate to if/while construct causing  
14441 incorrect if/while behavior

14443 Fix for Else block package lengths sometimes calculated wrong (if  
14444 block > 63 bytes)

14446 Fix for Processor object length field, was always zero

14448 Table load abort if FACP sanity check fails

14450 Fix for problem with Scope(name) if name already exists

14452 Warning emitted if a named object referenced cannot be found  
14453 (resolved) during method execution.

14459 -----  
14460 Summary of changes for this label: 9\_29\_00

14462 New table initialization interfaces: AcpiInitializeSubsystem no  
14463 longer has any parameters AcpiFindRootPointer - Find the RSDP (if  
14464 necessary) AcpiLoadTables (RSDP) - load all tables found at RSDP-  
14465 >RSDT Obsolete Interfaces AcpiLoadFirmwareTables - replaced by  
14466 AcpiLoadTables

14468 Note: These interface changes require changes to all existing OSDs

14470 The PCI\_Config default address space handler is always installed  
14471 at the root namespace object.

14473 -----  
14474 Summary of changes for this label: 09\_15\_00

14476 The new initialization architecture is implemented. New  
 14477 interfaces are: AcpiInitializeSubsystem (replaces AcpiInitialize)  
 14478 AcpiEnableSubsystem Obsolete Interfaces: AcpiLoadNamespace

14480 (Namespace is automatically loaded when a table is loaded)

14482 The ACPI\_OPERAND\_OBJECT has been optimized to shrink its size from  
 14483 52 bytes to 32 bytes. There is usually one of these for every  
 14484 namespace object, so the memory savings is significant.

14486 Implemented just-in-time evaluation of the CreateField operators.

14488 Bug fixes for IA-64 support have been integrated.

14490 Additional code review comments have been implemented

14492 The so-called "third pass parse" has been replaced by a final walk  
 14493 through the namespace to initialize all operation regions (address  
 14494 spaces) and fields that have not yet been initialized during the  
 14495 execution of the various \_INI and REG methods.

14497 New file - namespace/nsinit.c

14499 -----  
 14500 Summary of changes for this label: 09\_01\_00

14502 Namespace manager data structures have been reworked to change the  
 14503 primary object from a table to a single object. This has  
 14504 resulted in dynamic memory savings of 3X within the namespace and  
 14505 2X overall in the ACPI CA subsystem.

14507 Fixed problem where the call to AcpiEvFindPciRootBuses was  
 14508 inadvertently left commented out.

14510 Reduced the warning count when generating the source with the GCC  
 14511 compiler.

14513 Revision numbers added to each module header showing the  
 14514 SourceSafe version of the file. Please refer to this version  
 14515 number when giving us feedback or comments on individual modules.

14517 The main object types within the subsystem have been renamed to  
 14518 clarify their purpose:

14520 ACPI\_INTERNAL\_OBJECT -> ACPI\_OPERAND\_OBJECT  
 14521 ACPI\_GENERIC\_OP -> ACPI\_PARSE\_OBJECT  
 14522 ACPI\_NAME\_TABLE\_ENTRY -> ACPI\_NAMESPACE\_NODE

14524 NOTE: no changes to the initialization sequence are included in  
 14525 this label.

14527 -----  
 14528 Summary of changes for this label: 08\_23\_00

14530 Fixed problem where TerminateControlMethod was being called  
 14531 multiple times per method

14533 Fixed debugger problem where single stepping caused a semaphore to  
 14534 be oversignalled

14536 Improved performance through additional parse object caching -  
 14537 added ACPI\_EXTENDED\_OP type

14539 -----  
 14540 Summary of changes for this label: 08\_10\_00

14542 Parser/Interpreter integration: Eliminated the creation of  
 14543 complete parse trees for ACPI tables and control methods.  
 14544 Instead, parse subtrees are created and then deleted as soon as  
 14545 they are processed (Either entered into the namespace or executed  
 14546 by the interpreter). This reduces the use of dynamic kernel  
 14547 memory significantly. (about 10X)

14549 Exception codes broken into classes and renumbered. Be sure to  
 14550 recompile all code that includes acexcep.h. Hopefully we won't  
 14551 have to renumber the codes again now that they are split into  
 14552 classes (environment, programmer, AML code, ACPI table, and  
 14553 internal).

14555 Fixed some additional alignment issues in the Resource Manager  
 14556 subcomponent

14558 Implemented semaphore tracking in the AcpiExec utility, and fixed  
 14559 several places where mutexes/semaphores were being unlocked  
 14560 without a corresponding lock operation. There are no known  
 14561 semaphore or mutex "leaks" at this time.

14563 Fixed the case where an ASL Return operator is used to return an  
 14564 unnamed package.

14566 -----  
 14567 Summary of changes for this label: 07\_28\_00

14569 Fixed a problem with the way addresses were calculated in  
 14570 AcpiAmlReadFieldData() and AcpiAmlWriteFieldData(). This problem  
 14571 manifested itself when a Field was created with WordAccess or  
 14572 DwordAccess, but the field unit defined within the Field was less  
 14574 than a Word or Dword.

14576 Fixed a problem in AmlDumpOperands() module's loop to pull  
 14577 operands off of the operand stack to display information. The  
 14578 problem manifested itself as a TLB error on 64-bit systems when  
 14579 accessing an operand stack with two or more operands.

14581 Fixed a problem with the PCI configuration space handlers where  
 14582 context was getting confused between accesses. This required a  
 14583 change to the generic address space handler and address space  
 14584 setup definitions. Handlers now get both a global handler context  
 14585 (this is the one passed in by the user when executing  
 14586 AcpiInstallAddressSpaceHandler()) and a specific region context  
 14587 that is unique to each region (For example, the \_ADR, \_SEG and  
 14588 \_BBN values associated with a specific region). The generic  
 14589 function definitions have changed to the following:

14591 typedef ACPI\_STATUS (\*ADDRESS\_SPACE\_HANDLER) ( UINT32 Function,  
 14592 UINT32 Address, UINT32 BitWidth, UINT32 \*Value, void  
 14593 \*HandlerContext, // This used to be void \*Context void  
 14594 \*RegionContext); // This is an additional parameter

14596 typedef ACPI\_STATUS (\*ADDRESS\_SPACE\_SETUP) ( ACPI\_HANDLE  
 14597 RegionHandle, UINT32 Function, void \*HandlerContext, void  
 14598 \*\*RegionContext); // This used to be \*\*ReturnContext

14600 -----  
 14601 Summary of changes for this label: 07\_21\_00

14603 Major file consolidation and rename. All files within the  
 14604 interpreter have been renamed as well as most header files. This  
 14605 was done to prevent collisions with existing files in the host  
 14606 OSs -- filenames such as "config.h" and "global.h" seem to be  
 14607 quite common. The VC project files have been updated. All

14608 makefiles will require modification.

14610 The parser/interpreter integration continues in Phase 5 with the  
14611 implementation of a complete 2-pass parse (the AML is parsed  
14612 twice) for each table; This avoids the construction of a huge  
14613 parse tree and therefore reduces the amount of dynamic memory  
14614 required by the subsystem. Greater use of the parse object cache  
14615 means that performance is unaffected.

14617 Many comments from the two code reviews have been rolled in.

14619 The 64-bit alignment support is complete.

14621 -----  
14622 Summary of changes for this label: 06\_30\_00

14624 With a nod and a tip of the hat to the technology of yesteryear,  
14625 we've added support in the source code for 80 column output  
14626 devices. The code is now mostly constrained to 80 columns or  
14627 less to support environments and editors that 1) cannot display  
14628 or print more than 80 characters on a single line, and 2) cannot  
14629 disable line wrapping.

14631 A major restructuring of the namespace data structure has been  
14632 completed. The result is 1) cleaner and more  
14633 understandable/maintainable code, and 2) a significant reduction  
14634 in the dynamic memory requirement for each named ACPI object  
14635 (almost half).

14637 -----  
14638 Summary of changes for this label: 06\_23\_00

14640 Linux support has been added. In order to obtain approval to get  
14641 the ACPI CA subsystem into the Linux kernel, we've had to make  
14642 quite a few changes to the base subsystem that will affect all  
14643 users (all the changes are generic and OS- independent). The  
14644 effects of these global changes have been somewhat far reaching.  
14645 Files have been merged and/or renamed and interfaces have been  
14646 renamed. The major changes are described below.

14648 Osd\* interfaces renamed to AcpiOs\* to eliminate namespace  
14649 pollution/confusion within our target kernels. All OSD  
14650 interfaces must be modified to match the new naming convention.

14652 Files merged across the subsystem. A number of the smaller source  
14653 and header files have been merged to reduce the file count and  
14654 increase the density of the existing files. There are too many  
14655 to list here. In general, makefiles that call out individual  
14656 files will require rebuilding.

14658 Interpreter files renamed. All interpreter files now have the  
14659 prefix am\* instead of ie\* and is\*.

14661 Header files renamed: The acapi.h file is now acpixf.h. The  
14662 apciiosd.h file is now apciiosxf.h. We are removing references to  
14663 the acronym "API" since it is somewhat windowsy. The new name is  
14664 "external interface" or xface or xf in the filenames.j

14667 All manifest constants have been forced to upper case (some were  
14668 mixed case.) Also, the string "ACPI\_" has been prepended to many  
14669 (not all) of the constants, typedefs, and structs.

14671 The globals "DebugLevel" and "DebugLayer" have been renamed  
14672 "AcpiDbgLevel" and "AcpiDbgLayer" respectively.

14674 All other globals within the subsystem are now prefixed with  
14675 "AcpiGbl\_" Internal procedures within the subsystem are now  
14676 prefixed with "Acpi" (with only a few exceptions). The original  
14677 two-letter abbreviation for the subcomponent remains after "Acpi"  
14678 - for example, CmCAllocate became AcpiCmCAllocate.

14680 Added a source code translation/conversion utility. Used to  
14681 generate the Linux source code, it can be modified to generate  
14682 other types of source as well. Can also be used to cleanup  
14683 existing source by removing extraneous spaces and blank lines.  
14684 Found in tools/acpisrc/\*

14686 OsdUnMapMemory was renamed to OsdUnmapMemory and then  
14687 AcpiOsUnmapMemory. (UnMap became Unmap).

14689 A "MaxUnits" parameter has been added to AcpiOsCreateSemaphore.  
14690 When set to one, this indicates that the caller wants to use the

14692 semaphore as a mutex, not a counting semaphore. ACPI CA uses  
14693 both types. However, implementers of this call may want to use  
14694 different OS primitives depending on the type of semaphore  
14695 requested. For example, some operating systems provide separate

14697 "mutex" and "semaphore" interfaces - where the mutex interface is  
14698 much faster because it doesn't have all the overhead of a full  
14699 semaphore implementation.

14701 Fixed a deadlock problem where a method that accesses the PCI  
14702 address space can block forever if it is the first access to the  
14703 space.

14705 -----  
14706 Summary of changes for this label: 06\_02\_00

14708 Support for environments that cannot handle unaligned data  
14709 accesses (e.g. firmware and OS environments devoid of alignment  
14710 handler technology namely SAL/EFI and the IA-64 Linux kernel) has  
14711 been added (via configurable macros) in these three areas: -  
14712 Transfer of data from the raw AML byte stream is done via byte  
14713 moves instead of word/dword/qword moves. - External objects are  
14714 aligned within the user buffer, including package elements (sub-  
14715 objects). - Conversion of name strings to UINT32 Acpi Names is now  
14716 done byte-wise.

14718 The Store operator was modified to mimic Microsoft's  
14719 implementation when storing to a Buffer Field.

14721 Added a check of the BM\_STS bit before entering C3.

14723 The methods subdirectory has been obsoleted and removed. A new  
14724 file, cmeval.c subsumes the functionality.

14726 A 16-bit (DOS) version of AcpiExec has been developed. The  
14727 makefile is under the acpiexec directory.

```

*****
13858 Fri Jan 17 07:29:28 2014
new/usr/src/common/acpica/common/acgetline.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
*****
1 /*****
2 *
3 * Module Name: acgetline - local line editing
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "amlcode.h"
47 #include "acparser.h"
48 #include "acdebug.h"

50 #include <stdio.h>

52 /*
53 * This is an os-independent implementation of line-editing services needed
54 * by the AcpiExec utility. It uses getchar() and putchar() and the existing
55 * history support provided by the AML debugger. It assumes that the terminal
56 * is in the correct line-editing mode such as raw and noecho. The OSL
57 * interface AcpiOsInitialize should do this. AcpiOsTerminate should put the
58 * terminal back into the original mode.
59 */
60 #define _COMPONENT          ACPI_OS_SERVICES

```

```

61         ACPI_MODULE_NAME    ("acgetline")

64 /* Local prototypes */

66 static void
67 AcpiAcClearLine (
68     UINT32                EndOfLine,
69     UINT32                CursorPosition);

71 /* Various ASCII constants */

73 #define _ASCII_NUL                0
74 #define _ASCII_BACKSPACE          0x08
75 #define _ASCII_TAB                0x09
76 #define _ASCII_ESCAPE            0x1B
77 #define _ASCII_SPACE             0x20
78 #define _ASCII_LEFT_BRACKET      0x5B
79 #define _ASCII_DEL               0x7F
80 #define _ASCII_UP_ARROW          'A'
81 #define _ASCII_DOWN_ARROW        'B'
82 #define _ASCII_RIGHT_ARROW       'C'
83 #define _ASCII_LEFT_ARROW        'D'
84 #define _ASCII_NEWLINE           '\n'

86 extern UINT32                AcpiGbl_NextCmdNum;

88 /* Erase a single character on the input command line */

90 #define ACPI_CLEAR_CHAR() \
91     putchar (_ASCII_BACKSPACE); \
92     putchar (_ASCII_SPACE); \
93     putchar (_ASCII_BACKSPACE);

95 /* Backup cursor by Count positions */

97 #define ACPI_BACKUP_CURSOR(i, Count) \
98     for (i = 0; i < (Count); i++) \
99         {putchar (_ASCII_BACKSPACE);}

102 /*****
103 *
104 * FUNCTION:      AcpiAcClearLine
105 *
106 * PARAMETERS:    EndOfLine          - Current end-of-line index
107 *                CursorPosition      - Current cursor position within line
108 *
109 * RETURN:        None
110 *
111 * DESCRIPTION:   Clear the entire command line the hard way, but probably the
112 *                most portable.
113 *
114 *****/

116 static void
117 AcpiAcClearLine (
118     UINT32                EndOfLine,
119     UINT32                CursorPosition)
120 {
121     UINT32                i;

124     if (CursorPosition < EndOfLine)
125     {
126         /* 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     }
398
399     PreviousCommandIndex = CurrentCommandIndex;
400     CurrentCommandIndex++;
401
402     /* Get the historical command from the debugger */
403
404     NextCommand = AcpiDbGetHistoryByIndex (CurrentCommandIndex);
405     if (!NextCommand)
406     {
407         return (AE_ERROR);
408     }
409
410     /* Make this the active command and echo it */
411
412     AcpiAcClearLine (EndOfLine, CursorPosition);
413     strcpy (Buffer, NextCommand);
414     fprintf (stdout, "%s", Buffer);
415     EndOfLine = CursorPosition = strlen (Buffer);
416     continue;
417
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;
430
431     default:
432
433         /* Ignore random escape sequences that we don't care about */
434
435         continue;
436     }
437 }
438 }
439 }
440 }
```

new/usr/src/common/acpica/common/adfile.c

1

```
*****
9651 Fri Jan 17 07:29:28 2014
new/usr/src/common/acpica/common/adfile.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: adfile - Application-level disassembler file support routines
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "acpi.h"
45 #include "accommon.h"
46 #include "acapps.h"
47
48 #include <stdio.h>
49
50 #define _COMPONENT          ACPI_TOOLS
51 #define ACPI_MODULE_NAME    ("adfile")
52
53 /* Local prototypes */
```

new/usr/src/common/acpica/common/adfile.c

2

```
57 static INT32
58 AdWriteBuffer (
59     char                *Filename,
60     char                *Buffer,
61     UINT32              Length);
62
63 static char            FilenameBuf[20];
64
65 /*****
66 *
67 * FUNCTION:      AfGenerateFilename
68 *
69 * PARAMETERS:    Prefix            - prefix string
70 *                TableId          - The table ID
71 *
72 * RETURN:        Pointer to the completed string
73 *
74 * DESCRIPTION:   Build an output filename from an ACPI table ID string
75 *
76 * *****/
77
78 char *
79 AdGenerateFilename (
80     char                *Prefix,
81     char                *TableId)
82 {
83     UINT32              i;
84     UINT32              j;
85
86     for (i = 0; Prefix[i]; i++)
87     {
88         FilenameBuf[i] = Prefix[i];
89     }
90
91     FilenameBuf[i] = '_';
92     i++;
93
94     for (j = 0; j < 8 && (TableId[j] != ' ') && (TableId[j] != 0); j++, i++)
95     {
96         FilenameBuf[i] = TableId[j];
97     }
98
99     FilenameBuf[i] = 0;
100     strcat (FilenameBuf, ACPI_TABLE_FILE_SUFFIX);
101     return (FilenameBuf);
102 }
103
104 /*****
105 *
106 * FUNCTION:      AfWriteBuffer
107 *
108 * PARAMETERS:    Filename          - name of file
109 *                Buffer            - data to write
110 *                Length           - length of data
111 *
112 * RETURN:        Actual number of bytes written
113 *
114 * DESCRIPTION:   Open a file and write out a single buffer
115 *
116 * *****/
117
118 static INT32
119 AdWriteBuffer (
```

```

123     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 }

```

new/usr/src/common/acpica/common/adisasm.c

1

```
*****
27624 Fri Jan 17 07:29:29 2014
new/usr/src/common/acpica/common/adisasm.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: adisasm - Application-level disassembler routines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"
48 #include "amlcode.h"
49 #include "acdebug.h"
50 #include "acdisasm.h"
51 #include "acdispat.h"
52 #include "acnamesp.h"
53 #include "actables.h"
54 #include "acapps.h"

56 #include <stdio.h>
57 #include <time.h>
```

new/usr/src/common/acpica/common/adisasm.c

2

```
60 #define _COMPONENT ACPI_TOOLS
61 ACPI_MODULE_NAME ("adisasm")

63 /*
64 * Older versions of Bison won't emit this external in the generated header.
65 * Newer versions do emit the external, so we don't need to do it.
66 */
67 #ifndef ASLCOMPILER_ASLCOMPILERPARSE_H
68 extern int AslCompilerdebug;
69 #endif

71 ACPI_STATUS
72 NsDisplayNamespace (
73     void);

75 void
76 NsSetupNamespaceListing (
77     void *Handle);

80 /* Local prototypes */

82 static UINT32
83 AdGetFileSize (
84     FILE *File);

86 static void
87 AdCreateTableHeader (
88     char *Filename,
89     ACPI_TABLE_HEADER *Table);

91 /* Stubs for ASL compiler */

93 #ifndef ACPI_ASL_COMPILER
94 BOOLEAN
95 AcpiDsIsResultUsed (
96     ACPI_PARSE_OBJECT *Op,
97     ACPI_WALK_STATE *WalkState)
98 {
99     return TRUE;
100 }

102 ACPI_STATUS
103 AcpiDsMethodError (
104     ACPI_STATUS Status,
105     ACPI_WALK_STATE *WalkState)
106 {
107     return (Status);
108 }
109 #endif

111 ACPI_STATUS
112 AcpiNsLoadTable (
113     UINT32 TableIndex,
114     ACPI_NAMESPACE_NODE *Node)
115 {
116     return (AE_NOT_IMPLEMENTED);
117 }

119 ACPI_STATUS
120 AcpiDsRestartControlMethod (
121     ACPI_WALK_STATE *WalkState,
122     ACPI_OPERAND_OBJECT *ReturnDesc)
123 {
124     return (AE_OK);
125 }
```

```

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         AcpiNsDeleteNamespacesSubtree (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 */
523
524      if (AcpiGbl_DbOpt_disasm)
525      {
526          /* This is the real disassembly */
527
528          AdDisplayTables (Filename, Table);
529
530          /* Dump hex table if requested (-vt) */
531
532          AcpiDmDumpDataTable (Table);
533
534          fprintf (stderr, "Disassembly completed\n");
535          fprintf (stderr, "ASL Output:  %s - %u bytes\n",
536                  DisasmFilename, AdGetFileSize (File));
537      }
538  }
539
540  Cleanup:
541
542      if (Table && !AcpiUtIsAmlTable (Table))
543      {
544          ACPI_FREE (Table);
545      }
546
547      if (OutToFile && File)
548      {
549          if (AslCompilerdebug) /* Display final namespace, with transforms */
550          {
551              NsSetupNamespaceListing (File);
552              NsDisplayNamespace ();
553          }
554
555          fclose (File);
556          AcpiOsRedirectOutput (stdout);
557      }
558
559      AcpiPsDeleteParseTree (AcpiGbl_ParseOpRoot);
560      AcpiGbl_ParseOpRoot = NULL;
561      return (Status);
562  }
563
564  /******
565  *
566  * FUNCTION:      AdDisassemblerHeader
567  *
568  * PARAMETERS:    Filename          - Input file for the table
569  *
570  * RETURN:        None
571  *
572  * DESCRIPTION:   Create the disassembler header, including ACPI CA signon with
573  *                current time and date.
574  *
575  *
576  *****/
577
578  void
579  AdDisassemblerHeader (
580      char          *Filename)
581  {
582      time_t        Timer;
583
584      time (&Timer);
585
586      /* Header and input table info */

```

```

588      AcpiOsPrintf ("/*\n");
589      AcpiOsPrintf (ACPI_COMMON_HEADER ("AML Disassembler", " * "));
590
591      AcpiOsPrintf (" * Disassembly of %s, %s", Filename, ctime (&Timer));
592      AcpiOsPrintf (" *\n");
593  }
594
595  /******
596  *
597  * FUNCTION:      AdCreateTableHeader
598  *
599  * PARAMETERS:    Filename          - Input file for the table
600  *                Table            - Pointer to the raw table
601  *
602  * RETURN:        None
603  *
604  * DESCRIPTION:   Create the ASL table header, including ACPI CA signon with
605  *                current time and date.
606  *
607  *
608  *****/
609
610  static void
611  AdCreateTableHeader (
612      char          *Filename,
613      ACPI_TABLE_HEADER *Table)
614  {
615      char          *NewFilename;
616      UINT8         Checksum;
617
618      /*
619       * Print file header and dump original table header
620       */
621      AdDisassemblerHeader (Filename);
622
623      AcpiOsPrintf (" * Original Table Header:\n");
624      AcpiOsPrintf (" *      Signature      \"%4.4s\"\n", Table->Signature);
625      AcpiOsPrintf (" *      Length        0x%8.8X (%u)\n", Table->Length, Table->Length);
626
627      /* Print and validate the revision */
628
629      AcpiOsPrintf (" *      Revision      0x%2.2X", Table->Revision);
630
631      switch (Table->Revision)
632      {
633      case 0:
634
635          AcpiOsPrintf (" **** Invalid Revision");
636          break;
637
638      case 1:
639
640          /* Revision of DSDT controls the ACPI integer width */
641
642          if (ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_DSDT))
643          {
644              AcpiOsPrintf (" **** 32-bit table (V1), no 64-bit math support");
645          }
646          break;
647
648      default:
649
650          break;
651      }
652      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_RSMT))
787     {
788         PointerSize = sizeof (UINT32);
789     }
790     else
791     {
792         PointerSize = sizeof (UINT64);
793     }
794
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)) / PointerSize;
802     AcpiOsPrintf ("There are %u tables defined in the %4.4s\n",
803         NumTables, NewTable->Signature);
804
805     /* Get the FADT */
806
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");
815
816     /* Don't bother with FACS, it is usually all zeros */
817 }
818
819 /* Always get the DSDT */
820
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);
827
828     /* Store DSDT in the Table Manager */
829
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 }
843
844 #if 0
845 /* TBD: Future implementation */
846
847 AcpiOsPrintf ("\n");
848
849 /* Get all SSDTs */
850
851 ACPI_MOVE_32_TO_32 (TableHeader.Signature, ACPI_SIG_SSDT);

```

```

852     do
853     {
854         NewTable = NULL;
855         Status = AcpiOsTableOverride (&TableHeader, &NewTable);
856     } while (NewTable);
857 #endif
858
859     return (AE_OK);
860 }
861
862 /*****
863  * FUNCTION:      AdParseTable
864  * PARAMETERS:    Table          - Pointer to the raw table
865  *                OwnerId        - Returned OwnerId of the table
866  *                LoadTable     - If add table to the global table list
867  *                External       - If this is an external table
868  * RETURN:        Status
869  * DESCRIPTION:   Parse the DSDT.
870  */
871
872 ACPI_STATUS
873 AdParseTable (
874     ACPI_TABLE_HEADER *Table,
875     ACPI_OWNER_ID *OwnerId,
876     BOOLEAN LoadTable,
877     BOOLEAN External)
878 {
879     ACPI_STATUS Status = AE_OK;
880     ACPI_WALK_STATE *WalkState;
881     UINT8 *AmlStart;
882     UINT32 AmlLength;
883     UINT32 TableIndex;
884
885     if (!Table)
886     {
887         return (AE_NOT_EXIST);
888     }
889
890     /* Pass 1: Parse everything except control method bodies */
891
892     fprintf (stderr, "Pass 1 parse of [%4.4s]\n", (char *) Table->Signature);
893
894     AmlLength = Table->Length - sizeof (ACPI_TABLE_HEADER);
895     AmlStart = ((UINT8 *) Table + sizeof (ACPI_TABLE_HEADER));
896
897     /* Create the root object */
898
899     AcpiGbl_ParseOpRoot = AcpiPsCreateScopeOp ();
900     if (!AcpiGbl_ParseOpRoot)
901     {
902         return (AE_NO_MEMORY);
903     }
904
905     /* Create and initialize a new walk state */
906
907     WalkState = AcpiDsCreateWalkState (0,
908         AcpiGbl_ParseOpRoot, NULL, NULL);
909     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 }

```

new/usr/src/common/acpica/common/adwalk.c

1

```
*****
29204 Fri Jan 17 07:29:29 2014
new/usr/src/common/acpica/common/adwalk.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: adwalk - Application-level disassembler parse tree walk routines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"
48 #include "amlcode.h"
49 #include "acdisasm.h"
50 #include "acdispat.h"
51 #include "acnamesp.h"
52 #include "acapps.h"

55 #define _COMPONENT          ACPI_TOOLS
56         ACPI_MODULE_NAME    ("adwalk")
```

new/usr/src/common/acpica/common/adwalk.c

2

```
58 /*
59 * aslmap - opcode mappings and reserved method names
60 */
61 ACPI_OBJECT_TYPE
62 AslMapNamedOpcodeToDataType (
63     UINT16                                     Opcode);

65 /* Local prototypes */

67 static ACPI_STATUS
68 AcpiDmFindOrphanDescending (
69     ACPI_PARSE_OBJECT *Op,
70     UINT32             Level,
71     void               *Context);

73 static ACPI_STATUS
74 AcpiDmDumpDescending (
75     ACPI_PARSE_OBJECT *Op,
76     UINT32             Level,
77     void               *Context);

79 static ACPI_STATUS
80 AcpiDmXrefDescendingOp (
81     ACPI_PARSE_OBJECT *Op,
82     UINT32             Level,
83     void               *Context);

85 static ACPI_STATUS
86 AcpiDmCommonAscendingOp (
87     ACPI_PARSE_OBJECT *Op,
88     UINT32             Level,
89     void               *Context);

91 static ACPI_STATUS
92 AcpiDmLoadDescendingOp (
93     ACPI_PARSE_OBJECT *Op,
94     UINT32             Level,
95     void               *Context);

97 static UINT32
98 AcpiDmInspectPossibleArgs (
99     UINT32             CurrentOpArgCount,
100     UINT32             TargetCount,
101     ACPI_PARSE_OBJECT *Op);

103 static ACPI_STATUS
104 AcpiDmResourceDescendingOp (
105     ACPI_PARSE_OBJECT *Op,
106     UINT32             Level,
107     void               *Context);

110 /*****
111 *
112 * FUNCTION:      AcpiDmDumpTree
113 *
114 * PARAMETERS:    Origin - Starting object
115 *
116 * RETURN:        None
117 *
118 * DESCRIPTION:   Parse tree walk to format and output the nodes
119 *
120 *****/

122 void
123 AcpiDmDumpTree (
```

```

124     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     }

522     ArgCount = AcpiDmInspectPossibleArgs (2, 1, NextOp);
523     if (ArgCount <= 1)
524     {
525         /* One Arg means this is just a Store(Name,Target) */

527         AcpiDmAddOpToExternalList (ChildOp,
528             ChildOp->Common.Value.String, ACPI_TYPE_INTEGER, 0, 0);
529         return (AE_OK);
530     }

532     AcpiDmAddOpToExternalList (ChildOp,
533         ChildOp->Common.Value.String, ACPI_TYPE_METHOD, ArgCount, 0);
534     }
535     break;

537     case AML_INT_NAMEPATH_OP:

539         /* Must examine parent to see if this namepath is an argument */

541         ParentOp = Op->Common.Parent;
542         OpInfo = AcpiPsGetOpcodeInfo (ParentOp->Common.AmlOpcode);

544         if ((OpInfo->Class != AML_CLASS_EXECUTE) &&
545             (OpInfo->Class != AML_CLASS_CREATE) &&
546             (OpInfo->ObjectType != ACPI_TYPE_LOCAL_ALIAS) &&
547             (ParentOp->Common.AmlOpcode != AML_INT_METHODCALL_OP) &&
548             !Op->Common.Node)
549         {
550             ArgCount = AcpiDmInspectPossibleArgs (0, 0, Op->Common.Next);

552             /*
553              * Check if namepath is a predicate for if/while or lone parameter t
554              * a return.
555              */
556             if (ArgCount == 0)
557             {
558                 if (((ParentOp->Common.AmlOpcode == AML_IF_OP) ||
559                     (ParentOp->Common.AmlOpcode == AML_WHILE_OP) ||
560                     (ParentOp->Common.AmlOpcode == AML_RETURN_OP)) &&

562                     /* And namepath is the first argument */
563                     (ParentOp->Common.Value.Arg == Op))
564                 {
565                     AcpiDmAddOpToExternalList (Op,
566                         Op->Common.Value.String, ACPI_TYPE_INTEGER, 0, 0);
567                     break;
568                 }
569             }

571             /*
572              * This is a standalone namestring (not a parameter to another
573              * operator) - it *must* be a method invocation, nothing else is
574              * grammatically possible.
575              */
576             AcpiDmAddOpToExternalList (Op,
577                 Op->Common.Value.String, ACPI_TYPE_METHOD, ArgCount, 0);
578         }
579         break;

581     default:

583         break;
584 }

```

```

586     return (AE_OK);
587 }

590 /*****
591  *
592  * FUNCTION:      AcpiDmLoadDescendingOp
593  *
594  * PARAMETERS:    ASL_WALK_CALLBACK
595  *
596  * RETURN:        Status
597  *
598  * DESCRIPTION:    Descending handler for namespace control method object load
599  *
600  *****/

602 static ACPI_STATUS
603 AcpiDmLoadDescendingOp (
604     ACPI_PARSE_OBJECT *Op,
605     UINT32 Level,
606     void *Context)
607 {
608     ACPI_OP_WALK_INFO *Info = Context;
609     const ACPI_OPCODE_INFO *OpInfo;
610     ACPI_WALK_STATE *WalkState;
611     ACPI_OBJECT_TYPE ObjectType;
612     ACPI_STATUS Status;
613     char *Path = NULL;
614     ACPI_PARSE_OBJECT *NextOp;
615     ACPI_NAMESPACE_NODE *Node;
616     char FieldPath[5];
617     BOOLEAN PreDefined = FALSE;
618     UINT8 PreDefineIndex = 0;

621     WalkState = Info->WalkState;
622     OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
623     ObjectType = OpInfo->ObjectType;
624     ObjectType = AslMapNamedOpcodeToDataType (Op->Asl.AmlOpcode);

626     /* Only interested in operators that create new names */

628     if (!(OpInfo->Flags & AML_NAMED) &&
629         !(OpInfo->Flags & AML_CREATE))
630     {
631         goto Exit;
632     }

634     /* Get the NamePath from the appropriate place */

636     if (OpInfo->Flags & AML_NAMED)
637     {
638         /* For all named operators, get the new name */

640         Path = (char *) Op->Named.Path;

642         if (!Path && Op->Common.AmlOpcode == AML_INT_NAMEDFIELD_OP)
643         {
644             *ACPI_CAST_PTR (UINT32, &FieldPath[0]) = Op->Named.Name;
645             FieldPath[4] = 0;
646             Path = FieldPath;
647         }
648     }
649     else if (OpInfo->Flags & AML_CREATE)
650     {
651         /* 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 */
795
796     ObjectType = ACPI_TYPE_BUFFER; /* Change from TYPE_BUFFER_FIELD */
797
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 }
808
809 if (!Path)
810 {
811     goto Exit;
812 }
813
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 */
828
829     Status = AE_NOT_FOUND;
830 }
831
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 }
859
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;
870
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     }
880
881     Pathname = AcpiNsGetExternalPathname (Node);
882     if (!Pathname)
883     {
884         return (AE_NO_MEMORY);
885     }
886
887     AcpiDmAddNodeToExternalList (Node, (UINT8) ObjectType2,
888         ParamCount, ACPI_EXT_RESOLVED_REFERENCE);
889
890     ACPI_FREE (Pathname);
891     Op->Common.Node = Node;
892 }
893 else
894 {
895     Op->Common.Node = Node;
896 }
897
898 Exit:
899 /* Open new scope if necessary */
900
901 if (AcpiNsOpensScope (ObjectType))
902 {
903     if (Op->Common.Node)
904     {
905         Status = AcpiDsScopeStackPush (Op->Common.Node, ObjectType, WalkStat
906             if (ACPI_FAILURE (Status))
907             {
908                 return (Status);
909             }
910     }
911 }
912
913 return (AE_OK);
914 }

```

```

918 /*****
919 *
920 * FUNCTION:      AcpiDmResourceDescendingOp
921 *
922 * PARAMETERS:    ASL_WALK_CALLBACK
923 *
924 * RETURN:        None
925 *
926 * DESCRIPTION:   Process one parse op during symbolic resource index conversion.
927 *
928 *****/

930 static ACPI_STATUS
931 AcpiDmResourceDescendingOp (
932     ACPI_PARSE_OBJECT *Op,
933     UINT32 Level,
934     void *Context)
935 {
936     ACPI_OP_WALK_INFO *Info = Context;
937     const ACPI_OPCODE_INFO *OpInfo;
938     ACPI_WALK_STATE *WalkState;
939     ACPI_OBJECT_TYPE ObjectType;
940     ACPI_STATUS Status;

943     WalkState = Info->WalkState;
944     OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);

946     /* Open new scope if necessary */

948     ObjectType = OpInfo->ObjectType;
949     if (AcpiNsOpensScope (ObjectType))
950     {
951         if (Op->Common.Node)
952         {
954             Status = AcpiDsScopeStackPush (Op->Common.Node, ObjectType, WalkStat
955             if (ACPI_FAILURE (Status))
956             {
957                 return (Status);
958             }
959         }
960     }

962     /*
963     * Check if this operator contains a reference to a resource descriptor.
964     * If so, convert the reference into a symbolic reference.
965     */
966     AcpiDmCheckResourceReference (Op, WalkState);
967     return (AE_OK);
968 }

971 /*****
972 *
973 * FUNCTION:      AcpiDmCommonAscendingOp
974 *
975 * PARAMETERS:    ASL_WALK_CALLBACK
976 *
977 * RETURN:        None
978 *
979 * DESCRIPTION:   Ascending handler for combined parse/namespace walks. Closes
980 *               scope if necessary.
981 */

```

```

982 *****/

984 static ACPI_STATUS
985 AcpiDmCommonAscendingOp (
986     ACPI_PARSE_OBJECT *Op,
987     UINT32 Level,
988     void *Context)
989 {
990     ACPI_OP_WALK_INFO *Info = Context;
991     const ACPI_OPCODE_INFO *OpInfo;
992     ACPI_OBJECT_TYPE ObjectType;

995     /* Close scope if necessary */

997     OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
998     ObjectType = OpInfo->ObjectType;
999     ObjectType = AslMapNamedOpcodeToDataType (Op->Asl.AmlOpcode);

1001     if (AcpiNsOpensScope (ObjectType))
1002     {
1003         (void) AcpiDsScopeStackPop (Info->WalkState);
1004     }

1006     return (AE_OK);
1007 }

1010 /*****
1011 *
1012 * FUNCTION:      AcpiDmInspectPossibleArgs
1013 *
1014 * PARAMETERS:    CurrentOpArgCount - Which arg of the current op was the
1015 *               TargetCount        - possible method invocation found
1016 *               Op                  - Number of targets (0,1,2) for this op
1017 *               Parse op
1018 *
1019 * RETURN:        Status
1020 *
1021 * DESCRIPTION:   Examine following args and next ops for possible arguments
1022 *               for an unrecognized method invocation.
1023 *
1024 *****/

1026 static UINT32
1027 AcpiDmInspectPossibleArgs (
1028     UINT32 CurrentOpArgCount,
1029     UINT32 TargetCount,
1030     ACPI_PARSE_OBJECT *Op)
1031 {
1032     const ACPI_OPCODE_INFO *OpInfo;
1033     UINT32 i;
1034     UINT32 Last = 0;
1035     UINT32 Lookahead;

1038     Lookahead = (ACPI_METHOD_NUM_ARGS + TargetCount) - CurrentOpArgCount;

1040     /* Lookahead for the maximum number of possible arguments */

1042     for (i = 0; i < Lookahead; i++)
1043     {
1044         if (!Op)
1045         {
1046             break;
1047         }

```

```
1049     OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
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     }
1060     if ((OpInfo->Class != AML_CLASS_EXECUTE) &&
1061         (OpInfo->Class != AML_CLASS_CONTROL))
1062     {
1063         Last = i+1;
1064     }
1066     Op = Op->Common.Next;
1067 }
1069 return (Last);
1070 }
```

new/usr/src/common/acpica/common/ahpredef.c

1

```
*****
28510 Fri Jan 17 07:29:29 2014
new/usr/src/common/acpica/common/ahpredef.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: ahpredef - Table of all known ACPI predefined names
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "acpi.h"
45 #include "accommon.h"
46
47 /*
48 * iASL only needs a partial table (short descriptions only).
49 * AcpiHelp needs the full table.
50 */
51 #ifdef ACPI_ASL_COMPILER
52 #define AH_PREDEF(Name, ShortDesc, LongDesc) {Name, ShortDesc}
53 #else
54 #define AH_PREDEF(Name, ShortDesc, LongDesc) {Name, ShortDesc, LongDesc}
55 #endif
56
57 /*
58 * Predefined ACPI names, with short description and return value.
59 * This table was extracted directly from the ACPI specification.
60 */
```

new/usr/src/common/acpica/common/ahpredef.c

2

```
61 const AH_PREDEFINED_NAME AslPredefinedInfo[] =
62 {
63     AH_PREDEF ("_ACX", "Active Cooling", "Returns the active cooling policy t
64     AH_PREDEF ("_ADR", "Address", "Returns address of a device on parent bus,
65     AH_PREDEF ("_AET", "ACPI Event Interrupts", "Returns a list of GPIO event
66     AH_PREDEF ("_ALC", "Ambient Light Chromaticity", "Returns the ambient lig
67     AH_PREDEF ("_ALI", "Ambient Light Illuminance", "Returns the ambient ligh
68     AH_PREDEF ("_ALN", "Alignment", "Base alignment, Resource Descriptor fiel
69     AH_PREDEF ("_ALP", "Ambient Light Polling", "Returns the ambient light se
70     AH_PREDEF ("_ALR", "Ambient Light Response", "Returns the ambient light b
71     AH_PREDEF ("_ALT", "Ambient Light Temperature", "Returns the ambient ligh
72     AH_PREDEF ("_ALX", "Active List", "Returns a list of active cooling devic
73     AH_PREDEF ("_ART", "Active Cooling Relationship Table", "Returns thermal
74     AH_PREDEF ("_ASI", "Address Space Id", "Resource Descriptor field"),
75     AH_PREDEF ("_ASZ", "Access Size", "Resource Descriptor field"),
76     AH_PREDEF ("_ATT", "Type-Specific Attribute", "Resource Descriptor field"
77     AH_PREDEF ("_BAS", "Base Address", "Range base address, Resource Descript
78     AH_PREDEF ("_BBN", "BIOS Bus Number", "Returns the PCI bus number returne
79     AH_PREDEF ("_BCL", "Brightness Control Levels", "Returns a list of suppor
80     AH_PREDEF ("_BCM", "Brightness Control Method", "Sets the brightness leve
81     AH_PREDEF ("_BCT", "Battery Charge Time", "Returns time remaining to comp
82     AH_PREDEF ("_BDN", "BIOS Dock Name", "Returns the Dock ID returned by the
83     AH_PREDEF ("_BFS", "Back From Sleep", "Inform AML of a wake event"),
84     AH_PREDEF ("_BIF", "Battery Information", "Returns a Control Method Batte
85     AH_PREDEF ("_BIX", "Battery Information Extended", "Returns a Control Met
86     AH_PREDEF ("_BLT", "Battery Level Threshold", "Set battery level threshol
87     AH_PREDEF ("_BM", "Bus Master", "Resource Descriptor field"),
88     AH_PREDEF ("_BMA", "Battery Measurement Averaging Interval", "Sets batter
89     AH_PREDEF ("_BMC", "Battery Maintenance Control", "Sets battery maintenanc
90     AH_PREDEF ("_BMD", "Battery Maintenance Data", "Returns battery maintenanc
91     AH_PREDEF ("_BMS", "Battery Measurement Sampling Time", "Sets the battery
92     AH_PREDEF ("_BQC", "Brightness Query Current", "Returns the current displ
93     AH_PREDEF ("_BST", "Battery Status", "Returns a Control Method Battery st
94     AH_PREDEF ("_BTM", "Battery Time", "Returns the battery runtime"),
95     AH_PREDEF ("_BTP", "Battery Trip Point", "Sets a Control Method Battery t
96     AH_PREDEF ("_CBA", "Configuration Base Address", "Sets the base address f
97     AH_PREDEF ("_CDM", "Clock Domain", "Returns a logical processor's clock d
98     AH_PREDEF ("_CID", "Compatible ID", "Returns a device's Plug and Play Com
99     AH_PREDEF ("_CLS", "Class Code", "Returns PCI class code and subclass"),
100    AH_PREDEF ("_CPC", "Continuous Performance Control", "Returns a list of p
101    AH_PREDEF ("_CRS", "Current Resource Settings", "Returns the current reso
102    AH_PREDEF ("_CRT", "Critical Temperature", "Returns the shutdown critical
103    AH_PREDEF ("_CSD", "C-State Dependencies", "Returns a list of C-state dep
104    AH_PREDEF ("_CST", "C-States", "Returns a list of supported C-states"),
105    AH_PREDEF ("_CWS", "Clear Wake Alarm Status", "Clear the status of wake a
106    AH_PREDEF ("_DBT", "Debounce Timeout", "Timeout value, Resource Descripto
107    AH_PREDEF ("_DCK", "Dock Present", "Sets docking isolation. Presence indi
108    AH_PREDEF ("_DCS", "Display Current Status", "Returns status of the displ
109    AH_PREDEF ("_DDC", "Display Data Current", "Returns the EDID for the disp
110    AH_PREDEF ("_DDN", "DOS Device Name", "Returns a device logical name"),
111    AH_PREDEF ("_DEC", "Decode", "Device decoding type, Resource Descriptor f
112    AH_PREDEF ("_DEP", "Dependencies", "Returns a list of operation region de
113    AH_PREDEF ("_DGS", "Display Graphics State", "Return the current state of
114    AH_PREDEF ("_DIS", "Disable Device", "Disables a device"),
115    AH_PREDEF ("_DLM", "Device Lock Mutex", "Defines mutex for OS/AML sharing
116    AH_PREDEF ("_DMA", "Direct Memory Access", "Returns device current resour
117    AH_PREDEF ("_DOD", "Display Output Devices", "Enumerate all devices attac
118    AH_PREDEF ("_DOS", "Disable Output Switching", "Sets the display output s
119    AH_PREDEF ("_DPL", "Device Selection Polarity", "Polarity of Device Selec
120    AH_PREDEF ("_DRS", "Drive Strength", "Drive Strength setting for GPIO con
121    AH_PREDEF ("_DSM", "Device-Specific Method", "Executes device-specific fu
122    AH_PREDEF ("_DSS", "Device Set State", "Sets the display device state"),
123    AH_PREDEF ("_DSW", "Device Sleep Wake", "Sets the sleep and wake transiti
124    AH_PREDEF ("_DTI", "Device Temperature Indication", "Conveys native devic
125    AH_PREDEF ("_Exx", "Edge-Triggered GPE", "Method executed as a result of
126    AH_PREDEF ("_EC", "Embedded Controller", "returns EC offset and query in
```



```

127 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 ("_GAT", "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 ("_GTM", "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 ("_HPM", "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 ("_OST", "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 Capabilities", "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 ("_PRO", "Power Resources for D0", "Returns a list of dependent
211 AH_PREDEF ("_PRI", "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 };

```

new/usr/src/common/acpica/common/dmextern.c

1

```
*****
38442 Fri Jan 17 07:29:29 2014
new/usr/src/common/acpica/common/dmextern.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dmextern - Support for External() ASL statements
4 *
5 *****/
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "amlcode.h"
47 #include "acnamesp.h"
48 #include "acdisasm.h"
49 #include "aslcompiler.h"
50 #include <stdio.h>
51 #include <errno.h>
54 /*
55 * This module is used for application-level code (iASL disassembler) only.
56 *
57 * It contains the code to create and emit any necessary External() ASL
```

new/usr/src/common/acpica/common/dmextern.c

2

```
58 * statements for the module being disassembled.
59 */
60 #define _COMPONENT          ACPI_CA_DISASSEMBLER
61       ACPI_MODULE_NAME    ("dmextern")
64 /*
65 * This table maps ACPI_OBJECT_TYPES to the corresponding ASL
66 * ObjectTypeKeyword. Used to generate typed external declarations
67 */
68 static const char          *AcpiGbl_DmTypeNames[] =
69 {
70     /* 00 */ "",                               /* Type ANY */
71     /* 01 */ "IntObj",
72     /* 02 */ "StrObj",
73     /* 03 */ "BuffObj",
74     /* 04 */ "PkgObj",
75     /* 05 */ "FieldUnitObj",
76     /* 06 */ "DeviceObj",
77     /* 07 */ "EventObj",
78     /* 08 */ "MethodObj",
79     /* 09 */ "MutexObj",
80     /* 10 */ "OpRegionObj",
81     /* 11 */ "PowerResObj",
82     /* 12 */ "ProcessorObj",
83     /* 13 */ "ThermalZoneObj",
84     /* 14 */ "BuffFieldObj",
85     /* 15 */ "DDBHandleObj",
86     /* 16 */ "",                               /* Debug object */
87     /* 17 */ "FieldUnitObj",
88     /* 18 */ "FieldUnitObj",
89     /* 19 */ "FieldUnitObj"
90 };
92 #define METHOD_SEPARATORS    " \\t,()\\n"
95 /* Local prototypes */
97 static const char *
98 AcpiDmGetObjectTypeName (
99     ACPI_OBJECT_TYPE      Type);
101 static char *
102 AcpiDmNormalizeParentPrefix (
103     ACPI_PARSE_OBJECT     *Op,
104     char                  *Path);
106 static void
107 AcpiDmAddPathToExternalList (
108     char                  *Path,
109     UINT8                 Type,
110     UINT32                 Value,
111     UINT16                 Flags);
113 static ACPI_STATUS
114 AcpiDmCreateNewExternal (
115     char                  *ExternalPath,
116     char                  *InternalPath,
117     UINT8                 Type,
118     UINT32                 Value,
119     UINT16                 Flags);
122 /*****
123 *
```

```

124 * FUNCTION:      AcpiDmGetObjectTypeName
125 *
126 * PARAMETERS:    Type                - An ACPI_OBJECT_TYPE
127 *
128 * RETURN:        Pointer to a string
129 *
130 * DESCRIPTION:    Map an object type to the ASL object type string.
131 *
132 *****/
133
134 static const char *
135 AcpiDmGetObjectTypeName (
136     ACPI_OBJECT_TYPE    Type)
137 {
138
139     if (Type == ACPI_TYPE_LOCAL_SCOPE)
140     {
141         Type = ACPI_TYPE_DEVICE;
142     }
143
144     else if (Type > ACPI_TYPE_LOCAL_INDEX_FIELD)
145     {
146         return ("");
147     }
148
149     return (AcpiGbl_DmTypeNames[Type]);
150 }
151
152
153 /*****
154 *
155 * FUNCTION:      AcpiDmNormalizeParentPrefix
156 *
157 * PARAMETERS:    Op                - Parse op
158 *                Path              - Path with parent prefix
159 *
160 * RETURN:        The full pathname to the object (from the namespace root)
161 *
162 * DESCRIPTION:    Returns the full pathname of a path with parent prefix
163 *                The caller must free the fullpath returned.
164 *
165 *****/
166
167 static char *
168 AcpiDmNormalizeParentPrefix (
169     ACPI_PARSE_OBJECT    *Op,
170     char                 *Path)
171 {
172     ACPI_NAMESPACE_NODE    *Node;
173     char                   *Fullpath;
174     char                   *ParentPath;
175     ACPI_SIZE              Length;
176     UINT32                 Index = 0;
177
178     if (!Op)
179     {
180         return (NULL);
181     }
182
183     /* Search upwards in the parse tree until we reach the next namespace node */
184
185     Op = Op->Common.Parent;
186     while (Op)
187     {
188         if (Op->Common.Node)

```

```

190     {
191         break;
192     }
193
194     Op = Op->Common.Parent;
195 }
196
197 if (!Op)
198 {
199     return (NULL);
200 }
201
202 /*
203 * Find the actual parent node for the reference:
204 * Remove all carat prefixes from the input path.
205 * There may be multiple parent prefixes (For example, ^^M000)
206 */
207 Node = Op->Common.Node;
208 while (Node && (*Path == (UINT8) AML_PARENT_PREFIX))
209 {
210     Node = Node->Parent;
211     Path++;
212 }
213
214 if (!Node)
215 {
216     return (NULL);
217 }
218
219 /* Get the full pathname for the parent node */
220
221 ParentPath = AcpiNsGetExternalPathname (Node);
222 if (!ParentPath)
223 {
224     return (NULL);
225 }
226
227 Length = (ACPI_STRLEN (ParentPath) + ACPI_STRLEN (Path) + 1);
228 if (ParentPath[1])
229 {
230     /*
231     * If ParentPath is not just a simple '\', increment the length
232     * for the required dot separator (ParentPath.Path)
233     */
234     Length++;
235
236     /* For External() statements, we do not want a leading '\' */
237
238     if (*ParentPath == AML_ROOT_PREFIX)
239     {
240         Index = 1;
241     }
242 }
243
244 Fullpath = ACPI_ALLOCATE_ZEROED (Length);
245 if (!Fullpath)
246 {
247     goto Cleanup;
248 }
249
250 /*
251 * Concatenate parent fullpath and path. For example,
252 * parent fullpath "\_SB_", Path "^INIT", Fullpath "\_SB_.INIT"
253 *
254 * Copy the parent path
255 */

```

```

256     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     }
392
393     /* Open the file */
394
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     }
402
403     /* Each line defines a method */
404
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;
410
411         MethodName = strtok (NULL, METHOD_SEPARATORS); /* Method namepath */
412         if (!MethodName) continue;
413
414         Token = strtok (NULL, METHOD_SEPARATORS); /* "MethodObj" */
415         if (!Token) continue;
416         if (strcmp (Token, "MethodObj")) continue;
417
418         Token = strtok (NULL, METHOD_SEPARATORS); /* Arg count */
419         if (!Token) continue;
420
421         /* Convert arg count string to an integer */
422
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         }
435
436         /* Add this external to the global list */
437
438         AcpiOsPrintf ("%s: Importing method external (%u arguments) %s\\n",
439                     Gbl_ExternalRefFilename, ArgCount, MethodName);
440
441         AcpiDmAddPathToExternalList (MethodName, ACPI_TYPE_METHOD,
442                                     ArgCount, (ACPI_EXT_RESOLVED_REFERENCE | ACPI_EXT_ORIGIN_FROM_FILE))
443         ImportCount++;
444     }
445
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 ();
456
457     AcpiOsPrintf ("%s: Imported %u external method definitions\\n",
458                 Gbl_ExternalRefFilename, ImportCount);
459 }
460
461     fclose (ExternalRefFile);
462 }
463
464
465 /*****
466  * FUNCTION:      AcpiDmAddOpToExternalList
467  *
468  * PARAMETERS:    Op                - Current parser Op
469  *                Path              - Internal (AML) path to the object
470  *                Type              - ACPI object type to be added
471  *                Value             - Arg count if adding a Method object
472  *                Flags             - To be passed to the external object
473  *
474  * RETURN:        None
475  *
476  * DESCRIPTION:   Insert a new name into the global list of Externals which
477  *                will in turn be later emitted as an External() declaration
478  *                in the disassembled output.
479  *
480  *                This function handles the most common case where the referenced
481  *                name is simply not found in the constructed namespace.
482  *
483  *****/
484
485 void
486 AcpiDmAddOpToExternalList (
487     ACPI_PARSE_OBJECT *Op,
488     char *Path,
489     UINT8 Type,
490     UINT32 Value,
491     UINT16 Flags)
492 {
493     char *ExternalPath;
494     char *InternalPath = Path;
495     char *Temp;
496     ACPI_STATUS Status;
497
498     /*
499      * ACPI_FUNCTION_TRACE (DmAddOpToExternalList);
500      */
501
502     if (!Path)
503     {
504         return_VOID;
505     }
506
507     /* Remove a root backslash if present */
508
509     if ((*Path == AML_ROOT_PREFIX) && (Path[1]))
510     {
511         Path++;
512     }
513
514     /* Externalize the pathname */
515
516     Status = AcpiNsExternalizeName (ACPI_UINT32_MAX, Path,
517                                   NULL, &ExternalPath);
518     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);

```



```

884  /* Check all existing externals to ensure no duplicates */
886  NextExternal = AcpiGbl_ExternalList;
887  while (NextExternal)
888  {
889      if (!ACPI_STRCMP (ExternalPath, NextExternal->Path))
890      {
891          /* Duplicate method, check that the Value (ArgCount) is the same */
893          if ((NextExternal->Type == ACPI_TYPE_METHOD) &&
894              (NextExternal->Value != Value))
895          {
896              ACPI_ERROR ((AE_INFO,
897                  "External method arg count mismatch %s: Current %u, attempted %u",
898                  NextExternal->Path, NextExternal->Value, Value));
899          }
901          /* Allow upgrade of type from ANY */
903          else if (NextExternal->Type == ACPI_TYPE_ANY)
904          {
905              NextExternal->Type = Type;
906              NextExternal->Value = Value;
907          }
909          return_ACPI_STATUS (AE_ALREADY_EXISTS);
910      }
912      NextExternal = NextExternal->Next;
913  }
915  /* Allocate and init a new External() descriptor */
917  NewExternal = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_EXTERNAL_LIST));
918  if (!NewExternal)
919  {
920      return_ACPI_STATUS (AE_NO_MEMORY);
921  }
923  ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
924      "Adding external reference node (%s) type [%s]\n",
925      ExternalPath, AcpiUtGetTypeName (Type)));
927  NewExternal->Flags = Flags;
928  NewExternal->Value = Value;
929  NewExternal->Path = ExternalPath;
930  NewExternal->Type = Type;
931  NewExternal->Length = (UINT16) ACPI_STRLEN (ExternalPath);
932  NewExternal->InternalPath = InternalPath;
934  /* Link the new descriptor into the global list, alphabetically ordered */
936  NextExternal = AcpiGbl_ExternalList;
937  while (NextExternal)
938  {
939      if (AcpiUtStricmp (NewExternal->Path, NextExternal->Path) < 0)
940      {
941          if (PrevExternal)
942          {
943              PrevExternal->Next = NewExternal;
944          }
945          else
946          {
947              AcpiGbl_ExternalList = NewExternal;
948          }
949      }
950  }

```

```

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);
917     ObjDesc->Method.ParamCount = (UINT8) External->Value;
918     Node->Object = ObjDesc;
919     break;

921     case ACPI_TYPE_REGION:

923         /* Regions require a region sub-object */

925         ObjDesc = AcpiUtCreateInternalObject (ACPI_TYPE_REGION);
926         ObjDesc->Region.Node = Node;
927         Node->Object = ObjDesc;
928         break;

930     default:

932         break;
933     }

935     External = External->Next;
936 }
937 }

940 /*****
941 *
942 * FUNCTION:      AcpiDmGetExternalMethodCount
943 *
944 * PARAMETERS:    None
945 *
946 * RETURN:        The number of control method externals in the external list
947 *
948 * DESCRIPTION:   Return the number of method externals that have been generated.
949 *               If any control method externals have been found, we must
950 *               re-parse the entire definition block with the new information
951 *               (number of arguments for the methods.) This is limitation of
952 *               AML, we don't know the number of arguments from the control
953 *               method invocation itself.
954 *
955 *****/

957 UINT32
958 AcpiDmGetExternalMethodCount (
959     void)
960 {
961     ACPI_EXTERNAL_LIST    *External = AcpiGbl_ExternalList;
962     UINT32                 Count = 0;

965     while (External)
966     {
967         if (External->Type == ACPI_TYPE_METHOD)
968         {
969             Count++;
970         }

972         External = External->Next;
973     }

975     return (Count);
976 }

979 /*****
980 *
981 * FUNCTION:      AcpiDmClearExternalList

```

```

982 *
983 * PARAMETERS:    None
984 *
985 * RETURN:        None
986 *
987 * DESCRIPTION:   Free the entire External info list
988 *
989 *****/

991 void
992 AcpiDmClearExternalList (
993     void)
994 {
995     ACPI_EXTERNAL_LIST    *NextExternal;

998     while (AcpiGbl_ExternalList)
999     {
1000         NextExternal = AcpiGbl_ExternalList->Next;
1001         ACPI_FREE (AcpiGbl_ExternalList->Path);
1002         ACPI_FREE (AcpiGbl_ExternalList);
1003         AcpiGbl_ExternalList = NextExternal;
1004     }
1005 }

1008 /*****
1009 *
1010 * FUNCTION:      AcpiDmEmitExternals
1011 *
1012 * PARAMETERS:    None
1013 *
1014 * RETURN:        None
1015 *
1016 * DESCRIPTION:   Emit an External() ASL statement for each of the externals in
1017 *               the global external info list.
1018 *
1019 *****/

1021 void
1022 AcpiDmEmitExternals (
1023     void)
1024 {
1025     ACPI_EXTERNAL_LIST    *NextExternal;

1028     if (!AcpiGbl_ExternalList)
1029     {
1030         return;
1031     }

1033     /*
1034     * Determine the number of control methods in the external list, and
1035     * also how many of those externals were resolved via the namespace.
1036     */
1037     NextExternal = AcpiGbl_ExternalList;
1038     while (NextExternal)
1039     {
1040         if (NextExternal->Type == ACPI_TYPE_METHOD)
1041         {
1042             AcpiGbl_NumExternalMethods++;
1043             if (NextExternal->Flags & ACPI_EXT_RESOLVED_REFERENCE)
1044             {
1045                 AcpiGbl_ResolvedExternalMethods++;
1046             }
1047         }

```

```

1049     NextExternal = NextExternal->Next;
1050 }

1052 /* Check if any control methods were unresolved */

1054 AcpiDmUnresolvedWarning (1);

1056 /* Emit any unresolved method externals in a single text block */

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));

1068         AcpiOsPrintf (
1069             ")    // Warning: Unresolved Method, "
1070             "guessing %u arguments (may be incorrect, see warning above)\n",
1071             NextExternal->Value);

1073         NextExternal->Flags |= ACPI_EXT_EXTERNAL_EMITTED;
1074     }

1076     NextExternal = NextExternal->Next;
1077 }

1079 AcpiOsPrintf ("\n");

1082 /* Emit externals that were imported from a file */

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);

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));

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         }

1113         NextExternal = NextExternal->Next;

```

```

1114     }

1116     AcpiOsPrintf ("\n");
1117 }

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));

1130         /* For methods, add a comment with the number of arguments */

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     }

1143     /* Free this external info block and move on to next external */

1145     NextExternal = AcpiGbl_ExternalList->Next;
1146     if (AcpiGbl_ExternalList->Flags & ACPI_EXT_INTERNAL_PATH_ALLOCATED)
1147     {
1148         ACPI_FREE (AcpiGbl_ExternalList->InternalPath);
1149     }

1151     ACPI_FREE (AcpiGbl_ExternalList->Path);
1152     ACPI_FREE (AcpiGbl_ExternalList);
1153     AcpiGbl_ExternalList = NextExternal;
1154 }

1156 AcpiOsPrintf ("\n");
1157 }

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  *****/

1175 #if 0
1176 Summary of the external control method problem:

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).  
 1182 The disassembler tracks any such references, and will emit an External()  
 1183 statement for these types of methods, with the proper number of arguments .

1185 Without the SSDTs, the AML does not contain enough information to properly  
 1186 disassemble the control method invocation -- because the disassembler does  
 1187 not know how many arguments to parse.

1189 An example: Assume we have two control methods. ABCD has one argument, and  
 1190 EFGH has zero arguments. Further, we have two additional control methods  
 1191 that invoke ABCD and EFGH, named T1 and T2:

```
1193 Method (ABCD, 1)
1194 {
1195 }
1196 Method (EFGH, 0)
1197 {
1198 }
1199 Method (T1)
1200 {
1201     ABCD (Add (2, 7, Local0))
1202 }
1203 Method (T2)
1204 {
1205     EFGH ()
1206     Add (2, 7, Local0)
1207 }
```

1209 Here is the AML code that is generated for T1 and T2:

```
1211 185: Method (T1)
1213 0000034C: 14 10 54 31 5F 5F 00 ... "..T1__."
1215 186: {
1216 187:     ABCD (Add (2, 7, Local0))
1218 00000353: 41 42 43 44 ..... "ABCD"
1219 00000357: 72 0A 02 0A 07 60 ..... "r...."
1221 188: }
1223 190: Method (T2)
1225 0000035D: 14 10 54 32 5F 5F 00 ... "..T2__."
1227 191: {
1228 192:     EFGH ()
1230 00000364: 45 46 47 48 ..... "EFGH"
1232 193:     Add (2, 7, Local0)
1234 00000368: 72 0A 02 0A 07 60 ..... "r...."
1235 194: }
```

1237 Note that the AML code for T1 and T2 is essentially identical. When  
 1238 disassembling this code, the methods ABCD and EFGH must be known to the  
 1239 disassembler, otherwise it does not know how to handle the method invocations.

1241 In other words, if ABCD and EFGH are actually external control methods  
 1242 appearing in an SSDT, the disassembler does not know what to do unless  
 1243 the owning SSDT has been loaded via the -e option.  
 1244 #endif

```
1246 void
1247 AcpiDmUnresolvedWarning (
1248     UINT8                Type)
1249 {
1251     if (!AcpiGbl_NumExternalMethods)
1252     {
1253         return;
1254     }
1256     if (Type)
1257     {
1258         if (!AcpiGbl_ExternalFileList)
1259         {
1260             /* The -e option was not specified */
1262             AcpiOsPrintf (" /*\n"
1263                 " * iASL Warning: There were %u external control methods fou
1264                 " * disassembly, but additional ACPI tables to resolve these
1265                 " * were not specified. This resulting disassembler output f
1266                 " * compile because the disassembler did not know how many a
1267                 " * to assign to these methods. To specify the tables needed
1268                 " * external control method references, use the one of the f
1269                 " * example iASL invocations:\n"
1270                 " * iasl -e <ssdt1.aml,ssdt2.aml...> -d <dstdt.aml>\n"
1271                 " * iasl -e <dstdt.aml,ssdt2.aml...> -d <ssdt1.aml>\n"
1272                 " */\n",
1273                 AcpiGbl_NumExternalMethods);
1274         }
1275         else if (AcpiGbl_NumExternalMethods != AcpiGbl_ResolvedExternalMethods)
1276         {
1277             /* The -e option was specified, but there are still some unresolved
1279             AcpiOsPrintf (" /*\n"
1280                 " * iASL Warning: There were %u external control methods fou
1281                 " * disassembly, but only %u %s resolved (%u unresolved). Ad
1282                 " * ACPI tables are required to properly disassemble the cod
1283                 " * resulting disassembler output file may not compile becau
1284                 " * disassembler did not know how many arguments to assign t
1285                 " * unresolved methods.\n"
1286                 " */\n",
1287                 AcpiGbl_NumExternalMethods, AcpiGbl_ResolvedExternalMethods,
1288                 (AcpiGbl_ResolvedExternalMethods > 1 ? "were" : "was"),
1289                 (AcpiGbl_NumExternalMethods - AcpiGbl_ResolvedExternalMethods));
1290         }
1291     }
1292     else
1293     {
1294         if (!AcpiGbl_ExternalFileList)
1295         {
1296             /* The -e option was not specified */
1298             fprintf (stderr, "\n"
1299                 "iASL Warning: There were %u external control methods found duri
1300                 "disassembly, but additional ACPI tables to resolve these extern
1301                 "were not specified. The resulting disassembler output file may
1302                 "compile because the disassembler did not know how many argument
1303                 "to assign to these methods. To specify the tables needed to res
1304                 "external control method references, use the one of the followin
1305                 "example iASL invocations:\n"
1306                 " iasl -e <ssdt1.aml,ssdt2.aml...> -d <dstdt.aml>\n"
1307                 " iasl -e <dstdt.aml,ssdt2.aml...> -d <ssdt1.aml>\n",
1308                 AcpiGbl_NumExternalMethods);
1309         }
1310         else if (AcpiGbl_NumExternalMethods != AcpiGbl_ResolvedExternalMethods)
1311         {
```

```
1312      /* The -e option was specified, but there are still some unresolved
1313
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 }
```

new/usr/src/common/acpica/common/dmrestag.c

1

```
*****
33385 Fri Jan 17 07:29:30 2014
new/usr/src/common/acpica/common/dmrestag.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dmrestag - Add tags to resource descriptors (Application-level)
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"
48 #include "acdisasm.h"
49 #include "acnamesp.h"
50 #include "amlcode.h"

52 /* This module used for application-level code only */

54 #define _COMPONENT          ACPI_CA_DISASSEMBLER
55       ACPI_MODULE_NAME    ("dmrestag")

57 /* Local prototypes */

59 static void
```

new/usr/src/common/acpica/common/dmrestag.c

2

```
60 AcpiDmUpdateResourceName (
61     ACPI_NAMESPACE_NODE    *ResourceNode);

63 static char *
64 AcpiDmSearchTagList (
65     UINT32                  BitIndex,
66     const ACPI_RESOURCE_TAG *TagList);

68 static char *
69 AcpiDmGetResourceTag (
70     UINT32                  BitIndex,
71     AML_RESOURCE            *Resource,
72     UINT8                   ResourceIndex);

74 static char *
75 AcpiGetTagPathname (
76     ACPI_PARSE_OBJECT       *Op,
77     ACPI_NAMESPACE_NODE     *BufferNode,
78     ACPI_NAMESPACE_NODE     *ResourceNode,
79     UINT32                  BitIndex);

81 static ACPI_NAMESPACE_NODE *
82 AcpiDmGetResourceNode (
83     ACPI_NAMESPACE_NODE     *BufferNode,
84     UINT32                  BitIndex);

86 static ACPI_STATUS
87 AcpiDmAddResourceToNamespace (
88     UINT8                   *Aml,
89     UINT32                  Length,
90     UINT32                  Offset,
91     UINT8                   ResourceIndex,
92     void                    **Context);

94 static void
95 AcpiDmAddResourcesToNamespace (
96     ACPI_NAMESPACE_NODE     *BufferNode,
97     ACPI_PARSE_OBJECT       *Op);

100 /*****
101 *
102 * Resource Tag tables
103 *
104 * These are the predefined tags that refer to elements of a resource
105 * descriptor. Each name and offset is defined in the ACPI specification.
106 *
107 * Each table entry contains the bit offset of the field and the associated
108 * name.
109 *
110 *****/

112 static const ACPI_RESOURCE_TAG    AcpiDmIrqTags[] =
113 {
114     { ( 1 * 8),          ACPI_RESTAG_INTERRUPT},
115     { ( 3 * 8) + 0,      ACPI_RESTAG_INTERRUPTTYPE},
116     { ( 3 * 8) + 3,      ACPI_RESTAG_INTERRUPTLEVEL},
117     { ( 3 * 8) + 4,      ACPI_RESTAG_INTERRUPTSHARE},
118     { 0,                NULL},
119 };

121 static const ACPI_RESOURCE_TAG    AcpiDmDmaTags[] =
122 {
123     { ( 1 * 8),          ACPI_RESTAG_DMA},
124     { ( 2 * 8) + 0,      ACPI_RESTAG_XFERTYPE},
125     { ( 2 * 8) + 2,      ACPI_RESTAG_BUSMASTER},
```

```

126     {( 2 * 8) + 5,  ACPI_RESTAG_DMATYP},
127     {0,             NULL}
128 };

130 static const ACPI_RESOURCE_TAG    AcpiDmIoTags[] =
131 {
132     {( 1 * 8) + 0,  ACPI_RESTAG_DECODE},
133     {( 2 * 8),      ACPI_RESTAG_MINADDR},
134     {( 4 * 8),      ACPI_RESTAG_MAXADDR},
135     {( 6 * 8),      ACPI_RESTAG_ALIGNMENT},
136     {( 7 * 8),      ACPI_RESTAG_LENGTH},
137     {0,             NULL}
138 };

140 static const ACPI_RESOURCE_TAG    AcpiDmFixedIoTags[] =
141 {
142     {( 1 * 8),      ACPI_RESTAG_BASEADDRESS},
143     {( 3 * 8),      ACPI_RESTAG_LENGTH},
144     {0,             NULL}
145 };

147 static const ACPI_RESOURCE_TAG    AcpiDmFixedDmaTags[] =
148 {
149     {( 1 * 8),      ACPI_RESTAG_DMA},
150     {( 3 * 8),      ACPI_RESTAG_DMATYP},
151     {( 5 * 8),      ACPI_RESTAG_XFERTYP},
152     {0,             NULL}
153 };

155 static const ACPI_RESOURCE_TAG    AcpiDmMemory24Tags[] =
156 {
157     {( 3 * 8) + 0,  ACPI_RESTAG_READWRITETYP},
158     {( 4 * 8),      ACPI_RESTAG_MINADDR},
159     {( 6 * 8),      ACPI_RESTAG_MAXADDR},
160     {( 8 * 8),      ACPI_RESTAG_ALIGNMENT},
161     {(10 * 8),      ACPI_RESTAG_LENGTH},
162     {0,             NULL}
163 };

165 static const ACPI_RESOURCE_TAG    AcpiDmRegisterTags[] =
166 {
167     {( 3 * 8),      ACPI_RESTAG_ADDRESSSPACE},
168     {( 4 * 8),      ACPI_RESTAG_REGISTERBITWIDTH},
169     {( 5 * 8),      ACPI_RESTAG_REGISTERBITOFFSET},
170     {( 6 * 8),      ACPI_RESTAG_ACCESSSIZE},
171     {( 7 * 8),      ACPI_RESTAG_ADDRESS},
172     {0,             NULL}
173 };

175 static const ACPI_RESOURCE_TAG    AcpiDmMemory32Tags[] =
176 {
177     {( 3 * 8) + 0,  ACPI_RESTAG_READWRITETYP},
178     {( 4 * 8),      ACPI_RESTAG_MINADDR},
179     {( 8 * 8),      ACPI_RESTAG_MAXADDR},
180     {(12 * 8),      ACPI_RESTAG_ALIGNMENT},
181     {(16 * 8),      ACPI_RESTAG_LENGTH},
182     {0,             NULL}
183 };

185 static const ACPI_RESOURCE_TAG    AcpiDmFixedMemory32Tags[] =
186 {
187     {( 3 * 8) + 0,  ACPI_RESTAG_READWRITETYP},
188     {( 4 * 8),      ACPI_RESTAG_BASEADDRESS},
189     {( 8 * 8),      ACPI_RESTAG_LENGTH},
190     {0,             NULL}
191 };

```

```

193 static const ACPI_RESOURCE_TAG    AcpiDmInterruptTags[] =
194 {
195     {( 3 * 8) + 1,  ACPI_RESTAG_INTERRUPTTYP},
196     {( 3 * 8) + 2,  ACPI_RESTAG_INTERRUPTLEVEL},
197     {( 3 * 8) + 3,  ACPI_RESTAG_INTERRUPTSHARE},
198     {( 5 * 8),      ACPI_RESTAG_INTERRUPT},
199     {0,             NULL}
200 };

202 static const ACPI_RESOURCE_TAG    AcpiDmAddress16Tags[] =
203 {
204     {( 4 * 8) + 1,  ACPI_RESTAG_DECODE},
205     {( 4 * 8) + 2,  ACPI_RESTAG_MINTYP},
206     {( 4 * 8) + 3,  ACPI_RESTAG_MAXTYP},
207     {( 6 * 8),      ACPI_RESTAG_GRANULARITY},
208     {( 8 * 8),      ACPI_RESTAG_MINADDR},
209     {(10 * 8),      ACPI_RESTAG_MAXADDR},
210     {(12 * 8),      ACPI_RESTAG_TRANSLATION},
211     {(14 * 8),      ACPI_RESTAG_LENGTH},
212     {0,             NULL}
213 };

215 static const ACPI_RESOURCE_TAG    AcpiDmAddress32Tags[] =
216 {
217     {( 4 * 8) + 1,  ACPI_RESTAG_DECODE},
218     {( 4 * 8) + 2,  ACPI_RESTAG_MINTYP},
219     {( 4 * 8) + 3,  ACPI_RESTAG_MAXTYP},
220     {( 6 * 8),      ACPI_RESTAG_GRANULARITY},
221     {(10 * 8),      ACPI_RESTAG_MINADDR},
222     {(14 * 8),      ACPI_RESTAG_MAXADDR},
223     {(18 * 8),      ACPI_RESTAG_TRANSLATION},
224     {(22 * 8),      ACPI_RESTAG_LENGTH},
225     {0,             NULL}
226 };

228 static const ACPI_RESOURCE_TAG    AcpiDmAddress64Tags[] =
229 {
230     {( 4 * 8) + 1,  ACPI_RESTAG_DECODE},
231     {( 4 * 8) + 2,  ACPI_RESTAG_MINTYP},
232     {( 4 * 8) + 3,  ACPI_RESTAG_MAXTYP},
233     {( 6 * 8),      ACPI_RESTAG_GRANULARITY},
234     {(14 * 8),      ACPI_RESTAG_MINADDR},
235     {(22 * 8),      ACPI_RESTAG_MAXADDR},
236     {(30 * 8),      ACPI_RESTAG_TRANSLATION},
237     {(38 * 8),      ACPI_RESTAG_LENGTH},
238     {0,             NULL}
239 };

241 static const ACPI_RESOURCE_TAG    AcpiDmExtendedAddressTags[] =
242 {
243     {( 4 * 8) + 1,  ACPI_RESTAG_DECODE},
244     {( 4 * 8) + 2,  ACPI_RESTAG_MINTYP},
245     {( 4 * 8) + 3,  ACPI_RESTAG_MAXTYP},
246     {( 8 * 8),      ACPI_RESTAG_GRANULARITY},
247     {(16 * 8),      ACPI_RESTAG_MINADDR},
248     {(24 * 8),      ACPI_RESTAG_MAXADDR},
249     {(32 * 8),      ACPI_RESTAG_TRANSLATION},
250     {(40 * 8),      ACPI_RESTAG_LENGTH},
251     {(48 * 8),      ACPI_RESTAG_TYPESPECIFICATTRIBUTES},
252     {0,             NULL}
253 };

255 /* Subtype tables for GPIO descriptors */

257 static const ACPI_RESOURCE_TAG    AcpiDmGpioIntTags[] =

```

```

258 {
259     {( 7 * 8) + 0, ACPI_RESTAG_MODE},
260     {( 7 * 8) + 1, ACPI_RESTAG_POLARITY},
261     {( 7 * 8) + 3, ACPI_RESTAG_INTERRUPTSHARE},
262     {( 9 * 8), ACPI_RESTAG_PINCONFIG},
263     {(10 * 8), ACPI_RESTAG_DRIVESTRENGTH},
264     {(12 * 8), ACPI_RESTAG_DEBOUNCETIME},
265     {0, NULL},
266 };

268 static const ACPI_RESOURCE_TAG AcpiDmGpioIoTags[] =
269 {
270     {( 7 * 8) + 0, ACPI_RESTAG_IORESTRICTION},
271     {( 7 * 8) + 3, ACPI_RESTAG_INTERRUPTSHARE},
272     {( 9 * 8), ACPI_RESTAG_PINCONFIG},
273     {(10 * 8), ACPI_RESTAG_DRIVESTRENGTH},
274     {(12 * 8), ACPI_RESTAG_DEBOUNCETIME},
275     {0, NULL},
276 };

278 /* Subtype tables for SerialBus descriptors */

280 static const ACPI_RESOURCE_TAG AcpiDmI2cSerialBusTags[] =
281 {
282     {( 6 * 8) + 0, ACPI_RESTAG_SLAVEMODE},
283     {( 7 * 8) + 0, ACPI_RESTAG_MODE},
284     {(12 * 8), ACPI_RESTAG_SPEED},
285     {(16 * 8), ACPI_RESTAG_ADDRESS},
286     {0, NULL},
287 };

289 static const ACPI_RESOURCE_TAG AcpiDmSpiSerialBusTags[] =
290 {
291     {( 6 * 8) + 0, ACPI_RESTAG_SLAVEMODE},
292     {( 7 * 8) + 0, ACPI_RESTAG_MODE},
293     {( 7 * 8) + 1, ACPI_RESTAG_DEVICEPOLARITY},
294     {(12 * 8), ACPI_RESTAG_SPEED},
295     {(16 * 8), ACPI_RESTAG_LENGTH},
296     {(17 * 8), ACPI_RESTAG_PHASE},
297     {(18 * 8), ACPI_RESTAG_POLARITY},
298     {(19 * 8), ACPI_RESTAG_ADDRESS},
299     {0, NULL},
300 };

302 static const ACPI_RESOURCE_TAG AcpiDmUartSerialBusTags[] =
303 {
304     {( 6 * 8) + 0, ACPI_RESTAG_SLAVEMODE}, /* Note: not part of original macro
305     {( 7 * 8) + 0, ACPI_RESTAG_FLOWCONTROL},
306     {( 7 * 8) + 2, ACPI_RESTAG_STOPBITS},
307     {( 7 * 8) + 4, ACPI_RESTAG_LENGTH},
308     {( 7 * 8) + 7, ACPI_RESTAG_ENDIANNES},
309     {(12 * 8), ACPI_RESTAG_SPEED},
310     {(16 * 8), ACPI_RESTAG_LENGTH_RX},
311     {(18 * 8), ACPI_RESTAG_LENGTH_TX},
312     {(20 * 8), ACPI_RESTAG_PARITY},
313     {(21 * 8), ACPI_RESTAG_LINE},
314     {0, NULL},
315 };

317 /* Subtype tables for Address descriptor type-specific flags */

319 static const ACPI_RESOURCE_TAG AcpiDmMemoryFlagTags[] =
320 {
321     {( 5 * 8) + 0, ACPI_RESTAG_READWRITETYPE},
322     {( 5 * 8) + 1, ACPI_RESTAG_MEMTYPE},
323     {( 5 * 8) + 3, ACPI_RESTAG_MEMORYATTRIBUTES},

```

```

324     {( 5 * 8) + 5, ACPI_RESTAG_TYPE},
325     {0, NULL},
326 };

328 static const ACPI_RESOURCE_TAG AcpiDmIoFlagTags[] =
329 {
330     {( 5 * 8) + 0, ACPI_RESTAG_RANGETYPE},
331     {( 5 * 8) + 4, ACPI_RESTAG_TYPE},
332     {( 5 * 8) + 5, ACPI_RESTAG_TRANSTYPE},
333     {0, NULL},
334 };

337 /*
338  * Dispatch table used to obtain the correct tag table for a descriptor.
339  *
340  * A NULL in this table means one of three things:
341  * 1) The descriptor ID is reserved and invalid
342  * 2) The descriptor has no tags associated with it
343  * 3) The descriptor has subtypes and a separate table will be used.
344  */
345 static const ACPI_RESOURCE_TAG *AcpiGbl_ResourceTags[] =
346 {
347     /* Small descriptors */
348
349     NULL, /* 0x00, Reserved */
350     NULL, /* 0x01, Reserved */
351     NULL, /* 0x02, Reserved */
352     NULL, /* 0x03, Reserved */
353     AcpiDmIrqTags, /* 0x04, ACPI_RESOURCE_NAME_IRQ_FORMAT */
354     AcpiDmDmaTags, /* 0x05, ACPI_RESOURCE_NAME_DMA_FORMAT */
355     NULL, /* 0x06, ACPI_RESOURCE_NAME_START_DEPENDENT */
356     NULL, /* 0x07, ACPI_RESOURCE_NAME_END_DEPENDENT */
357     AcpiDmIoTags, /* 0x08, ACPI_RESOURCE_NAME_IO_PORT */
358     AcpiDmFixedIoTags, /* 0x09, ACPI_RESOURCE_NAME_FIXED_IO_PORT */
359     AcpiDmFixedDmaTags, /* 0x0A, ACPI_RESOURCE_NAME_FIXED_DMA */
360     NULL, /* 0x0B, Reserved */
361     NULL, /* 0x0C, Reserved */
362     NULL, /* 0x0D, Reserved */
363     NULL, /* 0x0E, ACPI_RESOURCE_NAME_SMALL_VENDOR */
364     NULL, /* 0x0F, ACPI_RESOURCE_NAME_END_TAG (not use)
365
366     /* Large descriptors */
367
368     NULL, /* 0x00, Reserved */
369     AcpiDmMemory24Tags, /* 0x01, ACPI_RESOURCE_NAME_MEMORY_24 */
370     AcpiDmRegisterTags, /* 0x02, ACPI_RESOURCE_NAME_GENERIC_REGISTER */
371     NULL, /* 0x03, Reserved */
372     NULL, /* 0x04, ACPI_RESOURCE_NAME_LARGE_VENDOR */
373     AcpiDmMemory32Tags, /* 0x05, ACPI_RESOURCE_NAME_MEMORY_32 */
374     AcpiDmFixedMemory32Tags, /* 0x06, ACPI_RESOURCE_NAME_FIXED_MEMORY_32 */
375     AcpiDmAddress32Tags, /* 0x07, ACPI_RESOURCE_NAME_DWORD_ADDRESS_SP */
376     AcpiDmAddress16Tags, /* 0x08, ACPI_RESOURCE_NAME_WORD_ADDRESS_SPA */
377     AcpiDmInterruptTags, /* 0x09, ACPI_RESOURCE_NAME_EXTENDED_INTERRUPT */
378     AcpiDmAddress64Tags, /* 0x0A, ACPI_RESOURCE_NAME_QWORD_ADDRESS_SP */
379     AcpiDmExtendedAddressTags, /* 0x0B, ACPI_RESOURCE_NAME_EXTENDED_ADDRESS */
380     NULL, /* 0x0C, ACPI_RESOURCE_NAME_GPIO - Use Subty */
381     NULL, /* 0x0D, Reserved */
382     NULL, /* 0x0E, ACPI_RESOURCE_NAME_SERIAL_BUS - Use
383
385 /* GPIO Subtypes */
387 static const ACPI_RESOURCE_TAG *AcpiGbl_GpioResourceTags[] =
388 {
389     AcpiDmGpioIntTags, /* 0x00 Interrupt Connection */

```



```

390     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', 'U', 'K', 'X'};

419 /*****
420  * FUNCTION:    AcpiDmCheckResourceReference
421  *
422  * PARAMETERS:  Op          - Parse Op for the AML opcode
423  *              WalkState    - Current walk state (with valid scope)
424  *
425  * RETURN:      None
426  *
427  * DESCRIPTION: Convert a reference to a resource descriptor to a symbolic
428  *              reference if possible
429  *
430  * NOTE:        Bit index is used to transparently handle both resource bit
431  *              fields and byte fields.
432  *
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;
523 }

525 /* Translate the Index to a resource tag pathname */

527 AcpiGetTagPathname (IndexOp, BufferNode, ResourceNode, BitIndex);
528 }

531 /*****
532 *
533 * FUNCTION:      AcpiDmGetResourceNode
534 *
535 * PARAMETERS:    BufferNode      - Node for the parent buffer
536 *                BitIndex       - Index into the resource descriptor
537 *
538 * RETURN:        Namespace node for the resource descriptor. NULL if not found
539 *
540 * DESCRIPTION:    Find a resource descriptor that corresponds to the bit index
541 *
542 *****/

544 static ACPI_NAMESPACE_NODE *
545 AcpiDmGetResourceNode (
546     ACPI_NAMESPACE_NODE *BufferNode,
547     UINT32 BitIndex)
548 {
549     ACPI_NAMESPACE_NODE *Node;
550     UINT32 ByteIndex = ACPI_DIV_8 (BitIndex);

553     /*
554     * Child list contains an entry for each resource descriptor. Find
555     * the descriptor that corresponds to the Index.
556     *
557     * If there are no children, this is not a resource template
558     */
559     Node = BufferNode->Child;
560     while (Node)
561     {
562         /*
563         * Check if the Index falls within this resource.
564         *
565         * Value contains the resource offset, Object contains the resource
566         * length (both in bytes)
567         */
568         if ((ByteIndex >= Node->Value) &&
569             (ByteIndex < (Node->Value + Node->Length)))
570         {
571             return (Node);
572         }

574         Node = Node->Peer;
575     }

577     return (NULL);
578 }

581 /*****
582 *
583 * FUNCTION:      AcpiGetTagPathname
584 *
585 * PARAMETERS:    BufferNode      - Node for the parent buffer
586 *                ResourceNode    - Node for a resource descriptor
587 *                BitIndex       - Index into the resource descriptor

```

```

588 *
589 * RETURN:        Full pathname for a resource tag. NULL if no match.
590 *                Path is returned in AML (packed) format.
591 *
592 * DESCRIPTION:    Convert a BitIndex into a symbolic resource tag (full pathname)
593 *
594 *****/

596 static char *
597 AcpiGetTagPathname (
598     ACPI_PARSE_OBJECT *IndexOp,
599     ACPI_NAMESPACE_NODE *BufferNode,
600     ACPI_NAMESPACE_NODE *ResourceNode,
601     UINT32 BitIndex)
602 {
603     ACPI_STATUS Status;
604     UINT32 ResourceBitIndex;
605     UINT8 ResourceTableIndex;
606     ACPI_SIZE RequiredSize;
607     char *Pathname;
608     AML_RESOURCE *Aml;
609     ACPI_PARSE_OBJECT *Op;
610     char *InternalPath;
611     char *Tag;

614     /* Get the Op that contains the actual buffer data */

616     Op = BufferNode->Op->Common.Value.Arg;
617     Op = Op->Common.Next;
618     if (!Op)
619     {
620         return (NULL);
621     }

623     /* Get the individual resource descriptor and validate it */

625     Aml = ACPI_CAST_PTR (AML_RESOURCE,
626         &Op->Named.Data[ResourceNode->Value]);

628     Status = AcpiUtValidateResource (NULL, Aml, &ResourceTableIndex);
629     if (ACPI_FAILURE (Status))
630     {
631         return (NULL);
632     }

634     /* Get offset into this descriptor (from offset into entire buffer) */

636     ResourceBitIndex = BitIndex - ACPI_MUL_8 (ResourceNode->Value);

638     /* Get the tag associated with this resource descriptor and offset */

640     Tag = AcpiDmGetResourceTag (ResourceBitIndex, Aml, ResourceTableIndex);
641     if (!Tag)
642     {
643         return (NULL);
644     }

646     /*
647     * Now that we know that we have a reference that can be converted to a
648     * symbol, change the name of the resource to a unique name.
649     */
650     AcpiDmUpdateResourceName (ResourceNode);

652     /* Get the full pathname to the parent buffer */

```

```

654 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 corresponding 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 be
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 {
883     /*
884      * Walk the null-terminated tag list to find a matching bit offset.
885      * We are looking for an exact match.
886      */
887     for ( ; TagList->Tag; TagList++)
888     {
889         if (BitIndex == TagList->BitIndex)
890         {
891             return (TagList->Tag);
892         }
893     }

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)
919 {
920     ACPI_PARSE_OBJECT    *Op = Root;
921     ACPI_PARSE_OBJECT    *Parent;

924     /* Walk the entire parse tree */

926     while (Op)
927     {
928         /* We are interested in Buffer() declarations */

930         if (Op->Common.AmlOpcode == AML_BUFFER_OP)
931         {
932             /* And only declarations of the form Name (XXXX, Buffer()... ) */

934             Parent = Op->Common.Parent;
935             if (Parent->Common.AmlOpcode == AML_NAME_OP)
936             {
937                 /*
938                  * If the buffer is a resource template, add the individual
939                  * resource descriptors to the namespace, as children of the
940                  * buffer node.
941                  */
942                 if (ACPI_SUCCESS (AcpiDmIsResourceTemplate (NULL, Op)))
943                 {
944                     Op->Common.DisasmOpcode = ACPI_DASM_RESOURCE;
945                     AcpiDmAddResourcesToNamespace (Parent->Common.Node, Op);
946                 }
947             }
948         }

950         Op = AcpiPsGetDepthNext (Root, Op);
951     }
952 }

955 /*****
956 *
957 * FUNCTION:      AcpiDmAddResourcesToNamespace
958 *
959 * PARAMETERS:    BufferNode      - Node for the parent buffer
960 *                Op              - Parse op for the buffer
961 *
962 * RETURN:        None
963 *
964 * DESCRIPTION:   Add an entire resource template to the namespace. Each
965 *                resource descriptor is added as a namespace node.
966 *
967 *****/

969 static void
970 AcpiDmAddResourcesToNamespace (
971     ACPI_NAMESPACE_NODE    *BufferNode,
972     ACPI_PARSE_OBJECT      *Op)
973 {
974     ACPI_PARSE_OBJECT      *NextOp;

977     /* Get to the ByteData list */

979     NextOp = Op->Common.Value.Arg;
980     NextOp = NextOp->Common.Next;
981     if (!NextOp)
982     {
983         return;

```

```

984     }

986     /* Set Node and Op to point to each other */

988     BufferNode->Op = Op;
989     Op->Common.Node = BufferNode;

991     /*
992     * Insert each resource into the namespace
993     * NextOp contains the Aml pointer and the Aml length
994     */
995     AcpiUtWalkAmlResources (NULL, (UINT8 *) NextOp->Named.Data,
996                             (ACPI_SIZE) NextOp->Common.Value.Integer,
997                             AcpiDmAddResourceToNamespace, (void **) BufferNode);
998 }

1001 /*****
1002 *
1003 * FUNCTION:      AcpiDmAddResourceToNamespace
1004 *
1005 * PARAMETERS:    ACPI_WALK_AML_CALLBACK
1006 *                BufferNode      - Node for the parent buffer
1007 *
1008 * RETURN:        Status
1009 *
1010 * DESCRIPTION:   Add one resource descriptor to the namespace as a child of the
1011 *                parent buffer. The same name is used for each descriptor. This
1012 *                is changed later to a unique name if the resource is actually
1013 *                referenced by an AML operator.
1014 *
1015 *****/

1017 static ACPI_STATUS
1018 AcpiDmAddResourceToNamespace (
1019     UINT8          *Aml,
1020     UINT32         Length,
1021     UINT32         Offset,
1022     UINT8          ResourceIndex,
1023     void           **Context)
1024 {
1025     ACPI_STATUS     Status;
1026     ACPI_GENERIC_STATE ScopeInfo;
1027     ACPI_NAMESPACE_NODE *Node;

1030     /* TBD: Don't need to add descriptors that have no tags defined? */

1032     /* Add the resource to the namespace, as child of the buffer */

1034     ScopeInfo.Scope.Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Context);
1035     Status = AcpiNsLookup (&ScopeInfo, "_TMP", ACPI_TYPE_LOCAL_RESOURCE,
1036                           ACPI_IMODE_LOAD_PASS2,
1037                           ACPI_NS_NO_UPSEARCH | ACPI_NS_DONT_OPEN_SCOPE | ACPI_NS_PREFIX_I,
1038                           NULL, &Node);
1039     if (ACPI_FAILURE (Status))
1040     {
1041         return (AE_OK);
1042     }

1044     /* Set the name to the default, changed later if resource is referenced */

1046     Node->Name.Integer = ACPI_DEFAULT_RESNAME;

1048     /* Save the offset of the descriptor (within the original buffer) */

```

new/usr/src/common/acpica/common/dmrestag.c

17

```
1050     Node->Value = Offset;  
1051     Node->Length = Length;  
1052     return (AE_OK);  
1053 }
```

new/usr/src/common/acpica/common/dmtable.c

1

```
*****
39952 Fri Jan 17 07:29:30 2014
new/usr/src/common/acpica/common/dmtable.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dmtable - Support for ACPI tables that contain no AML code
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acdisasm.h"
47 #include "actables.h"
48 #include "aslcompiler.h"
49 #include "dtcompiler.h"

51 /* This module used for application-level code only */

53 #define _COMPONENT          ACPI_CA_DISASSEMBLER
54         ACPI_MODULE_NAME    ("dmtable")

56 /* Local Prototypes */

58 static void
```

new/usr/src/common/acpica/common/dmtable.c

2

```
59 AcpiDmCheckAscii (
60     UINT8
61     char
62     UINT32
63     *Target,
64     *RepairedName,
65     Count);

66 /* Common format strings for commented values */

67 #define UINT8_FORMAT        "%2.2X [%s]\n"
68 #define UINT16_FORMAT       "%4.4X [%s]\n"
69 #define UINT32_FORMAT       "%8.8X [%s]\n"
70 #define STRING_FORMAT       "[%s]\n"

72 /* These tables map a subtable type to a description string */

74 static const char          *AcpiDmAsfSubnames[] =
75 {
76     "ASF Information",
77     "ASF Alerts",
78     "ASF Remote Control",
79     "ASF RMCP Boot Options",
80     "ASF Address",
81     "Unknown SubTable Type"          /* Reserved */
82 };

84 static const char          *AcpiDmDmarSubnames[] =
85 {
86     "Hardware Unit Definition",
87     "Reserved Memory Region",
88     "Root Port ATS Capability",
89     "Remapping Hardware Static Affinity",
90     "Unknown SubTable Type"          /* Reserved */
91 };

93 static const char          *AcpiDmEinjActions[] =
94 {
95     "Begin Operation",
96     "Get Trigger Table",
97     "Set Error Type",
98     "Get Error Type",
99     "End Operation",
100    "Execute Operation",
101    "Check Busy Status",
102    "Get Command Status",
103    "Set Error Type With Address",
104    "Unknown Action"
105 };

107 static const char          *AcpiDmEinjInstructions[] =
108 {
109     "Read Register",
110     "Read Register Value",
111     "Write Register",
112     "Write Register Value",
113     "Noop",
114     "Flush Cacheline",
115     "Unknown Instruction"
116 };

118 static const char          *AcpiDmErstActions[] =
119 {
120     "Begin Write Operation",
121     "Begin Read Operation",
122     "Begin Clear Operation",
123     "End Operation",
124     "Set Record Offset",
```

```

125 "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, AcpiDmTableInfoBgrrt, 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_ECDDT, AcpiDmTableInfoEcddt, 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_MTMt, NULL, AcpiDmDumpMtmr, DtCompileMtmr, Temp
313     {ACPI_SIG_PCCT, AcpiDmTableInfoPcct, AcpiDmDumpPcct, DtCompilePcct, Temp
314     {ACPI_SIG_PMTT, NULL, AcpiDmDumpPmtt, DtCompilePmtt, Temp
315     {ACPI_SIG_RSDDT, 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, AcpiDmTableInfoTcpc, 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_XSDDT, 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  * FUNCTION:    AcpiDmDumpDataTable
407  *
408  * PARAMETERS:  Table          - An ACPI table
409  *
410  * RETURN:      None.
411  *
412  * DESCRIPTION: Format the contents of an ACPI data table (any table other
413  *              than an SSDT or DSDT that does not contain executable AML code)
414  *
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 *
522 * FUNCTION:      AcpiDmLineHeader
523 *
524 * PARAMETERS:    Offset          - Current byte offset, from table start
525 *                ByteLength      - Length of the field in bytes, 0 for flags
526 *                Name            - Name of this field
527 *                Value           - Optional value, displayed on left of ':'
528 *
529 * RETURN:        None
530 *
531 * DESCRIPTION:    Utility routines for formatting output lines. Displays the
532 *                current table offset in hex and decimal, the field length,
533 *                and the field name.
534 *
535 *****/
537 void
538 AcpiDmLineHeader (
539     UINT32          Offset,
540     UINT32          ByteLength,
541     char            *Name)
542 {
543
544     /* Allow a null name for fields that span multiple lines (large buffers) */
545
546     if (!Name)
547     {
548         Name = "";
549     }
550
551     if (Gbl_DoTemplates && !Gbl_VerboseTemplates) /* Terse template */
552     {
553         if (ByteLength)
554         {
555             AcpiOsPrintf ("[%.4d] %34s : ", ByteLength, Name);
556         }
557         else
558         {
559             if (*Name)
560             {
561                 AcpiOsPrintf ("%41s : ", Name);
562             }
563             else
564             {
565                 AcpiOsPrintf ("%41s ", Name);
566             }
567         }
568     }
569     else /* Normal disassembler or verbose template */
570     {
571         if (ByteLength)
572         {
573             AcpiOsPrintf ("[%3.3Xh %4.4d %4d] %28s : ",
574                 Offset, Offset, ByteLength, Name);
575         }
576         else
577         {
578             if (*Name)
579             {
580                 AcpiOsPrintf ("%44s : ", Name);
581             }
582             else
583             {
584                 AcpiOsPrintf ("%44s ", Name);
585             }
586         }
587     }
588 }

```

```

587     }
588 }
589
590 void
591 AcpiDmLineHeader2 (
592     UINT32          Offset,
593     UINT32          ByteLength,
594     char            *Name,
595     UINT32          Value)
596 {
597
598     if (Gbl_DoTemplates && !Gbl_VerboseTemplates) /* Terse template */
599     {
600         if (ByteLength)
601         {
602             AcpiOsPrintf ("[%.4d] %30s %3d : ",
603                 ByteLength, Name, Value);
604         }
605         else
606         {
607             AcpiOsPrintf ("%36s % 3d : ",
608                 Name, Value);
609         }
610     }
611     else /* Normal disassembler or verbose template */
612     {
613         if (ByteLength)
614         {
615             AcpiOsPrintf ("[%3.3Xh %4.4d %3d] %24s %3d : ",
616                 Offset, Offset, ByteLength, Name, Value);
617         }
618         else
619         {
620             AcpiOsPrintf ("[%3.3Xh %4.4d ] %24s %3d : ",
621                 Offset, Offset, Name, Value);
622         }
623     }
624 }
625
626
627 *****/
628 *
629 * FUNCTION:      AcpiDmDumpTable
630 *
631 * PARAMETERS:    TableLength      - Length of the entire ACPI table
632 *                TableOffset      - Starting offset within the table for this
633 *                sub-descriptor (0 if main table)
634 *                Table            - The ACPI table
635 *                SubtableLength    - Length of this sub-descriptor
636 *                Info             - Info table for this ACPI table
637 *
638 * RETURN:        None
639 *
640 * DESCRIPTION:    Display ACPI table contents by walking the Info table.
641 *
642 * Note: This function must remain in sync with DtGetFieldLength.
643 *
644 *****/
646 ACPI_STATUS
647 AcpiDmDumpTable (
648     UINT32          TableLength,
649     UINT32          TableOffset,
650     void            *Table,
651     UINT32          SubtableLength,
652     ACPI_DMTABLE_INFO *Info)

```

```

653 {
654     UINT8          *Target;
655     UINT32          CurrentOffset;
656     UINT32          ByteLength;
657     UINT8          Temp8;
658     UINT16          Temp16;
659     UINT64          Value;
660     ACPI_DMTABLE_DATA *TableData;
661     const char      *Name;
662     BOOLEAN          LastOutputBlankLine = FALSE;
663     char             RepairedName[8];

666     if (!Info)
667     {
668         AcpiOsPrintf ("Display not implemented\n");
669         return (AE_NOT_IMPLEMENTED);
670     }

672     /* Walk entire Info table; Null name terminates */

674     for (; Info->Name; Info++)
675     {
676         /*
677          * Target points to the field within the ACPI Table. CurrentOffset is
678          * the offset of the field from the start of the main table.
679          */
680         Target = ACPI_ADD_PTR (UINT8, Table, Info->Offset);
681         CurrentOffset = TableOffset + Info->Offset;

683         /* Check for beyond EOT or beyond subtable end */

685         if ((CurrentOffset >= TableLength) ||
686             (SubtableLength && (Info->Offset >= SubtableLength)))
687         {
688             AcpiOsPrintf ("**** ACPI table terminates in the middle of a data st
689             return (AE_BAD_DATA);
690         }

692         /* Generate the byte length for this field */

694         switch (Info->Opcode)
695         {
696             case ACPI_DMT_UINT8:
697             case ACPI_DMT_CHKSUM:
698             case ACPI_DMT_SPACEID:
699             case ACPI_DMT_ACCWIDTH:
700             case ACPI_DMT_IVRS:
701             case ACPI_DMT_MADT:
702             case ACPI_DMT_PCCT:
703             case ACPI_DMT_PMTT:
704             case ACPI_DMT_SRAT:
705             case ACPI_DMT_ASF:
706             case ACPI_DMT_HESTNTYP:
707             case ACPI_DMT_FADTPM:
708             case ACPI_DMT_EINJACT:
709             case ACPI_DMT_EINJINST:
710             case ACPI_DMT_ERSTACT:
711             case ACPI_DMT_ERSTINST:

713                 ByteLength = 1;
714                 break;

716             case ACPI_DMT_UINT16:
717             case ACPI_DMT_DMAR:
718             case ACPI_DMT_HEST:

```

```

720         ByteLength = 2;
721         break;

723         case ACPI_DMT_UINT24:

725             ByteLength = 3;
726             break;

728         case ACPI_DMT_UINT32:
729         case ACPI_DMT_NAME4:
730         case ACPI_DMT_SIG:
731         case ACPI_DMT_SLIC:

733             ByteLength = 4;
734             break;

736         case ACPI_DMT_UINT40:

738             ByteLength = 5;
739             break;

741         case ACPI_DMT_UINT48:
742         case ACPI_DMT_NAME6:

744             ByteLength = 6;
745             break;

747         case ACPI_DMT_UINT56:
748         case ACPI_DMT_BUF7:

750             ByteLength = 7;
751             break;

753         case ACPI_DMT_UINT64:
754         case ACPI_DMT_NAME8:

756             ByteLength = 8;
757             break;

759         case ACPI_DMT_BUF10:

761             ByteLength = 10;
762             break;

764         case ACPI_DMT_BUF16:
765         case ACPI_DMT_UUID:

767             ByteLength = 16;
768             break;

770         case ACPI_DMT_BUF128:

772             ByteLength = 128;
773             break;

775         case ACPI_DMT_STRING:

777             ByteLength = ACPI_STRLEN (ACPI_CAST_PTR (char, Target)) + 1;
778             break;

780         case ACPI_DMT_GAS:

782             if (!LastOutputBlankLine)
783             {
784                 AcpiOsPrintf ("\n");

```

```

785         LastOutputBlankLine = TRUE;
786     }
787     ByteLength = sizeof (ACPI_GENERIC_ADDRESS);
788     break;
790 case ACPI_DMT_HESTNTFY:
791     if (!LastOutputBlankLine)
792     {
793         AcpiOsPrintf ("\n");
794         LastOutputBlankLine = TRUE;
795     }
796     ByteLength = sizeof (ACPI_HEST_NOTIFY);
797     break;
798
800 default:
801     ByteLength = 0;
802     break;
803 }
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 }
811
812 if (Info->Opcode == ACPI_DMT_EXTRA_TEXT)
813 {
814     AcpiOsPrintf ("%s", Info->Name);
815     continue;
816 }
817
818 /* Start a new line and decode the opcode */
819
820 AcpiDmLineHeader (CurrentOffset, ByteLength, Info->Name);
821
822 switch (Info->Opcode)
823 {
824     /* Single-bit Flag fields. Note: Opcode is the bit position */
825
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:
834
835     AcpiOsPrintf ("%1.1X\n", (*Target >> Info->Opcode) & 0x01);
836     break;
837
838 /* 2-bit Flag fields */
839
840 case ACPI_DMT_FLAGS0:
841
842     AcpiOsPrintf ("%1.1X\n", *Target & 0x03);
843     break;
844
845 case ACPI_DMT_FLAGS1:
846
847     AcpiOsPrintf ("%1.1X\n", (*Target >> 1) & 0x03);
848     break;
849
850 case ACPI_DMT_FLAGS2:

```

```

852     AcpiOsPrintf ("%1.1X\n", (*Target >> 2) & 0x03);
853     break;
854
855 case ACPI_DMT_FLAGS4:
856
857     AcpiOsPrintf ("%1.1X\n", (*Target >> 4) & 0x03);
858     break;
859
860 /* Integer Data Types */
861
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     }
881
882     if (!Value && (Info->Flags & DT_DESCRIPTES_OPTIONAL))
883     {
884         AcpiOsPrintf (" [Optional field not present]");
885     }
886
887     AcpiOsPrintf ("\n");
888     break;
889
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:
919
920         /* Convert 16-byte UUID buffer to 36-byte formatted UUID string */
921
922         (void) AuConvertUuidToString ((char *) Target, MsgBuffer);
923
924         AcpiOsPrintf ("%s\n", MsgBuffer);
925         break;
926
927     case ACPI_DMT_STRING:
928
929         AcpiOsPrintf ("%s\n", ACPI_CAST_PTR (char, Target));
930         break;
931
932     /* Fixed length ASCII name fields */
933
934     case ACPI_DMT_SIG:
935
936         AcpiDmCheckAscii (Target, RepairedName, 4);
937         AcpiOsPrintf ("%4s\n", 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;
948
949     case ACPI_DMT_NAME4:
950
951         AcpiDmCheckAscii (Target, RepairedName, 4);
952         AcpiOsPrintf ("%4s\n", RepairedName);
953         break;
954
955     case ACPI_DMT_NAME6:
956
957         AcpiDmCheckAscii (Target, RepairedName, 6);
958         AcpiOsPrintf ("%6s\n", RepairedName);
959         break;
960
961     case ACPI_DMT_NAME8:
962
963         AcpiDmCheckAscii (Target, RepairedName, 8);
964         AcpiOsPrintf ("%8s\n", RepairedName);
965         break;
966
967     /* Special Data Types */
968
969     case ACPI_DMT_CHKSUM:
970
971         /* Checksum, display and validate */
972
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;
984
985     case ACPI_DMT_SPACEID:
986
987         /* Address Space ID */
988
989         AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiUtGetRegionName (*Target));
990         break;
991
992     case ACPI_DMT_ACCWIDTH:
993
994         /* Encoded Access Width */
995
996         Temp8 = *Target;
997         if (Temp8 > ACPI_GAS_WIDTH_RESERVED)
998         {
999             Temp8 = ACPI_GAS_WIDTH_RESERVED;
1000         }
1001
1002         AcpiOsPrintf (UINT8_FORMAT, Temp8, AcpiDmGasAccessWidth[Temp8]);
1003         break;
1004
1005     case ACPI_DMT_GAS:
1006
1007         /* Generic Address Structure */
1008
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;
1015
1016     case ACPI_DMT_ASF:
1017
1018         /* ASF subtable types */
1019
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         }
1025
1026         AcpiOsPrintf (UINT8_FORMAT, *Target, AcpiDmAsfSubnames[Temp16]);
1027         break;
1028
1029     case ACPI_DMT_DMAR:
1030
1031         /* DMAR subtable types */
1032
1033         Temp16 = ACPI_GET16 (Target);
1034         if (Temp16 > ACPI_DMAR_TYPE_RESERVED)
1035         {
1036             Temp16 = ACPI_DMAR_TYPE_RESERVED;
1037         }
1038
1039         AcpiOsPrintf (UINT16_FORMAT, ACPI_GET16 (Target), AcpiDmDmarSubnames
1040         break;
1041
1042     case ACPI_DMT_EINJACT:
1043
1044         /* EINJ Action types */
1045
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 * FUNCTION:      AcpiDmCheckAscii
1261 * PARAMETERS:    Name           - Ascii string
1262 *                Count          - Number of characters to check
1263 * RETURN:        None
1264 * DESCRIPTION:   Ensure that the requested number of characters are printable
1265 *                Ascii characters. Sets non-printable and null chars to <space>.
1266 * *****/
1267 static void
1268 AcpiDmCheckAscii (
1269     UINT8          *Name,
1270     char           *RepairedName,
1271     UINT32         Count)
1272 {
1273     UINT32         i;
1274     for (i = 0; i < Count; i++)
1275     {
1276         RepairedName[i] = (char) Name[i];
1277         if (!Name[i])
1278         {
1279             return;
1280         }
1281         if (!isprint (Name[i]))
1282         {
1283             RepairedName[i] = ' ';
1284         }
1285     }
1286 }

```



new/usr/src/common/acpica/common/dmtdump.c

1

```
*****
73554 Fri Jan 17 07:29:30 2014
new/usr/src/common/acpica/common/dmtdump.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dmtdump - Dump ACPI data tables that contain no AML code
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "acpi.h"
45 #include "accommon.h"
46 #include "acdisasm.h"
47 #include "actables.h"
48
49 /* This module used for application-level code only */
50
51 #define _COMPONENT          ACPI_CA_DISASSEMBLER
52 #define ACPI_MODULE_NAME    ("dmtdump")
53
54
55 /* Local prototypes */
56
57 static void
58 AcpiDmValidateFadtLength (
```

new/usr/src/common/acpica/common/dmtdump.c

2

```
59     UINT32          Revision,
60     UINT32          Length);
61
62 static void
63 AcpiDmDumpBuffer (
64     void            *Table,
65     UINT32          BufferOffset,
66     UINT32          Length,
67     UINT32          AbsoluteOffset,
68     char            *Header);
69
70
71 /*****
72 *
73 * FUNCTION:      AcpiDmDumpBuffer
74 *
75 * PARAMETERS:    Table          - ACPI Table or subtable
76 *                BufferOffset    - Offset of buffer from Table above
77 *                Length          - Length of the buffer
78 *                AbsoluteOffset  - Offset of buffer in the main ACPI table
79 *                Header          - Name of the buffer field (printed on the
80 *                                first line only.)
81 *
82 * RETURN:        None
83 *
84 * DESCRIPTION:   Format the contents of an arbitrary length data buffer (in the
85 *                disassembler output format.)
86 *
87 *****/
88
89 static void
90 AcpiDmDumpBuffer (
91     void            *Table,
92     UINT32          BufferOffset,
93     UINT32          Length,
94     UINT32          AbsoluteOffset,
95     char            *Header)
96 {
97     UINT8           *Buffer;
98     UINT32          i;
99
100
101     if (!Length)
102     {
103         return;
104     }
105
106     Buffer = ACPI_CAST_PTR (UINT8, Table) + BufferOffset;
107     i = 0;
108
109     while (i < Length)
110     {
111         if (!(i % 16))
112         {
113             AcpiOsPrintf ("\n");
114             AcpiDmLineHeader (AbsoluteOffset,
115                             ((Length - i) > 16) ? 16 : (Length - i), Header);
116             Header = NULL;
117         }
118
119         AcpiOsPrintf ("%02X ", *Buffer);
120         i++;
121         Buffer++;
122         AbsoluteOffset++;
123     }
```

```

125     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:     AcpiDmDumpRsdtd

```

```

191  *
192  * PARAMETERS:  Table          - A RSDT
193  *
194  * RETURN:      None
195  *
196  * DESCRIPTION: Format the contents of a RSDT
197  *
198  *****/

200 void
201 AcpiDmDumpRsdtd (
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 {
288     /* Always dump the minimum FADT revision 1 fields (ACPI 1.0) */
289
291     AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoFadt1);
292
293     /* Check for FADT revision 2 fields (ACPI 1.0B MS extensions) */
294
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     }
300
301     /* Check for FADT revision 3/4 fields and up (ACPI 2.0+ extended data) */
302
303     else if (Table->Length > ACPI_FADT_V2_SIZE)
304     {
305         AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoFadt3);
306
307         /* Check for FADT revision 5 fields and up (ACPI 5.0+) */
308
309         if (Table->Length > ACPI_FADT_V3_SIZE)
310         {
311             AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoFadt5);
312         }
313     }
314
315     /* Validate various fields in the FADT, including length */
316
317     AcpiTbCreateLocalFadt (Table, Table->Length);
318
319     /* Validate FADT length against the revision */
320
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.Length);
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", 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 * FUNCTION:      AcpiDmDumpFpdt
1075 *
1076 * PARAMETERS:    Table          - A FPDT table
1077 *
1078 * RETURN:        None
1079 *
1080 * DESCRIPTION:   Format the contents of a FPDT. This table type consists
1081 *                of an open-ended number of subtables.
1082 *
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", SubTable->Type);

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 * FUNCTION:      AcpiDmDumpHest
1158 *
1159 * PARAMETERS:    Table          - A HEST table
1160 *
1161 * RETURN:        None
1162 *
1163 * DESCRIPTION:   Format the contents of a HEST. This table type consists
1164 *                of an open-ended number of subtables.
1165 *
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     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;

```



```

1445     AcpiOsPrintf (
1446         "\n**** Unknown IVRS device entry type/length: "
1447         "0x%.2X/0x%X at offset 0x%.4X: (header below)\n",
1448         EntryType, EntryLength, EntryOffset);
1449     break;
1450 }
1451
1452 /* Dump the Device Entry */
1453
1454 Status = AcpiDmDumpTable (Table->Length, EntryOffset,
1455     DeviceEntry, EntryLength, InfoTable);
1456
1457 EntryOffset += EntryLength;
1458 DeviceEntry = ACPI_ADD_PTR (ACPI_IVRS_DE_HEADER, DeviceEntry,
1459     EntryLength);
1460 }
1461 }
1462
1463 NextSubTable:
1464 /* Point to next sub-table */
1465
1466 Offset += SubTable->Length;
1467 SubTable = ACPI_ADD_PTR (ACPI_IVRS_HEADER, SubTable, SubTable->Length);
1468 }
1469 }
1470
1471 /*****
1472 *
1473 * FUNCTION:     AcpiDmDumpMadt
1474 *
1475 * PARAMETERS:  Table          - A MADT table
1476 *
1477 * RETURN:      None
1478 *
1479 * DESCRIPTION: Format the contents of a MADT. This table type consists
1480 *               of an open-ended number of subtables.
1481 *
1482 * *****/
1483
1484 void
1485 AcpiDmDumpMadt (
1486     ACPI_TABLE_HEADER *Table)
1487 {
1488     ACPI_STATUS      Status;
1489     ACPI_SUBTABLE_HEADER *SubTable;
1490     UINT32           Length = Table->Length;
1491     UINT32           Offset = sizeof (ACPI_TABLE_MADT);
1492     ACPI_DMTABLE_INFO *InfoTable;
1493
1494     /* Main table */
1495
1496     Status = AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoMadt);
1497     if (ACPI_FAILURE (Status))
1498     {
1499         return;
1500     }
1501
1502     /* Sub-tables */
1503
1504     SubTable = ACPI_ADD_PTR (ACPI_SUBTABLE_HEADER, Table, Offset);
1505     while (Offset < Table->Length)
1506     {
1507         /* Common sub-table header */

```

```

1511     AcpiOsPrintf ("\n");
1512     Status = AcpiDmDumpTable (Length, Offset, SubTable,
1513         SubTable->Length, AcpiDmTableInfoMadtHdr);
1514     if (ACPI_FAILURE (Status))
1515     {
1516         return;
1517     }
1518
1519     switch (SubTable->Type)
1520     {
1521     case ACPI_MADT_TYPE_LOCAL_APIC:
1522
1523         InfoTable = AcpiDmTableInfoMadt0;
1524         break;
1525
1526     case ACPI_MADT_TYPE_IO_APIC:
1527
1528         InfoTable = AcpiDmTableInfoMadt1;
1529         break;
1530
1531     case ACPI_MADT_TYPE_INTERRUPT_OVERRIDE:
1532
1533         InfoTable = AcpiDmTableInfoMadt2;
1534         break;
1535
1536     case ACPI_MADT_TYPE_NMI_SOURCE:
1537
1538         InfoTable = AcpiDmTableInfoMadt3;
1539         break;
1540
1541     case ACPI_MADT_TYPE_LOCAL_APIC_NMI:
1542
1543         InfoTable = AcpiDmTableInfoMadt4;
1544         break;
1545
1546     case ACPI_MADT_TYPE_LOCAL_APIC_OVERRIDE:
1547
1548         InfoTable = AcpiDmTableInfoMadt5;
1549         break;
1550
1551     case ACPI_MADT_TYPE_IO_SAPIC:
1552
1553         InfoTable = AcpiDmTableInfoMadt6;
1554         break;
1555
1556     case ACPI_MADT_TYPE_LOCAL_SAPIC:
1557
1558         InfoTable = AcpiDmTableInfoMadt7;
1559         break;
1560
1561     case ACPI_MADT_TYPE_INTERRUPT_SOURCE:
1562
1563         InfoTable = AcpiDmTableInfoMadt8;
1564         break;
1565
1566     case ACPI_MADT_TYPE_LOCAL_X2APIC:
1567
1568         InfoTable = AcpiDmTableInfoMadt9;
1569         break;
1570
1571     case ACPI_MADT_TYPE_LOCAL_X2APIC_NMI:
1572
1573         InfoTable = AcpiDmTableInfoMadt10;
1574         break;
1575
1576     case ACPI_MADT_TYPE_GENERIC_INTERRUPT:

```

```

1578         InfoTable = AcpiDmTableInfoMadt11;
1579         break;

1581     case ACPI_MADT_TYPE_GENERIC_DISTRIBUTOR:

1583         InfoTable = AcpiDmTableInfoMadt12;
1584         break;

1586     default:

1588         AcpiOsPrintf ("\n*** Unknown MADT sub-table type 0x%X\n", SubTable->Type);
1589         /* Attempt to continue */

1592         if (!SubTable->Length)
1593         {
1594             AcpiOsPrintf ("Invalid zero length subtable\n");
1595             return;
1596         }
1597         goto NextSubTable;
1598     }

1600     Status = AcpiDmDumpTable (Length, Offset, SubTable,
1601                               SubTable->Length, InfoTable);
1602     if (ACPI_FAILURE (Status))
1603     {
1604         return;
1605     }

1607 NextSubTable:
1608     /* Point to next sub-table */

1610     Offset += SubTable->Length;
1611     SubTable = ACPI_ADD_PTR (ACPI_SUBTABLE_HEADER, SubTable, SubTable->Length);
1612 }
1613 }

1616 /*****
1617 *
1618 * FUNCTION:      AcpiDmDumpMcfg
1619 *
1620 * PARAMETERS:    Table          - A MCFG Table
1621 *
1622 * RETURN:        None
1623 *
1624 * DESCRIPTION:   Format the contents of a MCFG table
1625 *
1626 *****/

1628 void
1629 AcpiDmDumpMcfg (
1630     ACPI_TABLE_HEADER *Table)
1631 {
1632     ACPI_STATUS Status;
1633     UINT32 Offset = sizeof (ACPI_TABLE_MCFG);
1634     ACPI_MCFG_ALLOCATION *SubTable;

1637     /* Main table */

1639     Status = AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoMcfg);
1640     if (ACPI_FAILURE (Status))
1641     {
1642         return;

```

```

1643     }

1645     /* Sub-tables */

1647     SubTable = ACPI_ADD_PTR (ACPI_MCFG_ALLOCATION, Table, Offset);
1648     while (Offset < Table->Length)
1649     {
1650         if (Offset + sizeof (ACPI_MCFG_ALLOCATION) > Table->Length)
1651         {
1652             AcpiOsPrintf ("Warning: there are %u invalid trailing bytes\n",
1653                           sizeof (ACPI_MCFG_ALLOCATION) - (Offset - Table->Length));
1654             return;
1655         }

1657         AcpiOsPrintf ("\n");
1658         Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
1659                                   sizeof (ACPI_MCFG_ALLOCATION), AcpiDmTableInfoMcfg0);
1660         if (ACPI_FAILURE (Status))
1661         {
1662             return;
1663         }

1665         /* Point to next sub-table (each subtable is of fixed length) */

1667         Offset += sizeof (ACPI_MCFG_ALLOCATION);
1668         SubTable = ACPI_ADD_PTR (ACPI_MCFG_ALLOCATION, SubTable,
1669                                   sizeof (ACPI_MCFG_ALLOCATION));
1670     }
1671 }

1674 /*****
1675 *
1676 * FUNCTION:      AcpiDmDumpMpst
1677 *
1678 * PARAMETERS:    Table          - A MPST Table
1679 *
1680 * RETURN:        None
1681 *
1682 * DESCRIPTION:   Format the contents of a MPST table
1683 *
1684 *****/

1686 void
1687 AcpiDmDumpMpst (
1688     ACPI_TABLE_HEADER *Table)
1689 {
1690     ACPI_STATUS Status;
1691     UINT32 Offset = sizeof (ACPI_TABLE_MPST);
1692     *SubTable0;
1693     ACPI_MPST_POWER_STATE *SubTable0A;
1694     ACPI_MPST_COMPONENT *SubTable0B;
1695     ACPI_MPST_DATA_HDR *SubTable1;
1696     ACPI_MPST_POWER_DATA *SubTable2;
1697     UINT16 SubtableCount;
1698     UINT32 PowerStateCount;
1699     UINT32 ComponentCount;

1702     /* Main table */

1704     Status = AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoMpst);
1705     if (ACPI_FAILURE (Status))
1706     {
1707         return;
1708     }

```

```

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);
1842 if (ACPI_FAILURE (Status))
1843 {
1844     return;
1845 }
1847 /* Sub-tables */
1849 SubTable = ACPI_ADD_PTR (ACPI_MSCT_PROXIMITY, Table, Offset);
1850 while (Offset < Table->Length)
1851 {
1852     /* Common sub-table header */
1854     AcpiOsPrintf ("\n");
1855     Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
1856                             sizeof (ACPI_MSCT_PROXIMITY), AcpiDmTableInfoMsct0);
1857     if (ACPI_FAILURE (Status))
1858     {
1859         return;
1860     }
1862     /* Point to next sub-table */
1864     Offset += sizeof (ACPI_MSCT_PROXIMITY);
1865     SubTable = ACPI_ADD_PTR (ACPI_MSCT_PROXIMITY, SubTable, sizeof (ACPI_MSCT_PROXIMITY));
1866 }
1867 }

1870 /*****
1871 *
1872 * FUNCTION:      AcpiDmDumpMtmr
1873 *
1874 * PARAMETERS:    Table              - A MTMR table
1875 *
1876 * RETURN:        None
1877 *
1878 * DESCRIPTION:   Format the contents of a MTMR
1879 *
1880 *****/
1882 void
1883 AcpiDmDumpMtmr (
1884     ACPI_TABLE_HEADER *Table)
1885 {
1886     ACPI_STATUS Status;
1887     UINT32 Offset = sizeof (ACPI_TABLE_MTMR);
1888     ACPI_MTMR_ENTRY *SubTable;
1891     /* Main table */
1893     Status = AcpiDmDumpTable (Table->Length, 0, Table, 0, AcpiDmTableInfoMtmr);
1894     if (ACPI_FAILURE (Status))
1895     {
1896         return;
1897     }
1899     /* Sub-tables */
1901     SubTable = ACPI_ADD_PTR (ACPI_MTMR_ENTRY, Table, Offset);
1902     while (Offset < Table->Length)
1903     {
1904         /* Common sub-table header */
1906         AcpiOsPrintf ("\n");

```

```

1907 Status = AcpiDmDumpTable (Table->Length, Offset, SubTable,
1908                          sizeof (ACPI_MTMR_ENTRY), AcpiDmTableInfoMtmr0);
1909 if (ACPI_FAILURE (Status))
1910 {
1911     return;
1912 }
1914 /* Point to next sub-table */
1916 Offset += sizeof (ACPI_MTMR_ENTRY);
1917 SubTable = ACPI_ADD_PTR (ACPI_MTMR_ENTRY, SubTable, sizeof (ACPI_MTMR_ENTRY));
1918 }
1919 }

1922 /*****
1923 *
1924 * FUNCTION:      AcpiDmDumpPcct
1925 *
1926 * PARAMETERS:    Table              - A PCCT table
1927 *
1928 * RETURN:        None
1929 *
1930 * DESCRIPTION:   Format the contents of a PCCT. This table type consists
1931 *                of an open-ended number of subtables.
1932 *
1933 *****/
1935 void
1936 AcpiDmDumpPcct (
1937     ACPI_TABLE_HEADER *Table)
1938 {
1939     ACPI_STATUS Status;
1940     ACPI_PCCT_SUBSPACE *SubTable;
1941     UINT32 Length = Table->Length;
1942     UINT32 Offset = sizeof (ACPI_TABLE_PCCT);
1945     /* Main table */
1947     Status = AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoPcct);
1948     if (ACPI_FAILURE (Status))
1949     {
1950         return;
1951     }
1953     /* Subtables */
1955     SubTable = ACPI_ADD_PTR (ACPI_PCCT_SUBSPACE, Table, Offset);
1956     while (Offset < Table->Length)
1957     {
1958         /* Common subtable header */
1960         AcpiOsPrintf ("\n");
1961         Status = AcpiDmDumpTable (Length, Offset, SubTable,
1962                                 SubTable->Header.Length, AcpiDmTableInfoPcctHdr);
1963         if (ACPI_FAILURE (Status))
1964         {
1965             return;
1966         }
1968         /* ACPI 5.0: Only one type of PCCT subtable is supported */
1970         if (SubTable->Header.Type != ACPI_PCCT_TYPE_GENERIC_SUBSPACE)
1971         {
1972             AcpiOsPrintf (

```

```

1973         "\n**** Unexpected or unknown PCCT subtable type 0x%X\n",
1974         SubTable->Header.Type);
1975     return;
1976 }
1977
1978     AcpiOsPrintf ("\n");
1979     Status = AcpiDmDumpTable (Length, Offset, SubTable,
1980         SubTable->Header.Length, AcpiDmTableInfoPcct0);
1981     if (ACPI_FAILURE (Status))
1982     {
1983         return;
1984     }
1985
1986     /* Point to next subtable */
1987
1988     Offset += SubTable->Header.Length;
1989     SubTable = ACPI_ADD_PTR (ACPI_PCCT_SUBSPACE, SubTable,
1990         SubTable->Header.Length);
1991 }
1992 }
1993
1994 /*****
1995  *
1996  * FUNCTION:      AcpiDmDumpPmtt
1997  *
1998  * PARAMETERS:    Table              - A PMTT table
1999  *
2000  * RETURN:        None
2001  *
2002  * DESCRIPTION:   Format the contents of a PMTT. This table type consists
2003  *               of an open-ended number of subtables.
2004  *
2005  * *****/
2006
2007 void
2008 AcpiDmDumpPmtt (
2009     ACPI_TABLE_HEADER *Table)
2010 {
2011     ACPI_STATUS      Status;
2012     ACPI_PMTT_HEADER *SubTable;
2013     ACPI_PMTT_HEADER *MemSubTable;
2014     ACPI_PMTT_HEADER *DimmSubTable;
2015     ACPI_PMTT_DOMAIN *DomainArray;
2016     UINT32           Length = Table->Length;
2017     UINT32           Offset = sizeof (ACPI_TABLE_PMTT);
2018     UINT32           MemOffset;
2019     UINT32           DimmOffset;
2020     UINT32           DomainOffset;
2021     UINT32           DomainCount;
2022
2023     /* Main table */
2024
2025     Status = AcpiDmDumpTable (Length, 0, Table, 0, AcpiDmTableInfoPmtt);
2026     if (ACPI_FAILURE (Status))
2027     {
2028         return;
2029     }
2030
2031     /* Subtables */
2032
2033     SubTable = ACPI_ADD_PTR (ACPI_PMTT_HEADER, Table, Offset);
2034     while (Offset < Table->Length)
2035     {
2036         /* Common subtable header */

```

```

2040     AcpiOsPrintf ("\n");
2041     Status = AcpiDmDumpTable (Length, Offset, SubTable,
2042         SubTable->Length, AcpiDmTableInfoPmttHdr);
2043     if (ACPI_FAILURE (Status))
2044     {
2045         return;
2046     }
2047
2048     /* Only Socket subtables are expected at this level */
2049
2050     if (SubTable->Type != ACPI_PMTT_TYPE_SOCKET)
2051     {
2052         AcpiOsPrintf (
2053             "\n**** Unexpected or unknown PMTT subtable type 0x%X\n",
2054             SubTable->Type);
2055         return;
2056     }
2057
2058     /* Dump the fixed-length portion of the subtable */
2059
2060     Status = AcpiDmDumpTable (Length, Offset, SubTable,
2061         SubTable->Length, AcpiDmTableInfoPmtt0);
2062     if (ACPI_FAILURE (Status))
2063     {
2064         return;
2065     }
2066
2067     /* Walk the memory controller subtables */
2068
2069     MemOffset = sizeof (ACPI_PMTT_SOCKET);
2070     MemSubTable = ACPI_ADD_PTR (ACPI_PMTT_HEADER, SubTable,
2071         sizeof (ACPI_PMTT_SOCKET));
2072
2073     while (((Offset + MemOffset) < Table->Length) &&
2074         (MemOffset < SubTable->Length))
2075     {
2076         /* Common subtable header */
2077
2078         AcpiOsPrintf ("\n");
2079         Status = AcpiDmDumpTable (Length,
2080             Offset + MemOffset, MemSubTable,
2081             MemSubTable->Length, AcpiDmTableInfoPmttHdr);
2082         if (ACPI_FAILURE (Status))
2083         {
2084             return;
2085         }
2086
2087         /* Only memory controller subtables are expected at this level */
2088
2089         if (MemSubTable->Type != ACPI_PMTT_TYPE_CONTROLLER)
2090         {
2091             AcpiOsPrintf (
2092                 "\n**** Unexpected or unknown PMTT subtable type 0x%X\n",
2093                 MemSubTable->Type);
2094             return;
2095         }
2096
2097         /* Dump the fixed-length portion of the controller subtable */
2098
2099         Status = AcpiDmDumpTable (Length,
2100             Offset + MemOffset, MemSubTable,
2101             MemSubTable->Length, AcpiDmTableInfoPmtt1);
2102         if (ACPI_FAILURE (Status))
2103         {
2104             return;

```



```

2105     }
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     DimmOffset);
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)
2223     /* 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
2263         /* Attempt to continue */
2264         if (!SubTable->Length)
2265         {
2266             AcpiOsPrintf ("Invalid zero length subtable\n");
2267             return 0;
2268         }
2269         goto NextSubTable;

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
2345             /* Attempt to continue */
2346             if (!SubTable->Length)
2347             {
2348                 AcpiOsPrintf ("Invalid zero length subtable\n");
2349                 return;
2350             }
2351             goto NextSubTable;

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 Fri Jan 17 07:29:31 2014
new/usr/src/common/acpica/common/dmtdbinfo.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dmtdbinfo - Table info for non-AML tables
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acdisasm.h"

48 /* This module used for application-level code only */

50 #define _COMPONENT          ACPI_CA_DISASSEMBLER
51 #define ACPI_MODULE_NAME    ("dmtdbinfo")

53 /*
54 * How to add a new table:
55 *
56 * - Add the C table definition to the actbl1.h or actbl2.h header.
57 * - Add ACPI_xxxx_OFFSET macro(s) for the table (and subtables) to list below.
58 * - Define the table in this file (for the disassembler). If any

```

```

59 * new data types are required (ACPI_DMT_*), see below.
60 * - Add an external declaration for the new table definition (AcpiDmTableInfo*)
61 * in acdisasm.h
62 * - Add new table definition to the dispatch table in dmtdb.c (AcpiDmTableDat
63 * If a simple table (with no subtables), no disassembly code is needed.
64 * Otherwise, create the AcpiDmDump* function for to disassemble the table
65 * and add it to the dmtdb.c file.
66 * - Add an external declaration for the new AcpiDmDump* function in acdisasm.h
67 * - Add the new AcpiDmDump* function to the dispatch table in dmtdb.c
68 * - Create a template for the new table
69 * - Add data table compiler support
70 *
71 * How to add a new data type (ACPI_DMT_*):
72 *
73 * - Add new type at the end of the ACPI_DMT list in acdisasm.h
74 * - Add length and implementation cases in dmtdb.c (disassembler)
75 * - Add type and length cases in dtutils.c (DT compiler)
76 */

78 /*
79 * Macros used to generate offsets to specific table fields
80 */
81 #define ACPI_FACS_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_FACS,f)
82 #define ACPI_GAS_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_GENERIC_ADDRE
83 #define ACPI_HDR_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_HEADER,
84 #define ACPI_RSDP_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_RSDP,f)
85 #define ACPI_BERT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_BERT,f)
86 #define ACPI_BGRT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_BGRT,f)
87 #define ACPI_BOOT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_BOOT,f)
88 #define ACPI_CPEP_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_CPEP,f)
89 #define ACPI_DBG2_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_DBG2,f)
90 #define ACPIDBGP_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLEDBGP,f)
91 #define ACPI_DMAR_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_DMAR,f)
92 #define ACPI_DRTM_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_DRTM,f)
93 #define ACPI_ECDT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_ECDT,f)
94 #define ACPI_EINJ_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_EINJ,f)
95 #define ACPI_ERST_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_ERST,f)
96 #define ACPI_GTDT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_GTDT,f)
97 #define ACPI_HEST_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_HEST,f)
98 #define ACPI_HPET_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_HPET,f)
99 #define ACPI_IVRS_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_IVRS,f)
100 #define ACPI_MADT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_MADT,f)
101 #define ACPI_MCFG_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_MCFG,f)
102 #define ACPI_MCHI_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_MCHI,f)
103 #define ACPI_MPST_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_MPST,f)
104 #define ACPI_MSCT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_MSCT,f)
105 #define ACPI_PCCT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_PCCT,f)
106 #define ACPI_PMTT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_PMTT,f)
107 #define ACPI_S3PT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_S3PT,f)
108 #define ACPI_SBST_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_SBST,f)
109 #define ACPI_SLIT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_SLIT,f)
110 #define ACPI_SPCR_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_SPCR,f)
111 #define ACPI_SPMI_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_SPMI,f)
112 #define ACPI_SRAT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_SRAT,f)
113 #define ACPI_TCPA_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_TCPA,f)
114 #define ACPI_TPM2_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_TPM2,f)
115 #define ACPI_UEFI_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_UEFI,f)
116 #define ACPI_WAET_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_WAET,f)
117 #define ACPI_WDAT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_WDAT,f)
118 #define ACPI_WDDT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_WDDT,f)
119 #define ACPI_WDRT_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_TABLE_WDRT,f)

121 /* Subtables */

123 #define ACPIASF0_OFFSET(f) (UINT16) ACPI_OFFSET (ACPIASF_INFO,f)
124 #define ACPIASF1_OFFSET(f) (UINT16) ACPI_OFFSET (ACPIASF_ALERT,f)

```

```

125 #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_DESCRIPTOR
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_FPDTH_HEADER,f
143 #define ACPI_FPDTH0_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_FPDTH_BOOT,f)
144 #define ACPI_FPDTH1_OFFSET(f) (UINT16) ACPI_OFFSET (ACPI_FPDTH_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,f
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_INTERRUPT
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_INTERRUPT
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_COMPONENT
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_MTMRO_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_PMTT_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 Length",
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      AcpiDmTableInfoRsdpl[] =
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 Address",
301     ACPI_DMT_TERMINATOR
302 };

```

```
304 /* ACPI 2.0+ Extensions */
```

```

306 ACPI_DMTABLE_INFO      AcpiDmTableInfoRsd2p[] =
307 {
308     {ACPI_DMT_UINT32,  ACPI_RSDP_OFFSET (Length),            "Length", DT
309     {ACPI_DMT_UINT64,  ACPI_RSDP_OFFSET (XsdtPhysicalAddress), "XSDT Address",
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 Address",
352     {ACPI_DMT_UINT32,  ACPI_FADT_OFFSET (Dsdt),              "DSDT Address",
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 Interrupt",
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 (PmlaEventBlock),     "PMLA Event",
362     {ACPI_DMT_UINT32,  ACPI_FADT_OFFSET (PmlbEventBlock),     "PMLB Event",
363     {ACPI_DMT_UINT32,  ACPI_FADT_OFFSET (PmlaControlBlock),   "PMLA Contro",
364     {ACPI_DMT_UINT32,  ACPI_FADT_OFFSET (PmlbControlBlock),   "PMLB 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 (PmlEventLength),     "PM1 Event B",
370     {ACPI_DMT_UINT8,   ACPI_FADT_OFFSET (PmlControlLength),  "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 0",
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 (XDsdtd),    "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 (EepromAddress),    "Eprom Adre
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    AcpiDmTableInfoBgrrt[] =
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 Adre
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 *
654 *****/

656 /* Main table consists only of the standard ACPI table header */

658 /* Resource Group subtable */

660 ACPI_DMTABLE_INFO      AcpiDmTableInfoCsrt0[] =
661 {
662     {ACPI_DMT_UINT32,    ACPI_CSRT0_OFFSET (Length),        "Length", 0},
663     {ACPI_DMT_UINT32,    ACPI_CSRT0_OFFSET (VendorId),      "Vendor ID",
664     {ACPI_DMT_UINT32,    ACPI_CSRT0_OFFSET (SubvendorId),   "Subvendor I
665     {ACPI_DMT_UINT16,    ACPI_CSRT0_OFFSET (DeviceId),      "Device ID",
666     {ACPI_DMT_UINT16,    ACPI_CSRT0_OFFSET (SubdeviceId),   "Subdevice I
667     {ACPI_DMT_UINT16,    ACPI_CSRT0_OFFSET (Revision),      "Revision",
668     {ACPI_DMT_UINT16,    ACPI_CSRT0_OFFSET (Reserved),       "Reserved",
669     {ACPI_DMT_UINT32,    ACPI_CSRT0_OFFSET (SharedInfoLength), "Shared Info
670     ACPI_DMT_TERMINATOR
671 };

673 /* Shared Info subtable */

675 ACPI_DMTABLE_INFO      AcpiDmTableInfoCsrt1[] =
676 {
677     {ACPI_DMT_UINT16,    ACPI_CSRT1_OFFSET (MajorVersion),  "Major Versi
678     {ACPI_DMT_UINT16,    ACPI_CSRT1_OFFSET (MinorVersion),  "Minor Versi
679     {ACPI_DMT_UINT32,    ACPI_CSRT1_OFFSET (MmioBaseLow),    "MMIO Base A
680     {ACPI_DMT_UINT32,    ACPI_CSRT1_OFFSET (MmioBaseHigh),   "MMIO Base A
681     {ACPI_DMT_UINT32,    ACPI_CSRT1_OFFSET (GsiInterrupt),   "GSI Interru
682     {ACPI_DMT_UINT8,     ACPI_CSRT1_OFFSET (InterruptPolarity), "Interrupt P
683     {ACPI_DMT_UINT8,     ACPI_CSRT1_OFFSET (InterruptMode),   "Interrupt M
684     {ACPI_DMT_UINT8,     ACPI_CSRT1_OFFSET (NumChannels),     "Num Channel
685     {ACPI_DMT_UINT8,     ACPI_CSRT1_OFFSET (DmaAddressWidth), "DMA Address
686     {ACPI_DMT_UINT16,    ACPI_CSRT1_OFFSET (BaseRequestLine), "Base Reques
687     {ACPI_DMT_UINT16,    ACPI_CSRT1_OFFSET (NumHandshakeSignals), "Num Handsha
688     {ACPI_DMT_UINT32,    ACPI_CSRT1_OFFSET (MaxBlockSize),   "Max Block S
689     ACPI_DMT_TERMINATOR
690 };

693 /* Resource Descriptor subtable */

695 ACPI_DMTABLE_INFO      AcpiDmTableInfoCsrt2[] =
696 {
697     {ACPI_DMT_UINT32,    ACPI_CSRT2_OFFSET (Length),        "Length", 0},
698     {ACPI_DMT_UINT16,    ACPI_CSRT2_OFFSET (Type),           "Type", 0},
699     {ACPI_DMT_UINT16,    ACPI_CSRT2_OFFSET (Subtype),        "Subtype", 0
700     {ACPI_DMT_UINT32,    ACPI_CSRT2_OFFSET (Uid),            "UID", 0},
701     ACPI_DMT_TERMINATOR
702 };

705 /*****
706 *
707 * DBG2 - Debug Port Table 2
708 *
709 *****/

711 ACPI_DMTABLE_INFO      AcpiDmTableInfoDbg2[] =
712 {
713     {ACPI_DMT_UINT32,    ACPI_DBG2_OFFSET (InfoOffset),      "Info Offset
714     {ACPI_DMT_UINT32,    ACPI_DBG2_OFFSET (InfoCount),       "Info Count"
715     ACPI_DMT_TERMINATOR
716 };

718 /* Debug Device Information Subtable */

```

```

720 ACPI_DMTABLE_INFO      AcpiDmTableInfoDbg2Device[] =
721 {
722     {ACPI_DMT_UINT8,     ACPI_DBG20_OFFSET (Revision),       "Revision",
723     {ACPI_DMT_UINT16,    ACPI_DBG20_OFFSET (Length),         "Length", DT
724     {ACPI_DMT_UINT8,     ACPI_DBG20_OFFSET (RegisterCount),   "Register Co
725     {ACPI_DMT_UINT16,    ACPI_DBG20_OFFSET (NamepathLength),  "Namepath Le
726     {ACPI_DMT_UINT16,    ACPI_DBG20_OFFSET (NamepathOffset),  "Namepath Of
727     {ACPI_DMT_UINT16,    ACPI_DBG20_OFFSET (OemDataLength),   "OEM Data Le
728     {ACPI_DMT_UINT16,    ACPI_DBG20_OFFSET (OemDataOffset),   "OEM Data Of
729     {ACPI_DMT_UINT16,    ACPI_DBG20_OFFSET (PortType),       "Port Type",
730     {ACPI_DMT_UINT16,    ACPI_DBG20_OFFSET (PortSubtype),    "Port Subtyp
731     {ACPI_DMT_UINT16,    ACPI_DBG20_OFFSET (Reserved),       "Reserved",
732     {ACPI_DMT_UINT16,    ACPI_DBG20_OFFSET (BaseAddressOffset), "Base Address
733     {ACPI_DMT_UINT16,    ACPI_DBG20_OFFSET (AddressSizeOffset), "Address Siz
734     ACPI_DMT_TERMINATOR
735 };

737 /* Variable-length data for the subtable */

739 ACPI_DMTABLE_INFO      AcpiDmTableInfoDbg2Addr[] =
740 {
741     {ACPI_DMT_GAS,       0,                                     "Base Address
742     ACPI_DMT_TERMINATOR
743 };

745 ACPI_DMTABLE_INFO      AcpiDmTableInfoDbg2Size[] =
746 {
747     {ACPI_DMT_UINT32,    0,                                     "Address Siz
748     ACPI_DMT_TERMINATOR
749 };

751 ACPI_DMTABLE_INFO      AcpiDmTableInfoDbg2Name[] =
752 {
753     {ACPI_DMT_STRING,    0,                                     "Namepath",
754     ACPI_DMT_TERMINATOR
755 };

757 ACPI_DMTABLE_INFO      AcpiDmTableInfoDbg2OemData[] =
758 {
759     {ACPI_DMT_BUFFER,    0,                                     "OEM Data",
760     ACPI_DMT_TERMINATOR
761 };

764 /*****
765 *
766 * DBGP - Debug Port
767 *
768 *****/

770 ACPI_DMTABLE_INFO      AcpiDmTableInfoDbgp[] =
771 {
772     {ACPI_DMT_UINT8,     ACPI_DBGP_OFFSET (Type),             "Interface T
773     {ACPI_DMT_UINT24,    ACPI_DBGP_OFFSET (Reserved[0]),      "Reserved",
774     {ACPI_DMT_GAS,       ACPI_DBGP_OFFSET (DebugPort),        "Debug Port
775     ACPI_DMT_TERMINATOR
776 };

779 /*****
780 *
781 * DMAR - DMA Remapping table
782 *
783 *****/

```

```

785 ACPI_DMTABLE_INFO      AcpiDmTableInfoDmar[] =
786 {
787     {ACPI_DMT_UINT8,      ACPI_DMAR_OFFSET (Width),      "Host Address",
788     {ACPI_DMT_UINT8,      ACPI_DMAR_OFFSET (Flags),      "Flags", 0},
789     {ACPI_DMT_UINT10,     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 Length",
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 Address",
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 Address",
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 {
867     ACPI_DMT_TERMINATOR
868 };

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",
910     {ACPI_DMT_UINT8,      ACPI_EINJ0_OFFSET (Reserved), "Reserved",
911     {ACPI_DMT_GAS,        ACPI_EINJ0_OFFSET (RegisterRegion), "Register Re",
912     {ACPI_DMT_UINT64,     ACPI_EINJ0_OFFSET (Value), "Value", 0},
913     {ACPI_DMT_UINT64,     ACPI_EINJ0_OFFSET (Mask), "Mask", 0},
914     ACPI_DMT_TERMINATOR
915 };

```

```

919 /*****
920 *
921 * ERST - Error Record Serialization table
922 *
923 *****/

925 ACPI_DMTABLE_INFO      AcpiDmTableInfoErst[] =
926 {
927     {ACPI_DMT_UINT32,    ACPI_ERST_OFFSET (HeaderLength),    "Serializati",
928     {ACPI_DMT_UINT32,    ACPI_ERST_OFFSET (Reserved),        "Reserved",
929     {ACPI_DMT_UINT32,    ACPI_ERST_OFFSET (Entries),        "Instruction",
930     ACPI_DMT_TERMINATOR
931 };

933 ACPI_DMTABLE_INFO      AcpiDmTableInfoErst0[] =
934 {
935     {ACPI_DMT_ERSTACT,   ACPI_ERST0_OFFSET (Action),        "Action", 0},
936     {ACPI_DMT_ERSTINST,  ACPI_ERST0_OFFSET (Instruction),   "Instruction",
937     {ACPI_DMT_UINT8,     ACPI_ERST0_OFFSET (Flags),        "Flags (deco",
938     {ACPI_DMT_FLAG0,     ACPI_ERST0_FLAG_OFFSET (Flags,0),  "Preserve Re

940     {ACPI_DMT_UINT8,     ACPI_ERST0_OFFSET (Reserved),      "Reserved",
941     {ACPI_DMT_GAS,       ACPI_ERST0_OFFSET (RegisterRegion), "Register Re",
942     {ACPI_DMT_UINT64,    ACPI_ERST0_OFFSET (Value),         "Value", 0},
943     {ACPI_DMT_UINT64,    ACPI_ERST0_OFFSET (Mask),          "Mask", 0},
944     ACPI_DMT_TERMINATOR
945 };

948 /*****
949 *
950 * FPDТ - Firmware Performance Data Table (ACPI 5.0)
951 *
952 *****/

954 /* Main table consists of only the standard ACPI header - subtables follow */

956 /* FPDТ subtable header */

958 ACPI_DMTABLE_INFO      AcpiDmTableInfoFpdtHdr[] =
959 {
960     {ACPI_DMT_UINT16,    ACPI_FPDТH_OFFSET (Type),          "Subtable Ty",
961     {ACPI_DMT_UINT8,     ACPI_FPDТH_OFFSET (Length),        "Length", DT
962     {ACPI_DMT_UINT8,     ACPI_FPDТH_OFFSET (Revision),      "Revision",
963     ACPI_DMT_TERMINATOR
964 };

966 /* 0: Firmware Basic Boot Performance Record */

968 ACPI_DMTABLE_INFO      AcpiDmTableInfoFpdt0[] =
969 {
970     {ACPI_DMT_UINT32,    ACPI_FPDТ0_OFFSET (Reserved),      "Reserved",
971     {ACPI_DMT_UINT64,    ACPI_FPDТ0_OFFSET (ResetEnd),      "Reset End",
972     {ACPI_DMT_UINT64,    ACPI_FPDТ0_OFFSET (LoadStart),      "Load Image",
973     {ACPI_DMT_UINT64,    ACPI_FPDТ0_OFFSET (StartupStart),   "Start Image",
974     {ACPI_DMT_UINT64,    ACPI_FPDТ0_OFFSET (ExitServicesEntry), "Exit Servic",
975     {ACPI_DMT_UINT64,    ACPI_FPDТ0_OFFSET (ExitServicesExit), "Exit Servic",
976     ACPI_DMT_TERMINATOR
977 };

979 /* 1: S3 Performance Table Pointer Record */

981 ACPI_DMTABLE_INFO      AcpiDmTableInfoFpdt1[] =
982 {

```

```

983     {ACPI_DMT_UINT32,    ACPI_FPDТ1_OFFSET (Reserved),      "Reserved",
984     {ACPI_DMT_UINT64,    ACPI_FPDТ1_OFFSET (Address),       "S3PT Address",
985     ACPI_DMT_TERMINATOR
986 };

989 /*****
990 *
991 * GTDT - Generic Timer Description Table
992 *
993 *****/

995 ACPI_DMTABLE_INFO      AcpiDmTableInfoGtdt[] =
996 {
997     {ACPI_DMT_UINT64,    ACPI_GTDT_OFFSET (Address),        "Timer Addre",
998     {ACPI_DMT_UINT32,    ACPI_GTDT_OFFSET (Flags),          "Flags (deco",
999     {ACPI_DMT_FLAG0,     ACPI_GTDT_FLAG_OFFSET (Flags,0),    "Memory Pres",
1000     ACPI_DMT_NEW_LINE,
1001     {ACPI_DMT_UINT32,    ACPI_GTDT_OFFSET (SecurePllInterrupt), "Secure Pll",
1002     {ACPI_DMT_UINT32,    ACPI_GTDT_OFFSET (SecurePllFlags),   "SPL1 Flags",
1003     {ACPI_DMT_FLAG0,     ACPI_GTDT_FLAG_OFFSET (SecurePllFlags,0), "Trigger Mod",
1004     {ACPI_DMT_FLAG1,     ACPI_GTDT_FLAG_OFFSET (SecurePllFlags,0), "Polarity",
1005     ACPI_DMT_NEW_LINE,
1006     {ACPI_DMT_UINT32,    ACPI_GTDT_OFFSET (NonSecurePllInterrupt), "Non-Secure",
1007     {ACPI_DMT_UINT32,    ACPI_GTDT_OFFSET (NonSecurePllFlags),   "NSPL1 Flags",
1008     {ACPI_DMT_FLAG0,     ACPI_GTDT_FLAG_OFFSET (NonSecurePllFlags,0), "Trigger Mod",
1009     {ACPI_DMT_FLAG1,     ACPI_GTDT_FLAG_OFFSET (NonSecurePllFlags,0), "Polarity",
1010     ACPI_DMT_NEW_LINE,
1011     {ACPI_DMT_UINT32,    ACPI_GTDT_OFFSET (VirtualTimerInterrupt), "Virtual Tim",
1012     {ACPI_DMT_UINT32,    ACPI_GTDT_OFFSET (VirtualTimerFlags),   "VT Flags (d",
1013     {ACPI_DMT_FLAG0,     ACPI_GTDT_FLAG_OFFSET (VirtualTimerFlags,0), "Trigger Mod",
1014     {ACPI_DMT_FLAG1,     ACPI_GTDT_FLAG_OFFSET (VirtualTimerFlags,0), "Polarity",
1015     ACPI_DMT_NEW_LINE,
1016     {ACPI_DMT_UINT32,    ACPI_GTDT_OFFSET (NonSecurePl2Interrupt), "Non-Secure",
1017     {ACPI_DMT_UINT32,    ACPI_GTDT_OFFSET (NonSecurePl2Flags),   "NSPL2 Flags",
1018     {ACPI_DMT_FLAG0,     ACPI_GTDT_FLAG_OFFSET (NonSecurePl2Flags,0), "Trigger Mod",
1019     {ACPI_DMT_FLAG1,     ACPI_GTDT_FLAG_OFFSET (NonSecurePl2Flags,0), "Polarity",
1020     ACPI_DMT_TERMINATOR
1021 };

1024 /*****
1025 *
1026 * HEST - Hardware Error Source table
1027 *
1028 *****/

1030 ACPI_DMTABLE_INFO      AcpiDmTableInfoHest[] =
1031 {
1032     {ACPI_DMT_UINT32,    ACPI_HEST_OFFSET (ErrorSourceCount), "Error Sourc",
1033     ACPI_DMT_TERMINATOR
1034 };

1036 /* Common HEST structures for subtables */

1038 #define ACPI_DM_HEST_HEADER \
1039     {ACPI_DMT_HEST,      ACPI_HEST0_OFFSET (Header.Type),    "Subtable Ty",
1040     {ACPI_DMT_UINT16,    ACPI_HEST0_OFFSET (Header.SourceId), "Source Id",

1042 #define ACPI_DM_HEST_AER \
1043     {ACPI_DMT_UINT16,    ACPI_HEST6_OFFSET (Aer.Reserved1),   "Reserve",
1044     {ACPI_DMT_UINT8,     ACPI_HEST6_OFFSET (Aer.Flags),        "Flags (",
1045     {ACPI_DMT_FLAG0,     ACPI_HEST6_FLAG_OFFSET (Aer.Flags,0), "Firmwar",
1046     {ACPI_DMT_UINT8,     ACPI_HEST6_OFFSET (Aer.Enabled),     "Enabled",
1047     {ACPI_DMT_UINT32,    ACPI_HEST6_OFFSET (Aer.RecordsToPreallocate), "Records",
1048     {ACPI_DMT_UINT32,    ACPI_HEST6_OFFSET (Aer.MaxSectionsPerRecord), "Max Sec

```

```

1049 {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), "Function"
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_HESTNTFY, 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_UINT8,    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 Address
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 Address",
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 Address
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 Address
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 Address
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_MTMRO_OFFSET (PhysicalAddress),      "PhysicalAdd
1670     {ACPI_DMT_UINT32,      ACPI_MTMRO_OFFSET (Frequency),      "Frequency",
1671     {ACPI_DMT_UINT32,      ACPI_MTMRO_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 Address
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",
1842 {ACPI_DMT_UINT32, ACPI_SBST_OFFSET (CriticalLevel), "Critical Le
1843 ACPI_DMT_TERMINATOR
1844 };

1847 /*****
1848 *
1849 * SLIC - Software Licensing Description Table. There is no common table, just
1850 * the standard ACPI header and then subtables.
1851 *
1852 *****/

1854 /* Common Subtable header (one per Subtable) */

1856 ACPI_DMTABLE_INFO AcpiDmTableInfoSlicHdr[] =
1857 {
1858 {ACPI_DMT_SLIC, ACPI_SLICH_OFFSET (Type), "Subtable Ty
1859 {ACPI_DMT_UINT32, ACPI_SLICH_OFFSET (Length), "Length", DT
1860 ACPI_DMT_TERMINATOR
1861 };

1863 ACPI_DMTABLE_INFO AcpiDmTableInfoSlic0[] =
1864 {
1865 {ACPI_DMT_UINT8, ACPI_SLIC0_OFFSET (KeyType), "Key Type",
1866 {ACPI_DMT_UINT8, ACPI_SLIC0_OFFSET (Version), "Version", 0
1867 {ACPI_DMT_UINT16, ACPI_SLIC0_OFFSET (Reserved), "Reserved",
1868 {ACPI_DMT_UINT32, ACPI_SLIC0_OFFSET (Algorithm), "Algorithm",
1869 {ACPI_DMT_NAME4, ACPI_SLIC0_OFFSET (Magic), "Magic", 0},
1870 {ACPI_DMT_UINT32, ACPI_SLIC0_OFFSET (BitLength), "BitLength",
1871 {ACPI_DMT_UINT32, ACPI_SLIC0_OFFSET (Exponent), "Exponent",
1872 {ACPI_DMT_BUF128, ACPI_SLIC0_OFFSET (Modulus[0]), "Modulus", 0
1873 ACPI_DMT_TERMINATOR
1874 };

1876 ACPI_DMTABLE_INFO AcpiDmTableInfoSlic1[] =
1877 {
1878 {ACPI_DMT_UINT32, ACPI_SLIC1_OFFSET (Version), "Version", 0
1879 {ACPI_DMT_NAME6, ACPI_SLIC1_OFFSET (OemId[0]), "Oem ID", 0}
1880 {ACPI_DMT_NAME8, ACPI_SLIC1_OFFSET (OemTableId[0]), "Oem Table I
1881 {ACPI_DMT_NAME8, ACPI_SLIC1_OFFSET (WindowsFlag[0]), "Windows Fla
1882 {ACPI_DMT_UINT32, ACPI_SLIC1_OFFSET (SlicVersion), "SLIC Versio
1883 {ACPI_DMT_BUF16, ACPI_SLIC1_OFFSET (Reserved[0]), "Reserved",
1884 {ACPI_DMT_BUF128, ACPI_SLIC1_OFFSET (Signature[0]), "Signature",
1885 ACPI_DMT_TERMINATOR
1886 };

1889 /*****
1890 *
1891 * SLIT - System Locality Information Table
1892 *
1893 *****/

1895 ACPI_DMTABLE_INFO AcpiDmTableInfoSlit[] =
1896 {
1897 {ACPI_DMT_UINT64, ACPI_SLIT_OFFSET (LocalityCount), "Localities",
1898 ACPI_DMT_TERMINATOR
1899 };

1902 /*****
1903 *
1904 * SPCR - Serial Port Console Redirection table
1905 *
1906 *****/

```

```

1908 ACPI_DMTABLE_INFO AcpiDmTableInfoSpcr[] =
1909 {
1910 {ACPI_DMT_UINT8, ACPI_SPCR_OFFSET (InterfaceType), "Interface T
1911 {ACPI_DMT_UINT24, ACPI_SPCR_OFFSET (Reserved[0]), "Reserved",
1912 {ACPI_DMT_GAS, ACPI_SPCR_OFFSET (SerialPort), "Serial Port
1913 {ACPI_DMT_UINT8, ACPI_SPCR_OFFSET (InterruptType), "Interrupt T
1914 {ACPI_DMT_UINT8, ACPI_SPCR_OFFSET (PcInterrupt), "PCAT-compat
1915 {ACPI_DMT_UINT32, ACPI_SPCR_OFFSET (Interrupt), "Interrupt",
1916 {ACPI_DMT_UINT8, ACPI_SPCR_OFFSET (BaudRate), "Baud Rate",
1917 {ACPI_DMT_UINT8, ACPI_SPCR_OFFSET (Parity), "Parity", 0}
1918 {ACPI_DMT_UINT8, ACPI_SPCR_OFFSET (StopBits), "Stop Bits",
1919 {ACPI_DMT_UINT8, ACPI_SPCR_OFFSET (FlowControl), "Flow Contro
1920 {ACPI_DMT_UINT8, ACPI_SPCR_OFFSET (TerminalType), "Terminal Ty
1921 {ACPI_DMT_UINT8, ACPI_SPCR_OFFSET (Reserved2), "Reserved",
1922 {ACPI_DMT_UINT16, ACPI_SPCR_OFFSET (PciDeviceId), "PCI Device
1923 {ACPI_DMT_UINT16, ACPI_SPCR_OFFSET (PciVendorId), "PCI Vendor
1924 {ACPI_DMT_UINT8, ACPI_SPCR_OFFSET (PciBus), "PCI Bus", 0
1925 {ACPI_DMT_UINT8, ACPI_SPCR_OFFSET (PciDevice), "PCI Device"
1926 {ACPI_DMT_UINT8, ACPI_SPCR_OFFSET (PciFunction), "PCI Functio
1927 {ACPI_DMT_UINT32, ACPI_SPCR_OFFSET (PciFlags), "PCI Flags",
1928 {ACPI_DMT_UINT8, ACPI_SPCR_OFFSET (PciSegment), "PCI Segment
1929 {ACPI_DMT_UINT32, ACPI_SPCR_OFFSET (Reserved2), "Reserved",
1930 ACPI_DMT_TERMINATOR
1931 };

1934 /*****
1935 *
1936 * SPMI - Server Platform Management Interface table
1937 *
1938 *****/

1940 ACPI_DMTABLE_INFO AcpiDmTableInfoSpmi[] =
1941 {
1942 {ACPI_DMT_UINT8, ACPI_SPMI_OFFSET (InterfaceType), "Interface T
1943 {ACPI_DMT_UINT8, ACPI_SPMI_OFFSET (Reserved), "Reserved",
1944 {ACPI_DMT_UINT16, ACPI_SPMI_OFFSET (SpecRevision), "IPMI Spec V
1945 {ACPI_DMT_UINT8, ACPI_SPMI_OFFSET (InterruptType), "Interrupt T
1946 {ACPI_DMT_UINT8, ACPI_SPMI_OFFSET (GpeNumber), "GPE Number"
1947 {ACPI_DMT_UINT8, ACPI_SPMI_OFFSET (Reserved1), "Reserved",
1948 {ACPI_DMT_UINT8, ACPI_SPMI_OFFSET (PciDeviceFlag), "PCI Device
1949 {ACPI_DMT_UINT32, ACPI_SPMI_OFFSET (Interrupt), "Interrupt",
1950 {ACPI_DMT_GAS, ACPI_SPMI_OFFSET (IpmiRegister), "IPMI Regist
1951 {ACPI_DMT_UINT8, ACPI_SPMI_OFFSET (PciSegment), "PCI Segment
1952 {ACPI_DMT_UINT8, ACPI_SPMI_OFFSET (PciBus), "PCI Bus", 0
1953 {ACPI_DMT_UINT8, ACPI_SPMI_OFFSET (PciDevice), "PCI Device"
1954 {ACPI_DMT_UINT8, ACPI_SPMI_OFFSET (PciFunction), "PCI Functio
1955 {ACPI_DMT_UINT8, ACPI_SPMI_OFFSET (Reserved2), "Reserved",
1956 ACPI_DMT_TERMINATOR
1957 };

1960 /*****
1961 *
1962 * SRAT - System Resource Affinity Table and Subtables
1963 *
1964 *****/

1966 ACPI_DMTABLE_INFO AcpiDmTableInfoSrat[] =
1967 {
1968 {ACPI_DMT_UINT32, ACPI_SRAT_OFFSET (TableRevision), "Table Revis
1969 {ACPI_DMT_UINT64, ACPI_SRAT_OFFSET (Reserved), "Reserved",
1970 ACPI_DMT_TERMINATOR
1971 };

```

```

1973 /* 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 Address
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 * TPCA - 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 !*/
```

new/usr/src/common/acpica/common/getopt.c

1

```
*****
7805 Fri Jan 17 07:29:31 2014
new/usr/src/common/acpica/common/getopt.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: getopt
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 * ACPICA getopt() implementation
46 *
47 * Option strings:
48 * "f" - Option has no arguments
49 * "f:" - Option requires an argument
50 * "f^" - Option has optional single-char sub-options
51 * "f|" - Option has required single-char sub-options
52 */

54 #include <stdio.h>
55 #include <string.h>
56 #include "acpi.h"
57 #include "accommon.h"
58 #include "acapps.h"
```

new/usr/src/common/acpica/common/getopt.c

2

```
60 #define ACPI_OPTION_ERROR(msg, badchar) \
61     if (AcpiGbl_Opterr) {fprintf (stderr, "%s%c\n", msg, badchar);}

64 int          AcpiGbl_Opterr = 1;
65 int          AcpiGbl_Optind = 1;
66 int          AcpiGbl_SubOptChar = 0;
67 char         *AcpiGbl_Optarg;

69 static int   CurrentCharPtr = 1;

72 /*****
73 *
74 * FUNCTION:   AcpiGetoptArgument
75 *
76 * PARAMETERS: argc, argv          - from main
77 *
78 * RETURN:    0 if an argument was found, -1 otherwise. Sets AcpiGbl_Optarg
79 *            to point to the next argument.
80 *
81 * DESCRIPTION: Get the next argument. Used to obtain arguments for the
82 *               two-character options after the original call to AcpiGetopt.
83 *               Note: Either the argument starts at the next character after
84 *               the option, or it is pointed to by the next argv entry.
85 *               (After call to AcpiGetopt, we need to backup to the previous
86 *               argv entry).
87 *
88 *****/

90 int
91 AcpiGetoptArgument (
92     int          argc,
93     char         **argv)
94 {
95     AcpiGbl_Optind--;
96     CurrentCharPtr++;

98     if (argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)] != '\0')
99     {
100         AcpiGbl_Optarg = &argv[AcpiGbl_Optind+1][(int) (CurrentCharPtr+1)];
101     }
102     else if (++AcpiGbl_Optind >= argc)
103     {
104         ACPI_OPTION_ERROR ("Option requires an argument: -, 'v'");

106         CurrentCharPtr = 1;
107         return (-1);
108     }
109     else
110     {
111         AcpiGbl_Optarg = argv[AcpiGbl_Optind+1];
112     }

114     CurrentCharPtr = 1;
115     return (0);
116 }

119 /*****
120 *
121 * FUNCTION:   AcpiGetopt
122 *
123 * PARAMETERS: argc, argv          - from main
124 *             opts                 - options info list
125 *
```



```

126 * RETURN:      Option character or EOF
127 *
128 * DESCRIPTION: Get the next option
129 *
130 *****/

132 int
133 AcpiGetopt(
134     int          argc,
135     char          **argv,
136     char          *opts)
137 {
138     int          CurrentChar;
139     char          *OptsPtr;

142     if (CurrentCharPtr == 1)
143     {
144         if (AcpiGbl_Optind >= argc ||
145             argv[AcpiGbl_Optind][0] != '-' ||
146             argv[AcpiGbl_Optind][1] == '\0')
147         {
148             return (EOF);
149         }
150         else if (strcmp (argv[AcpiGbl_Optind], "--") == 0)
151         {
152             AcpiGbl_Optind++;
153             return (EOF);
154         }
155     }

157     /* Get the option */

159     CurrentChar = argv[AcpiGbl_Optind][CurrentCharPtr];

161     /* Make sure that the option is legal */

163     if (CurrentChar == ':' ||
164         (OptsPtr = strchr (opts, CurrentChar)) == NULL)
165     {
166         ACPI_OPTION_ERROR ("Illegal option: -", CurrentChar);

168         if (argv[AcpiGbl_Optind][++CurrentCharPtr] == '\0')
169         {
170             AcpiGbl_Optind++;
171             CurrentCharPtr = 1;
172         }

174         return ('?');
175     }

177     /* Option requires an argument? */

179     if (++OptsPtr == ':')
180     {
181         if (argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)] != '\0')
182         {
183             AcpiGbl_Optarg = &argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)];
184         }
185         else if (++AcpiGbl_Optind >= argc)
186         {
187             ACPI_OPTION_ERROR ("Option requires an argument: -", CurrentChar);

189             CurrentCharPtr = 1;
190             return ('?');
191         }

```

```

192     else
193     {
194         AcpiGbl_Optarg = argv[AcpiGbl_Optind++];
195     }

197     CurrentCharPtr = 1;
198 }

200 /* Option has an optional argument? */

202 else if (*OptsPtr == '+')
203 {
204     if (argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)] != '\0')
205     {
206         AcpiGbl_Optarg = &argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)];
207     }
208     else if (++AcpiGbl_Optind >= argc)
209     {
210         AcpiGbl_Optarg = NULL;
211     }
212     else
213     {
214         AcpiGbl_Optarg = argv[AcpiGbl_Optind++];
215     }

217     CurrentCharPtr = 1;
218 }

220 /* Option has optional single-char arguments? */

222 else if (*OptsPtr == '^')
223 {
224     if (argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)] != '\0')
225     {
226         AcpiGbl_Optarg = &argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)];
227     }
228     else
229     {
230         AcpiGbl_Optarg = "^";
231     }

233     AcpiGbl_SubOptChar = AcpiGbl_Optarg[0];
234     AcpiGbl_Optind++;
235     CurrentCharPtr = 1;
236 }

238 /* Option has a required single-char argument? */

240 else if (*OptsPtr == '|')
241 {
242     if (argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)] != '\0')
243     {
244         AcpiGbl_Optarg = &argv[AcpiGbl_Optind][(int) (CurrentCharPtr+1)];
245     }
246     else
247     {
248         ACPI_OPTION_ERROR ("Option requires a single-character suboption: -"

250         CurrentCharPtr = 1;
251         return ('?');
252     }

254     AcpiGbl_SubOptChar = AcpiGbl_Optarg[0];
255     AcpiGbl_Optind++;
256     CurrentCharPtr = 1;
257 }

```

```
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 }
```

new/usr/src/common/acpica/compiler/aslanalyze.c

1

```
*****
15585 Fri Jan 17 07:29:31 2014
new/usr/src/common/acpica/compiler/aslanalyze.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslanalyze.c - Support functions for parse tree walks
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "aslcompiler.h"
45 #include "aslcompiler.y.h"
46 #include <string.h>
47
48 #define _COMPONENT          ACPI_COMPILER
49 #define ACPI_MODULE_NAME    ("aslanalyze")
50
51 /*****
52 *
53 * FUNCTION:      AnIsInternalMethod
54 *
55 * PARAMETERS:    Op
56 *                - Current op
57 *
58 *****/
```

new/usr/src/common/acpica/compiler/aslanalyze.c

2

```
60 * RETURN:      Boolean
61 *
62 * DESCRIPTION:  Check for an internal control method.
63 *
64 *****/
65
66 BOOLEAN
67 AnIsInternalMethod (
68     ACPI_PARSE_OBJECT *Op)
69 {
70     if ((!ACPI_STRCMP (Op->Asl.ExternalName, "\\_OSI")) ||
71         (!ACPI_STRCMP (Op->Asl.ExternalName, "_OSI")))
72     {
73         return (TRUE);
74     }
75
76     return (FALSE);
77 }
78
79 /*****
80 *
81 * FUNCTION:      AnGetInternalMethodReturnType
82 *
83 * PARAMETERS:    Op
84 *                - Current op
85 *
86 * RETURN:        Btype
87 *
88 * DESCRIPTION:   Get the return type of an internal method
89 *
90 *****/
91
92 UINT32
93 AnGetInternalMethodReturnType (
94     ACPI_PARSE_OBJECT *Op)
95 {
96     if ((!ACPI_STRCMP (Op->Asl.ExternalName, "\\_OSI")) ||
97         (!ACPI_STRCMP (Op->Asl.ExternalName, "_OSI")))
98     {
99         return (ACPI_BTTYPE_STRING);
100     }
101
102     return (0);
103 }
104
105 /*****
106 *
107 * FUNCTION:      AnCheckId
108 *
109 * PARAMETERS:    Op
110 *                - Current parse op
111 *                Type
112 *                - HID or CID
113 *
114 * RETURN:        None
115 *
116 * DESCRIPTION:   Perform various checks on _HID and _CID strings. Only limited
117 *                checks can be performed on _CID strings.
118 *
119 *****/
120
121 void
122 AnCheckId (
123     ACPI_PARSE_OBJECT *Op,
124     ACPI_NAME Type)
```

```

126 {
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 (
259     ACPI_PARSE_OBJECT *Op)
260 {
261     ACPI_PARSE_OBJECT *Next;

264     /* Check if last statement is a return */

266     Next = ASL_GET_CHILD_NODE (Op);
267     while (Next)
268     {
269         if ((!Next->Asl.Next) &&
270             (Next->Asl.ParseOpcode == PARSEOP_RETURN))
271         {
272             return (TRUE);
273         }

275         Next = ASL_GET_PEER_NODE (Next);
276     }

278     return (FALSE);
279 }

282 /*****
283 * FUNCTION:      AnCheckMethodReturnValue
284 *
285 * PARAMETERS:    Op          - Parent
286 *                OpInfo      - Parent info
287 *                ArgOp        - Method invocation op
288 *                RequiredBtypes - What caller requires
289 *                ThisNodeBtype - What this node returns (if anything)
290 *
291 * RETURN:        None
292 *
293 * DESCRIPTION:    Check a method invocation for 1) A return value and if it does
294 *                 in fact return a value, 2) check the type of the return value.
295 *
296 * *****/
297 void
298 AnCheckMethodReturnValue (
299     ACPI_PARSE_OBJECT *Op,
300     const ACPI_OPCODE_INFO *OpInfo,
301     ACPI_PARSE_OBJECT *ArgOp,
302     UINT32 RequiredBtypes,
303     UINT32 ThisNodeBtype)
304 {
305     ACPI_PARSE_OBJECT *OwningOp;
306     ACPI_NAMESPACE_NODE *Node;

311     Node = ArgOp->Asl.Node;

314     /* Examine the parent op of this method */

316     OwningOp = Node->Op;
317     if (OwningOp->Asl.CompileFlags & NODE_METHOD_NO_RETVAL)
318     {
319         /* Method NEVER returns a value */

321         AslError (ASL_ERROR, ASL_MSG_NO_RETVAL, Op, Op->Asl.ExternalName);
322     }
323     else if (OwningOp->Asl.CompileFlags & NODE_METHOD_SOME_NO_RETVAL)

```

```

324     {
325         /* Method SOMETIMES returns a value, SOMETIMES not */

327         AslError (ASL_WARNING, ASL_MSG_SOME_NO_RETVAL, Op, Op->Asl.ExternalName)
328     }
329     else if (!(ThisNodeBtype & RequiredBtypes))
330     {
331         /* Method returns a value, but the type is wrong */

333         AnFormatBtype (StringBuffer, ThisNodeBtype);
334         AnFormatBtype (StringBuffer2, RequiredBtypes);

336         /*
337          * The case where the method does not return any value at all
338          * was already handled in the namespace cross reference
339          * -- Only issue an error if the method in fact returns a value,
340          * but it is of the wrong type
341          */
342         if (ThisNodeBtype != 0)
343         {
344             sprintf (MsgBuffer,
345                 "Method returns [%s], %s operator requires [%s]",
346                 StringBuffer, OpInfo->Name, StringBuffer2);

348             AslError (ASL_ERROR, ASL_MSG_INVALID_TYPE, ArgOp, MsgBuffer);
349         }
350     }
351 }

354 /*****
355 * FUNCTION:      AnIsResultUsed
356 *
357 * PARAMETERS:    Op          - Parent op for the operator
358 *
359 * RETURN:        TRUE if result from this operation is actually consumed
360 *
361 * DESCRIPTION:    Determine if the function result value from an operator is
362 *                 used.
363 *
364 * *****/
365 BOOLEAN
366 AnIsResultUsed (
367     ACPI_PARSE_OBJECT *Op)
368 {
369     ACPI_PARSE_OBJECT *Parent;

371     switch (Op->Asl.ParseOpcode)
372     {
373     case PARSEOP_INCREMENT:
374     case PARSEOP_DECREMENT:

377         /* These are standalone operators, no return value */

379         return (TRUE);

381     default:

383         break;
384     }

385     /* Examine parent to determine if the return value is used */

```

```

390     Parent = Op->Asl.Parent;
391     switch (Parent->Asl.ParseOpcode)
392     {
393     /* If/While - check if the operator is the predicate */
394
395     case PARSEOP_IF:
396     case PARSEOP_WHILE:
397
398         /* First child is the predicate */
399
400         if (Parent->Asl.Child == Op)
401         {
402             return (TRUE);
403         }
404         return (FALSE);
405
406     /* Not used if one of these is the parent */
407
408     case PARSEOP_METHOD:
409     case PARSEOP_DEFINITIONBLOCK:
410     case PARSEOP_ELSE:
411
412         return (FALSE);
413
414     default:
415
416         /* Any other type of parent means that the result is used */
417
418         return (TRUE);
419     }
420 }
421
422 /*****
423  *
424  * FUNCTION:      ApCheckForGpeNameConflict
425  *
426  * PARAMETERS:    Op                - Current parse op
427  *
428  * RETURN:        None
429  *
430  * DESCRIPTION:   Check for a conflict between GPE names within this scope.
431  *                Conflict means two GPE names with the same GPE number, but
432  *                different types -- such as _L1C and _E1C.
433  *
434  *
435  *****/
436
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];
445
446     /* Need a null-terminated string version of NameSeg */
447
448     ACPI_MOVE_32_TO_32 (Name, &Op->Asl.NameSeg);
449     Name[ACPI_NAME_SIZE] = 0;
450
451     /*
452     * For a GPE method:
453     * 1st char must be underscore
454     * 2nd char must be L or E
455     */

```

```

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     }
463
464     /* Verify 3rd/4th chars are a valid hex value */
465
466     GpeNumber = ACPI_STRTOUL (&Name[2], NULL, 16);
467     if (GpeNumber == ACPI_UINT32_MAX)
468     {
469         return;
470     }
471
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);
477
478     /* Inject opposite letter ("L" versus "E") */
479
480     if (Name[1] == 'L')
481     {
482         Target[1] = 'E';
483     }
484     else /* Name[1] == 'E' */
485     {
486         Target[1] = 'L';
487     }
488
489     /* Search all peers (objects within this scope) for target match */
490
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 */
505
506                     AslError (ASL_ERROR, ASL_MSG_GPE_NAME_CONFLICT, NextOp,
507                             Name);
508                     return;
509                 }
510             }
511
512             NextOp = NextOp->Asl.Next;
513         }
514
515     /* OK, no conflict found */
516
517     return;
518 }
519
520 /*****

```

```
522 *
523 * FUNCTION:    ApCheckRegMethod
524 *
525 * PARAMETERS:  Op                - Current parse op
526 *
527 * RETURN:      None
528 *
529 * DESCRIPTION: Ensure that a _REG method has a corresponding Operation
530 *              Region declaration within the same scope. Note: _REG is defined
531 *              to have two arguments and must therefore be defined as a
532 *              control method.
533 *
534 *****/

536 void
537 ApCheckRegMethod (
538     ACPI_PARSE_OBJECT *Op)
539 {
540     ACPI_PARSE_OBJECT *Next;
541     ACPI_PARSE_OBJECT *Parent;

544     /* We are only interested in _REG methods */

546     if (!ACPI_COMPARE_NAME (METHOD_NAME__REG, &Op->Asl.NameSeg))
547     {
548         return;
549     }

551     /* Get the start of the current scope */

553     Parent = Op->Asl.Parent;
554     Next = Parent->Asl.Child;

556     /* Search entire scope for an operation region declaration */

558     while (Next)
559     {
560         if (Next->Asl.ParseOpcode == PARSEOP_OPERATIONREGION)
561         {
562             return; /* Found region, OK */
563         }

565         Next = Next->Asl.Next;
566     }

568     /* No region found, issue warning */

570     AslError (ASL_WARNING, ASL_MSG_NO_REGION, Op, NULL);
571 }
```

new/usr/src/common/acpica/compiler/aslbtypes.c

1

```
*****
14222 Fri Jan 17 07:29:32 2014
new/usr/src/common/acpica/compiler/aslbtypes.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: aslbtypes - Support for bitfield types
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "aslcompiler.h"
45 #include "aslcompiler.y.h"
46 #include "amlcode.h"
47
48 #define _COMPONENT          ACPI_COMPILER
49 #define ACPI_MODULE_NAME    ("aslbtypes")
50
51 /* Local prototypes */
52
53 static UINT32
54 AnMapEtypeToBtype (
55     UINT32                Etype);
```

new/usr/src/common/acpica/compiler/aslbtypes.c

2

```
60 /*****
61 *
62 * FUNCTION:      AnMapArgTypeToBtype
63 *
64 * PARAMETERS:    ArgType          - The ARGUMENT required type(s) for this
65 *                                argument, from the opcode info table
66 *
67 * RETURN:        The corresponding Bit-encoded types
68 *
69 * DESCRIPTION:    Convert an encoded ARGUMENT required argument type code into a
70 *                bitfield type code. Implements the implicit source conversion
71 *                rules.
72 *
73 *****/
74
75 UINT32
76 AnMapArgTypeToBtype (
77     UINT32                ArgType)
78 {
79     switch (ArgType)
80     {
81         /* Simple types */
82
83         case ARGUMENT_ANYTYPE:
84
85             return (ACPI_BTYPE_OBJECTS_AND_REFS);
86
87         case ARGUMENT_PACKAGE:
88
89             return (ACPI_BTYPE_PACKAGE);
90
91         case ARGUMENT_EVENT:
92
93             return (ACPI_BTYPE_EVENT);
94
95         case ARGUMENT_MUTEX:
96
97             return (ACPI_BTYPE_MUTEX);
98
99         case ARGUMENT_DDBHANDLE:
100             /*
101              * DDBHandleObject := SuperName
102              * ACPI_BTYPE_REFERENCE: Index reference as parameter of Load/Unload
103              */
104             return (ACPI_BTYPE_DDB_HANDLE | ACPI_BTYPE_REFERENCE);
105
106         /* Interchangeable types */
107         /*
108          * Source conversion rules:
109          * Integer, String, and Buffer are all interchangeable
110          */
111         case ARGUMENT_INTEGER:
112         case ARGUMENT_STRING:
113         case ARGUMENT_BUFFER:
114         case ARGUMENT_BUFFER_OR_STRING:
115         case ARGUMENT_COMPUTEDDATA:
116
117             return (ACPI_BTYPE_COMPUTE_DATA);
118
119         /* References */
120
121         case ARGUMENT_INTEGER_REF:
```



```

125     return (ACPI_BTYPE_INTEGER);

127     case ARG1_OBJECT_REF:

129         return (ACPI_BTYPE_ALL_OBJECTS);

131     case ARG1_DEVICE_REF:

133         return (ACPI_BTYPE_DEVICE_OBJECTS);

135     case ARG1_REFERENCE:

137         return (ACPI_BTYPE_REFERENCE);

139     case ARG1_TARGETREF:
140     case ARG1_FIXED_TARGET:
141     case ARG1_SIMPLE_TARGET:

143         return (ACPI_BTYPE_OBJECTS_AND_REFS);

145     /* Complex types */

147     case ARG1_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 ARG1_COMPLEXOBJ:

157         /* Buffer, String, or package */

159         return (ACPI_BTYPE_STRING | ACPI_BTYPE_BUFFER | ACPI_BTYPE_PACKAGE);

161     case ARG1_REF_OR_STRING:

163         return (ACPI_BTYPE_STRING | ACPI_BTYPE_REFERENCE);

165     case ARG1_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 ARG1_DATAAREFOBJ:

173         return (ACPI_BTYPE_INTEGER | ACPI_BTYPE_STRING | ACPI_BTYPE_BUFFER |
174                 ACPI_BTYPE_PACKAGE | ACPI_BTYPE_REFERENCE | ACPI_BTYPE_DDB_HANDLE);

176     default:

177         break;
178     }

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_BTTYPE_COMPUTE_DATA | ACPI_BTTYPE_FIELD_UNIT);
270  case ACPI_TYPE_LOCAL_ALIAS:
272      return (ACPI_BTTYPE_INTEGER);
275  case ACPI_TYPE_LOCAL_RESOURCE:
276  case ACPI_TYPE_LOCAL_RESOURCE_FIELD:
278      return (ACPI_BTTYPE_REFERENCE);
280  default:
282      printf ("Unhandled encoded type: %X\n", Etype);
283      return (0);
284  }
285 }

288 /*****
289  * FUNCTION:      AnFormatBtype
290  * PARAMETERS:    Btype          - Bitfield of ACPI types
291  *                Buffer          - Where to put the ascii string
292  * RETURN:        None.
293  * DESCRIPTION:   Convert a Btype to a string of ACPI types
294  *****/
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  * FUNCTION:      AnGetBtype
357  * PARAMETERS:    Op          - Parse node whose type will be returned.
358  * RETURN:        The Btype associated with the Op.
359  * DESCRIPTION:   Get the (bitfield) ACPI type associated with the parse node.
360  *                Handles the case where the node is a name or method call and
361  *                the actual type must be obtained from the namespace node.
362  *****/
363
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  * FUNCTION:      AnMapObjTypeToBtype
443  * PARAMETERS:    Op                - A parse node
444  * RETURN:        A Btype
445  * DESCRIPTION:   Map object to the associated "Btype"
446  *****/
447
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     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
```

new/usr/src/common/acpica/compiler/aslcodegen.c

1

```
*****
15247 Fri Jan 17 07:29:32 2014
new/usr/src/common/acpica/compiler/aslcodegen.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslcodegen - AML code generation
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"
47 #include "amlcode.h"

49 #define _COMPONENT          ACPI_COMPILER
50         ACPI_MODULE_NAME    ("aslcodegen")

52 /* Local prototypes */

54 static ACPI_STATUS
55 CgAmlWriteWalk (
56     ACPI_PARSE_OBJECT      *Op,
57     UINT32                  Level,
58     void                     *Context);
```

new/usr/src/common/acpica/compiler/aslcodegen.c

2

```
60 static void
61 CgLocalWriteAmlData (
62     ACPI_PARSE_OBJECT      *Op,
63     void                    *Buffer,
64     UINT32                  Length);

66 static void
67 CgWriteAmlOpcode (
68     ACPI_PARSE_OBJECT      *Op);

70 static void
71 CgWriteTableHeader (
72     ACPI_PARSE_OBJECT      *Op);

74 static void
75 CgCloseTable (
76     void);

78 static void
79 CgWriteNode (
80     ACPI_PARSE_OBJECT      *Op);

83 /*****
84 *
85 * FUNCTION:      CgGenerateAmlOutput
86 *
87 * PARAMETERS:    None.
88 *
89 * RETURN:        None
90 *
91 * DESCRIPTION:   Generate AML code. Currently generates the listing file
92 *                simultaneously.
93 *
94 *****/

96 void
97 CgGenerateAmlOutput (
98     void)
99 {
101     DbgPrint (ASL_DEBUG_OUTPUT, "\nWriting AML\n\n");

103     /* Generate the AML output file */

105     FlSeekFile (ASL_FILE_SOURCE_OUTPUT, 0);
106     Gbl_SourceLine = 0;
107     Gbl_NextError = Gbl_ErrorLog;

109     TrWalkParseTree (RootNode, ASL_WALK_VISIT_DOWNWARD,
110                     CgAmlWriteWalk, NULL, NULL);
111     CgCloseTable ();
112 }

115 /*****
116 *
117 * FUNCTION:      CgAmlWriteWalk
118 *
119 * PARAMETERS:    ASL_WALK_CALLBACK
120 *
121 * RETURN:        Status
122 *
123 * DESCRIPTION:   Parse tree walk to generate the AML code.
124 *
125 *****/>
```

```

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  * FUNCTION:      CgWriteAmlOpcode
226  *
227  * PARAMETERS:    Op          - Parse node with an AML opcode
228  *
229  * RETURN:        None.
230  *
231  * DESCRIPTION:    Write the AML opcode corresponding to a parse node.
232  *
233  *****/
234
235 static void
236 CgWriteAmlOpcode (
237     ACPI_PARSE_OBJECT *Op)
238 {
239     UINT8 PkgLenFirstByte;
240     union {
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:
262
263         /* These opcodes should not get here */
264
265         printf ("Found a node with an unassigned AML opcode\n");
266         FlPrintFile (ASL_FILE_STDERR, "Found a node with an unassigned AML opcode\n");
267         return;
268
269     case AML_INT_RESERVEDFIELD_OP:
270
271         /* Special opcodes for within a field definition */
272
273         Aml.Opcode = AML_FIELD_OFFSET_OP;
274         break;
275
276     case AML_INT_ACCESSFIELD_OP:
277
278         Aml.Opcode = AML_FIELD_ACCESS_OP;
279         break;
280
281     case AML_INT_CONNECTION_OP:
282
283         Aml.Opcode = AML_FIELD_CONNECTION_OP;
284         break;
285
286     default:
287
288         Aml.Opcode = Op->Asl.AmlOpcode;
289         break;
290     }
291
292     switch (Aml.Opcode)
293     {
294     case AML_PACKAGE_LENGTH:
295
296         /* Value is the length to be encoded (Used in field definitions) */
297
298         PkgLen.Len = (UINT32) Op->Asl.Value.Integer;
299         break;
300
301     default:
302
303         /* Check for two-byte opcode */
304
305         if (Aml.Opcode > 0x00FF)
306         {
307             /* Write the high byte first */
308
309             CgLocalWriteAmlData (Op, &Aml.OpcodeBytes[1], 1);
310
311             CgLocalWriteAmlData (Op, &Aml.OpcodeBytes[0], 1);
312
313             /* Subtreelength doesn't include length of package length bytes */
314
315             PkgLen.Len = Op->Asl.AmlSubtreeLength + Op->Asl.AmlPkgLenBytes;
316             break;
317         }
318
319         /* Does this opcode have an associated "PackageLength" field? */
320
321         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 */
328
329             CgLocalWriteAmlData (Op, &PkgLen.LenBytes[0], 1);
330
331         }
332         else if (Op->Asl.AmlPkgLenBytes != 0)
333         {
334             /*
335              * Encode the "bytes to follow" in the first byte, top two bits.
336              * The low-order nybble of the length is in the bottom 4 bits
337              */
338             PkgLenFirstByte = (UINT8)
339                 (((UINT32) (Op->Asl.AmlPkgLenBytes - 1) << 6) |
340                  (PkgLen.LenBytes[0] & 0x0F));
341
342             CgLocalWriteAmlData (Op, &PkgLenFirstByte, 1);
343
344             /*
345              * Shift the length over by the 4 bits we just stuffed
346              * in the first byte
347              */
348             PkgLen.Len >>= 4;
349
350             /* Now we can write the remaining bytes - either 1, 2, or 3 bytes */
351
352             for (i = 0; i < (UINT32) (Op->Asl.AmlPkgLenBytes - 1); i++)
353             {
354                 CgLocalWriteAmlData (Op, &PkgLen.LenBytes[i], 1);
355             }
356         }
357
358         switch (Aml.Opcode)
359         {
360         case AML_BYTE_OP:
361
362             CgLocalWriteAmlData (Op, &Op->Asl.Value.Integer, 1);
363             break;
364
365         case AML_WORD_OP:
366
367             CgLocalWriteAmlData (Op, &Op->Asl.Value.Integer, 2);
368             break;
369
370         case AML_DWORD_OP:
371
372             CgLocalWriteAmlData (Op, &Op->Asl.Value.Integer, 4);
373             break;
374
375         case AML_QWORD_OP:
376
377             CgLocalWriteAmlData (Op, &Op->Asl.Value.Integer, 8);
378             break;
379
380         case AML_STRING_OP:
381
382             CgLocalWriteAmlData (Op, Op->Asl.Value.String, Op->Asl.AmlLength);
383             break;
384
385         default:
386
387             /* All data opcodes must appear above */
388
389             break;

```

```

390     }
391 }

394 /*****
395  *
396  * FUNCTION:      CgWriteTableHeader
397  *
398  * PARAMETERS:    Op          - The DEFINITIONBLOCK node
399  *
400  * RETURN:        None
401  *
402  * DESCRIPTION:   Write a table header corresponding to the DEFINITIONBLOCK
403  *
404  *****/

406 static void
407 CgWriteTableHeader (
408     ACPI_PARSE_OBJECT *Op)
409 {
410     ACPI_PARSE_OBJECT *Child;

413     /* AML filename */

415     Child = Op->Asl.Child;

417     /* Signature */

419     Child = Child->Asl.Next;
420     strncpy (TableHeader.Signature, Child->Asl.Value.String, 4);

422     /* Revision */

424     Child = Child->Asl.Next;
425     TableHeader.Revision = (UINT8) Child->Asl.Value.Integer;

427     /* Command-line Revision override */

429     if (Gbl_RevisionOverride)
430     {
431         TableHeader.Revision = Gbl_RevisionOverride;
432     }

434     /* OEMID */

436     Child = Child->Asl.Next;
437     strncpy (TableHeader.OemId, Child->Asl.Value.String, 6);

439     /* OEM TableID */

441     Child = Child->Asl.Next;
442     strncpy (TableHeader.OemTableId, Child->Asl.Value.String, 8);

444     /* OEM Revision */

446     Child = Child->Asl.Next;
447     TableHeader.OemRevision = (UINT32) Child->Asl.Value.Integer;

449     /* Compiler ID */

451     ACPI_MOVE_NAME (TableHeader.AslCompilerId, ASL_CREATOR_ID);

453     /* Compiler version */

455     TableHeader.AslCompilerRevision = ASL_REVISION;

```

```

457     /* Table length. Checksum zero for now, will rewrite later */

459     TableHeader.Length = Gbl_TableLength;
460     TableHeader.Checksum = 0;

462     CgLocalWriteAmlData (Op, &TableHeader, sizeof (ACPI_TABLE_HEADER));
463 }

466 /*****
467  *
468  * FUNCTION:      CgCloseTable
469  *
470  * PARAMETERS:    None.
471  *
472  * RETURN:        None.
473  *
474  * DESCRIPTION:   Complete the ACPI table by calculating the checksum and
475  *                re-writing the header.
476  *
477  *****/

479 static void
480 CgCloseTable (
481     void)
482 {
483     signed char      Sum;
484     UINT8             FileByte;

487     FlSeekFile (ASL_FILE_AML_OUTPUT, 0);
488     Sum = 0;

490     /* Calculate the checksum over the entire file */

492     while (FlReadFile (ASL_FILE_AML_OUTPUT, &FileByte, 1) == AE_OK)
493     {
494         Sum = (signed char) (Sum + FileByte);
495     }

497     /* Re-write the table header with the checksum */

499     TableHeader.Checksum = (UINT8) (0 - Sum);

501     FlSeekFile (ASL_FILE_AML_OUTPUT, 0);
502     CgLocalWriteAmlData (NULL, &TableHeader, sizeof (ACPI_TABLE_HEADER));
503 }

506 /*****
507  *
508  * FUNCTION:      CgWriteNode
509  *
510  * PARAMETERS:    Op          - Parse node to write.
511  *
512  * RETURN:        None.
513  *
514  * DESCRIPTION:   Write the AML that corresponds to a parse node.
515  *
516  *****/

518 static void
519 CgWriteNode (
520     ACPI_PARSE_OBJECT *Op)
521 {

```



```

522     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 */

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     }

536     Op->Asl.FinalAmlLength = 0;

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:

545         CgLocalWriteAmlData (Op, &Op->Asl.Value.Integer, Op->Asl.AmlLength);
546         return;

549     case AML_RAW_DATA_BUFFER:

551         CgLocalWriteAmlData (Op, Op->Asl.Value.Buffer, Op->Asl.AmlLength);
552         return;

555     case AML_RAW_DATA_CHAIN:

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;

565     default:

567         /* Internal data opcodes must all appear above */

569         break;
570     }

572     switch (Op->Asl.ParseOpcode)
573     {
574     case PARSEOP_DEFAULT_ARG:

576         break;

578     case PARSEOP_DEFINITIONBLOCK:

580         CgWriteTableHeader (Op);
581         break;

583     case PARSEOP_NAMESEG:
584     case PARSEOP_NAMESTRING:
585     case PARSEOP_METHODCALL:

587         CgLocalWriteAmlData (Op, Op->Asl.Value.String, Op->Asl.AmlLength);

```

```

588         break;

590     default:

592         CgWriteAmlOpcode (Op);
593         break;
594     }
595 }

```

new/usr/src/common/acpica/compiler/aslcompile.c

1

```
*****
26878 Fri Jan 17 07:29:32 2014
new/usr/src/common/acpica/compiler/aslcompile.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslcompile - top level compile module
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "aslcompiler.h"
45 #include "dtcompiler.h"
46
47 #include <stdio.h>
48 #include <time.h>
49 #include <acapps.h>
50
51 #define _COMPONENT          ACPI_COMPILER
52 #define ACPI_MODULE_NAME    ("aslcompile")
53
54 /*
55 * Main parser entry
56 * External is here in case the parser emits the same external in the
57 * generated header. (Newer versions of Bison)
58 */
59 int
```

new/usr/src/common/acpica/compiler/aslcompile.c

2

```
60 AslCompilerparse(
61     void);
62
63 /* Local prototypes */
64
65 static void
66 CmFlushSourceCode (
67     void);
68
69 static void
70 FlConsumeAnsiComment (
71     FILE                *Handle,
72     ASL_FILE_STATUS     *Status);
73
74 static void
75 FlConsumeNewComment (
76     FILE                *Handle,
77     ASL_FILE_STATUS     *Status);
78
79 static void
80 CmDumpAllEvents (
81     void);
82
83
84 /*****
85 *
86 * FUNCTION:      AslCompilerSignon
87 *
88 * PARAMETERS:   FileId      - ID of the output file
89 *
90 * RETURN:       None
91 *
92 * DESCRIPTION:  Display compiler signon
93 *
94 *****/
95
96 void
97 AslCompilerSignon (
98     UINT32                FileId)
99 {
100     char                  *Prefix = "";
101     char                  *UtilityName;
102
103
104     /* Set line prefix depending on the destination file type */
105
106     switch (FileId)
107     {
108     case ASL_FILE_ASM_SOURCE_OUTPUT:
109     case ASL_FILE_ASM_INCLUDE_OUTPUT:
110
111         Prefix = "; ";
112         break;
113
114     case ASL_FILE_HEX_OUTPUT:
115
116         if (Gbl_HexOutputFlag == HEX_OUTPUT_ASM)
117         {
118             Prefix = "; ";
119         }
120         else if ((Gbl_HexOutputFlag == HEX_OUTPUT_C) ||
121             (Gbl_HexOutputFlag == HEX_OUTPUT_ASL))
122         {
123             FlPrintFile (ASL_FILE_HEX_OUTPUT, "/*\n");
124             Prefix = " * ";
125         }
126     }
```

```

126     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  * FUNCTION:      AslCompilerFileHeader
162  *
163  * PARAMETERS:    FileId      - ID of the output file
164  *
165  * RETURN:        None
166  *
167  * DESCRIPTION:   Header used at the beginning of output files
168  *
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_ASX))
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\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  *****/

```

```

259 static void
260 CmFlushSourceCode (
261     void)
262 {
263     char                Buffer;

266     while (FlReadFile (ASL_FILE_INPUT, &Buffer, 1) != AE_ERROR)
267     {
268         AslInsertLineBuffer ((int) Buffer);
269     }

271     AslResetCurrentLineBuffer ();
272 }

275 /*****
276  *
277  * FUNCTION:      FlConsume*
278  *
279  * PARAMETERS:    Handle          - Open input file
280  *                Status          - File current status struct
281  *
282  * RETURN:        Number of lines consumed
283  *
284  * DESCRIPTION:   Step over both types of comment during check for ascii chars
285  *
286  *****/

288 static void
289 FlConsumeAnsiComment (
290     FILE                *Handle,
291     ASL_FILE_STATUS     *Status)
292 {
293     UINT8               Byte;
294     BOOLEAN              ClosingComment = FALSE;

297     while (fread (&Byte, 1, 1, Handle) == 1)
298     {
299         /* Scan until comment close is found */

301         if (ClosingComment)
302         {
303             if (Byte == '/')
304             {
305                 return;
306             }

308             if (Byte != '*')
309             {
310                 /* Reset */

312                 ClosingComment = FALSE;
313             }
314         }
315         else if (Byte == '*')
316         {
317             ClosingComment = TRUE;
318         }

320         /* Maintain line count */

322         if (Byte == 0x0A)
323         {

```

```

324             Status->Line++;
325         }

327         Status->Offset++;
328     }
329 }

332 static void
333 FlConsumeNewComment (
334     FILE                *Handle,
335     ASL_FILE_STATUS     *Status)
336 {
337     UINT8               Byte;

340     while (fread (&Byte, 1, 1, Handle) == 1)
341     {
342         Status->Offset++;

344         /* Comment ends at newline */

346         if (Byte == 0x0A)
347         {
348             Status->Line++;
349             return;
350         }
351     }
352 }

355 /*****
356  *
357  * FUNCTION:      FlCheckForAcpiTable
358  *
359  * PARAMETERS:    Handle          - Open input file
360  *
361  * RETURN:        Status
362  *
363  * DESCRIPTION:   Determine if a file seems to be a binary ACPI table, via the
364  *                following checks on what would be the table header:
365  *                0) File must be at least as long as an ACPI_TABLE_HEADER
366  *                1) The header length field must match the file size
367  *                2) Signature, OemId, OemTableId, AslCompilerId must be ASCII
368  *
369  *****/

371 ACPI_STATUS
372 FlCheckForAcpiTable (
373     FILE                *Handle)
374 {
375     ACPI_TABLE_HEADER    Table;
376     UINT32               FileSize;
377     size_t               Actual;
378     UINT32               i;

381     /* Read a potential table header */

383     Actual = fread (&Table, 1, sizeof (ACPI_TABLE_HEADER), Handle);
384     fseek (Handle, 0, SEEK_SET);

386     if (Actual < sizeof (ACPI_TABLE_HEADER))
387     {
388         return (AE_ERROR);
389     }

```

```

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  *                spectacular 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\n",
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 = FOpenAmlOutputFile (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     HxDHexOutput ();

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 ("%llu : %s\n", TotalParseNodes, "Parse nodes");
931      printf ("%llu : %s\n", Gbl_NsLookupCount, "Namespace searches");
932      printf ("%llu : %s\n", TotalNamedObjects, "Named objects");
933      printf ("%llu : %s\n", TotalMethods, "Control methods");
934      printf ("%llu : %s\n", TotalAllocations, "Memory Allocations");
935      printf ("%llu : %s\n", TotalAllocated, "Total allocated memory");
936      printf ("%llu : %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  }

```

new/usr/src/common/acpica/compiler/aslcompiler.h

1

```
*****
25960 Fri Jan 17 07:29:32 2014
new/usr/src/common/acpica/compiler/aslcompiler.h
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: aslcompiler.h - common include file for iASL
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #ifndef __ASLCOMPILER_H
45 #define __ASLCOMPILER_H
46
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "amlresrc.h"
50 #include "acdebug.h"
51
52 /* Microsoft-specific */
53
54 #if (defined WIN32 || defined WIN64)
55
56 /* warn : used #pragma pack */
57 #pragma warning(disable:4103)
```

new/usr/src/common/acpica/compiler/aslcompiler.h

2

```
60 /* warn : named type definition in parentheses */
61 #pragma warning(disable:4115)
62 #endif
63
64 #include <stdio.h>
65 #include <stdlib.h>
66 #include <stdarg.h>
67 #include <string.h>
68 #include <errno.h>
69 #include <ctype.h>
70
71 /* Compiler headers */
72
73 #include "asldefine.h"
74 #include "asltypes.h"
75 #include "aslmessages.h"
76 #include "aslglobal.h"
77 #include "preprocess.h"
78
79 /*****
80 *
81 * Compiler prototypes
82 *
83 *****/
84
85 /*
86 * Main ASL parser - generated from flex/bison, lex/yacc, etc.
87 */
88 /*
89 ACPI_PARSE_OBJECT *
90 AslDoError (
91     void);
92
93 int
94 AslCompilerlex(
95     void);
96
97 void
98 AslResetCurrentLineBuffer (
99     void);
100
101 void
102 AslInsertLineBuffer (
103     int SourceChar);
104
105 int
106 AslPopInputFileStack (
107     void);
108
109 void
110 AslPushInputFileStack (
111     FILE *InputFile,
112     char *Filename);
113
114 /*
115 * aslstartup - entered from main()
116 */
117 void
118 AslInitializeGlobals (
119     void);
120
121 typedef
122 ACPI_STATUS (*ASL_PATHNAME_CALLBACK) (
123     char *);
```

```

125 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  * aslbtotypes - 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 - optimization
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
522 */
523 void
524 FOpenFile (
525     UINT32          FileId,
526     char            *Filename,
527     char            *Mode);

530 /*
531 * asllength - calculate/adjust AML package lengths
532 */
533 ACPI_STATUS
534 LnPackageLengthWalk (
535     ACPI_PARSE_OBJECT *Op,
536     UINT32            Level,
537     void              *Context);

539 ACPI_STATUS
540 LnInitLengthsWalk (
541     ACPI_PARSE_OBJECT *Op,
542     UINT32            Level,
543     void              *Context);

545 void
546 CgGenerateAmlLengths (
547     ACPI_PARSE_OBJECT *Op);

550 /*
551 * aslmap - opcode mappings and reserved method names
552 */
553 ACPI_OBJECT_TYPE
554 AslMapNamedOpcodeToDataType (
555     UINT16            Opcode);

558 /*
559 * aslpredef - ACPI predefined names support
560 */
561 BOOLEAN
562 ApCheckForPredefinedMethod (
563     ACPI_PARSE_OBJECT *Op,
564     ASL_METHOD_INFO    *MethodInfo);

566 void
567 ApCheckPredefinedReturnValue (
568     ACPI_PARSE_OBJECT *Op,
569     ASL_METHOD_INFO    *MethodInfo);

571 UINT32
572 ApCheckForPredefinedName (
573     ACPI_PARSE_OBJECT *Op,
574     char              *Name);

576 void
577 ApCheckForPredefinedObject (
578     ACPI_PARSE_OBJECT *Op,
579     char              *Name);

581 ACPI_STATUS
582 ApCheckObjectType (
583     const char        *PredefinedName,
584     ACPI_PARSE_OBJECT *Op,
585     UINT32            ExpectedBtypes,
586     UINT32            PackageIndex);

```

```

588 void
589 ApDisplayReservedNames (
590     void);

593 /*
594 * aslprepkg - ACPI predefined names support for packages
595 */
596 void
597 ApCheckPackage (
598     ACPI_PARSE_OBJECT *ParentOp,
599     const ACPI_PREDEFINED_INFO *Predefined);

602 /*
603 * asltransform - parse tree transformations
604 */
605 ACPI_STATUS
606 TrAmlTransformWalk (
607     ACPI_PARSE_OBJECT *Op,
608     UINT32            Level,
609     void              *Context);

612 /*
613 * asltree - parse tree support
614 */
615 ACPI_STATUS
616 TrWalkParseTree (
617     ACPI_PARSE_OBJECT *Op,
618     UINT32            Visitation,
619     ASL_WALK_CALLBACK DescendingCallback,
620     ASL_WALK_CALLBACK AscendingCallback,
621     void              *Context);

623 /* Values for "Visitation" parameter above */

625 #define ASL_WALK_VISIT_DOWNWARD    0x01
626 #define ASL_WALK_VISIT_UPWARD      0x02
627 #define ASL_WALK_VISIT_TWICE       (ASL_WALK_VISIT_DOWNWARD | ASL_WALK_VISIT_UP)

630 ACPI_PARSE_OBJECT *
631 TrAllocateNode (
632     UINT32            ParseOpcode);

634 void
635 TrReleaseNode (
636     ACPI_PARSE_OBJECT *Op);

638 ACPI_PARSE_OBJECT *
639 TrUpdateNode (
640     UINT32            ParseOpcode,
641     ACPI_PARSE_OBJECT *Op);

643 ACPI_PARSE_OBJECT *
644 TrCreateNode (
645     UINT32            ParseOpcode,
646     UINT32            NumChildren,
647     ...);

649 ACPI_PARSE_OBJECT *
650 TrCreateLeafNode (
651     UINT32            ParseOpcode);

```

```

653 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 (
918     ACPI_PARSE_OBJECT *Op,
919     ACPI_PARSE_OBJECT *NameNode);

921 ACPI_PARSE_OBJECT *
922 UtCheckIntegerRange (
923     ACPI_PARSE_OBJECT *Op,
924     UINT32 LowValue,
925     UINT32 HighValue);

927 UINT64
928 UtDoConstant (
929     char *String);

931 ACPI_STATUS
932 UtStrtoul64 (
933     char *String,
934     UINT32 Base,
935     UINT64 *RetInteger);

938 /*
939  * asluuid - UUID support
940  */
941 ACPI_STATUS
942 AuValidateUuid (
943     char *InString);

945 ACPI_STATUS
946 AuConvertStringToUuid (
947     char *InString,
948     char *UuidBuffer);

950 ACPI_STATUS
951 AuConvertUuidToString (
952     char *UuidBuffer,
953     char *OutString);

955 /*
956  * aslresource - Resource template generation utilities
957  */
958 void
959 RsSmallAddressCheck (
960     UINT8 Type,
961     UINT32 Minimum,
962     UINT32 Maximum,
963     UINT32 Length,
964     UINT32 Alignment,
965     ACPI_PARSE_OBJECT *MinOp,
966     ACPI_PARSE_OBJECT *MaxOp,
967     ACPI_PARSE_OBJECT *LengthOp,
968     ACPI_PARSE_OBJECT *AlignOp,
969     ACPI_PARSE_OBJECT *Op);

971 void
972 RsLargeAddressCheck (
973     UINT64 Minimum,
974     UINT64 Maximum,
975     UINT64 Length,
976     UINT64 Granularity,
977     UINT8 Flags,
978     ACPI_PARSE_OBJECT *MinOp,
979     ACPI_PARSE_OBJECT *MaxOp,
980     ACPI_PARSE_OBJECT *LengthOp,
981     ACPI_PARSE_OBJECT *GranOp,
982     ACPI_PARSE_OBJECT *Op);

```

```

984 UINT16
985 RsGetStringDataLength (
986     ACPI_PARSE_OBJECT *InitializerOp);

988 ASL_RESOURCE_NODE *
989 RsAllocateResourceNode (
990     UINT32 Size);

992 void
993 RsCreateResourceField (
994     ACPI_PARSE_OBJECT *Op,
995     char *Name,
996     UINT32 ByteOffset,
997     UINT32 BitOffset,
998     UINT32 BitLength);

1000 void
1001 RsSetFlagBits (
1002     UINT8 *Flags,
1003     ACPI_PARSE_OBJECT *Op,
1004     UINT8 Position,
1005     UINT8 DefaultBit);

1007 void
1008 RsSetFlagBits16 (
1009     UINT16 *Flags,
1010     ACPI_PARSE_OBJECT *Op,
1011     UINT8 Position,
1012     UINT8 DefaultBit);

1014 ACPI_PARSE_OBJECT *
1015 RsCompleteNodeAndGetNext (
1016     ACPI_PARSE_OBJECT *Op);

1018 void
1019 RsCheckListForDuplicates (
1020     ACPI_PARSE_OBJECT *Op);

1022 ASL_RESOURCE_NODE *
1023 RsDoOneResourceDescriptor (
1024     ACPI_PARSE_OBJECT *DescriptorTypeOp,
1025     UINT32 CurrentByteOffset,
1026     UINT8 *State);

1028 /* Values for State above */

1030 #define ACPI_RSTATE_NORMAL 0
1031 #define ACPI_RSTATE_START_DEPENDENT 1
1032 #define ACPI_RSTATE_DEPENDENT_LIST 2

1034 UINT32
1035 RsLinkDescriptorChain (
1036     ASL_RESOURCE_NODE **PreviousRnode,
1037     ASL_RESOURCE_NODE *Rnode);

1039 void
1040 RsDoResourceTemplate (
1041     ACPI_PARSE_OBJECT *Op);

1044 /*
1045  * aslrestype1 - Miscellaneous Small descriptors
1046  */
1047 ASL_RESOURCE_NODE *
1048 RsDoEndTagDescriptor (

```

```

1049     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  * aslrestype1 - 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 */

```

new/usr/src/common/acpica/compiler/aslcompiler.1

1

```
*****
33668 Fri Jan 17 07:29:32 2014
new/usr/src/common/acpica/compiler/aslcompiler.1
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 %{
2 /*****
3 *
4 * Module Name: aslcompiler.1 - Flex/lex input file
5 *
6 *****/
7
8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #include "aslcompiler.h"
46 #include "aslcompiler.y.h"
47
48 #include <stdlib.h>
49 #include <string.h>
50 YYSTYPE AslCompilerIval;
51
52 /*
53 * Generation: Use the following command line:
54 *
55 * flex.exe -PAslCompiler -i -o$(InputPath).c $(InputPath)
56 *
57 * -i: Scanner must be case-insensitive
58 */
```

new/usr/src/common/acpica/compiler/aslcompiler.1

2

```
60 #define _COMPONENT ACPI_COMPILER
61 ACPI_MODULE_NAME ("aslscanner")
62
63 /* Local prototypes */
64
65 static void
66 AslDoLineDirective (void);
67
68 static char
69 AslDoComment (void);
70
71 static char
72 AslDoCommentType2 (void);
73
74 static char
75 AslDoStringLiteral (void);
76
77 static void
78 count (int type);
79
80 /*! [Begin] no source code translation */
81
82 %{
83 /* Definitions */
84
85 LeadNameChar [A-Za-z_]
86 DigitChar [0-9]
87 HexDigitChar [A-Fa-f0-9]
88 RootChar [\]
89 Nothing []
90
91 NameChar [A-Za-z_0-9]
92 NameSeg1 {LeadNameChar}{NameChar}
93 NameSeg2 {LeadNameChar}{NameChar}{NameChar}
94 NameSeg3 {LeadNameChar}{NameChar}{NameChar}{NameChar}
95 NameSeg {LeadNameChar}{NameSeg1}|{NameSeg2}|{NameSeg3}
96
97 NameString {RootChar}|{RootChar}{NamePath}|[\\^]+{NamePath}|{Non
98 NonEmptyNamePath}?
99 NamePath {NameSeg}{NamePathTail}*
100 NamePathTail [.]{NameSeg}
101
102 %%
103 /* Rules */
104
105 [ ] { count (0); }
106 [\n] { count (0); } /* Handle files with both LF and CR/L
107 [\r] { count (0); } /* termination on both Unix and Windo
108 [\t] { count (0); }
109
110
111 /*
112 if (!AslDoComment ()) yyterminate ();
113 if (!AslDoCommentType2 ()) yyterminate ();
114
115
116 if (AslDoStringLiteral ()) return (PARSEOP_STRING_
117 count (0); return (';', );
118
119
120 0[xX]{HexDigitChar}+ |
121 {DigitChar}+ { AslCompilerIval.i = UtDoConstant ((char *) AslComp
122 count (1); return (PARSEOP_INTEGER); }
123
124 "Include" { count (1); return (PARSEOP_INCLUDE); }
125 "External" { count (1); return (PARSEOP_EXTERNAL); }
```

```

127 /*
128  * The #line directive is emitted by the preprocessor and handled
129  * here in the main iASL lexer - simply set the line number and
130  * optionally the current filename.
131  */
132 "#line"                { AslDoLineDirective ();}

135 /*****
136  *
137  * Main ASL operators
138  *
139  *****/

141 "AccessAs"             { count (1); return (PARSEOP_ACCESSAS); }
142 "Acquire"              { count (3); return (PARSEOP_ACQUIRE); }
143 "Add"                  { count (3); return (PARSEOP_ADD); }
144 "Alias"                { count (2); return (PARSEOP_ALIAS); }
145 "And"                  { count (3); return (PARSEOP_AND); }
146 "BankField"           { count (2); return (PARSEOP_BANKFIELD); }
147 "Break"               { count (3); return (PARSEOP_BREAK); }
148 "BreakPoint"          { count (3); return (PARSEOP_BREAKPOINT); }
149 "Buffer"              { count (1); return (PARSEOP_BUFFER); }
150 "Case"                { count (3); return (PARSEOP_CASE); }
151 "Concatenate"          { count (3); return (PARSEOP_CONCATENATE); }
152 "ConcatenateResTemplate" { count (3); return (PARSEOP_CONCATENATERESTEMPLATE); }
153 "CondRefOf"           { count (3); return (PARSEOP_CONDREFOF); }
154 "Connection"          { count (2); return (PARSEOP_CONNECTION); }
155 "Continue"            { count (3); return (PARSEOP_CONTINUE); }
156 "CopyObject"          { count (3); return (PARSEOP_COPYOBJECT); }
157 "CreateBitField"       { count (2); return (PARSEOP_CREATEBITFIELD); }
158 "CreateByteField"      { count (2); return (PARSEOP_CREATEBYTEFIELD); }
159 "CreateDWordField"     { count (2); return (PARSEOP_CREATEDWORDFIELD); }
160 "CreateField"          { count (2); return (PARSEOP_CREATEFIELD); }
161 "CreateQWordField"     { count (2); return (PARSEOP_CREATEQWORDFIELD); }
162 "CreateWordField"      { count (2); return (PARSEOP_CREATEWORDFIELD); }
163 "DataTableRegion"      { count (2); return (PARSEOP_DATATABLEREGION); }
164 "Debug"               { count (1); return (PARSEOP_DEBUG); }
165 "Decrement"           { count (3); return (PARSEOP_DECREMENT); }
166 "Default"             { count (3); return (PARSEOP_DEFAULT); }
167 "DefinitionBlock"      { count (1); return (PARSEOP_DEFINITIONBLOCK); }
168 "DeRefOf"             { count (3); return (PARSEOP_DEREFOf); }
169 "Device"              { count (2); return (PARSEOP_DEVICE); }
170 "Divide"              { count (3); return (PARSEOP_DIVIDE); }
171 "Eisaid"              { count (1); return (PARSEOP_EISAID); }
172 "Else"               { count (3); return (PARSEOP_ELSE); }
173 "ElseIf"             { count (3); return (PARSEOP_ELSEIF); }
174 "Event"              { count (2); return (PARSEOP_EVENT); }
175 "Fatal"              { count (3); return (PARSEOP_FATAL); }
176 "Field"              { count (2); return (PARSEOP_FIELD); }
177 "FindSetLeftBit"      { count (3); return (PARSEOP_FINDSETLEFTBIT); }
178 "FindSetRightBit"     { count (3); return (PARSEOP_FINDSETRIGHTBIT); }
179 "FromBcd"            { count (3); return (PARSEOP_FROMBCD); }
180 "Function"            { count (2); return (PARSEOP_FUNCTION); }
181 "If"                 { count (3); return (PARSEOP_IF); }
182 "Increment"          { count (3); return (PARSEOP_INCREMENT); }
183 "Index"              { count (3); return (PARSEOP_INDEX); }
184 "IndexField"          { count (2); return (PARSEOP_INDEXFIELD); }
185 "LAnd"               { count (3); return (PARSEOP_LAND); }
186 "LEqual"            { count (3); return (PARSEOP_LEQUAL); }
187 "LGreater"           { count (3); return (PARSEOP_LGREATER); }
188 "LGreaterEqual"      { count (3); return (PARSEOP_LGREATEREQUAL); }
189 "LLess"             { count (3); return (PARSEOP_LLESS); }
190 "LLessEqual"         { count (3); return (PARSEOP_LLESSEQUAL); }
191 "LNot"              { count (3); return (PARSEOP_LNOT); }

```

```

192 "LNotEqual"          { count (3); return (PARSEOP_LNOTEQUAL); }
193 "Load"               { count (3); return (PARSEOP_LOAD); }
194 "LoadTable"          { count (3); return (PARSEOP_LOADTABLE); }
195 "Lor"               { count (3); return (PARSEOP_LOR); }
196 "Match"             { count (3); return (PARSEOP_MATCH); }
197 "Method"            { count (2); return (PARSEOP_METHOD); }
198 "Mid"               { count (3); return (PARSEOP_MID); }
199 "Mod"               { count (3); return (PARSEOP_MOD); }
200 "Multiply"           { count (3); return (PARSEOP_MULTIPLY); }
201 "Mutex"             { count (2); return (PARSEOP_MUTEX); }
202 "Name"              { count (2); return (PARSEOP_NAME); }
203 "NAnd"              { count (3); return (PARSEOP_NAND); }
204 "Noop"              { if (!AcpiGbl_IgnoreNoopOperator) { count (3); return (PARSEOP_NOOP); } }
205 "Nor"               { count (3); return (PARSEOP_NOR); }
206 "Not"              { count (3); return (PARSEOP_NOT); }
207 "Notify"            { count (3); return (PARSEOP_NOTIFY); }
208 "ObjectType"        { count (3); return (PARSEOP_OBJECTTYPE); }
209 "Offset"           { count (1); return (PARSEOP_OFFSET); }
210 "One"              { count (1); return (PARSEOP_ONE); }
211 "Ones"             { count (1); return (PARSEOP_ONES); }
212 "OperationRegion"   { count (2); return (PARSEOP_OPERATIONREGION); }
213 "Or"               { count (3); return (PARSEOP_OR); }
214 "Package"          { count (1); return (PARSEOP_PACKAGE); }
215 "PowerResource"     { count (2); return (PARSEOP_POWERRESOURCE); }
216 "Processor"        { count (2); return (PARSEOP_PROCESSOR); }
217 "RefOf"            { count (3); return (PARSEOP_REFOF); }
218 "Release"          { count (3); return (PARSEOP_RELEASE); }
219 "Reset"            { count (3); return (PARSEOP_RESET); }
220 "Return"           { count (3); return (PARSEOP_RETURN); }
221 "Revision"         { count (1); return (PARSEOP_REVISION); }
222 "Scope"           { count (2); return (PARSEOP_SCOPE); }
223 "ShiftLeft"       { count (3); return (PARSEOP_SHIFTLLEFT); }
224 "ShiftRight"      { count (3); return (PARSEOP_SHIFTRIGHT); }
225 "Signal"          { count (3); return (PARSEOP_SIGNAL); }
226 "SizeOf"         { count (3); return (PARSEOP_SIZEOF); }
227 "Sleep"          { count (3); return (PARSEOP_SLEEP); }
228 "Stall"          { count (3); return (PARSEOP_STALL); }
229 "Store"          { count (3); return (PARSEOP_STORE); }
230 "Subtract"       { count (3); return (PARSEOP_SUBTRACT); }
231 "Switch"         { count (3); return (PARSEOP_SWITCH); }
232 "ThermalZone"    { count (2); return (PARSEOP_THERMALZONE); }
233 "Timer"          { count (3); return (PARSEOP_TIMER); }
234 "ToBcd"          { count (3); return (PARSEOP_TOBCD); }
235 "ToBuffer"       { count (3); return (PARSEOP_TOBUFFER); }
236 "ToDecimalString" { count (3); return (PARSEOP_TODECIMALSTRING); }
237 "ToHexString"    { count (3); return (PARSEOP_TOHEXSTRING); }
238 "ToInteger"      { count (3); return (PARSEOP_TOINTEGER); }
239 "ToString"       { count (3); return (PARSEOP_TOSTRING); }
240 "ToUuid"         { count (1); return (PARSEOP_TOUUID); }
241 "Unicode"        { count (1); return (PARSEOP_UNICODE); }
242 "Unload"         { count (3); return (PARSEOP_UNLOAD); }
243 "Wait"           { count (3); return (PARSEOP_WAIT); }
244 "While"          { count (3); return (PARSEOP_WHILE); }
245 "Xor"            { count (3); return (PARSEOP_XOR); }
246 "Zero"          { count (1); return (PARSEOP_ZERO); }

248 /* Control method arguments and locals */

250 "Arg0"            { count (1); return (PARSEOP_ARG0); }
251 "Arg1"            { count (1); return (PARSEOP_ARG1); }
252 "Arg2"            { count (1); return (PARSEOP_ARG2); }
253 "Arg3"            { count (1); return (PARSEOP_ARG3); }
254 "Arg4"            { count (1); return (PARSEOP_ARG4); }
255 "Arg5"            { count (1); return (PARSEOP_ARG5); }
256 "Arg6"            { count (1); return (PARSEOP_ARG6); }
257 "Local0"          { count (1); return (PARSEOP_LOCAL0); }

```

```

258 "Local1"      { count (1); return (PARSEOP_LOCAL1); }
259 "Local2"      { count (1); return (PARSEOP_LOCAL2); }
260 "Local3"      { count (1); return (PARSEOP_LOCAL3); }
261 "Local4"      { count (1); return (PARSEOP_LOCAL4); }
262 "Local5"      { count (1); return (PARSEOP_LOCAL5); }
263 "Local6"      { count (1); return (PARSEOP_LOCAL6); }
264 "Local7"      { count (1); return (PARSEOP_LOCAL7); }

267 /*****
268 *
269 * Resource Descriptor macros
270 *
271 *****/

273 "ResourceTemplate" { count (1); return (PARSEOP_RESOURCETEMPLATE); }
274 "RawDataBuffer"   { count (1); return (PARSEOP_DATABUFFER); }

276 "DMA" { count (1); return (PARSEOP_DMA); }
277 "DWordIO" { count (1); return (PARSEOP_DWORDIO); }
278 "DWordMemory" { count (1); return (PARSEOP_DWORDMEMORY); }
279 "DWordSpace" { count (1); return (PARSEOP_DWORDSPACE); }
280 "EndDependentFn" { count (1); return (PARSEOP_ENDDEPENDENTFN); }
281 "ExtendedIO" { count (1); return (PARSEOP_EXTENDEDIO); }
282 "ExtendedMemory" { count (1); return (PARSEOP_EXTENDEDMEMORY); }
283 "ExtendedSpace" { count (1); return (PARSEOP_EXTENDEDSPACE); }
284 "FixedDma" { count (1); return (PARSEOP_FIXEDDMA); }
285 "FixedIO" { count (1); return (PARSEOP_FIXEDIO); }
286 "GpioInt" { count (1); return (PARSEOP_GPIO_INT); }
287 "GpioIo" { count (1); return (PARSEOP_GPIO_IO); }
288 "I2cSerialBus" { count (1); return (PARSEOP_I2C_SERIALBUS); }
289 "Interrupt" { count (1); return (PARSEOP_INTERRUPT); }
290 "IO" { count (1); return (PARSEOP_IO); }
291 "IRQ" { count (1); return (PARSEOP_IRQ); }
292 "IRQNoFlags" { count (1); return (PARSEOP_IRQNOFLAGS); }
293 "Memory24" { count (1); return (PARSEOP_MEMORY24); }
294 "Memory32" { count (1); return (PARSEOP_MEMORY32); }
295 "Memory32Fixed" { count (1); return (PARSEOP_MEMORY32FIXED); }
296 "QWordIO" { count (1); return (PARSEOP_QWORDIO); }
297 "QWordMemory" { count (1); return (PARSEOP_QWORDMEMORY); }
298 "QWordSpace" { count (1); return (PARSEOP_QWORDSPACE); }
299 "Register" { count (1); return (PARSEOP_REGISTER); }
300 "SpiSerialBus" { count (1); return (PARSEOP_SPI_SERIALBUS); }
301 "StartDependentFn" { count (1); return (PARSEOP_STARTDEPENDENTFN); }
302 "StartDependentFnNoPri" { count (1); return (PARSEOP_STARTDEPENDENTFN_NOPRI); }
303 "UartSerialBus" { count (1); return (PARSEOP_UART_SERIALBUS); }
304 "VendorLong" { count (1); return (PARSEOP_VENDORLONG); }
305 "VendorShort" { count (1); return (PARSEOP_VENDORSHORT); }
306 "WordBusNumber" { count (1); return (PARSEOP_WORDBUSNUMBER); }
307 "WordIO" { count (1); return (PARSEOP_WORDIO); }
308 "WordSpace" { count (1); return (PARSEOP_WORDSPACE); }

311 /*****
312 *
313 * Keywords used as arguments to ASL operators and macros
314 *
315 *****/

317 /* AccessAttribKeyword: Serial Bus Attributes (ACPI 5.0) */

319 "AttribQuick" { count (0); return (PARSEOP_ACCESSATTRIB_QUICK); }
320 "AttribSendReceive" { count (0); return (PARSEOP_ACCESSATTRIB_SND_RCV); }
321 "AttribByte" { count (0); return (PARSEOP_ACCESSATTRIB_BYTE); }
322 "AttribWord" { count (0); return (PARSEOP_ACCESSATTRIB_WORD); }
323 "AttribBlock" { count (0); return (PARSEOP_ACCESSATTRIB_BLOCK); }

```

```

324 "AttribProcessCall" { count (0); return (PARSEOP_ACCESSATTRIB_WORD_CALL); }
325 "AttribBlockProcessCall" { count (0); return (PARSEOP_ACCESSATTRIB_BLOCK_CALL); }

327 /* AccessAttribKeyword: Legacy synonyms for above (pre-ACPI 5.0) */

329 "SMBQuick" { count (0); return (PARSEOP_ACCESSATTRIB_QUICK); }
330 "SMBSendReceive" { count (0); return (PARSEOP_ACCESSATTRIB_SND_RCV); }
331 "SMBByte" { count (0); return (PARSEOP_ACCESSATTRIB_BYTE); }
332 "SMBWord" { count (0); return (PARSEOP_ACCESSATTRIB_WORD); }
333 "SMBBlock" { count (0); return (PARSEOP_ACCESSATTRIB_BLOCK); }
334 "SMBProcessCall" { count (0); return (PARSEOP_ACCESSATTRIB_WORD_CALL); }
335 "SMBBlockProcessCall" { count (0); return (PARSEOP_ACCESSATTRIB_BLOCK_CALL); }

337 /* AccessTypeKeyword: Field Access Types */

339 "AnyAcc" { count (0); return (PARSEOP_ACCESSTYPE_ANY); }
340 "ByteAcc" { count (0); return (PARSEOP_ACCESSTYPE_BYTE); }
341 "WordAcc" { count (0); return (PARSEOP_ACCESSTYPE_WORD); }
342 "DWordAcc" { count (0); return (PARSEOP_ACCESSTYPE_DWORD); }
343 "QWordAcc" { count (0); return (PARSEOP_ACCESSTYPE_QWORD); }
344 "BufferAcc" { count (0); return (PARSEOP_ACCESSTYPE_BUF); }

346 /* AddressingModeKeyword: Mode - Resource Descriptors (ACPI 5.0) */

348 "AddressingMode7Bit" { count (0); return (PARSEOP_ADDRESSINGMODE_7BIT); }
349 "AddressingMode10Bit" { count (0); return (PARSEOP_ADDRESSINGMODE_10BIT); }

351 /* AddressKeyword: ACPI memory range types */

353 "AddressRangeMemory" { count (0); return (PARSEOP_ADDRESSTYPE_MEMORY); }
354 "AddressRangeReserved" { count (0); return (PARSEOP_ADDRESSTYPE_RESERVED); }
355 "AddressRangeNVS" { count (0); return (PARSEOP_ADDRESSTYPE_NVS); }
356 "AddressRangeACPI" { count (0); return (PARSEOP_ADDRESSTYPE_ACPI); }

358 /* BusMasterKeyword: DMA Bus Mastering */

360 "BusMaster" { count (0); return (PARSEOP_BUSMASTERTYPE_MASTER); }
361 "NotBusMaster" { count (0); return (PARSEOP_BUSMASTERTYPE_NOTMASTER); }

363 /* ByteLengthKeyword: Bits per Byte - Resource Descriptors (ACPI 5.0) */

365 "DataBitsFive" { count (0); return (PARSEOP_BITSPERBYTE_FIVE); }
366 "DataBitsSix" { count (0); return (PARSEOP_BITSPERBYTE_SIX); }
367 "DataBitsSeven" { count (0); return (PARSEOP_BITSPERBYTE_SEVEN); }
368 "DataBitsEight" { count (0); return (PARSEOP_BITSPERBYTE_EIGHT); }
369 "DataBitsNine" { count (0); return (PARSEOP_BITSPERBYTE_NINE); }

371 /* ClockPhaseKeyword: Resource Descriptors (ACPI 5.0) */

373 "ClockPhaseFirst" { count (0); return (PARSEOP_CLOCKPHASE_FIRST); }
374 "ClockPhaseSecond" { count (0); return (PARSEOP_CLOCKPHASE_SECOND); }

376 /* ClockPolarityKeyword: Resource Descriptors (ACPI 5.0) */

378 "ClockPolarityLow" { count (0); return (PARSEOP_CLOCKPOLARITY_LOW); }
379 "ClockPolarityHigh" { count (0); return (PARSEOP_CLOCKPOLARITY_HIGH); }

381 /* DecodeKeyword: Type of Memory Decoding - Resource Descriptors */

383 "PosDecode" { count (0); return (PARSEOP_DECODETYPE_POS); }
384 "SubDecode" { count (0); return (PARSEOP_DECODETYPE_SUB); }

386 /* DmaTypeKeyword: DMA Types - DMA Resource Descriptor */

388 "Compatibility" { count (0); return (PARSEOP_DMATYPE_COMPATIBILITY); }
389 "TypeA" { count (0); return (PARSEOP_DMATYPE_A); }

```

```

390 "TypeB"          { count (0); return (PARSEOP_DMATYPED_B); }
391 "TypeF"          { count (0); return (PARSEOP_DMATYPED_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); }
523 "IPMI"              { count (0); return (PARSEOP_REGIONSPACE_IPMI); }
524 "GeneralPurposeIo"   { count (0); return (PARSEOP_REGIONSPACE_GPIO); }
525 "GenericSerialBus"   { count (0); return (PARSEOP_REGIONSPACE_GSBUS); }
526 "PCC"               { count (0); return (PARSEOP_REGIONSPACE_PCC); }
527 "FFixedHW"          { count (0); return (PARSEOP_REGIONSPACE_FFIXEDHW); }

529 /* ResourceTypeKeyword: Resource Usage - Resource Descriptors */

531 "ResourceConsumer"   { count (0); return (PARSEOP_RESOURCETYPE_CONSUMER); }
532 "ResourceProducer"   { count (0); return (PARSEOP_RESOURCETYPE_PRODUCER); }

534 /* SerializeRuleKeyword: Control Method Serialization */

536 "Serialized"         { count (0); return (PARSEOP_SERIALIZERULE_SERIAL); }
537 "NotSerialized"      { count (0); return (PARSEOP_SERIALIZERULE_NOTSERIAL); }

539 /* ShareTypeKeyword: Interrupt Sharing - Resource Descriptors */

541 "Shared"             { count (0); return (PARSEOP_SHARETYPE_SHARED); }
542 "Exclusive"          { count (0); return (PARSEOP_SHARETYPE_EXCLUSIVE); }
543 "SharedAndWake"      { count (0); return (PARSEOP_SHARETYPE_SHAREDWAKE); }
544 "ExclusiveAndWake"   { count (0); return (PARSEOP_SHARETYPE_EXCLUSIVWAKE); }

546 /* SlaveModeKeyword: Resource Descriptors (ACPI 5.0) */

548 "ControllerInitiated" { count (0); return (PARSEOP_SLAVEMODE_CONTROLLERINI); }
549 "DeviceInitiated"    { count (0); return (PARSEOP_SLAVEMODE_DEVICEINIT); }

551 /* StopBitsKeyword: Resource Descriptors (ACPI 5.0) */

553 "StopBitsOne"        { count (0); return (PARSEOP_STOPBITS_ONE); }
554 "StopBitsOnePlusHalf" { count (0); return (PARSEOP_STOPBITS_ONEPLUSHALF); }
555 "StopBitsTwo"        { count (0); return (PARSEOP_STOPBITS_TWO); }
556 "StopBitsZero"       { count (0); return (PARSEOP_STOPBITS_ZERO); }

558 /* TransferWidthKeyword: DMA Widths - Fixed DMA Resource Descriptor (ACPI 5.0) */

560 "Width8bit"          { count (0); return (PARSEOP_XFERSIZE_8); }
561 "Width16bit"         { count (0); return (PARSEOP_XFERSIZE_16); }
562 "Width32bit"         { count (0); return (PARSEOP_XFERSIZE_32); }
563 "Width64bit"         { count (0); return (PARSEOP_XFERSIZE_64); }
564 "Width128bit"        { count (0); return (PARSEOP_XFERSIZE_128); }
565 "Width256bit"        { count (0); return (PARSEOP_XFERSIZE_256); }

567 /* TranslationKeyword: Translation Density Types - Resource Descriptors */

569 "SparseTranslation"   { count (0); return (PARSEOP_TRANSLATIONTYPE_SPARSE); }
570 "DenseTranslation"   { count (0); return (PARSEOP_TRANSLATIONTYPE_DENSE); }

572 /* TypeKeyword: Translation Types - Resource Descriptors */

574 "TypeTranslation"     { count (0); return (PARSEOP_TYPE_TRANSLATION); }
575 "TypeStatic"          { count (0); return (PARSEOP_TYPE_STATIC); }

577 /* UpdateRuleKeyword: Field Update Rules */

579 "Preserve"           { count (0); return (PARSEOP_UPDATERULE_PRESERVE); }
580 "WriteAsOnes"         { count (0); return (PARSEOP_UPDATERULE_ONES); }
581 "WriteAsZeros"        { count (0); return (PARSEOP_UPDATERULE_ZEROS); }

583 /* WireModeKeyword: SPI Wire Mode - Resource Descriptors (ACPI 5.0) */

585 "FourWireMode"        { count (0); return (PARSEOP_WIREMODE_FOUR); }
586 "ThreeWireMode"       { count (0); return (PARSEOP_WIREMODE_THREE); }

```

```

588 /* XferTypeKeyword: DMA Transfer Types */

590 "Transfer8"          { count (0); return (PARSEOP_XFERTYPE_8); }
591 "Transfer8_16"       { count (0); return (PARSEOP_XFERTYPE_8_16); }
592 "Transfer16"         { count (0); return (PARSEOP_XFERTYPE_16); }

594 /* Predefined compiler names */

596 "__DATE__"           { count (0); return (PARSEOP__DATE__); }
597 "__FILE__"           { count (0); return (PARSEOP__FILE__); }
598 "__LINE__"           { count (0); return (PARSEOP__LINE__); }
599 "__PATH__"           { count (0); return (PARSEOP__PATH__); }

602 "{ "                { count (0); return ('{'); }
603 "}"                  { count (0); return ('}'); }
604 " "                  { count (0); return (' '); }
605 "("                  { count (0); return ('('); }
606 ")"                  { count (0); return (')'); }

608 {NameSeg}           { char *s;
609                       count (0);
610                       s=malloc (ACPI_NAME_SIZE + 1);
611                       if (strcmp (AslCompilerText, "\\")
612                           {
613                           strcpy (s, "");
614                           AcpiUtStrupr (AslCompilerText);
615                           }
616                       memcpy (s, AslCompilerText, strlen (AslCompilerText));
617                       AslCompilerIval.s = s;
618                       DbgPrint (ASL_PARSE_OUTPUT, "NameSeg: %s\n", s);
619                       return (PARSEOP_NAMESEG); }

621 {NameString}         { char *s;
622                       count (0);
623                       s=malloc (strlen (AslCompilerText)+1);
624                       AcpiUtStrupr (AslCompilerText);
625                       strcpy (s, AslCompilerText);
626                       s[strlen (AslCompilerText)] = 0;
627                       AslCompilerIval.s = s;
628                       DbgPrint (ASL_PARSE_OUTPUT, "NameString: %s\n",
629                               return (PARSEOP_NAMESTRING); }

631 " " |                { count (1);
632 "/"                  { count (1);
633                       AslCompilerError ("Parse error, expecting ASL ke

635 .                    { count (1);
636                       sprintf (MsgBuffer,
637                               "Invalid character (0x%2.2X), expecting ASL
638                               *AslCompilerText);
639                       AslCompilerError (MsgBuffer); }

641 <<EOF>>              { if (AslPopInputFileStack ())
642                       yyterminate();
643                       else
644                       return (PARSEOP_INCLUDE_END); }

646 %%

648 /*! [End] no source code translation !*/

650 /*
651 * Bring in the scanner support routines
652 */
653 #include "aslsupport.1"

```



new/usr/src/common/acpica/compiler/aslcompiler.1

new/usr/src/common/acpica/compiler/aslcompiler.y

1

```
*****
126239 Fri Jan 17 07:29:33 2014
new/usr/src/common/acpica/compiler/aslcompiler.y
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 %{
2 /*****
3 *
4 * Module Name: aslcompiler.y - Bison/Yacc input file (ASL grammar and actions)
5 *
6 *****/
7
8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #include "aslcompiler.h"
46 #include <stdio.h>
47 #include <stdlib.h>
48 #include <string.h>
49 #include "acpi.h"
50 #include "accommon.h"
51
52 #define _COMPONENT          ACPI_COMPILER
53       ACPI_MODULE_NAME    ("aslparsing")
54
55 /*
56 * Global Notes:
57 *
58 * October 2005: The following list terms have been optimized (from the
59 * original ASL grammar in the ACPI specification) to force the immediate
```

new/usr/src/common/acpica/compiler/aslcompiler.y

2

```
60 * reduction of each list item so that the parse stack use doesn't increase on
61 * each list element and possibly overflow on very large lists (>4000 items).
62 * This dramatically reduces use of the parse stack overall.
63 *
64 * ArgList, TermList, ObjectList, ByteList, DWordList, PackageList,
65 * ResourceMacroList, and FieldUnitList
66 */
67
68 void * AslLocalAllocate (unsigned int Size);
69
70 /* Bison/yacc configuration */
71
72 #define static
73 #undef alloca
74 #define alloca          AslLocalAllocate
75 #define yyname          AslCompilername
76
77 #define YYINITDEPTH      600          /* State stack depth */
78 #define YYDEBUG          1           /* Enable debug output */
79 #define YYERROR_VERBOSE  1           /* Verbose error messages */
80
81 /* Define YYMALLOC/YYFREE to prevent redefinition errors */
82
83 #define YYMALLOC          malloc
84 #define YYFREE            free
85
86 /*
87 * The windows version of bison defines this incorrectly as "32768" (Not negativ
88 * We use a custom (edited binary) version of bison that defines YYFLAG as YYFBA
89 * instead (#define YYFBAD 32768), so we can define it correctly here.
90 *
91 * The problem is that if YYFLAG is positive, the extended syntax error messages
92 * are disabled.
93 */
94 #define YYFLAG            -32768
95
96 %}
97
98 /*
99 * Declare the type of values in the grammar
100 */
101 %union {
102     UINT64          i;
103     char            *s;
104     ACPI_PARSE_OBJECT *n;
105 }
106
107 /*! [Begin] no source code translation */
108
109 /*
110 * These shift/reduce conflicts are expected. There should be zero
111 * reduce/reduce conflicts.
112 */
113 %expect 86
114
115 /*****
116 *
117 * Token types: These are returned by the lexer
118 *
119 * NOTE: This list MUST match the AslKeywordMapping table found
120 *       in aslmap.c EXACTLY! Double check any changes!
121 *
122 *****/
123
124 %token <i> PARSEOP_ACCESSAS
125 %token <i> PARSEOP_ACCESSATTRIB_BLOCK
```

```
126 %token <i> PARSEOP_ACCESSATTRIB_BLOCK_CALL
127 %token <i> PARSEOP_ACCESSATTRIB_BYTE
128 %token <i> PARSEOP_ACCESSATTRIB_MULTIBYTE
129 %token <i> PARSEOP_ACCESSATTRIB_QUICK
130 %token <i> PARSEOP_ACCESSATTRIB_RAW_BYTES
131 %token <i> PARSEOP_ACCESSATTRIB_RAW_PROCESS
132 %token <i> PARSEOP_ACCESSATTRIB_SND_RCV
133 %token <i> PARSEOP_ACCESSATTRIB_WORD
134 %token <i> PARSEOP_ACCESSATTRIB_WORD_CALL
135 %token <i> PARSEOP_ACCESSTYPE_ANY
136 %token <i> PARSEOP_ACCESSTYPE_BUF
137 %token <i> PARSEOP_ACCESSTYPE_BYTE
138 %token <i> PARSEOP_ACCESSTYPE_DWORD
139 %token <i> PARSEOP_ACCESSTYPE_QWORD
140 %token <i> PARSEOP_ACCESSTYPE_WORD
141 %token <i> PARSEOP_ACQUIRE
142 %token <i> PARSEOP_ADD
143 %token <i> PARSEOP_ADDRESSINGMODE_7BIT
144 %token <i> PARSEOP_ADDRESSINGMODE_10BIT
145 %token <i> PARSEOP_ADDRESSTYPE ACPI
146 %token <i> PARSEOP_ADDRESSTYPE_MEMORY
147 %token <i> PARSEOP_ADDRESSTYPE_NVS
148 %token <i> PARSEOP_ADDRESSTYPE_RESERVED
149 %token <i> PARSEOP_ALIAS
150 %token <i> PARSEOP_AND
151 %token <i> PARSEOP_ARG0
152 %token <i> PARSEOP_ARG1
153 %token <i> PARSEOP_ARG2
154 %token <i> PARSEOP_ARG3
155 %token <i> PARSEOP_ARG4
156 %token <i> PARSEOP_ARG5
157 %token <i> PARSEOP_ARG6
158 %token <i> PARSEOP_BANKFIELD
159 %token <i> PARSEOP_BITSPERBYTE_EIGHT
160 %token <i> PARSEOP_BITSPERBYTE_FIVE
161 %token <i> PARSEOP_BITSPERBYTE_NINE
162 %token <i> PARSEOP_BITSPERBYTE_SEVEN
163 %token <i> PARSEOP_BITSPERBYTE_SIX
164 %token <i> PARSEOP_BREAK
165 %token <i> PARSEOP_BREAKPOINT
166 %token <i> PARSEOP_BUFFER
167 %token <i> PARSEOP_BUSMASTERTYPE_MASTER
168 %token <i> PARSEOP_BUSMASTERTYPE_NOTMASTER
169 %token <i> PARSEOP_BYTECONST
170 %token <i> PARSEOP_CASE
171 %token <i> PARSEOP_CLOCKPHASE_FIRST
172 %token <i> PARSEOP_CLOCKPHASE_SECOND
173 %token <i> PARSEOP_CLOCKPOLARITY_HIGH
174 %token <i> PARSEOP_CLOCKPOLARITY_LOW
175 %token <i> PARSEOP_CONCATENATE
176 %token <i> PARSEOP_CONCATENATERESTEMPLATE
177 %token <i> PARSEOP_CONDREFOF
178 %token <i> PARSEOP_CONNECTION
179 %token <i> PARSEOP_CONTINUE
180 %token <i> PARSEOP_COPYOBJECT
181 %token <i> PARSEOP_CREATEBITFIELD
182 %token <i> PARSEOP_CREATEBYTEFIELD
183 %token <i> PARSEOP_CREATEDWORDFIELD
184 %token <i> PARSEOP_CREATEFIELD
185 %token <i> PARSEOP_CREATEQWORDFIELD
186 %token <i> PARSEOP_CREATEWORDFIELD
187 %token <i> PARSEOP_DATABUFFER
188 %token <i> PARSEOP_DATATABLEREGION
189 %token <i> PARSEOP_DEBUG
190 %token <i> PARSEOP_DECODETYPE_POS
191 %token <i> PARSEOP_DECODETYPE_SUB
```

```
192 %token <i> PARSEOP_DECREMENT
193 %token <i> PARSEOP_DEFAULT
194 %token <i> PARSEOP_DEFAULT_ARG
195 %token <i> PARSEOP_DEFINITIONBLOCK
196 %token <i> PARSEOP_DEREFEOF
197 %token <i> PARSEOP_DEVICE
198 %token <i> PARSEOP_DEVICEPOLARITY_HIGH
199 %token <i> PARSEOP_DEVICEPOLARITY_LOW
200 %token <i> PARSEOP_DIVIDE
201 %token <i> PARSEOP_DMA
202 %token <i> PARSEOP_DMATYPE_A
203 %token <i> PARSEOP_DMATYPE_COMPATIBILITY
204 %token <i> PARSEOP_DMATYPE_B
205 %token <i> PARSEOP_DMATYPE_F
206 %token <i> PARSEOP_DWORDCONST
207 %token <i> PARSEOP_DWORDIO
208 %token <i> PARSEOP_DWORDMEMORY
209 %token <i> PARSEOP_DWORDSPACE
210 %token <i> PARSEOP_EISAID
211 %token <i> PARSEOP_ELSE
212 %token <i> PARSEOP_ELSEIF
213 %token <i> PARSEOP_ENDDEPENDENTFN
214 %token <i> PARSEOP_ENDIAN_BIG
215 %token <i> PARSEOP_ENDIAN_LITTLE
216 %token <i> PARSEOP_ENDTAG
217 %token <i> PARSEOP_ERRORNODE
218 %token <i> PARSEOP_EVENT
219 %token <i> PARSEOP_EXTENDEDIO
220 %token <i> PARSEOP_EXTENDEDMEMORY
221 %token <i> PARSEOP_EXTENDEDSPACE
222 %token <i> PARSEOP_EXTERNAL
223 %token <i> PARSEOP_FATAL
224 %token <i> PARSEOP_FIELD
225 %token <i> PARSEOP_FINDSETLEFTBIT
226 %token <i> PARSEOP_FINDSETRIGHTBIT
227 %token <i> PARSEOP_FIXEDDMA
228 %token <i> PARSEOP_FIXEDIO
229 %token <i> PARSEOP_FLOWCONTROL_HW
230 %token <i> PARSEOP_FLOWCONTROL_NONE
231 %token <i> PARSEOP_FLOWCONTROL_SW
232 %token <i> PARSEOP_FROMBCD
233 %token <i> PARSEOP_FUNCTION
234 %token <i> PARSEOP_GPIO_INT
235 %token <i> PARSEOP_GPIO_IO
236 %token <i> PARSEOP_I2C_SERIALBUS
237 %token <i> PARSEOP_IF
238 %token <i> PARSEOP_INCLUDE
239 %token <i> PARSEOP_INCLUDE_END
240 %token <i> PARSEOP_INCREMENT
241 %token <i> PARSEOP_INDEX
242 %token <i> PARSEOP_INDEXFIELD
243 %token <i> PARSEOP_INTEGER
244 %token <i> PARSEOP_INTERRUPT
245 %token <i> PARSEOP_INTLEVEL_ACTIVEBOTH
246 %token <i> PARSEOP_INTLEVEL_ACTIVEHIGH
247 %token <i> PARSEOP_INTLEVEL_ACTIVELOW
248 %token <i> PARSEOP_INTTYPE_EDGE
249 %token <i> PARSEOP_INTTYPE_LEVEL
250 %token <i> PARSEOP_IO
251 %token <i> PARSEOP_IODECODETYPE_10
252 %token <i> PARSEOP_IODECODETYPE_16
253 %token <i> PARSEOP_IORESTRICT_IN
254 %token <i> PARSEOP_IORESTRICT_NONE
255 %token <i> PARSEOP_IORESTRICT_OUT
256 %token <i> PARSEOP_IORESTRICT_PRESERVE
257 %token <i> PARSEOP_IRQ
```

```
258 %token <i> PARSEOP_IRQNOFLAGS
259 %token <i> PARSEOP_LAND
260 %token <i> PARSEOP_LEQUAL
261 %token <i> PARSEOP_LGREATER
262 %token <i> PARSEOP_LGREATEREQUAL
263 %token <i> PARSEOP_LLESS
264 %token <i> PARSEOP_LLESSEQUAL
265 %token <i> PARSEOP_LNOT
266 %token <i> PARSEOP_LNOTEQUAL
267 %token <i> PARSEOP_LOAD
268 %token <i> PARSEOP_LOADTABLE
269 %token <i> PARSEOP_LOCAL0
270 %token <i> PARSEOP_LOCAL1
271 %token <i> PARSEOP_LOCAL2
272 %token <i> PARSEOP_LOCAL3
273 %token <i> PARSEOP_LOCAL4
274 %token <i> PARSEOP_LOCAL5
275 %token <i> PARSEOP_LOCAL6
276 %token <i> PARSEOP_LOCAL7
277 %token <i> PARSEOP_LOCKRULE_LOCK
278 %token <i> PARSEOP_LOCKRULE_NOLOCK
279 %token <i> PARSEOP_LOR
280 %token <i> PARSEOP_MATCH
281 %token <i> PARSEOP_MATCHTYPE_MEQ
282 %token <i> PARSEOP_MATCHTYPE_MGE
283 %token <i> PARSEOP_MATCHTYPE_MGT
284 %token <i> PARSEOP_MATCHTYPE_MLE
285 %token <i> PARSEOP_MATCHTYPE_MLT
286 %token <i> PARSEOP_MATCHTYPE_MTR
287 %token <i> PARSEOP_MAXTYPE_FIXED
288 %token <i> PARSEOP_MAXTYPE_NOTFIXED
289 %token <i> PARSEOP_MEMORY24
290 %token <i> PARSEOP_MEMORY32
291 %token <i> PARSEOP_MEMORY32FIXED
292 %token <i> PARSEOP_MEMTYPE_CACHEABLE
293 %token <i> PARSEOP_MEMTYPE_NONCACHEABLE
294 %token <i> PARSEOP_MEMTYPE_PREFETCHABLE
295 %token <i> PARSEOP_MEMTYPE_WRITECOMBINING
296 %token <i> PARSEOP_METHOD
297 %token <i> PARSEOP_METHODCALL
298 %token <i> PARSEOP_MID
299 %token <i> PARSEOP_MINTYPE_FIXED
300 %token <i> PARSEOP_MINTYPE_NOTFIXED
301 %token <i> PARSEOP_MOD
302 %token <i> PARSEOP_MULTIPLY
303 %token <i> PARSEOP_MUTEX
304 %token <i> PARSEOP_NAME
305 %token <s> PARSEOP_NAMESEG
306 %token <s> PARSEOP_NAMESTRING
307 %token <i> PARSEOP_NAND
308 %token <i> PARSEOP_NOOP
309 %token <i> PARSEOP_NOR
310 %token <i> PARSEOP_NOT
311 %token <i> PARSEOP_NOTIFY
312 %token <i> PARSEOP_OBJECTTYPE
313 %token <i> PARSEOP_OBJECTTYPE_BFF
314 %token <i> PARSEOP_OBJECTTYPE_BUF
315 %token <i> PARSEOP_OBJECTTYPE_DDB
316 %token <i> PARSEOP_OBJECTTYPE_DEV
317 %token <i> PARSEOP_OBJECTTYPE_EVT
318 %token <i> PARSEOP_OBJECTTYPE_FLD
319 %token <i> PARSEOP_OBJECTTYPE_INT
320 %token <i> PARSEOP_OBJECTTYPE_MTH
321 %token <i> PARSEOP_OBJECTTYPE_MTX
322 %token <i> PARSEOP_OBJECTTYPE_OPR
323 %token <i> PARSEOP_OBJECTTYPE_PKG
```

```
324 %token <i> PARSEOP_OBJECTTYPE_POW
325 %token <i> PARSEOP_OBJECTTYPE_PRO
326 %token <i> PARSEOP_OBJECTTYPE_STR
327 %token <i> PARSEOP_OBJECTTYPE_THZ
328 %token <i> PARSEOP_OBJECTTYPE_UNK
329 %token <i> PARSEOP_OFFSET
330 %token <i> PARSEOP_ONE
331 %token <i> PARSEOP_ONES
332 %token <i> PARSEOP_OPERATIONREGION
333 %token <i> PARSEOP_OR
334 %token <i> PARSEOP_PACKAGE
335 %token <i> PARSEOP_PACKAGE_LENGTH
336 %token <i> PARSEOP_PARITYTYPE_EVEN
337 %token <i> PARSEOP_PARITYTYPE_MARK
338 %token <i> PARSEOP_PARITYTYPE_NONE
339 %token <i> PARSEOP_PARITYTYPE_ODD
340 %token <i> PARSEOP_PARITYTYPE_SPACE
341 %token <i> PARSEOP_PIN_NOPULL
342 %token <i> PARSEOP_PIN_PULLDEFAULT
343 %token <i> PARSEOP_PIN_PULLDOWN
344 %token <i> PARSEOP_PIN_PULLUP
345 %token <i> PARSEOP_POWERRESOURCE
346 %token <i> PARSEOP_PROCESSOR
347 %token <i> PARSEOP_QWORDCONST
348 %token <i> PARSEOP_QWORDIO
349 %token <i> PARSEOP_QWORDMEMORY
350 %token <i> PARSEOP_QWORDSPACE
351 %token <i> PARSEOP_RANGETYPE_ENTIRE
352 %token <i> PARSEOP_RANGETYPE_ISAONLY
353 %token <i> PARSEOP_RANGETYPE_NONISAONLY
354 %token <i> PARSEOP_RAW_DATA
355 %token <i> PARSEOP_READWRITETYPE_BOTH
356 %token <i> PARSEOP_READWRITETYPE_READONLY
357 %token <i> PARSEOP_REFOF
358 %token <i> PARSEOP_REGIONSPACE_CMOS
359 %token <i> PARSEOP_REGIONSPACE_EC
360 %token <i> PARSEOP_REGIONSPACE_FFIXEDHW
361 %token <i> PARSEOP_REGIONSPACE_GPIO
362 %token <i> PARSEOP_REGIONSPACE_GSBUS
363 %token <i> PARSEOP_REGIONSPACE_IO
364 %token <i> PARSEOP_REGIONSPACE_IPMI
365 %token <i> PARSEOP_REGIONSPACE_MEM
366 %token <i> PARSEOP_REGIONSPACE_PCC
367 %token <i> PARSEOP_REGIONSPACE_PCI
368 %token <i> PARSEOP_REGIONSPACE_PCIBAR
369 %token <i> PARSEOP_REGIONSPACE_SMBUS
370 %token <i> PARSEOP_REGISTER
371 %token <i> PARSEOP_RELEASE
372 %token <i> PARSEOP_RESERVED_BYTES
373 %token <i> PARSEOP_RESET
374 %token <i> PARSEOP_RESOURCETEMPLATE
375 %token <i> PARSEOP_RESOURCETYPE_CONSUMER
376 %token <i> PARSEOP_RESOURCETYPE_PRODUCER
377 %token <i> PARSEOP_RETURN
378 %token <i> PARSEOP_REVISION
379 %token <i> PARSEOP_SCOPE
380 %token <i> PARSEOP_SERIALIZERULE_NOTSERIAL
381 %token <i> PARSEOP_SERIALIZERULE_SERIAL
382 %token <i> PARSEOP_SHARETYPE_EXCLUSIVE
383 %token <i> PARSEOP_SHARETYPE_EXCLUSIVEWAKE
384 %token <i> PARSEOP_SHARETYPE_SHARED
385 %token <i> PARSEOP_SHARETYPE_SHAREDWAKE
386 %token <i> PARSEOP_SHIFTLIFT
387 %token <i> PARSEOP_SHIFTRIGHT
388 %token <i> PARSEOP_SIGNAL
389 %token <i> PARSEOP_SIZEOF
```

```

390 %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_WORDSSPACE
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> OptionalBufferLast
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
787 %type <n> OptionalType_Last
788 %type <n> OptionalWireMode
789 %type <n> OptionalWordConst
790 %type <n> OptionalWordConstExpr
791 %type <n> OptionalXferSize

793 %%
794 /*****
795 *
796 * Production rules start here
797 *
798 *****/

800 /*
801 * ASL Names
802 */

805 /*
806 * Root rule. Allow multiple #line directives before the definition block
807 * to handle output from preprocessors
808 */
809 ASLCode
810 : DefinitionBlockTerm
811 | error {YYABORT; $$ = NULL;}
812 ;

814 /*
815 * Blocks, Data, and Opcodes
816 */

818 /*
819 * Note concerning support for "module-level code".
820 *
821 * ACPI 1.0 allowed Type1 and Type2 executable opcodes outside of control
822 * methods (the so-called module-level code.) This support was explicitly
823 * removed in ACPI 2.0, but this type of code continues to be created by
824 * BIOS vendors. In order to support the disassembly and recompilation of
825 * such code (and the porting of ASL code to iASL), iASL supports this
826 * code in violation of the current ACPI specification.
827 *
828 * The grammar change to support module-level code is to revert the
829 * {ObjectList} portion of the DefinitionBlockTerm in ACPI 2.0 to the
830 * original use of {TermList} instead (see below.) This allows the use
831 * of Type1 and Type2 opcodes at module level.
832 */
833 DefinitionBlockTerm
834 : PARSEOP_DEFINITIONBLOCK '(' {<n>$ = TrCreateLeafNode (PARSEOP_DEFINITIO
835   String ','
836   String ','
837   ByteConst ','
838   String ','
839   String ','
840   DWordConst
841   ')' {TrSetEndLineNumber ($<n>3);}
842   '{' TermList '}' {$$ = TrLinkChildren ($<n>3,7,$4,$6,$8,$10,$
843 ;

845 /* ACPI 3.0 -- allow semicolons between terms */

847 TermList
848 : {$$ = NULL;}
849 | TermList Term {$$ = TrLinkPeerNode (TrSetNodeFlags ($1, NO
850 | TermList Term ';' {$$ = TrLinkPeerNode (TrSetNodeFlags ($1, NO
851 | TermList ';' Term {$$ = TrLinkPeerNode (TrSetNodeFlags ($1, NO

```

```

852 | TermList ';' Term ';' {$$ = TrLinkPeerNode (TrSetNodeFlags ($1, NO
853 ;

855 Term
856 : Object {}
857 | Type1Opcode {}
858 | Type2Opcode {}
859 | Type2IntegerOpcode {}
860 | Type2StringOpcode {}
861 | Type2BufferOpcode {}
862 | Type2BufferOrStringOpcode {}
863 | error {$$ = AslDoError(); yyclearin;}
864 ;

866 CompilerDirective
867 : IncludeTerm {}
868 | ExternalTerm {}
869 ;

871 ObjectList
872 : {$$ = NULL;}
873 | ObjectList Object {$$ = TrLinkPeerNode ($1,$2);}
874 | error {$$ = AslDoError(); yyclearin;}
875 ;

877 Object
878 : CompilerDirective {}
879 | NamedObject {}
880 | NameSpaceModifier {}
881 ;

883 DataObject
884 : BufferData {}
885 | PackageData {}
886 | IntegerData {}
887 | StringData {}
888 ;

890 BufferData
891 : Type5Opcode {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME
892 | Type2BufferOrStringOpcode {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME
893 | Type2BufferOpcode {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME
894 | BufferTerm {}
895 ;

897 PackageData
898 : PackageTerm {}
899 ;

901 IntegerData
902 : Type2IntegerOpcode {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME
903 | Type3Opcode {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME
904 | Integer {}
905 | ConstTerm {}
906 ;

908 StringData
909 : Type2StringOpcode {$$ = TrSetNodeFlags ($1, NODE_COMPILE_TIME
910 | String {}
911 ;

913 NamedObject
914 : BankFieldTerm {}
915 | CreateBitFieldTerm {}
916 | CreateByteFieldTerm {}
917 | CreateDWordFieldTerm {}

```



```

918 | 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 '('
943   | ArgList ')'
944 ;

946 ArgList
947 :
948   | TermArg
949   | ArgList ','
950   | ArgList ','
951   | TermArg
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 :
973   | ','
974   | ',' SuperName
975 ;

977 RequiredTarget
978 : ',' SuperName
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 :
991   | ObjectTypeKeyword      {$$ = NULL;}
992   | ParameterTypePackage ','
993   | ObjectTypeKeyword      {$$ = TrLinkPeerNodes (2,$1,$3);}
994 ;

996 ParameterTypePackageList
997 :
998   | ObjectTypeKeyword      {$$ = NULL;}
999   | '{' ParameterTypePackage '}' {$$ = $1;}
1000 ;

1002 OptionalParameterTypePackage
1003 :
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 :
1017   | ObjectTypeKeyword      {$$ = NULL;}
1018   | '{' ParameterTypesPackage '}' {$$ = $1;}
1019 ;

1021 OptionalParameterTypesPackage
1022 :
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 '}' { $$ = TrLinkChildren ($<n>3,7,$4,$5,$6,$8,$1
1186      | PARSEOP_BANKFIELD '(' { $$ = TrLinkChildren ($<n>3,7,$4,$5,$6,$8,$1
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
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   ')' { $$ = TrCreateNode (PARSEOP_CONNECTION,1,$3)
1232 | PARSEOP_CONNECTION '('
1233   ResourceMacroTerm
1234   ')' { $$ = TrLinkChildren ($<n>3, 1,
1235   TrLinkChildren (TrCreateLeafNode (PA
1236   TrCreateLeafNode (PARSEOP_DEFAULT
1237   TrCreateLeafNode (PARSEOP_DEFAULT
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      '',' { $$ = TrLinkChildren ($<n>3,3,$4,$5,TrSetNod
1249      | PARSEOP_CREATEBITFIELD '(' { $<n>$ = TrCreateLeafNode (PARSEOP_CREATEBYT
1250      error ')' { $$ = 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 CreateWordFieldTerm
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 '}'
1320 | PARSEOP_DEVICE '(' {<n>$ = TrLinkChildren (<n>3,2,TrSetNodeFlags
1321   error ')' {<n>$ = AslDoError(); yyclearin;}
1322 ;

1324 EventTerm
1325 : PARSEOP_EVENT '(' {<n>$ = TrCreateLeafNode (PARSEOP_EVENT);}
1326   NameString
1327   ')'
1328 | PARSEOP_EVENT '(' {<n>$ = TrLinkChildren (<n>3,1,TrSetNodeFlags
1329   error ')' {<n>$ = AslDoError(); yyclearin;}
1330 ;

1332 FieldTerm
1333 : PARSEOP_FIELD '(' {<n>$ = TrCreateLeafNode (PARSEOP_FIELD);}
1334   NameString
1335   ',' AccessTypeKeyword
1336   ',' LockRuleKeyword
1337   ',' UpdateRuleKeyword
1338   ',' '{'
1339   FieldUnitList '}' {<n>$ = TrLinkChildren (<n>3,5,$4,$6,$8,$10,$
1340 | PARSEOP_FIELD '(' {<n>$ = TrLinkChildren (<n>3,1,TrSetNodeFlags
1341   error ')' '{' error '}' {<n>$ = AslDoError(); yyclearin;}
1342 ;

1344 FunctionTerm
1345 : PARSEOP_FUNCTION '(' {<n>$ = TrCreateLeafNode (PARSEOP_METHOD);}
1346   NameString
1347   OptionalParameterTypePackage
1348   OptionalParameterTypesPackage
1349   ')' '{'
1350   TermList '}' {<n>$ = TrLinkChildren (<n>3,7,TrSetNodeFlags
1351   TrCreateValuedLeafNode (PARSEOP_BYTECONS
1352   TrCreateLeafNode (PARSEOP_SERIALIZERULE_
1353   TrCreateValuedLeafNode (PARSEOP_BYTECONS
1354 | PARSEOP_FUNCTION '(' {<n>$ = TrLinkChildren (<n>3,5,$4,$6,$8,$10,$
1355   error ')' '{' error '}' {<n>$ = 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 '}' {<n>$ = TrLinkChildren (<n>3,6,$4,$5,$7,$9,$1
1367 | PARSEOP_INDEXFIELD '(' {<n>$ = TrLinkChildren (<n>3,5,$4,$6,$8,$10,$
1368   error ')' '{' error '}' {<n>$ = 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 '}' {<n>$ = TrLinkChildren (<n>3,7,TrSetNodeFlags
1381 | PARSEOP_METHOD '(' {<n>$ = TrLinkChildren (<n>3,7,TrSetNodeFlags
1382   error ')' {<n>$ = AslDoError(); yyclearin;}
1383 ;

1385 MutexTerm
1386 : PARSEOP_MUTEX '(' {<n>$ = TrCreateLeafNode (PARSEOP_MUTEX);}
1387   NameString
1388   ',' ByteConstExpr
1389   ')'
1390 | PARSEOP_MUTEX '(' {<n>$ = TrLinkChildren (<n>3,2,TrSetNodeFlags
1391   error ')' {<n>$ = AslDoError(); yyclearin;}
1392 ;

1394 OpRegionTerm
1395 : PARSEOP_OPERATIONREGION '(' {<n>$ = TrCreateLeafNode (PARSEOP_OPERATION
1396   NameString
1397   ',' OpRegionSpaceIdTerm
1398   TermArgItem
1399   TermArgItem
1400   ')'
1401 | PARSEOP_OPERATIONREGION '(' {<n>$ = TrLinkChildren (<n>3,4,TrSetNodeFlags
1402   error ')' {<n>$ = AslDoError(); yyclearin;}
1403 ;

1405 OpRegionSpaceIdTerm
1406 : RegionSpaceKeyword {}
1407 | ByteConst {<n>$ = UtCheckIntegerRange ($1, 0x80, 0xFF);}
1408 ;

1410 PowerResTerm
1411 : PARSEOP_POWERRESOURCE '(' {<n>$ = TrCreateLeafNode (PARSEOP_POWERRESO
1412   NameString
1413   ',' ByteConstExpr
1414   ',' WordConstExpr
1415   ',' '{'
1416   ObjectList '}' {<n>$ = TrLinkChildren (<n>3,4,TrSetNodeFlags
1417 | PARSEOP_POWERRESOURCE '(' {<n>$ = TrLinkChildren (<n>3,4,TrSetNodeFlags
1418   error ')' {<n>$ = AslDoError(); yyclearin;}
1419 ;

1421 ProcessorTerm
1422 : PARSEOP_PROCESSOR '(' {<n>$ = TrCreateLeafNode (PARSEOP_PROCESSOR
1423   NameString
1424   ',' ByteConstExpr
1425   OptionalDWordConstExpr
1426   OptionalByteConstExpr
1427   ')' '{'
1428   ObjectList '}' {<n>$ = TrLinkChildren (<n>3,5,TrSetNodeFlags
1429 | PARSEOP_PROCESSOR '(' {<n>$ = TrLinkChildren (<n>3,5,TrSetNodeFlags
1430   error ')' {<n>$ = AslDoError(); yyclearin;}
1431 ;

1433 ThermalZoneTerm
1434 : PARSEOP_THERMALZONE '(' {<n>$ = TrCreateLeafNode (PARSEOP_THERMALZO
1435   NameString
1436   ',' '{'
1437   ObjectList '}' {<n>$ = TrLinkChildren (<n>3,2,TrSetNodeFlags
1438 | PARSEOP_THERMALZONE '(' {<n>$ = TrLinkChildren (<n>3,2,TrSetNodeFlags
1439   error ')' {<n>$ = AslDoError(); yyclearin;}
1440 ;

1443 /***** Namespace modifiers *****/

```

```

1446 AliasTerm
1447 : PARSEOP_ALIAS '('
1448   NameString
1449   NameStringItem
1450   ')'
1451 | PARSEOP_ALIAS '('
1452   error ')'
1453 ;

1455 NameTerm
1456 : PARSEOP_NAME '('
1457   NameString
1458   ',' DataObject
1459   ')'
1460 | PARSEOP_NAME '('
1461   error ')'
1462 ;

1464 ScopeTerm
1465 : PARSEOP_SCOPE '('
1466   NameString
1467   ',' '{'
1468   ObjectList '}'
1469 | PARSEOP_SCOPE '('
1470   error ')'
1471 ;

1474 /***** Type 1 opcodes *****/

1477 BreakTerm
1478 : PARSEOP_BREAK
1479 ;

1481 BreakPointTerm
1482 : PARSEOP_BREAKPOINT
1483 ;

1485 ContinueTerm
1486 : PARSEOP_CONTINUE
1487 ;

1489 FatalTerm
1490 : PARSEOP_FATAL '('
1491   ByteConstExpr
1492   ',' DWordConstExpr
1493   TermArgItem
1494   ')'
1495 | PARSEOP_FATAL '('
1496   error ')'
1497 ;

1499 IfElseTerm
1500 : IfTerm ElseTerm
1501 ;

1503 IfTerm
1504 : PARSEOP_IF '('
1505   TermArg
1506   ',' '{'
1507   TermList '}'
1508 | PARSEOP_IF '('
1509   error ')'
1510 ;

```

```

1513 ElseTerm
1514 :
1515 | PARSEOP_ELSE '{'
1516   TermList '}'
1517 | PARSEOP_ELSE '{'
1518   error '}'
1519 ;

1521 | PARSEOP_ELSE
1522   error
1523 ;

1524 | PARSEOP_ELSEIF '('
1525   TermArg
1526   ',' '{'
1527   TermList '}'
1528   ElseTerm
1529 ;

1531 | PARSEOP_ELSEIF '('
1532   error ')'
1533 ;

1534 | PARSEOP_ELSEIF
1535   error
1536 ;

1538 LoadTerm
1539 : PARSEOP_LOAD '('
1540   NameString
1541   RequiredTarget
1542   ')'
1543 | PARSEOP_LOAD '('
1544   error ')'
1545 ;

1547 NoOpTerm
1548 : PARSEOP_NOOP
1549 ;

1551 NotifyTerm
1552 : PARSEOP_NOTIFY '('
1553   SuperName
1554   TermArgItem
1555   ')'
1556 | PARSEOP_NOTIFY '('
1557   error ')'
1558 ;

1560 ReleaseTerm
1561 : PARSEOP_RELEASE '('
1562   SuperName
1563   ')'
1564 | PARSEOP_RELEASE '('
1565   error ')'
1566 ;

1568 ResetTerm
1569 : PARSEOP_RESET '('
1570   SuperName
1571   ')'
1572 | PARSEOP_RESET '('
1573   error ')'
1574 ;

1576 ReturnTerm
1577 : PARSEOP_RETURN '('

```

```

1578     OptionalReturnArg
1579     ' '
1580     | PARSEOP_RETURN
1581     | PARSEOP_RETURN '('
1582     | error ')'
1583     ;

1585 SignalTerm
1586 : PARSEOP_SIGNAL '('
1587   SuperName
1588   ')'
1589 | PARSEOP_SIGNAL '('
1590   error ')'
1591 ;

1593 SleepTerm
1594 : PARSEOP_SLEEP '('
1595   TermArg
1596   ')'
1597 | PARSEOP_SLEEP '('
1598   error ')'
1599 ;

1601 StallTerm
1602 : PARSEOP_STALL '('
1603   TermArg
1604   ')'
1605 | PARSEOP_STALL '('
1606   error ')'
1607 ;

1609 SwitchTerm
1610 : PARSEOP_SWITCH '('
1611   TermArg
1612   ' ' '{ '
1613   CaseDefaultTermList '}'
1614   ' ' '{ '
1615   | PARSEOP_SWITCH '('
1616   error ')'
1617 ;

1619 /*
1620  * Case-Default list; allow only one Default term and unlimited Case terms
1621  */

1623 CaseDefaultTermList
1624 :
1625   | CaseTerm {}
1626   | DefaultTerm {}
1627   | CaseDefaultTermList
1628   | CaseTerm
1629   | CaseDefaultTermList
1630   | DefaultTerm

1632 /* Original - attempts to force zero or one default term within the switch */

1634 /*
1635 CaseDefaultTermList
1636 :
1637   | CaseTermList
1638   | DefaultTerm
1639   | CaseTermList
1640   | CaseTermList
1641   | CaseTerm
1642 ;

```

```

1644 CaseTermList
1645 :
1646   | CaseTerm
1647   | CaseTermList
1648   | CaseTerm
1649 ;
1650 /*

1652 CaseTerm
1653 : PARSEOP_CASE '('
1654   DataObject
1655   ' ' '{ '
1656   TermList '}'
1657 | PARSEOP_CASE '('
1658   error ')'
1659 ;

1661 DefaultTerm
1662 : PARSEOP_DEFAULT '{ '
1663   TermList '}'
1664 | PARSEOP_DEFAULT '{ '
1665   error '}'
1666 ;

1668 UnloadTerm
1669 : PARSEOP_UNLOAD '('
1670   SuperName
1671   ' ' '{ '
1672   | PARSEOP_UNLOAD '('
1673   error ')'
1674 ;

1676 WhileTerm
1677 : PARSEOP_WHILE '('
1678   TermArg
1679   ' ' '{ ' TermList '}'
1680   ' ' '{ '
1681   | PARSEOP_WHILE '('
1682   error ')'
1683 ;

1686 /***** Type 2 opcodes *****/

1688 AcquireTerm
1689 : PARSEOP_ACQUIRE '('
1690   SuperName
1691   ' ' WordConstExpr
1692   ' ' '{ '
1693   | PARSEOP_ACQUIRE '('
1694   error ')'
1695 ;

1697 AddTerm
1698 : PARSEOP_ADD '('
1699   TermArg
1700   TermArgItem
1701   Target
1702   ' ' '{ '
1703   | PARSEOP_ADD '('
1704   error ')'
1705 ;

1707 AndTerm
1708 : PARSEOP_AND '('
1709   TermArg

```

```

1710      TermArgItem
1711      Target
1712      '('
1713      | PARSEOP_AND '('
1714      error ')'
1715      ;

1717 ConcatTerm
1718 : PARSEOP_CONCATENATE '('
1719   TermArg
1720   TermArgItem
1721   Target
1722   ')'
1723 | PARSEOP_CONCATENATE '('
1724 error ')'
1725 ;

1727 ConcatResTerm
1728 : PARSEOP_CONCATENATERESTEMPLATE '('
1729   TermArg
1730   TermArgItem
1731   Target
1732   ')'
1733 | PARSEOP_CONCATENATERESTEMPLATE '('
1734 error ')'
1735 ;

1737 CondRefOfTerm
1738 : PARSEOP_CONDREFOF '('
1739   SuperName
1740   Target
1741   ')'
1742 | PARSEOP_CONDREFOF '('
1743 error ')'
1744 ;

1746 CopyObjectTerm
1747 : PARSEOP_COPYOBJECT '('
1748   TermArg
1749   ',' SimpleTarget
1750   ')'
1751 | PARSEOP_COPYOBJECT '('
1752 error ')'
1753 ;

1755 DecTerm
1756 : PARSEOP_DECREMENT '('
1757   SuperName
1758   ')'
1759 | PARSEOP_DECREMENT '('
1760 error ')'
1761 ;

1763 DerefofTerm
1764 : PARSEOP_DEREOF '('
1765   TermArg
1766   ')'
1767 | PARSEOP_DEREOF '('
1768 error ')'
1769 ;

1771 DivideTerm
1772 : PARSEOP_DIVIDE '('
1773   TermArg
1774   TermArgItem
1775   Target

```

```

1776      Target
1777      ')'
1778      | PARSEOP_DIVIDE '('
1779      error ')'
1780      ;

1782 FindSetLeftBitTerm
1783 : PARSEOP_FINDSETLEFTBIT '('
1784   TermArg
1785   Target
1786   ')'
1787 | PARSEOP_FINDSETLEFTBIT '('
1788 error ')'
1789 ;

1791 FindSetRightBitTerm
1792 : PARSEOP_FINDSETRIGHTBIT '('
1793   TermArg
1794   Target
1795   ')'
1796 | PARSEOP_FINDSETRIGHTBIT '('
1797 error ')'
1798 ;

1800 FromBCDTerm
1801 : PARSEOP_FROMBCD '('
1802   TermArg
1803   Target
1804   ')'
1805 | PARSEOP_FROMBCD '('
1806 error ')'
1807 ;

1809 IncTerm
1810 : PARSEOP_INCREMENT '('
1811   SuperName
1812   ')'
1813 | PARSEOP_INCREMENT '('
1814 error ')'
1815 ;

1817 IndexTerm
1818 : PARSEOP_INDEX '('
1819   TermArg
1820   TermArgItem
1821   Target
1822   ')'
1823 | PARSEOP_INDEX '('
1824 error ')'
1825 ;

1827 LAndTerm
1828 : PARSEOP_LAND '('
1829   TermArg
1830   TermArgItem
1831   ')'
1832 | PARSEOP_LAND '('
1833 error ')'
1834 ;

1836 LEqualTerm
1837 : PARSEOP_LEQUAL '('
1838   TermArg
1839   TermArgItem
1840   ')'
1841 | PARSEOP_LEQUAL '('

```

```

1842         error ')'          { $$ = AslDoError(); yyclearin; }
1843     ;

1845 LGreaterTerm
1846 : PARSEOP_LGREATER '('      { $<n>$ = TrCreateLeafNode (PARSEOP_LGREATER)
1847     TermArg                  { $$ = TrLinkChildren ($<n>3,2,$4,$5); }
1848     TermArgItem              { $$ = TrLinkChildren ($<n>3,2,$4,$5); }
1849     ')'
1850 | PARSEOP_LGREATER '('      { $$ = AslDoError(); yyclearin; }
1851     error ')'
1852 ;

1854 LGreaterEqualTerm
1855 : PARSEOP_LGREATEREQUAL '(' { $<n>$ = TrCreateLeafNode (PARSEOP_LLESS); }
1856     TermArg                  { $$ = TrCreateNode (PARSEOP_LNOT, 1, TrLinkC
1857     TermArgItem              { $$ = TrLinkChildren ($<n>3,2,$4,$5); }
1858     ')'
1859 | PARSEOP_LGREATEREQUAL '(' { $$ = AslDoError(); yyclearin; }
1860     error ')'
1861 ;

1863 LLessTerm
1864 : PARSEOP_LLESS '('         { $<n>$ = TrCreateLeafNode (PARSEOP_LLESS); }
1865     TermArg                  { $$ = TrLinkChildren ($<n>3,2,$4,$5); }
1866     TermArgItem              { $$ = TrLinkChildren ($<n>3,2,$4,$5); }
1867     ')'
1868 | PARSEOP_LLESS '('         { $$ = AslDoError(); yyclearin; }
1869     error ')'
1870 ;

1872 LLessEqualTerm
1873 : PARSEOP_LLESSEQUAL '('    { $<n>$ = TrCreateLeafNode (PARSEOP_LGREATER)
1874     TermArg                  { $$ = TrLinkChildren ($<n>3,4,$4,$5,$6,$7); }
1875     TermArgItem              { $$ = TrLinkChildren ($<n>3,4,$4,$5,$6,$7); }
1876     ')'
1877 | PARSEOP_LLESSEQUAL '('    { $$ = AslDoError(); yyclearin; }
1878     error ')'
1879 ;

1881 LNotTerm
1882 : PARSEOP_LNOT '('          { $<n>$ = TrCreateLeafNode (PARSEOP_LNOT); }
1883     TermArg                  { $$ = TrLinkChildren ($<n>3,1,$4); }
1884     ')'
1885 | PARSEOP_LNOT '('          { $$ = AslDoError(); yyclearin; }
1886     error ')'
1887 ;

1889 LNotEqualTerm
1890 : PARSEOP_LNOTEQUAL '('     { $<n>$ = TrCreateLeafNode (PARSEOP_LEQUAL); }
1891     TermArg                  { $$ = TrCreateNode (PARSEOP_LNOT, 1, TrLinkC
1892     TermArgItem              { $$ = TrLinkChildren ($<n>3,3,$4,$5,$6); }
1893     ')'
1894 | PARSEOP_LNOTEQUAL '('     { $$ = AslDoError(); yyclearin; }
1895     error ')'
1896 ;

1898 LoadTableTerm
1899 : PARSEOP_LOADTABLE '('     { $<n>$ = TrCreateLeafNode (PARSEOP_LOADTABLE)
1900     TermArg                  { $$ = TrLinkChildren ($<n>3,6,$4,$5,$6,$7,$8
1901     TermArgItem              {
1902     TermArgItem              {
1903     OptionalListString
1904     OptionalListString
1905     OptionalReference
1906     ')'
1907 | PARSEOP_LOADTABLE '('

```

```

1908         error ')'          { $$ = AslDoError(); yyclearin; }
1909     ;

1911 LOrTerm
1912 : PARSEOP_LOR '('           { $<n>$ = TrCreateLeafNode (PARSEOP_LOR); }
1913     TermArg                  { $$ = TrLinkChildren ($<n>3,2,$4,$5); }
1914     TermArgItem              { $$ = TrLinkChildren ($<n>3,2,$4,$5); }
1915     ')'
1916 | PARSEOP_LOR '('           { $$ = AslDoError(); yyclearin; }
1917     error ')'
1918 ;

1920 MatchTerm
1921 : PARSEOP_MATCH '('         { $<n>$ = TrCreateLeafNode (PARSEOP_MATCH); }
1922     TermArg                  {
1923     ',' MatchOpKeyword
1924     TermArgItem
1925     ',' MatchOpKeyword
1926     TermArgItem
1927     TermArgItem
1928     ')'
1929 | PARSEOP_MATCH '('         { $$ = TrLinkChildren ($<n>3,6,$4,$6,$7,$9,$1
1930     error ')'
1931 ;

1933 MidTerm
1934 : PARSEOP_MID '('           { $<n>$ = TrCreateLeafNode (PARSEOP_MID); }
1935     TermArg                  {
1936     TermArgItem
1937     TermArgItem
1938     Target
1939     ')'
1940 | PARSEOP_MID '('           { $$ = TrLinkChildren ($<n>3,4,$4,$5,$6,$7); }
1941     error ')'
1942 ;

1944 ModTerm
1945 : PARSEOP_MOD '('           { $<n>$ = TrCreateLeafNode (PARSEOP_MOD); }
1946     TermArg                  {
1947     TermArgItem
1948     Target
1949     ')'
1950 | PARSEOP_MOD '('           { $$ = TrLinkChildren ($<n>3,3,$4,$5,$6); }
1951     error ')'
1952 ;

1954 MultiplyTerm
1955 : PARSEOP_MULTIPLY '('      { $<n>$ = TrCreateLeafNode (PARSEOP_MULTIPLY)
1956     TermArg                  {
1957     TermArgItem
1958     Target
1959     ')'
1960 | PARSEOP_MULTIPLY '('      { $$ = TrLinkChildren ($<n>3,3,$4,$5,$6); }
1961     error ')'
1962 ;

1964 NAndTerm
1965 : PARSEOP_NAND '('          { $<n>$ = TrCreateLeafNode (PARSEOP_NAND); }
1966     TermArg                  {
1967     TermArgItem
1968     Target
1969     ')'
1970 | PARSEOP_NAND '('          { $$ = TrLinkChildren ($<n>3,3,$4,$5,$6); }
1971     error ')'
1972 ;

```



```

1974 NOrTerm
1975 : PARSEOP_NOR '('
1976     TermArg
1977     TermArgItem
1978     Target
1979     ')'
1980 | PARSEOP_NOR '('
1981     error ')'
1982 ;

1984 NotTerm
1985 : PARSEOP_NOT '('
1986     TermArg
1987     Target
1988     ')'
1989 | PARSEOP_NOT '('
1990     error ')'
1991 ;

1993 ObjectTypeTerm
1994 : PARSEOP_OBJECTTYPE '('
1995     ObjectTypeName
1996     ')'
1997 | PARSEOP_OBJECTTYPE '('
1998     error ')'
1999 ;

2001 OrTerm
2002 : PARSEOP_OR '('
2003     TermArg
2004     TermArgItem
2005     Target
2006     ')'
2007 | PARSEOP_OR '('
2008     error ')'
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 '('
2017     SuperName
2018     ')'
2019 | PARSEOP_REFOF '('
2020     error ')'
2021 ;

2023 ShiftLeftTerm
2024 : PARSEOP_SHIFLEFT '('
2025     TermArg
2026     TermArgItem
2027     Target
2028     ')'
2029 | PARSEOP_SHIFLEFT '('
2030     error ')'
2031 ;

2033 ShiftRightTerm
2034 : PARSEOP_SHIFTRIGHT '('
2035     TermArg
2036     TermArgItem
2037     Target
2038     ')'
2039 | PARSEOP_SHIFTRIGHT '('

```

```

2040     error ')'
2041 ;

2043 SizeOfTerm
2044 : PARSEOP_SIZEOF '('
2045     SuperName
2046     ')'
2047 | PARSEOP_SIZEOF '('
2048     error ')'
2049 ;

2051 StoreTerm
2052 : PARSEOP_STORE '('
2053     TermArg
2054     ',' SuperName
2055     ')'
2056 | PARSEOP_STORE '('
2057     error ')'
2058 ;

2060 SubtractTerm
2061 : PARSEOP_SUBTRACT '('
2062     TermArg
2063     TermArgItem
2064     Target
2065     ')'
2066 | PARSEOP_SUBTRACT '('
2067     error ')'
2068 ;

2070 TimerTerm
2071 : PARSEOP_TIMER '('
2072     ')'
2073 | PARSEOP_TIMER
2074 | PARSEOP_TIMER '('
2075     error ')'
2076 ;

2078 ToBCDTerm
2079 : PARSEOP_TOBCD '('
2080     TermArg
2081     Target
2082     ')'
2083 | PARSEOP_TOBCD '('
2084     error ')'
2085 ;

2087 ToBufferTerm
2088 : PARSEOP_TOBUFFER '('
2089     TermArg
2090     Target
2091     ')'
2092 | PARSEOP_TOBUFFER '('
2093     error ')'
2094 ;

2096 ToDecimalStringTerm
2097 : PARSEOP_TODECIMALSTRING '('
2098     TermArg
2099     Target
2100     ')'
2101 | PARSEOP_TODECIMALSTRING '('
2102     error ')'
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_DMATTYPE_COMPATIBILITY    { $$ = TrCreateLeafNode (PARSEOP_DMAT
2239 | PARSEOP_DMATTYPE_B                { $$ = TrCreateLeafNode (PARSEOP_DMAT
2240 | PARSEOP_DMATTYPE_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         { $$ = TrCreateLeafNode (PARSEOP_REGI
2365 | PARSEOP_REGIONSPACE_FFIEDHW     { $$ = 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        { $$ = TrCreateLeafNode (PARSEOP_SHAR
2382 | PARSEOP_SHARETYPE_EXCLUSIVWAKE      { $$ = 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 | : Type60Opcode

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 */
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 ')'
2640 | PARSEOP_EISAID '('
2641   error ')'
2642 ;
2644 UnicodeTerm
2645 : PARSEOP_UNICODE '('
2646   StringData
2647   ')'
2648 | PARSEOP_UNICODE '('
2649   error ')'
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 '}'
2664   TrCreateLeafNode (PARSEOP_DEFAULT_ARG)
2665   TrCreateLeafNode (PARSEOP_DEFAULT_ARG)
2666   $5,
2667   TrCreateLeafNode (PARSEOP_ENDTAG));
2668 ;
2670 ResourceMacroList
2671 :
2672 | ResourceMacroList
2673   ResourceMacroTerm
2674 ;
2676 ResourceMacroTerm
2677 : DMArm
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 DMArm
2713 : PARSEOP_DMA '('
2714   DMATypeKeyword
2715   OptionalBusMasterKeyword
2716   ',' XferTypeKeyword
2717   OptionalNameString_Last
2718   ')'
2719   ByteList '}'
2720 | PARSEOP_DMA '('
2721   error ')'
2722 ;
2724 DWordIOTerm
2725 : 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   ')'
2742 | PARSEOP_DWORDIO '('
2743   error ')'
2744 ;
2746 DWordMemoryTerm
2747 : PARSEOP_DWORDMEMORY '('
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   ')'
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   ')' { $$ = TrLinkChildren ($<n>3,14,$4,$6,$7,$8,$
2786 | PARSEOP_DWORDSPACE '(' { $$ = AslDoError(); yyclearin; }
2787   error ')'
2788 ;

2791 EndDependentFnTerm
2792 : PARSEOP_ENDDEPENDENTFN '(' { $$ = TrCreateLeafNode (PARSEOP_ENDDEPENDENT
2793   ')'
2794 | PARSEOP_ENDDEPENDENTFN '(' { $$ = AslDoError(); yyclearin; }
2795   error ')'
2796 ;

2798 ExtendedIOTerm
2799 : PARSEOP_EXTENDEDIO '(' { $<n>$ = TrCreateLeafNode (PARSEOP_EXTENDEDIO
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   ')' { $$ = TrLinkChildren ($<n>3,14,$4,$5,$6,$7,$
2815 | PARSEOP_EXTENDEDIO '(' { $$ = AslDoError(); yyclearin; }
2816   error ')'
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   ')' { $$ = TrLinkChildren ($<n>3,15,$4,$5,$6,$7,$
2837 | PARSEOP_EXTENDEDMEMORY '(' { $$ = AslDoError(); yyclearin; }
2838   error ')'
2839 ;

2841 ExtendedSpaceTerm
2842 : PARSEOP_EXTENDEDSPACE '(' { $<n>$ = TrCreateLeafNode (PARSEOP_EXTENDED
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   ')' { $$ = TrLinkChildren ($<n>3,13,$4,$6,$7,$8,$
2857 | PARSEOP_EXTENDEDSPACE '(' { $$ = AslDoError(); yyclearin; }
2858   error ')'
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   ')' { $$ = TrLinkChildren ($<n>3,4,$4,$6,$7,$8); }
2868 | PARSEOP_FIXEDDMA '(' { $$ = AslDoError(); yyclearin; }
2869   error ')'
2870 ;

2872 FixedIOTerm
2873 : PARSEOP_FIXEDIO '(' { $<n>$ = TrCreateLeafNode (PARSEOP_FIXEDIO);
2874   WordConstExpr
2875   ',' ByteConstExpr
2876   OptionalWordConstExpr_Last
2877   ')' { $$ = TrLinkChildren ($<n>3,3,$4,$6,$7); }
2878 | PARSEOP_FIXEDIO '(' { $$ = AslDoError(); yyclearin; }
2879   error ')'
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 '}' { $$ = TrLinkChildren ($<n>3,11,$4,$6,$7,$9,$
2896 | PARSEOP_GPIO_INT '(' { $$ = AslDoError(); yyclearin; }
2897   error ')'

```

```

2898 ;

2900 GpioIoTerm
2901 : PARSEOP_GPIO_IO '('
2902     OptionalShareType_First
2903     ',' PinConfigByte
2904     OptionalWordConstExpr
2905     OptionalWordConstExpr
2906     OptionalIoRestriction
2907     ',' StringData
2908     OptionalByteConstExpr
2909     OptionalResourceType
2910     OptionalNameString
2911     OptionalBuffer_Last
2912     ')' '{'
2913     DWordList '}'
2914 | PARSEOP_GPIO_IO '('
2915     error ')'
2916 ;

2918 I2cSerialBusTerm
2919 : PARSEOP_I2C_SERIALBUS '('
2920     WordConstExpr
2921     OptionalSlaveMode
2922     ',' DWordConstExpr
2923     OptionalAddressingMode
2924     ',' StringData
2925     OptionalByteConstExpr
2926     OptionalResourceType
2927     OptionalNameString
2928     OptionalBuffer_Last
2929     ')'
2930 | PARSEOP_I2C_SERIALBUS '('
2931     error ')'
2932 ;

2934 InterruptTerm
2935 : PARSEOP_INTERRUPT '('
2936     OptionalResourceType_First
2937     ',' InterruptTypeKeyword
2938     ',' InterruptLevel
2939     OptionalShareType
2940     OptionalByteConstExpr
2941     OptionalStringData
2942     OptionalNameString_Last
2943     ')' '{'
2944     DWordList '}'
2945 | PARSEOP_INTERRUPT '('
2946     error ')'
2947 ;

2949 IOTerm
2950 : PARSEOP_IO '('
2951     IODecodeKeyword
2952     ',' WordConstExpr
2953     ',' WordConstExpr
2954     ',' ByteConstExpr
2955     ',' ByteConstExpr
2956     OptionalNameString_Last
2957     ')'
2958 | PARSEOP_IO '('
2959     error ')'
2960 ;

2962 IRQNoFlagsTerm
2963 : PARSEOP_IRQNOFLAGS '('

```

```

    {<n>$ = TrCreateLeafNode (PARSEOP_GPIO_IO);
    /* 04: SharedType */
    /* 06: PinConfig */
    /* 07: DebounceTimeout */
    /* 08: DriveStrength */
    /* 09: IoRestriction */
    /* 11: ResourceSource */
    /* 12: ResourceSourceIndex */
    /* 13: ResourceType */
    /* 14: DescriptorName */
    /* 15: VendorData */
    }
    {<n>$ = TrLinkChildren (<n>3,11,$4,$6,$7,$8,$
    {<n>$ = AslDoError(); yyclearin;}

    {<n>$ = TrCreateLeafNode (PARSEOP_I2C_SERIA
    /* 04: SlaveAddress */
    /* 05: SlaveMode */
    /* 07: ConnectionSpeed */
    /* 08: AddressingMode */
    /* 10: ResourceSource */
    /* 11: ResourceSourceIndex */
    /* 12: ResourceType */
    /* 13: DescriptorName */
    /* 14: VendorData */
    }
    {<n>$ = TrLinkChildren (<n>3,9,$4,$5,$7,$8,$1
    {<n>$ = AslDoError(); yyclearin;}

    {<n>$ = TrCreateLeafNode (PARSEOP_INTERRUPT
    {<n>$ = TrLinkChildren (<n>3,8,$4,$6,$8,$9,$1
    {<n>$ = AslDoError(); yyclearin;}

    {<n>$ = TrCreateLeafNode (PARSEOP_IO);}
    {<n>$ = TrLinkChildren (<n>3,6,$4,$6,$8,$10,$
    {<n>$ = AslDoError(); yyclearin;}

    {<n>$ = TrCreateLeafNode (PARSEOP_IRQNOFLAG

```

```

2964     OptionalNameString_First
2965     ')' '{'
2966         ByteList '}'
2967 | PARSEOP_IRQNOFLAGS '('
2968     error ')'
2969 ;

2971 IRQTerm
2972 : PARSEOP_IRQ '('
2973     InterruptTypeKeyword
2974     ',' InterruptLevel
2975     OptionalShareType
2976     OptionalNameString_Last
2977     ')' '{'
2978         ByteList '}'
2979 | PARSEOP_IRQ '('
2980     error ')'
2981 ;

2983 Memory24Term
2984 : PARSEOP_MEMORY24 '('
2985     OptionalReadWriteKeyword
2986     ',' WordConstExpr
2987     ',' WordConstExpr
2988     ',' WordConstExpr
2989     ',' WordConstExpr
2990     OptionalNameString_Last
2991     ')'
2992 | PARSEOP_MEMORY24 '('
2993     error ')'
2994 ;

2996 Memory32FixedTerm
2997 : PARSEOP_MEMORY32FIXED '('
2998     OptionalReadWriteKeyword
2999     ',' DWordConstExpr
3000     ',' DWordConstExpr
3001     OptionalNameString_Last
3002     ')'
3003 | PARSEOP_MEMORY32FIXED '('
3004     error ')'
3005 ;

3007 Memory32Term
3008 : PARSEOP_MEMORY32 '('
3009     OptionalReadWriteKeyword
3010     ',' DWordConstExpr
3011     ',' DWordConstExpr
3012     ',' DWordConstExpr
3013     ',' DWordConstExpr
3014     OptionalNameString_Last
3015     ')'
3016 | PARSEOP_MEMORY32 '('
3017     error ')'
3018 ;

3020 QWordIOTerm
3021 : PARSEOP_QWORDIO '('
3022     OptionalResourceType_First
3023     OptionalMinType
3024     OptionalMaxType
3025     OptionalDecodeType
3026     OptionalRangeType
3027     ',' QWordConstExpr
3028     ',' QWordConstExpr
3029     ',' QWordConstExpr

```

```

    {<n>$ = TrLinkChildren (<n>3,2,$4,$7);}
    {<n>$ = AslDoError(); yyclearin;}

    {<n>$ = TrCreateLeafNode (PARSEOP_IRQ);}
    {<n>$ = TrLinkChildren (<n>3,5,$4,$6,$7,$8,$1
    {<n>$ = AslDoError(); yyclearin;}

    {<n>$ = TrCreateLeafNode (PARSEOP_MEMORY24)
    {<n>$ = TrLinkChildren (<n>3,6,$4,$6,$8,$10,$
    {<n>$ = AslDoError(); yyclearin;}

    {<n>$ = TrCreateLeafNode (PARSEOP_MEMORY32F
    {<n>$ = TrLinkChildren (<n>3,4,$4,$6,$8,$9);}
    {<n>$ = AslDoError(); yyclearin;}

    {<n>$ = TrCreateLeafNode (PARSEOP_MEMORY32)
    {<n>$ = TrLinkChildren (<n>3,6,$4,$6,$8,$10,$
    {<n>$ = AslDoError(); yyclearin;}

    {<n>$ = TrCreateLeafNode (PARSEOP_QWORDIO);

```



```

3030      ' ' QWordConstExpr
3031      ' ' QWordConstExpr
3032      OptionalByteConstExpr
3033      OptionalStringData
3034      OptionalNameString
3035      OptionalType
3036      OptionalTranslationType_Last
3037      ' ' { $$ = TrLinkChildren ($<n>3,15,$4,$5,$6,$7,$
3038      | PARSEOP_QWORDIO '(' { $$ = AslDoError(); yyclearin;}
3039      error ') '
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   ' ' { $$ = TrLinkChildren ($<n>3,16,$4,$5,$6,$7,$
3061   | PARSEOP_QWORDMEMORY '(' { $$ = AslDoError(); yyclearin;}
3062   error ') '
3063   ;

3065 QWordSpaceTerm
3066 : PARSEOP_QWORDSPACE '(' { $<n>$ = TrCreateLeafNode (PARSEOP_QWORDSPAC
3067   ByteConstExpr
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   ' ' { $$ = TrLinkChildren ($<n>3,14,$4,$6,$7,$8,$
3082   | PARSEOP_QWORDSPACE '(' { $$ = AslDoError(); yyclearin;}
3083   error ') '
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   ' ' { $$ = TrLinkChildren ($<n>3,13,$4,$5,$6,$8,$
3115   | PARSEOP_SPI_SERIALBUS '(' { $$ = AslDoError(); yyclearin;}
3116   error ') '
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   ' ' { $$ = TrLinkChildren ($<n>3,14,$4,$5,$6,$8,$
3154   | PARSEOP_UART_SERIALBUS '(' { $$ = AslDoError(); yyclearin;}
3155   error ') '
3156   ;

3158 VendorLongTerm
3159 : PARSEOP_VENDORLONG '(' { $<n>$ = TrCreateLeafNode (PARSEOP_VENDORLON
3160   OptionalNameString_First
3161   ' ' '{

```

```

3162         ByteList '}'          { $$ = TrLinkChildren ($<n>3,2,$4,$7); }
3163     | PARSEOP_VENDORLONG '('   { $$ = AslDoError(); yyclearin; }
3164         error ')'
3165     ;

3167 VendorShortTerm
3168 : PARSEOP_VENDORSHORT '('     { $<n>$ = TrCreateLeafNode (PARSEOP_VENDORSHO
3169     OptionalNameString_First
3170     '}' '}'
3171     ByteList '}'              { $$ = TrLinkChildren ($<n>3,2,$4,$7); }
3172     | PARSEOP_VENDORSHORT '(' { $$ = AslDoError(); yyclearin; }
3173     error ')'
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     '}'
3191     | PARSEOP_WORDBUSNUMBER '(' { $$ = TrLinkChildren ($<n>3,12,$4,$5,$6,$7,$
3192     error ')'
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     '}'
3213     | PARSEOP_WORDIO '('       { $$ = TrLinkChildren ($<n>3,15,$4,$5,$6,$7,$
3214     error ')'
3215     ;

3217 WordSpaceTerm
3218 : PARSEOP_WORDSAPCE '('       { $<n>$ = TrCreateLeafNode (PARSEOP_WORDSAPCE
3219     ByteConstExpr
3220     {UtCheckIntegerRange ($4, 0xC0, 0xFF);}
3221     OptionalResourceType
3222     OptionalDecodeType
3223     OptionalMinType
3224     OptionalMaxType
3225     '',' ByteConstExpr
3226     '',' WordConstExpr
3227     '',' WordConstExpr

```

```

3228     '',' WordConstExpr
3229     '',' WordConstExpr
3230     OptionalByteConstExpr
3231     OptionalStringData
3232     OptionalNameString_Last
3233     '}'
3234     | PARSEOP_WORDSAPCE '('   { $$ = TrLinkChildren ($<n>3,14,$4,$6,$7,$8,$
3235     error ')'
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
3261     ;

3263 NameStringItem
3264 : '',' NameString
3265     | '',' error
3266     ;

3268 TermArgItem
3269 : '',' TermArg
3270     | '',' error
3271     ;

3273 OptionalBusMasterKeyword
3274 : '','
3275     | '',' PARSEOP_BUSMASTERTYPE_MASTER { $$ = TrCreateLeafNode (PARSEOP_
3276     '',' PARSEOP_BUSMASTERTYPE_NOTMASTER { $$ = TrCreateLeafNode (PARSEOP_
3277     ;

3279 OptionalAccessAttribTerm
3280 :
3281     | '','
3282     | '',' ByteConstExpr
3283     | '',' AccessAttribKeyword
3284     ;

3286 OptionalAccessSize
3287 :
3288     | '','
3289     | '',' ByteConstExpr
3290     ;

3292 OptionalAddressingMode
3293 : '','

```

```

3294 | ',' AddressingModeKeyword    {$$ = $2;}
3295 ;

3297 OptionalAddressRange
3298 :                               {$$ = NULL;}
3299 | ',' AddressKeyword           {$$ = $2;}
3300 |                               {$$ = $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 :
3429 | 'ResourceTypeKeyword'
3430 | 'ResourceTypeKeyword'
3431 ;
3432
3433 OptionalReturnArg
3434 :
3435 | TermArg
3436 ;
3437
3438 OptionalSerializeRuleKeyword
3439 :
3440 | 'SerializeRuleKeyword'
3441 | 'SerializeRuleKeyword'
3442 ;
3443
3444 OptionalSlaveMode
3445 :
3446 | 'SlaveModeKeyword'
3447 ;
3448
3449 OptionalShareType
3450 :
3451 | 'ShareTypeKeyword'
3452 | 'ShareTypeKeyword'
3453 ;
3454
3455 OptionalShareType_First
3456 :
3457 | ShareTypeKeyword
3458 ;
3459
3460 OptionalStopBits
3461 :
3462 | 'StopBitsKeyword'
3463 ;
3464
3465 OptionalStringData
3466 :
3467 | 'StringData'
3468 | 'StringData'
3469 ;
3470
3471 OptionalTermArg
3472 :
3473 | TermArg
3474 ;
3475
3476 OptionalType
3477 :
3478 | 'TypeKeyword'
3479 | 'TypeKeyword'
3480 ;
3481
3482 OptionalType_Last
3483 :
3484 | 'TypeKeyword'
3485 | 'TypeKeyword'
3486 ;
3487
3488 OptionalTranslationType_Last
3489 :
3490 | 'TranslationKeyword'
3491 | 'TranslationKeyword'

```

```

3492 ;
3493
3494 OptionalWireMode
3495 :
3496 | 'WireModeKeyword'
3497 ;
3498
3499 OptionalWordConst
3500 :
3501 | WordConst
3502 ;
3503
3504 OptionalWordConstExpr
3505 :
3506 | 'WordConstExpr'
3507 ;
3508
3509 OptionalXferSize
3510 :
3511 | 'XferSizeKeyword'
3512 | 'XferSizeKeyword'
3513 ;
3514
3515 %%
3516 /*****
3517 *
3518 * Local support functions
3519 *
3520 *****/
3521
3522 int
3523 AslCompilerwrap(void)
3524 {
3525     return (1);
3526 }
3527
3528 /*! [End] no source code translation !*/
3529
3530 void *
3531 AslLocalAllocate (unsigned int Size)
3532 {
3533     void *Mem;
3534
3535     DbgPrint (ASL_PARSE_OUTPUT, "\nAslLocalAllocate: Expanding Stack to %u\n\n",
3536             Mem = ACPI_ALLOCATE_ZEROED (Size);
3537     if (!Mem)
3538     {
3539         AslCommonError (ASL_ERROR, ASL_MSG_MEMORY_ALLOCATION,
3540             Gbl_CurrentLineNumber, Gbl_LogicalLineNumber,
3541             Gbl_InputByteCount, Gbl_CurrentColumn,
3542             Gbl_Files[ASL_FILE_INPUT].Filename, NULL);
3543         exit (1);
3544     }
3545     return (Mem);
3546 }
3547
3548 ACPI_PARSE_OBJECT *
3549 AslDoError (void)
3550 {
3551     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 }
```

new/usr/src/common/acpica/compiler/asldefine.h

1

```
*****
6470 Fri Jan 17 07:29:33 2014
new/usr/src/common/acpica/compiler/asldefine.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: asldefine.h - Common defines for the iASL compiler
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #ifndef __ASLDEFINE_H
45 #define __ASLDEFINE_H
46
47
48
49 /*
50 * Compiler versions and names
51 */
52 #define ASL_REVISION ACPI_CA_VERSION
53 #define ASL_COMPILER_NAME "ASL Optimizing Compiler"
54 #define AML_DISASSEMBLER_NAME "AML Disassembler"
55 #define ASL_INVOCATION_NAME "iasl"
56 #define ASL_CREATOR_ID "INTL"
57
58 #define ASL_COMPLIANCE "Supports ACPI Specification Revision 5.0A"
```

new/usr/src/common/acpica/compiler/asldefine.h

2

```
61 /* Configuration constants */
62
63 #define ASL_MAX_ERROR_COUNT 200
64 #define ASL_NODE_CACHE_SIZE 1024
65 #define ASL_STRING_CACHE_SIZE 32768
66
67 #define ASL_FIRST_PARSE_OPCODE PARSEOP_ACCESSAS
68 #define ASL_PARSE_OPCODE_BASE PARSEOP_ACCESSAS /* First Lex type */
69
70
71 /*
72 * Per-parser-generator configuration. These values are used to cheat and
73 * directly access the bison/yacc token name table (yyname or yytname).
74 * Note: These values are the index in yyname for the first lex token
75 * (PARSEOP_ACCESSAS).
76 */
77 #if defined (YBISON)
78 #define ASL_YTNAME_START 3 /* Bison */
79 #elif defined (YBYACC)
80 #define ASL_YTNAME_START 257 /* Berkeley yacc */
81 #endif
82
83
84 /*
85 * Macros
86 */
87 #define ASL_RESDISC_OFFSET(m) ACPI_OFFSET (AML_RESOURCE, m)
88 #define ASL_PTR_DIFF(a,b) ((UINT8 *) (b) - (UINT8 *) (a))
89 #define ASL_PTR_ADD(a,b) ((UINT8 *) (a) + ((UINT8 *) (a) + (b)))
90 #define ASL_GET_CHILD_NODE(a) (a)->Asl.Child
91 #define ASL_GET_PEER_NODE(a) (a)->Asl.Next
92 #define OP_TABLE_ENTRY(a,b,c,d) {b,d,a,c}
93
94
95 /* Internal AML opcodes */
96
97 #define AML_RAW_DATA_BYTE (UINT16) 0xAA01 /* write one raw byte */
98 #define AML_RAW_DATA_WORD (UINT16) 0xAA02 /* write 2 raw bytes */
99 #define AML_RAW_DATA_DWORD (UINT16) 0xAA04 /* write 4 raw bytes */
100 #define AML_RAW_DATA_QWORD (UINT16) 0xAA08 /* write 8 raw bytes */
101 #define AML_RAW_DATA_BUFFER (UINT16) 0xAA0B /* raw buffer with length */
102 #define AML_RAW_DATA_CHAIN (UINT16) 0xAA0C /* chain of raw buffers */
103 #define AML_PACKAGE_LENGTH (UINT16) 0xAA10
104 #define AML_UNASSIGNED_OPCODE (UINT16) 0xEEEE
105 #define AML_DEFAULT_ARG_OP (UINT16) 0xDDDD
106
107
108 /* filename suffixes for output files */
109
110 #define FILE_SUFFIX_PREPROCESSOR "i"
111 #define FILE_SUFFIX_AML_CODE "aml"
112 #define FILE_SUFFIX_LISTING "lst"
113 #define FILE_SUFFIX_HEX_DUMP "hex"
114 #define FILE_SUFFIX_DEBUG "txt"
115 #define FILE_SUFFIX_SOURCE "src"
116 #define FILE_SUFFIX_NAMESPACES "nsp"
117 #define FILE_SUFFIX_ASM_SOURCE "asm"
118 #define FILE_SUFFIX_C_SOURCE "c"
119 #define FILE_SUFFIX_DISASSEMBLY "dsl"
120 #define FILE_SUFFIX_ASM_INCLUDE "inc"
121 #define FILE_SUFFIX_C_INCLUDE "h"
122 #define FILE_SUFFIX_ASL_CODE "asl"
123 #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_ASF    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 */
```

new/usr/src/common/acpica/compiler/aslerror.c

1

```
*****
26638 Fri Jan 17 07:29:33 2014
new/usr/src/common/acpica/compiler/aslerror.c
update to acpica-unix2-20140114
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 - 2014, 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 MERCHANTABILITY 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 /*****
```

new/usr/src/common/acpica/compiler/aslerror.c

2

```
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  * FUNCTION:      AePrintException
150  *
151  * PARAMETERS:    FileId      - ID of output file
152  *                Enode       - Error node to print
153  *                Header      - Additional text before each message
154  *
155  * RETURN:        None
156  *
157  * DESCRIPTION:   Print the contents of an error node.
158  *
159  * NOTE:          We don't use the FlxxxFile I/O functions here because on error
160  *                they abort the compiler and call this function! Since we
161  *                are reporting errors here, we ignore most output errors and
162  *                just try to get out as much as we can.
163  *
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 file */

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 }
477
478 /*****
479 *
480 * FUNCTION:      AePrintErrorLog
481 *
482 * PARAMETERS:   FileId          - Where to output the error log
483 *
484 * RETURN:       None
485 *
486 * DESCRIPTION:  Print the entire contents of the error log
487 *
488 * *****/
489
490 void
491 AePrintErrorLog (
492     UINT32          FileId)
493 {
494     ASL_ERROR_MSG    *Enode = Gbl_ErrorLog;
495
496     /* Walk the error node list */
497
498     while (Enode)
499     {
500         AePrintException (FileId, Enode, NULL);
501         Enode = Enode->Next;
502     }
503 }
504
505 /*****
506 *
507 * FUNCTION:      AslCommonError2
508 *
509 * PARAMETERS:   Level          - Seriousness (Warning/error, etc.)
510 *               MessageId      - Index into global message buffer
511 *               LineNumber     - Actual file line number
512 *               Column         - Column in current line
513 *               SourceLine     - Actual source code line
514 *               Filename       - source filename
515 *               ExtraMessage   - additional error message
516 *
517 * *****/

```

```

520 * RETURN:      None
521 *
522 * DESCRIPTION: Create a new error node and add it to the error log
523 *
524 *****/

526 void
527 AslCommonError2 (
528     UINT8          Level,
529     UINT8          MessageId,
530     UINT32         LineNumber,
531     UINT32         Column,
532     char           *SourceLine,
533     char           *Filename,
534     char           *ExtraMessage)
535 {
536     char           *MessageBuffer = NULL;
537     char           *LineBuffer;
538     ASL_ERROR_MSG *Enode;

541     Enode = UtLocalCalloc (sizeof (ASL_ERROR_MSG));

543     if (ExtraMessage)
544     {
545         /* Allocate a buffer for the message and a new error node */
546
547         MessageBuffer = UtLocalCalloc (strlen (ExtraMessage) + 1);
548
549         /* Keep a copy of the extra message */
550
551         ACPI_STRCPY (MessageBuffer, ExtraMessage);
552     }

554     LineBuffer = UtLocalCalloc (strlen (SourceLine) + 1);
555     ACPI_STRCPY (LineBuffer, SourceLine);

557     /* Initialize the error node */

559     if (Filename)
560     {
561         Enode->Filename      = Filename;
562         Enode->FilenameLength = strlen (Filename);
563         if (Enode->FilenameLength < 6)
564         {
565             Enode->FilenameLength = 6;
566         }
567     }

569     Enode->MessageId      = MessageId;
570     Enode->Level          = Level;
571     Enode->LineNumber     = LineNumber;
572     Enode->LogicalLineNumber = LineNumber;
573     Enode->LogicalByteOffset = 0;
574     Enode->Column         = Column;
575     Enode->Message        = MessageBuffer;
576     Enode->SourceLine     = LineBuffer;

578     /* Add the new node to the error node list */

580     AeAddToErrorLog (Enode);

582     if (Gbl_DebugFlag)
583     {
584         /* stderr is a file, send error to it immediately */

```

```

586     AePrintException (ASL_FILE_STDERR, Enode, NULL);
587 }

589     Gbl_ExceptionCount[Level]++;
590 }

593 *****/
594 *
595 * FUNCTION:      AslCommonError
596 *
597 * PARAMETERS:   Level          - Seriousness (Warning/error, etc.)
598 *               MessageId      - Index into global message buffer
599 *               CurrentLineNumber - Actual file line number
600 *               LogicalLineNumber - Cumulative line number
601 *               LogicalByteOffset - Byte offset in source file
602 *               Column         - Column in current line
603 *               Filename        - source filename
604 *               ExtraMessage    - additional error message
605 *
606 * RETURN:       None
607 *
608 * DESCRIPTION: Create a new error node and add it to the error log
609 *
610 *****/

612 void
613 AslCommonError (
614     UINT8          Level,
615     UINT8          MessageId,
616     UINT32         CurrentLineNumber,
617     UINT32         LogicalLineNumber,
618     UINT32         LogicalByteOffset,
619     UINT32         Column,
620     char           *Filename,
621     char           *ExtraMessage)
622 {
623     char           *MessageBuffer = NULL;
624     ASL_ERROR_MSG *Enode;

627     Enode = UtLocalCalloc (sizeof (ASL_ERROR_MSG));

629     if (ExtraMessage)
630     {
631         /* Allocate a buffer for the message and a new error node */
632
633         MessageBuffer = UtLocalCalloc (strlen (ExtraMessage) + 1);
634
635         /* Keep a copy of the extra message */
636
637         ACPI_STRCPY (MessageBuffer, ExtraMessage);
638     }

640     /* Initialize the error node */

642     if (Filename)
643     {
644         Enode->Filename      = Filename;
645         Enode->FilenameLength = strlen (Filename);
646         if (Enode->FilenameLength < 6)
647         {
648             Enode->FilenameLength = 6;
649         }
650     }

```

```

652     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\n" 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 {
817     /* Check if user wants to ignore this exception */
818
819     if (AslIsExceptionDisabled (Level, MessageId))
820     {
821         return;
822     }
823
824     if (Op)
825     {
826         AslCommonError (Level, MessageId, Op->Asl.LineNumber,
827             Op->Asl.LogicalLineNumber,
828             Op->Asl.LogicalByteOffset,
829             Op->Asl.Column,
830             Op->Asl.Filename, ExtraMessage);
831     }
832     else
833     {
834         AslCommonError (Level, MessageId, 0,
835             0, 0, 0, NULL, ExtraMessage);
836     }
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 *****/

```

```

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 {
864     sprintf (MsgBuffer, "%s %s", AcpiFormatException (Status), ExtraMessage);
865
866     if (Op)
867     {
868         AslCommonError (ASL_ERROR, ASL_MSG_CORE_EXCEPTION, Op->Asl.LineNumber,
869             Op->Asl.LogicalLineNumber,
870             Op->Asl.LogicalByteOffset,
871             Op->Asl.Column,
872             Op->Asl.Filename, MsgBuffer);
873     }
874     else
875     {
876         AslCommonError (ASL_ERROR, ASL_MSG_CORE_EXCEPTION, 0,
877             0, 0, 0, NULL, MsgBuffer);
878     }
879
880     if (Abort)
881     {
882         AslAbort ();
883     }
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 {
906     AslCommonError (ASL_ERROR, ASL_MSG_SYNTAX, Gbl_CurrentLineNumber,
907         Gbl_LogicalLineNumber, Gbl_CurrentLineOffset,
908         Gbl_CurrentColumn, Gbl_Files[ASL_FILE_INPUT].Filename,
909         ACPI_CAST_PTR (char, CompilerMessage));
910
911     return (0);
912 }

```

new/usr/src/common/acpica/compiler/aslfileio.c

1

```
*****
10151 Fri Jan 17 07:29:34 2014
new/usr/src/common/acpica/compiler/aslfileio.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: aslfileio - File I/O support
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"

46 #define _COMPONENT          ACPI_COMPILER
47 #define ACPI_MODULE_NAME    ("aslfileio")

50 /*****
51 *
52 * FUNCTION:      AslAbort
53 *
54 * PARAMETERS:    None
55 *
56 * RETURN:        None
57 *
58 * DESCRIPTION:   Dump the error log and abort the compiler. Used for serious
59 *                I/O errors.
60 */
```

new/usr/src/common/acpica/compiler/aslfileio.c

2

```
61 *****/

63 void
64 AslAbort (
65     void)
66 {
67
68     AePrintErrorLog (ASL_FILE_STDERR);
69     if (Gbl_DebugFlag)
70     {
71         /* Print error summary to stdout also */
72
73         AePrintErrorLog (ASL_FILE_STDOUT);
74     }
75
76     exit (1);
77 }

80 /*****
81 *
82 * FUNCTION:      FlFileError
83 *
84 * PARAMETERS:    FileId          - Index into file info array
85 *                ErrorId         - Index into error message array
86 *
87 * RETURN:        None
88 *
89 * DESCRIPTION:   Decode errno to an error message and add the entire error
90 *                to the error log.
91 *
92 *****/

94 void
95 FlFileError (
96     UINT32          FileId,
97     UINT8           ErrorId)
98 {
99
100     sprintf (MsgBuffer, "\\%s\" (%s)", Gbl_Files[FileId].Filename,
101             strerror (errno));
102     AslCommonError (ASL_ERROR, ErrorId, 0, 0, 0, 0, NULL, MsgBuffer);
103 }

106 /*****
107 *
108 * FUNCTION:      FlOpenFile
109 *
110 * PARAMETERS:    FileId          - Index into file info array
111 *                Filename        - file pathname to open
112 *                Mode            - Open mode for fopen
113 *
114 * RETURN:        None
115 *
116 * DESCRIPTION:   Open a file.
117 *                NOTE: Aborts compiler on any error.
118 *
119 *****/

121 void
122 FlOpenFile (
123     UINT32          FileId,
124     char            *Filename,
125     char            *Mode)
126 {
```

```

127     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  *
188  * NOTE: Aborts compiler on any error.
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  *
232  * NOTE: Aborts compiler on any error.
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
260 *               Format      - Printf format string
261 *               ...         - Printf arguments
262 *
263 * RETURN:      None
264 *
265 * DESCRIPTION: Formatted write to an open file.
266 *               NOTE: Aborts compiler on any error.
267 *
268 *****/

270 void
271 FlPrintFile (
272     UInt32      FileId,
273     char        *Format,
274     ...)
275 {
276     INT32      Actual;
277     va_list     Args;

280     va_start (Args, Format);

282     Actual = vfprintf (Gbl_Files[FileId].Handle, Format, Args);
283     va_end (Args);

285     if (Actual == -1)
286     {
287         FlFileError (FileId, ASL_MSG_WRITE);
288         AslAbort ();
289     }
290 }

293 /*****
294 *
295 * FUNCTION:      FlSeekFile
296 *
297 * PARAMETERS:  FileId      - Index into file info array
298 *               Offset      - Absolute byte offset in file
299 *
300 * RETURN:      None
301 *
302 * DESCRIPTION:  Seek to absolute offset.
303 *               NOTE: Aborts compiler on any error.
304 *
305 *****/

307 void
308 FlSeekFile (
309     UInt32      FileId,
310     long        Offset)
311 {
312     int         Error;

315     Error = fseek (Gbl_Files[FileId].Handle, Offset, SEEK_SET);
316     if (Error)
317     {
318         FlFileError (FileId, ASL_MSG_SEEK);
319         AslAbort ();
320     }
321 }

324 /*****

```

```

325 *
326 * FUNCTION:      FlCloseFile
327 *
328 * PARAMETERS:  FileId      - Index into file info array
329 *
330 * RETURN:      None
331 *
332 * DESCRIPTION:  Close an open file. Aborts compiler on error
333 *
334 *****/

336 void
337 FlCloseFile (
338     UInt32      FileId)
339 {
340     int         Error;

343     if (!Gbl_Files[FileId].Handle)
344     {
345         return;
346     }

348     Error = fclose (Gbl_Files[FileId].Handle);
349     if (Error)
350     {
351         FlFileError (FileId, ASL_MSG_CLOSE);
352         AslAbort ();
353     }

355     Gbl_Files[FileId].Handle = NULL;
356     return;
357 }

360 /*****
361 *
362 * FUNCTION:      FlDeleteFile
363 *
364 * PARAMETERS:  FileId      - Index into file info array
365 *
366 * RETURN:      None
367 *
368 * DESCRIPTION:  Delete a file.
369 *
370 *****/

372 void
373 FlDeleteFile (
374     UInt32      FileId)
375 {
376     ASL_FILE_INFO *Info = &Gbl_Files[FileId];

379     if (!Info->Filename)
380     {
381         return;
382     }

384     if (remove (Info->Filename))
385     {
386         printf ("%s (%s file) ",
387             Info->Filename, Info->Description);
388         perror ("Could not delete");
389     }

```

new/usr/src/common/acpica/compiler/aslfileio.c

7

```
391     Info->Filename = NULL;  
392     return;  
393 }
```

new/usr/src/common/acpica/compiler/aslfiles.c

1

```
*****
23426 Fri Jan 17 07:29:34 2014
new/usr/src/common/acpica/compiler/aslfiles.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: aslfiles - File support functions
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"

47 #define _COMPONENT          ACPI_COMPILER
48         ACPI_MODULE_NAME    ("aslfiles")

50 /* Local prototypes */

52 FILE *
53 FlOpenIncludeWithPrefix (
54     char        *PrefixDir,
55     char        *Filename);

58 #ifdef ACPI_OBSOLETE_FUNCTIONS
```

new/usr/src/common/acpica/compiler/aslfiles.c

2

```
59 ACPI_STATUS
60 FlParseInputPathname (
61     char        *InputFilename);
62 #endif

65 /*****
66 *
67 * FUNCTION:      FlSetLineNumber
68 *
69 * PARAMETERS:    Op          - Parse node for the LINE asl statement
70 *
71 * RETURN:        None.
72 *
73 * DESCRIPTION:   Set the current line number
74 *
75 *****/

77 void
78 FlSetLineNumber (
79     UINT32          LineNumber)
80 {
82     DbgPrint (ASL_PARSE_OUTPUT, "\n#line: New line number %u (old %u)\n",
83             LineNumber, Gbl_LogicalLineNumber);

85     Gbl_CurrentLineNumber = LineNumber;
86     Gbl_LogicalLineNumber = LineNumber;
87 }

90 /*****
91 *
92 * FUNCTION:      FlSetFilename
93 *
94 * PARAMETERS:    Op          - Parse node for the LINE asl statement
95 *
96 * RETURN:        None.
97 *
98 * DESCRIPTION:   Set the current filename
99 *
100 *****/

102 void
103 FlSetFilename (
104     char        *Filename)
105 {
107     DbgPrint (ASL_PARSE_OUTPUT, "\n#line: New filename %s (old %s)\n",
108             Filename, Gbl_Files[ASL_FILE_INPUT].Filename);

110     Gbl_Files[ASL_FILE_INPUT].Filename = Filename;
111 }

114 /*****
115 *
116 * FUNCTION:      FlAddIncludeDirectory
117 *
118 * PARAMETERS:    Dir          - Directory pathname string
119 *
120 * RETURN:        None
121 *
122 * DESCRIPTION:   Add a directory the list of include prefix directories.
123 *
124 *****/
```

```

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 && (!strcmp (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 = FOpenIncludeWithPrefix (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 = FOpenIncludeWithPrefix (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:      FOpenInputFile
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  FOpenInputFile (
441      char                *InputFilename)
442  {

444      /* Open the input ASL file, text mode */

446      FOpenFile (ASL_FILE_INPUT, InputFilename, "rt");
447      AslCompilerin = Gbl_Files[ASL_FILE_INPUT].Handle;

449      return (AE_OK);
450  }

453  /*****
454  *

```

```

455  * FUNCTION:      FOpenAmlOutputFile
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  FOpenAmlOutputFile (
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 = FGenerateFilename (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      FOpenFile (ASL_FILE_AML_OUTPUT, Filename, "wb");
492      return (AE_OK);
493  }

496  /*****
497  *
498  * FUNCTION:      FOpenMiscOutputFiles
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  FOpenMiscOutputFiles (
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  * FUNCTION:      FlParseInputPathname
767  *
768  * PARAMETERS:    InputFilename      - The user-specified ASL source file to be
769  *                                     compiled
770  *
771  * RETURN:        Status
772  *
773  * DESCRIPTION:    Split the input path into a directory and filename part
774  *                 1) Directory part used to open include files
775  *                 2) Filename part used to generate output filenames
776  *
777  *****/
778

780 ACPI_STATUS
781 FlParseInputPathname (
782     char                *InputFilename)
783 {
784     char                *Substring;

```



```
787     if (!InputFilename)
788     {
789         return (AE_OK);
790     }
791
792     /* Get the path to the input filename's directory */
793
794     Gbl_DirectoryPath = strdup (InputFilename);
795     if (!Gbl_DirectoryPath)
796     {
797         return (AE_NO_MEMORY);
798     }
799
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     }
809
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     }
826
827     UtConvertBackslashes (Gbl_OutputFilenamePrefix);
828     return (AE_OK);
829 }
830 #endif
```

new/usr/src/common/acpica/compiler/aslfold.c

1

```
*****
15815 Fri Jan 17 07:29:34 2014
new/usr/src/common/acpica/compiler/aslfold.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslfold - Constant folding
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"
47 #include "amlcode.h"

49 #include "acdispat.h"
50 #include "acparser.h"

52 #define _COMPONENT          ACPI_COMPILER
53       ACPI_MODULE_NAME    ("aslfold")

55 /* Local prototypes */

57 static ACPI_STATUS
58 OpcAmlEvaluationWalk1 (
59     ACPI_PARSE_OBJECT *Op,
```

new/usr/src/common/acpica/compiler/aslfold.c

2

```
60     UINT32                Level,
61     void                  *Context);

63 static ACPI_STATUS
64 OpcAmlEvaluationWalk2 (
65     ACPI_PARSE_OBJECT *Op,
66     UINT32            Level,
67     void              *Context);

69 static ACPI_STATUS
70 OpcAmlCheckForConstant (
71     ACPI_PARSE_OBJECT *Op,
72     UINT32            Level,
73     void              *Context);

75 static void
76 OpcUpdateIntegerNode (
77     ACPI_PARSE_OBJECT *Op,
78     UINT64            Value);

81 /*****
82 *
83 * FUNCTION:      OpcAmlEvaluationWalk1
84 *
85 * PARAMETERS:    ASL_WALK_CALLBACK
86 *
87 * RETURN:        Status
88 *
89 * DESCRIPTION:    Descending callback for AML execution of constant subtrees
90 *
91 *****/

93 static ACPI_STATUS
94 OpcAmlEvaluationWalk1 (
95     ACPI_PARSE_OBJECT *Op,
96     UINT32            Level,
97     void              *Context)
98 {
99     ACPI_WALK_STATE *WalkState = Context;
100     ACPI_STATUS      Status;
101     ACPI_PARSE_OBJECT *OutOp;

104     WalkState->Op = Op;
105     WalkState->Opcode = Op->Common.AmlOpcode;
106     WalkState->OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);

108     /* Copy child pointer to Arg for compatibility with Interpreter */

110     if (Op->Asl.Child)
111     {
112         Op->Common.Value.Arg = Op->Asl.Child;
113     }

115     /* Call AML dispatcher */

117     Status = AcpiDsExecBeginOp (WalkState, &OutOp);
118     if (ACPI_FAILURE (Status))
119     {
120         AcpiOsPrintf ("Constant interpretation failed - %s\n",
121             AcpiFormatException (Status));
122     }

124     return (Status);
125 }
```

```

128 /*****
129 *
130 * FUNCTION:      OpcAmlEvaluationWalk2
131 *
132 * PARAMETERS:    ASL_WALK_CALLBACK
133 *
134 * RETURN:        Status
135 *
136 * DESCRIPTION:   Ascending callback for AML execution of constant subtrees
137 *
138 *****/
140 static ACPI_STATUS
141 OpcAmlEvaluationWalk2 (
142     ACPI_PARSE_OBJECT *Op,
143     UINT32 Level,
144     void *Context)
145 {
146     ACPI_WALK_STATE *WalkState = Context;
147     ACPI_STATUS Status;

150     WalkState->Op = Op;
151     WalkState->Opcode = Op->Common.AmlOpcode;
152     WalkState->OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);

154     /* Copy child pointer to Arg for compatibility with Interpreter */

156     if (Op->Asl.Child)
157     {
158         Op->Common.Value.Arg = Op->Asl.Child;
159     }

161     /* Call AML dispatcher */

163     Status = AcpiDsExecEndOp (WalkState);
164     if (ACPI_FAILURE (Status))
165     {
166         AcpiOsPrintf ("Constant interpretation failed - %s\n",
167             AcpiFormatException (Status));
168     }

170     return (Status);
171 }

174 /*****
175 *
176 * FUNCTION:      OpcAmlCheckForConstant
177 *
178 * PARAMETERS:    ASL_WALK_CALLBACK
179 *
180 * RETURN:        Status
181 *
182 * DESCRIPTION:   Check one Op for a type 3/4/5 AML opcode
183 *
184 *****/
186 static ACPI_STATUS
187 OpcAmlCheckForConstant (
188     ACPI_PARSE_OBJECT *Op,
189     UINT32 Level,
190     void *Context)
191 {

```

```

192     ACPI_WALK_STATE *WalkState = Context;

195     WalkState->Op = Op;
196     WalkState->Opcode = Op->Common.AmlOpcode;
197     WalkState->OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);

199     DbgPrint (ASL_PARSE_OUTPUT, "[%4d] Opcode: %12.12s ",
200         Op->Asl.LogicalLineNumber, Op->Asl.ParseOpName);

202     /*
203     * These opcodes do not appear in the OpcodeInfo table, but
204     * they represent constants, so abort the constant walk now.
205     */
206     if ((WalkState->Opcode == AML_RAW_DATA_BYTE) ||
207         (WalkState->Opcode == AML_RAW_DATA_WORD) ||
208         (WalkState->Opcode == AML_RAW_DATA_DWORD) ||
209         (WalkState->Opcode == AML_RAW_DATA_QWORD))
210     {
211         WalkState->WalkType = ACPI_WALK_CONST_OPTIONAL;
212         return (AE_TYPE);
213     }

215     if (!(WalkState->OpInfo->Flags & AML_CONSTANT))
216     {
217         /* The opcode is not a Type 3/4/5 opcode */

219         if (Op->Asl.CompileFlags & NODE_IS_TARGET)
220         {
221             DbgPrint (ASL_PARSE_OUTPUT,
222                 "**** Valid Target, cannot reduce ****\n");
223         }
224         else
225         {
226             DbgPrint (ASL_PARSE_OUTPUT,
227                 "**** Not a Type 3/4/5 opcode ****\n");
228         }

230         if (WalkState->WalkType == ACPI_WALK_CONST_OPTIONAL)
231         {
232             /*
233             * We are looking at a normal expression to see if it can be
234             * reduced. It can't. No error
235             */
236             return (AE_TYPE);
237         }

239         /*
240         * This is an expression that MUST reduce to a constant, and it
241         * can't be reduced. This is an error
242         */
243         if (Op->Asl.CompileFlags & NODE_IS_TARGET)
244         {
245             AslError (ASL_ERROR, ASL_MSG_INVALID_TARGET, Op,
246                 Op->Asl.ParseOpName);
247         }
248         else
249         {
250             AslError (ASL_ERROR, ASL_MSG_INVALID_CONSTANT_OP, Op,
251                 Op->Asl.ParseOpName);
252         }

254         return (AE_TYPE);
255     }

257     /* Debug output */

```

```

259     DbgPrint (ASL_PARSE_OUTPUT, "TYPE_345");

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     }

270     DbgPrint (ASL_PARSE_OUTPUT, "\n");
271     return (AE_OK);
272 }

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  *****/

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;

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     }

311     /* Set the walk type based on the reduction used for this op */

313     if (Op->Asl.CompileFlags & NODE_IS_TERM_ARG)
314     {
315         /* Op is a TermArg, constant folding is merely optional */

317         if (!Gbl_FoldConstants)
318         {
319             return (AE_CTRL_DEPTH);
320         }

322         WalkType = ACPI_WALK_CONST_OPTIONAL;
323     }

```

```

324     else
325     {
326         /* Op is a DataObject, the expression MUST reduced to a constant */

328         WalkType = ACPI_WALK_CONST_REQUIRED;
329     }

331     /* Create a new walk state */

333     WalkState = AcpiDsCreateWalkState (0, NULL, NULL, NULL);
334     if (!WalkState)
335     {
336         return (AE_NO_MEMORY);
337     }

339     WalkState->NextOp = NULL;
340     WalkState->Params = NULL;
341     WalkState->WalkType = WalkType;
342     WalkState->CallerReturnDesc = &ObjDesc;

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);

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 */

359         if (WalkState->WalkType == ACPI_WALK_CONST_OPTIONAL)
360         {
361             AcpiDsDeleteWalkState (WalkState);
362             return (AE_OK);
363         }

365         /* Don't descend any further, and use a default "constant" value */

367         Status = AE_CTRL_DEPTH;
368     }
369     else
370     {
371         /* Subtree can be reduced */

373         /* Allocate a new temporary root for this subtree */

375         RootOp = TrAllocateNode (PARSEOP_INTEGER);
376         if (!RootOp)
377         {
378             return (AE_NO_MEMORY);
379         }

381         RootOp->Common.AmlOpcode = AML_INT_EVAL_SUBTREE_OP;

383         OriginalParentOp = Op->Common.Parent;
384         Op->Common.Parent = RootOp;

386         /* Hand off the subtree to the AML interpreter */

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 {
527
528     Op->Common.Value.Integer = Value;
529
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:
537
538         TrUpdateNode (PARSEOP_BYTECONST, Op);
539         Op->Asl.AmlOpcode = AML_RAW_DATA_BYTE;
540         break;
541
542     case 2:
543
544         TrUpdateNode (PARSEOP_WORDCONST, Op);
545         Op->Asl.AmlOpcode = AML_RAW_DATA_WORD;
546         break;
547
548     case 4:
549
550         TrUpdateNode (PARSEOP_DWORDCONST, Op);
551         Op->Asl.AmlOpcode = AML_RAW_DATA_DWORD;
552         break;
553
554     case 8:
555
556         TrUpdateNode (PARSEOP_QWORDCONST, Op);
557         Op->Asl.AmlOpcode = AML_RAW_DATA_QWORD;
558         break;
559
560     case 0:
561     default:
562
563         OpcSetOptimalIntegerSize (Op);
564         TrUpdateNode (PARSEOP_INTEGER, Op);
565         break;
566     }
567
568     Op->Asl.AmlLength = 0;
569 }
```

new/usr/src/common/acpica/compiler/aslglobal.h

1

```
*****
12572 Fri Jan 17 07:29:34 2014
new/usr/src/common/acpica/compiler/aslglobal.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: aslglobal.h - Global variable definitions
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */

43
44
45 #ifndef __ASLGLOBAL_H
46 #define __ASLGLOBAL_H
47
48
49 /*
50 * Global variables. Defined in aslmain.c only, externed in all other files
51 */
52
53 #undef ASL_EXTERN
54
55 #ifdef _DECLARE_GLOBALS
56 #define ASL_EXTERN
57 #define ASL_INIT_GLOBAL(a,b) (a)=(b)
58 #else
```

new/usr/src/common/acpica/compiler/aslglobal.h

2

```
59 #define ASL_EXTERN          extern
60 #define ASL_INIT_GLOBAL(a,b) (a)
61 #endif
62
63
64 #ifdef _DECLARE_GLOBALS
65 UINT32          Gbl_ExceptionCount[ASL_NUM_REPORT_LEVELS] =
66 char            AslHexLookup[] =
67 {
68     '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'
69 };
70
71
72 /* Table below must match ASL_FILE_TYPES in asltypes.h */
73
74 ASL_FILE_INFO          Gbl_Files [ASL_NUM_FILES] =
75 {
76     {NULL, NULL, "stdout:", "Standard Output"},
77     {NULL, NULL, "stderr:", "Standard Error"},
78     {NULL, NULL, "Table Input:", "Source Input"},
79     {NULL, NULL, "Binary Output:", "AML Output"},
80     {NULL, NULL, "Source Output:", "Source Output"},
81     {NULL, NULL, "Preprocessor:", "Preprocessor Output"},
82     {NULL, NULL, "Listing File:", "Listing Output"},
83     {NULL, NULL, "Hex Dump:", "Hex Table Output"},
84     {NULL, NULL, "Namespace:", "Namespace Output"},
85     {NULL, NULL, "Debug File:", "Debug Output"},
86     {NULL, NULL, "ASM Source:", "Assembly Code Output"},
87     {NULL, NULL, "C Source:", "C Code Output"},
88     {NULL, NULL, "ASM Include:", "Assembly Header Output"},
89     {NULL, NULL, "C Include:", "C Header Output"},
90     {NULL, NULL, "Offset Table:", "C Offset Table Output"}
91 };
92
93 #else
94 extern UINT32          Gbl_ExceptionCount[ASL_NUM_REPORT_LEVELS];
95 extern char            AslHexLookup[];
96 extern ASL_FILE_INFO  Gbl_Files [ASL_NUM_FILES];
97 #endif
98
99
100 /*
101 * Parser and other externals
102 */
103 extern int             yydebug;
104 extern FILE            *AslCompilerin;
105 extern int             AslCompilerdebug;
106 extern int             DtParserdebug;
107 extern int             PrParserdebug;
108 extern const ASL_MAPPING_ENTRY AslKeywordMapping[];
109 extern char            *AslCompilertext;
110
111 #define ASL_DEFAULT_LINE_BUFFER_SIZE    (1024 * 32) /* 32K */
112 #define ASL_MSG_BUFFER_SIZE            4096
113 #define ASL_MAX_DISABLED_MESSAGES      32
114 #define HEX_TABLE_LINE_SIZE            8
115 #define HEX_LISTING_LINE_SIZE          8
116
117
118 /* Source code buffers and pointers for error reporting */
119
120 ASL_EXTERN char        ASL_INIT_GLOBAL (*Gbl_CurrentLineBuffer, NUL
121 ASL_EXTERN char        ASL_INIT_GLOBAL (*Gbl_LineBufPtr, NULL);
122 ASL_EXTERN UINT32      ASL_INIT_GLOBAL (Gbl_LineBufferSize, ASL_DEF
123 ASL_EXTERN UINT32      ASL_INIT_GLOBAL (Gbl_CurrentColumn, 0);
124 ASL_EXTERN UINT32      ASL_INIT_GLOBAL (Gbl_PreviousLineNumber, 0);
```

```

125 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, FALSE);
149 ASL_EXTERN BOOLEAN          ASL_INIT_GLOBAL (Gbl_AsmIncludeOutputFlag, FALSE);
150 ASL_EXTERN BOOLEAN          ASL_INIT_GLOBAL (Gbl_C_IncludeOutputFlag, FALSE);
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, FALSE);
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, FALSE);
164 ASL_EXTERN BOOLEAN          ASL_INIT_GLOBAL (Gbl_ReferenceOptimizationFlag, FALSE);
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, FALSE);
168 ASL_EXTERN UINT8            ASL_INIT_GLOBAL (Gbl_WarningLevel, ASL_WARNI);
169 ASL_EXTERN BOOLEAN          ASL_INIT_GLOBAL (Gbl_UseOriginalCompilerId, FALSE);
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, NULL);
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, NULL);

```

```

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, 0);
231 ASL_EXTERN UINT8             ASL_INIT_GLOBAL (Gbl_HexBytesWereWritten, FALSE);
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_MESSAGES];

```



new/usr/src/common/acpica/compiler/aslglobal.h

5

```
259 #endif /* __ASLGLOBAL_H */
```

new/usr/src/common/acpica/compiler/aslhex.c

1

```
*****
11549 Fri Jan 17 07:29:34 2014
new/usr/src/common/acpica/compiler/aslhex.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: aslhex - ASCII hex output file generation (C, ASM, and ASL)
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"

47 #define _COMPONENT          ACPI_COMPILER
48     ACPI_MODULE_NAME        ("ashex")

50 /*
51 * This module emits ASCII hex output files in either C, ASM, or ASL format
52 */

55 /* Local prototypes */

57 static void
58 HxDoHexOutputC (
59     void);
```

new/usr/src/common/acpica/compiler/aslhex.c

2

```
61 static void
62 HxDoHexOutputAsl (
63     void);

65 static void
66 HxDoHexOutputAsm (
67     void);

69 static UINT32
70 HxReadAmlOutputFile (
71     UINT8                *Buffer);

74 /*****
75 *
76 * FUNCTION:      HxDoHexOutput
77 *
78 * PARAMETERS:    None
79 *
80 * RETURN:        None
81 *
82 * DESCRIPTION:   Create the hex output file. Note: data is obtained by reading
83 *                the entire AML output file that was previously generated.
84 *
85 *****/

87 void
88 HxDoHexOutput (
89     void)
90 {
91
92     switch (Gbl_HexOutputFlag)
93     {
94     case HEX_OUTPUT_C:
95
96         HxDoHexOutputC ();
97         break;
98
99     case HEX_OUTPUT_ASM:
100
101         HxDoHexOutputAsm ();
102         break;
103
104     case HEX_OUTPUT_AS_L:
105
106         HxDoHexOutputAsl ();
107         break;
108
109     default:
110
111         /* No other output types supported */
112
113         break;
114     }
115 }

118 /*****
119 *
120 * FUNCTION:      HxReadAmlOutputFile
121 *
122 * PARAMETERS:    Buffer - Where to return data
123 *
124 * RETURN:        None
125 *
126 * DESCRIPTION:   Read a line of the AML output prior to formatting the data
```

```

127  *
128  *****/
130 static UINT32
131 HxReadAmlOutputFile (
132     UINT8          *Buffer)
133 {
134     UINT32          Actual;
135
136     Actual = fread (Buffer, 1, HEX_TABLE_LINE_SIZE,
137         Gbl_Files[ASL_FILE_AML_OUTPUT].Handle);
138
139     if (ferror (Gbl_Files[ASL_FILE_AML_OUTPUT].Handle))
140     {
141         FlFileError (ASL_FILE_AML_OUTPUT, ASL_MSG_READ);
142         AslAbort ();
143     }
144
145     return (Actual);
146 }
147
148
149
150 /*****
151  *
152  * FUNCTION:      HxDoHexOutputC
153  *
154  * PARAMETERS:    None
155  *
156  * RETURN:        None
157  *
158  * DESCRIPTION:   Create the hex output file. This is the same data as the AML
159  *                output file, but formatted into hex/ascii bytes suitable for
160  *                inclusion into a C source file.
161  *
162  *****/
163
164 static void
165 HxDoHexOutputC (
166     void)
167 {
168     UINT8          FileData[HEX_TABLE_LINE_SIZE];
169     UINT32          LineLength;
170     UINT32          Offset = 0;
171     UINT32          AmlFileSize;
172     UINT32          i;
173
174     /* Get AML size, seek back to start */
175
176     AmlFileSize = FlGetFileSize (ASL_FILE_AML_OUTPUT);
177     FlSeekFile (ASL_FILE_AML_OUTPUT, 0);
178
179     FlPrintFile (ASL_FILE_HEX_OUTPUT, " * C source code output\n");
180     FlPrintFile (ASL_FILE_HEX_OUTPUT, " * AML code block contains 0x%X bytes\n *",
181         AmlFileSize);
182     FlPrintFile (ASL_FILE_HEX_OUTPUT, "unsigned char AmlCode[] =\n{\n");
183
184     while (Offset < AmlFileSize)
185     {
186         /* Read enough bytes needed for one output line */
187
188         LineLength = HxReadAmlOutputFile (FileData);
189         if (!LineLength)
190         {
191             break;
192         }

```

```

193     }
194
195     FlPrintFile (ASL_FILE_HEX_OUTPUT, "    ");
196
197     for (i = 0; i < LineLength; i++)
198     {
199         /*
200          * Print each hex byte.
201          * Add a comma until the very last byte of the AML file
202          * (Some C compilers complain about a trailing comma)
203          */
204         FlPrintFile (ASL_FILE_HEX_OUTPUT, "0x%2.2X", FileData[i]);
205         if ((Offset + i + 1) < AmlFileSize)
206         {
207             FlPrintFile (ASL_FILE_HEX_OUTPUT, ",");
208         }
209         else
210         {
211             FlPrintFile (ASL_FILE_HEX_OUTPUT, " ");
212         }
213     }
214
215     /* Add fill spaces if needed for last line */
216
217     if (LineLength < HEX_TABLE_LINE_SIZE)
218     {
219         FlPrintFile (ASL_FILE_HEX_OUTPUT, "%*s",
220             5 * (HEX_TABLE_LINE_SIZE - LineLength), " ");
221     }
222
223     /* Emit the offset and ascii dump for the entire line */
224
225     FlPrintFile (ASL_FILE_HEX_OUTPUT, " /* %8.8X", Offset);
226     LsDumpAsciiInComment (ASL_FILE_HEX_OUTPUT, LineLength, FileData);
227     FlPrintFile (ASL_FILE_HEX_OUTPUT, "%*s*\n",
228         HEX_TABLE_LINE_SIZE - LineLength + 1, " ");
229
230     Offset += LineLength;
231 }
232
233 FlPrintFile (ASL_FILE_HEX_OUTPUT, "};\n");
234 }
235
236
237 /*****
238  *
239  * FUNCTION:      HxDoHexOutputAsl
240  *
241  * PARAMETERS:    None
242  *
243  * RETURN:        None
244  *
245  * DESCRIPTION:   Create the hex output file. This is the same data as the AML
246  *                output file, but formatted into hex/ascii bytes suitable for
247  *                inclusion into a C source file.
248  *
249  *****/
250
251 static void
252 HxDoHexOutputAsl (
253     void)
254 {
255     UINT8          FileData[HEX_TABLE_LINE_SIZE];
256     UINT32          LineLength;
257     UINT32          Offset = 0;
258     UINT32          AmlFileSize;

```

```

259     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, "0x%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 }
```

new/usr/src/common/acpica/compiler/aslength.c

1

```
*****
12065 Fri Jan 17 07:29:35 2014
new/usr/src/common/acpica/compiler/aslength.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslength - Tree walk to determine package and opcode lengths
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"
47 #include "amlcode.h"

50 #define _COMPONENT          ACPI_COMPILER
51     ACPI_MODULE_NAME        ("aslength")

53 /* Local prototypes */

55 static UINT8
56 CgGetPackageLenByteCount (
57     ACPI_PARSE_OBJECT *Op,
58     UINT32 PackageLength);
```

new/usr/src/common/acpica/compiler/aslength.c

2

```
60 static void
61 CgGenerateAmlOpcodeLength (
62     ACPI_PARSE_OBJECT *Op);

65 #ifdef ACPI_OBSOLETE_FUNCTIONS
66 void
67 LnAdjustLengthToRoot (
68     ACPI_PARSE_OBJECT *Op,
69     UINT32 LengthDelta);
70 #endif

73 /*****
74 *
75 * FUNCTION:      LnInitLengthsWalk
76 *
77 * PARAMETERS:    ASL_WALK_CALLBACK
78 *
79 * RETURN:        Status
80 *
81 * DESCRIPTION:    Walk callback to initialize (and re-initialize) the node
82 * subtree length(s) to zero. The Subtree lengths are bubbled
83 * up to the root node in order to get a total AML length.
84 *
85 *****/

87 ACPI_STATUS
88 LnInitLengthsWalk (
89     ACPI_PARSE_OBJECT *Op,
90     UINT32 Level,
91     void *Context)
92 {
94     Op->Asl.AmlSubtreeLength = 0;
95     return (AE_OK);
96 }

99 /*****
100 *
101 * FUNCTION:      LnPackageLengthWalk
102 *
103 * PARAMETERS:    ASL_WALK_CALLBACK
104 *
105 * RETURN:        Status
106 *
107 * DESCRIPTION:    Walk callback to calculate the total AML length.
108 * 1) Calculate the AML lengths (opcode, package length, etc.) for
109 * THIS node.
110 * 2) Bubble up all of these lengths to the parent node by summing
111 * them all into the parent subtree length.
112 *
113 * Note: The SubtreeLength represents the total AML length of all child nodes
114 * in all subtrees under a given node. Therefore, once this walk is
115 * complete, the Root Node subtree length is the AML length of the entire
116 * tree (and thus, the entire ACPI table)
117 *
118 *****/

120 ACPI_STATUS
121 LnPackageLengthWalk (
122     ACPI_PARSE_OBJECT *Op,
123     UINT32 Level,
124     void *Context)
125 {
```

```

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  * FUNCTION:      CgGetPackageLenByteCount
147  *
148  * PARAMETERS:    Op          - Parse node
149  *                PackageLength - Length to be encoded
150  *
151  * RETURN:        Required length of the package length encoding
152  *
153  * DESCRIPTION:   Calculate the number of bytes required to encode the given
154  *                package length.
155  *
156  *****/
157

159 static UINT8
160 CgGetPackageLenByteCount (
161     ACPI_PARSE_OBJECT *Op,
162     UINT32 PackageLength)
163 {
164
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 */
189
190         AslError (ASL_ERROR, ASL_MSG_ENCODING_LENGTH, Op, NULL);
191     }

```

```

193     return (0);
194 }

197 /*****
198  * FUNCTION:      CgGenerateAmlOpcodeLength
199  *
200  * PARAMETERS:    Op          - Parse node whose AML opcode lengths will be
201  *                                calculated
202  *
203  * RETURN:        None.
204  *
205  * DESCRIPTION:   Calculate the AmlOpcodeLength, AmlPkgLenBytes, and AmlLength
206  *                fields for this node.
207  *
208  *****/
209

211 static void
212 CgGenerateAmlOpcodeLength (
213     ACPI_PARSE_OBJECT *Op)
214 {
215
216     /* Check for two-byte opcode */
217
218     if (Op->Asl.AmlOpcode > 0x00FF)
219     {
220         Op->Asl.AmlOpcodeLength = 2;
221     }
222     else
223     {
224         Op->Asl.AmlOpcodeLength = 1;
225     }
226
227     /* Does this opcode have an associated "PackageLength" field? */
228
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     }
235
236     /* Data opcode lengths are easy */
237
238     switch (Op->Asl.AmlOpcode)
239     {
240     case AML_BYTE_OP:
241
242         Op->Asl.AmlLength = 1;
243         break;
244
245     case AML_WORD_OP:
246
247         Op->Asl.AmlLength = 2;
248         break;
249
250     case AML_DWORD_OP:
251
252         Op->Asl.AmlLength = 4;
253         break;
254
255     case AML_QWORD_OP:
256
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

```

new/usr/src/common/acpica/compiler/asllisting.c

1

```
*****
17879 Fri Jan 17 07:29:35 2014
new/usr/src/common/acpica/compiler/asllisting.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: asllisting - Listing file generation
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"
46 #include "amlcode.h"
47 #include "acparser.h"
48 #include "acnamesp.h"

51 #define _COMPONENT          ACPI_COMPILER
52 #define ACPI_MODULE_NAME    ("asllisting")

55 /* Local prototypes */

57 static void
58 LsGenerateListing (
59     UINT32                FileId);
```

new/usr/src/common/acpica/compiler/asllisting.c

2

```
61 static ACPI_STATUS
62 LsAmlListingWalk (
63     ACPI_PARSE_OBJECT      *Op,
64     UINT32                 Level,
65     void                   *Context);

67 static ACPI_STATUS
68 LsTreeWriteWalk (
69     ACPI_PARSE_OBJECT      *Op,
70     UINT32                 Level,
71     void                   *Context);

73 static void
74 LsWriteNodeToListing (
75     ACPI_PARSE_OBJECT      *Op,
76     UINT32                 FileId);

78 static void
79 LsFinishSourceListing (
80     UINT32                 FileId);

83 /*****
84 *
85 * FUNCTION:      LsDoListings
86 *
87 * PARAMETERS:    None. Examines the various output file global flags.
88 *
89 * RETURN:        None
90 *
91 * DESCRIPTION:   Generate all requested listing files.
92 *
93 *****/

95 void
96 LsDoListings (
97     void)
98 {
100     if (Gbl_C_OutputFlag)
101     {
102         LsGenerateListing (ASL_FILE_C_SOURCE_OUTPUT);
103     }

105     if (Gbl_ListingFlag)
106     {
107         LsGenerateListing (ASL_FILE_LISTING_OUTPUT);
108     }

110     if (Gbl_AsmOutputFlag)
111     {
112         LsGenerateListing (ASL_FILE_ASM_SOURCE_OUTPUT);
113     }

115     if (Gbl_C_IncludeOutputFlag)
116     {
117         LsGenerateListing (ASL_FILE_C_INCLUDE_OUTPUT);
118     }

120     if (Gbl_AsmIncludeOutputFlag)
121     {
122         LsGenerateListing (ASL_FILE_ASM_INCLUDE_OUTPUT);
123     }

125     if (Gbl_C_OffsetTableFlag)
```

```

126 {
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 {
149     /* Start at the beginning of both the source and AML files */
150
151     FlSeekFile (ASL_FILE_SOURCE_OUTPUT, 0);
152     FlSeekFile (ASL_FILE_AML_OUTPUT, 0);
153     Gbl_SourceLine = 0;
154     Gbl_CurrentHexColumn = 0;
155     LsPushNode (Gbl_Files[ASL_FILE_INPUT].Filename);
156
157     if (FileId == ASL_FILE_C_OFFSET_OUTPUT)
158     {
159         Gbl_CurrentAmlOffset = 0;
160
161         /* Offset table file has a special header and footer */
162
163         LsDoOffsetTableHeader (FileId);
164
165         TrWalkParseTree (RootNode, ASL_WALK_VISIT_DOWNWARD, LsAmlOffsetWalk,
166             NULL, (void *) ACPI_TO_POINTER (FileId));
167         LsDoOffsetTableFooter (FileId);
168         return;
169     }
170
171     /* Process all parse nodes */
172
173     TrWalkParseTree (RootNode, ASL_WALK_VISIT_DOWNWARD, LsAmlListingWalk,
174         NULL, (void *) ACPI_TO_POINTER (FileId));
175
176     /* Final processing */
177
178     LsFinishSourceListing (FileId);
179 }

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);
204
205     LsWriteNodeToListing (Op, FileId);
206
207     if (Op->Asl.CompileFlags & NODE_IS_RESOURCE_DATA)
208     {
209         /* Buffer is a resource template, don't dump the data all at once */
210
211         return (AE_OK);
212     }
213
214     /* Write the hex bytes to the listing file(s) (if requested) */
215
216     for (i = 0; i < Op->Asl.FinalAmlLength; i++)
217     {
218         if (ACPI_FAILURE (FlReadFile (ASL_FILE_AML_OUTPUT, &FileByte, 1)))
219         {
220             FlFileError (ASL_FILE_AML_OUTPUT, ASL_MSG_READ);
221             AslAbort ();
222         }
223         LsWriteListingHexBytes (&FileByte, 1, FileId);
224     }
225
226     return (AE_OK);
227 }

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 {
247     if (!Gbl_DebugFlag)
248     {
249         return;
250     }
251
252     DbgPrint (ASL_TREE_OUTPUT, "\nOriginal parse tree from parser:\n\n");
253     TrWalkParseTree (RootNode, ASL_WALK_VISIT_DOWNWARD,
254         LsTreeWriteWalk, NULL, NULL);
255 }

```

```

259 static ACPI_STATUS
260 LsTreeWriteWalk (
261     ACPI_PARSE_OBJECT *Op,
262     UINT32 Level,
263     void *Context)
264 {
265
266     /* Debug output */
267
268     DbgPrint (ASL_TREE_OUTPUT,
269         "%5.5d [%2d]", Op->Asl.LogicalLineNumber, Level);
270
271     UtPrintFormattedName (Op->Asl.ParseOpcode, Level);
272
273     DbgPrint (ASL_TREE_OUTPUT, "    (%.4X)\n", Op->Asl.ParseOpcode);
274     return (AE_OK);
275 }
276
277 /*****
278  *
279  * FUNCTION:      LsWriteNodeToListing
280  *
281  * PARAMETERS:    Op          - Parse node to write to the listing file.
282  *                FileId      - ID of current listing file
283  *
284  * RETURN:        None.
285  *
286  * DESCRIPTION:   Write "a node" to the listing file. This means to
287  *                1) Write out all of the source text associated with the node
288  *                2) Write out all of the AML bytes associated with the node
289  *                3) Write any compiler exceptions associated with the node
290  *
291  *****/
292
293 static void
294 LsWriteNodeToListing (
295     ACPI_PARSE_OBJECT *Op,
296     UINT32 FileId)
297 {
298     const ACPI_OPCODE_INFO *OpInfo;
299     UINT32 OpClass;
300     char *Pathname;
301     UINT32 Length;
302     UINT32 i;
303
304     OpInfo = AcpiPsGetOpcodeInfo (Op->Asl.AmlOpcode);
305     OpClass = OpInfo->Class;
306
307     /* TBD: clean this up with a single flag that says:
308      * I start a named output block
309      */
310     if (FileId == ASL_FILE_C_SOURCE_OUTPUT)
311     {
312         switch (Op->Asl.ParseOpcode)
313         {
314             case PARSEOP_DEFINITIONBLOCK:
315             case PARSEOP_METHODCALL:
316             case PARSEOP_INCLUDE:
317             case PARSEOP_INCLUDE_END:
318             case PARSEOP_DEFAULT_ARG:
319
320                 break;

```

```

324     default:
325
326         switch (OpClass)
327         {
328             case AML_CLASS_NAMED_OBJECT:
329
330                 switch (Op->Asl.AmlOpcode)
331                 {
332                     case AML_SCOPE_OP:
333                     case AML_ALIAS_OP:
334
335                         break;
336
337                     default:
338
339                         if (Op->Asl.ExternalName)
340                         {
341                             LsFlushListingBuffer (FileId);
342                             FlPrintFile (FileId, "    }; \n");
343                         }
344                         break;
345                     }
346                 break;
347
348             default:
349
350                 /* Don't care about other objects */
351
352                 break;
353         }
354     }
355 }
356
357 /* These cases do not have a corresponding AML opcode */
358
359 switch (Op->Asl.ParseOpcode)
360 {
361     case PARSEOP_DEFINITIONBLOCK:
362
363         LsWriteSourceLines (Op->Asl.EndLine, Op->Asl.EndLogicalLine, FileId);
364
365         /* Use the table Signature and TableId to build a unique name */
366
367         if (FileId == ASL_FILE_ASM_SOURCE_OUTPUT)
368         {
369             FlPrintFile (FileId,
370                 "%s%s_Header \\ \n",
371                 Gbl_TableSignature, Gbl_TableId);
372         }
373         if (FileId == ASL_FILE_C_SOURCE_OUTPUT)
374         {
375             FlPrintFile (FileId,
376                 "    unsigned char    %s%s_Header [] = \n    { \n",
377                 Gbl_TableSignature, Gbl_TableId);
378         }
379         if (FileId == ASL_FILE_ASM_INCLUDE_OUTPUT)
380         {
381             FlPrintFile (FileId,
382                 "extern %s%s_Header : byte \n",
383                 Gbl_TableSignature, Gbl_TableId);
384         }
385         if (FileId == ASL_FILE_C_INCLUDE_OUTPUT)
386         {
387             FlPrintFile (FileId,
388                 "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 */
523
524      Pathname = AcpiNsGetExternalPathname (Op->Asl.Node);
525      Length = strlen (Pathname);
526      if (Length >= 4)
527      {
528          /* Convert all dots in the path to underscores */
529
530          for (i = 0; i < Length; i++)
531          {
532              if (Pathname[i] == '.')
533              {
534                  Pathname[i] = '_';
535              }
536          }
537
538          /* Create the appropriate symbol in the output file */
539
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                  "extern %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;
568
569      default:
570
571          /* Nothing to do for listing file */
572
573          break;
574      }
575  }
576  break;
577
578  case AML_CLASS_EXECUTE:
579  case AML_CLASS_CREATE:
580  default:
581
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;
591
592      case AML_CLASS_UNKNOWN:
593
594          break;
595      }
596  }
597
598  /*****
599  *
600  * FUNCTION:      LsFinishSourceListing
601  *
602  * PARAMETERS:    FileId          - ID of current listing file.
603  *
604  * RETURN:        None
605  *
606  * DESCRIPTION:   Cleanup routine for the listing file. Flush the hex AML
607  *                listing buffer, and flush out any remaining lines in the
608  *                source input file.
609  *
610  *****/
611
612  static void
613  LsFinishSourceListing (
614      UINT32
615      FileId)
616  {
617      if ((FileId == ASL_FILE_ASM_INCLUDE_OUTPUT) ||
618          (FileId == ASL_FILE_C_INCLUDE_OUTPUT))
619      {
620          return;
621      }
622
623      LsFlushListingBuffer (FileId);
624      Gbl_CurrentAmlOffset = 0;
625
626      /* Flush any remaining text in the source file */
627
628      if (FileId == ASL_FILE_C_SOURCE_OUTPUT)
629      {
630          FlPrintFile (FileId, "    /*\n");
631      }
632
633      while (LsWriteOneSourceLine (FileId))
634      { ; }
635
636      if (FileId == ASL_FILE_C_SOURCE_OUTPUT)
637      {
638          FlPrintFile (FileId, "\n    */\n    };\n");
639      }
640
641      FlPrintFile (FileId, "\n");
642
643      if (FileId == ASL_FILE_LISTING_OUTPUT)
644      {
645          /* Print a summary of the compile exceptions */
646
647          FlPrintFile (FileId, "\n\nSummary of errors and warnings\n\n");
648          AePrintErrorLog (FileId);
649          FlPrintFile (FileId, "\n");
650          UtDisplaySummary (FileId);
651          FlPrintFile (FileId, "\n");
652      }
653  }

```

654 }

```

*****
19439 Fri Jan 17 07:29:35 2014
new/usr/src/common/acpica/compiler/asllistsup.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: asllistsup - Listing file support utilities
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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    ("asllistsup")

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, "  \n");
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, "\n");
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, "  \n");
121     for (i = 0; i < Count; i++)
122     {
123         LastChar = BufChar;
124         BufChar = Buffer[i];

126         if (isprint (BufChar))

```



```

127     {
128         /* Handle embedded C comment sequences */
129
130         if (((LastChar == '*'') && (BufChar == '/'')) ||
131             ((LastChar == '/'') && (BufChar == '*'')))
132         {
133             /* Insert a space to break the sequence */
134
135             FlPrintFile (FileId, ".", BufChar);
136         }
137
138         FlPrintFile (FileId, "%c", BufChar);
139     }
140     else
141     {
142         /* Not a printable character, just put out a dot */
143
144         FlPrintFile (FileId, ".");
145     }
146 }
147
148 FlPrintFile (FileId, "\\");
149 }

```

```

152 /*****
153 *
154 * FUNCTION:    LsCheckException
155 *
156 * PARAMETERS:  LineNumber      - Current logical (cumulative) line #
157 *              FileId          - ID of output listing file
158 *
159 * RETURN:      None
160 *
161 * DESCRIPTION: Check if there is an exception for this line, and if there is,
162 *              put it in the listing immediately. Handles multiple errors
163 *              per line. Gbl_NextError points to the next error in the
164 *              sorted (by line #) list of compile errors/warnings.
165 *
166 *****/
167
168 void
169 LsCheckException (
170     UINT32      LineNumber,
171     UINT32      FileId)
172 {
173
174     if ((!Gbl_NextError) ||
175         (LineNumber < Gbl_NextError->LogicalLineNumber))
176     {
177         return;
178     }
179
180     /* Handle multiple errors per line */
181
182     if (FileId == ASL_FILE_LISTING_OUTPUT)
183     {
184         while (Gbl_NextError &&
185             (LineNumber >= Gbl_NextError->LogicalLineNumber))
186         {
187             AePrintException (FileId, Gbl_NextError, "\\n[****iasl****]\\n");
188
189             Gbl_NextError = Gbl_NextError->Next;
190         }
191
192         FlPrintFile (FileId, "\\n");

```

```

193     }
194 }

```

```

197 /*****
198 *
199 * FUNCTION:    LsWriteListingHexBytes
200 *
201 * PARAMETERS:  Buffer          - AML code buffer
202 *              Length         - Number of AML bytes to write
203 *              FileId         - ID of current listing file.
204 *
205 * RETURN:      None
206 *
207 * DESCRIPTION: Write the contents of the AML buffer to the listing file via
208 *              the listing buffer. The listing buffer is flushed every 16
209 *              AML bytes.
210 *
211 *****/
212
213 void
214 LsWriteListingHexBytes (
215     UINT8      *Buffer,
216     UINT32     Length,
217     UINT32     FileId)
218 {
219     UINT32     i;
220
221     /* Transfer all requested bytes */
222
223     for (i = 0; i < Length; i++)
224     {
225         /* Print line header when buffer is empty */
226
227         if (Gbl_CurrentHexColumn == 0)
228         {
229             if (Gbl_HasIncludeFiles)
230             {
231                 FlPrintFile (FileId, "%*s", 10, " ");
232             }
233
234             switch (FileId)
235             {
236             case ASL_FILE_LISTING_OUTPUT:
237
238                 FlPrintFile (FileId, "%8.8X%s", Gbl_CurrentAmlOffset,
239                     ASL_LISTING_LINE_PREFIX);
240                 break;
241
242             case ASL_FILE_ASM_SOURCE_OUTPUT:
243
244                 FlPrintFile (FileId, "    db ");
245                 break;
246
247             case ASL_FILE_C_SOURCE_OUTPUT:
248
249                 FlPrintFile (FileId, "        ");
250                 break;
251
252             default:
253
254                 /* No other types supported */
255
256                 return;
257             }
258

```

```

259     }

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 /*****
524 *
525 * FUNCTION:      LsFlushListingBuffer
526 *
527 * PARAMETERS:   FileId          - ID of the listing file
528 *
529 * RETURN:       None
530 *
531 * DESCRIPTION:  Flush out the current contents of the 16-byte hex AML code
532 *               buffer. Usually called at the termination of a single line
533 *               of source code or when the buffer is full.
534 *
535 *****/

537 void
538 LsFlushListingBuffer (
539     UINT32          FileId)
540 {
541     UINT32          i;

544     if (Gbl_CurrentHexColumn == 0)
545     {
546         return;
547     }

549     /* Write the hex bytes */

551     switch (FileId)
552     {
553     case ASL_FILE_LISTING_OUTPUT:

555         for (i = 0; i < Gbl_CurrentHexColumn; i++)
556         {
557             FlPrintFile (FileId, "%2.2X ", Gbl_AmlBuffer[i]);
558         }

560         for (i = 0; i < ((HEX_LISTING_LINE_SIZE - Gbl_CurrentHexColumn) * 3); i++)
561         {
562             FlWriteFile (FileId, ".", 1);
563         }

565         /* Write the ASCII character associated with each of the bytes */

567         LsDumpAscii (FileId, Gbl_CurrentHexColumn, Gbl_AmlBuffer);
568         break;

571     case ASL_FILE_ASM_SOURCE_OUTPUT:

573         for (i = 0; i < Gbl_CurrentHexColumn; i++)
574         {
575             if (i > 0)
576             {
577                 FlPrintFile (FileId, ",");
578             }
579             FlPrintFile (FileId, "0%2.2Xh", Gbl_AmlBuffer[i]);
580         }

582         for (i = 0; i < ((HEX_LISTING_LINE_SIZE - Gbl_CurrentHexColumn) * 5); i++)
583         {
584             FlWriteFile (FileId, " ", 1);
585         }

587         FlPrintFile (FileId, "    ;%8.8X",
588                     Gbl_CurrentAmlOffset - HEX_LISTING_LINE_SIZE);

```

```

590     /* Write the ASCII character associated with each of the bytes */

592     LsDumpAscii (FileId, Gbl_CurrentHexColumn, Gbl_AmlBuffer);
593     break;

596     case ASL_FILE_C_SOURCE_OUTPUT:

598         for (i = 0; i < Gbl_CurrentHexColumn; i++)
599         {
600             FlPrintFile (FileId, "0x%2.2X", Gbl_AmlBuffer[i]);
601         }

603         /* Pad hex output with spaces if line is shorter than max line size */

605         for (i = 0; i < ((HEX_LISTING_LINE_SIZE - Gbl_CurrentHexColumn) * 5); i++)
606         {
607             FlWriteFile (FileId, " ", 1);
608         }

610         /* AML offset for the start of the line */

612         FlPrintFile (FileId, "    /* %8.8X",
613                     Gbl_CurrentAmlOffset - Gbl_CurrentHexColumn);

615         /* Write the ASCII character associated with each of the bytes */

617         LsDumpAsciiInComment (FileId, Gbl_CurrentHexColumn, Gbl_AmlBuffer);
618         FlPrintFile (FileId, " */");
619         break;

621     default:

623         /* No other types supported */

625         return;
626     }

628     FlPrintFile (FileId, "\n");

630     Gbl_CurrentHexColumn = 0;
631     Gbl_HexBytesWereWritten = TRUE;
632 }

635 /*****
636 *
637 * FUNCTION:      LsPushNode
638 *
639 * PARAMETERS:   Filename          - Pointer to the include filename
640 *
641 * RETURN:       None
642 *
643 * DESCRIPTION:  Push a listing node on the listing/include file stack. This
644 *               stack enables tracking of include files (infinitely nested)
645 *               and resumption of the listing of the parent file when the
646 *               include file is finished.
647 *
648 *****/

650 void
651 LsPushNode (
652     char          *Filename)
653 {
654     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 }
```

new/usr/src/common/acpica/compiler/aslload.c

1

```
*****
26830 Fri Jan 17 07:29:35 2014
new/usr/src/common/acpica/compiler/aslload.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dswload - Dispatcher namespace load callbacks
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __ASLLOAD_C__

46 #include "aslcompiler.h"
47 #include "amlcode.h"
48 #include "acdispat.h"
49 #include "acnamesp.h"

51 #include "aslcompiler.y.h"

53 #define _COMPONENT          ACPI_COMPILER
54     ACPI_MODULE_NAME      ("aslload")

56 /* Local prototypes */

58 static ACPI_STATUS
59 LdLoadFieldElements (
```

new/usr/src/common/acpica/compiler/aslload.c

2

```
60     ACPI_PARSE_OBJECT      *Op,
61     ACPI_WALK_STATE        *WalkState);

63 static ACPI_STATUS
64 LdLoadResourceElements (
65     ACPI_PARSE_OBJECT      *Op,
66     ACPI_WALK_STATE        *WalkState);

68 static ACPI_STATUS
69 LdNamespacelBegin (
70     ACPI_PARSE_OBJECT      *Op,
71     UINT32                 Level,
72     void                   *Context);

74 static ACPI_STATUS
75 LdNamespace2Begin (
76     ACPI_PARSE_OBJECT      *Op,
77     UINT32                 Level,
78     void                   *Context);

80 static ACPI_STATUS
81 LdCommonNamespaceEnd (
82     ACPI_PARSE_OBJECT      *Op,
83     UINT32                 Level,
84     void                   *Context);

87 /*****
88 *
89 * FUNCTION:      LdLoadNamespace
90 *
91 * PARAMETERS:    RootOp      - Root of the parse tree
92 *
93 * RETURN:        Status
94 *
95 * DESCRIPTION:   Perform a walk of the parse tree that in turn loads all of the
96 *               named ASL/AML objects into the namespace. The namespace is
97 *               constructed in order to resolve named references and references
98 *               to named fields within resource templates/descriptors.
99 *
100 *****/

102 ACPI_STATUS
103 LdLoadNamespace (
104     ACPI_PARSE_OBJECT      *RootOp)
105 {
106     ACPI_WALK_STATE        *WalkState;

109     DbgPrint (ASL_DEBUG_OUTPUT, "\nCreating namespace\n\n");

111     /* Create a new walk state */

113     WalkState = AcpiDsCreateWalkState (0, NULL, NULL, NULL);
114     if (!WalkState)
115     {
116         return (AE_NO_MEMORY);
117     }

119     /* Walk the entire parse tree, first pass */

121     TrWalkParseTree (RootOp, ASL_WALK_VISIT_TWICE, LdNamespacelBegin,
122                     LdCommonNamespaceEnd, WalkState);

124     /* Second pass to handle forward references */
```

```

126 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         }
214       }

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)
259 {
260     ACPI_PARSE_OBJECT     *InitializerOp = NULL;
261     ACPI_NAMESPACE_NODE    *Node;
262     ACPI_STATUS            Status;

265     /*
266     * Enter the resource name into the namespace. Name must not already exist.
267     * This opens a scope, so later field names are guaranteed to be new/unique.
268     */
269     Status = AcpiNsLookup (WalkState->ScopeInfo, Op->Asl.Namepath,
270         ACPI_TYPE_LOCAL_RESOURCE, ACPI_IMODE_LOAD_PASS1,
271         ACPI_NS_NO_UPSEARCH | ACPI_NS_ERROR_IF_FOUND,
272         WalkState, &Node);
273     if (ACPI_FAILURE (Status))
274     {
275         if (Status == AE_ALREADY_EXISTS)
276         {
277             /* Actual node causing the error was saved in ParentMethod */

279             AslError (ASL_ERROR, ASL_MSG_NAME_EXISTS,
280                 (ACPI_PARSE_OBJECT *) Op->Asl.ParentMethod, Op->Asl.Namepath);
281             return (AE_OK);
282         }
283         return (Status);
284     }

286     Node->Value = (UINT32) Op->Asl.Value.Integer;
287     Node->Op = Op;
288     Op->Asl.Node = Node;

290     /*
291     * Now enter the predefined fields, for easy lookup when referenced
292     * by the source ASL
293     */
294     InitializerOp = ASL_GET_CHILD_NODE (Op);
295     while (InitializerOp)
296     {
297         if (InitializerOp->Asl.ExternalName)
298         {
299             Status = AcpiNsLookup (WalkState->ScopeInfo,
300                 InitializerOp->Asl.ExternalName,
301                 ACPI_TYPE_LOCAL_RESOURCE_FIELD,
302                 ACPI_IMODE_LOAD_PASS1,
303                 ACPI_NS_NO_UPSEARCH | ACPI_NS_DONT_OPEN_SCOPE,
304                 NULL, &Node);
305             if (ACPI_FAILURE (Status))
306             {
307                 return (Status);
308             }

310             /*
311             * Store the field offset and length in the namespace node
312             * so it can be used when the field is referenced
313             */
314             Node->Value = InitializerOp->Asl.Value.Tag.BitOffset;
315             Node->Length = InitializerOp->Asl.Value.Tag.BitLength;
316             InitializerOp->Asl.Node = Node;
317             Node->Op = InitializerOp;
318         }

320         InitializerOp = ASL_GET_PEER_NODE (InitializerOp);
321     }

323     return (AE_OK);

```

```

324 }

327 /*****
328  *
329  * FUNCTION:      LdNamespacelBegin
330  *
331  * PARAMETERS:    ASL_WALK_CALLBACK
332  *
333  * RETURN:        Status
334  *
335  * DESCRIPTION:    Descending callback used during the parse tree walk. If this
336  *                  is a named AML opcode, enter into the namespace
337  *
338  *****/

340 static ACPI_STATUS
341 LdNamespacelBegin (
342     ACPI_PARSE_OBJECT      *Op,
343     UINT32                 Level,
344     void                   *Context)
345 {
346     ACPI_WALK_STATE        *WalkState = (ACPI_WALK_STATE *) Context;
347     ACPI_NAMESPACE_NODE    *Node;
348     ACPI_STATUS            Status;
349     ACPI_OBJECT_TYPE        ObjectType;
350     ACPI_OBJECT_TYPE        ActualObjectType = ACPI_TYPE_ANY;
351     char                   *Path;
352     UINT32                 Flags = ACPI_NS_NO_UPSEARCH;
353     ACPI_PARSE_OBJECT      *Arg;
354     UINT32                 i;
355     BOOLEAN                 ForceNewScope = FALSE;

358     ACPI_FUNCTION_NAME (LdNamespacelBegin);
359     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Op %p [%s]\n",
360         Op, Op->Asl.ParseOpName));

363     /*
364     * We are only interested in opcodes that have an associated name
365     * (or multiple names)
366     */
367     switch (Op->Asl.AmlOpcode)
368     {
369     case AML_BANK_FIELD_OP:
370     case AML_INDEX_FIELD_OP:
371     case AML_FIELD_OP:

373         Status = LdLoadFieldElements (Op, WalkState);
374         return (Status);

376     default:

378         /* All other opcodes go below */

380         break;
381     }

383     /* Check if this object has already been installed in the namespace */

385     if (Op->Asl.Node)
386     {
387         return (AE_OK);
388     }

```



```

390 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     *     Method (X)
450     *     {
451     *         Store (\_SB_.PCI0.ABCD, Local0)
452     *     }
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 [%s]",
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_OPEN,
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 }

```

new/usr/src/common/acpica/compiler/aslookup.c

1

```
*****
7479 Fri Jan 17 07:29:35 2014
new/usr/src/common/acpica/compiler/aslookup.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: aslookup- Namespace lookup functions
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "aslcompiler.h"
45 #include "aslcompiler.y.h"
46 #include "acparser.h"
47 #include "amlcode.h"
48 #include "acnamesp.h"
49 #include "acdispat.h"
50
51 #define _COMPONENT          ACPI_COMPILER
52 #define ACPI_MODULE_NAME    ("aslookup")
53
54 /* Local prototypes */
55
56 static ACPI_STATUS
```

new/usr/src/common/acpica/compiler/aslookup.c

2

```
59 LkIsObjectUsed (
60     ACPI_HANDLE          ObjHandle,
61     UINT32               Level,
62     void                 *Context,
63     void                 **ReturnValue);
64
65 static ACPI_PARSE_OBJECT *
66 LkGetNameOp (
67     ACPI_PARSE_OBJECT    *Op);
68
69 /*****
70 *
71 * FUNCTION:      LkFindUnreferencedObjects
72 *
73 * PARAMETERS:    None
74 *
75 * RETURN:        None
76 *
77 * DESCRIPTION:   Namespace walk to find objects that are not referenced in any
78 * way. Must be called after the namespace has been cross
79 * referenced.
80 *
81 *
82 *****/
83
84 void
85 LkFindUnreferencedObjects (
86     void)
87 {
88     /* Walk entire namespace from the supplied root */
89
90     (void) AcpiNsWalkNamespace (ACPI_TYPE_ANY, ACPI_ROOT_OBJECT,
91                                ACPI_UINT32_MAX, FALSE, LkIsObjectUsed, NULL,
92                                NULL, NULL);
93 }
94
95 /*****
96 *
97 * FUNCTION:      LkIsObjectUsed
98 *
99 * PARAMETERS:    ACPI_WALK_CALLBACK
100 *
101 * RETURN:        Status
102 *
103 * DESCRIPTION:   Check for an unreferenced namespace object and emit a warning.
104 * We have to be careful, because some types and names are
105 * typically or always unreferenced, we don't want to issue
106 * excessive warnings. Note: Names that are declared within a
107 * control method are temporary, so we always issue a remark
108 * if they are not referenced.
109 *
110 *
111 *
112 *****/
113
114 static ACPI_STATUS
115 LkIsObjectUsed (
116     ACPI_HANDLE          ObjHandle,
117     UINT32               Level,
118     void                 *Context,
119     void                 **ReturnValue)
120 {
121     ACPI_NAMESPACE_NODE *Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, ObjHandle);
122     ACPI_NAMESPACE_NODE *Next;
```

```

125  /* 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  }

```

new/usr/src/common/acpica/compiler/aslmain.c

1

```
*****
13086 Fri Jan 17 07:29:36 2014
new/usr/src/common/acpica/compiler/aslmain.c
update to acpica-unix2-20140114
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 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define _DECLARE_GLOBALS
45
46 #include "aslcompiler.h"
47 #include "acapps.h"
48 #include "acdisasm.h"
49 #include <signal.h>
50
51 #define _COMPONENT          ACPI_COMPILER
52 #define ACPI_MODULE_NAME    ("aslmain")
53
54 /*
55 * Main routine for the iASL compiler.
56 *
57 * Portability note: The compiler depends upon the host for command-line
```

new/usr/src/common/acpica/compiler/aslmain.c

2

```
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 */
65
66 /* Local prototypes */
67
68 static void ACPI_SYSTEM_XFACE
69 AslSignalHandler (
70     int                Sig);
71
72 static void
73 AslInitialize (
74     void);
75
76
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 *****/
89
90 void
91 Usage (
92     void)
93 {
94     printf ("%s\n\n", ASL_COMPLIANCE);
95     ACPI_USAGE_HEADER ("iasl [Options] [Files]");
96
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");
105
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");
112
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");
118
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;

264     for (i = 0; i < ASL_NUM_FILES; i++)
265     {
266         Gbl_Files[i].Handle = NULL;
267         Gbl_Files[i].Filename = NULL;
268     }

269     Gbl_Files[ASL_FILE_STDOUT].Handle = stdout;
270     Gbl_Files[ASL_FILE_STDOUT].Filename = "STDOUT";

271     Gbl_Files[ASL_FILE_STDERR].Handle = stderr;
272     Gbl_Files[ASL_FILE_STDERR].Filename = "STDERR";
273 }

274 /*****
275 * FUNCTION:      main
276 * PARAMETERS:    Standard argc/argv
277 * RETURN:        Program termination code
278 * DESCRIPTION:   C main routine for the Asl Compiler. Handle command line
279 *                options and begin the compile for each file on the command line
280 *****/
281 int ACPI_SYSTEM_XFACE
282 main (
283     int          argc,
284     char         **argv)
285 {
286     ACPI_STATUS   Status;
287     int           Index1;
288     int           Index2;

289     ACPI_DEBUG_INITIALIZE (); /* For debug version only */

290     /* Initialize preprocessor and compiler before command line processing */

291     signal (SIGINT, AslSignalHandler);
292     AcpiGbl_ExternalFileList = NULL;
293     AcpiDbgLevel = 0;
294     PrInitializePreprocessor ();
295     AslInitialize ();

296     Index1 = Index2 = AslCommandLine (argc, argv);

297     /* Allocate the line buffer(s), must be after command line */

298     Gbl_LineBufferSize /= 2;
299     UtExpandLineBuffers ();

300     /* 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     }

331     if (Gbl_DisassembleAll)
332     {
333         while (argv[Index1])
334         {
335             Status = AcpiDmAddToExternalFileList (argv[Index1]);
336             if (ACPI_FAILURE (Status))
337             {
338                 return (-1);
339             }
340             Index1++;
341         }
342     }

343     /* Process each pathname/filename in the list, with possible wildcards */

344     while (argv[Index2])
345     {
346         /*
347          * If -p not specified, we will use the input filename as the
348          * output filename prefix
349          */
350         if (Gbl_UseDefaultAmlFilename)
351         {
352             Gbl_OutputFilenamePrefix = argv[Index2];
353             UtConvertBackslashes (Gbl_OutputFilenamePrefix);
354         }

355         Status = AslDoOneFile (argv[Index2]);
356         if (ACPI_FAILURE (Status))
357         {
358             return (-1);
359         }

360         Index2++;
361     }

362     if (AcpiGbl_ExternalFileList)
363     {
364         AcpiDmClearExternalFileList();
365     }

366     return (0);
367 }

```

new/usr/src/common/acpica/compiler/aslmap.c

1

```
*****
48474 Fri Jan 17 07:29:36 2014
new/usr/src/common/acpica/compiler/aslmap.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslmap - parser to AML opcode mapping table
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "amlcode.h"
46 #include "acparser.h"

49 #define _COMPONENT          ACPI_COMPILER
50       ACPI_MODULE_NAME    ("aslmap")

53 /*****
54 *
55 * FUNCTION:      AslMapNamedOpcodeToDataType
56 *
57 * PARAMETERS:    Opcode                - The Named AML opcode to map
58 *
59 * RETURN:        The ACPI type associated with the named opcode
60 *****/
```

new/usr/src/common/acpica/compiler/aslmap.c

2

```
60 *
61 * DESCRIPTION: Convert a raw Named AML opcode to the associated data type.
62 *             Named opcodes are a subset of the AML opcodes.
63 *
64 *****/

66 ACPI_OBJECT_TYPE
67 AslMapNamedOpcodeToDataType (
68     UINT16      Opcode)
69 {
70     const ACPI_OPCODE_INFO *OpInfo;

73     /*
74      * There are some differences from the opcode table types, we
75      * catch them here.
76      */
77     OpInfo = AcpiPsGetOpcodeInfo (Opcode);

79     if (Opcode == AML_INT_NAMEPATH_OP)
80     {
81         return (ACPI_TYPE_ANY);
82     }

84     if (Opcode == AML_INT_METHODCALL_OP)
85     {
86         return (ACPI_TYPE_ANY);
87     }

89     if (OpInfo->Flags & AML_NSOBJECT)
90     {
91         return (OpInfo->ObjectType);
92     }

94     return (ACPI_TYPE_ANY);
95 }

98 /*****
99 *
100 * DATA STRUCTURE: AslKeywordMapping
101 *
102 * DESCRIPTION:      Maps the ParseOpcode to the actual AML opcode. The parse
103 *                   opcodes are generated from Bison, and this table must
104 *                   track any additions to them.
105 *
106 * Each entry in the table contains the following items:
107 *
108 * AML opcode       - Opcode that is written to the AML file
109 * Value            - Value of the object to be written (if applicable)
110 * Flags            - 1) Whether this opcode opens an AML "package".
111 *
112 *****/
113 /*
114 * TBD:
115 * AccessAttrib
116 * AccessType
117 * AMlop for DMA?
118 * ObjectType keywords
119 * Register
120 */

122 const ASL_MAPPING_ENTRY AslKeywordMapping [] =
123 {
124     /*! [Begin] no source code translation (keep the table structure) */
```

```

127 /* ACCESSAS */ OP_TABLE_ENTRY (AML_INT_ACCESSFIELD_OP, 0,
128 /* ACCESSATTRIB_BLOCK */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
129 /* ACCESSATTRIB_BLOCK_CALL */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
130 /* ACCESSATTRIB_BYTE */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
131 /* ACCESSATTRIB_MULTIBYTE */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
132 /* ACCESSATTRIB_QUICK */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
133 /* ACCESSATTRIB_RAW_BYTES */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
134 /* ACCESSATTRIB_RAW_PROCESS */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
135 /* ACCESSATTRIB_SND_RCV */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
136 /* ACCESSATTRIB_WORD */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
137 /* ACCESSATTRIB_WORD_CALL */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
138 /* ACESSTYPE_ANY */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
139 /* ACESSTYPE_BUF */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
140 /* ACESSTYPE_BYTE */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
141 /* ACESSTYPE_DWORD */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
142 /* ACESSTYPE_QWORD */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
143 /* ACESSTYPE_WORD */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
144 /* ACQUIRE */ OP_TABLE_ENTRY (AML_ACQUIRE_OP, 0,
145 /* ADD */ OP_TABLE_ENTRY (AML_ADD_OP, 0,
146 /* ADDRESSINGMODE_7BIT */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
147 /* ADDRESSINGMODE_10BIT */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
148 /* ADDRESSTYPE_ACPI */ OP_TABLE_ENTRY (AML_BYTE_OP, 2,
149 /* ADDRESSTYPE_MEMORY */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
150 /* ADDRESSTYPE_NVS */ OP_TABLE_ENTRY (AML_BYTE_OP, 3,
151 /* ADDRESSTYPE_RESERVED */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
152 /* ALIAS */ OP_TABLE_ENTRY (AML_ALIAS_OP, 0,
153 /* AND */ OP_TABLE_ENTRY (AML_BIT_AND_OP, 0,
154 /* ARG0 */ OP_TABLE_ENTRY (AML_ARG0, 0,
155 /* ARG1 */ OP_TABLE_ENTRY (AML_ARG1, 0,
156 /* ARG2 */ OP_TABLE_ENTRY (AML_ARG2, 0,
157 /* ARG3 */ OP_TABLE_ENTRY (AML_ARG3, 0,
158 /* ARG4 */ OP_TABLE_ENTRY (AML_ARG4, 0,
159 /* ARG5 */ OP_TABLE_ENTRY (AML_ARG5, 0,
160 /* ARG6 */ OP_TABLE_ENTRY (AML_ARG6, 0,
161 /* BANKFIELD */ OP_TABLE_ENTRY (AML_BANK_FIELD_OP, 0,
162 /* BITSPERBYTE_EIGHT */ OP_TABLE_ENTRY (AML_BYTE_OP, 3,
163 /* BITSPERBYTE_FIVE */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
164 /* BITSPERBYTE_NINE */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
165 /* BITSPERBYTE_SEVEN */ OP_TABLE_ENTRY (AML_BYTE_OP, 2,
166 /* BITSPERBYTE_SIX */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
167 /* BREAK */ OP_TABLE_ENTRY (AML_BREAK_OP, 0,
168 /* BREAKPOINT */ OP_TABLE_ENTRY (AML_BREAK_POINT_OP, 0,
169 /* BUFFER */ OP_TABLE_ENTRY (AML_BUFFER_OP, 0,
170 /* BUSMASTERTYPE_MASTER */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
171 /* BUSMASTERTYPE_NOTMASTER */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
172 /* BYTECONST */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, 0,
173 /* CASE */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
174 /* CLOCKPHASE_FIRST */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
175 /* CLOCKPHASE_SECOND */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
176 /* CLOCKPOLARITY_HIGH */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
177 /* CLOCKPOLARITY_LOW */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
178 /* CONCATENATE */ OP_TABLE_ENTRY (AML_CONCAT_OP, 0,
179 /* CONCATENATERESTEMPLATE */ OP_TABLE_ENTRY (AML_CONCAT_RES_OP, 0,
180 /* CONDREFOF */ OP_TABLE_ENTRY (AML_COND_REF_OF_OP, 0,
181 /* CONNECTION */ OP_TABLE_ENTRY (AML_INT_CONNECTION_OP, 0,
182 /* CONTINUE */ OP_TABLE_ENTRY (AML_CONTINUE_OP, 0,
183 /* COPY */ OP_TABLE_ENTRY (AML_COPY_OP, 0,
184 /* CREATEBITFIELD */ OP_TABLE_ENTRY (AML_CREATE_BIT_FIELD_OP, 0,
185 /* CREATEBYTEFIELD */ OP_TABLE_ENTRY (AML_CREATE_BYTE_FIELD_OP, 0,
186 /* CREATDWORDFIELD */ OP_TABLE_ENTRY (AML_CREATE_DWORD_FIELD_OP, 0,
187 /* CREATEFIELD */ OP_TABLE_ENTRY (AML_CREATE_FIELD_OP, 0,
188 /* CREATEQWORDFIELD */ OP_TABLE_ENTRY (AML_CREATE_QWORD_FIELD_OP, 0,
189 /* CREATWORDFIELD */ OP_TABLE_ENTRY (AML_CREATE_WORD_FIELD_OP, 0,
190 /* DATABUFFER */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
191 /* DATATABLEREGION */ OP_TABLE_ENTRY (AML_DATA_REGION_OP, 0,

```

```

192 /* DEBUG */ OP_TABLE_ENTRY (AML_DEBUG_OP, 0,
193 /* DECODETYPE_POS */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
194 /* DECODETYPE_SUB */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
195 /* DECREMENT */ OP_TABLE_ENTRY (AML_DECREMENT_OP, 0,
196 /* DEFAULT */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
197 /* DEFAULT_ARG */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
198 /* DEFINITIONBLOCK */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
199 /* DEREF_OF */ OP_TABLE_ENTRY (AML_DEREF_OF_OP, 0,
200 /* DEVICE */ OP_TABLE_ENTRY (AML_DEVICE_OP, 0,
201 /* DEVICEPOLARITY_HIGH */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
202 /* DEVICEPOLARITY_LOW */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
203 /* DIVIDE */ OP_TABLE_ENTRY (AML_DIVIDE_OP, 0,
204 /* DMA */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
205 /* DMATYPE_A */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
206 /* DMATYPE_COMPATIBILITY */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
207 /* DMATYPE_B */ OP_TABLE_ENTRY (AML_BYTE_OP, 2,
208 /* DMATYPE_F */ OP_TABLE_ENTRY (AML_BYTE_OP, 3,
209 /* DWORDCONST */ OP_TABLE_ENTRY (AML_RAW_DATA_DWORD, 0,
210 /* DWORDIO */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
211 /* DWORDMEMORY */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
212 /* DWORDSPACE */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
213 /* EISAID */ OP_TABLE_ENTRY (AML_DWORD_OP, 0,
214 /* ELSE */ OP_TABLE_ENTRY (AML_ELSE_OP, 0,
215 /* ELSEIF */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
216 /* ENDEPENDENTFN */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
217 /* ENDIAN_BIG */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
218 /* ENDIAN_LITTLE */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
219 /* ENDTAG */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
220 /* ERRORNODE */ OP_TABLE_ENTRY (AML_NOOP_OP, 0,
221 /* EVENT */ OP_TABLE_ENTRY (AML_EVENT_OP, 0,
222 /* EXTENDEDIO */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
223 /* EXTENDEDMEMORY */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
224 /* EXTENDEDSPACE */ OP_TABLE_ENTRY (AML_RAW_DATA_QWORD, 0,
225 /* EXTERNAL */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
226 /* FATAL */ OP_TABLE_ENTRY (AML_FATAL_OP, 0,
227 /* FIELD */ OP_TABLE_ENTRY (AML_FIELD_OP, 0,
228 /* FINDSETLEFTBIT */ OP_TABLE_ENTRY (AML_FIND_SET_LEFT_BIT_OP, 0,
229 /* FINDSETRIGHTBIT */ OP_TABLE_ENTRY (AML_FIND_SET_RIGHT_BIT_OP, 0,
230 /* FIXEDDMA */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
231 /* FIXEDIO */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
232 /* FLOWCONTROL_HW */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
233 /* FLOWCONTROL_NONE */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
234 /* FLOWCONTROL_SW */ OP_TABLE_ENTRY (AML_BYTE_OP, 2,
235 /* FROMBCD */ OP_TABLE_ENTRY (AML_FROM_BCD_OP, 0,
236 /* FUNCTION */ OP_TABLE_ENTRY (AML_METHOD_OP, 0,
237 /* GPIOINT */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
238 /* GPIOIO */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
239 /* I2CSERIALBUS */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
240 /* IF */ OP_TABLE_ENTRY (AML_IF_OP, 0,
241 /* INCLUDE */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
242 /* INCLUDE_END */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
243 /* INCREMENT */ OP_TABLE_ENTRY (AML_INCREMENT_OP, 0,
244 /* INDEX */ OP_TABLE_ENTRY (AML_INDEX_OP, 0,
245 /* INDEXFIELD */ OP_TABLE_ENTRY (AML_INDEX_FIELD_OP, 0,
246 /* INTEGER */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
247 /* INTERRUPT */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
248 /* INTLEVEL_ACTIVEBOTH */ OP_TABLE_ENTRY (AML_BYTE_OP, 2,
249 /* INTLEVEL_ACTIVEHIGH */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
250 /* INTLEVEL_ACTIVELOW */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
251 /* INTTYPE_EDGE */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
252 /* INTTYPE_LEVEL */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
253 /* IO */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
254 /* IODECODETYPE_10 */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
255 /* IODECODETYPE_16 */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
256 /* IORESTRICT_IN */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
257 /* IORESTRICT_NONE */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,

```

```

258 /* IORESTRICT_OUT */ OP_TABLE_ENTRY (AML_BYTE_OP, 2,
259 /* IORESTRICT_PRESERVE */ OP_TABLE_ENTRY (AML_BYTE_OP, 3,
260 /* IRQ */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
261 /* IRQNOFLAGS */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
262 /* LAND */ OP_TABLE_ENTRY (AML_LAND_OP, 0,
263 /* LEQUAL */ OP_TABLE_ENTRY (AML_LEQUAL_OP, 0,
264 /* LGREATER */ OP_TABLE_ENTRY (AML_LGREATER_OP, 0,
265 /* LGREATEREQUAL */ OP_TABLE_ENTRY (AML_LGREATEREQUAL_OP, 0,
266 /* LLESS */ OP_TABLE_ENTRY (AML_LLESS_OP, 0,
267 /* LLESSEQUAL */ OP_TABLE_ENTRY (AML_LLESSEQUAL_OP, 0,
268 /* LNOT */ OP_TABLE_ENTRY (AML_LNOT_OP, 0,
269 /* LNOTEQUAL */ OP_TABLE_ENTRY (AML_LNOTEQUAL_OP, 0,
270 /* LOAD */ OP_TABLE_ENTRY (AML_LOAD_OP, 0,
271 /* LOADTABLE */ OP_TABLE_ENTRY (AML_LOAD_TABLE_OP, 0,
272 /* LOCAL0 */ OP_TABLE_ENTRY (AML_LOCAL0, 0,
273 /* LOCAL1 */ OP_TABLE_ENTRY (AML_LOCAL1, 0,
274 /* LOCAL2 */ OP_TABLE_ENTRY (AML_LOCAL2, 0,
275 /* LOCAL3 */ OP_TABLE_ENTRY (AML_LOCAL3, 0,
276 /* LOCAL4 */ OP_TABLE_ENTRY (AML_LOCAL4, 0,
277 /* LOCAL5 */ OP_TABLE_ENTRY (AML_LOCAL5, 0,
278 /* LOCAL6 */ OP_TABLE_ENTRY (AML_LOCAL6, 0,
279 /* LOCAL7 */ OP_TABLE_ENTRY (AML_LOCAL7, 0,
280 /* LOCKRULE_LOCK */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
281 /* LOCKRULE_NOLOCK */ OP_TABLE_ENTRY (AML_BYTE_OP, AML_
282 /* LOR */ OP_TABLE_ENTRY (AML_LOR_OP, 0,
283 /* MATCH */ OP_TABLE_ENTRY (AML_MATCH_OP, 0,
284 /* MATCHTYPE_MEQ */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, MATC
285 /* MATCHTYPE_MGE */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, MATC
286 /* MATCHTYPE_MGT */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, MATC
287 /* MATCHTYPE_MLE */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, MATC
288 /* MATCHTYPE_MLT */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, MATC
289 /* MATCHTYPE_MTR */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, MATC
290 /* MAXTYPE_FIXED */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
291 /* MAXTYPE_NOTFIXED */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
292 /* MEMORY24 */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
293 /* MEMORY32 */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
294 /* MEMORY32FIXED */ OP_TABLE_ENTRY (AML_DEFAULT_ARG_OP, 0,
295 /* MEMTYPE_CACHEABLE */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
296 /* MEMTYPE_NONCACHEABLE */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
297 /* MEMTYPE_PREFETCHABLE */ OP_TABLE_ENTRY (AML_BYTE_OP, 3,
298 /* MEMTYPE_WRITECOMBINING */ OP_TABLE_ENTRY (AML_BYTE_OP, 2,
299 /* METHOD */ OP_TABLE_ENTRY (AML_METHOD_OP, 0,
300 /* METHODCALL */ OP_TABLE_ENTRY (AML_INT_METHODCALL_OP, 0,
301 /* MID */ OP_TABLE_ENTRY (AML_MID_OP, 0,
302 /* MINTYPE_FIXED */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
303 /* MINTYPE_NOTFIXED */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
304 /* MOD */ OP_TABLE_ENTRY (AML_MOD_OP, 0,
305 /* MULTIPLY */ OP_TABLE_ENTRY (AML_MULTIPLY_OP, 0,
306 /* MUTEX */ OP_TABLE_ENTRY (AML_MUTEX_OP, 0,
307 /* NAME */ OP_TABLE_ENTRY (AML_NAME_OP, 0,
308 /* NAMESEG */ OP_TABLE_ENTRY (AML_INT_NAMEPATH_OP, 0,
309 /* NAMESTRING */ OP_TABLE_ENTRY (AML_INT_NAMEPATH_OP, 0,
310 /* NAND */ OP_TABLE_ENTRY (AML_BIT_NAND_OP, 0,
311 /* NOOP */ OP_TABLE_ENTRY (AML_NOOP_OP, 0,
312 /* NOR */ OP_TABLE_ENTRY (AML_BIT_NOR_OP, 0,
313 /* NOT */ OP_TABLE_ENTRY (AML_BIT_NOT_OP, 0,
314 /* NOTIFY */ OP_TABLE_ENTRY (AML_NOTIFY_OP, 0,
315 /* OBJECTTYPE */ OP_TABLE_ENTRY (AML_TYPE_OP, 0,
316 /* OBJECTTYPE_BFF */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
317 /* OBJECTTYPE_BUF */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
318 /* OBJECTTYPE_DDB */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
319 /* OBJECTTYPE_DEV */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
320 /* OBJECTTYPE_EVT */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
321 /* OBJECTTYPE_FLD */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
322 /* OBJECTTYPE_INT */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
323 /* OBJECTTYPE_MTH */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI

```

```

324 /* OBJECTTYPE_MTX */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
325 /* OBJECTTYPE_OPR */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
326 /* OBJECTTYPE_PKG */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
327 /* OBJECTTYPE_POW */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
328 /* OBJECTTYPE_PRO */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
329 /* OBJECTTYPE_STR */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
330 /* OBJECTTYPE_THZ */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
331 /* OBJECTTYPE_UNK */ OP_TABLE_ENTRY (AML_BYTE_OP, ACPI
332 /* OFFSET */ OP_TABLE_ENTRY (AML_INT_RESERVEDFIELD_OP, 0,
333 /* ONE */ OP_TABLE_ENTRY (AML_ONE_OP, 0,
334 /* ONES */ OP_TABLE_ENTRY (AML_ONES_OP, 0,
335 /* OPERATIONREGION */ OP_TABLE_ENTRY (AML_REGION_OP, 0,
336 /* OR */ OP_TABLE_ENTRY (AML_BIT_OR_OP, 0,
337 /* PACKAGE */ OP_TABLE_ENTRY (AML_PACKAGE_OP, 0,
338 /* PACKAGE_LENGTH */ OP_TABLE_ENTRY (AML_PACKAGE_LENGTH, 0,
339 /* PARITYTYPE_EVEN */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
340 /* PARITYTYPE_MARK */ OP_TABLE_ENTRY (AML_BYTE_OP, 3,
341 /* PARITYTYPE_NONE */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
342 /* PARITYTYPE_ODD */ OP_TABLE_ENTRY (AML_BYTE_OP, 2,
343 /* PARITYTYPE_SPACE */ OP_TABLE_ENTRY (AML_BYTE_OP, 4,
344 /* PIN_NOPULL */ OP_TABLE_ENTRY (AML_BYTE_OP, 3,
345 /* PIN_PULLDEFAULT */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
346 /* PIN_PULLDOWN */ OP_TABLE_ENTRY (AML_BYTE_OP, 2,
347 /* PIN_PULLUP */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
348 /* POWERRESOURCE */ OP_TABLE_ENTRY (AML_POWER_RES_OP, 0,
349 /* PROCESSOR */ OP_TABLE_ENTRY (AML_PROCESSOR_OP, 0,
350 /* QWORDCONST */ OP_TABLE_ENTRY (AML_RAW_DATA_QWORD, 0,
351 /* QWORDIO */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
352 /* QWORDMEMORY */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
353 /* QWORDSPACE */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
354 /* RANGE_TYPE_ENTIRE */ OP_TABLE_ENTRY (AML_BYTE_OP, 3,
355 /* RANGE_TYPE_ISAONLY */ OP_TABLE_ENTRY (AML_BYTE_OP, 2,
356 /* RANGE_TYPE_NONISAONLY */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
357 /* RAW_DATA */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
358 /* READWRITETYPE_BOTH */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
359 /* READWRITETYPE_READONLY */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
360 /* REFOP */ OP_TABLE_ENTRY (AML_REF_OF_OP, 0,
361 /* REGIONSPEACE_CMOS */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, ACPI
362 /* REGIONSPEACE_EC */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, ACPI
363 /* REGIONSPEACE_FFIXEDHW */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, ACPI
364 /* REGIONSPEACE_GPIO */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, ACPI
365 /* REGIONSPEACE_GSBUS */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, ACPI
366 /* REGIONSPEACE_IO */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, ACPI
367 /* REGIONSPEACE_IPMI */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, ACPI
368 /* REGIONSPEACE_MEM */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, ACPI
369 /* REGIONSPEACE_PCC */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, ACPI
370 /* REGIONSPEACE_PCI */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, ACPI
371 /* REGIONSPEACE_PCIBAR */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, ACPI
372 /* REGIONSPEACE_SMBUS */ OP_TABLE_ENTRY (AML_RAW_DATA_BYTE, ACPI
373 /* REGISTER */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
374 /* RELEASE */ OP_TABLE_ENTRY (AML_RELEASE_OP, 0,
375 /* RESERVED_BYTES */ OP_TABLE_ENTRY (AML_INT_RESERVEDFIELD_OP, 0,
376 /* RESET */ OP_TABLE_ENTRY (AML_RESET_OP, 0,
377 /* RESOURCETEMPLATE */ OP_TABLE_ENTRY (AML_BUFFER_OP, 0,
378 /* RESOURCETYPE_CONSUMER */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
379 /* RESOURCETYPE_PRODUCER */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
380 /* RETURN */ OP_TABLE_ENTRY (AML_RETURN_OP, 0,
381 /* REVISION */ OP_TABLE_ENTRY (AML_REVISION_OP, 0,
382 /* SCOPE */ OP_TABLE_ENTRY (AML_SCOPE_OP, 0,
383 /* SERIALIZERULE_NOTSERIAL */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
384 /* SERIALIZERULE_SERIAL */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
385 /* SHARETYPE_EXCLUSIVE */ OP_TABLE_ENTRY (AML_BYTE_OP, 0,
386 /* SHARETYPE_EXCLUSIVEWAKE */ OP_TABLE_ENTRY (AML_BYTE_OP, 2,
387 /* SHARETYPE_SHARED */ OP_TABLE_ENTRY (AML_BYTE_OP, 1,
388 /* SHARETYPE_SHAREDWAKE */ OP_TABLE_ENTRY (AML_BYTE_OP, 3,
389 /* SHIFLEFT */ OP_TABLE_ENTRY (AML_SHIFT_LEFT_OP, 0,

```

```

390 /* 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 /* SPI SERIALBUS */      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 };

```

new/usr/src/common/acpica/compiler/aslmessages.h

1

```
*****
21841 Fri Jan 17 07:29:36 2014
new/usr/src/common/acpica/compiler/aslmessages.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: aslmessages.h - Compiler error/warning messages
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #ifndef __ASLMESSAGES_H
45 #define __ASLMESSAGES_H
46
47 typedef enum
48 {
49     ASL_OPTIMIZATION = 0,
50     ASL_REMARK,
51     ASL_WARNING,
52     ASL_WARNING2,
53     ASL_WARNING3,
54     ASL_ERROR,
55     ASL_NUM_REPORT_LEVELS
56 }
```

new/usr/src/common/acpica/compiler/aslmessages.h

2

```
59 } ASL_MESSAGE_TYPES;
60
61 #ifdef ASL_EXCEPTIONS
62
63 /* Strings for message reporting levels, must match values above */
64
65 const char
66     *AslErrorLevel [ASL_NUM_REPORT_LEVELS] = {
67     "Optimize",
68     "Remark ",
69     "Warning ",
70     "Warning ",
71     "Warning ",
72     "Error "
73 };
74
75 /* All lowercase versions for IDEs */
76
77 const char
78     *AslErrorLevelIde [ASL_NUM_REPORT_LEVELS] = {
79     "optimize",
80     "remark ",
81     "warning ",
82     "warning ",
83     "warning ",
84     "error "
85 };
86
87 #define ASL_ERROR_LEVEL_LENGTH 8 /* Length of strings above */
88 #endif
89
90 /*
91 * Values for all compiler messages.
92 *
93 * NOTE: With the introduction of the -vw option to disable specific messages,
94 * new messages should only be added to the end of this list, so that values
95 * for existing messages are not disturbed.
96 */
97 typedef enum
98 {
99     ASL_MSG_RESERVED = 0,
100     ASL_MSG_ALIGNMENT,
101     ASL_MSG_ALPHANUMERIC_STRING,
102     ASL_MSG_AML_NOT_IMPLEMENTED,
103     ASL_MSG_ARG_COUNT_HI,
104     ASL_MSG_ARG_COUNT_LO,
105     ASL_MSG_ARG_INIT,
106     ASL_MSG_BACKWARDS_OFFSET,
107     ASL_MSG_BUFFER_LENGTH,
108     ASL_MSG_CLOSE,
109     ASL_MSG_COMPILER_INTERNAL,
110     ASL_MSG_COMPILER_RESERVED,
111     ASL_MSG_CONNECTION_MISSING,
112     ASL_MSG_CONNECTION_INVALID,
113     ASL_MSG_CONSTANT_EVALUATION,
114     ASL_MSG_CONSTANT_FOLDED,
115     ASL_MSG_CORE_EXCEPTION,
116     ASL_MSG_DEBUG_FILE_OPEN,
117     ASL_MSG_DEBUG_FILENAME,
118     ASL_MSG_DEPENDENT_NESTING,
119     ASL_MSG_DMA_CHANNEL,
120     ASL_MSG_DMA_LIST,
121     ASL_MSG_DUPLICATE_CASE,
122     ASL_MSG_DUPLICATE_ITEM,
123     ASL_MSG_EARLY_EOF,
124     ASL_MSG_ENCODING_LENGTH,
125     ASL_MSG_EX_INTERRUPT_LIST,
```

```
125 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 expres",
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-ASCII",
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 declar",
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_RETURN */ "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 argue",
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
390 /* ASL_MSG_RESERVED_RETURN_VALUE */ "Reserved method must return a value
391 /* ASL_MSG_RESERVED_USE */ "Invalid use of reserved name",
392 /* ASL_MSG_RESERVED_WORD */ "Use of reserved name",
393 /* ASL_MSG_RESOURCE_FIELD */ "Resource field name cannot be used
394 /* ASL_MSG_RESOURCE_INDEX */ "Missing ResourceSourceIndex (requir
395 /* ASL_MSG_RESOURCE_LIST */ "Too many resource items (internal e
396 /* ASL_MSG_RESOURCE_SOURCE */ "Missing ResourceSource string (requ
397 /* ASL_MSG_RESULT_NOT_USED */ "Result is not used, operator has no
398 /* ASL_MSG_RETURN_TYPES */ "Not all control paths return a valu
399 /* ASL_MSG_SCOPE_FWD_REF */ "Forward references from Scope opera
400 /* ASL_MSG_SCOPE_TYPE */ "Existing object has invalid type fo
401 /* ASL_MSG_SEEK */ "Could not seek file",
402 /* ASL_MSG_SERIALIZED */ "Control Method marked Serialized",
403 /* ASL_MSG_SERIALIZED_REQUIRED */ "Control Method should be made Seria
404 /* ASL_MSG_SINGLE_NAME_OPTIMIZATION */ "NamePath optimized to NameSeg (uses
405 /* ASL_MSG_SOME_NO_RETVAL */ "Called method may not always return
406 /* ASL_MSG_STRING_LENGTH */ "String literal too long",
407 /* ASL_MSG_SWITCH_TYPE */ "Switch expression is not a static I
408 /* ASL_MSG_SYNC_LEVEL */ "SyncLevel must be in the range 0-15
409 /* ASL_MSG_SYNTAX */ "",
410 /* ASL_MSG_TABLE_SIGNATURE */ "Invalid Table Signature",
411 /* ASL_MSG_TAG_LARGER */ "ResourceTag larger than Field",
412 /* ASL_MSG_TAG_SMALLER */ "ResourceTag smaller than Field",
413 /* ASL_MSG_TIMEOUT */ "Result is not used, possible operat
414 /* ASL_MSG_TOO_MANY_TEMPS */ "Method requires too many temporary
415 /* ASL_MSG_TRUNCATION */ "64-bit return value will be truncat
416 /* ASL_MSG_UNKNOWN_RESERVED_NAME */ "Unknown reserved name",
417 /* ASL_MSG_UNREACHABLE_CODE */ "Statement is unreachable",
418 /* ASL_MSG_UNSUPPORTED */ "Unsupported feature",
419 /* ASL_MSG_UPPER_CASE */ "Non-hex letters must be upper case"
420 /* ASL_MSG_VENDOR_LIST */ "Too many vendor data bytes (7 max)"
421 /* ASL_MSG_WRITE */ "Could not write file",
422 /* ASL_MSG_RANGE */ "Constant out of range",
423 /* ASL_MSG_BUFFER_ALLOCATION */ "Could not allocate line buffer",

425 /* Preprocessor */

427 /* ASL_MSG_DIRECTIVE_SYNTAX */ "Invalid directive syntax",
428 /* ASL_MSG_ENDIF_MISMATCH */ "Mismatched #endif",
429 /* ASL_MSG_ERROR_DIRECTIVE */ "#error",
430 /* ASL_MSG_EXISTING_NAME */ "Name is already defined",
431 /* ASL_MSG_INVALID_INVOCATION */ "Invalid macro invocation",
432 /* ASL_MSG_MACRO_SYNTAX */ "Invalid macro syntax",
433 /* ASL_MSG_TOO_MANY_ARGUMENTS */ "Too many macro arguments",
434 /* ASL_MSG_UNKNOWN_DIRECTIVE */ "Unknown directive",
435 /* ASL_MSG_UNKNOWN_PRAGMA */ "Unknown pragma",
436 /* ASL_MSG_WARNING_DIRECTIVE */ "#warning",

438 /* Table compiler */

440 /* ASL_MSG_BUFFER_ELEMENT */ "Invalid element in buffer initializ
441 /* ASL_MSG_DIVIDE_BY_ZERO */ "Expression contains divide-by-zero"
442 /* ASL_MSG_FLAG_VALUE */ "Flag value is too large",
443 /* ASL_MSG_INTEGER_SIZE */ "Integer too large for target",
444 /* ASL_MSG_INVALID_EXPRESSION */ "Invalid expression",
445 /* ASL_MSG_INVALID_FIELD_NAME */ "Invalid Field Name",
446 /* ASL_MSG_INVALID_HEX_INTEGER */ "Invalid hex integer constant",
447 /* ASL_MSG_OEM_TABLE */ "OEM table - unknown contents",
448 /* ASL_MSG_RESERVED_VALUE */ "Reserved field must be zero",
449 /* ASL_MSG_UNKNOWN_LABEL */ "Label is undefined",
450 /* ASL_MSG_UNKNOWN_SUBTABLE */ "Unknown subtable type",
451 /* ASL_MSG_UNKNOWN_TABLE */ "Unknown ACPI table signature",
452 /* ASL_MSG_ZERO_VALUE */ "Value must be non-zero"
453 };

```

```

455 #endif /* ASL_EXCEPTIONS */
457 #endif /* __ASLMESSAGES_H */

```

new/usr/src/common/acpica/compiler/aslmethod.c

1

```
*****
20665 Fri Jan 17 07:29:36 2014
new/usr/src/common/acpica/compiler/aslmethod.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: aslmethod.c - Control method analysis walk
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"
47 #include "acparser.h"
48 #include "amlcode.h"

51 #define _COMPONENT          ACPI_COMPILER
52 #define ACPI_MODULE_NAME    ("aslmethod")

55 /* Local prototypes */

57 void
58 MtCheckNamedObjectInMethod (
59     ACPI_PARSE_OBJECT *Op,
60     ASL_METHOD_INFO *MethodInfo);
```

new/usr/src/common/acpica/compiler/aslmethod.c

2

```
63 /*****
64 *
65 * FUNCTION:      MtMethodAnalysisWalkBegin
66 *
67 * PARAMETERS:    ASL_WALK_CALLBACK
68 *
69 * RETURN:        Status
70 *
71 * DESCRIPTION:    Descending callback for the analysis walk. Check methods for:
72 *                  1) Initialized local variables
73 *                  2) Valid arguments
74 *                  3) Return types
75 *
76 *****/

78 ACPI_STATUS
79 MtMethodAnalysisWalkBegin (
80     ACPI_PARSE_OBJECT *Op,
81     UINT32 Level,
82     void *Context)
83 {
84     ASL_ANALYSIS_WALK_INFO *WalkInfo = (ASL_ANALYSIS_WALK_INFO *) Context;
85     ASL_METHOD_INFO *MethodInfo = WalkInfo->MethodStack;
86     ACPI_PARSE_OBJECT *Next;
87     UINT32 RegisterNumber;
88     UINT32 i;
89     char LocalName[] = "Local0";
90     char ArgName[] = "Arg0";
91     ACPI_PARSE_OBJECT *ArgNode;
92     ACPI_PARSE_OBJECT *NextType;
93     ACPI_PARSE_OBJECT *NextParamType;
94     UINT8 ActualArgs = 0;

97     switch (Op->Asl.ParseOpcode)
98     {
99     case PARSEOP_METHOD:

101         TotalMethods++;

103         /* Create and init method info */

105         MethodInfo = UtLocalCalloc (sizeof (ASL_METHOD_INFO));
106         MethodInfo->Next = WalkInfo->MethodStack;
107         MethodInfo->Op = Op;

109         WalkInfo->MethodStack = MethodInfo;

111         /* Get the name node, ignored here */

113         Next = Op->Asl.Child;

115         /* Get the NumArguments node */

117         Next = Next->Asl.Next;
118         MethodInfo->NumArguments = (UINT8)
119             (((UINT8) Next->Asl.Value.Integer) & 0x07);

121         /* Get the SerializeRule and SyncLevel nodes, ignored here */

123         Next = Next->Asl.Next;
124         MethodInfo->ShouldBeSerialized = (UINT8) Next->Asl.Value.Integer;

126         Next = Next->Asl.Next;
```

```

127     ArgNode = Next;
129     /* Get the Return type node */
131     Next = Next->Asl.Next;
133     NextType = Next->Asl.Child;
134     while (NextType)
135     {
136         /* Get and map each of the Return types */
138         MethodInfo->ValidReturnTypes |= AnMapObjTypeToBtype (NextType);
139         NextType->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
140         NextType = NextType->Asl.Next;
141     }
143     /* Get the Parameter type 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:
525         break;
526     }
527 }
528
529 switch (Op->Asl.ParseOpcode)
530 {
531     case PARSEOP_METHOD:
532
533         WalkInfo->MethodStack = MethodInfo->Next;
534
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         }
549
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         }
561
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         }
580
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);
591
592         /*
593          * Special check for _REG: Must have an operation region definition
594          * within the same scope!
595          */
596         ApCheckRegMethod (Op);
597     }
598
599     ACPI_FREE (MethodInfo);
600     break;
601
602     case PARSEOP_NAME:
603
604         /* Special check for two names like _L01 and _E01 in same scope */
605
606         ApCheckForGpeNameConflict (Op);
607         break;
608
609     case PARSEOP_RETURN:
610
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);
616
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;
622
623         /* Used in the "typing" pass later */
624
625         Op->Asl.ParentMethod = MethodInfo->Op;
626
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;
637
638     case PARSEOP_IF:
639
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;
652
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 */
673
674             Op->Asl.Parent->Asl.CompileFlags |= NODE_HAS_NO_EXIT;
675         }
676         break;
677     }

679     return (AE_OK);
680 }
```

new/usr/src/common/acpica/compiler/aslnamespace.c

1

```
*****
12397 Fri Jan 17 07:29:36 2014
new/usr/src/common/acpica/compiler/aslnamespace.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: aslnamespace - Namespace output file generation
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"
47 #include "acnamesp.h"

50 #define _COMPONENT          ACPI_COMPILER
51       ACPI_MODULE_NAME    ("aslnamespace")

53 /* Local prototypes */

55 static ACPI_STATUS
56 NsDoOneNamespaceObject (
57     ACPI_HANDLE             ObjHandle,
58     UINT32                  Level,
59     void                     *Context,
60     void                     **ReturnValue);
```

new/usr/src/common/acpica/compiler/aslnamespace.c

2

```
62 static ACPI_STATUS
63 NsDoOnePathname (
64     ACPI_HANDLE             ObjHandle,
65     UINT32                  Level,
66     void                     *Context,
67     void                     **ReturnValue);

70 /*****
71 *
72 * FUNCTION:      NsSetupNamespaceListing
73 *
74 * PARAMETERS:    Handle          - local file handle
75 *
76 * RETURN:        None
77 *
78 * DESCRIPTION:   Set the namespace output file to the input handle
79 *
80 *****/

82 void
83 NsSetupNamespaceListing (
84     void                     *Handle)
85 {
87     Gbl_NsOutputFlag = TRUE;
88     Gbl_Files[ASL_FILE_NAMESPACE_OUTPUT].Handle = Handle;
89 }

92 /*****
93 *
94 * FUNCTION:      NsDisplayNamespace
95 *
96 * PARAMETERS:    None
97 *
98 * RETURN:        Status
99 *
100 * DESCRIPTION:   Walk the namespace and display information about each node
101 *                in the tree. Information is written to the optional
102 *                namespace output file.
103 *
104 *****/

106 ACPI_STATUS
107 NsDisplayNamespace (
108     void)
109 {
110     ACPI_STATUS              Status;

113     if (!Gbl_NsOutputFlag)
114     {
115         return (AE_OK);
116     }

118     Gbl_NumNamespaceObjects = 0;

120     /* File header */

122     FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "Contents of ACPI Namespace\n\n");
123     FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "Count Depth  Name - Type\n\n");

125     /* Walk entire namespace from the root */
```



```

127 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     }
210 }
211 else
212 {
213     switch (Node->Type)
214     {
215         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:
261
262         switch (Op->Asl.ParseOpcode)
263         {
264             case PARSEOP_CREATEBYTEFIELD:
265
266                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "    [BYTE ( 8 bit)]");
267                 break;
268
269             case PARSEOP_CREATEDWORDFIELD:
270
271                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "    [DWORD (32 bit)]");
272                 break;
273
274             case PARSEOP_CREATEQWORDFIELD:
275
276                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "    [QWORD (64 bit)]");
277                 break;
278
279             case PARSEOP_CREATEWORDFIELD:
280
281                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "    [WORD (16 bit)]");
282                 break;
283
284             case PARSEOP_CREATEBITFIELD:
285
286                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "    [BIT ( 1 bit)]");
287                 break;
288
289             case PARSEOP_CREATEFIELD:
290
291                 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "    [Arbitrary Bit Field]");
292                 break;
293
294             default:
295
296                 break;
297         }
298     }
299     break;
300
301 case ACPI_TYPE_PACKAGE:
302
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;
313
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;
322
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;
335
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;
343
344 case ACPI_TYPE_METHOD:
345
346     FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
347                 "    [Code Length 0x%.4X bytes]",
348                 Op->Asl.AmlSubtreeLength);
349     break;
350
351 case ACPI_TYPE_LOCAL_RESOURCE:
352
353     FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
354                 "    [Desc Offset 0x%.4X Bytes]", Node->Value);
355     break;
356
357 case ACPI_TYPE_LOCAL_RESOURCE_FIELD:
358
359     FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT,
360                 "    [Field Offset 0x%.4X Bits 0x%.4X Bytes] ",
361                 Node->Value, Node->Value / 8);
362
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;
374
375 default:
376
377     /* Nothing to do for other types */
378
379     break;
380 }
381
382 FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "\n");
383 return (AE_OK);
384 }
385
386 /*****
387 *
388 * FUNCTION: NsDoOnePathname

```

```
391 *
392 * PARAMETERS:  ACPI_WALK_CALLBACK
393 *
394 * RETURN:      Status
395 *
396 * DESCRIPTION: Print the full pathname for a namespace node.
397 *
398 *****/
400 static ACPI_STATUS
401 NsDoOnePathname (
402     ACPI_HANDLE          ObjHandle,
403     UINT32               Level,
404     void                 *Context,
405     void                 **ReturnValue)
406 {
407     ACPI_NAMESPACE_NODE  *Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
408     ACPI_STATUS          Status;
409     ACPI_BUFFER           TargetPath;
410
411     TargetPath.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
412     Status = AcpiNsHandleToPathname (Node, &TargetPath);
413     if (ACPI_FAILURE (Status))
414     {
415         return (Status);
416     }
417
418     FlPrintFile (ASL_FILE_NAMESPACE_OUTPUT, "%s\n", TargetPath.Pointer);
419     ACPI_FREE (TargetPath.Pointer);
420
421     return (AE_OK);
422 }
423 }
```

new/usr/src/common/acpica/compiler/asloffset.c

1

```
*****
15308 Fri Jan 17 07:29:37 2014
new/usr/src/common/acpica/compiler/asloffset.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: asloffset - Generate a C "offset table" for BIOS use.
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"
46 #include "amlcode.h"
47 #include "acnamesp.h"

50 #define _COMPONENT          ACPI_COMPILER
51         ACPI_MODULE_NAME    ("asloffset")

54 /* Local prototypes */

56 static void
57 LsEmitOffsetTableEntry (
58     UINT32                FileId,
59     ACPI_NAMESPACE_NODE    *Node,
60     UINT32                NamepathOffset,
```

new/usr/src/common/acpica/compiler/asloffset.c

2

```
61     UINT32                Offset,
62     char                  *OpName,
63     UINT64                Value,
64     UINT8                 AmlOpcode,
65     UINT16                ParentOpcode);

68 /*****
69 *
70 * FUNCTION:      LsAmlOffsetWalk
71 *
72 * PARAMETERS:    ASL_WALK_CALLBACK
73 *
74 * RETURN:        Status
75 *
76 * DESCRIPTION:   Process one node during a offset table file generation.
77 *
78 * Three types of objects are currently emitted to the offset table:
79 * 1) Tagged (named) resource descriptors
80 * 2) Named integer objects with constant integer values
81 * 3) Named package objects
82 * 4) Operation Regions that have constant Offset (address) parameters
83 * 5) Control methods
84 *
85 * The offset table allows the BIOS to dynamically update the values of these
86 * objects at boot time.
87 *
88 *****/

90 ACPI_STATUS
91 LsAmlOffsetWalk (
92     ACPI_PARSE_OBJECT      *Op,
93     UINT32                 Level,
94     void                   *Context)
95 {
96     UINT32                 FileId = (UINT32) ACPI_TO_INTEGER (Context);
97     ACPI_NAMESPACE_NODE    *Node;
98     UINT32                 Length;
99     UINT32                 NamepathOffset;
100    UINT32                 DataOffset;
101    ACPI_PARSE_OBJECT        *NextOp;

104    /* Ignore actual data blocks for resource descriptors */

106    if (Op->Asl.CompileFlags & NODE_IS_RESOURCE_DATA)
107    {
108        return (AE_OK); /* Do NOT update the global AML offset */
109    }

111    /* We are only interested in named objects (have a namespace node) */

113    Node = Op->Asl.Node;
114    if (!Node)
115    {
116        Gbl_CurrentAmlOffset += Op->Asl.FinalAmlLength;
117        return (AE_OK);
118    }

120    /* Named resource descriptor (has a descriptor tag) */

122    if ((Node->Type == ACPI_TYPE_LOCAL_RESOURCE) &&
123        (Op->Asl.CompileFlags & NODE_IS_RESOURCE_DESC))
124    {
125        LsEmitOffsetTableEntry (FileId, Node, 0, Gbl_CurrentAmlOffset,
126                                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      "      {%s 0x%4.4X, 0x%8.8X, 0x%2.2X, 0x%8.8X, 0x%8.8X%8.8X}, /* %s */\n",
385      MsgBuffer, ParentOpcode, NamepathOffset, AmlOpcode,
386      Offset, ACPI_FORMAT_UINT64 (Value), OpName);
387  }

390  /*****

```

```

391 *
392 * FUNCTION:      LsDoOffsetTableHeader, LsDoOffsetTableFooter
393 *
394 * PARAMETERS:    FileId          - ID of current listing file
395 *
396 * RETURN:        None
397 *
398 * DESCRIPTION:   Header and footer for the offset table file.
399 *
400 *****/
401
402 void
403 LsDoOffsetTableHeader (
404     UINT32          FileId)
405 {
406
407     FlPrintFile (FileId,
408         "#ifndef __AML_OFFSET_TABLE_H\n"
409         "#define __AML_OFFSET_TABLE_H\n");
410
411     FlPrintFile (FileId, "typedef struct {\n"
412         "    char          *Pathname;      /* Full pathname (from root)\n"
413         "    unsigned short ParentOpcode;   /* AML opcode for the parent\n"
414         "    unsigned long  NamesegOffset;  /* Offset of last nameseg in\n"
415         "    unsigned char  Opcode;         /* AML opcode for the data *\n"
416         "    unsigned long  Offset;         /* Offset for the data */\n"
417         "    unsigned long long Value;      /* Original value of the dat\n"
418         "} AML_OFFSET_TABLE_ENTRY;\n");
419
420     FlPrintFile (FileId,
421         "#endif /* __AML_OFFSET_TABLE_H */\n");
422
423     FlPrintFile (FileId,
424         "/*\n"
425         " * Information specific to the supported object types:\n"
426         " *\n"
427         " * Integers:\n"
428         " *   Opcode is the integer prefix, indicates length of the data\n"
429         " *   (One of: BYTE, WORD, DWORD, QWORD, ZERO, ONE, ONES)\n"
430         " *   Offset points to the actual integer data\n"
431         " *   Value is the existing value in the AML\n"
432         " *\n"
433         " * Packages:\n"
434         " *   Opcode is the package or var_package opcode\n"
435         " *   Offset points to the package opcode\n"
436         " *   Value is the package element count\n"
437         " *\n"
438         " * Operation Regions:\n"
439         " *   Opcode is the address integer prefix, indicates length of the dat\n"
440         " *   Offset points to the region address\n"
441         " *   Value is the existing address value in the AML\n"
442         " *\n"
443         " * Control Methods:\n"
444         " *   Offset points to the method flags byte\n"
445         " *   Value is the existing flags value in the AML\n"
446         " *\n"
447         " * Processors:\n"
448         " *   Offset points to the first byte of the PBlock Address\n"
449         " *\n"
450         " * Resource Descriptors:\n"
451         " *   Opcode is the descriptor type\n"
452         " *   Offset points to the start of the descriptor\n"
453         " *\n"
454         " * Scopes/Devices/ThermalZones:\n"
455         " *   Nameseg offset only\n"
456         " */\n");

```

```

458     FlPrintFile (FileId,
459         "AML_OFFSET_TABLE_ENTRY    %s %s_OffsetTable[] =\n{\n",
460         Gbl_TableSignature, Gbl_TableId);
461 }
462
463 void
464 LsDoOffsetTableFooter (
465     UINT32          FileId)
466 {
467
468     FlPrintFile (FileId,
469         "    {NULL,0,0,0,0} /* Table terminator */\n};\n");
470     Gbl_CurrentAmlOffset = 0;
471 }

```

new/usr/src/common/acpica/compiler/aslopcodes.c

1

```
*****
22710 Fri Jan 17 07:29:37 2014
new/usr/src/common/acpica/compiler/aslopcodes.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslopcodes - AML opcode generation
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"
47 #include "amlcode.h"

49 #define _COMPONENT          ACPI_COMPILER
50       ACPI_MODULE_NAME    ("aslopcodes")

53 /* Local prototypes */

55 static void
56 OpcDoAccessAs (
57     ACPI_PARSE_OBJECT      *Op);

59 static void
```

new/usr/src/common/acpica/compiler/aslopcodes.c

2

```
60 OpcDoConnection (
61     ACPI_PARSE_OBJECT      *Op);

63 static void
64 OpcDoUnicode (
65     ACPI_PARSE_OBJECT      *Op);

67 static void
68 OpcDoEisaId (
69     ACPI_PARSE_OBJECT      *Op);

71 static void
72 OpcDoUuid (
73     ACPI_PARSE_OBJECT      *Op);

76 /*****
77 *
78 * FUNCTION:      OpcAmlOpcodeUpdateWalk
79 *
80 * PARAMETERS:    ASL_WALK_CALLBACK
81 *
82 * RETURN:        Status
83 *
84 * DESCRIPTION:    Opcode update walk, ascending callback
85 *
86 *****/

88 ACPI_STATUS
89 OpcAmlOpcodeUpdateWalk (
90     ACPI_PARSE_OBJECT      *Op,
91     UINT32                  Level,
92     void                    *Context)
93 {

95     /*
96     * Handle the Package() case where the actual opcode cannot be determined
97     * until the PackageLength operand has been folded and minimized.
98     * (PackageOp versus VarPackageOp)
99     *
100     * This is (as of ACPI 3.0) the only case where the AML opcode can change
101     * based upon the value of a parameter.
102     *
103     * The parser always inserts a VarPackage opcode, which can possibly be
104     * optimized to a Package opcode.
105     */
106     if (Op->Asl.ParseOpcode == PARSEOP_VAR_PACKAGE)
107     {
108         OpnDoPackage (Op);
109     }

111     return (AE_OK);
112 }

115 /*****
116 *
117 * FUNCTION:      OpcAmlOpcodeWalk
118 *
119 * PARAMETERS:    ASL_WALK_CALLBACK
120 *
121 * RETURN:        Status
122 *
123 * DESCRIPTION:    Parse tree walk to generate both the AML opcodes and the AML
124 * operands.
125 */
```



```

126 *****/
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  * FUNCTION: OpcGetIntegerWidth
145  *
146  * PARAMETERS: Op - DEFINITION BLOCK op
147  *
148  * RETURN: none
149  *
150  * DESCRIPTION: Extract integer width from the table revision
151  * *****/
152
153 void
154 OpcGetIntegerWidth (
155     ACPI_PARSE_OBJECT *Op)
156 {
157     ACPI_PARSE_OBJECT *Child;
158
159     if (!Op)
160     {
161         return;
162     }
163     if (Gbl_RevisionOverride)
164     {
165         AcpiUtSetIntegerWidth (Gbl_RevisionOverride);
166     }
167     else
168     {
169         Child = Op->Asl.Child;
170         Child = Child->Asl.Next;
171         Child = Child->Asl.Next;
172     }
173
174     /* Use the revision to set the integer width */
175     AcpiUtSetIntegerWidth ((UINT8) Child->Asl.Value.Integer);
176 }

177
178 /*****
179  * FUNCTION: OpcSetOptimalIntegerSize
180  *
181  * PARAMETERS: Op - A parse tree node
182  *
183  * RETURN: Integer width, in bytes. Also sets the node AML opcode to the
184  * optimal integer AML prefix opcode.
185  *****/

```

```

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 {
203     if 0
204     /*
205      * TBD: - we don't want to optimize integers in the block header, but the
206      * code below does not work correctly.
207      */
208     if (Op->Asl.Parent &&
209         Op->Asl.Parent->Asl.Parent &&
210         (Op->Asl.Parent->Asl.Parent->Asl.ParseOpcode == PARSEOP_DEFINITIONBLOCK))
211     {
212         return (0);
213     }
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:
229
230         Op->Asl.AmlOpcode = AML_ZERO_OP;
231         AslError (ASL_OPTIMIZATION, ASL_MSG_INTEGER_OPTIMIZATION,
232             Op, "Zero");
233         return (1);
234
235     case 1:
236
237         Op->Asl.AmlOpcode = AML_ONE_OP;
238         AslError (ASL_OPTIMIZATION, ASL_MSG_INTEGER_OPTIMIZATION,
239             Op, "One");
240         return (1);
241
242     case ACPI_UINT32_MAX:
243
244         /* Check for table integer width (32 or 64) */
245
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;
254
255     case ACPI_UINT64_MAX:
256
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:
269         break;
270     }
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);
297     }
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
391 *
392 * RETURN:      None
393 *
394 * DESCRIPTION: Implement the Connection ASL keyword.
395 *
396 *****/

398 static void
399 OpcDoConnection (
400     ACPI_PARSE_OBJECT *Op)
401 {
402     ASL_RESOURCE_NODE *Rnode;
403     ACPI_PARSE_OBJECT *BufferOp;
404     ACPI_PARSE_OBJECT *BufferLengthOp;
405     ACPI_PARSE_OBJECT *BufferDataOp;
406     UINT8 State;

409     Op->Asl.AmlOpcodeLength = 1;

411     if (Op->Asl.Child->Asl.AmlOpcode == AML_INT_NAMEPATH_OP)
412     {
413         return;
414     }

416     BufferOp = Op->Asl.Child;
417     BufferLengthOp = BufferOp->Asl.Child;
418     BufferDataOp = BufferLengthOp->Asl.Next;

420     State = ACPI_RSTATE_NORMAL;
421     Rnode = RsDoOneResourceDescriptor (BufferDataOp->Asl.Next, 0, &State);
422     if (!Rnode)
423     {
424         return; /* error */
425     }

427     /*
428     * Transform the nodes into the following
429     *
430     * Op      -> AML_BUFFER_OP
431     * First Child -> BufferLength
432     * Second Child -> Descriptor Buffer (raw byte data)
433     */
434     BufferOp->Asl.ParseOpcode = PARSEOP_BUFFER;
435     BufferOp->Asl.AmlOpcode = AML_BUFFER_OP;
436     BufferOp->Asl.CompileFlags = NODE_AML_PACKAGE | NODE_IS_RESOURCE_DESC;
437     UtSetParseOpName (BufferOp);

439     BufferLengthOp->Asl.ParseOpcode = PARSEOP_INTEGER;
440     BufferLengthOp->Asl.Value.Integer = Rnode->BufferLength;
441     (void) OpcSetOptimalIntegerSize (BufferLengthOp);
442     UtSetParseOpName (BufferLengthOp);

444     BufferDataOp->Asl.ParseOpcode = PARSEOP_RAW_DATA;
445     BufferDataOp->Asl.AmlOpcode = AML_RAW_DATA_CHAIN;
446     BufferDataOp->Asl.AmlOpcodeLength = 0;
447     BufferDataOp->Asl.AmlLength = Rnode->BufferLength;
448     BufferDataOp->Asl.Value.Buffer = (UINT8 *) Rnode;
449     UtSetParseOpName (BufferDataOp);
450 }

453 /*****
454 *
455 * FUNCTION:      OpcDoUnicode

```

```

456 *
457 * PARAMETERS: Op      - Parse node
458 *
459 * RETURN:      None
460 *
461 * DESCRIPTION: Implement the UNICODE ASL "macro". Convert the input string
462 *              to a unicode buffer. There is no Unicode AML opcode.
463 *
464 * Note: The Unicode string is 16 bits per character, no leading signature,
465 *       with a 16-bit terminating NULL.
466 *
467 *****/

469 static void
470 OpcDoUnicode (
471     ACPI_PARSE_OBJECT *Op)
472 {
473     ACPI_PARSE_OBJECT *InitializerOp;
474     UINT32 Length;
475     UINT32 Count;
476     UINT32 i;
477     UINT8 *AsciiString;
478     UINT16 *UnicodeString;
479     ACPI_PARSE_OBJECT *BufferLengthOp;

482     /* Change op into a buffer object */

484     Op->Asl.CompileFlags &= ~NODE_COMPILE_TIME_CONST;
485     Op->Asl.ParseOpcode = PARSEOP_BUFFER;
486     UtSetParseOpName (Op);

488     /* Buffer Length is first, followed by the string */

490     BufferLengthOp = Op->Asl.Child;
491     InitializerOp = BufferLengthOp->Asl.Next;

493     AsciiString = (UINT8 *) InitializerOp->Asl.Value.String;

495     /* Create a new buffer for the Unicode string */

497     Count = strlen (InitializerOp->Asl.Value.String) + 1;
498     Length = Count * sizeof (UINT16);
499     UnicodeString = UtLocalCalloc (Length);

501     /* Convert to Unicode string (including null terminator) */

503     for (i = 0; i < Count; i++)
504     {
505         UnicodeString[i] = (UINT16) AsciiString[i];
506     }

508     /*
509     * Just set the buffer size node to be the buffer length, regardless
510     * of whether it was previously an integer or a default_arg placeholder
511     */
512     BufferLengthOp->Asl.ParseOpcode = PARSEOP_INTEGER;
513     BufferLengthOp->Asl.AmlOpcode = AML_DWORD_OP;
514     BufferLengthOp->Asl.Value.Integer = Length;
515     UtSetParseOpName (BufferLengthOp);

517     (void) OpcSetOptimalIntegerSize (BufferLengthOp);

519     /* The Unicode string is a raw data buffer */

521     InitializerOp->Asl.Value.Buffer = (UINT8 *) UnicodeString;

```

```

522     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       - 3rd 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         }

613         if (ACPI_FAILURE (Status))
614         {
615             AslError (ASL_ERROR, ASL_MSG_INVALID_EISAID, Op, Op->Asl.Value.String);
616         }
617     }
618     else
619     {
620         /* Create ID big-endian first (bits are contiguous) */

622         BigEndianId =
623             (UINT32) ((UINT8) (InString[0] - 0x40)) << 26 |
624             (UINT32) ((UINT8) (InString[1] - 0x40)) << 21 |
625             (UINT32) ((UINT8) (InString[2] - 0x40)) << 16 |

626             (UIntCharToValue (InString[3])) << 12 |
627             (UIntCharToValue (InString[4])) << 8 |
628             (UIntCharToValue (InString[5])) << 4 |
629             UIntCharToValue (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
655 *
656 * PARAMETERS:    Op          - Parse node
657 *
658 * RETURN:        None
659 *
660 * DESCRIPTION:    Convert UUID string to 16-byte buffer
661 *
662 *****/

664 static void
665 OpcDoUuid (
666     ACPI_PARSE_OBJECT *Op)
667 {
668     char *InString;
669     char *Buffer;
670     ACPI_STATUS Status = AE_OK;
671     ACPI_PARSE_OBJECT *NewOp;

674     InString = (char *) Op->Asl.Value.String;
675     Buffer = UtLocalCalloc (16);

677     Status = AuValidateUuid (InString);
678     if (ACPI_FAILURE (Status))
679     {
680         AslError (ASL_ERROR, ASL_MSG_INVALID_UUID, Op, Op->Asl.Value.String);
681     }
682     else
683     {
684         (void) AuConvertStringToUuid (InString, Buffer);
685     }

687     /* Change Op to a Buffer */

689     Op->Asl.ParseOpcode = PARSEOP_BUFFER;
690     Op->Common.AmlOpcode = AML_BUFFER_OP;

692     /* Disable further optimization */

694     Op->Asl.CompileFlags &= ~NODE_COMPILE_TIME_CONST;
695     UtSetParseOpName (Op);

697     /* Child node is the buffer length */

699     NewOp = TrAllocateNode (PARSEOP_INTEGER);

701     NewOp->Asl.AmlOpcode = AML_BYTE_OP;
702     NewOp->Asl.Value.Integer = 16;
703     NewOp->Asl.Parent = Op;

705     Op->Asl.Child = NewOp;
706     Op = NewOp;

708     /* Peer to the child is the raw buffer data */

710     NewOp = TrAllocateNode (PARSEOP_RAW_DATA);
711     NewOp->Asl.AmlOpcode = AML_RAW_DATA_BUFFER;
712     NewOp->Asl.AmlLength = 16;
713     NewOp->Asl.Value.String = (char *) Buffer;
714     NewOp->Asl.Parent = Op->Asl.Parent;

716     Op->Asl.Next = NewOp;
717 }

```

```

720 *****/
721 *
722 * FUNCTION:      OpcGenerateAmlOpcode
723 *
724 * PARAMETERS:    Op          - Parse node
725 *
726 * RETURN:        None
727 *
728 * DESCRIPTION:    Generate the AML opcode associated with the node and its
729 *                  parse (lex/flex) keyword opcode. Essentially implements
730 *                  a mapping between the parse opcodes and the actual AML opcodes.
731 *
732 *****/

734 void
735 OpcGenerateAmlOpcode (
736     ACPI_PARSE_OBJECT *Op)
737 {
738     UINT16 Index;

742     Index = (UINT16) (Op->Asl.ParseOpcode - ASL_PARSE_OPCODE_BASE);

744     Op->Asl.AmlOpcode = AslKeywordMapping[Index].AmlOpcode;
745     Op->Asl.AcpibType = AslKeywordMapping[Index].AcpibType;
746     Op->Asl.CompileFlags |= AslKeywordMapping[Index].Flags;

748     if (!Op->Asl.Value.Integer)
749     {
750         Op->Asl.Value.Integer = AslKeywordMapping[Index].Value;
751     }

753     /* Special handling for some opcodes */

755     switch (Op->Asl.ParseOpcode)
756     {
757     case PARSEOP_INTEGER:
758         /*
759          * Set the opcode based on the size of the integer
760          */
761         (void) OpcSetOptimalIntegerSize (Op);
762         break;

764     case PARSEOP_OFFSET:

766         Op->Asl.AmlOpcodeLength = 1;
767         break;

769     case PARSEOP_ACCESSAS:

771         OpcDoAccessAs (Op);
772         break;

774     case PARSEOP_CONNECTION:

776         OpcDoConnection (Op);
777         break;

779     case PARSEOP_EISAID:

781         OpcDoEisaId (Op);
782         break;

784     case PARSEOP_TOUUID:

```

```
786     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 }
```

new/usr/src/common/acpica/compiler/asloperands.c

1

```
*****
31757 Fri Jan 17 07:29:37 2014
new/usr/src/common/acpica/compiler/asloperands.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: asloperands - AML operand processing
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "aslcompiler.h"
45 #include "aslcompiler.y.h"
46 #include "amlcode.h"
47
48 #define _COMPONENT          ACPI_COMPILER
49 #define ACPI_MODULE_NAME    ("asloperands")
50
51 /* Local prototypes */
52
53 static void
54 OpnDoField (
55     ACPI_PARSE_OBJECT      *Op);
56
57 static void
```

new/usr/src/common/acpica/compiler/asloperands.c

2

```
59 OpnDoBankField (
60     ACPI_PARSE_OBJECT      *Op);
61
62 static void
63 OpnDoBuffer (
64     ACPI_PARSE_OBJECT      *Op);
65
66 static void
67 OpnDoDefinitionBlock (
68     ACPI_PARSE_OBJECT      *Op);
69
70 static void
71 OpnDoFieldCommon (
72     ACPI_PARSE_OBJECT      *FieldOp,
73     ACPI_PARSE_OBJECT      *Op);
74
75 static void
76 OpnDoIndexField (
77     ACPI_PARSE_OBJECT      *Op);
78
79 static void
80 OpnDoLoadTable (
81     ACPI_PARSE_OBJECT      *Op);
82
83 static void
84 OpnDoMethod (
85     ACPI_PARSE_OBJECT      *Op);
86
87 static void
88 OpnDoMutex (
89     ACPI_PARSE_OBJECT      *Op);
90
91 static void
92 OpnDoRegion (
93     ACPI_PARSE_OBJECT      *Op);
94
95 static void
96 OpnAttachNameToNode (
97     ACPI_PARSE_OBJECT      *Op);
98
99 /*****
100 *
101 * FUNCTION:      OpnDoMutex
102 *
103 * PARAMETERS:   Op          - The parent parse node
104 *
105 * RETURN:       None
106 *
107 * DESCRIPTION:  Construct the operands for the MUTEX ASL keyword.
108 *
109 * *****/
110
111 static void
112 OpnDoMutex (
113     ACPI_PARSE_OBJECT      *Op)
114 {
115     ACPI_PARSE_OBJECT      *Next;
116
117     Next = Op->Asl.Child;
118     Next = Next->Asl.Next;
119
120     if (Next->Asl.Value.Integer > 15)
121     {
122         AslError (ASL_ERROR, ASL_MSG_SYNC_LEVEL, Next, NULL);
123     }
124 }
```

```

125     }
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;
791
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 }
801
802 /* Remaining nodes are handled via the tree walk */
803 }
804
805 /*****
806 *
807 * FUNCTION:      OpnDoLoadTable
808 *
809 * PARAMETERS:   Op          - The parent parse node
810 *
811 * RETURN:       None
812 *
813 * DESCRIPTION:  Construct the AML operands for the LOADTABLE ASL keyword.
814 *
815 * *****/
816
817 static void
818 OpnDoLoadTable (
819     ACPI_PARSE_OBJECT *Op)
820 {
821     ACPI_PARSE_OBJECT *Next;
822
823     /* Opcode is parent node */
824     /* First child is the table signature */
825
826     Next = Op->Asl.Child;
827
828     /* Second child is the OEM ID */
829
830     Next = Next->Asl.Next;
831
832     /* Third child is the OEM table ID */
833
834     Next = Next->Asl.Next;
835
836     /* Fourth child is the RootPath string */
837
838     Next = Next->Asl.Next;
839     if (Next->Asl.ParseOpcode == PARSEOP_ZERO)
840     {
841         Next->Asl.ParseOpcode = PARSEOP_STRING_LITERAL;
842         Next->Asl.Value.String = "\\\";
843         Next->Asl.AmlLength = 2;
844         OpcGenerateAmlOpcode (Next);
845     }
846
847 #ifdef ASL_FUTURE_IMPLEMENTATION

```

```

851     /* TBD: NOT IMPLEMENTED */
852     /* Fifth child is the [optional] ParameterPathString */
853     /* Sixth child is the [optional] ParameterData */
854
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     }
862
863     Next = Next->Asl.Next;
864     if (Next->Asl.ParseOpcode == DEFAULT_ARG)
865     {
866         Next->Asl.AmlLength = 1;
867         Next->Asl.ParseOpcode = ZERO;
868         OpcGenerateAmlOpcode (Next);
869     }
870 }
871 #endif
872 }
873
874 /*****
875 *
876 * FUNCTION:      OpnDoDefinitionBlock
877 *
878 * PARAMETERS:   Op          - The parent parse node
879 *
880 * RETURN:       None
881 *
882 * DESCRIPTION:  Construct the AML operands for the DEFINITIONBLOCK ASL keyword
883 *
884 * *****/
885
886 static void
887 OpnDoDefinitionBlock (
888     ACPI_PARSE_OBJECT *Op)
889 {
890     ACPI_PARSE_OBJECT *Child;
891     ACPI_SIZE Length;
892     UINT32 i;
893     char *Filename;
894
895     /*
896     * These nodes get stuffed into the table header. They are special
897     * cased when the table is written to the output file.
898     *
899     * Mark all of these nodes as non-usable so they won't get output
900     * as AML opcodes!
901     */
902
903     /* Get AML filename. Use it if non-null */
904
905     Child = Op->Asl.Child;
906     if (Child->Asl.Value.Buffer &&
907         *Child->Asl.Value.Buffer &&
908         (Gbl_UseDefaultAmlFilename))
909     {
910         /*
911         * We will use the AML filename that is embedded in the source file
912         * for the output filename.
913         */
914         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;

```

```

1050     if (Op->Asl.ParseOpcode == PARSEOP_EXTERNAL)
1051     {
1052         Child = UtGetArg (Op, 0);
1053     }
1054     else switch (Op->Asl.AmlOpcode)
1055     {
1056     case AML_DATA_REGION_OP:
1057     case AML_DEVICE_OP:
1058     case AML_EVENT_OP:
1059     case AML_METHOD_OP:
1060     case AML_MUTEX_OP:
1061     case AML_REGION_OP:
1062     case AML_POWER_RES_OP:
1063     case AML_PROCESSOR_OP:
1064     case AML_THERMAL_ZONE_OP:
1065     case AML_NAME_OP:
1066     case AML_SCOPE_OP:
1068         Child = UtGetArg (Op, 0);
1069         break;
1071     case AML_ALIAS_OP:
1073         Child = UtGetArg (Op, 1);
1074         break;
1076     case AML_CREATE_BIT_FIELD_OP:
1077     case AML_CREATE_BYTE_FIELD_OP:
1078     case AML_CREATE_WORD_FIELD_OP:
1079     case AML_CREATE_DWORD_FIELD_OP:
1080     case AML_CREATE_QWORD_FIELD_OP:
1082         Child = UtGetArg (Op, 2);
1083         break;
1085     case AML_CREATE_FIELD_OP:
1087         Child = UtGetArg (Op, 3);
1088         break;
1090     case AML_BANK_FIELD_OP:
1091     case AML_INDEX_FIELD_OP:
1092     case AML_FIELD_OP:
1094         return;
1096     default:
1098         return;
1099     }
1101     if (Child)
1102     {
1103         UtAttachNamepathToOwner (Op, Child);
1104     }
1105 }

```

```

1108 /*****
1109  *
1110  * FUNCTION:      OpnGenerateAmlOperands
1111  *
1112  * PARAMETERS:   Op          - The parent parse node
1113  *
1114  * RETURN:       None

```

```

1115  *
1116  * DESCRIPTION:  Prepare nodes to be output as AML data and operands. The more
1117  *              complex AML opcodes require processing of the child nodes
1118  *              (arguments/operands).
1119  *
1120  *****/
1122 void
1123 OpnGenerateAmlOperands (
1124     ACPI_PARSE_OBJECT *Op)
1125 {
1128     if (Op->Asl.AmlOpcode == AML_RAW_DATA_BYTE)
1129     {
1130         return;
1131     }
1133     switch (Op->Asl.ParseOpcode)
1134     {
1135     case PARSEOP_DEFINITIONBLOCK:
1137         OpnDoDefinitionBlock (Op);
1138         break;
1140     case PARSEOP_METHOD:
1142         OpnDoMethod (Op);
1143         break;
1145     case PARSEOP_MUTEX:
1147         OpnDoMutex (Op);
1148         break;
1150     case PARSEOP_FIELD:
1152         OpnDoField (Op);
1153         break;
1155     case PARSEOP_INDEXFIELD:
1157         OpnDoIndexField (Op);
1158         break;
1160     case PARSEOP_BANKFIELD:
1162         OpnDoBankField (Op);
1163         break;
1165     case PARSEOP_BUFFER:
1167         OpnDoBuffer (Op);
1168         break;
1170     case PARSEOP_LOADTABLE:
1172         OpnDoLoadTable (Op);
1173         break;
1175     case PARSEOP_OPERATIONREGION:
1177         OpnDoRegion (Op);
1178         break;
1180     case PARSEOP_RESOURCETEMPLATE:

```

```
1182         RsDoResourceTemplate (Op);
1183         break;
1184
1185     case PARSEOP_NAMESEG:
1186     case PARSEOP_NAMESTRING:
1187     case PARSEOP_METHODCALL:
1188     case PARSEOP_STRING_LITERAL:
1189
1190         break;
1191
1192     default:
1193
1194         break;
1195     }
1196
1197     /* TBD: move */
1198
1199     OpnAttachNameToNode (Op);
1200 }
```



new/usr/src/common/acpica/compiler/aslopt.c

1

```
*****
26782 Fri Jan 17 07:29:37 2014
new/usr/src/common/acpica/compiler/aslopt.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: aslopt- Compiler optimizations
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "aslcompiler.h"
45 #include "aslcompiler.y.h"
46
47 #include "acparser.h"
48 #include "amlcode.h"
49 #include "acnamesp.h"
50
51 #define _COMPONENT          ACPI_COMPILER
52 #define ACPI_MODULE_NAME    ("aslopt")
53
54 static UINT32 OptTotal = 0;
```

new/usr/src/common/acpica/compiler/aslopt.c

2

```
59 /* Local prototypes */
60
61 static ACPI_STATUS
62 OptSearchToRoot (
63     ACPI_PARSE_OBJECT *Op,
64     ACPI_WALK_STATE *WalkState,
65     ACPI_NAMESPACE_NODE *CurrentNode,
66     ACPI_NAMESPACE_NODE *TargetNode,
67     ACPI_BUFFER *TargetPath,
68     char **NewPath);
69
70 static ACPI_STATUS
71 OptBuildShortestPath (
72     ACPI_PARSE_OBJECT *Op,
73     ACPI_WALK_STATE *WalkState,
74     ACPI_NAMESPACE_NODE *CurrentNode,
75     ACPI_NAMESPACE_NODE *TargetNode,
76     ACPI_BUFFER *CurrentPath,
77     ACPI_BUFFER *TargetPath,
78     ACPI_SIZE AmlNameStringLength,
79     UINT8 IsDeclaration,
80     char **ReturnNewPath);
81
82 static ACPI_STATUS
83 OptOptimizeNameDeclaration (
84     ACPI_PARSE_OBJECT *Op,
85     ACPI_WALK_STATE *WalkState,
86     ACPI_NAMESPACE_NODE *CurrentNode,
87     ACPI_NAMESPACE_NODE *TargetNode,
88     char *AmlNameString,
89     char **NewPath);
90
91 /*****
92 *
93 * FUNCTION:      OptSearchToRoot
94 *
95 * PARAMETERS:    Op                - Current parser op
96 *                WalkState         - Current state
97 *                CurrentNode        - Where we are in the namespace
98 *                TargetNode         - Node to which we are referring
99 *                TargetPath         - External full path to the target node
100 *                NewPath            - Where the optimized path is returned
101 *
102 * RETURN:        Status
103 *
104 * DESCRIPTION:   Attempt to optimize a reference to a single 4-character ACPI
105 *                name utilizing the search-to-root name resolution algorithm
106 *                that is used by AML interpreters.
107 *
108 *
109 *****/
110
111 static ACPI_STATUS
112 OptSearchToRoot (
113     ACPI_PARSE_OBJECT *Op,
114     ACPI_WALK_STATE *WalkState,
115     ACPI_NAMESPACE_NODE *CurrentNode,
116     ACPI_NAMESPACE_NODE *TargetNode,
117     ACPI_BUFFER *TargetPath,
118     char **NewPath)
119 {
120     ACPI_NAMESPACE_NODE *Node;
121     ScopeInfo;
122     Status;
123     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 */
264
265         break;
266     }
267 }
268
269 ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, " COMMON: %u",
270     NumCommonSegments));
271
272 /* There must be at least 1 common NameSeg in order to optimize */
273
274 if (NumCommonSegments == 0)
275 {
276     return (AE_NOT_FOUND);
277 }
278
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 }
292
293 /* Determine how many prefix Carats are required */
294
295 NumCarats = (CurrentPath->Length / ACPI_PATH_SEGMENT_LENGTH) -
296     NumCommonSegments;
297
298 /*
299  * Construct a new target string
300  */
301 NewPathExternal = ACPI_ALLOCATE_ZEROED (
302     TargetPath->Length + NumCarats + 1);
303
304 /* Insert the Carats into the Target string */
305
306 for (i = 0; i < NumCarats; i++)
307 {
308     NewPathExternal[i] = AML_PARENT_PREFIX;
309 }
310
311 /*
312  * Copy only the necessary (optimal) segments from the original
313  * target string
314  */
315 Index = (NumCommonSegments * ACPI_PATH_SEGMENT_LENGTH) + 1;
316
317 /* Special handling for exact subpath in a name declaration */
318
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;
327
328     /* Special handling for Scope() operator */
329
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 }
337
338 /* Make sure we haven't gone off the end of the target path */
339
340 if (Index > TargetPath->Length)
341 {
342     Index = TargetPath->Length;
343 }
344
345 ACPI_STRCPY (&NewPathExternal[Index], &((char *) TargetPath->Pointer)[Index]);
346 ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, " %-24s", NewPathExternal));
347
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 }
361
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 }
370
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 */
382
383     if (Node == TargetNode)
384     {
385         /* The lookup matched the node, accept this optimization */
386
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 */
394
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 */
404
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 }
409
410 ACPI_FREE (NewPathExternal);
411 return (Status);
412 }
413
414 /*****
415 *
416 * FUNCTION:      OptOptimizeNameDeclaration
417 *
418 * PARAMETERS:    Op                - Current parser op
419 *                WalkState         - Current state
420 *                CurrentNode       - Where we are in the namespace
421 *                AmlNameString     - Unoptimized namepath
422 *                NewPath           - Where the optimized path is returned
423 *
424 * RETURN:        Status. AE_OK If path is optimized
425 *
426 * DESCRIPTION:   Perform a simple optimization of removing an extraneous
427 *                backslash prefix if we are already at the root scope.
428 *
429 * *****/
430
431 static ACPI_STATUS
432 OptOptimizeNameDeclaration (
433     ACPI_PARSE_OBJECT *Op,
434     ACPI_WALK_STATE *WalkState,
435     ACPI_NAMESPACE_NODE *CurrentNode,
436     ACPI_NAMESPACE_NODE *TargetNode,
437     char *AmlNameString,
438     char **NewPath)
439 {
440     ACPI_STATUS Status;
441     char *NewPathExternal;
442     ACPI_NAMESPACE_NODE *Node;
443
444     ACPI_FUNCTION_TRACE (OptOptimizeNameDeclaration);
445
446     if (((CurrentNode == AcpiGbl_RootNode) ||
447         (Op->Common.Parent->Asl.ParseOpcode == PARSEOP_DEFINITIONBLOCK)) &&
448         (ACPI_IS_ROOT_PREFIX (AmlNameString[0])))
449     {
450         /*
451          * 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];
458
459     /* Debug output */
460
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     }
469
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 */
483
484         if (Node == TargetNode)
485         {
486             /* The lookup matched the node, accept this optimization */
487
488             AslError (ASL_OPTIMIZATION, ASL_MSG_NAME_OPTIMIZATION,
489                 Op, NewPathExternal);
490
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 */
497
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 */
508
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     }
514
515     ACPI_FREE (NewPathExternal);
516     return (Status);
517 }
518
519 /* Could not optimize */

```

```

521     return (AE_NOT_FOUND);
522 }

525 /*****
526  * FUNCTION:      OptOptimizeNamePath
527  *
528  * PARAMETERS:    Op          - Current parser op
529  *                Flags       - Opcode info flags
530  *                WalkState    - Current state
531  *                AmlNameString - Unoptimized namepath
532  *                TargetNode    - Node to which AmlNameString refers
533  *
534  * RETURN:        None. If path is optimized, the Op is updated with new path
535  *
536  * DESCRIPTION:   Optimize a Named Declaration or Reference to the minimal length.
537  *                Must take into account both the current location in the
538  *                namespace and the actual reference path.
539  *
540  *****/
541 void
542 OptOptimizeNamePath (
543     ACPI_PARSE_OBJECT *Op,
544     UINT32 Flags,
545     ACPI_WALK_STATE *WalkState,
546     char *AmlNameString,
547     ACPI_NAMESPACE_NODE *TargetNode)
548 {
549     ACPI_STATUS Status;
550     TargetPath;
551     CurrentPath;
552     ACPI_BUFFER AmlNameStringLength;
553     *CurrentNode;
554     char *ExternalNameString;
555     char *NewPath = NULL;
556     ACPI_SIZE HowMuchShorter;
557     *NextOp;
558
559     ACPI_FUNCTION_TRACE (OptOptimizeNamePath);
560
561     /* This is an optional optimization */
562
563     if (!Gbl_ReferenceOptimizationFlag)
564     {
565         return_VOID;
566     }
567
568     /* Various required items */
569
570     if (!TargetNode || !WalkState || !AmlNameString || !Op->Common.Parent)
571     {
572         return_VOID;
573     }
574
575     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS,
576         "PATH OPTIMIZE: Line %5d ParentOp [%12.12s] ThisOp [%12.12s] ",
577         Op->Asl.LogicalLineNumber,
578         AcpiPsGetOpcodeName (Op->Common.Parent->Common.AmlOpcode),
579         AcpiPsGetOpcodeName (Op->Common.AmlOpcode)));
580
581     if (!(Flags & (AML_NAMED | AML_CREATE)))
582     {

```

```

587         if (Op->Asl.CompileFlags & NODE_IS_NAME_DECLARATION)
588         {
589             /* We don't want to fuss with actual name declaration nodes here */
590
591             ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS,
592                 "***** NAME DECLARATION\n"));
593             return_VOID;
594         }
595     }
596
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     }
608
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     }
619
620     if (Flags & (AML_NAMED | AML_CREATE))
621     {
622         /* This is the declaration of a new name */
623
624         ACPI_DEBUG_PRINT_RAW ((ACPI_DB_OPTIMIZATIONS, "NAME\n"));
625
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 */
649
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 optimization 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 }
```

new/usr/src/common/acpica/compiler/asloptions.c

1

```
*****
19027 Fri Jan 17 07:29:37 2014
new/usr/src/common/acpica/compiler/asloptions.c
update to acpica-unix2-20140114
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 - 2014, 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 MERCHANTABILITY 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);
```

new/usr/src/common/acpica/compiler/asloptions.c

2

```
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_Optind]);
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_ASF;
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':
655         Gbl_VerboseTemplates = TRUE;
656         break;
659     case 'w':
660         /* Get the required argument */
661
662         if (AcpiGetoptArgument (argc, argv))
663         {
664             return (-1);
665         }
666
667         Status = AslDisableException (AcpiGbl_Optarg);
668         if (ACPI_FAILURE (Status))
669         {
670             return (-1);
671         }
672         break;
673
675     default:
676         printf ("Unknown option: -v%s\n", AcpiGbl_Optarg);
677         return (-1);
678     }
679     break;
680
682     case 'w': /* Set warning levels */
683
684         switch (AcpiGbl_Optarg[0])
685         {
686             case '1':
687
688                 Gbl_WarningLevel = ASL_WARNING;
689                 break;
691             case '2':
692
693                 Gbl_WarningLevel = ASL_WARNING2;
694                 break;
696             case '3':
697
698                 Gbl_WarningLevel = ASL_WARNING3;
699                 break;
701             case 'e':
702
703                 Gbl_WarningsAsErrors = TRUE;
704                 break;
706             default:
707
708                 printf ("Unknown option: -w%s\n", AcpiGbl_Optarg);
709                 return (-1);
710             }
711             break;
713     case 'x': /* Set debug print output level */
714
715         AcpiDbgLevel = strtoul (AcpiGbl_Optarg, NULL, 16);
716         break;
718     case 'z':

```

```

720     Gbl_UseOriginalCompilerId = TRUE;
721     break;
723     default:
724
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;
751
752     *OutBuffer = 0;
753
754     Token = strtok (InBuffer, ASL_TOKEN_SEPARATORS);
755     while (Token)
756     {
757         strcat (OutBuffer, Token);
758         Token = strtok (NULL, ASL_TOKEN_SEPARATORS);
759     }
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 }
794
795 /* Must save the current GetOpt globals */
796
797 Opterr = AcpiGbl_Opterr;
798 Optind = AcpiGbl_Optind;
799
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 */
807
808     AslMergeOptionTokens (StringBuffer, StringBuffer2);
809
810     /* Process the option */
811
812     AcpiGbl_Opterr = 0;
813     AcpiGbl_Optind = 0;
814
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 }
823
824 /* Restore the GetOpt globals */
825
826 AcpiGbl_Opterr = Opterr;
827 AcpiGbl_Optind = Optind;
828
829 fclose (ResponseFile);
830 return (OptStatus);
831 }
```

new/usr/src/common/acpica/compiler/aslpredef.c

1

```
*****
23978 Fri Jan 17 07:29:38 2014
new/usr/src/common/acpica/compiler/aslpredef.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslpredef - support for ACPI predefined names
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 ACPI_CREATE_PREDEFINED_TABLE
45 #define ACPI_CREATE_RESOURCE_TABLE

47 #include "aslcompiler.h"
48 #include "aslcompiler.y.h"
49 #include "acpredef.h"
50 #include "acnamesp.h"

53 #define _COMPONENT          ACPI_COMPILER
54     ACPI_MODULE_NAME    ("aslpredef")

57 /* Local prototypes */

59 static void
```

new/usr/src/common/acpica/compiler/aslpredef.c

2

```
60 ApCheckForUnexpectedReturnValue (
61     ACPI_PARSE_OBJECT *Op,
62     ASL_METHOD_INFO *MethodInfo);

64 static UINT32
65 ApCheckForSpecialName (
66     ACPI_PARSE_OBJECT *Op,
67     char *Name);

70 /*****
71 *
72 * FUNCTION:      ApCheckForPredefinedMethod
73 *
74 * PARAMETERS:    Op          - A parse node of type "METHOD".
75 *                MethodInfo   - Saved info about this method
76 *
77 * RETURN:        None
78 *
79 * DESCRIPTION:   If method is a predefined name, check that the number of
80 *                arguments and the return type (returns a value or not)
81 *                is correct.
82 *
83 *****/

85 BOOLEAN
86 ApCheckForPredefinedMethod (
87     ACPI_PARSE_OBJECT *Op,
88     ASL_METHOD_INFO *MethodInfo)
89 {
90     UINT32 Index;
91     UINT32 RequiredArgCount;
92     const ACPI_PREDEFINED_INFO *ThisName;

95     /* Check for a match against the predefined name list */

97     Index = ApCheckForPredefinedName (Op, Op->Asl.NameSeg);

99     switch (Index)
100     {
101     case ACPI_NOT_RESERVED_NAME: /* No underscore or _Txx or _xxx name no
102     case ACPI_PREDEFINED_NAME:  /* Resource Name or reserved scope name
103     case ACPI_COMPILER_RESERVED_NAME: /* A _Txx that was not emitted by compil

105         /* Just return, nothing to do */
106         return (FALSE);

109     case ACPI_EVENT_RESERVED_NAME: /* _Lxx/_Exx/_Wxx/_Qxx methods */

111         Gbl_ReservedMethods++;

113         /* NumArguments must be zero for all _Lxx/_Exx/_Wxx/_Qxx methods */

115         if (MethodInfo->NumArguments != 0)
116         {
117             sprintf (MsgBuffer, "%s requires %u", Op->Asl.ExternalName, 0);

119             AslError (ASL_WARNING, ASL_MSG_RESERVED_ARG_COUNT_HI, Op,
120                     MsgBuffer);
121         }
122         break;

125     default:
```

```

126  /*
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 *****/
201
202 static void
203 ApCheckForUnexpectedReturnValue (
204     ACPI_PARSE_OBJECT *Op,
205     ASL_METHOD_INFO *MethodInfo)
206 {
207     ACPI_PARSE_OBJECT *ReturnValueOp;
208
209     /* Ignore Return() and Return(Zero) (they are the same) */
210
211     ReturnValueOp = Op->Asl.Child;
212     if (ReturnValueOp->Asl.ParseOpcode == PARSEOP_ZERO)
213     {
214         return;
215     }
216
217     /* We have a valid return value, but the reserved name did not expect it */
218
219     AslError (ASL_WARNING, ASL_MSG_RESERVED_NO_RETURN_VAL,
220         Op, MethodInfo->Op->Asl.ExternalName);
221 }
222
223 /*****
224 *
225 * FUNCTION:      ApCheckPredefinedReturnValue
226 *
227 * PARAMETERS:   Op                - A parse node of type "RETURN".
228 *               MethodInfo         - Saved info about this method
229 *
230 * RETURN:       None
231 *
232 * DESCRIPTION:  If method is a predefined name, attempt to validate the return
233 *               value. Only "static" types can be validated - a simple return
234 *               of an integer/string/buffer/package or a named reference to
235 *               a static object. Values such as a Localx or Argx or a control
236 *               method invocation are not checked. Issue a warning if there is
237 *               a valid return value, but the reserved method defines no
238 *               return value.
239 *
240 * *****/
241
242 void
243 ApCheckPredefinedReturnValue (
244     ACPI_PARSE_OBJECT *Op,
245     ASL_METHOD_INFO *MethodInfo)
246 {
247     UINT32 Index;
248     ACPI_PARSE_OBJECT *ReturnValueOp;
249     const ACPI_PREDEFINED_INFO *ThisName;
250
251     /* Check parent method for a match against the predefined name list */
252
253     Index = ApCheckForPredefinedName (MethodInfo->Op,
254         MethodInfo->Op->Asl.NameSeg);

```



```

259     switch (Index)
260     {
261     case ACPI_EVENT_RESERVED_NAME:      /* _Lxx/_Exx/_Wxx/_Qxx methods */
262
263         /* No return value expected, warn if there is one */
264
265         ApCheckForUnexpectedReturnValue (Op, MethodInfo);
266         return;
267
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
271
272         /* Just return, nothing to do */
273         return;
274
275     default: /* A standard predefined ACPI name */
276
277         ThisName = &AcpiGbl_PredefinedMethods[Index];
278         if (!ThisName->Info.ExpectedBtypes)
279         {
280             /* No return value expected, warn if there is one */
281
282             ApCheckForUnexpectedReturnValue (Op, MethodInfo);
283             return;
284         }
285
286         /* Get the object returned, it is the next argument */
287
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:
298
299             /* Static data return object - check against expected type */
300
301             ApCheckObjectType (ThisName->Info.Name, ReturnValueOp,
302                               ThisName->Info.ExpectedBtypes, ACPI_NOT_PACKAGE_ELEMENT);
303
304             /* For packages, check the individual package elements */
305
306             if (ReturnValueOp->Asl.ParseOpcode == PARSEOP_PACKAGE)
307             {
308                 ApCheckPackage (ReturnValueOp, ThisName);
309             }
310             break;
311
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 implementation of the name), and that
339 *               the type of the object is one of the expected types for the
340 *               predefined name.
341 *
342 *****/
343
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;
352
353     /*
354      * Check for a real predefined name -- not a resource descriptor name
355      * or a predefined scope name
356      */
357     Index = ApCheckForPredefinedName (Op, Name);
358
359     switch (Index)
360     {
361     case ACPI_NOT_RESERVED_NAME:        /* No underscore or _Txx or _xxx name no
362     case ACPI_PREDEFINED_NAME:          /* Resource Name or reserved scope name
363     case ACPI_COMPILER_RESERVED_NAME:   /* A _Txx that was not emitted by compil
364
365         /* Nothing to do */
366         return;
367
368     case ACPI_EVENT_RESERVED_NAME:      /* _Lxx/_Exx/_Wxx/_Qxx methods */
369
370         /*
371          * These names must be control methods, by definition in ACPI spec.
372          * Also because they are defined to return no value. None of them
373          * require any arguments.
374          */
375         AslError (ASL_ERROR, ASL_MSG_RESERVED_METHOD, Op,
376                 "with zero arguments");
377         return;
378
379     default:
380
381         break;
382     }
383
384     /* A standard predefined ACPI name */
385
386     /*
387      * If this predefined name requires input arguments, then
388      * it must be implemented as a control method
389      */

```

```

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  }
398
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  }
411
412  /* Typecheck the actual object, it is the next argument */
413
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);
417
418  /* For packages, check the individual package elements */
419
420  if (ObjectOp->Asl.ParseOpcode == PARSEOP_PACKAGE)
421  {
422      ApCheckPackage (ObjectOp, ThisName);
423  }
424 }
425
426 /*****
427  *
428  * FUNCTION:      ApCheckForPredefinedName
429  *
430  * PARAMETERS:    Op          - A parse node
431  *                Name        - NameSeg to check
432  *
433  * RETURN:        None
434  *
435  * DESCRIPTION:   Check a NameSeg against the reserved list.
436  *
437  *****/
438
439
440  UINT32
441  ApCheckForPredefinedName (
442      ACPI_PARSE_OBJECT *Op,
443      char *Name)
444  {
445      UINT32 i;
446      const ACPI_PREDEFINED_INFO *ThisName;
447
448      if (Name[0] == 0)
449      {
450          AcpiOsPrintf ("Found a null name, external = %s\n",
451                      Op->Asl.ExternalName);
452      }
453
454      /* All reserved names are prefixed with a single underscore */

```

```

457  if (Name[0] != '_')
458  {
459      return (ACPI_NOT_RESERVED_NAME);
460  }
461
462  /* Check for a standard predefined method name */
463
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      }
472
473      ThisName++; /* Does not account for extra package data, but is OK */
474  }
475
476  /* Check for resource names and predefined scope names */
477
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      }
485
486      ThisName++;
487  }
488
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      }
496
497      ThisName++;
498  }
499
500  /* Check for _Lxx/_Exx/_Wxx/_Qxx/_T_x. Warning if unknown predefined name */
501
502  return (ApCheckForSpecialName (Op, Name));
503 }
504
505 /*****
506  *
507  * FUNCTION:      ApCheckForSpecialName
508  *
509  * PARAMETERS:    Op          - A parse node
510  *                Name        - NameSeg to check
511  *
512  * RETURN:        None
513  *
514  * DESCRIPTION:   Check for the "special" predefined names -
515  *                _Lxx, _Exx, _Qxx, _Wxx, and _T_x
516  *
517  *****/
518
519
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         * disassembly, 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.ExternalName);
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 * FUNCTION:    ApCheckObjectType
586 */

```

```

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     if (!Op)
614     {
615         return (AE_TYPE);
616     }
617
618     /* Map the parse opcode to a bitmapped return type (RTYPE) */
619
620     switch (Op->Asl.ParseOpcode)
621     {
622     case PARSEOP_ZERO:
623     case PARSEOP_ONE:
624     case PARSEOP_ONES:
625     case PARSEOP_INTEGER:
626
627         ReturnBtype = ACPI_RTYPE_INTEGER;
628         TypeName = "Integer";
629         break;
630
631     case PARSEOP_STRING_LITERAL:
632
633         ReturnBtype = ACPI_RTYPE_STRING;
634         TypeName = "String";
635         break;
636
637     case PARSEOP_BUFFER:
638
639         ReturnBtype = ACPI_RTYPE_BUFFER;
640         TypeName = "Buffer";
641         break;
642
643     case PARSEOP_PACKAGE:
644     case PARSEOP_VAR_PACKAGE:
645
646         ReturnBtype = ACPI_RTYPE_PACKAGE;
647         TypeName = "Package";
648         break;
649
650     case PARSEOP_NAMESEG:
651     case PARSEOP_NAMESTRING:
652     /*

```

```

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     }
667
668     ReturnBtype = ACPI_RTYPE_REFERENCE;
669     TypeName = "Reference";
670     break;
671
672     default:
673
674         /* Not one of the supported object types */
675
676         TypeName = UtGetOpName (Op->Asl.ParseOpcode);
677         goto TypeErrorExit;
678     }
679
680     /* Exit if the object is one of the expected types */
681
682     if (ReturnBtype & ExpectedBtypes)
683     {
684         return (AE_OK);
685     }
686
687     TypeErrorExit:
688
689     /* Format the expected types and emit an error message */
690
691     AcpiUtGetExpectedReturnTypes (StringBuffer, ExpectedBtypes);
692
693     if (PackageIndex == ACPI_NOT_PACKAGE_ELEMENT)
694     {
695         sprintf (MsgBuffer, "%4.4s: found %s, %s required",
696             PredefinedName, TypeName, StringBuffer);
697     }
698     else
699     {
700         sprintf (MsgBuffer, "%4.4s: found %s at index %u, %s required",
701             PredefinedName, TypeName, PackageIndex, StringBuffer);
702     }
703
704     AslError (ASL_ERROR, ASL_MSG_RESERVED_OPERAND_TYPE, Op, MsgBuffer);
705     return (AE_TYPE);
706 }
707
708 /*****
709 *
710 * FUNCTION:      ApDisplayReservedNames
711 *
712 * PARAMETERS:    None
713 *
714 * RETURN:        None
715 *
716 * DESCRIPTION:   Dump information about the ACPI predefined names and predefined
717 *                resource descriptor names.
718 */

```

```

720     *
721     *****/
722
723     void
724     ApDisplayReservedNames (
725         void)
726     {
727         const ACPI_PREDEFINED_INFO *ThisName;
728         UINT32 Count;
729         UINT32 NumTypes;
730
731         /*
732          * Predefined names/methods
733          */
734         printf ("\nPredefined Name Information\n\n");
735
736         Count = 0;
737         ThisName = AcpiGbl_PredefinedMethods;
738         while (ThisName->Info.Name[0])
739         {
740             AcpiUtDisplayPredefinedMethod (MsgBuffer, ThisName, FALSE);
741             Count++;
742             ThisName = AcpiUtGetNextPredefinedMethod (ThisName);
743         }
744
745         printf ("%u Predefined Names are recognized\n", Count);
746
747         /*
748          * Resource Descriptor names
749          */
750         printf ("\nPredefined Names for Resource Descriptor Fields\n\n");
751
752         Count = 0;
753         ThisName = AcpiGbl_ResourceNames;
754         while (ThisName->Info.Name[0])
755         {
756             NumTypes = AcpiUtGetResourceBitWidth (MsgBuffer,
757                 ThisName->Info.ArgumentList);
758
759             printf ("%4.4s Field is %s bits wide\n",
760                 ThisName->Info.Name, MsgBuffer,
761                 (NumTypes > 1) ? " (depending on descriptor type)" : "");
762
763             Count++;
764             ThisName++;
765         }
766
767         printf ("%u Resource Descriptor Field Names are recognized\n", Count);
768
769         /*
770          * Predefined scope names
771          */
772         printf ("\nPredefined Scope/Device Names (automatically created at root)\n\n");
773
774         ThisName = AcpiGbl_ScopeNames;
775         while (ThisName->Info.Name[0])
776         {
777             printf ("%4.4s Scope/Device\n", ThisName->Info.Name);
778             ThisName++;
779         }
780     }
781 }

```

new/usr/src/common/acpica/compiler/aslprepkg.c

1

```
*****
21555 Fri Jan 17 07:29:38 2014
new/usr/src/common/acpica/compiler/aslprepkg.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: aslprepkg - support for ACPI predefined name package objects
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"
46 #include "acpredef.h"

49 #define _COMPONENT          ACPI_COMPILER
50     ACPI_MODULE_NAME    ("aslprepkg")

53 /* Local prototypes */

55 static void
56 ApCheckPackageElements (
57     const char                *PredefinedName,
58     ACPI_PARSE_OBJECT         *Op,
59     UINT8                     Type1,
60     UINT32                     Count1,
```

new/usr/src/common/acpica/compiler/aslprepkg.c

2

```
61     UINT8                     Type2,
62     UINT32                     Count2);

64 static void
65 ApCheckPackageList (
66     const char                *PredefinedName,
67     ACPI_PARSE_OBJECT         *ParentOp,
68     const ACPI_PREDEFINED_INFO *Package,
69     UINT32                     StartIndex,
70     UINT32                     Count);

72 static void
73 ApPackageTooSmall (
74     const char                *PredefinedName,
75     ACPI_PARSE_OBJECT         *Op,
76     UINT32                     Count,
77     UINT32                     ExpectedCount);

79 static void
80 ApZeroLengthPackage (
81     const char                *PredefinedName,
82     ACPI_PARSE_OBJECT         *Op);

84 static void
85 ApPackageTooLarge (
86     const char                *PredefinedName,
87     ACPI_PARSE_OBJECT         *Op,
88     UINT32                     Count,
89     UINT32                     ExpectedCount);

92 /*****
93 *
94 * FUNCTION:      ApCheckPackage
95 *
96 * PARAMETERS:   ParentOp      - Parser op for the package
97 *               Predefined    - Pointer to package-specific info for
98 *                               the method
99 *
100 * RETURN:       None
101 *
102 * DESCRIPTION:  Top-level validation for predefined name return package
103 *               objects.
104 *
105 *****/

107 void
108 ApCheckPackage (
109     ACPI_PARSE_OBJECT         *ParentOp,
110     const ACPI_PREDEFINED_INFO *Predefined)
111 {
112     ACPI_PARSE_OBJECT         *Op;
113     const ACPI_PREDEFINED_INFO *Package;
114     ACPI_STATUS                Status;
115     UINT32                     ExpectedCount;
116     UINT32                     Count;
117     UINT32                     i;

120     /* The package info for this name is in the next table entry */

122     Package = Predefined + 1;

124     /* First child is the package length */

126     Op = ParentOp->Asl.Child;
```

```

127     Count = (UINT32) Op->Asl.Value.Integer;

129     /*
130     * Many of the variable-length top-level packages are allowed to simply
131     * have zero elements. This allows the BIOS to tell the host that even
132     * though the predefined name/method exists, the feature is not supported.
133     * Other package types require one or more elements. In any case, there
134     * is no need to continue validation.
135     */
136     if (!Count)
137     {
138         switch (Package->RetInfo.Type)
139         {
140             case ACPI_PTYPE1_FIXED:
141             case ACPI_PTYPE1_OPTION:
142             case ACPI_PTYPE2_PKG_COUNT:
143             case ACPI_PTYPE2_REV_FIXED:

145                 ApZeroLengthPackage (Predefined->Info.Name, ParentOp);
146                 break;

148             case ACPI_PTYPE1_VAR:
149             case ACPI_PTYPE2:
150             case ACPI_PTYPE2_COUNT:
151             case ACPI_PTYPE2_FIXED:
152             case ACPI_PTYPE2_MIN:
153             case ACPI_PTYPE2_FIX_VAR:
154             default:

156                 break;
157         }

159         return;
160     }

162     /* Get the first element of the package */

164     Op = Op->Asl.Next;

166     /* Decode the package type */

168     switch (Package->RetInfo.Type)
169     {
170     case ACPI_PTYPE1_FIXED:
171         /*
172         * The package count is fixed and there are no sub-packages
173         *
174         * If package is too small, exit.
175         * If package is larger than expected, issue warning but continue
176         */
177         ExpectedCount = Package->RetInfo.Count1 + Package->RetInfo.Count2;
178         if (Count < ExpectedCount)
179         {
180             goto PackageTooSmall;
181         }
182         else if (Count > ExpectedCount)
183         {
184             ApPackageTooLarge (Predefined->Info.Name, ParentOp,
185                               Count, ExpectedCount);
186         }

188         /* Validate all elements of the package */

190         ApCheckPackageElements (Predefined->Info.Name, Op,
191                               Package->RetInfo.ObjectType1, Package->RetInfo.Count1,
192                               Package->RetInfo.ObjectType2, Package->RetInfo.Count2);

```

```

193         break;

195     case ACPI_PTYPE1_VAR:
196         /*
197         * The package count is variable, there are no sub-packages,
198         * and all elements must be of the same type
199         */
200         for (i = 0; i < Count; i++)
201         {
202             ApCheckObjectType (Predefined->Info.Name, Op,
203                               Package->RetInfo.ObjectType1, i);
204             Op = Op->Asl.Next;
205         }
206         break;

208     case ACPI_PTYPE1_OPTION:
209         /*
210         * The package count is variable, there are no sub-packages.
211         * There are a fixed number of required elements, and a variable
212         * number of optional elements.
213         *
214         * Check if package is at least as large as the minimum required
215         */
216         ExpectedCount = Package->RetInfo3.Count;
217         if (Count < ExpectedCount)
218         {
219             goto PackageTooSmall;
220         }

222         /* Variable number of sub-objects */

224         for (i = 0; i < Count; i++)
225         {
226             if (i < Package->RetInfo3.Count)
227             {
228                 /* These are the required package elements (0, 1, or 2) */

230                 ApCheckObjectType (Predefined->Info.Name, Op,
231                                   Package->RetInfo3.ObjectType[i], i);
232             }
233             else
234             {
235                 /* These are the optional package elements */

237                 ApCheckObjectType (Predefined->Info.Name, Op,
238                                   Package->RetInfo3.TailObjectType, i);
239             }
240             Op = Op->Asl.Next;
241         }
242         break;

244     case ACPI_PTYPE2_REV_FIXED:
245         /* First element is the (Integer) revision */

248         ApCheckObjectType (Predefined->Info.Name, Op,
249                           ACPI_RTYPE_INTEGER, 0);

251         Op = Op->Asl.Next;
252         Count--;

254         /* Examine the sub-packages */

256         ApCheckPackageList (Predefined->Info.Name, Op,
257                             Package, 1, Count);
258         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
392 *          ACPI_PTYPE2_MIN
393 *          ACPI_PTYPE2_COUNT
394 *          ACPI_PTYPE2_FIX_VAR
395 *
396 *****/

398 static void
399 ApCheckPackageList (
400     const char          *PredefinedName,
401     ACPI_PARSE_OBJECT   *ParentOp,
402     const ACPI_PREDEFINED_INFO *Package,
403     UINT32              StartIndex,
404     UINT32              ParentCount)
405 {
406     ACPI_PARSE_OBJECT   *SubPackageOp = ParentOp;
407     ACPI_PARSE_OBJECT   *Op;
408     ACPI_STATUS          Status;
409     UINT32              Count;
410     UINT32              ExpectedCount;
411     UINT32              i;
412     UINT32              j;

415     /*
416      * Validate each subpackage in the parent Package
417      *
418      * Note: We ignore NULL package elements on the assumption that
419      * they will be initialized by the BIOS or other ASL code.
420      */
421     for (i = 0; (i < ParentCount) && SubPackageOp; i++)
422     {
423         /* Each object in the list must be of type Package */

425         Status = ApCheckObjectType (PredefinedName, SubPackageOp,
426                                     ACPI_RTYPE_PACKAGE, i + StartIndex);
427         if (ACPI_FAILURE (Status))
428         {
429             goto NextSubpackage;
430         }

432         /* Examine the different types of expected subpackages */

434         Op = SubPackageOp->Asl.Child;

436         /* First child is the package length */

438         Count = (UINT32) Op->Asl.Value.Integer;
439         Op = Op->Asl.Next;

441         /* The subpackage must have at least one element */

443         if (!Count)
444         {
445             ApZeroLengthPackage (PredefinedName, SubPackageOp);
446             goto NextSubpackage;
447         }

449         /*
450          * Decode the package type.
451          * PTYPE2 indicates that a "package of packages" is expected for
452          * this name. The various flavors of PTYPE2 indicate the number
453          * and format of the subpackages.
454          */
455         switch (Package->RetInfo.Type)
456         {

```

```

457     case ACPI_PTYPE2:
458     case ACPI_PTYPE2_PKG_COUNT:
459     case ACPI_PTYPE2_REV_FIXED:

461         /* Each subpackage has a fixed number of elements */

463         ExpectedCount = Package->RetInfo.Count1 + Package->RetInfo.Count2;
464         if (Count < ExpectedCount)
465         {
466             ApPackageTooSmall (PredefinedName, SubPackageOp,
467                                 Count, ExpectedCount);
468             break;
469         }

471         ApCheckPackageElements (PredefinedName, Op,
472                                 Package->RetInfo.ObjectType1, Package->RetInfo.Count1,
473                                 Package->RetInfo.ObjectType2, Package->RetInfo.Count2);
474         break;

476     case ACPI_PTYPE2_FIX_VAR:
477         /*
478          * Each subpackage has a fixed number of elements and an
479          * optional element
480          */
481         ExpectedCount = Package->RetInfo.Count1 + Package->RetInfo.Count2;
482         if (Count < ExpectedCount)
483         {
484             ApPackageTooSmall (PredefinedName, SubPackageOp,
485                                 Count, ExpectedCount);
486             break;
487         }

489         ApCheckPackageElements (PredefinedName, Op,
490                                 Package->RetInfo.ObjectType1, Package->RetInfo.Count1,
491                                 Package->RetInfo.ObjectType2,
492                                 Count - Package->RetInfo.Count1);
493         break;

495     case ACPI_PTYPE2_FIXED:

497         /* Each sub-package has a fixed length */

499         ExpectedCount = Package->RetInfo2.Count;
500         if (Count < ExpectedCount)
501         {
502             ApPackageTooSmall (PredefinedName, SubPackageOp,
503                                 Count, ExpectedCount);
504             break;
505         }

507         /* Check each object/type combination */

509         for (j = 0; j < ExpectedCount; j++)
510         {
511             ApCheckObjectType (PredefinedName, Op,
512                                 Package->RetInfo2.ObjectType[j], j);

514             Op = Op->Asl.Next;
515         }
516         break;

518     case ACPI_PTYPE2_MIN:

520         /* Each sub-package has a variable but minimum length */

522         ExpectedCount = Package->RetInfo.Count1;

```



```

523     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 }

```

```

590 NextSubpackage:
591     SubPackageOp = SubPackageOp->Asl.Next;
592 }
593 }
594
595 /*****
596 *
597 * FUNCTION:      ApPackageTooSmall
598 *
599 * PARAMETERS:    PredefinedName      - Name of the predefined object
600 *                Op                  - Current parser op
601 *                Count                - Actual package element count
602 *                ExpectedCount        - Expected package element count
603 *
604 * RETURN:        None
605 *
606 * DESCRIPTION:   Issue error message for a package that is smaller than
607 *                required.
608 *
609 * *****/
610
611 static void
612 ApPackageTooSmall (
613     const char          *PredefinedName,
614     ACPI_PARSE_OBJECT *Op,
615     UINT32              Count,
616     UINT32              ExpectedCount)
617 {
618     sprintf (MsgBuffer, "%s: length %u, required minimum is %u",
619         PredefinedName, Count, ExpectedCount);
620
621     AslError (ASL_ERROR, ASL_MSG_RESERVED_PACKAGE_LENGTH, Op, MsgBuffer);
622 }
623
624 /*****
625 *
626 * FUNCTION:      ApZeroLengthPackage
627 *
628 * PARAMETERS:    PredefinedName      - Name of the predefined object
629 *                Op                  - Current parser op
630 *
631 * RETURN:        None
632 *
633 * DESCRIPTION:   Issue error message for a zero-length package (a package that
634 *                is required to have a non-zero length). Variable length
635 *                packages seem to be allowed to have zero length, however.
636 *                Even if not allowed, BIOS code does it.
637 *
638 * *****/
639
640 static void
641 ApZeroLengthPackage (
642     const char          *PredefinedName,
643     ACPI_PARSE_OBJECT *Op)
644 {
645     sprintf (MsgBuffer, "%s: length is zero", PredefinedName);
646
647     AslError (ASL_ERROR, ASL_MSG_RESERVED_PACKAGE_LENGTH, Op, MsgBuffer);
648 }

```

```
655 /*****
656 *
657 * FUNCTION:      ApPackageTooLarge
658 *
659 * PARAMETERS:    PredefinedName      - Name of the predefined object
660 *                Op                    - Current parser op
661 *                Count                  - Actual package element count
662 *                ExpectedCount          - Expected package element count
663 *
664 * RETURN:        None
665 *
666 * DESCRIPTION:   Issue a remark for a package that is larger than expected.
667 *
668 *****/
670 static void
671 ApPackageTooLarge (
672     const char      *PredefinedName,
673     ACPI_PARSE_OBJECT *Op,
674     UINT32           Count,
675     UINT32           ExpectedCount)
676 {
678     sprintf (MsgBuffer, "%s: length is %u, only %u required",
679             PredefinedName, Count, ExpectedCount);
681     AslError (ASL_REMARK, ASL_MSG_RESERVED_PACKAGE_LENGTH, Op, MsgBuffer);
682 }
```

new/usr/src/common/acpica/compiler/aslresource.c

1

```
*****
32978 Fri Jan 17 07:29:38 2014
new/usr/src/common/acpica/compiler/aslresource.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslresource - Resource template/descriptor utilities
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"
47 #include "amlcode.h"

50 #define _COMPONENT          ACPI_COMPILER
51 #define ACPI_MODULE_NAME    ("aslresource")

54 /*****
55 *
56 * FUNCTION:      RsSmallAddressCheck
57 *
58 * PARAMETERS:    Minimum          - Address Min value
59 *                Maximum          - Address Max value
60 *****/
```

new/usr/src/common/acpica/compiler/aslresource.c

2

```
60 *          Length          - Address range value
61 *          Alignment       - Address alignment value
62 *          MinOp           - Original Op for Address Min
63 *          MaxOp           - Original Op for Address Max
64 *          LengthOp        - Original Op for address range
65 *          AlignOp         - Original Op for address alignment. If
66 *                          NULL, means "zero value for alignment is
67 *                          OK, and means 64K alignment" (for
68 *                          Memory24 descriptor)
69 *          Op              - Parent Op for entire construct
70 *
71 * RETURN:          None. Adds error messages to error log if necessary
72 *
73 * DESCRIPTION:     Perform common value checks for "small" address descriptors.
74 *                  Currently:
75 *                  Io, Memory24, Memory32
76 *
77 *****/

79 void
80 RsSmallAddressCheck (
81     UINT8          Type,
82     UINT32          Minimum,
83     UINT32          Maximum,
84     UINT32          Length,
85     UINT32          Alignment,
86     ACPI_PARSE_OBJECT *MinOp,
87     ACPI_PARSE_OBJECT *MaxOp,
88     ACPI_PARSE_OBJECT *LengthOp,
89     ACPI_PARSE_OBJECT *AlignOp,
90     ACPI_PARSE_OBJECT *Op)
91 {
92
93     if (Gbl_NoResourceChecking)
94     {
95         return;
96     }
97
98     /*
99     * Check for a so-called "null descriptor". These are descriptors that are
100     * created with most fields set to zero. The intent is that the descriptor
101     * will be updated/completed at runtime via a BufferField.
102     *
103     * If the descriptor does NOT have a resource tag, it cannot be referenced
104     * by a BufferField and we will flag this as an error. Conversely, if
105     * the descriptor has a resource tag, we will assume that a BufferField
106     * will be used to dynamically update it, so no error.
107     *
108     * A possible enhancement to this check would be to verify that in fact
109     * a BufferField is created using the resource tag, and perhaps even
110     * verify that a Store is performed to the BufferField.
111     *
112     * Note: for these descriptors, Alignment is allowed to be zero
113     */
114     if (!Minimum && !Maximum && !Length)
115     {
116         if (!Op->Asl.ExternalName)
117         {
118             /* No resource tag. Descriptor is fixed and is also illegal */
119
120             AslError (ASL_ERROR, ASL_MSG_NULL_DESCRIPTOR, Op, NULL);
121         }
122     }
123
124     return;
```

```

126  /* 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)
391 {
392
393     while (InitializerOp)
394     {
395         if (InitializerOp->Asl.ParseOpcode == PARSEOP_STRING_LITERAL)
396         {
397             return ((UINT16) (strlen (InitializerOp->Asl.Value.String) + 1));
398         }
399         InitializerOp = ASL_GET_PEER_NODE (InitializerOp);
400     }
401
402     return (0);
403 }

```

```

406 /*****
407 *
408 * FUNCTION:      RsAllocateResourceNode
409 *
410 * PARAMETERS:   Size          - Size of node in bytes
411 *
412 * RETURN:       The allocated node - aborts on allocation failure
413 *
414 * DESCRIPTION:  Allocate a resource description node and the resource
415 *               descriptor itself (the nodes are used to link descriptors).
416 *
417 *****/

```

```

419 ASL_RESOURCE_NODE *
420 RsAllocateResourceNode (
421     UINT32          Size)
422 {
423     ASL_RESOURCE_NODE      *Rnode;
424
425     /* Allocate the node */
426
427     Rnode = UtLocalCalloc (sizeof (ASL_RESOURCE_NODE));
428
429     /* Allocate the resource descriptor itself */
430
431     Rnode->Buffer = UtLocalCalloc (Size);
432     Rnode->BufferLength = Size;
433
434     return (Rnode);
435 }

```

```

439 /*****
440 *
441 * FUNCTION:      RsCreateResourceField
442 *
443 * PARAMETERS:   Op          - Resource field node
444 *               Name        - Name of the field (Used only to reference
445 *                           the field in the ASL, not in the AML)
446 *               ByteOffset  - Offset from the field start
447 *               BitOffset   - Additional bit offset
448 *               BitLength   - Number of bits in the field
449 *
450 * RETURN:       None, sets fields within the input node
451 *
452 * DESCRIPTION:  Utility function to generate a named bit field within a
453 *               resource descriptor. Mark a node as 1) a field in a resource
454 *               descriptor, and 2) set the value to be a BIT offset
455 *

```

```

456 *****/

```

```

458 void
459 RsCreateResourceField (
460     ACPI_PARSE_OBJECT      *Op,
461     char                   *Name,
462     UINT32                 ByteOffset,
463     UINT32                 BitOffset,
464     UINT32                 BitLength)
465 {
466     Op->Asl.ExternalName = Name;
467     Op->Asl.CompileFlags |= NODE_IS_RESOURCE_FIELD;
468
469     Op->Asl.Value.Tag.BitOffset = (ByteOffset * 8) + BitOffset;
470     Op->Asl.Value.Tag.BitLength = BitLength;
471 }

```

```

476 /*****
477 *
478 * FUNCTION:      RsSetFlagBits
479 *
480 * PARAMETERS:   *Flags      - Pointer to the flag byte
481 *               Op          - Flag initialization node
482 *               Position     - Bit position within the flag byte
483 *               Default      - Used if the node is DEFAULT.
484 *
485 * RETURN:       Sets bits within the *Flags output byte.
486 *
487 * DESCRIPTION:  Set a bit in a cumulative flags word from an initialization
488 *               node. Will use a default value if the node is DEFAULT, meaning
489 *               that no value was specified in the ASL. Used to merge multiple
490 *               keywords into a single flags byte.
491 *
492 *****/

```

```

494 void
495 RsSetFlagBits (
496     UINT8          *Flags,
497     ACPI_PARSE_OBJECT *Op,
498     UINT8          Position,
499     UINT8          DefaultBit)
500 {
501     if (Op->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
502     {
503         /* Use the default bit */
504
505         *Flags |= (DefaultBit << Position);
506     }
507     else
508     {
509         /* Use the bit specified in the initialization node */
510
511         *Flags |= (((UINT8) Op->Asl.Value.Integer) << Position);
512     }
513 }

```

```

517 void
518 RsSetFlagBits16 (
519     UINT16         *Flags,
520     ACPI_PARSE_OBJECT *Op,
521     UINT8          Position,

```

```

522     UINT8           DefaultBit)
523 {
524
525     if (Op->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
526     {
527         /* Use the default bit */
528
529         *Flags |= (DefaultBit << Position);
530     }
531     else
532     {
533         /* Use the bit specified in the initialization node */
534
535         *Flags |= (((UINT16) Op->Asl.Value.Integer) << Position);
536     }
537 }
538
539 /*****
540 *
541 * FUNCTION:      RsCompleteNodeAndGetNext
542 *
543 * PARAMETERS:   Op          - Resource node to be completed
544 *
545 * RETURN:       The next peer to the input node.
546 *
547 * DESCRIPTION:  Mark the current node completed and return the next peer.
548 *               The node ParseOpcode is set to DEFAULT_ARG, meaning that
549 *               this node is to be ignored from now on.
550 *
551 * *****/
552
553 ACPI_PARSE_OBJECT *
554 RsCompleteNodeAndGetNext (
555     ACPI_PARSE_OBJECT *Op)
556 {
557     /* Mark this node unused */
558
559     Op->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
560
561     /* Move on to the next peer node in the initializer list */
562
563     return (ASL_GET_PEER_NODE (Op));
564 }
565
566 /*****
567 *
568 * FUNCTION:      RsCheckListForDuplicates
569 *
570 * PARAMETERS:   Op          - First op in the initializer list
571 *
572 * RETURN:       None
573 *
574 * DESCRIPTION:  Check an initializer list for duplicate values. Emits an error
575 *               if any duplicates are found.
576 *
577 * *****/
578
579 void
580 RsCheckListForDuplicates (
581     ACPI_PARSE_OBJECT *Op)
582 {
583     ACPI_PARSE_OBJECT *NextValueOp = Op;
584     ACPI_PARSE_OBJECT *NextOp;

```

```

585     Value;
586
587     if (!Op)
588     {
589         return;
590     }
591
592     /* Search list once for each value in the list */
593
594     while (NextValueOp)
595     {
596         Value = (UINT32) NextValueOp->Asl.Value.Integer;
597
598         /* Compare this value to all remaining values in the list */
599
600         NextOp = ASL_GET_PEER_NODE (NextValueOp);
601         while (NextOp)
602         {
603             if (NextOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
604             {
605                 /* Compare values */
606
607                 if (Value == (UINT32) NextOp->Asl.Value.Integer)
608                 {
609                     /* Emit error only once per duplicate node */
610
611                     if (!(NextOp->Asl.CompileFlags & NODE_IS_DUPLICATE))
612                     {
613                         NextOp->Asl.CompileFlags |= NODE_IS_DUPLICATE;
614                         AslError (ASL_ERROR, ASL_MSG_DUPLICATE_ITEM,
615                             NextOp, NULL);
616                     }
617                 }
618             }
619             NextOp = ASL_GET_PEER_NODE (NextOp);
620         }
621         NextValueOp = ASL_GET_PEER_NODE (NextValueOp);
622     }
623 }
624
625 /*****
626 *
627 * FUNCTION:      RsDoOneResourceDescriptor
628 *
629 * PARAMETERS:   DescriptorTypeOp - Parent parse node of the descriptor
630 *               CurrentByteOffset - Offset in the resource descriptor
631 *               buffer.
632 *
633 * RETURN:       A valid resource node for the descriptor
634 *
635 * DESCRIPTION:  Dispatches the processing of one resource descriptor
636 *
637 * *****/
638
639 ASL_RESOURCE_NODE *
640 RsDoOneResourceDescriptor (
641     ACPI_PARSE_OBJECT *DescriptorTypeOp,
642     UINT32 CurrentByteOffset,
643     UINT8 *State)
644 {
645     ASL_RESOURCE_NODE *Rnode = NULL;

```





```

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:
919         /* Just ignore any of these, they are used as fillers/placeholders */
920         break;
921
922     default:
923
924         printf ("Unknown resource descriptor type [%s]\n",
925             DescriptorTypeOp->Asl.ParseOpName);
926         break;
927     }
928
929     /*
930     * Mark original node as unused, but head of a resource descriptor.
931     * This allows the resource to be installed in the namespace so that
932     * references to the descriptor can be resolved.
933     */
934     DescriptorTypeOp->Asl.ParseOpcode = PARSEOP_DEFAULT_ARG;
935     DescriptorTypeOp->Asl.CompileFlags = NODE_IS_RESOURCE_DESC;
936     DescriptorTypeOp->Asl.Value.Integer = CurrentByteOffset;
937
938     if (Rnode)
939     {
940         DescriptorTypeOp->Asl.FinalAmlLength = Rnode->BufferLength;
941         DescriptorTypeOp->Asl.Extra = ((AML_RESOURCE *) Rnode->Buffer)->Descript
942     }
943
944     return (Rnode);
945 }
946
947 /*****
948 *
949 * FUNCTION:      RsLinkDescriptorChain
950 *
951 * PARAMETERS:    PreviousRnode      - Pointer to the node that will be previous
952 *                                     to the linked node, At exit, set to the
953 *                                     last node in the new chain.
954 *                                     Rnode      - Resource node to link into the list
955 *
956 * RETURN:        Cumulative buffer byte offset of the new segment of chain
957 *
958 * DESCRIPTION:   Link a descriptor chain at the end of an existing chain.
959 *
960 * *****/
961
962 UINT32
963 RsLinkDescriptorChain (
964     ASL_RESOURCE_NODE    **PreviousRnode,
965     ASL_RESOURCE_NODE    *Rnode)
966 {
967     ASL_RESOURCE_NODE    *LastRnode;
968     UINT32                CurrentByteOffset;
969
970     /* Anything to do? */
971
972     if (!Rnode)
973     {
974         return (0);
975     }
976
977     /* Point the previous node to the new node */
978
979     (*PreviousRnode)->Next = Rnode;
980     CurrentByteOffset = Rnode->BufferLength;

```

```

985     /* Walk to the end of the chain headed by Rnode */
986
987     LastRnode = Rnode;
988     while (LastRnode->Next)
989     {
990         LastRnode = LastRnode->Next;
991         CurrentByteOffset += LastRnode->BufferLength;
992     }
993
994     /* Previous node becomes the last node in the chain */
995
996     *PreviousRnode = LastRnode;
997     return (CurrentByteOffset);
998 }
999
1000 /*****
1001 *
1002 * FUNCTION:      RsDoResourceTemplate
1003 *
1004 * PARAMETERS:    Op      - Parent of a resource template list
1005 *
1006 * RETURN:        None. Sets input node to point to a list of AML code
1007 *
1008 * DESCRIPTION:   Merge a list of resource descriptors into a single AML buffer,
1009 *                 in preparation for output to the AML output file.
1010 *
1011 * *****/
1012
1013 void
1014 RsDoResourceTemplate (
1015     ACPI_PARSE_OBJECT    *Op)
1016 {
1017     ACPI_PARSE_OBJECT    *BufferLengthOp;
1018     ACPI_PARSE_OBJECT    *BufferOp;
1019     ACPI_PARSE_OBJECT    *DescriptorTypeOp;
1020     ACPI_PARSE_OBJECT    *LastOp = NULL;
1021     UINT32                CurrentByteOffset = 0;
1022     ASL_RESOURCE_NODE    HeadRnode;
1023     ASL_RESOURCE_NODE    *PreviousRnode;
1024     ASL_RESOURCE_NODE    *Rnode;
1025     UINT8                 State;
1026
1027     /* Mark parent as containing a resource template */
1028
1029     if (Op->Asl.Parent)
1030     {
1031         Op->Asl.Parent->Asl.CompileFlags |= NODE_IS_RESOURCE_DESC;
1032     }
1033
1034     /* ResourceTemplate Opcode is first (Op) */
1035     /* Buffer Length node is first child */
1036
1037     BufferLengthOp = ASL_GET_CHILD_NODE (Op);
1038
1039     /* Buffer Op is first peer */
1040
1041     BufferOp = ASL_GET_PEER_NODE (BufferLengthOp);
1042
1043     /* First Descriptor type is next */
1044
1045     DescriptorTypeOp = ASL_GET_PEER_NODE (BufferOp);
1046
1047     /*

```

```
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);
1061
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);
1069
1070         /* Get the next descriptor in the list */
1071
1072         LastOp = DescriptorTypeOp;
1073         DescriptorTypeOp = ASL_GET_PEER_NODE (DescriptorTypeOp);
1074     }
1075
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     }
1084
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);
1096
1097     BufferLengthOp->Asl.ParseOpcode = PARSEOP_INTEGER;
1098     BufferLengthOp->Asl.Value.Integer = CurrentByteOffset;
1099     (void) OpcSetOptimalIntegerSize (BufferLengthOp);
1100     UtSetParseOpName (BufferLengthOp);
1101
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);
1109
1110     return;
1111 }
```



```

126     return (Rnode);
127 }

130 /*****
131  *
132  * FUNCTION:      RsDoMemory24Descriptor
133  *
134  * PARAMETERS:    Op                - Parent resource descriptor parse node
135  *                CurrentByteOffset - Offset into the resource template AML
136  *                buffer (to track references to the desc)
137  *
138  * RETURN:        Completed resource node
139  *
140  * DESCRIPTION: Construct a short "Memory24" descriptor
141  *
142  *****/

144 ASL_RESOURCE_NODE *
145 RsDoMemory24Descriptor (
146     ACPI_PARSE_OBJECT *Op,
147     UINT32 CurrentByteOffset)
148 {
149     AML_RESOURCE *Descriptor;
150     ACPI_PARSE_OBJECT *InitializerOp;
151     ACPI_PARSE_OBJECT *MinOp = NULL;
152     ACPI_PARSE_OBJECT *MaxOp = NULL;
153     ACPI_PARSE_OBJECT *LengthOp = NULL;
154     ASL_RESOURCE_NODE *Rnode;
155     UINT32 i;

158     InitializerOp = Op->Asl.Child;
159     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_MEMORY24));

161     Descriptor = Rnode->Buffer;
162     Descriptor->Memory24.DescriptorType = ACPI_RESOURCE_NAME_MEMORY24;
163     Descriptor->Memory24.ResourceLength = 9;

165     /* Process all child initialization nodes */
167     for (i = 0; InitializerOp; i++)
168     {
169         switch (i)
170         {
171             case 0: /* Read/Write type */

173                 RsSetFlagBits (&Descriptor->Memory24.Flags, InitializerOp, 0, 1);
174                 RsCreateBitField (InitializerOp, ACPI_RESTAG_READWRITETYPE,
175                     CurrentByteOffset + ASL_RESDDESC_OFFSET (Memory24.Flags), 0);
176                 break;

178             case 1: /* Min Address */

180                 Descriptor->Memory24.Minimum = (UINT16) InitializerOp->Asl.Value.Int
181                 RsCreateWordField (InitializerOp, ACPI_RESTAG_MINADDR,
182                     CurrentByteOffset + ASL_RESDDESC_OFFSET (Memory24.Minimum));
183                 MinOp = InitializerOp;
184                 break;

186             case 2: /* Max Address */

188                 Descriptor->Memory24.Maximum = (UINT16) InitializerOp->Asl.Value.Int
189                 RsCreateWordField (InitializerOp, ACPI_RESTAG_MAXADDR,
190                     CurrentByteOffset + ASL_RESDDESC_OFFSET (Memory24.Maximum));
191                 MaxOp = InitializerOp;

```

```

192     break;

194     case 3: /* Alignment */

196         Descriptor->Memory24.Alignment = (UINT16) InitializerOp->Asl.Value.I
197         RsCreateWordField (InitializerOp, ACPI_RESTAG_ALIGNMENT,
198             CurrentByteOffset + ASL_RESDDESC_OFFSET (Memory24.Alignment));
199         break;

201     case 4: /* Length */

203         Descriptor->Memory24.AddressLength = (UINT16) InitializerOp->Asl.Val
204         RsCreateWordField (InitializerOp, ACPI_RESTAG_LENGTH,
205             CurrentByteOffset + ASL_RESDDESC_OFFSET (Memory24.AddressLength))
206         LengthOp = InitializerOp;
207         break;

209     case 5: /* Name */

211         UtAttachNamepathToOwner (Op, InitializerOp);
212         break;

214     default:

216         AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
217         break;
218     }

220     InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
221 }

223 /* Validate the Min/Max/Len/Align values (Alignment==0 means 64K) */

225 RsSmallAddressCheck (ACPI_RESOURCE_NAME_MEMORY24,
226     Descriptor->Memory24.Minimum,
227     Descriptor->Memory24.Maximum,
228     Descriptor->Memory24.AddressLength,
229     Descriptor->Memory24.Alignment,
230     MinOp, MaxOp, LengthOp, NULL, Op);

232     return (Rnode);
233 }

236 /*****
237  *
238  * FUNCTION:      RsDoMemory32Descriptor
239  *
240  * PARAMETERS:    Op                - Parent resource descriptor parse node
241  *                CurrentByteOffset - Offset into the resource template AML
242  *                buffer (to track references to the desc)
243  *
244  * RETURN:        Completed resource node
245  *
246  * DESCRIPTION: Construct a short "Memory32" descriptor
247  *
248  *****/

250 ASL_RESOURCE_NODE *
251 RsDoMemory32Descriptor (
252     ACPI_PARSE_OBJECT *Op,
253     UINT32 CurrentByteOffset)
254 {
255     AML_RESOURCE *Descriptor;
256     ACPI_PARSE_OBJECT *InitializerOp;
257     ACPI_PARSE_OBJECT *MinOp = NULL;

```

```

258     ACPI_PARSE_OBJECT      *MaxOp = NULL;
259     ACPI_PARSE_OBJECT      *LengthOp = NULL;
260     ACPI_PARSE_OBJECT      *AlignOp = NULL;
261     ASL_RESOURCE_NODE      *Rnode;
262     UINT32                  i;

265     InitializerOp = Op->Asl.Child;
266     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_MEMORY32));

268     Descriptor = Rnode->Buffer;
269     Descriptor->Memory32.DescriptorType = ACPI_RESOURCE_NAME_MEMORY32;
270     Descriptor->Memory32.ResourceLength = 17;

272     /* Process all child initialization nodes */

274     for (i = 0; InitializerOp; i++)
275     {
276         switch (i)
277         {
278             case 0: /* Read/Write type */

280                 RsSetFlagBits (&Descriptor->Memory32.Flags, InitializerOp, 0, 1);
281                 RsCreateBitField (InitializerOp, ACPI_RESTAG_READWRITETYPE,
282                     CurrentByteOffset + ASL_RESDISC_OFFSET (Memory32.Flags), 0);
283                 break;

285             case 1: /* Min Address */

287                 Descriptor->Memory32.Minimum = (UINT32) InitializerOp->Asl.Value.Int
288                 RsCreateDwordField (InitializerOp, ACPI_RESTAG_MINADDR,
289                     CurrentByteOffset + ASL_RESDISC_OFFSET (Memory32.Minimum));
290                 MinOp = InitializerOp;
291                 break;

293             case 2: /* Max Address */

295                 Descriptor->Memory32.Maximum = (UINT32) InitializerOp->Asl.Value.Int
296                 RsCreateDwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
297                     CurrentByteOffset + ASL_RESDISC_OFFSET (Memory32.Maximum));
298                 MaxOp = InitializerOp;
299                 break;

301             case 3: /* Alignment */

303                 Descriptor->Memory32.Alignment = (UINT32) InitializerOp->Asl.Value.I
304                 RsCreateDwordField (InitializerOp, ACPI_RESTAG_ALIGNMENT,
305                     CurrentByteOffset + ASL_RESDISC_OFFSET (Memory32.Alignment));
306                 AlignOp = InitializerOp;
307                 break;

309             case 4: /* Length */

311                 Descriptor->Memory32.AddressLength = (UINT32) InitializerOp->Asl.Val
312                 RsCreateDwordField (InitializerOp, ACPI_RESTAG_LENGTH,
313                     CurrentByteOffset + ASL_RESDISC_OFFSET (Memory32.AddressLength))
314                 LengthOp = InitializerOp;
315                 break;

317             case 5: /* Name */

319                 UtAttachNamepathToOwner (Op, InitializerOp);
320                 break;

322             default:

```

```

324         AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
325         break;
326     }

328     InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
329 }

331     /* Validate the Min/Max/Len/Align values */

333     RsSmallAddressCheck (ACPI_RESOURCE_NAME_MEMORY32,
334         Descriptor->Memory32.Minimum,
335         Descriptor->Memory32.Maximum,
336         Descriptor->Memory32.AddressLength,
337         Descriptor->Memory32.Alignment,
338         MinOp, MaxOp, LengthOp, AlignOp, Op);

340     return (Rnode);
341 }

344 /*****
345  *
346  * FUNCTION:      RsDoMemory32FixedDescriptor
347  *
348  * PARAMETERS:    Op          - Parent resource descriptor parse node
349  *                CurrentByteOffset - Offset into the resource template AML
350  *                buffer (to track references to the desc)
351  *
352  * RETURN:        Completed resource node
353  *
354  * DESCRIPTION:   Construct a short "Memory32Fixed" descriptor
355  *
356  *****/

358 ASL_RESOURCE_NODE *
359 RsDoMemory32FixedDescriptor (
360     ACPI_PARSE_OBJECT      *Op,
361     UINT32                  CurrentByteOffset)
362 {
363     AML_RESOURCE            *Descriptor;
364     ACPI_PARSE_OBJECT      *InitializerOp;
365     ASL_RESOURCE_NODE      *Rnode;
366     UINT32                  i;

369     InitializerOp = Op->Asl.Child;
370     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_FIXED_MEMORY32));

372     Descriptor = Rnode->Buffer;
373     Descriptor->FixedMemory32.DescriptorType = ACPI_RESOURCE_NAME_FIXED_MEMORY32
374     Descriptor->FixedMemory32.ResourceLength = 9;

376     /* Process all child initialization nodes */

378     for (i = 0; InitializerOp; i++)
379     {
380         switch (i)
381         {
382             case 0: /* Read/Write type */

384                 RsSetFlagBits (&Descriptor->FixedMemory32.Flags, InitializerOp, 0, 1
385                 RsCreateBitField (InitializerOp, ACPI_RESTAG_READWRITETYPE,
386                     CurrentByteOffset + ASL_RESDISC_OFFSET (FixedMemory32.Flags), 0)
387                 break;

389             case 1: /* Address */

```

```

391     Descriptor->FixedMemory32.Address = (UINT32) InitializerOp->Asl.Valu
392     RsCreateDwordField (InitializerOp, ACPI_RESTAG_BASEADDRESS,
393     CurrentByteOffset + ASL_RESDDESC_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_RESDDESC_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_RESDDESC_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)
523 *
524 * RETURN:      Completed resource node
525 *
526 * DESCRIPTION: Construct a short "StartDependentNoPri" descriptor
527 *
528 *****/
530 ASL_RESOURCE_NODE *
531 RsDoStartDependentNoPriDescriptor (
532     ACPI_PARSE_OBJECT *Op,
533     UINT32 CurrentByteOffset)
534 {
535     AML_RESOURCE *Descriptor;
536     ACPI_PARSE_OBJECT *InitializerOp;
537     ASL_RESOURCE_NODE *Rnode;
538     ASL_RESOURCE_NODE *PreviousRnode;
539     ASL_RESOURCE_NODE *NextRnode;
540     UINT8 State;
541
542     InitializerOp = Op->Asl.Child;
543     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_START_DEPENDENT_NOPRIO))
544
545     Descriptor = Rnode->Buffer;
546     Descriptor->StartDpf.DescriptorType = ACPI_RESOURCE_NAME_START_DEPENDENT |
547                                         ASL_RDESC_ST_DEPEND_SIZE;
548     PreviousRnode = Rnode;
549
550     /* Increment offset past StartDependentNoPri descriptor */
551
552     CurrentByteOffset += sizeof (AML_RESOURCE_START_DEPENDENT_NOPRIO);
553
554     /* Process all child initialization nodes */
555
556     State = ACPI_RSTATE_START_DEPENDENT;
557     while (InitializerOp)
558     {
559         NextRnode = RsDoOneResourceDescriptor (InitializerOp,
560                                                 CurrentByteOffset, &State);
561
562         /*
563          * Update current byte offset to indicate the number of bytes from the
564          * start of the buffer. Buffer can include multiple descriptors, we
565          * must keep track of the offset of not only each descriptor, but each
566          * element (field) within each descriptor as well.
567          */
568         CurrentByteOffset += RsLinkDescriptorChain (&PreviousRnode, NextRnode);
569
570         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
571     }
572
573     return (Rnode);
574 }
575
576 *****/
578 /*****
579 *
580 * FUNCTION:      RsDoVendorSmallDescriptor
581 *
582 * PARAMETERS:    Op          - Parent resource descriptor parse node
583 *                CurrentByteOffset - Offset into the resource template AML
584 *                                     buffer (to track references to the desc)
585 *
586 * RETURN:      Completed resource node
587 *

```

```

588 * DESCRIPTION: Construct a short "VendorShort" descriptor
589 *
590 *****/
592 ASL_RESOURCE_NODE *
593 RsDoVendorSmallDescriptor (
594     ACPI_PARSE_OBJECT *Op,
595     UINT32 CurrentByteOffset)
596 {
597     AML_RESOURCE *Descriptor;
598     ACPI_PARSE_OBJECT *InitializerOp;
599     ASL_RESOURCE_NODE *Rnode;
600     UINT8 *VendorData;
601     UINT32 i;
602
603     InitializerOp = Op->Asl.Child;
604
605     /* Allocate worst case - 7 vendor bytes */
606
607     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_VENDOR_SMALL) + 7);
608
609     Descriptor = Rnode->Buffer;
610     Descriptor->VendorSmall.DescriptorType = ACPI_RESOURCE_NAME_VENDOR_SMALL;
611     VendorData = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_VENDOR_SMALL_HEADER);
612
613     /* Process all child initialization nodes */
614
615     InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
616     for (i = 0; InitializerOp; i++)
617     {
618         if (InitializerOp->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
619         {
620             break;
621         }
622
623         /* Maximum 7 vendor data bytes allowed (0-6) */
624
625         if (i >= 7)
626         {
627             AslError (ASL_ERROR, ASL_MSG_VENDOR_LIST, InitializerOp, NULL);
628
629             /* Eat the excess initializers */
630
631             while (InitializerOp)
632             {
633                 InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
634             }
635             break;
636         }
637
638         VendorData[i] = (UINT8) InitializerOp->Asl.Value.Integer;
639         InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
640     }
641
642     /* Adjust the Rnode buffer size, so correct number of bytes are emitted */
643
644     Rnode->BufferLength -= (7 - i);
645
646     /* Set the length in the Type Tag */
647
648     Descriptor->VendorSmall.DescriptorType |= (UINT8) i;
649     return (Rnode);
650 }
651

```



new/usr/src/common/acpica/compiler/aslrestypeli.c

1

```
*****
21406 Fri Jan 17 07:29:39 2014
new/usr/src/common/acpica/compiler/aslrestypeli.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslrestypeli - Small I/O-related resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"

48 #define _COMPONENT          ACPI_COMPILER
49 #define ACPI_MODULE_NAME    ("aslrestypeli")

51 /*
52 * This module contains the I/O-related small resource descriptors:
53 *
54 * DMA
55 * FixedDMA
56 * FixedIO
57 * IO
58 * IRQ
59 * IRQNoFlags
```

new/usr/src/common/acpica/compiler/aslrestypeli.c

2

```
60 */

62 /*****
63 *
64 * FUNCTION:      RsDoDmaDescriptor
65 *
66 * PARAMETERS:    Op                - Parent resource descriptor parse node
67 *                CurrentByteOffset - Offset into the resource template AML
68 *                buffer (to track references to the desc)
69 *
70 * RETURN:        Completed resource node
71 *
72 * DESCRIPTION:    Construct a short "DMA" descriptor
73 *
74 *****/

76 ASL_RESOURCE_NODE *
77 RsDoDmaDescriptor (
78     ACPI_PARSE_OBJECT *Op,
79     UINT32 CurrentByteOffset)
80 {
81     AML_RESOURCE *Descriptor;
82     ACPI_PARSE_OBJECT *InitializerOp;
83     ASL_RESOURCE_NODE *Rnode;
84     UINT32 i;
85     UINT8 DmaChannelMask = 0;
86     UINT8 DmaChannels = 0;

89     InitializerOp = Op->Asl.Child;
90     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_DMA));

92     Descriptor = Rnode->Buffer;
93     Descriptor->Dma.DescriptorType = ACPI_RESOURCE_NAME_DMA |
94                                     ASL_RDESC_DMA_SIZE;

96     /* Process all child initialization nodes */

98     for (i = 0; InitializerOp; i++)
99     {
100         switch (i)
101         {
102             case 0: /* DMA type */

104                 RsSetFlagBits (&Descriptor->Dma.Flags, InitializerOp, 5, 0);
105                 RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_DMATYPE,
106                                         CurrentByteOffset + ASL_RESDDESC_OFFSET (Dma.Flags), 5, 2);
107                 break;

109             case 1: /* Bus Master */

111                 RsSetFlagBits (&Descriptor->Dma.Flags, InitializerOp, 2, 0);
112                 RsCreateBitField (InitializerOp, ACPI_RESTAG_BUSMASTER,
113                                   CurrentByteOffset + ASL_RESDDESC_OFFSET (Dma.Flags), 2);
114                 break;

116             case 2: /* Xfer Type (transfer width) */

118                 RsSetFlagBits (&Descriptor->Dma.Flags, InitializerOp, 0, 0);
119                 RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_XFERTYPE,
120                                         CurrentByteOffset + ASL_RESDDESC_OFFSET (Dma.Flags), 0, 2);
121                 break;

123             case 3: /* Name */

125                 UtAttachNamepathToOwner (Op, InitializerOp);
```

```

126         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.Value.Integer;
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.Integer;
231                 RsCreateWordField (InitializerOp, ACPI_RESTAG_DMA,
232                     CurrentByteOffset + ASL_RESDESC_OFFSET (FixedDma.Channels));
233                 break;
235             case 2: /* Transfer Width [BYTE] (_SIZ) */
237                 Descriptor->FixedDma.Width = (UINT8) InitializerOp->Asl.Value.Integer;
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 }

```

```

259 /*****
260 *
261 * FUNCTION:      RsDoFixedIoDescriptor
262 *
263 * PARAMETERS:   Op          - Parent resource descriptor parse node
264 *               CurrentByteOffset - Offset into the resource template AML
265 *               buffer (to track references to the desc)
266 *
267 * RETURN:       Completed resource node
268 *
269 * DESCRIPTION: Construct a short "FixedIO" descriptor
270 *
271 *****/

273 ASL_RESOURCE_NODE *
274 RsDoFixedIoDescriptor (
275     ACPI_PARSE_OBJECT *Op,
276     UINT32 CurrentByteOffset)
277 {
278     AML_RESOURCE *Descriptor;
279     ACPI_PARSE_OBJECT *InitializerOp;
280     ACPI_PARSE_OBJECT *AddressOp = NULL;
281     ASL_RESOURCE_NODE *Rnode;
282     UINT32 i;

285     InitializerOp = Op->Asl.Child;
286     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_FIXED_IO));

288     Descriptor = Rnode->Buffer;
289     Descriptor->Io.DescriptorType = ACPI_RESOURCE_NAME_FIXED_IO |
290         ASL_RDESC_FIXED_IO_SIZE;

292     /* Process all child initialization nodes */

294     for (i = 0; InitializerOp; i++)
295     {
296         switch (i)
297         {
298             case 0: /* Base Address */

300                 Descriptor->FixedIo.Address =
301                     (UINT16) InitializerOp->Asl.Value.Integer;
302                 RsCreateWordField (InitializerOp, ACPI_RESTAG_BASEADDRESS,
303                     CurrentByteOffset + ASL_RESDESC_OFFSET (FixedIo.Address));
304                 AddressOp = InitializerOp;
305                 break;

307             case 1: /* Length */

309                 Descriptor->FixedIo.AddressLength =
310                     (UINT8) InitializerOp->Asl.Value.Integer;
311                 RsCreateByteField (InitializerOp, ACPI_RESTAG_LENGTH,
312                     CurrentByteOffset + ASL_RESDESC_OFFSET (FixedIo.AddressLength));
313                 break;

315             case 2: /* Name */

317                 UtAttachNamepathToOwner (Op, InitializerOp);
318                 break;

320             default:

322                 AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
323                 break;

```

```

324     }

326     InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
327 }

329 /* Error checks */

331 if (Descriptor->FixedIo.Address > 0x03FF)
332 {
333     AslError (ASL_WARNING, ASL_MSG_ISA_ADDRESS, AddressOp, NULL);
334 }

336 return (Rnode);
337 }

340 /*****
341 *
342 * FUNCTION:      RsDoIoDescriptor
343 *
344 * PARAMETERS:   Op          - Parent resource descriptor parse node
345 *               CurrentByteOffset - Offset into the resource template AML
346 *               buffer (to track references to the desc)
347 *
348 * RETURN:       Completed resource node
349 *
350 * DESCRIPTION: Construct a short "IO" descriptor
351 *
352 *****/

354 ASL_RESOURCE_NODE *
355 RsDoIoDescriptor (
356     ACPI_PARSE_OBJECT *Op,
357     UINT32 CurrentByteOffset)
358 {
359     AML_RESOURCE *Descriptor;
360     ACPI_PARSE_OBJECT *InitializerOp;
361     ACPI_PARSE_OBJECT *MinOp = NULL;
362     ACPI_PARSE_OBJECT *MaxOp = NULL;
363     ACPI_PARSE_OBJECT *LengthOp = NULL;
364     ACPI_PARSE_OBJECT *AlignOp = NULL;
365     ASL_RESOURCE_NODE *Rnode;
366     UINT32 i;

369     InitializerOp = Op->Asl.Child;
370     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_IO));

372     Descriptor = Rnode->Buffer;
373     Descriptor->Io.DescriptorType = ACPI_RESOURCE_NAME_IO |
374         ASL_RDESC_IO_SIZE;

376     /* Process all child initialization nodes */

378     for (i = 0; InitializerOp; i++)
379     {
380         switch (i)
381         {
382             case 0: /* Decode size */

384                 RsSetFlagBits (&Descriptor->Io.Flags, InitializerOp, 0, 1);
385                 RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
386                     CurrentByteOffset + ASL_RESDESC_OFFSET (Io.Flags), 0);
387                 break;

389             case 1: /* Min Address */

```

```

391     Descriptor->Io.Minimum =
392         (UINT16) InitializerOp->Asl.Value.Integer;
393     RsCreateWordField (InitializerOp, ACPI_RESTAG_MINADDR,
394         CurrentByteOffset + ASL_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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 */
523
524      if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
525      {
526          /* Up to 16 interrupts can be specified in the list */
527
528          Interrupts++;
529          if (Interrupts > 16)
530          {
531              AslError (ASL_ERROR, ASL_MSG_INTERRUPT_LIST,
532                      InitializerOp, NULL);
533              return (Rnode);
534          }
535
536          /* Only interrupts 0-15 are allowed (mask is 16 bits) */
537
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      }
548
549      /* Case 4: First IRQ value in list */
550
551      if (i == 4)
552      {
553          /* Check now for duplicates in list */
554
555          RsCheckListForDuplicates (InitializerOp);
556
557          /* Create a named field at the start of the list */
558
559          RsCreateWordField (InitializerOp, ACPI_RESTAG_INTERRUPT,
560                           CurrentByteOffset + ASL_RESDESC_OFFSET (Irq.IrqMask));
561      }
562      break;
563 }
564
565 InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
566 }
567
568 /* Now we can set the channel mask */
569
570 Descriptor->Irq.IrqMask = IrqMask;
571 return (Rnode);
572 }
573
574 /*****
575 *
576 * FUNCTION:      RsDoIrqNoFlagsDescriptor
577 *
578 * PARAMETERS:  Op          - Parent resource descriptor parse node
579 *              CurrentByteOffset - Offset into the resource template AML
580 *                               buffer (to track references to the desc)
581 *
582 * RETURN:      Completed resource node
583 *
584 * DESCRIPTION: Construct a short "IRQNoFlags" descriptor
585 *
586 * *****/

```

```

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;
600
601     InitializerOp = Op->Asl.Child;
602     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_IRQ_NOFLAGS));
603
604     Descriptor = Rnode->Buffer;
605     Descriptor->Irq.DescriptorType = ACPI_RESOURCE_NAME_IRQ |
606                                     ASL_RDESC_IRQ_SIZE;
607
608     /* Process all child initialization nodes */
609
610     for (i = 0; InitializerOp; i++)
611     {
612         switch (i)
613         {
614             case 0: /* Name */
615
616                 UtAttachNamepathToOwner (Op, InitializerOp);
617                 break;
618
619             default:
620
621                 /* IRQ bytes are handled here, after the flags and name */
622
623                 if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
624                 {
625                     /* Up to 16 interrupts can be specified in the list */
626
627                     Interrupts++;
628                     if (Interrupts > 16)
629                     {
630                         AslError (ASL_ERROR, ASL_MSG_INTERRUPT_LIST,
631                                 InitializerOp, NULL);
632                         return (Rnode);
633                     }
634
635                     /* Only interrupts 0-15 are allowed (mask is 16 bits) */
636
637                     if (InitializerOp->Asl.Value.Integer > 15)
638                     {
639                         AslError (ASL_ERROR, ASL_MSG_INTERRUPT_NUMBER,
640                                 InitializerOp, NULL);
641                     }
642                     else
643                     {
644                         IrqMask |= (1 << ((UINT8) InitializerOp->Asl.Value.Integer))
645                     }
646                 }
647             }
648         }
649
650     /* Case 1: First IRQ value in list */
651
652     if (i == 1)
653     {
654         /* 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_RESDDESC_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 }
```

new/usr/src/common/acpica/compiler/aslrestype2.c

1

```
*****
15490 Fri Jan 17 07:29:39 2014
new/usr/src/common/acpica/compiler/aslrestype2.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslrestype2 - Miscellaneous Large resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"
47 #include "amlcode.h"

49 #define _COMPONENT          ACPI_COMPILER
50         ACPI_MODULE_NAME    ("aslrestype2")

52 /*
53 * This module contains miscellaneous large resource descriptors:
54 *
55 * Register
56 * Interrupt
57 * VendorLong
58 */
```

new/usr/src/common/acpica/compiler/aslrestype2.c

2

```
60 /*****
61 *
62 * FUNCTION:      RsDoGeneralRegisterDescriptor
63 *
64 * PARAMETERS:    Op                - Parent resource descriptor parse node
65 *                CurrentByteOffset - Offset into the resource template AML
66 *                buffer (to track references to the desc)
67 *
68 * RETURN:        Completed resource node
69 *
70 * DESCRIPTION:    Construct a long "Register" descriptor
71 *
72 *****/

74 ASL_RESOURCE_NODE *
75 RsDoGeneralRegisterDescriptor (
76     ACPI_PARSE_OBJECT *Op,
77     UINT32 CurrentByteOffset)
78 {
79     AML_RESOURCE *Descriptor;
80     ACPI_PARSE_OBJECT *InitializerOp;
81     ASL_RESOURCE_NODE *Rnode;
82     UINT32 i;

85     InitializerOp = Op->Asl.Child;
86     Rnode = RsAllocateResourceNode (sizeof (AML_RESOURCE_GENERIC_REGISTER));

88     Descriptor = Rnode->Buffer;
89     Descriptor->GenericReg.DescriptorType = ACPI_RESOURCE_NAME_GENERIC_REGISTER;
90     Descriptor->GenericReg.ResourceLength = 12;

92     /* Process all child initialization nodes */

94     for (i = 0; InitializerOp; i++)
95     {
96         switch (i)
97         {
98             case 0: /* Address space */

100                 Descriptor->GenericReg.AddressSpaceId = (UINT8) InitializerOp->Asl.V
101                 RsCreateByteField (InitializerOp, ACPI_RESTAG_ADDRESSSPACE,
102                 CurrentByteOffset + ASL_RESDDESC_OFFSET (GenericReg.AddressSpaceI
103                 break;

105             case 1: /* Register Bit Width */

107                 Descriptor->GenericReg.BitWidth = (UINT8) InitializerOp->Asl.Value.I
108                 RsCreateByteField (InitializerOp, ACPI_RESTAG_REGISTERBITWIDTH,
109                 CurrentByteOffset + ASL_RESDDESC_OFFSET (GenericReg.BitWidth));
110                 break;

112             case 2: /* Register Bit Offset */

114                 Descriptor->GenericReg.BitOffset = (UINT8) InitializerOp->Asl.Value.
115                 RsCreateByteField (InitializerOp, ACPI_RESTAG_REGISTERBITOFFSET,
116                 CurrentByteOffset + ASL_RESDDESC_OFFSET (GenericReg.BitOffset));
117                 break;

119             case 3: /* Register Address */

121                 Descriptor->GenericReg.Address = InitializerOp->Asl.Value.Integer;
122                 RsCreateQwordField (InitializerOp, ACPI_RESTAG_ADDRESS,
123                 CurrentByteOffset + ASL_RESDDESC_OFFSET (GenericReg.Address));
124                 break;
```

```

126     case 4: /* Access Size (ACPI 3.0) */
127
128         Descriptor->GenericReg.AccessSize = (UINT8) InitializerOp->Asl.Value
129         RsCreateByteField (InitializerOp, ACPI_RESTAG_ACCESSSIZE,
130             CurrentByteOffset + ASL_RESDISC_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 }

```

```

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 *****/

```

```

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     InitializerOp = Op->Asl.Child;
187     StringLength = RsGetStringDataLength (InitializerOp);
188
189     /* Count the interrupt numbers */

```

```

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_RESDISC_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_RESDISC_OFFSET (ExtendedIrq.Flags), 2);
254                 break;
255
256             case 3: /* Share Type - Default: exclusive (0) */

```



```

258     RsSetFlagBits (&Descriptor->ExtendedIrq.Flags, InitializerOp, 3, 0);
259     RsCreateBitField (InitializerOp, ACPI_RESTAG_INTERRUPTSHARE,
260         CurrentByteOffset + ASL_RESDDESC_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                 ResResourceString = (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_RESDDESC_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 && ResResourceString)
378     {
380         strcpy ((char *) Rover, (char *) ResResourceString);
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_RESDDESC_OFFSET (ExtendedIrq.Interrupts[0]) -
389         ASL_RESDDESC_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 }

```

new/usr/src/common/acpica/compiler/aslrestype2d.c

1

```
*****
26143 Fri Jan 17 07:29:39 2014
new/usr/src/common/acpica/compiler/aslrestype2d.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslrestype2d - Large DWord address resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"

48 #define _COMPONENT          ACPI_COMPILER
49 #define ACPI_MODULE_NAME    ("aslrestype2d")

51 /*
52 * This module contains the Dword (32-bit) address space descriptors:
53 *
54 * DwordIO
55 * DwordMemory
56 * DwordSpace
57 */

59 /*****
```

new/usr/src/common/acpica/compiler/aslrestype2d.c

2

```
60 *
61 * FUNCTION:      RsDoDwordIoDescriptor
62 *
63 * PARAMETERS:    Op                - Parent resource descriptor parse node
64 *                CurrentByteOffset - Offset into the resource template AML
65 *                                     buffer (to track references to the desc)
66 *
67 * RETURN:        Completed resource node
68 *
69 * DESCRIPTION:    Construct a long "DwordIO" descriptor
70 *
71 *****/

73 ASL_RESOURCE_NODE *
74 RsDoDwordIoDescriptor (
75     ACPI_PARSE_OBJECT *Op,
76     UINT32              CurrentByteOffset)
77 {
78     AML_RESOURCE *Descriptor;
79     ACPI_PARSE_OBJECT *InitializerOp;
80     ACPI_PARSE_OBJECT *MinOp = NULL;
81     ACPI_PARSE_OBJECT *MaxOp = NULL;
82     ACPI_PARSE_OBJECT *LengthOp = NULL;
83     ACPI_PARSE_OBJECT *GranOp = NULL;
84     ASL_RESOURCE_NODE *Rnode;
85     UINT16              StringLength = 0;
86     UINT32              OptionIndex = 0;
87     UINT8               *OptionalFields;
88     UINT32              i;
89     BOOLEAN              ResSourceIndex = FALSE;

92     InitializerOp = Op->Asl.Child;
93     StringLength = RsGetStringDataLength (InitializerOp);

95     Rnode = RsAllocateResourceNode (
96         sizeof (AML_RESOURCE_ADDRESS32) + 1 + StringLength);

98     Descriptor = Rnode->Buffer;
99     Descriptor->Address32.DescriptorType = ACPI_RESOURCE_NAME_ADDRESS32;
100     Descriptor->Address32.ResourceType = ACPI_ADDRESS_TYPE_IO_RANGE;

102     /*
103      * Initial descriptor length -- may be enlarged if there are
104      * optional fields present
105      */
106     OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS32);
107     Descriptor->Address32.ResourceLength = (UINT16)
108         (sizeof (AML_RESOURCE_ADDRESS32) -
109          sizeof (AML_RESOURCE_LARGE_HEADER));

111     /* Process all child initialization nodes */

113     for (i = 0; InitializerOp; i++)
114     {
115         switch (i)
116         {
117             case 0: /* Resource Usage */

119                 RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 0, 1);
120                 break;

122             case 1: /* MinType */

124                 RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 2, 0);
125                 RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
```

```

126     CurrentByteOffset + ASL_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_OFFSET (Address32.AddressLength)
191         LengthOp = InitializerOp;

```

```

192     break;

194     case 10: /* ResourceIndex [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             ResourceIndex = TRUE;
204         }
205         break;

207     case 11: /* Resource [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 (!ResourceIndex)
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 (ResourceIndex)
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_RESDDESC_OFFSET (Address32.SpecificFlags)
256         break;

```

```

258     case 14: /* Translation Type */
259
260         RsSetFlagBits (&Descriptor->Address32.SpecificFlags, InitializerOp,
261             RsCreateBitField (InitializerOp, ACPI_RESTAG_TRANSTYPE,
262                 CurrentByteOffset + ASL_RESDDESC_OFFSET (Address32.SpecificFlags)
263             break;
264
265     default:
266
267         AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
268         break;
269     }
270
271     InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
272 }
273
274 /* Validate the Min/Max/Len/Gran values */
275
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);
283
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);
325
326     Rnode = RsAllocateResourceNode (
327         sizeof (AML_RESOURCE_ADDRESS32) + 1 + StringLength);
328
329     Descriptor = Rnode->Buffer;
330     Descriptor->Address32.DescriptorType = ACPI_RESOURCE_NAME_ADDRESS32;
331     Descriptor->Address32.ResourceType = ACPI_ADDRESS_TYPE_MEMORY_RANGE;
332
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));
341
342     /* Process all child initialization nodes */
343
344     for (i = 0; InitializerOp; i++)
345     {
346         switch (i)
347         {
348             case 0: /* Resource Usage */
349
350                 RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 0, 1);
351                 break;
352
353             case 1: /* DecodeType */
354
355                 RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 1, 0);
356                 RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
357                     CurrentByteOffset + ASL_RESDDESC_OFFSET (Address32.Flags), 1);
358                 break;
359
360             case 2: /* MinType */
361
362                 RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 2, 0);
363                 RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
364                     CurrentByteOffset + ASL_RESDDESC_OFFSET (Address32.Flags), 2);
365                 break;
366
367             case 3: /* MaxType */
368
369                 RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 3, 0);
370                 RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
371                     CurrentByteOffset + ASL_RESDDESC_OFFSET (Address32.Flags), 3);
372                 break;
373
374             case 4: /* Memory Type */
375
376                 RsSetFlagBits (&Descriptor->Address32.SpecificFlags, InitializerOp,
377                     RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_MEMTYPE,
378                         CurrentByteOffset + ASL_RESDDESC_OFFSET (Address32.SpecificFlags)
379                     break;
380
381             case 5: /* Read/Write Type */
382
383                 RsSetFlagBits (&Descriptor->Address32.SpecificFlags, InitializerOp,
384                     RsCreateBitField (InitializerOp, ACPI_RESTAG_READWRITETYPE,
385                         CurrentByteOffset + ASL_RESDDESC_OFFSET (Address32.SpecificFlags)
386                     break;
387
388             case 6: /* Address Granularity */

```

```

391     Descriptor->Address32.Granularity =
392         (UINT32) InitializerOp->Asl.Value.Integer;
393     RsCreateDwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
394         CurrentByteOffset + ASL_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_OFFSET (Address32.SpecificFlags)
492         break;

494 case 15: /* Type */

496     RsSetFlagBits (&Descriptor->Address32.SpecificFlags, InitializerOp,
497         RsCreateBitField (InitializerOp, ACPI_RESTAG_TYPE,
498             CurrentByteOffset + ASL_RESDDESC_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);
523 }

526 /*****
527 *
528 * FUNCTION:      RsDoDwordSpaceDescriptor
529 *
530 * PARAMETERS:    Op                - Parent resource descriptor parse node
531 *                CurrentByteOffset - Offset into the resource template AML
532 *                buffer (to track references to the desc)
533 *
534 * RETURN:        Completed resource node
535 *
536 * DESCRIPTION:   Construct a long "DwordSpace" descriptor
537 *
538 *****/

540 ASL_RESOURCE_NODE *
541 RsDoDwordSpaceDescriptor (
542     ACPI_PARSE_OBJECT *Op,
543     UINT32 CurrentByteOffset)
544 {
545     AML_RESOURCE *Descriptor;
546     ACPI_PARSE_OBJECT *InitializerOp;
547     ACPI_PARSE_OBJECT *MinOp = NULL;
548     ACPI_PARSE_OBJECT *MaxOp = NULL;
549     ACPI_PARSE_OBJECT *LengthOp = NULL;
550     ACPI_PARSE_OBJECT *GranOp = NULL;
551     ASL_RESOURCE_NODE *Rnode;
552     UINT8 *OptionalFields;
553     UINT16 StringLength = 0;
554     UINT32 OptionIndex = 0;
555     UINT32 i;
556     BOOLEAN ResSourceIndex = FALSE;

559     InitializerOp = Op->Asl.Child;
560     StringLength = RsGetStringDataLength (InitializerOp);

562     Rnode = RsAllocateResourceNode (
563         sizeof (AML_RESOURCE_ADDRESS32) + 1 + StringLength);

565     Descriptor = Rnode->Buffer;
566     Descriptor->Address32.DescriptorType = ACPI_RESOURCE_NAME_ADDRESS32;

568     /*
569     * Initial descriptor length -- may be enlarged if there are
570     * optional fields present
571     */
572     OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS32);
573     Descriptor->Address32.ResourceLength = (UINT16)
574         (sizeof (AML_RESOURCE_ADDRESS32) -
575          sizeof (AML_RESOURCE_LARGE_HEADER));

577     /* Process all child initialization nodes */

579     for (i = 0; InitializerOp; i++)
580     {
581         switch (i)
582         {
583             case 0: /* Resource Type */

585                 Descriptor->Address32.ResourceType =
586                     (UINT8) InitializerOp->Asl.Value.Integer;
587                 break;

```

```

589         case 1: /* Resource Usage */

591             RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 0, 1);
592             break;

594         case 2: /* DecodeType */

596             RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 1, 0);
597             RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
598                 CurrentByteOffset + ASL_RESDISC_OFFSET (Address32.Flags), 1);
599             break;

601         case 3: /* MinType */

603             RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 2, 0);
604             RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
605                 CurrentByteOffset + ASL_RESDISC_OFFSET (Address32.Flags), 2);
606             break;

608         case 4: /* MaxType */

610             RsSetFlagBits (&Descriptor->Address32.Flags, InitializerOp, 3, 0);
611             RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
612                 CurrentByteOffset + ASL_RESDISC_OFFSET (Address32.Flags), 3);
613             break;

615         case 5: /* Type-Specific flags */

617             Descriptor->Address32.SpecificFlags =
618                 (UINT8) InitializerOp->Asl.Value.Integer;
619             break;

621         case 6: /* Address Granularity */

623             Descriptor->Address32.Granularity =
624                 (UINT32) InitializerOp->Asl.Value.Integer;
625             RsCreateDwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
626                 CurrentByteOffset + ASL_RESDISC_OFFSET (Address32.Granularity));
627             GranOp = InitializerOp;
628             break;

630         case 7: /* Min Address */

632             Descriptor->Address32.Minimum =
633                 (UINT32) InitializerOp->Asl.Value.Integer;
634             RsCreateDwordField (InitializerOp, ACPI_RESTAG_MINADDR,
635                 CurrentByteOffset + ASL_RESDISC_OFFSET (Address32.Minimum));
636             MinOp = InitializerOp;
637             break;

639         case 8: /* Max Address */

641             Descriptor->Address32.Maximum =
642                 (UINT32) InitializerOp->Asl.Value.Integer;
643             RsCreateDwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
644                 CurrentByteOffset + ASL_RESDISC_OFFSET (Address32.Maximum));
645             MaxOp = InitializerOp;
646             break;

648         case 9: /* Translation Offset */

650             Descriptor->Address32.TranslationOffset =
651                 (UINT32) InitializerOp->Asl.Value.Integer;
652             RsCreateDwordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
653                 CurrentByteOffset + ASL_RESDISC_OFFSET (Address32.TranslationOff

```

```

654         break;

655     case 10: /* Address Length */

656         Descriptor->Address32.AddressLength =
657             (UINT32) InitializerOp->Asl.Value.Integer;
658         RsCreateDwordField (InitializerOp, ACPI_RESTAG_LENGTH,
659             CurrentByteOffset + ASL_RESDDESC_OFFSET (Address32.AddressLength)
660             LengthOp = InitializerOp;
661             break;

662     case 11: /* ResSourceIndex [Optional Field - BYTE] */

663         if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
664         {
665             OptionalFields[0] = (UINT8) InitializerOp->Asl.Value.Integer;
666             OptionIndex++;
667             Descriptor->Address32.ResourceLength++;
668             ResSourceIndex = TRUE;
669         }
670         break;

671     case 12: /* ResSource [Optional Field - STRING] */

672         if ((InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
673             (InitializerOp->Asl.Value.String))
674         {
675             if (StringLength)
676             {
677                 Descriptor->Address32.ResourceLength = (UINT16)
678                     (Descriptor->Address32.ResourceLength + StringLength);

679                 strcpy ((char *)
680                     &OptionalFields[OptionIndex],
681                     InitializerOp->Asl.Value.String);

682                 /* ResourceSourceIndex must also be valid */

683                 if (!ResSourceIndex)
684                 {
685                     AslError (ASL_ERROR, ASL_MSG_RESOURCE_INDEX,
686                         InitializerOp, NULL);
687                 }
688             }
689         }

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;

712     case 13: /* ResourceTag */

713         UtAttachNamepathToOwner (Op, InitializerOp);
714         break;

715     default:

```

```

720         AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST,
721             InitializerOp, NULL);
722         break;
723     }

724     InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
725 }

726 /* Validate the Min/Max/Len/Gran values */

727 RsLargeAddressCheck (
728     (UINT64) Descriptor->Address32.Minimum,
729     (UINT64) Descriptor->Address32.Maximum,
730     (UINT64) Descriptor->Address32.AddressLength,
731     (UINT64) Descriptor->Address32.Granularity,
732     Descriptor->Address32.Flags,
733     MinOp, MaxOp, LengthOp, GranOp, Op);

734 Rnode->BufferLength = sizeof (AML_RESOURCE_ADDRESS32) +
735     OptionIndex + StringLength;
736 return (Rnode);
737 }

```



new/usr/src/common/acpica/compiler/aslrestype2e.c

1

```
*****
21325 Fri Jan 17 07:29:39 2014
new/usr/src/common/acpica/compiler/aslrestype2e.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslrestype2e - Large Extended address resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"

47 #define _COMPONENT          ACPI_COMPILER
48         ACPI_MODULE_NAME    ("aslrestype2e")

50 /*
51 * This module contains the Extended (64-bit) address space descriptors:
52 * ExtendedIO
53 * ExtendedMemory
54 * ExtendedSpace
55 */

58 /*****
59 *
```

new/usr/src/common/acpica/compiler/aslrestype2e.c

2

```
60 * FUNCTION:      RsDoExtendedIoDescriptor
61 *
62 * PARAMETERS:    Op          - Parent resource descriptor parse node
63 *                CurrentByteOffset - Offset into the resource template AML
64 *                buffer (to track references to the desc)
65 *
66 * RETURN:        Completed resource node
67 *
68 * DESCRIPTION:    Construct a long "ExtendedIO" descriptor
69 *
70 *****/

72 ASL_RESOURCE_NODE *
73 RsDoExtendedIoDescriptor (
74     ACPI_PARSE_OBJECT *Op,
75     UINT32 CurrentByteOffset)
76 {
77     AML_RESOURCE *Descriptor;
78     ACPI_PARSE_OBJECT *InitializerOp;
79     ACPI_PARSE_OBJECT *MinOp = NULL;
80     ACPI_PARSE_OBJECT *MaxOp = NULL;
81     ACPI_PARSE_OBJECT *LengthOp = NULL;
82     ACPI_PARSE_OBJECT *GranOp = NULL;
83     ASL_RESOURCE_NODE *Rnode;
84     UINT16 StringLength = 0;
85     UINT32 i;

88     InitializerOp = Op->Asl.Child;
89     StringLength = RsGetStringDataLength (InitializerOp);

91     Rnode = RsAllocateResourceNode (
92         sizeof (AML_RESOURCE_EXTENDED_ADDRESS64) + 1 + StringLength);

94     Descriptor = Rnode->Buffer;
95     Descriptor->ExtAddress64.DescriptorType = ACPI_RESOURCE_NAME_EXTENDED_ADDRE
96     Descriptor->ExtAddress64.ResourceType = ACPI_ADDRESS_TYPE_IO_RANGE;
97     Descriptor->ExtAddress64.RevisionID = AML_RESOURCE_EXTENDED_ADDRESS_REV

99     Descriptor->ExtAddress64.ResourceLength = (UINT16)
100         (sizeof (AML_RESOURCE_EXTENDED_ADDRESS64) -
101          sizeof (AML_RESOURCE_LARGE_HEADER));

103     /* Process all child initialization nodes */

105     for (i = 0; InitializerOp; i++)
106     {
107         switch (i)
108         {
109             case 0: /* Resource Usage */

111                 RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 0, 1)
112                 break;

114             case 1: /* MinType */

116                 RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 2, 0)
117                 RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
118                     CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.Flags), 2);
119                 break;

121             case 2: /* MaxType */

123                 RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 3, 0)
124                 RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
125                     CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.Flags), 3);
```

```

126         break;

128     case 3: /* DecodeType */

130         RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 1, 0)
131         RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
132             CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.Flags), 1);
133         break;

135     case 4: /* Range Type */

137         RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerOp, 1, 0)
138         RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_RANGETYPE,
139             CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.SpecificFlags), 1);
140         break;

142     case 5: /* Address Granularity */

144         Descriptor->ExtAddress64.Granularity = InitializerOp->Asl.Value.Integer;
145         RsCreateQwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
146             CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.Granularity), 1);
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_RESDISC_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_RESDISC_OFFSET (ExtAddress64.Maximum));
163         MaxOp = InitializerOp;
164         break;

166     case 8: /* Translation Offset */

168         Descriptor->ExtAddress64.TranslationOffset = InitializerOp->Asl.Value.Integer;
169         RsCreateQwordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
170             CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.TranslationOffset), 1);
171         break;

173     case 9: /* Address Length */

175         Descriptor->ExtAddress64.AddressLength = InitializerOp->Asl.Value.Integer;
176         RsCreateQwordField (InitializerOp, ACPI_RESTAG_LENGTH,
177             CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.AddressLength), 1);
178         LengthOp = InitializerOp;
179         break;

181     case 10: /* Type-Specific Attributes */

183         Descriptor->ExtAddress64.TypeSpecific = InitializerOp->Asl.Value.Integer;
184         RsCreateQwordField (InitializerOp, ACPI_RESTAG_TYPESPECIFICATTRIBUTE,
185             CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.TypeSpecificAttributes), 1);
186         break;

188     case 11: /* ResourceTag */

190         UtAttachNamepathToOwner (Op, InitializerOp);
191         break;

```

```

193     case 12: /* Type */

195         RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerOp, 1, 0)
196         RsCreateBitField (InitializerOp, ACPI_RESTAG_TYPE,
197             CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.SpecificFlags), 1);
198         break;

200     case 13: /* Translation Type */

202         RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerOp, 1, 0)
203         RsCreateBitField (InitializerOp, ACPI_RESTAG_TRANSTYPE,
204             CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.SpecificFlags), 1);
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) + StringLength;
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);

264     Rnode = RsAllocateResourceNode (
265         sizeof (AML_RESOURCE_EXTENDED_ADDRESS64) + 1 + StringLength);

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

272     Descriptor->ExtAddress64.ResourceLength = (UINT16)
273         (sizeof (AML_RESOURCE_EXTENDED_ADDRESS64) -
274          sizeof (AML_RESOURCE_LARGE_HEADER));

276     /* Process all child initialization nodes */

278     for (i = 0; InitializerOp; i++)
279     {
280         switch (i)
281         {
282             case 0: /* Resource Usage */

284                 RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 0, 1)
285                 break;

287             case 1: /* DecodeType */

289                 RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 1, 0)
290                 RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
291                 CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.Flags), 1);
292                 break;

294             case 2: /* MinType */

296                 RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 2, 0)
297                 RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
298                 CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.Flags), 2);
299                 break;

301             case 3: /* MaxType */

303                 RsSetFlagBits (&Descriptor->ExtAddress64.Flags, InitializerOp, 3, 0)
304                 RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
305                 CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.Flags), 3);
306                 break;

308             case 4: /* Memory Type */

310                 RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerOp
311                 RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_MEMTYPE,
312                 CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.SpecificFla
313                 break;

315             case 5: /* Read/Write Type */

317                 RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerOp
318                 RsCreateBitField (InitializerOp, ACPI_RESTAG_READWRITETYPE,
319                 CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.SpecificFla
320                 break;

322             case 6: /* Address Granularity */

```

```

324         Descriptor->ExtAddress64.Granularity = InitializerOp->Asl.Value.Inte
325         RsCreateQwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
326         CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.Granularity
327         GranOp = InitializerOp;
328         break;

330     case 7: /* Min Address */

332         Descriptor->ExtAddress64.Minimum = InitializerOp->Asl.Value.Integer;
333         RsCreateQwordField (InitializerOp, ACPI_RESTAG_MINADDR,
334         CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.Minimum));
335         MinOp = InitializerOp;
336         break;

338     case 8: /* Max Address */

340         Descriptor->ExtAddress64.Maximum = InitializerOp->Asl.Value.Integer;
341         RsCreateQwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
342         CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.Maximum));
343         MaxOp = InitializerOp;
344         break;

346     case 9: /* Translation Offset */

348         Descriptor->ExtAddress64.TranslationOffset = InitializerOp->Asl.Valu
349         RsCreateQwordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
350         CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.Translation
351         break;

353     case 10: /* Address Length */

355         Descriptor->ExtAddress64.AddressLength = InitializerOp->Asl.Value.In
356         RsCreateQwordField (InitializerOp, ACPI_RESTAG_LENGTH,
357         CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.AddressLeng
358         LengthOp = InitializerOp;
359         break;

361     case 11: /* Type-Specific Attributes */

363         Descriptor->ExtAddress64.TypeSpecific = InitializerOp->Asl.Value.Int
364         RsCreateQwordField (InitializerOp, ACPI_RESTAG_TYPESPECIFICATTRIBUTE
365         CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.TypeSpecifi
366         break;

368     case 12: /* ResourceTag */

370         UtAttachNamepathToOwner (Op, InitializerOp);
371         break;

374     case 13: /* Address Range */

376         RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerOp
377         RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_MEMATTRIBUTES,
378         CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.SpecificFla
379         break;

381     case 14: /* Type */

383         RsSetFlagBits (&Descriptor->ExtAddress64.SpecificFlags, InitializerOp
384         RsCreateBitField (InitializerOp, ACPI_RESTAG_TYPE,
385         CurrentByteOffset + ASL_RESDISC_OFFSET (ExtAddress64.SpecificFla
386         break;

388     default:

```

```

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) + StringLength;
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_ADDRESS64;
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_RESDISC_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_RESDISC_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_RESDISC_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.Integer;
503         RsCreateQwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
504             CurrentByteOffset + ASL_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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 }
```

new/usr/src/common/acpica/compiler/aslrestype2q.c

1

```
*****
25533 Fri Jan 17 07:29:39 2014
new/usr/src/common/acpica/compiler/aslrestype2q.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslrestype2q - Large QWord address resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"

48 #define _COMPONENT          ACPI_COMPILER
49 #define ACPI_MODULE_NAME    ("aslrestype2q")

51 /*
52 * This module contains the QWord (64-bit) address space descriptors:
53 *
54 * QWordIO
55 * QWordMemory
56 * QWordSpace
57 */
59 /*****
```

new/usr/src/common/acpica/compiler/aslrestype2q.c

2

```
60 *
61 * FUNCTION:      RsDoQwordIoDescriptor
62 *
63 * PARAMETERS:    Op                - Parent resource descriptor parse node
64 *                CurrentByteOffset - Offset into the resource template AML
65 *                                     buffer (to track references to the desc)
66 *
67 * RETURN:        Completed resource node
68 *
69 * DESCRIPTION:    Construct a long "QwordIO" descriptor
70 *
71 *****/

73 ASL_RESOURCE_NODE *
74 RsDoQwordIoDescriptor (
75     ACPI_PARSE_OBJECT *Op,
76     UINT32              CurrentByteOffset)
77 {
78     AML_RESOURCE *Descriptor;
79     ACPI_PARSE_OBJECT *InitializerOp;
80     ACPI_PARSE_OBJECT *MinOp = NULL;
81     ACPI_PARSE_OBJECT *MaxOp = NULL;
82     ACPI_PARSE_OBJECT *LengthOp = NULL;
83     ACPI_PARSE_OBJECT *GranOp = NULL;
84     ASL_RESOURCE_NODE *Rnode;
85     UINT8 *OptionalFields;
86     UINT16 StringLength = 0;
87     UINT32 OptionIndex = 0;
88     UINT32 i;
89     BOOLEAN ResSourceIndex = FALSE;

92     InitializerOp = Op->Asl.Child;
93     StringLength = RsGetStringDataLength (InitializerOp);

95     Rnode = RsAllocateResourceNode (
96         sizeof (AML_RESOURCE_ADDRESS64) + 1 + StringLength);

98     Descriptor = Rnode->Buffer;
99     Descriptor->Address64.DescriptorType = ACPI_RESOURCE_NAME_ADDRESS64;
100     Descriptor->Address64.ResourceType = ACPI_ADDRESS_TYPE_IO_RANGE;

102     /*
103      * Initial descriptor length -- may be enlarged if there are
104      * optional fields present
105      */
106     OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS64);
107     Descriptor->Address64.ResourceLength = (UINT16)
108         (sizeof (AML_RESOURCE_ADDRESS64) -
109          sizeof (AML_RESOURCE_LARGE_HEADER));

111     /* Process all child initialization nodes */

113     for (i = 0; InitializerOp; i++)
114     {
115         switch (i)
116         {
117             case 0: /* Resource Usage */

119                 RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 0, 1);
120                 break;

122             case 1: /* MinType */

124                 RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 2, 0);
125                 RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
```

```

126     CurrentByteOffset + ASL_RESDDESC_OFFSET (Address64.Flags), 2);
127     break;
129     case 2: /* MaxType */
131         RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 3, 0);
132         RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
133             CurrentByteOffset + ASL_RESDDESC_OFFSET (Address64.Flags), 3);
134         break;
136     case 3: /* DecodeType */
138         RsSetFlagBits (&Descriptor->Address64.Flags, InitializerOp, 1, 0);
139         RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
140             CurrentByteOffset + ASL_RESDDESC_OFFSET (Address64.Flags), 1);
141         break;
143     case 4: /* Range Type */
145         RsSetFlagBits (&Descriptor->Address64.SpecificFlags, InitializerOp,
146             RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_RANGETYPE,
147                 CurrentByteOffset + ASL_RESDDESC_OFFSET (Address64.SpecificFlags)
148                 break;
150     case 5: /* Address Granularity */
152         Descriptor->Address64.Granularity = InitializerOp->Asl.Value.Integer
153         RsCreateQwordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
154             CurrentByteOffset + ASL_RESDDESC_OFFSET (Address64.Granularity));
155         GranOp = InitializerOp;
156         break;
158     case 6: /* Address Min */
160         Descriptor->Address64.Minimum = InitializerOp->Asl.Value.Integer;
161         RsCreateQwordField (InitializerOp, ACPI_RESTAG_MINADDR,
162             CurrentByteOffset + ASL_RESDDESC_OFFSET (Address64.Minimum));
163         MinOp = InitializerOp;
164         break;
166     case 7: /* Address Max */
168         Descriptor->Address64.Maximum = InitializerOp->Asl.Value.Integer;
169         RsCreateQwordField (InitializerOp, ACPI_RESTAG_MAXADDR,
170             CurrentByteOffset + ASL_RESDDESC_OFFSET (Address64.Maximum));
171         MaxOp = InitializerOp;
172         break;
174     case 8: /* Translation Offset */
176         Descriptor->Address64.TranslationOffset = InitializerOp->Asl.Value.I
177         RsCreateByteField (InitializerOp, ACPI_RESTAG_TRANSLATION,
178             CurrentByteOffset + ASL_RESDDESC_OFFSET (Address64.TranslationOff
179             break;
181     case 9: /* Address Length */
183         Descriptor->Address64.AddressLength = InitializerOp->Asl.Value.Integ
184         RsCreateQwordField (InitializerOp, ACPI_RESTAG_LENGTH,
185             CurrentByteOffset + ASL_RESDDESC_OFFSET (Address64.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->Address64.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->Address64.ResourceLength = (UINT16)
208                     (Descriptor->Address64.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->Address64.SpecificFlags, InitializerOp,
245             RsCreateBitField (InitializerOp, ACPI_RESTAG_TYPE,
246                 CurrentByteOffset + ASL_RESDDESC_OFFSET (Address64.SpecificFlags)
247             break;
249     case 14: /* Translation Type */
251         RsSetFlagBits (&Descriptor->Address64.SpecificFlags, InitializerOp,
252             RsCreateBitField (InitializerOp, ACPI_RESTAG_TRANSTYPE,
253                 CurrentByteOffset + ASL_RESDDESC_OFFSET (Address64.SpecificFlags)
254             break;
256     default:

```

```

258     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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_OFFSET (Address64.SpecificFlags)
477             break;

479     case 15: /* Type */

481         RsSetFlagBits (&Descriptor->Address64.SpecificFlags, InitializerOp,
482             RsCreateBitField (InitializerOp, ACPI_RESTAG_TYPE,
483                 CurrentByteOffset + ASL_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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;

655     case 12: /* Resource [Optional Field - STRING] */

656         if ((InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
657             (InitializerOp->Asl.Value.String))
658         {
659             if (StringLength)
660             {
661                 Descriptor->Address64.ResourceLength = (UINT16)
662                     (Descriptor->Address64.ResourceLength + StringLength);

663                 strcpy ((char *)
664                     &OptionalFields[OptionIndex],
665                     InitializerOp->Asl.Value.String);

666                 /* ResourceSourceIndex must also be valid */

667                 if (!ResourceSourceIndex)
668                 {
669                     AslError (ASL_ERROR, ASL_MSG_RESOURCE_INDEX,
670                         InitializerOp, NULL);
671                 }
672             }
673         }

674     #if 0
675     /*
676      * Not a valid ResourceSource, ResourceSourceIndex must also
677      * be invalid
678      */
679     else if (ResourceSourceIndex)
680     {
681         AslError (ASL_ERROR, ASL_MSG_RESOURCE_SOURCE,
682             InitializerOp, NULL);
683     }
684     #endif

685     break;

686     case 13: /* ResourceTag */

687         UtAttachNamepathToOwner (Op, InitializerOp);
688         break;

689     default:

690         AslError (ASL_ERROR, ASL_MSG_RESOURCE_LIST, InitializerOp, NULL);
691         break;

692     }

693     InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
694 }

695 /* Validate the Min/Max/Len/Gran values */

696 RsLargeAddressCheck (
697     Descriptor->Address64.Minimum,
698     Descriptor->Address64.Maximum,
699     Descriptor->Address64.AddressLength,
700     Descriptor->Address64.Granularity,
701     Descriptor->Address64.Flags,
702     MinOp, MaxOp, LengthOp, GranOp, Op);

703 Rnode->BufferLength = sizeof (AML_RESOURCE_ADDRESS64) +
704     OptionIndex + StringLength;
705 return (Rnode);

```

```

720 }

```

new/usr/src/common/acpica/compiler/aslrestype2s.c

1

```
*****
42048 Fri Jan 17 07:29:40 2014
new/usr/src/common/acpica/compiler/aslrestype2s.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: aslrestype2s - Serial Large resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"
47 #include "amlcode.h"

49 #define _COMPONENT          ACPI_COMPILER
50 #define ACPI_MODULE_NAME    ("aslrestype2s")

53 static UINT16
54 RsGetBufferDataLength (
55     ACPI_PARSE_OBJECT      *InitializerOp);

57 static UINT16
58 RsGetInterruptDataLength (
59     ACPI_PARSE_OBJECT      *InitializerOp);
```

new/usr/src/common/acpica/compiler/aslrestype2s.c

2

```
61 static BOOLEAN
62 RsGetVendorData (
63     ACPI_PARSE_OBJECT      *InitializerOp,
64     UINT8                  *VendorData,
65     ACPI_SIZE               DescriptorOffset);

67 /*
68 * This module contains descriptors for serial buses and GPIO:
69 *
70 * GpioInt
71 * GpioIo
72 * I2cSerialBus
73 * SpiSerialBus
74 * UartSerialBus
75 */

78 /*****
79 *
80 * FUNCTION:      RsGetBufferDataLength
81 *
82 * PARAMETERS:    InitializerOp      - Current parse op, start of the resource
83 *                                     descriptor
84 *
85 * RETURN:        Length of the data buffer
86 *
87 * DESCRIPTION:   Get the length of a RawDataBuffer, used for vendor data.
88 *
89 *****/

91 static UINT16
92 RsGetBufferDataLength (
93     ACPI_PARSE_OBJECT      *InitializerOp)
94 {
95     UINT16                  ExtraDataSize = 0;
96     ACPI_PARSE_OBJECT      *DataList;

99     /* Find the byte-initializer list */

101     while (InitializerOp)
102     {
103         if (InitializerOp->Asl.ParseOpcode == PARSEOP_DATABUFFER)
104         {
105             /* First child is the optional length (ignore it here) */

107             DataList = InitializerOp->Asl.Child;
108             DataList = ASL_GET_PEER_NODE (DataList);

110             /* Count the data items (each one is a byte of data) */

112             while (DataList)
113             {
114                 ExtraDataSize++;
115                 DataList = ASL_GET_PEER_NODE (DataList);
116             }

118             return (ExtraDataSize);
119         }

121         InitializerOp = ASL_GET_PEER_NODE (InitializerOp);
122     }

124     return (ExtraDataSize);
125 }
```

```

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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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 */
    if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
    {
397         Descriptor->Gpio.ResSourceIndex = (UINT8) InitializerOp->Asl.Val
398     }
399 }
400 break;
401

403 case 7: /* Resource Usage (consumer/producer) */
    RsSetFlagBits16 (&Descriptor->Gpio.Flags, InitializerOp, 0, 1);
405 break;
406

408 case 8: /* Resource Tag (Descriptor Name) */
    UtAttachNamepathToOwner (Op, InitializerOp);
410 break;
411

413 case 9: /* Vendor Data (Optional - Buffer of BYTES) (_VEN) */
    /*
415     * Always set the VendorOffset even if there is no Vendor Data.
416     * This field is required in order to calculate the length
417     * of the ResourceSource at runtime.
418     */
419     Descriptor->Gpio.VendorOffset = (UINT16)
420         ACPI_PTR_DIFF (VendorData, Descriptor);
421

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 */
443     if (i == 10)
444     {
445         if (InitializerOp->Asl.ParseOpcode == PARSEOP_DEFAULT_ARG)
446         {
447             /* Must be at least one interrupt */
448
449             AslError (ASL_ERROR, ASL_MSG_EX_INTERRUPT_LIST_MIN,
450                 InitializerOp, NULL);
451         }
452
454         /* Check now for duplicates in list */
455         RsCheckListForDuplicates (InitializerOp);

```

```

458     /* Create a named field at the start of the list */
459     RsCreateDwordField (InitializerOp, ACPI_RESTAG_PIN,
460         CurrentByteOffset + Descriptor->Gpio.PinTableOffset);
461     }
462     break;
463 }
464

466     InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
467 }
468
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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_OFFSET (Gpio.IntFlags), 0, 2);
588             break;

```

```

590 case 5: /* ResSource [Optional Field - STRING] */
591     if (ResSourceLength)
592     {
593         /* Copy string to the descriptor */
594         strcpy (ResourceSource,
595             InitializerOp->Asl.Value.String);
596     }
597     break;

601 case 6: /* Resource Index */
602     if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
603     {
604         Descriptor->Gpio.ResSourceIndex = (UINT8) InitializerOp->Asl.Val
605     }
606     break;

609 case 7: /* Resource Usage (consumer/producer) */
610     RsSetFlagBits16 (&Descriptor->Gpio.Flags, InitializerOp, 0, 1);
611     break;

614 case 8: /* Resource Tag (Descriptor Name) */
615     UtAttachNamepathToOwner (Op, InitializerOp);
616     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_RESDISC_OFFSET (I2cSerialBus.SlaveAddress
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_RESDISC_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_RESDISC_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_RESDISC_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             break;
795         }

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 AML_RESOURCE_NODE *
840 RsDoSpiSerialBusDescriptor (
841     ACPI_PARSE_OBJECT *Op,
842     UINT32 CurrentByteOffset)
843 {
844     AML_RESOURCE *Descriptor;
845     ACPI_PARSE_OBJECT *InitializerOp;
846     AML_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_DEVICEPOLARITY,
901                     CurrentByteOffset + ASL_RESDISC_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_RESDISC_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_RESDISC_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_RESDISC_OFFSET (SpiSerialBus.DataBitLeng
923     break;

925     case 4: /* Slave Mode [Flag] (_SLV) */

927     RsSetFlagBits (&Descriptor->SpiSerialBus.Flags, InitializerOp, 0, 0)
928     RsCreateBitField (InitializerOp, ACPI_RESTAG_SLAVEMODE,
929     CurrentByteOffset + ASL_RESDISC_OFFSET (SpiSerialBus.Flags), 0);
930     break;

932     case 5: /* Connection Speed [DWORD] (_SPE) */

934     Descriptor->SpiSerialBus.ConnectionSpeed = (UINT32) InitializerOp->A
935     RsCreateDwordField (InitializerOp, ACPI_RESTAG_SPEED,
936     CurrentByteOffset + ASL_RESDISC_OFFSET (SpiSerialBus.Connections
937     break;

939     case 6: /* Clock Polarity [BYTE] (_POL) */

941     Descriptor->SpiSerialBus.ClockPolarity = (UINT8) InitializerOp->Asl.
942     RsCreateByteField (InitializerOp, ACPI_RESTAG_POLARITY,
943     CurrentByteOffset + ASL_RESDISC_OFFSET (SpiSerialBus.ClockPolari
944     break;

946     case 7: /* Clock Phase [BYTE] (_PHA) */

948     Descriptor->SpiSerialBus.ClockPhase = (UINT8) InitializerOp->Asl.Val
949     RsCreateByteField (InitializerOp, ACPI_RESTAG_PHASE,
950     CurrentByteOffset + ASL_RESDISC_OFFSET (SpiSerialBus.ClockPhase)
951     break;

953     case 8: /* ResSource [Optional Field - STRING] */

955     if (ResSourceLength)
956     {
957         /* Copy string to the descriptor */

959         strcpy (ResourceSource,
960         InitializerOp->Asl.Value.String);
961     }
962     break;

964     case 9: /* Resource Index */

966     if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
967     {
968         Descriptor->SpiSerialBus.ResSourceIndex = (UINT8) InitializerOp-
969     }
970     break;

972     case 10: /* Resource Usage (consumer/producer) */

974     RsSetFlagBits (&Descriptor->SpiSerialBus.Flags, InitializerOp, 1, 1)
975     break;

977     case 11: /* Resource Tag (Descriptor Name) */

979     UtAttachNamepathToOwner (Op, InitializerOp);
980     break;

982     case 12: /* Vendor Data (Optional - Buffer of BYTES) (_VEN) */

984     RsGetVendorData (InitializerOp, VendorData,

```

```

985     CurrentByteOffset + sizeof (AML_RESOURCE_SPI_SERIALBUS));
986     break;

988     default: /* Ignore any extra nodes */

990     break;
991 }

993     InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);
994 }

996     return (Rnode);
997 }

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 *****/

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     *ResourceSource = NULL;
1023     *VendorData = NULL;
1024     ResSourceLength;
1025     VendorLength;
1026     DescriptorSize;
1027     i;

1030     InitializerOp = Op->Asl.Child;

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);

1040     DescriptorSize = ACPI_AML_SIZE_LARGE (AML_RESOURCE_UART_SERIALBUS) +
1041     ResSourceLength + VendorLength;

1043     /* Allocate the local resource node and initialize */

1045     Rnode = RsAllocateResourceNode (DescriptorSize + sizeof (AML_RESOURCE_LARGE_

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;
1052 Descriptor->UartSerialBus.Type = AML_RESOURCE_UART_SERIALBUSTYPE;
1053 Descriptor->UartSerialBus.TypeDataLength = AML_RESOURCE_UART_MIN_DATA_LEN +

1055 /* Build pointers to optional areas */

1057 VendorData = ACPI_ADD_PTR (UINT8, Descriptor, sizeof (AML_RESOURCE_UART_SERI
1058 ResourceSource = ACPI_ADD_PTR (char, VendorData, VendorLength);

1060 DbgPrint (ASL_DEBUG_OUTPUT,
1061 "%16s - Actual: %.2X, Base: %.2X, ResLen: %.2X, VendLen: %.2X, TypLen: %
1062 "UartSerialBus", Descriptor->UartSerialBus.ResourceLength,
1063 (UINT16) sizeof (AML_RESOURCE_UART_SERIALBUS), ResSourceLength,
1064 VendorLength, Descriptor->UartSerialBus.TypeDataLength);

1066 /* Process all child initialization nodes */

1068 for (i = 0; InitializerOp; i++)
1069 {
1070     switch (i)
1071     {
1072     case 0: /* Connection Speed (Baud Rate) [DWORD] (_SPE) */

1074         Descriptor->UartSerialBus.DefaultBaudRate = (UINT32) InitializerOp->
1075         RsCreateDwordField (InitializerOp, ACPI_RESTAG_SPEED,
1076             CurrentByteOffset + ASL_RESDISC_OFFSET (UartSerialBus.DefaultBau
1077             break;

1079     case 1: /* Bits Per Byte [Flags] (_LEN) */

1081         RsSetFlagBits16 (&Descriptor->UartSerialBus.TypeSpecificFlags, Initi
1082         RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_LENGTH,
1083             CurrentByteOffset + ASL_RESDISC_OFFSET (UartSerialBus.TypeSpecif
1084             break;

1086     case 2: /* Stop Bits [Flags] (_STB) */

1088         RsSetFlagBits16 (&Descriptor->UartSerialBus.TypeSpecificFlags, Initi
1089         RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_STOPBITS,
1090             CurrentByteOffset + ASL_RESDISC_OFFSET (UartSerialBus.TypeSpecif
1091             break;

1093     case 3: /* Lines In Use [BYTE] (_LIN) */

1095         Descriptor->UartSerialBus.LinesEnabled = (UINT8) InitializerOp->Asl.
1096         RsCreateByteField (InitializerOp, ACPI_RESTAG_LINE,
1097             CurrentByteOffset + ASL_RESDISC_OFFSET (UartSerialBus.LinesEnabl
1098             break;

1100     case 4: /* Endianness [Flag] (_END) */

1102         RsSetFlagBits16 (&Descriptor->UartSerialBus.TypeSpecificFlags, Initi
1103         RsCreateBitField (InitializerOp, ACPI_RESTAG_ENDIANNES,
1104             CurrentByteOffset + ASL_RESDISC_OFFSET (UartSerialBus.TypeSpecif
1105             break;

1107     case 5: /* Parity [BYTE] (_PAR) */

1109         Descriptor->UartSerialBus.Parity = (UINT8) InitializerOp->Asl.Value.
1110         RsCreateByteField (InitializerOp, ACPI_RESTAG_PARITY,
1111             CurrentByteOffset + ASL_RESDISC_OFFSET (UartSerialBus.Parity));
1112         break;

1114     case 6: /* Flow Control [Flags] (_FLC) */

1116         RsSetFlagBits16 (&Descriptor->UartSerialBus.TypeSpecificFlags, Initi

```

```

1117 RsCreateMultiBitField (InitializerOp, ACPI_RESTAG_FLOWCONTROL,
1118     CurrentByteOffset + ASL_RESDISC_OFFSET (UartSerialBus.TypeSpecif
1119     break;

1121     case 7: /* Rx Buffer Size [WORD] (_RXL) */

1123         Descriptor->UartSerialBus.RxFifoSize = (UINT16) InitializerOp->Asl.V
1124         RsCreateWordField (InitializerOp, ACPI_RESTAG_LENGTH_RX,
1125             CurrentByteOffset + ASL_RESDISC_OFFSET (UartSerialBus.RxFifoSize
1126             break;

1128     case 8: /* Tx Buffer Size [WORD] (_TXL) */

1130         Descriptor->UartSerialBus.TxFifoSize = (UINT16) InitializerOp->Asl.V
1131         RsCreateWordField (InitializerOp, ACPI_RESTAG_LENGTH_TX,
1132             CurrentByteOffset + ASL_RESDISC_OFFSET (UartSerialBus.TxFifoSize
1133             break;

1135     case 9: /* ResSource [Optional Field - STRING] */

1137         if (ResSourceLength)
1138         {
1139             /* Copy string to the descriptor */

1141             strcpy (ResourceSource,
1142                 InitializerOp->Asl.Value.String);
1143         }
1144         break;

1146     case 10: /* Resource Index */

1148         if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
1149         {
1150             Descriptor->UartSerialBus.ResSourceIndex = (UINT8) InitializerOp
1151         }
1152         break;

1154     case 11: /* Resource Usage (consumer/producer) */

1156         RsSetFlagBits (&Descriptor->UartSerialBus.Flags, InitializerOp, 1, 1

1158         /*
1159         * Slave Mode [Flag] (_SLV)
1160         *
1161         * Note: There is no SlaveMode argument to the UartSerialBus macro,
1162         * we add this name anyway to allow the flag to be set by ASL in the
1163         * rare case where there is a slave mode associated with the UART.
1164         */
1165         RsCreateBitField (InitializerOp, ACPI_RESTAG_SLAVEMODE,
1166             CurrentByteOffset + ASL_RESDISC_OFFSET (UartSerialBus.Flags), 0)
1167         break;

1169     case 12: /* Resource Tag (Descriptor Name) */

1171         UtAttachNamepathToOwner (Op, InitializerOp);
1172         break;

1174     case 13: /* Vendor Data (Optional - Buffer of BYTES) (_VEN) */

1176         RsGetVendorData (InitializerOp, VendorData,
1177             CurrentByteOffset + sizeof (AML_RESOURCE_UART_SERIALBUS));
1178         break;

1180     default: /* Ignore any extra nodes */

1182         break;

```

```
1183     }  
1185     InitializerOp = RsCompleteNodeAndGetNext (InitializerOp);  
1186 }  
1188     return (Rnode);  
1189 }
```

new/usr/src/common/acpica/compiler/aslrestype2w.c

1

```
*****
24719 Fri Jan 17 07:29:40 2014
new/usr/src/common/acpica/compiler/aslrestype2w.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslrestype2w - Large Word address resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"

48 #define _COMPONENT          ACPI_COMPILER
49 #define ACPI_MODULE_NAME    ("aslrestype2w")

51 /*
52 * This module contains the Word (16-bit) address space descriptors:
53 *
54 * WordIO
55 * WordMemory
56 * WordSpace
57 */
59 /*****
```

new/usr/src/common/acpica/compiler/aslrestype2w.c

2

```
60 *
61 * FUNCTION:      RsDoWordIoDescriptor
62 *
63 * PARAMETERS:    Op                - Parent resource descriptor parse node
64 *                CurrentByteOffset - Offset into the resource template AML
65 *                                     buffer (to track references to the desc)
66 *
67 * RETURN:        Completed resource node
68 *
69 * DESCRIPTION:    Construct a long "WordIO" descriptor
70 *
71 *****/

73 ASL_RESOURCE_NODE *
74 RsDoWordIoDescriptor (
75     ACPI_PARSE_OBJECT *Op,
76     UINT32             CurrentByteOffset)
77 {
78     AML_RESOURCE *Descriptor;
79     ACPI_PARSE_OBJECT *InitializerOp;
80     ACPI_PARSE_OBJECT *MinOp = NULL;
81     ACPI_PARSE_OBJECT *MaxOp = NULL;
82     ACPI_PARSE_OBJECT *LengthOp = NULL;
83     ACPI_PARSE_OBJECT *GranOp = NULL;
84     ASL_RESOURCE_NODE *Rnode;
85     UINT8 *OptionalFields;
86     UINT16 StringLength = 0;
87     UINT32 OptionIndex = 0;
88     UINT32 i;
89     BOOLEAN ResSourceIndex = FALSE;

92     InitializerOp = Op->Asl.Child;
93     StringLength = RsGetStringDataLength (InitializerOp);

95     Rnode = RsAllocateResourceNode (
96         sizeof (AML_RESOURCE_ADDRESS16) + 1 + StringLength);

98     Descriptor = Rnode->Buffer;
99     Descriptor->Address16.DescriptorType = ACPI_RESOURCE_NAME_ADDRESS16;
100     Descriptor->Address16.ResourceType = ACPI_ADDRESS_TYPE_IO_RANGE;

102     /*
103      * Initial descriptor length -- may be enlarged if there are
104      * optional fields present
105      */
106     OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS16);
107     Descriptor->Address16.ResourceLength = (UINT16)
108         (sizeof (AML_RESOURCE_ADDRESS16) -
109          sizeof (AML_RESOURCE_LARGE_HEADER));

111     /* Process all child initialization nodes */

113     for (i = 0; InitializerOp; i++)
114     {
115         switch (i)
116         {
117             case 0: /* Resource Usage */

119                 RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 0, 1);
120                 break;

122             case 1: /* MinType */

124                 RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 2, 0);
125                 RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
```

```

126     CurrentByteOffset + ASL_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_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_RESDDESC_OFFSET (Address16.SpecificFlags)
254             break;
256     default:

```

```

258     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     (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);

275     Rnode->BufferLength = sizeof (AML_RESOURCE_ADDRESS16) +
276         OptionIndex + StringLength;
277     return (Rnode);
278 }

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  *****/

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;

314     InitializerOp = Op->Asl.Child;
315     StringLength = RsGetStringDataLength (InitializerOp);

317     Rnode = RsAllocateResourceNode (
318         sizeof (AML_RESOURCE_ADDRESS16) + 1 + StringLength);

320     Descriptor = Rnode->Buffer;
321     Descriptor->Address16.DescriptorType = ACPI_RESOURCE_NAME_ADDRESS16;
322     Descriptor->Address16.ResourceType = ACPI_ADDRESS_TYPE_BUS_NUMBER_RANGE;

```

```

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));

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->Address16.Flags, InitializerOp, 0, 1);
342                 break;

344             case 1: /* MinType */

346                 RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 2, 0);
347                 RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
348                     CurrentByteOffset + ASL_RESDISC_OFFSET (Address16.Flags), 2);
349                 break;

351             case 2: /* MaxType */

353                 RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 3, 0);
354                 RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
355                     CurrentByteOffset + ASL_RESDISC_OFFSET (Address16.Flags), 3);
356                 break;

358             case 3: /* DecodeType */

360                 RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 1, 0);
361                 RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
362                     CurrentByteOffset + ASL_RESDISC_OFFSET (Address16.Flags), 1);
363                 break;

365             case 4: /* Address Granularity */

367                 Descriptor->Address16.Granularity =
368                     (UINT16) InitializerOp->Asl.Value.Integer;
369                 RsCreateWordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
370                     CurrentByteOffset + ASL_RESDISC_OFFSET (Address16.Granularity));
371                 GranOp = InitializerOp;
372                 break;

374             case 5: /* Min Address */

376                 Descriptor->Address16.Minimum =
377                     (UINT16) InitializerOp->Asl.Value.Integer;
378                 RsCreateWordField (InitializerOp, ACPI_RESTAG_MINADDR,
379                     CurrentByteOffset + ASL_RESDISC_OFFSET (Address16.Minimum));
380                 MinOp = InitializerOp;
381                 break;

383             case 6: /* Max Address */

385                 Descriptor->Address16.Maximum =
386                     (UINT16) InitializerOp->Asl.Value.Integer;
387                 RsCreateWordField (InitializerOp, ACPI_RESTAG_MAXADDR,
388                     CurrentByteOffset + ASL_RESDISC_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_RESDISC_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_RESDISC_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);

```

```

523 Rnode = RsAllocateResourceNode (
524     sizeof (AML_RESOURCE_ADDRESS16) + 1 + StringLength);

526 Descriptor = Rnode->Buffer;
527 Descriptor->Address16.DescriptorType = ACPI_RESOURCE_NAME_ADDRESS16;

529 /*
530  * Initial descriptor length -- may be enlarged if there are
531  * optional fields present
532  */
533 OptionalFields = ((UINT8 *) Descriptor) + sizeof (AML_RESOURCE_ADDRESS16);
534 Descriptor->Address16.ResourceLength = (UINT16)
535     (sizeof (AML_RESOURCE_ADDRESS16) -
536      sizeof (AML_RESOURCE_LARGE_HEADER));

538 /* Process all child initialization nodes */

540 for (i = 0; InitializerOp; i++)
541 {
542     switch (i)
543     {
544     case 0: /* Resource Type */

546         Descriptor->Address16.ResourceType =
547             (UINT8) InitializerOp->Asl.Value.Integer;
548         break;

550     case 1: /* Resource Usage */

552         RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 0, 1);
553         break;

555     case 2: /* DecodeType */

557         RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 1, 0);
558         RsCreateBitField (InitializerOp, ACPI_RESTAG_DECODE,
559             CurrentByteOffset + ASL_RESDISC_OFFSET (Address16.Flags), 1);
560         break;

562     case 3: /* MinType */

564         RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 2, 0);
565         RsCreateBitField (InitializerOp, ACPI_RESTAG_MINTYPE,
566             CurrentByteOffset + ASL_RESDISC_OFFSET (Address16.Flags), 2);
567         break;

569     case 4: /* MaxType */

571         RsSetFlagBits (&Descriptor->Address16.Flags, InitializerOp, 3, 0);
572         RsCreateBitField (InitializerOp, ACPI_RESTAG_MAXTYPE,
573             CurrentByteOffset + ASL_RESDISC_OFFSET (Address16.Flags), 3);
574         break;

576     case 5: /* Type-Specific flags */

578         Descriptor->Address16.SpecificFlags =
579             (UINT8) InitializerOp->Asl.Value.Integer;
580         break;

582     case 6: /* Address Granularity */

584         Descriptor->Address16.Granularity =
585             (UINT16) InitializerOp->Asl.Value.Integer;
586         RsCreateWordField (InitializerOp, ACPI_RESTAG_GRANULARITY,
587             CurrentByteOffset + ASL_RESDISC_OFFSET (Address16.Granularity));

```

```

588     GranOp = InitializerOp;
589     break;

591     case 7: /* Min Address */

593         Descriptor->Address16.Minimum =
594             (UINT16) InitializerOp->Asl.Value.Integer;
595         RsCreateWordField (InitializerOp, ACPI_RESTAG_MINADDR,
596             CurrentByteOffset + ASL_RESDISC_OFFSET (Address16.Minimum));
597         MinOp = InitializerOp;
598         break;

600     case 8: /* Max Address */

602         Descriptor->Address16.Maximum =
603             (UINT16) InitializerOp->Asl.Value.Integer;
604         RsCreateWordField (InitializerOp, ACPI_RESTAG_MAXADDR,
605             CurrentByteOffset + ASL_RESDISC_OFFSET (Address16.Maximum));
606         MaxOp = InitializerOp;
607         break;

609     case 9: /* Translation Offset */

611         Descriptor->Address16.TranslationOffset =
612             (UINT16) InitializerOp->Asl.Value.Integer;
613         RsCreateWordField (InitializerOp, ACPI_RESTAG_TRANSLATION,
614             CurrentByteOffset + ASL_RESDISC_OFFSET (Address16.TranslationOff
615             break;

617     case 10: /* Address Length */

619         Descriptor->Address16.AddressLength =
620             (UINT16) InitializerOp->Asl.Value.Integer;
621         RsCreateWordField (InitializerOp, ACPI_RESTAG_LENGTH,
622             CurrentByteOffset + ASL_RESDISC_OFFSET (Address16.AddressLength)
623         LengthOp = InitializerOp;
624         break;

626     case 11: /* ResSourceIndex [Optional Field - BYTE] */

628         if (InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG)
629         {
630             OptionalFields[0] = (UINT8) InitializerOp->Asl.Value.Integer;
631             OptionIndex++;
632             Descriptor->Address16.ResourceLength++;
633             ResSourceIndex = TRUE;
634         }
635         break;

637     case 12: /* ResSource [Optional Field - STRING] */

639         if ((InitializerOp->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG) &&
640             (InitializerOp->Asl.Value.String))
641         {
642             if (StringLength)
643             {
644                 Descriptor->Address16.ResourceLength = (UINT16)
645                     (Descriptor->Address16.ResourceLength + StringLength);

647                 strcpy ((char *)
648                     &OptionalFields[OptionIndex],
649                     InitializerOp->Asl.Value.String);

651                 /* ResourceSourceIndex must also be valid */

653                 if (!ResSourceIndex)

```

```
654         {
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 }
```

new/usr/src/common/acpica/compiler/aslstartup.c

1

```
*****
13219 Fri Jan 17 07:29:40 2014
new/usr/src/common/acpica/compiler/aslstartup.c
update to acpica-unix2-20140114
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 - 2014, 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 MERCHANTABILITY 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 (
```

new/usr/src/common/acpica/compiler/aslstartup.c

2

```
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;
125 }
126 }

129 /*****
130 *
131 * FUNCTION:     AslDetectSourceFileType
132 *
133 * PARAMETERS:  Info          - Name/Handle for the file (must be open)
134 *
135 * RETURN:      File Type
136 *
137 * DESCRIPTION: Determine the type of the input file. Either binary (contains
138 *              non-ASCII characters), ASL file, or an ACPI Data Table file.
139 *
140 *****/

142 static UINT8
143 AslDetectSourceFileType (
144     ASL_FILE_INFO      *Info)
145 {
146     char                *FileChar;
147     UINT8               Type;
148     ACPI_STATUS         Status;

151     /* Check for a valid binary ACPI table */

153     Status = FlCheckForAcpiTable (Info->Handle);
154     if (ACPI_SUCCESS (Status))
155     {
156         Type = ASL_INPUT_TYPE_ACPI_TABLE;
157         goto Cleanup;
158     }

160     /* Check for 100% ASCII source file (comments are ignored) */

162     Status = FlCheckForAscii (Info->Handle, Info->Filename, TRUE);
163     if (ACPI_FAILURE (Status))
164     {
165         printf ("Non-ascii input file - %s\n", Info->Filename);

167         if (!Gbl_IgnoreErrors)
168         {
169             Type = ASL_INPUT_TYPE_BINARY;
170             goto Cleanup;
171         }
172     }

174     /*
175     * File is ASCII. Determine if this is an ASL file or an ACPI data
176     * table file.
177     */
178     while (fgets (Gbl_CurrentLineBuffer, Gbl_LineBufferSize, Info->Handle))
179     {
180         /* Uppercase the buffer for caseless compare */

182         FileChar = Gbl_CurrentLineBuffer;
183         while (*FileChar)
184         {
185             *FileChar = (char) toupper ((int) *FileChar);
186             FileChar++;
187         }

189         /* Presence of "DefinitionBlock" indicates actual ASL code */

```

```

191     if (strstr (Gbl_CurrentLineBuffer, "DEFINITIONBLOCK"))
192     {
193         /* Appears to be an ASL file */

195         Type = ASL_INPUT_TYPE_ASCII_ASL;
196         goto Cleanup;
197     }
198 }

200 /* Not an ASL source file, default to a data table source file */

202     Type = ASL_INPUT_TYPE_ASCII_DATA;

204 Cleanup:

206     /* Must seek back to the start of the file */

208     fseek (Info->Handle, 0, SEEK_SET);
209     return (Type);
210 }

213 /*****
214 *
215 * FUNCTION:     AslDoDisassembly
216 *
217 * PARAMETERS:  None
218 *
219 * RETURN:      Status
220 *
221 * DESCRIPTION: Initiate AML file disassembly. Uses ACPICA subsystem to build
222 *              namespace.
223 *
224 *****/

226 static ACPI_STATUS
227 AslDoDisassembly (
228     void)
229 {
230     ACPI_STATUS         Status;

233     /* ACPICA subsystem initialization */

235     Status = AdInitialize ();
236     if (ACPI_FAILURE (Status))
237     {
238         return (Status);
239     }

241     Status = AcpiAllocateRootTable (4);
242     if (ACPI_FAILURE (Status))
243     {
244         AcpiOsPrintf ("Could not initialize ACPI Table Manager, %s\n",
245             AcpiFormatException (Status));
246         return (Status);
247     }

249     /* This is where the disassembly happens */

251     AcpiGbl_DbOpt_disasm = TRUE;
252     Status = AdAmlDisassemble (AslToFile,
253         Gbl_Files[ASL_FILE_INPUT].Filename, Gbl_OutputFilenamePrefix,
254         &Gbl_Files[ASL_FILE_INPUT].Filename, Gbl_GetAllTables);
255     if (ACPI_FAILURE (Status))

```

```

256 {
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_ASF:

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  }

```

new/usr/src/common/acpica/compiler/aslstubs.c

1

```
*****
5855 Fri Jan 17 07:29:40 2014
new/usr/src/common/acpica/compiler/aslstubs.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: aslstubs - Stubs used to link to Aml interpreter
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acdispat.h"
46 #include "actables.h"
47 #include "acevents.h"
48 #include "acinterp.h"
49 #include "acnamesp.h"

51 #define _COMPONENT          ACPI_COMPILER
52       ACPI_MODULE_NAME    ("aslstubs")

55 /*
56 * Stubs to simplify linkage to the ACPI CA core subsystem.
57 * Things like Events, Global Lock, etc. are not used
58 * by the compiler, so they are stubbed out here.
```

new/usr/src/common/acpica/compiler/aslstubs.c

2

```
59 */
60 ACPI_PHYSICAL_ADDRESS
61 AeLocalGetRootPointer (
62     void)
63 {
64     return (0);
65 }

67 void
68 AcpiNsExecModuleCodeList (
69     void)
70 {
71 }

73 ACPI_STATUS
74 AcpiHwReadPort (
75     ACPI_IO_ADDRESS      Address,
76     UINT32                *Value,
77     UINT32                Width)
78 {
79     return (AE_OK);
80 }

82 ACPI_STATUS
83 AcpiHwWritePort (
84     ACPI_IO_ADDRESS      Address,
85     UINT32                Value,
86     UINT32                Width)
87 {
88     return (AE_OK);
89 }

91 ACPI_STATUS
92 AcpiDsMethodError (
93     ACPI_STATUS           Status,
94     ACPI_WALK_STATE       *WalkState)
95 {
96     return (Status);
97 }

99 ACPI_STATUS
100 AcpiDsMethodDataGetValue (
101     UINT8                 Type,
102     UINT32                Index,
103     ACPI_WALK_STATE       *WalkState,
104     ACPI_OPERAND_OBJECT  **DestDesc)
105 {
106     return (AE_OK);
107 }

109 ACPI_STATUS
110 AcpiDsMethodDataGetNode (
111     UINT8                 Type,
112     UINT32                Index,
113     ACPI_WALK_STATE       *WalkState,
114     ACPI_NAMESPACE_NODE  **Node)
115 {
116     return (AE_OK);
117 }

119 ACPI_STATUS
120 AcpiDsStoreObjectToLocal (
121     UINT8                 Type,
122     UINT32                Index,
123     ACPI_OPERAND_OBJECT  *SrcDesc,
124     ACPI_WALK_STATE       *WalkState)
```



```

125 {
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 }

```

new/usr/src/common/acpica/compiler/aslsupport.1

1

\*\*\*\*\*

21957 Fri Jan 17 07:29:40 2014

new/usr/src/common/acpica/compiler/aslsupport.1

update to acpica-unix2-20140114

acpica-unix2-20130823

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: aslsupport.1 - Flex/lex scanner C support routines.
4 *
5 * NOTE: Included into aslcompile.1, not compiled by itself.
6 *
7 *
8 */
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 /* Configuration */
46
47 #define ASL_SPACES_PER_TAB      4
48
49 #define ASL_NORMAL_CHAR        0
50 #define ASL_ESCAPE_SEQUENCE    1
51 #define ASL_OCTAL_CONSTANT     2
52 #define ASL_HEX_CONSTANT       3
53
54
55 /* File node - used for "Include" operator file stack */
56
57 typedef struct asl_file_node
58 {
59     FILE          *File;
```

new/usr/src/common/acpica/compiler/aslsupport.1

2

```
61     UINT32          CurrentLineNumber;
62     YY_BUFFER_STATE State;
63     char            *Filename;
64     struct asl_file_node *Next;
65
66 } ASL_FILE_NODE;
67
68 /* File stack for the "Include" operator (NOT #include operator) */
69
70 ASL_FILE_NODE      *Gbl_IncludeFileStack = NULL;
71
72
73 /*****
74 *
75 * FUNCTION:      AslDoLineDirective
76 *
77 * PARAMETERS:    None. Uses input() to access current source code line
78 *
79 * RETURN:        Updates global line number and filename
80 *
81 * DESCRIPTION:   Handle #line directives emitted by the preprocessor.
82 *
83 * The #line directive is emitted by the preprocessor, and is used to
84 * pass through line numbers from the original source code file to the
85 * preprocessor output file (.i). This allows any compiler-generated
86 * error messages to be displayed with the correct line number.
87 *
88 *****/
89
90 static void
91 AslDoLineDirective (
92     void)
93 {
94     int             C;
95     char            *Token;
96     UINT32          LineNumber;
97     char            *Filename;
98     UINT32          i;
99
100     /* Eat the entire line that contains the #line directive */
101
102     Gbl_LineBufPtr = Gbl_CurrentLineBuffer;
103
104     while ((C = input()) != '\n' && C != EOF)
105     {
106         *Gbl_LineBufPtr = C;
107         Gbl_LineBufPtr++;
108     }
109     *Gbl_LineBufPtr = 0;
110
111     /* First argument is the actual line number */
112
113     Token = strtok (Gbl_CurrentLineBuffer, " ");
114     if (!Token)
115     {
116         goto ResetAndExit;
117     }
118
119     /* First argument is the line number */
120
121     LineNumber = (UINT32) UtDoConstant (Token);
122
123     /* Emit the appropriate number of newlines */
124
125     Gbl_CurrentColumn = 0;
```

```

127     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 */
261
262  Gbl_Files[ASL_FILE_INPUT].Filename = Filename;
263  Gbl_CurrentLineNumber = 1;
264  yyin = InputFile;
265 }
266
267 /*****
268 *
269 * FUNCTION:      AslResetCurrentLineBuffer
270 *
271 * PARAMETERS:   None
272 *
273 * RETURN:       None
274 *
275 * DESCRIPTION:  Reset the Line Buffer to zero, increment global line numbers.
276 *
277 * *****/
278
279 void
280 AslResetCurrentLineBuffer (
281     void)
282 {
283
284     if (Gbl_Files[ASL_FILE_SOURCE_OUTPUT].Handle)
285     {
286         FlWriteFile (ASL_FILE_SOURCE_OUTPUT, Gbl_CurrentLineBuffer,
287                     Gbl_LineBufPtr - Gbl_CurrentLineBuffer);
288     }
289
290     Gbl_CurrentLineOffset += Gbl_CurrentColumn;
291     Gbl_CurrentColumn = 0;
292
293     Gbl_CurrentLineNumber++;
294     Gbl_LogicalLineNumber++;
295     Gbl_LineBufPtr = Gbl_CurrentLineBuffer;
296 }
297
298 /*****
299 *
300 * FUNCTION:      AslInsertLineBuffer
301 *
302 * PARAMETERS:   SourceChar      - One char from the input ASL source file
303 *
304 * RETURN:       None
305 *
306 * DESCRIPTION:  Put one character of the source file into the temp line buffer
307 *
308 * *****/
309
310 void
311 AslInsertLineBuffer (
312     int          SourceChar)
313 {
314     uint32 i;
315     uint32 Count = 1;
316
317     if (SourceChar == EOF)
318     {
319         return;
320     }
321 }

```

```

325  Gbl_InputByteCount++;
326
327  /* Handle tabs. Convert to spaces */
328
329  if (SourceChar == '\t')
330  {
331      SourceChar = ' ';
332      Count = ASL_SPACES_PER_TAB -
333              (Gbl_CurrentColumn & (ASL_SPACES_PER_TAB-1));
334  }
335
336  for (i = 0; i < Count; i++)
337  {
338      Gbl_CurrentColumn++;
339
340      /* Insert the character into the line buffer */
341
342      *Gbl_LineBufPtr = (UINT8) SourceChar;
343      Gbl_LineBufPtr++;
344
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
358
359          AslResetCurrentLineBuffer ();
360      }
361      else if (SourceChar == '\n')
362      {
363          /* End of line */
364
365          AslResetCurrentLineBuffer ();
366      }
367  }
368 }
369
370 /*****
371 *
372 * FUNCTION:      count
373 *
374 * PARAMETERS:   ytext      - Contains the matched keyword.
375 *               Type       - Keyword/Character type:
376 *                           0 = anything except a keyword
377 *                           1 = pseudo-keywords
378 *                           2 = non-executable ASL keywords
379 *                           3 = executable ASL keywords
380 *
381 * RETURN:       None
382 *
383 * DESCRIPTION:  Count keywords and put them into the line buffer
384 *
385 * *****/
386
387 static void
388 count (
389     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 accidentally 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     }

530     if (c == EOF)
531     {
532         /* End of file is OK, change to newline. Let parser detect EOF later */

534         c = '\n';
535     }

537     AslInsertLineBuffer (c);
538     return (TRUE);
539 }

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  *****/

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];

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);

578     DoCharacter:
579         switch (State)
580         {
581             case ASL_NORMAL_CHAR:

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;

593         case '\"':

595             /* String terminator */

597             goto CompletedString;

599         default:

601             break;
602         }
603         break;

606     case ASL_ESCAPE_SEQUENCE:

608         State = ASL_NORMAL_CHAR;
609         switch (StringChar)
610         {
611             case 'a':

613                 StringChar = 0x07;      /* BELL */
614                 break;

616             case 'b':

618                 StringChar = 0x08;      /* BACKSPACE */
619                 break;

621             case 'f':

623                 StringChar = 0x0C;      /* FORMFEED */
624                 break;

626             case 'n':

628                 StringChar = 0x0A;      /* LINEFEED */
629                 break;

631             case 'r':

633                 StringChar = 0x0D;      /* CARRIAGE RETURN*/
634                 break;

636             case 't':

638                 StringChar = 0x09;      /* HORIZONTAL TAB */
639                 break;

641             case 'v':

643                 StringChar = 0x0B;      /* VERTICAL TAB */
644                 break;

646             case 'x':

648                 State = ASL_HEX_CONSTANT;
649                 i = 0;
650                 continue;

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_STRTOL (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_STRTOL (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     AslCompilerIval.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 }
```



new/usr/src/common/acpica/compiler/asltransform.c

1

```
*****
23962 Fri Jan 17 07:29:41 2014
new/usr/src/common/acpica/compiler/asltransform.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: asltransform - Parse tree transforms
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"

48 #define _COMPONENT ACPI_COMPILER
49 ACPI_MODULE_NAME ("asltransform")

51 /* Local prototypes */

53 static void
54 TrTransformSubtree (
55     ACPI_PARSE_OBJECT *Op);

57 static char *
58 TrAmlGetNextTempName (
59     ACPI_PARSE_OBJECT *Op,
```

new/usr/src/common/acpica/compiler/asltransform.c

2

```
60     UINT8 *TempCount);

62 static void
63 TrAmlInitLineNumbers (
64     ACPI_PARSE_OBJECT *Op,
65     ACPI_PARSE_OBJECT *Neighbor);

67 static void
68 TrAmlInitNode (
69     ACPI_PARSE_OBJECT *Op,
70     UINT16 ParseOpcode);

72 static void
73 TrAmlSetSubtreeParent (
74     ACPI_PARSE_OBJECT *Op,
75     ACPI_PARSE_OBJECT *Parent);

77 static void
78 TrAmlInsertPeer (
79     ACPI_PARSE_OBJECT *Op,
80     ACPI_PARSE_OBJECT *NewPeer);

82 static void
83 TrDoDefinitionBlock (
84     ACPI_PARSE_OBJECT *Op);

86 static void
87 TrDoSwitch (
88     ACPI_PARSE_OBJECT *StartNode);

91 /*****
92 *
93 * FUNCTION: TrAmlGetNextTempName
94 *
95 * PARAMETERS: Op - Current parse op
96 * TempCount - Current temporary counter. Was originally
97 * per-module; Currently per method, could be
98 * expanded to per-scope.
99 *
100 * RETURN: A pointer to name (allocated here).
101 *
102 * DESCRIPTION: Generate an ACPI name of the form _T_x. These names are
103 * reserved for use by the ASL compiler. (_T_0 through _T_Z)
104 *
105 *****/

107 static char *
108 TrAmlGetNextTempName (
109     ACPI_PARSE_OBJECT *Op,
110     UINT8 *TempCount)
111 {
112     char *TempName;

115     if (*TempCount >= (10+26)) /* 0-35 valid: 0-9 and A-Z for TempName[3] */
116     {
117         /* Too many temps */

119         AslError (ASL_ERROR, ASL_MSG_TOO_MANY_TEMPS, Op, NULL);
120         return (NULL);
121     }

123     TempName = UtLocalCalloc (5);

125     if (*TempCount < 10) /* 0-9 */
```

```

126 {
127     TempName[3] = (char) (*TempCount + '0');
128 }
129 else
130     /* 10-35: A-Z */
131 {
132     TempName[3] = (char) (*TempCount + ('A' - 10));
133     (*TempCount)++;
134 }
135
136 /* First three characters are always "_T_" */
137
138 TempName[0] = '_';
139 TempName[1] = 'T';
140 TempName[2] = '_';
141
142 return (TempName);
143 }
144
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 *****/
157
158 static void
159 TrAmlInitLineNumbers (
160     ACPI_PARSE_OBJECT *Op,
161     ACPI_PARSE_OBJECT *Neighbor)
162 {
163
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 }
170
171 /*****
172 *
173 * FUNCTION:      TrAmlInitNode
174 *
175 * PARAMETERS:    Op          - Op to be initialized
176 *                ParseOpcode  - Opcode for this node
177 *
178 * RETURN:        None
179 *
180 * DESCRIPTION:   Initialize a node with the parse opcode and opcode name.
181 *
182 *
183 *****/
184
185 static void
186 TrAmlInitNode (
187     ACPI_PARSE_OBJECT *Op,
188     UINT16 ParseOpcode)
189 {
190
191     Op->Asl.ParseOpcode = ParseOpcode;

```

```

192     UtSetParseOpName (Op);
193 }
194
195 /*****
196 *
197 * FUNCTION:      TrAmlSetSubtreeParent
198 *
199 * PARAMETERS:    Op          - First node in a list of peer nodes
200 *                Parent      - Parent of the subtree
201 *
202 * RETURN:        None
203 *
204 * DESCRIPTION:   Set the parent for all peer nodes in a subtree
205 *
206 *****/
207
208 static void
209 TrAmlSetSubtreeParent (
210     ACPI_PARSE_OBJECT *Op,
211     ACPI_PARSE_OBJECT *Parent)
212 {
213     ACPI_PARSE_OBJECT *Next;
214
215     Next = Op;
216     while (Next)
217     {
218         Next->Asl.Parent = Parent;
219         Next = Next->Asl.Next;
220     }
221 }
222
223 /*****
224 *
225 * FUNCTION:      TrAmlInsertPeer
226 *
227 * PARAMETERS:    Op          - First node in a list of peer nodes
228 *                NewPeer     - Peer node to insert
229 *
230 * RETURN:        None
231 *
232 * DESCRIPTION:   Insert a new peer node into a list of peers.
233 *
234 *****/
235
236 static void
237 TrAmlInsertPeer (
238     ACPI_PARSE_OBJECT *Op,
239     ACPI_PARSE_OBJECT *NewPeer)
240 {
241     ACPI_PARSE_OBJECT *Next;
242
243     Next = Op->Asl.Next;
244     Op->Asl.Next = NewPeer;
245     NewPeer->Asl.Next = Next;
246 }
247
248 /*****
249 *
250 * FUNCTION:      TrAmlTransformWalk
251 *
252 * PARAMETERS:    ASL_WALK_CALLBACK
253 *
254 * RETURN:        None
255 *
256 *****/

```

```

258 * DESCRIPTION: Parse tree walk to generate both the AML opcodes and the AML
259 * operands.
260 *
261 *****/
263 ACPI_STATUS
264 TrAmlTransformWalk (
265     ACPI_PARSE_OBJECT *Op,
266     UINT32 Level,
267     void *Context)
268 {
269     TrTransformSubtree (Op);
270     return (AE_OK);
271 }
272
273 /*****/
274 *
275 * FUNCTION: TrTransformSubtree
276 *
277 * PARAMETERS: Op - The parent parse node
278 *
279 * RETURN: None
280 *
281 * DESCRIPTION: Prepare nodes to be output as AML data and operands. The more
282 * complex AML opcodes require processing of the child nodes
283 * (arguments/operands).
284 *
285 *****/
286
287 static void
288 TrTransformSubtree (
289     ACPI_PARSE_OBJECT *Op)
290 {
291     if (Op->Asl.AmlOpcode == AML_RAW_DATA_BYTE)
292     {
293         return;
294     }
295
296     switch (Op->Asl.ParseOpcode)
297     {
298     case PARSEOP_DEFINITIONBLOCK:
299         TrDoDefinitionBlock (Op);
300         break;
301
302     case PARSEOP_SWITCH:
303         TrDoSwitch (Op);
304         break;
305
306     case PARSEOP_METHOD:
307         /*
308          * TBD: Zero the tempname (_T_x) count. Probably shouldn't be a global,
309          * however
310          */
311         Gbl_TempCount = 0;
312         break;
313
314     default:
315         /* Nothing to do here for other opcodes */
316         break;
317 }

```

```

324 }
325 }
326
327 /*****/
328 *
329 * FUNCTION: TrDoDefinitionBlock
330 *
331 * PARAMETERS: Op - Parse node
332 *
333 * RETURN: None
334 *
335 * DESCRIPTION: Find the end of the definition block and set a global to this
336 * node. It is used by the compiler to insert compiler-generated
337 * names at the root level of the namespace.
338 *
339 *****/
340
341 static void
342 TrDoDefinitionBlock (
343     ACPI_PARSE_OBJECT *Op)
344 {
345     ACPI_PARSE_OBJECT *Next;
346     UINT32 i;
347
348     Next = Op->Asl.Child;
349     for (i = 0; i < 5; i++)
350     {
351         Next = Next->Asl.Next;
352         if (i == 0)
353         {
354             /*
355              * This is the table signature. Only the DSDT can be assumed
356              * to be at the root of the namespace; Therefore, namespace
357              * optimization can only be performed on the DSDT.
358              */
359             if (!ACPI_COMPARE_NAME (Next->Asl.Value.String, ACPI_SIG_DSDT))
360             {
361                 Gbl_ReferenceOptimizationFlag = FALSE;
362             }
363         }
364     }
365
366     Gbl_FirstLevelInsertionNode = Next;
367 }
368
369 /*****/
370 *
371 * FUNCTION: TrDoSwitch
372 *
373 * PARAMETERS: StartNode - Parse node for SWITCH
374 *
375 * RETURN: None
376 *
377 * DESCRIPTION: Translate ASL SWITCH statement to if/else pairs. There is
378 * no actual AML opcode for SWITCH -- it must be simulated.
379 *
380 *****/
381
382 static void
383 TrDoSwitch (
384     ACPI_PARSE_OBJECT *StartNode)
385 {

```

```

390     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          * If (LNotEqual (Match (Package(<size>){<data>},
479          *                      MEQ, _T_x, MTR, Zero, Zero), Ones))
480          */
481         NewOp2          = TrCreateLeafNode (PARSEOP_MATCHTYPE_MEQ);
482         Predicate->Asl.Next = NewOp2;
483         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 it 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; /* Return type */
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_BTTYPE_INTEGER:

707      NewOp2->Asl.Next = TrCreateValuedLeafNode (PARSEOP_ZERO,
708                                                  (UINT64) 0);
709      TrAmlInitLineNumbers (NewOp2->Asl.Next, NewOp);
710      break;

712  case ACPI_BTTYPE_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_BTTYPE_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 }

new/usr/src/common/acpica/compiler/asltree.c

1

```
*****
32561 Fri Jan 17 07:29:41 2014
new/usr/src/common/acpica/compiler/asltree.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: asltree - parse tree management
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"
47 #include "acapps.h"
48 #include <time.h>

50 #define _COMPONENT          ACPI_COMPILER
51 #define ACPI_MODULE_NAME    ("asltree")

53 /* Local prototypes */

55 static ACPI_PARSE_OBJECT *
56 TrGetNextNode (
57     void);

59 static char *
```

new/usr/src/common/acpica/compiler/asltree.c

2

```
60 TrGetNodeFlagName (
61     UINT32                Flags);

64 /*****
65 *
66 * FUNCTION:      TrGetNextNode
67 *
68 * PARAMETERS:    None
69 *
70 * RETURN:        New parse node. Aborts on allocation failure
71 *
72 * DESCRIPTION:    Allocate a new parse node for the parse tree. Bypass the local
73 *                 dynamic memory manager for performance reasons (This has a
74 *                 major impact on the speed of the compiler.)
75 *
76 *****/

78 static ACPI_PARSE_OBJECT *
79 TrGetNextNode (
80     void)
81 {
82
83     if (Gbl_NodeCacheNext >= Gbl_NodeCacheLast)
84     {
85         Gbl_NodeCacheNext = UtLocalCalloc (sizeof (ACPI_PARSE_OBJECT) *
86                                             ASL_NODE_CACHE_SIZE);
87         Gbl_NodeCacheLast = Gbl_NodeCacheNext + ASL_NODE_CACHE_SIZE;
88     }

90     return (Gbl_NodeCacheNext++);
91 }

94 /*****
95 *
96 * FUNCTION:      TrAllocateNode
97 *
98 * PARAMETERS:    ParseOpcode          - Opcode to be assigned to the node
99 *
100 * RETURN:        New parse node. Aborts on allocation failure
101 *
102 * DESCRIPTION:    Allocate and initialize a new parse node for the parse tree
103 *
104 *****/

106 ACPI_PARSE_OBJECT *
107 TrAllocateNode (
108     UINT32                ParseOpcode)
109 {
110     ACPI_PARSE_OBJECT      *Op;

113     Op = TrGetNextNode ();

115     Op->Asl.ParseOpcode      = (UINT16) ParseOpcode;
116     Op->Asl.Filename         = Gbl_Files[ASL_FILE_INPUT].Filename;
117     Op->Asl.LineNumber        = Gbl_CurrentLineNumber;
118     Op->Asl.LogicalLineNumber = Gbl_LogicalLineNumber;
119     Op->Asl.LogicalByteOffset = Gbl_CurrentLineOffset;
120     Op->Asl.Column            = Gbl_CurrentColumn;

122     UtSetParseOpName (Op);
123     return (Op);
124 }
```



```

127 /*****
128 *
129 * FUNCTION:      TrReleaseNode
130 *
131 * PARAMETERS:    Op          - Op to be released
132 *
133 * RETURN:        None
134 *
135 * DESCRIPTION:   "release" a node. In truth, nothing is done since the node
136 *               is part of a larger buffer
137 *
138 *****/
140 void
141 TrReleaseNode (
142     ACPI_PARSE_OBJECT *Op)
143 {
144
145     return;
146 }
147
148 /*****
149 *
150 * FUNCTION:      TrUpdateNode
151 *
152 * PARAMETERS:    ParseOpcode  - New opcode to be assigned to the node
153 *               Op            - An existing parse node
154 *
155 * RETURN:        The updated node
156 *
157 * DESCRIPTION:   Change the parse opcode assigned to a node. Usually used to
158 *               change an opcode to DEFAULT_ARG so that the node is ignored
159 *               during the code generation. Also used to set generic integers
160 *               to a specific size (8, 16, 32, or 64 bits)
161 *
162 *****/
163
164 ACPI_PARSE_OBJECT *
165 TrUpdateNode (
166     UINT32          ParseOpcode,
167     ACPI_PARSE_OBJECT *Op)
168 {
169     if (!Op)
170     {
171         return (NULL);
172     }
173
174     DbgPrint (ASL_PARSE_OUTPUT,
175         "\nUpdateNode: Old - %s, New - %s\n\n",
176         UtGetOpName (Op->Asl.ParseOpcode),
177         UtGetOpName (ParseOpcode));
178
179     /* Assign new opcode and name */
180
181     if (Op->Asl.ParseOpcode == PARSEOP_ONES)
182     {
183         switch (ParseOpcode)
184         {
185             case PARSEOP_BYTECONST:
186
187                 Op->Asl.Value.Integer = ACPI_UINT8_MAX;
188                 break;

```

```

192     case PARSEOP_WORDCONST:
193
194         Op->Asl.Value.Integer = ACPI_UINT16_MAX;
195         break;
196
197     case PARSEOP_DWORDCONST:
198
199         Op->Asl.Value.Integer = ACPI_UINT32_MAX;
200         break;
201
202     /* Don't need to do the QWORD case */
203
204     default:
205
206         /* Don't care about others */
207         break;
208     }
209 }
210
211 Op->Asl.ParseOpcode = (UINT16) ParseOpcode;
212 UtSetParseOpName (Op);
213
214 /*
215 * For the BYTE, WORD, and DWORD constants, make sure that the integer
216 * that was passed in will actually fit into the data type
217 */
218 switch (ParseOpcode)
219 {
220     case PARSEOP_BYTECONST:
221
222         UtCheckIntegerRange (Op, 0x00, ACPI_UINT8_MAX);
223         Op->Asl.Value.Integer &= ACPI_UINT8_MAX;
224         break;
225
226     case PARSEOP_WORDCONST:
227
228         UtCheckIntegerRange (Op, 0x00, ACPI_UINT16_MAX);
229         Op->Asl.Value.Integer &= ACPI_UINT16_MAX;
230         break;
231
232     case PARSEOP_DWORDCONST:
233
234         UtCheckIntegerRange (Op, 0x00, ACPI_UINT32_MAX);
235         Op->Asl.Value.Integer &= ACPI_UINT32_MAX;
236         break;
237
238     default:
239
240         /* Don't care about others, don't need to check QWORD */
241
242         break;
243 }
244
245 return (Op);
246 }
247
248 /*****
249 *
250 * FUNCTION:      TrGetNodeFlagName
251 *
252 * PARAMETERS:    Flags          - Flags word to be decoded
253 *
254 * RETURN:        Name string. Always returns a valid string pointer.
255 *
256 * DESCRIPTION:   Decode a flags word

```

```

258 *
259 *****/
261 static char *
262 TrGetNodeFlagName (
263     UINT32          Flags)
264 {
266     switch (Flags)
267     {
268     case NODE_VISITED:
270         return ("NODE_VISITED");
272     case NODE_AML_PACKAGE:
274         return ("NODE_AML_PACKAGE");
276     case NODE_IS_TARGET:
278         return ("NODE_IS_TARGET");
280     case NODE_IS_RESOURCE_DESC:
282         return ("NODE_IS_RESOURCE_DESC");
284     case NODE_IS_RESOURCE_FIELD:
286         return ("NODE_IS_RESOURCE_FIELD");
288     case NODE_HAS_NO_EXIT:
290         return ("NODE_HAS_NO_EXIT");
292     case NODE_IF_HAS_NO_EXIT:
294         return ("NODE_IF_HAS_NO_EXIT");
296     case NODE_NAME_INTERNALIZED:
298         return ("NODE_NAME_INTERNALIZED");
300     case NODE_METHOD_NO_RETVAL:
302         return ("NODE_METHOD_NO_RETVAL");
304     case NODE_METHOD_SOME_NO_RETVAL:
306         return ("NODE_METHOD_SOME_NO_RETVAL");
308     case NODE_RESULT_NOT_USED:
310         return ("NODE_RESULT_NOT_USED");
312     case NODE_METHOD_TYPED:
314         return ("NODE_METHOD_TYPED");
316     case NODE_COMPILE_TIME_CONST:
318         return ("NODE_COMPILE_TIME_CONST");
320     case NODE_IS_TERM_ARG:
322         return ("NODE_IS_TERM_ARG");

```

```

324     case NODE_WAS_ONES_OP:
326         return ("NODE_WAS_ONES_OP");
328     case NODE_IS_NAME_DECLARATION:
330         return ("NODE_IS_NAME_DECLARATION");
332     default:
334         return ("Multiple Flags (or unknown flag) set");
335     }
336 }
339 /*****
340 *
341 * FUNCTION:      TrSetNodeFlags
342 *
343 * PARAMETERS:    Op          - An existing parse node
344 *                Flags       - New flags word
345 *
346 * RETURN:        The updated parser op
347 *
348 * DESCRIPTION:   Set bits in the node flags word. Will not clear bits, only set
349 *
350 *****/
352 ACPI_PARSE_OBJECT *
353 TrSetNodeFlags (
354     ACPI_PARSE_OBJECT *Op,
355     UINT32          Flags)
356 {
358     DbgPrint (ASL_PARSE_OUTPUT,
359         "\nSetNodeFlags: Op %p, %8.8X %s\n\n", Op, Flags,
360         TrGetNodeFlagName (Flags));
362     if (!Op)
363     {
364         return (NULL);
365     }
367     Op->Asl.CompileFlags |= Flags;
368     return (Op);
369 }
372 /*****
373 *
374 * FUNCTION:      TrSetNodeAmlLength
375 *
376 * PARAMETERS:    Op          - An existing parse node
377 *                Length      - AML Length
378 *
379 * RETURN:        The updated parser op
380 *
381 * DESCRIPTION:   Set the AML Length in a node. Used by the parser to indicate
382 *                the presence of a node that must be reduced to a fixed length
383 *                constant.
384 *
385 *****/
387 ACPI_PARSE_OBJECT *
388 TrSetNodeAmlLength (
389     ACPI_PARSE_OBJECT *Op,

```

```

390     UINT32          Length)
391 {
392
393     DbgPrint (ASL_PARSE_OUTPUT,
394         "\nSetNodeAmlLength: Op %p, %8.8X\n", Op, Length);
395
396     if (!Op)
397     {
398         return (NULL);
399     }
400
401     Op->Asl.AmlLength = Length;
402     return (Op);
403 }
404
405 /*****
406 * FUNCTION:      TrSetEndLineNumber
407 *
408 * PARAMETERS:    Op          - An existing parse node
409 *
410 * RETURN:        None.
411 *
412 * DESCRIPTION:   Set the ending line numbers (file line and logical line) of a
413 *                parse node to the current line numbers.
414 *
415 *
416 *
417 *****/
418
419 void
420 TrSetEndLineNumber (
421     ACPI_PARSE_OBJECT *Op)
422 {
423
424     /* If the end line # is already set, just return */
425
426     if (Op->Asl.EndLine)
427     {
428         return;
429     }
430
431     Op->Asl.EndLine      = Gbl_CurrentLineNumber;
432     Op->Asl.EndLogicalLine = Gbl_LogicalLineNumber;
433 }
434
435 /*****
436 * FUNCTION:      TrCreateLeafNode
437 *
438 * PARAMETERS:    ParseOpcode - New opcode to be assigned to the node
439 *
440 * RETURN:        Pointer to the new node. Aborts on allocation failure
441 *
442 * DESCRIPTION:   Create a simple leaf node (no children or peers, and no value
443 *                assigned to the node)
444 *
445 *
446 *
447 *****/
448
449 ACPI_PARSE_OBJECT *
450 TrCreateLeafNode (
451     UINT32          ParseOpcode)
452 {
453     ACPI_PARSE_OBJECT *Op;

```

```

454     Op = TrAllocateNode (ParseOpcode);
455
456     DbgPrint (ASL_PARSE_OUTPUT,
457         "\nCreateLeafNode Ln/Col %u/%u NewNode %p Op %s\n\n",
458         Op->Asl.LineNumber, Op->Asl.Column, Op, UtGetOpName(ParseOpcode));
459
460     return (Op);
461 }
462
463 /*****
464 * FUNCTION:      TrCreateConstantLeafNode
465 *
466 * PARAMETERS:    ParseOpcode - The constant opcode
467 *
468 * RETURN:        Pointer to the new node. Aborts on allocation failure
469 *
470 * DESCRIPTION:   Create a leaf node (no children or peers) for one of the
471 *                special constants - __LINE__, __FILE__, and __DATE__.
472 *
473 * Note: An implementation of __FUNC__ cannot happen here because we don't
474 * have a full parse tree at this time and cannot find the parent control
475 * method. If it is ever needed, __FUNC__ must be implemented later, after
476 * the parse tree has been fully constructed.
477 *
478 *****/
479
480 ACPI_PARSE_OBJECT *
481 TrCreateConstantLeafNode (
482     UINT32          ParseOpcode)
483 {
484     ACPI_PARSE_OBJECT *Op = NULL;
485     time_t             CurrentTime;
486     char                *StaticTimeString;
487     char                *TimeString;
488     char                *Path;
489     char                *Filename;
490
491     switch (ParseOpcode)
492     {
493     case PARSEOP__LINE__:
494
495         Op = TrAllocateNode (PARSEOP_INTEGER);
496         Op->Asl.Value.Integer = Op->Asl.LineNumber;
497         break;
498
499     case PARSEOP__PATH__:
500
501         Op = TrAllocateNode (PARSEOP_STRING_LITERAL);
502
503         /* Op->Asl.Filename contains the full pathname to the file */
504
505         Op->Asl.Value.String = Op->Asl.Filename;
506         break;
507
508     case PARSEOP__FILE__:
509
510         Op = TrAllocateNode (PARSEOP_STRING_LITERAL);
511
512         /* Get the simple filename from the full path */
513
514         FlSplitInputPathname (Op->Asl.Filename, &Path, &Filename);
515         ACPI_FREE (Path);
516         Op->Asl.Value.String = Filename;

```

```

522     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);
655
656     /* Allocate one new node */
657
658     Op = TrAllocateNode (ParseOpcode);
659
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));
663
664     /* Some extra debug output based on the parse opcode */
665
666     switch (ParseOpcode)
667     {
668     case PARSEOP_DEFINITIONBLOCK:
669
670         RootNode = Op;
671         DbgPrint (ASL_PARSE_OUTPUT, "DEFINITION_BLOCK (Tree Completed)->");
672         break;
673
674     case PARSEOP_OPERATIONREGION:
675
676         DbgPrint (ASL_PARSE_OUTPUT, "OPREGION->");
677         break;
678
679     case PARSEOP_OR:
680
681         DbgPrint (ASL_PARSE_OUTPUT, "OR->");
682         break;
683
684     default:
685
686         /* Nothing to do for other opcodes */
687
688         break;
689     }
690
691     /* Link the new node to its children */
692
693     PrevChild = NULL;
694     FirstChild = TRUE;
695     for (i = 0; i < NumChildren; i++)
696     {
697         /* Get the next child */
698
699         Child = va_arg (ap, ACPI_PARSE_OBJECT *);
700         DbgPrint (ASL_PARSE_OUTPUT, "%p, ", Child);
701
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         }
712
713         /* Link first child to parent */
714
715         if (FirstChild)
716         {
717             FirstChild = FALSE;
718             Op->Asl.Child = Child;
719         }

```

```

721     /* Point all children to parent */
722
723     Child->Asl.Parent = Op;
724
725     /* Link children in a peer list */
726
727     if (PrevChild)
728     {
729         PrevChild->Asl.Next = Child;
730     };
731
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     }
741
742     PrevChild = Child;
743 }
744 va_end(ap);
745
746 DbgPrint (ASL_PARSE_OUTPUT, "\n\n");
747 return (Op);
748 }
749
750 /*****
751  * FUNCTION:      TrLinkChildren
752  *
753  * PARAMETERS:    Op          - An existing parse node
754  *                  NumChildren - Number of children to follow
755  *                  ...        - A list of child nodes to link to the new
756  *                               node. NumChildren long.
757  *
758  * RETURN:        The updated (linked) node
759  *
760  * DESCRIPTION:   Link a group of nodes to an existing parse node
761  *****/
762
763
764
765 ACPI_PARSE_OBJECT *
766 TrLinkChildren (
767     ACPI_PARSE_OBJECT *Op,
768     UINT32 NumChildren,
769     ...)
770 {
771     ACPI_PARSE_OBJECT *Child;
772     ACPI_PARSE_OBJECT *PrevChild;
773     va_list ap;
774     UINT32 i;
775     BOOLEAN FirstChild;
776
777     va_start (ap, NumChildren);
778
779     TrSetEndLineNumber (Op);
780
781     DbgPrint (ASL_PARSE_OUTPUT,
782              "\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     }
1176     else
1177     {
1178         /* 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         }
1247     }

```

```

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 }

```

new/usr/src/common/acpica/compiler/asltypes.h

1

```
*****
7356 Fri Jan 17 07:29:41 2014
new/usr/src/common/acpica/compiler/asltypes.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: asltypes.h - compiler data types and struct definitions
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ASLTYPES_H
46 #define __ASLTYPES_H

49 /*****
50 *
51 * Structure definitions
52 *
53 *****/

56 /* Op flags for the ACPI_PARSE_OBJECT */

58 #define NODE_VISITED          0x00000001
59 #define NODE_AML_PACKAGE      0x00000002
```

new/usr/src/common/acpica/compiler/asltypes.h

2

```
60 #define NODE_IS_TARGET          0x00000004
61 #define NODE_IS_RESOURCE_DESC    0x00000008
62 #define NODE_IS_RESOURCE_FIELD   0x00000010
63 #define NODE_HAS_NO_EXIT         0x00000020
64 #define NODE_IF_HAS_NO_EXIT      0x00000040
65 #define NODE_NAME_INTERNALIZED   0x00000080
66 #define NODE_METHOD_NO_RETVAL    0x00000100
67 #define NODE_METHOD_SOME_NO_RETVAL 0x00000200
68 #define NODE_RESULT_NOT_USED     0x00000400
69 #define NODE_METHOD_TYPED        0x00000800
70 #define NODE_UNUSED_FLAG        0x00001000
71 #define NODE_COMPILE_TIME_CONST 0x00002000
72 #define NODE_IS_TERM_ARG        0x00004000
73 #define NODE_WAS_ONES_OP        0x00008000
74 #define NODE_IS_NAME_DECLARATION 0x00010000
75 #define NODE_COMPILER_EMITTED   0x00020000
76 #define NODE_IS_DUPLICATE        0x00040000
77 #define NODE_IS_RESOURCE_DATA    0x00080000
78 #define NODE_IS_NULL_RETURN      0x00100000

80 /* Keeps information about individual control methods */

82 typedef struct asl_method_info
83 {
84     ACPI_PARSE_OBJECT *Op;
85     struct asl_method_info *Next;
86     UINT32 ValidArgTypes[ACPI_METHOD_NUM_ARGS];
87     UINT32 ValidReturnTypes;
88     UINT32 NumReturnNoValue;
89     UINT32 NumReturnWithValue;
90     UINT8 NumArguments;
91     UINT8 LocalInitialized[ACPI_METHOD_NUM_LOCALS];
92     UINT8 ArgInitialized[ACPI_METHOD_NUM_ARGS];
93     UINT8 HasBeenTyped;
94     UINT8 ShouldBeSerialized;
95 } ASL_METHOD_INFO;

99 /* Parse tree walk info for control method analysis */

101 typedef struct asl_analysis_walk_info
102 {
103     ASL_METHOD_INFO *MethodStack;
104 } ASL_ANALYSIS_WALK_INFO;

108 /* An entry in the ParseOpcode to AmlOpcode mapping table */

110 typedef struct asl_mapping_entry
111 {
112     UINT32 Value;
113     UINT32 AcpiBtype; /* Object type or return type */
114     UINT16 AmlOpcode;
115     UINT8 Flags;
116 } ASL_MAPPING_ENTRY;

120 /* Parse tree walk info structure */

122 typedef struct asl_walk_info
123 {
124     ACPI_PARSE_OBJECT **NodePtr;
125     UINT32 *LevelPtr;
```

```

127 } ASL_WALK_INFO;

130 /* File info */

132 typedef struct asl_file_info
133 {
134     FILE                *Handle;
135     char                *Filename;
136     const char          *ShortDescription;
137     const char          *Description;
138 } ASL_FILE_INFO;

141 typedef struct asl_file_status
142 {
143     UINT32              Line;
144     UINT32              Offset;
145 } ASL_FILE_STATUS;

149 /*
150  * File types. Note: Any changes to this table must also be reflected
151  * in the Gbl_Files array.
152  */
153 typedef enum
154 {
155     ASL_FILE_STDOUT      = 0,
156     ASL_FILE_STDERR,
157     ASL_FILE_INPUT,
158     ASL_FILE_AML_OUTPUT, /* Don't move these first 4 file types */
159     ASL_FILE_SOURCE_OUTPUT,
160     ASL_FILE_PREPROCESSOR,
161     ASL_FILE_LISTING_OUTPUT,
162     ASL_FILE_HEX_OUTPUT,
163     ASL_FILE_NAMESPACE_OUTPUT,
164     ASL_FILE_DEBUG_OUTPUT,
165     ASL_FILE_ASM_SOURCE_OUTPUT,
166     ASL_FILE_C_SOURCE_OUTPUT,
167     ASL_FILE_ASM_INCLUDE_OUTPUT,
168     ASL_FILE_C_INCLUDE_OUTPUT,
169     ASL_FILE_C_OFFSET_OUTPUT
170 } ASL_FILE_TYPES;

174 #define ASL_MAX_FILE_TYPE      14
175 #define ASL_NUM_FILES          (ASL_MAX_FILE_TYPE + 1)

178 typedef struct asl_include_dir
179 {
180     char                *Dir;
181     struct asl_include_dir *Next;
182 } ASL_INCLUDE_DIR;

186 /* An entry in the exception list, one for each error/warning */

188 typedef struct asl_error_msg
189 {
190     UINT32              LineNumber;
191     UINT32              LogicalLineNumber;

```

```

192     UINT32              LogicalByteOffset;
193     UINT32              Column;
194     char                *Message;
195     struct asl_error_msg *Next;
196     char                *Filename;
197     char                *SourceLine;
198     UINT32              FilenameLength;
199     UINT8               MessageId;
200     UINT8               Level;
201 } ASL_ERROR_MSG;

205 /* An entry in the listing file stack (for include files) */

207 typedef struct asl_listing_node
208 {
209     char                *Filename;
210     UINT32              LineNumber;
211     struct asl_listing_node *Next;
212 } ASL_LISTING_NODE;

216 /* Callback interface for a parse tree walk */

218 /*
219  * TBD - another copy of this is in adisasm.h, fix
220  */
221 #ifndef ASL_WALK_CALLBACK_DEFINED
222 typedef
223 ACPI_STATUS (*ASL_WALK_CALLBACK) (
224     ACPI_PARSE_OBJECT *Op,
225     UINT32              Level,
226     void                *Context);
227 #define ASL_WALK_CALLBACK_DEFINED
228 #endif

231 typedef struct asl_event_info
232 {
233     UINT64              StartTime;
234     UINT64              EndTime;
235     char                *EventName;
236     BOOLEAN              Valid;
237 } ASL_EVENT_INFO;

241 #endif /* __ASLTYPES_H */

```

new/usr/src/common/acpica/compiler/aslutils.c

1

```
*****
28300 Fri Jan 17 07:29:41 2014
new/usr/src/common/acpica/compiler/aslutils.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslutils -- compiler utilities
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"
47 #include "acdisasm.h"
48 #include "acnamesp.h"
49 #include "amlcode.h"
50 #include <acapps.h>

52 #define _COMPONENT          ACPI_COMPILER
53       ACPI_MODULE_NAME    ("aslutils")

56 /* Local prototypes */

58 static void
59 UtPadNameWithUnderscores (
```

new/usr/src/common/acpica/compiler/aslutils.c

2

```
60     char          *NameSeg,
61     char          *PaddedNameSeg);

63 static void
64 UtAttachNameSeg (
65     ACPI_PARSE_OBJECT *Op,
66     char              *Name);

69 /*****
70 *
71 * FUNCTION:      UtDisplaySupportedTables
72 *
73 * PARAMETERS:    None
74 *
75 * RETURN:        None
76 *
77 * DESCRIPTION:   Print all supported ACPI table names.
78 *
79 *****/

81 #define ACPI_TABLE_HELP_FORMAT    "%8u) %s      %s\n"

83 void
84 UtDisplaySupportedTables (
85     void)
86 {
87     ACPI_DM_TABLE_DATA    *TableData;
88     UINT32                i;

91     printf ("\nACPI tables supported by iASL version %8.8X:\n"
92             " (Compiler, Disassembler, Template Generator)\n\n",
93             ACPI_CA_VERSION);

95     /* Special tables */

97     printf (" Special tables and AML tables:\n");
98     printf (ACPI_TABLE_HELP_FORMAT, 1, ACPI_RSDP_NAME, "Root System Description
99     printf (ACPI_TABLE_HELP_FORMAT, 2, ACPI_SIG_FACS, "Firmware ACPI Control Str
100    printf (ACPI_TABLE_HELP_FORMAT, 3, ACPI_SIG_DSDT, "Differentiated System Des
101    printf (ACPI_TABLE_HELP_FORMAT, 4, ACPI_SIG_SSDT, "Secondary System Descript

103     /* All data tables with common table header */

105     printf ("\n Standard ACPI data tables:\n");
106     for (TableData = AcpiDmTableData, i = 5; TableData->Signature; TableData++,
107     {
108         printf (ACPI_TABLE_HELP_FORMAT, i, TableData->Signature, TableData->Name
109     }
110 }

113 /*****
114 *
115 * FUNCTION:      UtDisplayConstantOpcodes
116 *
117 * PARAMETERS:    None
118 *
119 * RETURN:        None
120 *
121 * DESCRIPTION:   Print AML opcodes that can be used in constant expressions.
122 *
123 *****/

125 void
```

```

126 UtDisplayConstantOpcodes (
127     void)
128 {
129     UINT32          i;

132     printf ("Constant expression opcode information\n\n");

134     for (i = 0; i < sizeof (AcpiGbl_AmlOpInfo) / sizeof (ACPI_OPCODE_INFO); i++)
135     {
136         if (AcpiGbl_AmlOpInfo[i].Flags & AML_CONSTANT)
137         {
138             printf ("%s\n", AcpiGbl_AmlOpInfo[i].Name);
139         }
140     }
141 }

144 /*****
145  *
146  * FUNCTION:      UtLocalCalloc
147  *
148  * PARAMETERS:    Size                - Bytes to be allocated
149  *
150  * RETURN:        Pointer to the allocated memory. Guaranteed to be valid.
151  *
152  * DESCRIPTION:   Allocate zero-initialized memory. Aborts the compile on an
153  *                allocation failure, on the assumption that nothing more can be
154  *                accomplished.
155  *
156  *****/

158 void *
159 UtLocalCalloc (
160     UINT32          Size)
161 {
162     void            *Allocated;

165     Allocated = ACPI_ALLOCATE_ZEROED (Size);
166     if (!Allocated)
167     {
168         AslCommonError (ASL_ERROR, ASL_MSG_MEMORY_ALLOCATION,
169             Gbl_CurrentLineNumber, Gbl_LogicalLineNumber,
170             Gbl_InputByteCount, Gbl_CurrentColumn,
171             Gbl_Files[ASL_FILE_INPUT].Filename, NULL);

173         CmCleanupAndExit ();
174         exit (1);
175     }

177     TotalAllocations++;
178     TotalAllocated += Size;
179     return (Allocated);
180 }

183 /*****
184  *
185  * FUNCTION:      UtBeginEvent
186  *
187  * PARAMETERS:    Name                - Ascii name of this event
188  *
189  * RETURN:        Event number (integer index)
190  *
191  * DESCRIPTION:   Saves the current time with this event

```

```

192  *
193  *****/

195 UINT8
196 UtBeginEvent (
197     char            *Name)
198 {
199     if (AslGbl_NextEvent >= ASL_NUM_EVENTS)
200     {
201         AcpiOsPrintf ("Ran out of compiler event structs!\n");
202         return (AslGbl_NextEvent);
203     }

206     /* Init event with current (start) time */

208     AslGbl_Events[AslGbl_NextEvent].StartTime = AcpiOsGetTimer ();
209     AslGbl_Events[AslGbl_NextEvent].EventName = Name;
210     AslGbl_Events[AslGbl_NextEvent].Valid = TRUE;

212     return (AslGbl_NextEvent++);
213 }

216 /*****
217  *
218  * FUNCTION:      UtEndEvent
219  *
220  * PARAMETERS:    Event                - Event number (integer index)
221  *
222  * RETURN:        None
223  *
224  * DESCRIPTION:   Saves the current time (end time) with this event
225  *
226  *****/

228 void
229 UtEndEvent (
230     UINT8          Event)
231 {
232     if (Event >= ASL_NUM_EVENTS)
233     {
234         return;
235     }

238     /* Insert end time for event */

240     AslGbl_Events[Event].EndTime = AcpiOsGetTimer ();
241 }

244 /*****
245  *
246  * FUNCTION:      UtHexCharToValue
247  *
248  * PARAMETERS:    HexChar              - Hex character in Ascii
249  *
250  * RETURN:        The binary value of the hex character
251  *
252  * DESCRIPTION:   Perform ascii-to-hex translation
253  *
254  *****/

256 UINT8
257 UtHexCharToValue (

```

```

258     int                HexChar)
259 {
260
261     if (HexChar <= 0x39)
262     {
263         return ((UINT8) (HexChar - 0x30));
264     }
265
266     if (HexChar <= 0x46)
267     {
268         return ((UINT8) (HexChar - 0x37));
269     }
270
271     return ((UINT8) (HexChar - 0x57));
272 }
273
274 /*****
275 *
276 * FUNCTION:      UtConvertByteToHex
277 *
278 * PARAMETERS:   RawByte          - Binary data
279 *               Buffer            - Pointer to where the hex bytes will be
280 *                               stored
281 *
282 * RETURN:       Ascii hex byte is stored in Buffer.
283 *
284 * DESCRIPTION:  Perform hex-to-ascii translation. The return data is prefixed
285 *               with "0x"
286 *
287 * *****/
288
289 void
290 UtConvertByteToHex (
291     UINT8          RawByte,
292     UINT8          *Buffer)
293 {
294
295     Buffer[0] = '0';
296     Buffer[1] = 'x';
297
298     Buffer[2] = (UINT8) AslHexLookup[(RawByte >> 4) & 0xF];
299     Buffer[3] = (UINT8) AslHexLookup[RawByte & 0xF];
300 }
301
302 /*****
303 *
304 * FUNCTION:      UtConvertByteToAsmHex
305 *
306 * PARAMETERS:   RawByte          - Binary data
307 *               Buffer            - Pointer to where the hex bytes will be
308 *                               stored
309 *
310 * RETURN:       Ascii hex byte is stored in Buffer.
311 *
312 * DESCRIPTION:  Perform hex-to-ascii translation. The return data is prefixed
313 *               with "0x"
314 *
315 * *****/
316
317 void
318 UtConvertByteToAsmHex (
319     UINT8          RawByte,
320     UINT8          *Buffer)
321 {

```

```

325     Buffer[0] = '0';
326     Buffer[1] = (UINT8) AslHexLookup[(RawByte >> 4) & 0xF];
327     Buffer[2] = (UINT8) AslHexLookup[RawByte & 0xF];
328     Buffer[3] = 'h';
329 }
330
331 /*****
332 *
333 * FUNCTION:      DbgPrint
334 *
335 * PARAMETERS:   Type          - Type of output
336 *               Fmt            - Printf format string
337 *               ...            - variable printf list
338 *
339 * RETURN:       None
340 *
341 * DESCRIPTION:  Conditional print statement. Prints to stderr only if the
342 *               debug flag is set.
343 *
344 * *****/
345
346 void
347 DbgPrint (
348     UINT32          Type,
349     char            *Fmt,
350     ...             Args)
351 {
352     va_list          Args;
353
354     if (!Gbl_DebugFlag)
355     {
356         return;
357     }
358
359     if ((Type == ASL_PARSE_OUTPUT) &&
360         (!(AslCompilerdebug)))
361     {
362         return;
363     }
364
365     va_start (Args, Fmt);
366     (void) vfprintf (stderr, Fmt, Args);
367     va_end (Args);
368     return;
369 }
370
371 /*****
372 *
373 * FUNCTION:      UtPrintFormattedName
374 *
375 * PARAMETERS:   ParseOpcode     - Parser keyword ID
376 *               Level            - Indentation level
377 *
378 * RETURN:       None
379 *
380 * DESCRIPTION:  Print the ascii name of the parse opcode.
381 *
382 * *****/
383
384 #define TEXT_OFFSET 10
385
386 void

```

```

390 UtPrintFormattedName (
391     UINT16          ParseOpcode,
392     UINT32          Level)
393 {
394     if (Level)
395     {
396         DbgPrint (ASL_TREE_OUTPUT,
397             "%*s", (3 * Level), " ");
398     }
399     DbgPrint (ASL_TREE_OUTPUT,
400         " %-20.20s", UtGetOpName (ParseOpcode));
401
402     if (Level < TEXT_OFFSET)
403     {
404         DbgPrint (ASL_TREE_OUTPUT,
405             "%*s", (TEXT_OFFSET - Level) * 3, " ");
406     }
407 }
408
409 /*****
410 *
411 * FUNCTION:      UtSetParseOpName
412 *
413 * PARAMETERS:    Op                - Parse op to be named.
414 *
415 * RETURN:        None
416 *
417 * DESCRIPTION:   Insert the ascii name of the parse opcode
418 *
419 * *****/
420
421 void
422 UtSetParseOpName (
423     ACPI_PARSE_OBJECT *Op)
424 {
425     strncpy (Op->Asl.ParseOpName, UtGetOpName (Op->Asl.ParseOpcode),
426         ACPI_MAX_PARSEOP_NAME);
427 }
428
429 /*****
430 *
431 * FUNCTION:      UtDisplaySummary
432 *
433 * PARAMETERS:    FileID            - ID of output file
434 *
435 * RETURN:        None
436 *
437 * DESCRIPTION:   Display compilation statistics
438 *
439 * *****/
440
441 void
442 UtDisplaySummary (
443     UINT32          FileId)
444 {
445     if (FileId != ASL_FILE_STDOUT)
446     {
447         /* 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 }
459
460 /* Summary of main input and output files */
461
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);
469
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);
485
486     /* AML summary */
487
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 }
497
498 /* Display summary of any optional files */
499
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     }
506
507     /* .SRC is a temp file unless specifically requested */
508
509     if ((i == ASL_FILE_SOURCE_OUTPUT) && (!Gbl_SourceOutputFlag))
510     {
511         continue;
512     }
513
514     /* .I is a temp file unless specifically requested */
515
516     if ((i == ASL_FILE_PREPROCESSOR) && (!Gbl_PreprocessorOutputFlag))
517     {
518         continue;
519     }
520
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 {
566     if (!Op)
567     {
568         return (NULL);
569     }

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 */
722
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 *
787 * FUNCTION:      UtAttachNameSeg
788 *
789 * PARAMETERS:    Op          - Parent parse node
790 *               Name        - Full ExternalName
791 *
792 * RETURN:        None; Sets the NameSeg field in parent node
793 *
794 * DESCRIPTION:    Extract the last nameseg of the ExternalName and store it
795 *               in the NameSeg field of the Op.
796 *
797 *****/
799 static void
800 UtAttachNameSeg (
801     ACPI_PARSE_OBJECT *Op,
802     char *Name)
803 {
804     char *NameSeg;
805     char PaddedNameSeg[4];
806
807     if (!Name)
808     {
809         return;
810     }
811
812     /* Look for the last dot in the namepath */
813
814     NameSeg = strrchr (Name, '.');
815     if (NameSeg)
816     {
817         /* Found last dot, we have also found the final nameseg */
818
819         NameSeg++;
820         UtPadNameWithUnderscores (NameSeg, PaddedNameSeg);
821     }
822     else
823     {
824         /* No dots in the namepath, there is only a single nameseg. */
825         /* Handle prefixes */
826
827         while (ACPI_IS_ROOT_PREFIX (*Name) ||
828              ACPI_IS_PARENT_PREFIX (*Name))
829         {
830             Name++;
831         }
832
833         /* Remaining string should be one single nameseg */
834
835         UtPadNameWithUnderscores (Name, PaddedNameSeg);
836     }
837
838     ACPI_MOVE_NAME (Op->Asl.NameSeg, PaddedNameSeg);
839 }
840
841 *****/
842 *
843 * FUNCTION:      UtAttachNamepathToOwner
844 *
845 * PARAMETERS:    Op          - Parent parse node
846 *               NameOp       - Node that contains the name
847 *
848 * RETURN:        Sets the ExternalName and Namepath in the parent node
849 *
850 *****/

```

```

852 * DESCRIPTION:    Store the name in two forms in the parent node: The original
853 *               (external) name, and the internalized name that is used within
854 *               the ACPI namespace manager.
855 *
856 *****/
857
858 void
859 UtAttachNamepathToOwner (
860     ACPI_PARSE_OBJECT *Op,
861     ACPI_PARSE_OBJECT *NameOp)
862 {
863     ACPI_STATUS Status;
864
865     /* Full external path */
866
867     Op->Asl.ExternalName = NameOp->Asl.Value.String;
868
869     /* Save the NameOp for possible error reporting later */
870
871     Op->Asl.ParentMethod = (void *) NameOp;
872
873     /* Last nameseg of the path */
874
875     UtAttachNameSeg (Op, Op->Asl.ExternalName);
876
877     /* Create internalized path */
878
879     Status = UtInternalizeName (NameOp->Asl.Value.String, &Op->Asl.Namepath);
880     if (ACPI_FAILURE (Status))
881     {
882         /* TBD: abort on no memory */
883     }
884 }
885
886 *****/
887 *
888 * FUNCTION:      UtDoConstant
889 *
890 * PARAMETERS:    String      - Hex, Octal, or Decimal string
891 *
892 * RETURN:        Converted Integer
893 *
894 * DESCRIPTION:    Convert a string to an integer, with error checking.
895 *
896 *****/
897
898
899 UINT64
900 UtDoConstant (
901     char *String)
902 {
903     ACPI_STATUS Status;
904     UINT64 Converted;
905     char ErrBuf[64];
906
907     Status = UtStrtoul64 (String, 0, &Converted);
908     if (ACPI_FAILURE (Status))
909     {
910         sprintf (ErrBuf, "%s %s\n", "Conversion error:",
911                 AcpiFormatException (Status));
912         AslCompilerError (ErrBuf);
913     }
914
915     return (Converted);
916 }

```

```

918 }

921 /* TBD: use version in ACPI CA main code base? */

923 /*****
924  *
925  * FUNCTION:      UtStrtol64
926  *
927  * PARAMETERS:   String          - Null terminated string
928  *               Terminator      - Where a pointer to the terminating byte
929  *                               is returned
930  *               Base            - Radix of the string
931  *
932  * RETURN:       Converted value
933  *
934  * DESCRIPTION:  Convert a string into an unsigned value.
935  *
936  *****/

938 ACPI_STATUS
939 UtStrtol64 (
940     char          *String,
941     UINT32        Base,
942     UINT64        *RetInteger)
943 {
944     UINT32        Index;
945     UINT32        Sign;
946     UINT64        ReturnValue = 0;
947     ACPI_STATUS    Status = AE_OK;

950     *RetInteger = 0;

952     switch (Base)
953     {
954     case 0:
955     case 8:
956     case 10:
957     case 16:

959         break;

961     default:
962         /*
963          * The specified Base parameter is not in the domain of
964          * this function:
965          */
966         return (AE_BAD_PARAMETER);
967     }

969     /* Skip over any white space in the buffer: */
971     while (isspace ((int) *String) || *String == '\t')
972     {
973         ++String;
974     }

976     /*
977     * The buffer may contain an optional plus or minus sign.
978     * If it does, then skip over it but remember what it was:
979     */
980     if (*String == '-')
981     {
982         Sign = NEGATIVE;
983         ++String;

```

```

984     }
985     else if (*String == '+')
986     {
987         ++String;
988         Sign = POSITIVE;
989     }
990     else
991     {
992         Sign = POSITIVE;
993     }

995     /*
996     * If the input parameter Base is zero, then we need to
997     * determine if it is octal, decimal, or hexadecimal:
998     */
999     if (Base == 0)
1000     {
1001         if (*String == '0')
1002         {
1003             if (tolower ((int) *(++String)) == 'x')
1004             {
1005                 Base = 16;
1006                 ++String;
1007             }
1008             else
1009             {
1010                 Base = 8;
1011             }
1012         }
1013         else
1014         {
1015             Base = 10;
1016         }
1017     }

1019     /*
1020     * For octal and hexadecimal bases, skip over the leading
1021     * 0 or 0x, if they are present.
1022     */
1023     if (Base == 8 && *String == '0')
1024     {
1025         String++;
1026     }

1028     if (Base == 16 &&
1029         *String == '0' &&
1030         tolower ((int) *(++String)) == 'x')
1031     {
1032         String++;
1033     }

1035     /* Main loop: convert the string to an unsigned long */
1037     while (*String)
1038     {
1039         if (isdigit ((int) *String))
1040         {
1041             Index = ((UINT8) *String) - '0';
1042         }
1043         else
1044         {
1045             Index = (UINT8) toupper ((int) *String);
1046             if (isupper ((int) Index))
1047             {
1048                 Index = Index - 'A' + 10;
1049             }

```

```

1050         else
1051         {
1052             goto ErrorExit;
1053         }
1054     }
1055
1056     if (Index >= Base)
1057     {
1058         goto ErrorExit;
1059     }
1060
1061     /* Check to see if value is out of range: */
1062
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     }
1073
1074     ++String;
1075 }
1076
1077 /* If a minus sign was present, then "the conversion is negated": */
1078
1079 if (Sign == NEGATIVE)
1080 {
1081     ReturnValue = (ACPI_UINT32_MAX - ReturnValue) + 1;
1082 }
1083
1084 *RetInteger = ReturnValue;
1085 return (Status);
1086
1087 ErrorExit:
1088 switch (Base)
1089 {
1090     case 8:
1091
1092         Status = AE_BAD_OCTAL_CONSTANT;
1093         break;
1094
1095     case 10:
1096
1097         Status = AE_BAD_DECIMAL_CONSTANT;
1098         break;
1099
1100     case 16:
1101
1102         Status = AE_BAD_HEX_CONSTANT;
1103         break;
1104
1105     default:
1106
1107         /* Base validated above */
1108
1109         break;
1110 }
1111
1112 return (Status);
1113 }

```

new/usr/src/common/acpica/compiler/asluuid.c

1

```
*****
6676 Fri Jan 17 07:29:42 2014
new/usr/src/common/acpica/compiler/asluuid.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: asluuid-- compiler UUID support
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"

47 #define _COMPONENT          ACPI_COMPILER
48         ACPI_MODULE_NAME    ("asluuid")

51 /*
52 * UUID support functions.
53 *
54 * This table is used to convert an input UUID ascii string to a 16 byte
55 * buffer and the reverse. The table maps a UUID buffer index 0-15 to
56 * the index within the 36-byte UUID string where the associated 2-byte
57 * hex value can be found.
58 *
59 * 36-byte UUID strings are of the form:
```

new/usr/src/common/acpica/compiler/asluuid.c

2

```
60 *      aabbccdd-eeff-gghh-iijj-kllmmnnnoopp
61 * Where aa-pp are one byte hex numbers, made up of two hex digits
62 *
63 * Note: This table is basically the inverse of the string-to-offset table
64 * found in the ACPI spec in the description of the ToUUID macro.
65 */
66 static UINT8      Gbl_MapToUuidOffset[16] =
67 {
68     6,4,2,0,11,9,16,14,19,21,24,26,28,30,32,34
69 };

71 #define UUID_BUFFER_LENGTH      16
72 #define UUID_STRING_LENGTH      36

74 /* Positions for required hyphens (dashes) in UUID strings */

76 #define UUID_HYPHEN1_OFFSET      8
77 #define UUID_HYPHEN2_OFFSET      13
78 #define UUID_HYPHEN3_OFFSET      18
79 #define UUID_HYPHEN4_OFFSET      23

82 /*****
83 *
84 * FUNCTION:      AuValidateUuid
85 *
86 * PARAMETERS:    InString          - 36-byte formatted UUID string
87 *
88 * RETURN:        Status
89 *
90 * DESCRIPTION:    Check all 36 characters for correct format
91 *
92 *****/

94 ACPI_STATUS
95 AuValidateUuid (
96     char                *InString)
97 {
98     UINT32                i;

101     if (!InString || (ACPI_STRLEN (InString) != UUID_STRING_LENGTH))
102     {
103         return (AE_BAD_PARAMETER);
104     }

106     /* Check all 36 characters for correct format */

108     for (i = 0; i < UUID_STRING_LENGTH; i++)
109     {
110         /* Must have 4 hyphens (dashes) in these positions: */

112         if ((i == UUID_HYPHEN1_OFFSET) ||
113             (i == UUID_HYPHEN2_OFFSET) ||
114             (i == UUID_HYPHEN3_OFFSET) ||
115             (i == UUID_HYPHEN4_OFFSET))
116         {
117             if (InString[i] != '-')
118             {
119                 return (AE_BAD_PARAMETER);
120             }
121         }

123         /* All other positions must contain hex digits */

125         else
```

```

126     {
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 }

```

new/usr/src/common/acpica/compiler/aslwalks.c

1

```
*****
21004 Fri Jan 17 07:29:42 2014
new/usr/src/common/acpica/compiler/aslwalks.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aslwalks.c - Miscellaneous analytical parse tree walks
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"
47 #include "acparser.h"
48 #include "amlcode.h"

51 #define _COMPONENT          ACPI_COMPILER
52 #define ACPI_MODULE_NAME    ("aslwalks")

55 /*****
56 *
57 * FUNCTION:      AnMethodTypingWalkEnd
58 *
59 * PARAMETERS:    ASL_WALK_CALLBACK
```

new/usr/src/common/acpica/compiler/aslwalks.c

2

```
60 *
61 * RETURN:      Status
62 *
63 * DESCRIPTION: Ascending callback for typing walk. Complete the method
64 * return analysis. Check methods for:
65 * 1) Initialized local variables
66 * 2) Valid arguments
67 * 3) Return types
68 *
69 *****/

71 ACPI_STATUS
72 AnMethodTypingWalkEnd (
73     ACPI_PARSE_OBJECT *Op,
74     UINT32 Level,
75     void *Context)
76 {
77     UINT32 ThisNodeType;

80     switch (Op->Asl.ParseOpcode)
81     {
82     case PARSEOP_METHOD:

84         Op->Asl.CompileFlags |= NODE_METHOD_TYPED;
85         break;

87     case PARSEOP_RETURN:

89         if ((Op->Asl.Child) &&
90             (Op->Asl.Child->Asl.ParseOpcode != PARSEOP_DEFAULT_ARG))
91         {
92             ThisNodeType = AnGetBtype (Op->Asl.Child);

94             if ((Op->Asl.Child->Asl.ParseOpcode == PARSEOP_METHODCALL) &&
95                 (ThisNodeType == (ACPI_UINT32_MAX - 1)))
96             {
97                 /*
98                  * The called method is untyped at this time (typically a
99                  * forward reference).
100                  *
101                  * Check for a recursive method call first.
102                  */
103                 if (Op->Asl.ParentMethod != Op->Asl.Child->Asl.Node->Op)
104                 {
105                     /* We must type the method here */

107                     TrWalkParseTree (Op->Asl.Child->Asl.Node->Op,
108                                     ASL_WALK_VISIT_UPWARD, NULL,
109                                     AnMethodTypingWalkEnd, NULL);

111                     ThisNodeType = AnGetBtype (Op->Asl.Child);
112                 }
113             }

115             /* Returns a value, save the value type */

117             if (Op->Asl.ParentMethod)
118             {
119                 Op->Asl.ParentMethod->Asl.AcpiBtype |= ThisNodeType;
120             }
121         }
122         break;

124     default:
```

```

126     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 == ARG_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 ARG_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 resource
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_DEREF))
340     {
341         break;
342     }

344     ThisNodeBtype = RequiredBtypes;
345     break;

348     case ARG_REFERENCE: /* References */
349     case ARG_INTEGER_REF:
350     case ARG_OBJECT_REF:
351     case ARG_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 local
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, so 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:
391
392             break;
393
394     }
395     break;
396
397     case ARG1_INTEGER:
398     default:
399
400         break;
401     }
402
403     CommonBtypes = ThisNodeBtype & RequiredBtypes;
404
405     if (ArgOp->Asl.ParseOpcode == PARSEOP_METHODCALL)
406     {
407         if (AnIsInternalMethod (ArgOp))
408         {
409             return (AE_OK);
410         }
411     }
412
413     /* Check a method call for a valid return value */
414
415     AnCheckMethodReturnValue (Op, OpInfo, ArgOp,
416                             RequiredBtypes, ThisNodeBtype);
417 }
418
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 */
426
427     AnFormatBtype (StringBuffer, ThisNodeBtype);
428     AnFormatBtype (StringBuffer2, RequiredBtypes);
429
430     sprintf (MsgBuffer, "[%s] found, %s operator requires [%s]",
431             StringBuffer, OpInfo->Name, StringBuffer2);
432
433     AslError (ASL_ERROR, ASL_MSG_INVALID_TYPE, ArgOp, MsgBuffer);
434 }
435
436     NextArgument:
437     ArgOp = ArgOp->Asl.Next;
438     INCREMENT_ARG_LIST (RuntimeArgTypes2);
439 }
440 break;
441
442 default:
443
444     break;
445 }
446
447 return (AE_OK);
448 }
449
450 /*****
451 *
452 * FUNCTION:    AnOtherSemanticAnalysisWalkBegin
453 *
454 * PARAMETERS:  ASL_WALK_CALLBACK
455 */

```

```

456 *
457 * RETURN:      Status
458 *
459 * DESCRIPTION: Descending callback for the analysis walk. Checks for
460 *               miscellaneous issues in the code.
461 *
462 *****/
463
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;
474
475     OpInfo = AcpiPsGetOpcodeInfo (Op->Asl.AmlOpcode);
476
477     /*
478     * Determine if an execution class operator actually does something by
479     * checking if it has a target and/or the function return value is used.
480     * (Target is optional, so a standalone statement can actually do nothing.)
481     */
482     if ((OpInfo->Class == AML_CLASS_EXECUTE) &&
483         (OpInfo->Flags & AML_HAS_RETURN) &&
484         (!AnIsResultUsed (Op)))
485     {
486         if (OpInfo->Flags & AML_HAS_TARGET)
487         {
488             /*
489             * Find the target node, it is always the last child. If the target
490             * is not specified in the ASL, a default node of type Zero was
491             * created by the parser.
492             */
493             ArgNode = Op->Asl.Child;
494             while (ArgNode->Asl.Next)
495             {
496                 PrevArgNode = ArgNode;
497                 ArgNode = ArgNode->Asl.Next;
498             }
499
500             /* Divide() is the only weird case, it has two targets */
501
502             if (Op->Asl.AmlOpcode == AML_DIVIDE_OP)
503             {
504                 if ((ArgNode->Asl.ParseOpcode == PARSEOP_ZERO) &&
505                     (PrevArgNode->Asl.ParseOpcode == PARSEOP_ZERO))
506                 {
507                     AslError (ASL_ERROR, ASL_MSG_RESULT_NOT_USED,
508                             Op, Op->Asl.ExternalName);
509                 }
510             }
511             else if (ArgNode->Asl.ParseOpcode == PARSEOP_ZERO)
512             {
513                 AslError (ASL_ERROR, ASL_MSG_RESULT_NOT_USED,
514                         Op, Op->Asl.ExternalName);
515             }
516         }
517     }
518     else
519     {
520         /*
521         *
522         */

```

```

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:
530
531         break;
532
533     default:
534
535         AslError (ASL_ERROR, ASL_MSG_RESULT_NOT_USED,
536                 Op, Op->Asl.ExternalName);
537         break;
538     }
539 }
540
541 /*
542  * Semantic checks for individual ASL operators
543  */
544 switch (Op->Asl.ParseOpcode)
545 {
546 case PARSEOP_ACQUIRE:
547 case PARSEOP_WAIT:
548     /*
549      * Emit a warning if the timeout parameter for these operators is not
550      * ACPI_WAIT_FOREVER, and the result value from the operator is not
551      * checked, meaning that a timeout could happen, but the code
552      * would not know about it.
553      */
554
555     /* First child is the namepath, 2nd child is timeout */
556
557     ArgNode = Op->Asl.Child;
558     ArgNode = ArgNode->Asl.Next;
559
560     /*
561      * Check for the WAIT_FOREVER case - defined by the ACPI spec to be
562      * 0xFFFF or greater
563      */
564     if (((ArgNode->Asl.ParseOpcode == PARSEOP_WORDCONST) ||
565         (ArgNode->Asl.ParseOpcode == PARSEOP_INTEGER)) &&
566         (ArgNode->Asl.Value.Integer >= (UINT64) ACPI_WAIT_FOREVER))
567     {
568         break;
569     }
570
571     /*
572      * The operation could timeout. If the return value is not used
573      * (indicates timeout occurred), issue a warning
574      */
575     if (!AnIsResultUsed (Op))
576     {
577         AslError (ASL_WARNING, ASL_MSG_TIMEOUT, ArgNode,
578                 Op->Asl.ExternalName);
579     }
580     break;
581
582 case PARSEOP_CREATEFIELD:
583     /*
584      * Check for a zero Length (NumBits) operand. NumBits is the 3rd operand
585      */

```

```

588     ArgNode = Op->Asl.Child;
589     ArgNode = ArgNode->Asl.Next;
590     ArgNode = ArgNode->Asl.Next;
591
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;
599
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 */
608     if (!Node)
609     {
610         break;
611     }
612
613     ArgNode = Node->Op; /* OpRegion definition */
614     ArgNode = ArgNode->Asl.Child; /* First child is the OpRegion Name */
615     ArgNode = ArgNode->Asl.Next; /* Next peer is the SPACE_ID (what we wa
616
617     /*
618      * The Connection() operator is only valid for the following operation
619      * region SpaceIds: GeneralPurposeIo and GenericSerialBus.
620      */
621     if ((ArgNode->Asl.Value.Integer != ACPI_ADR_SPACE_GPIO) &&
622         (ArgNode->Asl.Value.Integer != ACPI_ADR_SPACE_GSBUS))
623     {
624         AslError (ASL_ERROR, ASL_MSG_CONNECTION_INVALID, Op, NULL);
625     }
626     break;
627
628 case PARSEOP_FIELD:
629     /*
630      * Ensure that fields for GeneralPurposeIo and GenericSerialBus
631      * contain at least one Connection() operator
632      */
633     ArgNode = Op->Asl.Child; /* 1st child is the OpRegion Name */
634     Node = ArgNode->Asl.Node; /* OpRegion namespace node */
635     if (!Node)
636     {
637         break;
638     }
639
640     ArgNode = Node->Op; /* OpRegion definition */
641     ArgNode = ArgNode->Asl.Child; /* First child is the OpRegion Name */
642     ArgNode = ArgNode->Asl.Next; /* Next peer is the SPACE_ID (what we wa
643
644     /* We are only interested in GeneralPurposeIo and GenericSerialBus */
645
646     if ((ArgNode->Asl.Value.Integer != ACPI_ADR_SPACE_GPIO) &&
647         (ArgNode->Asl.Value.Integer != ACPI_ADR_SPACE_GSBUS))
648     {
649         break;
650     }
651
652     ArgNode = Op->Asl.Child; /* 1st child is the OpRegion Name */
653     ArgNode = ArgNode->Asl.Next; /* AccessType */

```

```
654     ArgNode = ArgNode->Asl.Next;    /* LockRule */
655     ArgNode = ArgNode->Asl.Next;    /* UpdateRule */
656     ArgNode = ArgNode->Asl.Next;    /* Start of FieldUnitList */

658     /* Walk the FieldUnitList */

660     while (ArgNode)
661     {
662         if (ArgNode->Asl.ParseOpcode == PARSEOP_CONNECTION)
663         {
664             break;
665         }
666         else if (ArgNode->Asl.ParseOpcode == PARSEOP_NAMESEG)
667         {
668             AslError (ASL_ERROR, ASL_MSG_CONNECTION_MISSING, ArgNode, NULL);
669             break;
670         }

672         ArgNode = ArgNode->Asl.Next;
673     }
674     break;

676 default:
678     break;
679 }

681 return (AE_OK);
682 }
```

new/usr/src/common/acpica/compiler/aslxref.c

1

```
*****
27654 Fri Jan 17 07:29:42 2014
new/usr/src/common/acpica/compiler/aslxref.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: aslxref - Namespace cross-reference
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "aslcompiler.y.h"
47 #include "acparser.h"
48 #include "amlcode.h"
49 #include "acnamesp.h"
50 #include "acdispat.h"

53 #define _COMPONENT          ACPI_COMPILER
54         ACPI_MODULE_NAME    ("aslxref")

56 /* Local prototypes */

58 static ACPI_STATUS
59 XfNamespaceLocateBegin (
```

new/usr/src/common/acpica/compiler/aslxref.c

2

```
60     ACPI_PARSE_OBJECT      *Op,
61     UINT32                  Level,
62     void                     *Context);

64 static ACPI_STATUS
65 XfNamespaceLocateEnd (
66     ACPI_PARSE_OBJECT      *Op,
67     UINT32                  Level,
68     void                     *Context);

70 static BOOLEAN
71 XfObjectExists (
72     char                     *Name);

74 static ACPI_STATUS
75 XfCompareOneNamespaceObject (
76     ACPI_HANDLE             ObjHandle,
77     UINT32                  Level,
78     void                     *Context,
79     void                     **ReturnValue);

81 static void
82 XfCheckFieldRange (
83     ACPI_PARSE_OBJECT      *Op,
84     UINT32                  RegionBitLength,
85     UINT32                  FieldBitOffset,
86     UINT32                  FieldBitLength,
87     UINT32                  AccessBitWidth);

90 /*****
91 *
92 * FUNCTION:      XfCrossReferenceNamespace
93 *
94 * PARAMETERS:    None
95 *
96 * RETURN:        Status
97 *
98 * DESCRIPTION:   Perform a cross reference check of the parse tree against the
99 *                namespace. Every named referenced within the parse tree
100 *                should be get resolved with a namespace lookup. If not, the
101 *                original reference in the ASL code is invalid -- i.e., refers
102 *                to a non-existent object.
103 *
104 * NOTE:          The ASL "External" operator causes the name to be inserted into the
105 *                namespace so that references to the external name will be resolved
106 *                correctly here.
107 *
108 *****/

110 ACPI_STATUS
111 XfCrossReferenceNamespace (
112     void)
113 {
114     ACPI_WALK_STATE          *WalkState;

117     DbgPrint (ASL_DEBUG_OUTPUT, "\nCross referencing namespace\n\n");

119     /*
120     * Create a new walk state for use when looking up names
121     * within the namespace (Passed as context to the callbacks)
122     */
123     WalkState = AcpiDsCreateWalkState (0, NULL, NULL, NULL);
124     if (!WalkState)
125     {
```

```

126     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 */
261     AslError (ASL_ERROR, ASL_MSG_FIELD_UNIT_ACCESS_WIDTH, Op, NULL);
262 }
263 }

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 *****/

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;

308     ACPI_FUNCTION_TRACE_PTR (XfNamespaceLocateBegin, Op);

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     }

322     /* We are only interested in opcodes that have an associated name */

```

```

324     OpInfo = AcpiPsGetOpcodeInfo (Op->Asl.AmlOpcode);

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     }

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     }

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     }

362     /* Get the NamePath from the appropriate place */

364     if (OpInfo->Flags & AML_NAMED)
365     {
366         /* For nearly all NAMED operators, the name reference is the first child

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 */

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 }

394 ObjectType = AslMapNamedOpcodeToDataType (Op->Asl.AmlOpcode);
395 ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
396     "Type=%s\n", AcpiUtGetTypeName (ObjectType)));

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++;

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 */

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 */

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 */

442             if (Path[0] == AML_ROOT_PREFIX)
443             {
444                 /* Gave full path, the object does not exist */

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 }

460     Status = AE_OK;
461 }

463     return ACPI_STATUS (Status);
464 }

466 /* Check for a reference vs. name declaration */

468 if (!(OpInfo->Flags & AML_NAMED) &&
469     !(OpInfo->Flags & AML_CREATE))
470 {
471     /* This node has been referenced, mark it for reference check */

473     Node->Flags |= ANOBJ_IS_REFERENCED;
474 }

476 /* Attempt to optimize the NamePath */

478 OptOptimizeNamePath (Op, OpInfo->Flags, WalkState, Path, Node);

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 */

489     NextOp = Node->Op;

491     /* The first child is the alias target op */

493     NextOp = NextOp->Asl.Child;

495     /* That in turn points back to original target alias node */

497     if (NextOp->Asl.Node)
498     {
499         Node = NextOp->Asl.Node;
500     }

502     /* Else - forward reference to alias, will be resolved later */
503 }

505 /* 2) Check for a reference to a resource descriptor */

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;

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;
535
536     case AML_CREATE_BIT_FIELD_OP:
537
538         FieldBitLength = 1;
539         break;
540
541     case AML_CREATE_BYTE_FIELD_OP:
542     case AML_INDEX_OP:
543
544         FieldBitLength = 8;
545         break;
546
547     case AML_CREATE_WORD_FIELD_OP:
548
549         FieldBitLength = 16;
550         break;
551
552     case AML_CREATE_DWORD_FIELD_OP:
553
554         FieldBitLength = 32;
555         break;
556
557     case AML_CREATE_QWORD_FIELD_OP:
558
559         FieldBitLength = 64;
560         break;
561
562     default:
563
564         FieldBitLength = 0;
565         break;
566     }
567
568     /* Check the field length against the length of the resource tag */
569
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         }
580
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" : "");
586
587             AslError (ASL_WARNING, Message, Op, MsgBuffer);

```

```

588     }
589 }
590
591 /* Convert the BitOffset to a ByteOffset for certain opcodes */
592
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:
600
601         Offset = ACPI_DIV_8 (Offset);
602         break;
603
604     default:
605
606         break;
607 }
608
609 /* Now convert this node to an integer whose value is the field offset */
610
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;
615
616 OpcGenerateAmlOpcode (Op);
617 }
618
619 /* 3) Check for a method invocation */
620
621 else if (((Op->Asl.ParseOpcode == PARSEOP_NAMESTRING) || (Op->Asl.ParseOpcode == PARSEOP_METHODCALL)) ||
622     ((Node->Type == ACPI_TYPE_METHOD) &&
623     (Op->Asl.Parent) &&
624     (Op->Asl.Parent->Asl.ParseOpcode != PARSEOP_METHOD))) ||
625     ((Op->Asl.ParseOpcode == PARSEOP_METHODCALL))
626 {
627     /*
628      * A reference to a method within one of these opcodes is not an
629      * invocation of the method, it is simply a reference to the method.
630      */
631     if ((Op->Asl.Parent) &&
632         ((Op->Asl.Parent->Asl.ParseOpcode == PARSEOP_REFOF) ||
633         (Op->Asl.Parent->Asl.ParseOpcode == PARSEOP_DEREF_OF) ||
634         (Op->Asl.Parent->Asl.ParseOpcode == PARSEOP_OBJECTTYPE)))
635     {
636         return ACPI_STATUS (AE_OK);
637     }
638     /*
639      * There are two types of method invocation:
640      * 1) Invocation with arguments -- the parser recognizes this
641      *    as a METHODCALL.
642      * 2) Invocation with no arguments -- the parser cannot determine that
643      *    this is a method invocation, therefore we have to figure it out
644      *    here.
645      */
646     if (Node->Type != ACPI_TYPE_METHOD)
647     {
648         sprintf (MsgBuffer, "%s is a %s",
649             Op->Asl.ExternalName, AcpiUtGetTypeName (Node->Type));
650
651         AslError (ASL_ERROR, ASL_MSG_NOT_METHOD, Op, MsgBuffer);
652         return ACPI_STATUS (AE_OK);

```

```

654     }

655     /* 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:
738                 MinimumLength = 1;
739                 break;

742             case AML_FIELD_ACCESS_WORD:
743                 MinimumLength = 2;
744                 break;

747             case AML_FIELD_ACCESS_DWORD:
748                 MinimumLength = 4;
749                 break;

752             case AML_FIELD_ACCESS_QWORD:
753                 MinimumLength = 8;
754                 break;
755         }

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_ACCESS_
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);
855     /* 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 }

```

new/usr/src/common/acpica/compiler/dtcompile.c

1

```
*****
16440 Fri Jan 17 07:29:42 2014
new/usr/src/common/acpica/compiler/dtcompile.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dtcompile.c - Front-end for data table compiler
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define __DT_COMPILE_C__
45 #define _DECLARE_DT_GLOBALS_
46
47 #include "aslcompiler.h"
48 #include "dtcompiler.h"
49
50 #define _COMPONENT DT_COMPILER
51 ACPI_MODULE_NAME ("dtcompile")
52
53 static char VersionString[9];
54
55 /* Local prototypes */
56
57 static ACPI_STATUS
```

new/usr/src/common/acpica/compiler/dtcompile.c

2

```
59 DtInitialize (
60     void);
61
62 static ACPI_STATUS
63 DtCompileDataTable (
64     DT_FIELD **Field);
65
66 static void
67 DtInsertCompilerIds (
68     DT_FIELD *FieldList);
69
70
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 *****/
85
86 ACPI_STATUS
87 DtDoCompile (
88     void)
89 {
90     ACPI_STATUS Status;
91     UINT8 Event;
92     DT_FIELD *FieldList;
93
94     /* Initialize globals */
95
96     Status = DtInitialize ();
97     if (ACPI_FAILURE (Status))
98     {
99         printf ("Error during compiler initialization, 0x%X\n", Status);
100         return (Status);
101     }
102
103     /* Preprocessor */
104
105     Event = UtBeginEvent ("Preprocess input file");
106     PrDoPreprocess ();
107     UtEndEvent (Event);
108
109     if (Gbl_PreprocessOnly)
110     {
111         return (AE_OK);
112     }
113
114     /*
115     * Scan the input file (file is already open) and
116     * build the parse tree
117     */
118     Event = UtBeginEvent ("Scan and parse input file");
119     FieldList = DtScanFile (Gbl_Files[ASL_FILE_INPUT].Handle);
120     UtEndEvent (Event);
121
122     /* 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  * FUNCTION:      DtInitialize
180  *
181  * PARAMETERS:    None
182  *
183  * RETURN:        Status
184  *
185  * DESCRIPTION:   Initialize data table compiler globals. Enables multiple
186  *               compiles per invocation.
187  *
188  *
189  *****/

```

```

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  * FUNCTION:      DtInsertCompilerIds
221  *
222  * PARAMETERS:    FieldList          - Current field list pointer
223  *
224  * RETURN:        None
225  *
226  * DESCRIPTION:   Insert the IDs (Name, Version) of the current compiler into
227  *               the original ACPI table header.
228  *
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     }

```

```

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     *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 */
522
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  }

```



new/usr/src/common/acpica/compiler/dtcompiler.h

1

```
*****
13639 Fri Jan 17 07:29:42 2014
new/usr/src/common/acpica/compiler/dtcompiler.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dtcompiler.h - header for data table compiler
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define __DTCOMPILER_H__
45
46 #ifndef __DTCOMPILER
47 #define __DTCOMPILER
48
49 #include <stdio.h>
50 #include "acdisasm.h"
51
52
53 #undef DT_EXTERN
54
55 #ifdef _DECLARE_DT_GLOBALS
56 #define DT_EXTERN
57 #define DT_INIT_GLOBAL(a,b) (a)=(b)
58 #else
```

new/usr/src/common/acpica/compiler/dtcompiler.h

2

```
59 #define DT_EXTERN extern
60 #define DT_INIT_GLOBAL(a,b) (a)
61 #endif
62
63
64 /* Types for individual fields (one per input line) */
65
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
77
78
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;
94
95 } DT_FIELD;
96
97 /* Flags for above */
98
99 #define DT_FIELD_NOT_ALLOCATED 1
100
101
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;
118
119 } DT_SUBTABLE;
120
121
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     DT_FIELD          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 /* dtttable - 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
393     **PFieldList);

394 ACPI_STATUS
395 DtCompileCpep (
396     void
397     **PFieldList);

398 ACPI_STATUS
399 DtCompileCsrt (
400     void
401     **PFieldList);

402 ACPI_STATUS
403 DtCompileDbg2 (
404     void
405     **PFieldList);

406 ACPI_STATUS
407 DtCompileDmar (
408     void
409     **PFieldList);

410 ACPI_STATUS
411 DtCompileEinj (
412     void
413     **PFieldList);

414 ACPI_STATUS
415 DtCompileErst (
416     void
417     **PFieldList);

418 ACPI_STATUS
419 DtCompileFadt (
420     void
421     **PFieldList);

422 ACPI_STATUS
423 DtCompileFpdt (
424     void
425     **PFieldList);

426 ACPI_STATUS
427 DtCompileHest (
428     void
429     **PFieldList);

430 ACPI_STATUS
431 DtCompileIvrs (
432     void
433     **PFieldList);

434 ACPI_STATUS
435 DtCompileMadt (
436     void
437     **PFieldList);

438 ACPI_STATUS
439 DtCompileMcfg (
440     void
441     **PFieldList);

442 ACPI_STATUS
443 DtCompileMpst (
444     void
445     **PFieldList);

446 ACPI_STATUS
447 DtCompileMsct (
448     void
449     **PFieldList);

450 ACPI_STATUS
451 DtCompileMtmr (
452     void
453     **PFieldList);

454 ACPI_STATUS

```

```

455 DtCompilePmtt (
456     void
457     **PFieldList);

458 ACPI_STATUS
459 DtCompilePcct (
460     void
461     **PFieldList);

462 ACPI_STATUS
463 DtCompileRsdt (
464     void
465     **PFieldList);

466 ACPI_STATUS
467 DtCompileS3pt (
468     DT_FIELD
469     **PFieldList);

470 ACPI_STATUS
471 DtCompileSlic (
472     void
473     **PFieldList);

474 ACPI_STATUS
475 DtCompileSlit (
476     void
477     **PFieldList);

478 ACPI_STATUS
479 DtCompileSrat (
480     void
481     **PFieldList);

482 ACPI_STATUS
483 DtCompileUefi (
484     void
485     **PFieldList);

486 ACPI_STATUS
487 DtCompileVrtc (
488     void
489     **PFieldList);

490 ACPI_STATUS
491 DtCompileWdat (
492     void
493     **PFieldList);

494 ACPI_STATUS
495 DtCompileXsdt (
496     void
497     **PFieldList);

498 ACPI_STATUS
499 DtCompileGeneric (
500     void
501     **PFieldList);

502 ACPI_DMTABLE_INFO *
503 DtGetGenericTableInfo (
504     char
505     *Name);

506 /* ACPI Table templates */

507 extern const unsigned char TemplateAsf[];
508 extern const unsigned char TemplateBoot[];
509 extern const unsigned char TemplateBert[];
510 extern const unsigned char TemplateBgrrt[];
511 extern const unsigned char TemplateCpep[];
512 extern const unsigned char TemplateCsrt[];
513 extern const unsigned char TemplateDbg2[];
514 extern const unsigned char TemplateDbgpl[];
515 extern const unsigned char TemplateDmar[];
516 extern const unsigned char TemplateEcdrv[];
517 extern const unsigned char TemplateEinj[];
518 extern const unsigned char TemplateErst[];
519 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 TemplateSpocr[];
540 extern const unsigned char TemplateSpmi[];
541 extern const unsigned char TemplateSrat[];
542 extern const unsigned char TemplateTcpc[];
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
```

new/usr/src/common/acpica/compiler/dtexpress.c

1

```
*****
10905 Fri Jan 17 07:29:43 2014
new/usr/src/common/acpica/compiler/dtexpress.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dtexpress.c - Support for integer expressions and labels
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __DTEXPRESS_C__

46 #include "aslcompiler.h"
47 #include "dtcompiler.h"
48 #include "dtparser.y.h"

50 #define _COMPONENT DT_COMPILER
51 ACPI_MODULE_NAME ("dtexpress")

54 /* Local prototypes */

56 static void
57 DtInsertLabelField (
58     DT_FIELD *Field);
```

new/usr/src/common/acpica/compiler/dtexpress.c

2

```
60 static DT_FIELD *
61 DtLookupLabel (
62     char *Name);

64 /* Global used for errors during parse and related functions */

66 DT_FIELD *Gbl_CurrentField;

69 /*****
70 *
71 * FUNCTION: DtResolveIntegerExpression
72 *
73 * PARAMETERS: Field - Field object with Integer expression
74 *              ReturnValue - Where the integer is returned
75 *
76 * RETURN: Status, and the resolved 64-bit integer value
77 *
78 * DESCRIPTION: Resolve an integer expression to a single value. Supports
79 *              both integer constants and labels.
80 *
81 *****/

83 ACPI_STATUS
84 DtResolveIntegerExpression (
85     DT_FIELD *Field,
86     UINT64 *ReturnValue)
87 {
88     UINT64 Result;

91     DbgPrint (ASL_DEBUG_OUTPUT, "Full Integer expression: %s\n",
92         Field->Value);

94     Gbl_CurrentField = Field;

96     Result = DtEvaluateExpression (Field->Value);
97     *ReturnValue = Result;
98     return (AE_OK);
99 }

102 /*****
103 *
104 * FUNCTION: DtDoOperator
105 *
106 * PARAMETERS: LeftValue - First 64-bit operand
107 *              Operator - Parse token for the operator (EXPOP_*)
108 *              RightValue - Second 64-bit operand
109 *
110 * RETURN: 64-bit result of the requested operation
111 *
112 * DESCRIPTION: Perform the various 64-bit integer math functions
113 *
114 *****/

116 UINT64
117 DtDoOperator (
118     UINT64 LeftValue,
119     UINT32 Operator,
120     UINT64 RightValue)
121 {
122     UINT64 Result;

125     /* Perform the requested operation */
```

```

127     switch (Operator)
128     {
129     case EXPOP_ONES_COMPLIMENT:
130
131         Result = ~RightValue;
132         break;
133
134     case EXPOP_LOGICAL_NOT:
135
136         Result = !RightValue;
137         break;
138
139     case EXPOP_MULTIPLY:
140
141         Result = LeftValue * RightValue;
142         break;
143
144     case EXPOP_DIVIDE:
145
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;
154
155     case EXPOP_MODULO:
156
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;
165
166     case EXPOP_ADD:
167         Result = LeftValue + RightValue;
168         break;
169
170     case EXPOP_SUBTRACT:
171
172         Result = LeftValue - RightValue;
173         break;
174
175     case EXPOP_SHIFT_RIGHT:
176
177         Result = LeftValue >> RightValue;
178         break;
179
180     case EXPOP_SHIFT_LEFT:
181
182         Result = LeftValue << RightValue;
183         break;
184
185     case EXPOP_LESS:
186
187         Result = LeftValue < RightValue;
188         break;
189
190     case EXPOP_GREATER:

```

```

192         Result = LeftValue > RightValue;
193         break;
194
195     case EXPOP_LESS_EQUAL:
196
197         Result = LeftValue <= RightValue;
198         break;
199
200     case EXPOP_GREATER_EQUAL:
201
202         Result = LeftValue >= RightValue;
203         break;
204
205     case EXPOP_EQUAL:
206
207         Result = LeftValue == RightValue;
208         break;
209
210     case EXPOP_NOT_EQUAL:
211
212         Result = LeftValue != RightValue;
213         break;
214
215     case EXPOP_AND:
216
217         Result = LeftValue & RightValue;
218         break;
219
220     case EXPOP_XOR:
221
222         Result = LeftValue ^ RightValue;
223         break;
224
225     case EXPOP_OR:
226
227         Result = LeftValue | RightValue;
228         break;
229
230     case EXPOP_LOGICAL_AND:
231
232         Result = LeftValue && RightValue;
233         break;
234
235     case EXPOP_LOGICAL_OR:
236
237         Result = LeftValue || RightValue;
238         break;
239
240     default:
241
242         /* Unknown operator */
243
244         DtFatal (ASL_MSG_INVALID_EXPRESSION,
245                 Gbl_CurrentField, NULL);
246         return (0);
247     }
248
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));
255
256     return (Result);
257 }

```

```

260 /*****
261 *
262 * FUNCTION:      DtResolveLabel
263 *
264 * PARAMETERS:   LabelString          - Contains the label
265 *
266 * RETURN:       Table offset associated with the label
267 *
268 * DESCRIPTION:  Lookup a label and return its value.
269 *
270 *****/
272 UINT64
273 DtResolveLabel (
274     char                *LabelString)
275 {
276     DT_FIELD            *LabelField;

279     DbgPrint (ASL_DEBUG_OUTPUT, "Resolve Label: %s\n", LabelString);

281     /* Resolve a label reference to an integer (table offset) */

283     if (*LabelString != '$')
284     {
285         return (0);
286     }

288     LabelField = DtLookupLabel (LabelString);
289     if (!LabelField)
290     {
291         DtError (ASL_ERROR, ASL_MSG_UNKNOWN_LABEL,
292                 Gbl_CurrentField, LabelString);
293         return (0);
294     }

296     /* All we need from the label is the offset in the table */

298     DbgPrint (ASL_DEBUG_OUTPUT, "Resolved Label: 0x%8.8X\n",
299             LabelField->TableOffset);

301     return (LabelField->TableOffset);
302 }

305 /*****
306 *
307 * FUNCTION:      DtDetectAllLabels
308 *
309 * PARAMETERS:   FieldList          - Field object at start of generic list
310 *
311 * RETURN:       None
312 *
313 * DESCRIPTION:  Detect all labels in a list of "generic" opcodes (such as
314 *              a ACPI table.) and insert them into the global label list.
315 *
316 *****/
318 void
319 DtDetectAllLabels (
320     DT_FIELD            *FieldList)
321 {
322     ACPI_DMTABLE_INFO    *Info;
323     DT_FIELD            *GenericField;

```

```

324     UINT32              TableOffset;

327     TableOffset = Gbl_CurrentTableOffset;
328     GenericField = FieldList;

330     /*
331     * Process all "Label:" fields within the parse tree. We need
332     * to know the offsets for all labels before we can compile
333     * the parse tree in order to handle forward references. Traverse
334     * tree and get/set all field lengths of all operators in order to
335     * determine the label offsets.
336     */
337     while (GenericField)
338     {
339         Info = DtGetGenericTableInfo (GenericField->Name);
340         if (Info)
341         {
342             /* Maintain table offsets */

344             GenericField->TableOffset = TableOffset;
345             TableOffset += DtGetFieldLength (GenericField, Info);

347             /* Insert all labels in the global label list */

349             if (Info->Opcode == ACPI_DMT_LABEL)
350             {
351                 DtInsertLabelField (GenericField);
352             }
353         }

355         GenericField = GenericField->Next;
356     }
357 }

360 /*****
361 *
362 * FUNCTION:      DtInsertLabelField
363 *
364 * PARAMETERS:   Field          - Field object with Label to be inserted
365 *
366 * RETURN:       None
367 *
368 * DESCRIPTION:  Insert a label field into the global label list
369 *
370 *****/
372 static void
373 DtInsertLabelField (
374     DT_FIELD            *Field)
375 {
377     DbgPrint (ASL_DEBUG_OUTPUT,
378             "DtInsertLabelField: Found Label : %s at output table offset %X\n",
379             Field->Value, Field->TableOffset);

381     Field->NextLabel = Gbl_LabelList;
382     Gbl_LabelList = Field;
383 }

386 /*****
387 *
388 * FUNCTION:      DtLookupLabel
389 *

```



```
390 * 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 }
```

new/usr/src/common/acpica/compiler/dtfield.c

1

```
*****
15206 Fri Jan 17 07:29:43 2014
new/usr/src/common/acpica/compiler/dtfield.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dtfield.c - Code generation for individual source fields
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __DTFIELD_C__
46 #include "aslcompiler.h"
47 #include "dtcompiler.h"

49 #define _COMPONENT DT_COMPILER
50 #define ACPI_MODULE_NAME ("dtfield")

53 /* Local prototypes */

55 static void
56 DtCompileString (
57     UINT8 *Buffer,
58     DT_FIELD *Field,
```

new/usr/src/common/acpica/compiler/dtfield.c

2

```
59     UINT32 ByteLength);

61 static void
62 DtCompileUnicode (
63     UINT8 *Buffer,
64     DT_FIELD *Field,
65     UINT32 ByteLength);

67 static ACPI_STATUS
68 DtCompileUuid (
69     UINT8 *Buffer,
70     DT_FIELD *Field,
71     UINT32 ByteLength);

73 static char *
74 DtNormalizeBuffer (
75     char *Buffer,
76     UINT32 *Count);

79 /*****
80 *
81 * FUNCTION: DtCompileOneField
82 *
83 * PARAMETERS: Buffer - Output buffer
84 *              Field - Field to be compiled
85 *              ByteLength - Byte length of the field
86 *              Type - Field type
87 *
88 * RETURN: None
89 *
90 * DESCRIPTION: Compile a field value to binary
91 *
92 *****/

94 void
95 DtCompileOneField (
96     UINT8 *Buffer,
97     DT_FIELD *Field,
98     UINT32 ByteLength,
99     UINT8 Type,
100     UINT8 Flags)
101 {
102     ACPI_STATUS Status;

104     switch (Type)
105     {
106     case DT_FIELD_TYPE_INTEGER:

108         DtCompileInteger (Buffer, Field, ByteLength, Flags);
109         break;

111     case DT_FIELD_TYPE_STRING:

113         DtCompileString (Buffer, Field, ByteLength);
114         break;

116     case DT_FIELD_TYPE_UUID:

118         Status = DtCompileUuid (Buffer, Field, ByteLength);
119         if (ACPI_SUCCESS (Status))
120         {
121             break;
122         }

124     /* Fall through. */
```

```

126     case DT_FIELD_TYPE_BUFFER:
127
128         DtCompileBuffer (Buffer, Field->Value, Field, ByteLength);
129         break;
130
131     case DT_FIELD_TYPE_UNICODE:
132
133         DtCompileUnicode (Buffer, Field, ByteLength);
134         break;
135
136     case DT_FIELD_TYPE_DEVICE_PATH:
137
138         break;
139
140     default:
141
142         DtFatal (ASL_MSG_COMPILER_INTERNAL, Field, "Invalid field type");
143         break;
144     }
145 }
146
147 /*****
148  *
149  * FUNCTION:      DtCompileString
150  *
151  * PARAMETERS:    Buffer          - Output buffer
152  *                Field          - String to be copied to buffer
153  *                ByteLength     - Maximum length of string
154  *
155  * RETURN:        None
156  *
157  * DESCRIPTION:   Copy string to the buffer
158  *
159  *****/
160
161 static void
162 DtCompileString (
163     UINT8          *Buffer,
164     DT_FIELD       *Field,
165     UINT32         ByteLength)
166 {
167     UINT32         Length;
168
169     Length = ACPI_STRLEN (Field->Value);
170
171     /* Check if the string is too long for the field */
172
173     if (Length > ByteLength)
174     {
175         sprintf (MsgBuffer, "Maximum %u characters", ByteLength);
176         DtError (ASL_ERROR, ASL_MSG_STRING_LENGTH, Field, MsgBuffer);
177         Length = ByteLength;
178     }
179
180     ACPI_MEMCPY (Buffer, Field->Value, Length);
181 }
182
183 /*****
184  *
185  * FUNCTION:      DtCompileUnicode
186  *
187  * PARAMETERS:    Buffer          - Output buffer
188  *****/

```

```

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  *****/
202
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;
213
214     AsciiString = Field->Value;
215     UnicodeString = (UINT16 *) Buffer;
216     Count = ACPI_STRLEN (AsciiString) + 1;
217
218     /* Convert to Unicode string (including null terminator) */
219
220     for (i = 0; i < Count; i++)
221     {
222         UnicodeString[i] = (UINT16) AsciiString[i];
223     }
224 }
225
226 /*****
227  *
228  * FUNCTION:      DtCompileUuid
229  *
230  * PARAMETERS:    Buffer          - Output buffer
231  *                Field          - String to be copied to buffer
232  *                ByteLength     - Maximum length of string
233  *
234  * RETURN:        None
235  *
236  * DESCRIPTION:   Convert UUID string to 16-byte buffer
237  *
238  *****/
239
240 static ACPI_STATUS
241 DtCompileUuid (
242     UINT8          *Buffer,
243     DT_FIELD       *Field,
244     UINT32         ByteLength)
245 {
246     char           *InString;
247     ACPI_STATUS    Status;
248
249     InString = Field->Value;
250
251     Status = AuValidateUuid (InString);
252     if (ACPI_FAILURE (Status))
253     {

```

```

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 }
264
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:
525
526         BitPosition = Info->Opcode;
527         BitLength = 1;
528         break;
529
530     case ACPI_DMT_FLAGS0:
531
532         BitPosition = 0;
533         BitLength = 2;
534         break;
535
536     case ACPI_DMT_FLAGS1:
537
538         BitPosition = 1;
539         BitLength = 2;
540         break;
541
542     case ACPI_DMT_FLAGS2:
543
544         BitPosition = 2;
545         BitLength = 2;
546         break;
547
548     case ACPI_DMT_FLAGS4:
549
550         BitPosition = 4;
551         BitLength = 2;
552         break;
553
554     default:
555
556         DtFatal (ASL_MSG_COMPILER_INTERNAL, Field, "Invalid flag opcode");
557         break;
558     }
559
560     /* Check range of the input flag value */
561
562     if (Value >= ((UINT64) 1 << BitLength))
563     {
564         sprintf (MsgBuffer, "Maximum %u bit", BitLength);
565         DtError (ASL_ERROR, ASL_MSG_FLAG_VALUE, Field, MsgBuffer);
566         Value = 0;
567     }
568
569     *Buffer |= (UINT8) (Value << BitPosition);
570 }
```

new/usr/src/common/acpica/compiler/dtio.c

1

```
*****
28481 Fri Jan 17 07:29:43 2014
new/usr/src/common/acpica/compiler/dtio.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dtio.c - File I/O support for data table compiler
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __DTIO_C__

46 #include "aslcompiler.h"
47 #include "dtcompiler.h"

49 #define _COMPONENT DT_COMPILER
50 ACPI_MODULE_NAME ("dtio")

53 /* Local prototypes */

55 static char *
56 DtTrim (
57     char *String);

59 static void
```

new/usr/src/common/acpica/compiler/dtio.c

2

```
60 DtLinkField (
61     DT_FIELD *Field);

63 static ACPI_STATUS
64 DtParseLine (
65     char *LineBuffer,
66     UINT32 Line,
67     UINT32 Offset);

69 static void
70 DtWriteBinary (
71     DT_SUBTABLE *Subtable,
72     void *Context,
73     void *ReturnValue);

75 static void
76 DtDumpBuffer (
77     UINT32 FileId,
78     UINT8 *Buffer,
79     UINT32 Offset,
80     UINT32 Length);

82 static void
83 DtDumpSubtableInfo (
84     DT_SUBTABLE *Subtable,
85     void *Context,
86     void *ReturnValue);

88 static void
89 DtDumpSubtableTree (
90     DT_SUBTABLE *Subtable,
91     void *Context,
92     void *ReturnValue);

95 /* States for DtGetNextLine */

97 #define DT_NORMAL_TEXT 0
98 #define DT_START_QUOTED_STRING 1
99 #define DT_START_COMMENT 2
100 #define DT_SLASH_asterisk_COMMENT 3
101 #define DT_SLASH_SLASH_COMMENT 4
102 #define DT_END_COMMENT 5
103 #define DT_MERGE_LINES 6
104 #define DT_ESCAPE_SEQUENCE 7

106 static UINT32 Gbl_NextLineOffset;

109 /*****
110 *
111 * FUNCTION: DtTrim
112 *
113 * PARAMETERS: String - Current source code line to trim
114 *
115 * RETURN: Trimmed line. Must be freed by caller.
116 *
117 * DESCRIPTION: Trim left and right spaces
118 *
119 *****/

121 static char *
122 DtTrim (
123     char *String)
124 {
125     char *Start;
```

```

126 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             else if (*Start != ' ')
299             {
300                 break;
301             }
302             Start++;
303         }
304     }

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 }

```

```

392 /*****
393 *
394 * FUNCTION:      DtGetNextLine
395 *
396 * PARAMETERS:   Handle          - Open file handle for the source file
397 *
398 * RETURN:       Filled line buffer and offset of start-of-line (ASL_EOF on EOF)
399 *
400 * DESCRIPTION:  Get the next valid source line. Removes all comments.
401 *              Ignores empty lines.
402 *
403 * Handles both slash-asterisk and slash-slash comments.
404 * Also, quoted strings, but no escapes within.
405 *
406 * Line is returned in Gbl_CurrentLineBuffer.
407 * Line number in original file is returned in Gbl_CurrentLineNumber.
408 *
409 *****/
411 UINT32
412 DtGetNextLine (
413     FILE          *Handle)
414 {
415     BOOLEAN        LineNotAllBlanks = FALSE;
416     UINT32         State = DT_NORMAL_TEXT;
417     UINT32         CurrentLineOffset;
418     UINT32         i;
419     int            c;

422     for (i = 0; ; )
423     {
424         /*
425          * If line is too long, expand the line buffers. Also increases
426          * Gbl_LineBufferSize.
427          */
428         if (i >= Gbl_LineBufferSize)
429         {
430             UtExpandLineBuffers ();
431         }

433         c = getc (Handle);
434         if (c == EOF)
435         {
436             switch (State)
437             {
438                 case DT_START_QUOTED_STRING:
439                 case DT_SLASH_ASTERISK_COMMENT:

441                     AcpiOsPrintf ("**** EOF within comment/string %u\n", State);
442                     break;

444                 default:

446                     break;
447             }

449             /* Standalone EOF is OK */

451             if (i == 0)
452             {
453                 return (ASL_EOF);
454             }

```

```

456         /*
457          * Received an EOF in the middle of a line. Terminate the
458          * line with a newline. The next call to this function will
459          * return a standalone EOF. Thus, the upper parsing software
460          * never has to deal with an EOF within a valid line (or
461          * the last line does not get tossed on the floor.)
462          */
463         c = '\n';
464         State = DT_NORMAL_TEXT;
465     }

467     switch (State)
468     {
469     case DT_NORMAL_TEXT:

471         /* Normal text, insert char into line buffer */

473         Gbl_CurrentLineBuffer[i] = (char) c;
474         switch (c)
475         {
476         case '/':

478             State = DT_START_COMMENT;
479             break;

481         case '"':

483             State = DT_START_QUOTED_STRING;
484             LineNotAllBlanks = TRUE;
485             i++;
486             break;

488         case '\\':
489             /*
490              * The continuation char MUST be last char on this line.
491              * Otherwise, it will be assumed to be a valid ASL char.
492              */
493             State = DT_MERGE_LINES;
494             break;

496         case '\n':

498             CurrentLineOffset = Gbl_NextLineOffset;
499             Gbl_NextLineOffset = (UINT32) ftell (Handle);
500             Gbl_CurrentLineNumber++;

502             /*
503              * Exit if line is complete. Ignore empty lines (only \n)
504              * or lines that contain nothing but blanks.
505              */
506             if ((i != 0) && LineNotAllBlanks)
507             {
508                 if ((i + 1) >= Gbl_LineBufferSize)
509                 {
510                     UtExpandLineBuffers ();
511                 }

513                 Gbl_CurrentLineBuffer[i+1] = 0; /* Terminate string */
514                 return (CurrentLineOffset);
515             }

517             /* Toss this line and start a new one */

519             i = 0;
520             LineNotAllBlanks = FALSE;
521             break;

```

```

523     default:
524     {
525         if (c != ' ')
526         {
527             LineNotAllBlanks = TRUE;
528         }
529     }
530     i++;
531     break;
532 }
533 break;
534
535 case DT_START_QUOTED_STRING:
536
537     /* Insert raw chars until end of quoted string */
538
539     Gbl_CurrentLineBuffer[i] = (char) c;
540     i++;
541
542     switch (c)
543     {
544     case '"':
545         State = DT_NORMAL_TEXT;
546         break;
547
548     case '\\':
549         State = DT_ESCAPE_SEQUENCE;
550         break;
551
552     case '\n':
553         AcpiOsPrintf ("ERROR at line %u: Unterminated quoted string\n",
554             Gbl_CurrentLineNumber++);
555         State = DT_NORMAL_TEXT;
556         break;
557
558     default: /* Get next character */
559         break;
560     }
561     break;
562
563 case DT_ESCAPE_SEQUENCE:
564
565     /* Just copy the escaped character. TBD: sufficient for table compil
566
567     Gbl_CurrentLineBuffer[i] = (char) c;
568     i++;
569     State = DT_START_QUOTED_STRING;
570     break;
571
572 case DT_START_COMMENT:
573
574     /* Open comment if this character is an asterisk or slash */
575
576     switch (c)
577     {
578     case '*':
579         State = DT_SLASH_ASTERISK_COMMENT;
580         break;
581
582     case '/':

```

```

589     State = DT_SLASH_SLASH_COMMENT;
590     break;
591
592     default: /* Not a comment */
593
594         i++; /* Save the preceding slash */
595         if (i >= Gbl_LineBufferSize)
596         {
597             UtExpandLineBuffers ();
598         }
599
600         Gbl_CurrentLineBuffer[i] = (char) c;
601         i++;
602         State = DT_NORMAL_TEXT;
603         break;
604     }
605     break;
606
607 case DT_SLASH_ASTERISK_COMMENT:
608
609     /* Ignore chars until an asterisk-slash is found */
610
611     switch (c)
612     {
613     case '\n':
614         Gbl_NextLineOffset = (UINT32) ftell (Handle);
615         Gbl_CurrentLineNumber++;
616         break;
617
618     case '*':
619         State = DT_END_COMMENT;
620         break;
621
622     default:
623         break;
624     }
625     break;
626
627 case DT_SLASH_SLASH_COMMENT:
628
629     /* Ignore chars until end-of-line */
630
631     if (c == '\n')
632     {
633         /* We will exit via the NORMAL_TEXT path */
634
635         ungetc (c, Handle);
636         State = DT_NORMAL_TEXT;
637     }
638     break;
639
640 case DT_END_COMMENT:
641
642     /* End comment if this char is a slash */
643
644     switch (c)
645     {
646     case '/':
647         State = DT_NORMAL_TEXT;
648         break;

```

```

654     case '\n':
655         CurrentLineOffset = Gbl_NextLineOffset;
656         Gbl_NextLineOffset = (UINT32) ftell (Handle);
657         Gbl_CurrentLineNumber++;
658         break;
659
661     case '*':
662         /* Consume all adjacent asterisks */
663         break;
664
666     default:
667         State = DT_SLASH_ASTERISK_COMMENT;
668         break;
669     }
670     break;
671
673     case DT_MERGE_LINES:
674         if (c != '\n')
675         {
676             /*
677              * This is not a continuation backslash, it is a normal
678              * normal ASL backslash - for example: Scope(\_SB_)
679              */
680             i++; /* Keep the backslash that is already in the buffer */
681
682             ungetc (c, Handle);
683             State = DT_NORMAL_TEXT;
684         }
685         else
686         {
687             /*
688              * This is a continuation line -- a backslash followed
689              * immediately by a newline. Insert a space between the
690              * lines (overwrite the backslash)
691              */
692             Gbl_CurrentLineBuffer[i] = ' ';
693             i++;
694
695             /* Ignore newline, this will merge the lines */
696
697             CurrentLineOffset = Gbl_NextLineOffset;
698             Gbl_NextLineOffset = (UINT32) ftell (Handle);
699             Gbl_CurrentLineNumber++;
700             State = DT_NORMAL_TEXT;
701         }
702         break;
703
705     default:
706         DtFatal (ASL_MSG_COMPILER_INTERNAL, NULL, "Unknown input state");
707         return (ASL_EOF);
708     }
709 }
710
714 /*****
715 * FUNCTION:    DtScanFile
716 * PARAMETERS:  Handle          - Open file handle for the source file
717 */

```

```

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;
733
735     ACPI_FUNCTION_NAME (DtScanFile);
736
738     /* Get the file size */
739
740     Gbl_InputByteCount = DtGetFileSize (Handle);
741
742     Gbl_CurrentLineNumber = 0;
743     Gbl_CurrentLineOffset = 0;
744     Gbl_NextLineOffset = 0;
745
746     /* Scan line-by-line */
747
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));
752
753         Status = DtParseLine (Gbl_CurrentLineBuffer, Gbl_CurrentLineNumber, Offs
754             if (Status == AE_NOT_FOUND)
755             {
756                 break;
757             }
758         }
759
760         /* Dump the parse tree if debug enabled */
761
762         DtDumpFieldList (Gbl_FieldList);
763         return (Gbl_FieldList);
764     }
765
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,
787 void          *ReturnValue)
788 {
790     FlWriteFile (ASL_FILE_AML_OUTPUT, Subtable->Buffer, Subtable->Length);
791 }

794 /*****
795 *
796 * FUNCTION:      DtOutputBinary
797 *
798 * PARAMETERS:
799 *
800 * RETURN:        Status
801 *
802 * DESCRIPTION: Write entire binary ACPI table (result of compilation)
803 *
804 *****/

806 void
807 DtOutputBinary (
808     DT_SUBTABLE      *RootTable)
809 {
811     if (!RootTable)
812     {
813         return;
814     }

816     /* Walk the entire parse tree, emitting the binary data */

818     DtWalkTableTree (RootTable, DtWriteBinary, NULL, NULL);
819     Gbl_TableLength = DtGetFileSize (Gbl_Files[ASL_FILE_AML_OUTPUT].Handle);
820 }

823 /*
824 * Listing support
825 */

827 /*****
828 *
829 * FUNCTION:      DtDumpBuffer
830 *
831 * PARAMETERS:    FileID          - Where to write buffer data
832 *                Buffer           - Buffer to dump
833 *                Offset          - Offset in current table
834 *                Length          - Buffer Length
835 *
836 * RETURN:        None
837 *
838 * DESCRIPTION: Another copy of DumpBuffer routine (unfortunately).
839 *
840 * TBD: merge dump buffer routines
841 *
842 *****/

844 static void
845 DtDumpBuffer (
846     UINT32      FileId,
847     UINT8       *Buffer,
848     UINT32      Offset,
849     UINT32      Length)
850 {
851     UINT32      i;

```

```

852     UINT32      j;
853     UINT8       BufChar;

856     FlPrintFile (FileId, "Output: [%3.3Xh %4.4d %3d] ",
857         Offset, Offset, Length);

859     i = 0;
860     while (i < Length)
861     {
862         if (i >= 16)
863         {
864             FlPrintFile (FileId, "%24s", "");
865         }

867         /* Print 16 hex chars */

869         for (j = 0; j < 16; j++)
870         {
871             if (i + j >= Length)
872             {
873                 /* Dump fill spaces */

875                 FlPrintFile (FileId, "    ");
876                 j++;
877                 continue;
878             }

880             FlPrintFile (FileId, "%02X ", Buffer[i+j]);
881             j++;
882         }

884         FlPrintFile (FileId, " ");
885         for (j = 0; j < 16; j++)
886         {
887             if (i + j >= Length)
888             {
889                 FlPrintFile (FileId, "\n\n");
890                 return;
891             }

893             BufChar = Buffer[(ACPI_SIZE) i + j];
894             if (ACPI_IS_PRINT (BufChar))
895             {
896                 FlPrintFile (FileId, "%c", BufChar);
897             }
898             else
899             {
900                 FlPrintFile (FileId, ".");
901             }
902         }

904         /* Done with that line. */

906         FlPrintFile (FileId, "\n");
907         i += 16;
908     }

910     FlPrintFile (FileId, "\n\n");
911 }

914 /*****
915 *
916 * FUNCTION:      DtDumpFieldList
917 *

```

```

918 * PARAMETERS: Field          - Root field
919 *
920 * RETURN:      None
921 *
922 * DESCRIPTION: Dump the entire field list
923 *
924 *****/

926 void
927 DtDumpFieldList (
928     DT_FIELD          *Field)
929 {
930
931     if (!Gbl_DebugFlag || !Field)
932     {
933         return;
934     }
935
936     DbgPrint (ASL_DEBUG_OUTPUT, "\nField List:\n"
937         "LineNo  ByteOff  NameCol  Column  TableOff  "
938         "Flags    %32s : %s\n\n", "Name", "Value");
939     while (Field)
940     {
941         DbgPrint (ASL_DEBUG_OUTPUT,
942             "%.08X %.08X %.08X %.08X %.08X %.08X %32s : %s\n",
943             Field->Line, Field->ByteOffset, Field->NameColumn,
944             Field->Column, Field->TableOffset, Field->Flags,
945             Field->Name, Field->Value);
946
947         Field = Field->Next;
948     }
949
950     DbgPrint (ASL_DEBUG_OUTPUT, "\n\n");
951 }

954 /*****
955 * FUNCTION:      DtDumpSubtableInfo, DtDumpSubtableTree
956 *
957 * PARAMETERS:    DT_WALK_CALLBACK
958 *
959 * RETURN:        None
960 *
961 * DESCRIPTION:   Info - dump a subtable tree entry with extra information.
962 *                Tree - dump a subtable tree formatted by depth indentation.
963 *
964 *
965 *****/

967 static void
968 DtDumpSubtableInfo (
969     DT_SUBTABLE          *Subtable,
970     void                  *Context,
971     void                  *ReturnValue)
972 {
973
974     DbgPrint (ASL_DEBUG_OUTPUT,
975         "[%04X] %.08X %.08X %.08X %.08X %.08X %p %p %p\n",
976         Subtable->Depth, Subtable->Length, Subtable->TotalLength,
977         Subtable->SizeOfLengthField, Subtable->Flags, Subtable,
978         Subtable->Parent, Subtable->Child, Subtable->Peer);
979 }

981 static void
982 DtDumpSubtableTree (
983     DT_SUBTABLE          *Subtable,

```

```

984     void                  *Context,
985     void                  *ReturnValue)
986 {
987
988     DbgPrint (ASL_DEBUG_OUTPUT,
989         "[%04X] %s%08X (%.02X) - (%.02X)\n",
990         Subtable->Depth, (4 * Subtable->Depth), " ",
991         Subtable, Subtable->Length, Subtable->TotalLength);
992 }

995 /*****
996 *
997 * FUNCTION:      DtDumpSubtableList
998 *
999 * PARAMETERS:    None
1000 *
1001 * RETURN:        None
1002 *
1003 * DESCRIPTION:   Dump the raw list of subtables with information, and also
1004 *                dump the subtable list in formatted tree format. Assists with
1005 *                the development of new table code.
1006 *
1007 *****/

1009 void
1010 DtDumpSubtableList (
1011     void)
1012 {
1013
1014     if (!Gbl_DebugFlag || !Gbl_RootTable)
1015     {
1016         return;
1017     }
1018
1019     DbgPrint (ASL_DEBUG_OUTPUT,
1020         "Subtable Info:\n"
1021         "Depth  Length  TotalLen LenSize  Flags    "
1022         "This      Parent  Child    Peer\n\n");
1023     DtWalkTableTree (Gbl_RootTable, DtDumpSubtableInfo, NULL, NULL);
1024
1025     DbgPrint (ASL_DEBUG_OUTPUT,
1026         "\nSubtable Tree: (Depth, Subtable, Length, TotalLength)\n\n");
1027     DtWalkTableTree (Gbl_RootTable, DtDumpSubtableTree, NULL, NULL);
1028 }

1031 /*****
1032 *
1033 * FUNCTION:      DtWriteFieldToListing
1034 *
1035 * PARAMETERS:    Buffer          - Contains the compiled data
1036 *                Field          - Field node for the input line
1037 *                Length         - Length of the output data
1038 *
1039 * RETURN:        None
1040 *
1041 * DESCRIPTION:   Write one field to the listing file (if listing is enabled).
1042 *
1043 *****/

1045 void
1046 DtWriteFieldToListing (
1047     UINT8          *Buffer,
1048     DT_FIELD        *Field,
1049     UINT32          Length)

```

```

1050 {
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 }

```

new/usr/src/common/acpica/compiler/dtparser.1

1

```
*****
4461 Fri Jan 17 07:29:43 2014
new/usr/src/common/acpica/compiler/dtparser.1
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 %{
2 /*****
3 *
4 * Module Name: dtparser.1 - Flex input file for table compiler lexer
5 *
6 *****/
7
8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #include "aslcompiler.h"
46 #include "dtparser.y.h"
47
48 #define YY_NO_INPUT /* No file input, we use strings only */
49
50 #define _COMPONENT ACPI_COMPILER
51 #define ACPI_MODULE_NAME ("dtscanner")
52 %}
53
54 %option noyywrap
55 %option nounput
56
57 Number [0-9a-fA-F]+
58 HexNumber 0[xX][0-9a-fA-F]+
59 DecimalNumber 0[dD][0-9]+
60 LabelRef $[a-zA-Z][0-9a-zA-Z]*
```

new/usr/src/common/acpica/compiler/dtparser.1

2

```
61 WhiteSpace [ \t\v\r]+
62 NewLine [\n]
63
64 %%
65
66 \(\ return (EXPOP_PAREN_OPEN);
67 \) return (EXPOP_PAREN_CLOSE);
68 \~ return (EXPOP_ONES_COMPLIMENT);
69 \! return (EXPOP_LOGICAL_NOT);
70 \* return (EXPOP_MULTIPLY);
71 \/ return (EXPOP_DIVIDE);
72 \% return (EXPOP_MODULO);
73 \+ return (EXPOP_ADD);
74 \- return (EXPOP_SUBTRACT);
75 ">>" return (EXPOP_SHIFT_RIGHT);
76 "<<" return (EXPOP_SHIFT_LEFT);
77 < return (EXPOP_LESS);
78 > return (EXPOP_GREATER);
79 "<=" return (EXPOP_LESS_EQUAL);
80 ">=" return (EXPOP_GREATER_EQUAL);
81 "=" return (EXPOP_EQUAL);
82 "!=" return (EXPOP_NOT_EQUAL);
83 \& return (EXPOP_AND);
84 \^ return (EXPOP_XOR);
85 \| return (EXPOP_OR);
86 "&&" return (EXPOP_LOGICAL_AND);
87 "||" return (EXPOP_LOGICAL_OR);
88 <<EOF>> return (EXPOP_EOF); /* null end-of-string */
89
90 {LabelRef} return (EXPOP_LABEL);
91 {Number} return (EXPOP_NUMBER);
92 {HexNumber} return (EXPOP_HEX_NUMBER);
93 {NewLine} return (EXPOP_NEW_LINE);
94 {WhiteSpace} /* Ignore */
95
96 . return (EXPOP_EOF);
97
98 %%
99
100 /*
101 * Local support functions
102 */
103 YY_BUFFER_STATE LexBuffer;
104
105 /*****
106 *
107 * FUNCTION: DtInitLexer, DtTerminateLexer
108 *
109 * PARAMETERS: String - Input string to be parsed
110 *
111 * RETURN: None
112 *
113 * DESCRIPTION: Initialization and termination routines for lexer. Lexer needs
114 * a buffer to handle strings instead of a file.
115 *
116 *****/
117
118 int
119 DtInitLexer (
120 char *String)
121 {
122
123 LexBuffer = yy_scan_string (String);
124 return (LexBuffer == NULL);
125 }
```



new/usr/src/common/acpica/compiler/dtparser.1

3

```
127 void
128 DtTerminateLexer (
129     void)
130 {
131
132     yy_delete_buffer (LexBuffer);
133 }
```

new/usr/src/common/acpica/compiler/dtparser.y

1

```
*****
10045 Fri Jan 17 07:29:43 2014
new/usr/src/common/acpica/compiler/dtparser.y
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 %{
2 /*****
3 *
4 * Module Name: dtparser.y - Bison input file for table compiler parser
5 *
6 *****/
7
8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #include "aslcompiler.h"
46 #include "dtcompiler.h"
47
48 #define _COMPONENT DT_COMPILER
49 #define ACPI_MODULE_NAME ("dtparser")
50
51 int DtParserlex (void);
52 int DtParserparse (void);
53 void DtParsererror (char const *msg);
54 extern char *DtParserstext;
55 extern DT_FIELD *Gbl_CurrentField;
56
57 UINT64 DtParserResult; /* Expression return value */
58
59 /* Bison/yacc configuration */
```

new/usr/src/common/acpica/compiler/dtparser.y

2

```
61 #define yytname DtParsername
62 #define YYDEBUG 1 /* Enable debug output */
63 #define YYERROR_VERBOSE 1 /* Verbose error messages */
64 #define YYFLAG -32768
65
66 /* Define YYMALLOC/YYFREE to prevent redefinition errors */
67
68 #define YYMALLOC malloc
69 #define YYFREE free
70 %}
71
72 %union
73 {
74     UINT64 value;
75     UINT32 op;
76 }
77
78 /*! [Begin] no source code translation */
79
80 %type <value> Expression
81
82 %token <op> EXPOP_EOF
83 %token <op> EXPOP_NEW_LINE
84 %token <op> EXPOP_NUMBER
85 %token <op> EXPOP_HEX_NUMBER
86 %token <op> EXPOP_DECIMAL_NUMBER
87 %token <op> EXPOP_LABEL
88 %token <op> EXPOP_PAREN_OPEN
89 %token <op> EXPOP_PAREN_CLOSE
90
91 %left <op> EXPOP_LOGICAL_OR
92 %left <op> EXPOP_LOGICAL_AND
93 %left <op> EXPOP_OR
94 %left <op> EXPOP_XOR
95 %left <op> EXPOP_AND
96 %left <op> EXPOP_EQUAL EXPOP_NOT_EQUAL
97 %left <op> EXPOP_GREATER EXPOP_LESS EXPOP_GREATER_EQUAL EXPOP_LESS_EQUAL
98 %left <op> EXPOP_SHIFT_RIGHT EXPOP_SHIFT_LEFT
99 %left <op> EXPOP_ADD EXPOP_SUBTRACT
100 %left <op> EXPOP_MULTIPLY EXPOP_DIVIDE EXPOP_MODULO
101 %right <op> EXPOP_ONES_COMPLIMENT EXPOP_LOGICAL_NOT
102
103 %%
104
105 /*
106 * Operator precedence rules (from K&R)
107 *
108 * 1) ( )
109 * 2) ! ~ (unary operators that are supported here)
110 * 3) * / %
111 * 4) + -
112 * 5) >> <<
113 * 6) < > <= >=
114 * 7) == !=
115 * 8) &
116 * 9) ^
117 * 10) |
118 * 11) &&
119 * 12) ||
120 */
121 Value
122 : Expression EXPOP_NEW_LINE { DtParserResult=$1; return
123 | Expression EXPOP_EOF { DtParserResult=$1; return
124 ;
```

```

126 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 Fri Jan 17 07:29:44 2014
new/usr/src/common/acpica/compiler/dtsubtable.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dtsubtable.c - handling of subtables within ACPI tables
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __DTSUBTABLE_C__

46 #include "aslcompiler.h"
47 #include "dtcompiler.h"

49 #define _COMPONENT DT_COMPILER
50 ACPI_MODULE_NAME ("dtsubtable")

53 /*****
54 *
55 * FUNCTION: DtCreateSubtable
56 *
57 * PARAMETERS: Buffer - Input buffer
58 * Length - Buffer length
59 * RetSubtable - Returned newly created subtable

```

```

60 *
61 * RETURN: None
62 *
63 * DESCRIPTION: Create a subtable that is not listed with ACPI_DMTABLE_INFO
64 * For example, FACS has 24 bytes reserved at the end
65 * and it's not listed at AcpiDmTableInfoFacs
66 *
67 *****/

69 void
70 DtCreateSubtable (
71     UINT8          *Buffer,
72     UINT32         Length,
73     DT_SUBTABLE    **RetSubtable)
74 {
75     DT_SUBTABLE    *Subtable;

78     Subtable = UtLocalCalloc (sizeof (DT_SUBTABLE));

80     /* Create a new buffer for the subtable data */

82     Subtable->Buffer = UtLocalCalloc (Length);
83     ACPI_MEMCPY (Subtable->Buffer, Buffer, Length);

85     Subtable->Length = Length;
86     Subtable->TotalLength = Length;

88     *RetSubtable = Subtable;
89 }

92 /*****
93 *
94 * FUNCTION: DtInsertSubtable
95 *
96 * PARAMETERS: ParentTable - The Parent of the new subtable
97 * Subtable - The new subtable to insert
98 *
99 * RETURN: None
100 *
101 * DESCRIPTION: Insert the new subtable to the parent table
102 *
103 *****/

105 void
106 DtInsertSubtable (
107     DT_SUBTABLE    *ParentTable,
108     DT_SUBTABLE    *Subtable)
109 {
110     DT_SUBTABLE    *ChildTable;

113     Subtable->Peer = NULL;
114     Subtable->Parent = ParentTable;
115     Subtable->Depth = ParentTable->Depth + 1;

117     /* Link the new entry into the child list */

119     if (!ParentTable->Child)
120     {
121         ParentTable->Child = Subtable;
122     }
123     else
124     {
125         /* Walk to the end of the child list */

```

```

127     ChildTable = ParentTable->Child;
128     while (ChildTable->Peer)
129     {
130         ChildTable = ChildTable->Peer;
131     }

133     /* Add new subtable at the end of the child list */

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;

181     Subtable = Gbl_SubtableStack;

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 */

238         return (ParentTable->Child);
239     }

241     /* Otherwise just return the next peer (NULL if at end-of-list) */

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 {
263
264     if (!Subtable)
265     {
266         return (NULL);
267     }
268
269     return (Subtable->Parent);
270 }
271
272
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 *****/
285
286 UINT32
287 DtGetSubtableLength (
288     DT_FIELD          *Field,
289     ACPI_DMTABLE_INFO *Info)
290 {
291     UINT32      ByteLength = 0;
292     UINT8       Step;
293     UINT8       i;
294
295
296     /* Walk entire Info table; Null name terminates */
297
298     for (; Info->Name; Info++)
299     {
300         if (Info->Opcode == ACPI_DMT_EXTRA_TEXT)
301         {
302             continue;
303         }
304
305         if (!Field)
306         {
307             goto Error;
308         }
309
310         ByteLength += DtGetFieldLength (Field, Info);
311
312         switch (Info->Opcode)
313         {
314             case ACPI_DMT_GAS:
315
316                 Step = 5;
317                 break;
318
319             case ACPI_DMT_HESTNTFY:
320
321                 Step = 9;
322                 break;

```

```

324         default:
325
326             Step = 1;
327             break;
328         }
329
330         for (i = 0; i < Step; i++)
331         {
332             if (!Field)
333             {
334                 goto Error;
335             }
336
337             Field = Field->Next;
338         }
339     }
340
341     return (ByteLength);
342
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     }
350
351     return (ASL_EOF);
352 }
353
354
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 *****/
366
367 void
368 DtSetSubtableLength (
369     DT_SUBTABLE      *Subtable)
370 {
371
372     if (!Subtable->LengthField)
373     {
374         return;
375     }
376
377     ACPI_MEMCPY (Subtable->LengthField, &Subtable->TotalLength,
378                 Subtable->SizeOfLengthField);
379 }

```

new/usr/src/common/acpica/compiler/dtttable.c

1

```
*****
64596 Fri Jan 17 07:29:44 2014
new/usr/src/common/acpica/compiler/dtttable.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dtttable.c - handling for specific ACPI tables
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define __DTTABLE_C__
45
46 /* Compile all complex data tables */
47
48 #include "aslcompiler.h"
49 #include "dtcompiler.h"
50
51 #define _COMPONENT DT_COMPILER
52 ACPI_MODULE_NAME ("dtttable")
53
54
55 /* TBD: merge these into dmtbinfo.c? */
56
57 static ACPI_DMABLE_INFO TableInfoAsfAddress[] =
58 {
```

new/usr/src/common/acpica/compiler/dtttable.c

2

```
59 {ACPI_DMT_BUFFER, 0, "Addresses", 0},
60 {ACPI_DMT_EXIT, 0, NULL, 0}
61 };
62
63 static ACPI_DMABLE_INFO TableInfoDmarPciPath[] =
64 {
65 {ACPI_DMT_PCI_PATH, 0, "PCI Path", 0},
66 {ACPI_DMT_EXIT, 0, NULL, 0}
67 };
68
69 /* TBD: move to acmacros.h */
70
71 #define ACPI_SUB_PTR(t, a, b) \
72 ACPI_CAST_PTR (t, (ACPI_CAST_PTR (UINT8, (a)) - (ACPI_SIZE)(b)))
73
74 /* Local prototypes */
75
76 static ACPI_STATUS
77 DtCompileTwoSubtables (
78 void **List,
79 ACPI_DMABLE_INFO *TableInfo1,
80 ACPI_DMABLE_INFO *TableInfo2);
81
82
83
84
85 /*****
86 *
87 * FUNCTION: DtCompileTwoSubtables
88 *
89 * PARAMETERS: List - Current field list pointer
90 * TableInfo1 - Info table 1
91 * TableInfo2 - Info table 2
92 *
93 * RETURN: Status
94 *
95 * DESCRIPTION: Compile tables with a header and one or more same subtables.
96 * Include CPEP, EINJ, ERST, MCFG, MSCT, WDAT
97 *
98 *****/
99
100 static ACPI_STATUS
101 DtCompileTwoSubtables (
102 void **List,
103 ACPI_DMABLE_INFO *TableInfo1,
104 ACPI_DMABLE_INFO *TableInfo2)
105 {
106 ACPI_STATUS Status;
107 DT_SUBTABLE *Subtable;
108 DT_SUBTABLE *ParentTable;
109 DT_FIELD **PFieldList = (DT_FIELD **) List;
110
111 Status = DtCompileTable (PFieldList, TableInfo1, &Subtable, TRUE);
112 if (ACPI_FAILURE (Status))
113 {
114 return (Status);
115 }
116
117 ParentTable = DtPeekSubtable ();
118 DtInsertSubtable (ParentTable, Subtable);
119
120 while (*PFieldList)
121 {
122 Status = DtCompileTable (PFieldList, TableInfo2, &Subtable, FALSE);
123 if (ACPI_FAILURE (Status))
124
```



```

125     {
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);
390     if (ACPI_FAILURE (Status))
391     {
392         return (Status);
393     }
394
395     DtInsertSubtable (ParentTable, Subtable);
396 }
397 break;
398 }
399 }
400
401 DtPopSubtable ();
402 }
403
404 return (AE_OK);
405 }
406
407 /*****
408 * FUNCTION:      DtCompileCpep
409 *
410 * PARAMETERS:    List          - Current field list pointer
411 *
412 * RETURN:        Status
413 *
414 * DESCRIPTION:    Compile CPEP.
415 *
416 *****/
417
418 ACPI_STATUS
419 DtCompileCpep (
420     void                **List)
421 {
422     ACPI_STATUS          Status;
423
424     Status = DtCompileTwoSubtables (List,
425                                     AcpiDmTableInfoCpep, AcpiDmTableInfoCpep0);
426     return (Status);
427 }
428
429 /*****
430 * FUNCTION:      DtCompileCsrt
431 *
432 * PARAMETERS:    List          - Current field list pointer
433 *
434 * RETURN:        Status
435 *
436 * DESCRIPTION:    Compile CSRT.
437 *
438 *****/
439
440 ACPI_STATUS
441 DtCompileCsrt (
442     void                **List)
443 {
444     ACPI_STATUS          Status = AE_OK;
445     DT_SUBTABLE          *Subtable;
446     DT_SUBTABLE          *ParentTable;
447     **PFieldList = (DT_FIELD **) List;
448     UINT32               DescriptorCount;
449     UINT32               GroupLength;

```

```

457     /* Sub-tables (Resource Groups) */
458
459     while (*PFieldList)
460     {
461         /* Resource group subtable */
462
463         Status = DtCompileTable (PFieldList, AcpiDmTableInfoCsrt0,
464                                 &Subtable, TRUE);
465         if (ACPI_FAILURE (Status))
466         {
467             return (Status);
468         }
469
470         /* Compute the number of resource descriptors */
471
472         GroupLength =
473             (ACPI_CAST_PTR (ACPI_CSRT_GROUP,
474                             Subtable->Buffer))->Length -
475             (ACPI_CAST_PTR (ACPI_CSRT_GROUP,
476                             Subtable->Buffer))->SharedInfoLength -
477             sizeof (ACPI_CSRT_GROUP);
478
479         DescriptorCount = (GroupLength /
480                           sizeof (ACPI_CSRT_DESCRIPTOR));
481
482         ParentTable = DtPeekSubtable ();
483         DtInsertSubtable (ParentTable, Subtable);
484         DtPushSubtable (Subtable);
485
486         /* Shared info subtable (One per resource group) */
487
488         Status = DtCompileTable (PFieldList, AcpiDmTableInfoCsrt1,
489                                 &Subtable, TRUE);
490         if (ACPI_FAILURE (Status))
491         {
492             return (Status);
493         }
494
495         ParentTable = DtPeekSubtable ();
496         DtInsertSubtable (ParentTable, Subtable);
497
498         /* Sub-Subtables (Resource Descriptors) */
499
500         while (*PFieldList && DescriptorCount)
501         {
502             Status = DtCompileTable (PFieldList, AcpiDmTableInfoCsrt2,
503                                     &Subtable, TRUE);
504             if (ACPI_FAILURE (Status))
505             {
506                 return (Status);
507             }
508
509             ParentTable = DtPeekSubtable ();
510             DtInsertSubtable (ParentTable, Subtable);
511             DescriptorCount--;
512         }
513
514         DtPopSubtable ();
515     }
516
517     return (Status);
518 }

```

```

521 /*****
522 *
523 * FUNCTION:      DtCompileDbg2
524 *
525 * PARAMETERS:    List          - Current field list pointer
526 *
527 * RETURN:        Status
528 *
529 * DESCRIPTION:    Compile DBG2.
530 *
531 *****/

533 ACPI_STATUS
534 DtCompileDbg2 (
535     void                **List)
536 {
537     ACPI_STATUS          Status;
538     DT_SUBTABLE          *Subtable;
539     DT_SUBTABLE          *ParentTable;
540     DT_FIELD             **PFieldList = (DT_FIELD **) List;
541     UINT32               SubtableCount;
542     ACPI_DBG2_HEADER     *Dbg2Header;
543     ACPI_DBG2_DEVICE     *DeviceInfo;
544     UINT16               CurrentOffset;
545     UINT32               i;

548     /* Main table */

550     Status = DtCompileTable (PFieldList, AcpiDmTableInfoDbg2, &Subtable, TRUE);
551     if (ACPI_FAILURE (Status))
552     {
553         return (Status);
554     }

556     ParentTable = DtPeekSubtable ();
557     DtInsertSubtable (ParentTable, Subtable);

559     /* Main table fields */

561     Dbg2Header = ACPI_CAST_PTR (ACPI_DBG2_HEADER, Subtable->Buffer);
562     Dbg2Header->InfoOffset = sizeof (ACPI_TABLE_HEADER) + ACPI_PTR_DIFF (
563         ACPI_ADD_PTR (UINT8, Dbg2Header, sizeof (ACPI_DBG2_HEADER)), Dbg2Header)

565     SubtableCount = Dbg2Header->InfoCount;
566     DtPushSubtable (Subtable);

568     /* Process all Device Information subtables (Count = InfoCount) */

570     while (*PFieldList && SubtableCount)
571     {
572         /* Subtable: Debug Device Information */

574         Status = DtCompileTable (PFieldList, AcpiDmTableInfoDbg2Device,
575             &Subtable, TRUE);
576         if (ACPI_FAILURE (Status))
577         {
578             return (Status);
579         }

581         DeviceInfo = ACPI_CAST_PTR (ACPI_DBG2_DEVICE, Subtable->Buffer);
582         CurrentOffset = (UINT16) sizeof (ACPI_DBG2_DEVICE);

584         ParentTable = DtPeekSubtable ();
585         DtInsertSubtable (ParentTable, Subtable);
586         DtPushSubtable (Subtable);

```

```

588     ParentTable = DtPeekSubtable ();

590     /* BaseAddressRegister GAS array (Required, size is RegisterCount) */

592     DeviceInfo->BaseAddressOffset = CurrentOffset;
593     for (i = 0; *PFieldList && (i < DeviceInfo->RegisterCount); i++)
594     {
595         Status = DtCompileTable (PFieldList, AcpiDmTableInfoDbg2Addr,
596             &Subtable, TRUE);
597         if (ACPI_FAILURE (Status))
598         {
599             return (Status);
600         }

602         CurrentOffset += (UINT16) sizeof (ACPI_GENERIC_ADDRESS);
603         DtInsertSubtable (ParentTable, Subtable);
604     }

606     /* AddressSize array (Required, size = RegisterCount) */

608     DeviceInfo->AddressSizeOffset = CurrentOffset;
609     for (i = 0; *PFieldList && (i < DeviceInfo->RegisterCount); i++)
610     {
611         Status = DtCompileTable (PFieldList, AcpiDmTableInfoDbg2Size,
612             &Subtable, TRUE);
613         if (ACPI_FAILURE (Status))
614         {
615             return (Status);
616         }

618         CurrentOffset += (UINT16) sizeof (UINT32);
619         DtInsertSubtable (ParentTable, Subtable);
620     }

622     /* NamespaceString device identifier (Required, size = NamePathLength) */

624     DeviceInfo->NamepathOffset = CurrentOffset;
625     Status = DtCompileTable (PFieldList, AcpiDmTableInfoDbg2Name,
626         &Subtable, TRUE);
627     if (ACPI_FAILURE (Status))
628     {
629         return (Status);
630     }

632     /* Update the device info header */

634     DeviceInfo->NamepathLength = (UINT16) Subtable->Length;
635     CurrentOffset += (UINT16) DeviceInfo->NamepathLength;
636     DtInsertSubtable (ParentTable, Subtable);

638     /* OemData - Variable-length data (Optional, size = OemDataLength) */

640     Status = DtCompileTable (PFieldList, AcpiDmTableInfoDbg2OemData,
641         &Subtable, TRUE);
642     if (ACPI_FAILURE (Status))
643     {
644         return (Status);
645     }

647     /* Update the device info header (zeros if no OEM data present) */

649     DeviceInfo->OemDataOffset = 0;
650     DeviceInfo->OemDataLength = 0;

652     /* 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_DM_TABLE_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);
918     if (ACPI_FAILURE (Status))
919     {
920         return (Status);
921     }
922
923     DtInsertSubtable (ParentTable, Subtable);
924
925     if (Revision >= 5)
926     {
927         Status = DtCompileTable (PFieldList, AcpiDmTableInfoFadt5,
928                                 &Subtable, TRUE);
929         if (ACPI_FAILURE (Status))
930         {
931             return (Status);
932         }
933
934         DtInsertSubtable (ParentTable, Subtable);
935     }
936 }
937
938 return (AE_OK);
939 }

```

```

942 /*****
943 *
944 * FUNCTION:      DtCompileFpdt
945 *
946 * PARAMETERS:    List                - Current field list pointer
947 *
948 * RETURN:        Status
949 *
950 * DESCRIPTION:    Compile FPDT.
951 *
952 *****/

```

```

954 ACPI_STATUS
955 DtCompileFpdt (
956     void                **List)
957 {
958     ACPI_STATUS          Status;
959     ACPI_FPDT_HEADER     *FpdtHeader;
960     DT_SUBTABLE          *Subtable;
961     DT_SUBTABLE          *ParentTable;
962     ACPI_DMTABLE_INFO    *InfoTable;
963     DT_FIELD             **PFieldList = (DT_FIELD **) List;
964     DT_FIELD             *SubtableStart;
965
966     while (*PFieldList)
967     {
968         SubtableStart = *PFieldList;
969         Status = DtCompileTable (PFieldList, AcpiDmTableInfoFpdtHdr,
970                                 &Subtable, TRUE);
971         if (ACPI_FAILURE (Status))
972         {
973             return (Status);
974         }
975
976         ParentTable = DtPeekSubtable ();
977         DtInsertSubtable (ParentTable, Subtable);
978         DtPushSubtable (Subtable);
979
980         FpdtHeader = ACPI_CAST_PTR (ACPI_FPDT_HEADER, Subtable->Buffer);

```

```

983     switch (FpdtHeader->Type)
984     {
985     case ACPI_FPDT_TYPE_BOOT:
986
987         InfoTable = AcpiDmTableInfoFpdt0;
988         break;
989
990     case ACPI_FPDT_TYPE_S3PERF:
991
992         InfoTable = AcpiDmTableInfoFpdt1;
993         break;
994
995     default:
996
997         DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "FPDT");
998         return (AE_ERROR);
999         break;
1000     }
1001
1002     Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
1003     if (ACPI_FAILURE (Status))
1004     {
1005         return (Status);
1006     }
1007
1008     ParentTable = DtPeekSubtable ();
1009     DtInsertSubtable (ParentTable, Subtable);
1010     DtPopSubtable ();
1011 }
1012
1013 return (AE_OK);
1014 }

```

```

1017 /*****
1018 *
1019 * FUNCTION:      DtCompileHest
1020 *
1021 * PARAMETERS:    List                - Current field list pointer
1022 *
1023 * RETURN:        Status
1024 *
1025 * DESCRIPTION:    Compile HEST.
1026 *
1027 *****/

```

```

1029 ACPI_STATUS
1030 DtCompileHest (
1031     void                **List)
1032 {
1033     ACPI_STATUS          Status;
1034     DT_SUBTABLE          *Subtable;
1035     DT_SUBTABLE          *ParentTable;
1036     DT_FIELD             **PFieldList = (DT_FIELD **) List;
1037     DT_FIELD             *SubtableStart;
1038     ACPI_DMTABLE_INFO    *InfoTable;
1039     UINT16               Type;
1040     UINT32               BankCount;
1041
1042     Status = DtCompileTable (PFieldList, AcpiDmTableInfoHest,
1043                             &Subtable, TRUE);
1044     if (ACPI_FAILURE (Status))
1045     {
1046         return (Status);
1047     }
1048 }

```

```

1050 ParentTable = DtPeekSubtable ();
1051 DtInsertSubtable (ParentTable, Subtable);

1053 while (*PFieldList)
1054 {
1055     /* Get subtable type */

1057     SubtableStart = *PFieldList;
1058     DtCompileInteger ((UINT8 *) &Type, *PFieldList, 2, 0);

1060     switch (Type)
1061     {
1062     case ACPI_HEST_TYPE_IA32_CHECK:

1064         InfoTable = AcpiDmTableInfoHest0;
1065         break;

1067     case ACPI_HEST_TYPE_IA32_CORRECTED_CHECK:

1069         InfoTable = AcpiDmTableInfoHest1;
1070         break;

1072     case ACPI_HEST_TYPE_IA32_NMI:

1074         InfoTable = AcpiDmTableInfoHest2;
1075         break;

1077     case ACPI_HEST_TYPE_AER_ROOT_PORT:

1079         InfoTable = AcpiDmTableInfoHest6;
1080         break;

1082     case ACPI_HEST_TYPE_AER_ENDPOINT:

1084         InfoTable = AcpiDmTableInfoHest7;
1085         break;

1087     case ACPI_HEST_TYPE_AER_BRIDGE:

1089         InfoTable = AcpiDmTableInfoHest8;
1090         break;

1092     case ACPI_HEST_TYPE_GENERIC_ERROR:

1094         InfoTable = AcpiDmTableInfoHest9;
1095         break;

1097     default:

1099         /* Cannot continue on unknown type */

1101         DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "HEST");
1102         return (AE_ERROR);
1103     }

1105     Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
1106     if (ACPI_FAILURE (Status))
1107     {
1108         return (Status);
1109     }

1111     DtInsertSubtable (ParentTable, Subtable);

1113     /*
1114     * Additional subtable data - IA32 Error Bank(s)

```

```

1115     */
1116     BankCount = 0;
1117     switch (Type)
1118     {
1119     case ACPI_HEST_TYPE_IA32_CHECK:

1121         BankCount = (ACPI_CAST_PTR (ACPI_HEST_IA_MACHINE_CHECK,
1122                                     Subtable->Buffer))->NumHardwareBanks;
1123         break;

1125     case ACPI_HEST_TYPE_IA32_CORRECTED_CHECK:

1127         BankCount = (ACPI_CAST_PTR (ACPI_HEST_IA_CORRECTED,
1128                                     Subtable->Buffer))->NumHardwareBanks;
1129         break;

1131     default:

1133         break;
1134     }

1136     while (BankCount)
1137     {
1138         Status = DtCompileTable (PFieldList, AcpiDmTableInfoHestBank,
1139                                 &Subtable, TRUE);
1140         if (ACPI_FAILURE (Status))
1141         {
1142             return (Status);
1143         }

1145         DtInsertSubtable (ParentTable, Subtable);
1146         BankCount--;
1147     }
1148 }

1150 return (AE_OK);
1151 }

1154 /*****
1155  *
1156  * FUNCTION:      DtCompileIvrs
1157  *
1158  * PARAMETERS:    List                - Current field list pointer
1159  *
1160  * RETURN:        Status
1161  *
1162  * DESCRIPTION:    Compile IVRS.
1163  *
1164  *****/

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;

1180     Status = DtCompileTable (PFieldList, AcpiDmTableInfoIvrs,

```



```

1181         &Subtable, TRUE);
1182     if (ACPI_FAILURE (Status))
1183     {
1184         return (Status);
1185     }

1187     ParentTable = DtPeekSubtable ();
1188     DtInsertSubtable (ParentTable, Subtable);

1190     while (*PFieldList)
1191     {
1192         SubtableStart = *PFieldList;
1193         Status = DtCompileTable (PFieldList, AcpiDmTableInfoIvrsHdr,
1194                                 &Subtable, TRUE);
1195         if (ACPI_FAILURE (Status))
1196         {
1197             return (Status);
1198         }

1200         ParentTable = DtPeekSubtable ();
1201         DtInsertSubtable (ParentTable, Subtable);
1202         DtPushSubtable (Subtable);

1204         IvrsHeader = ACPI_CAST_PTR (ACPI_IVRS_HEADER, Subtable->Buffer);

1206         switch (IvrsHeader->Type)
1207         {
1208             case ACPI_IVRS_TYPE_HARDWARE:

1210                 InfoTable = AcpiDmTableInfoIvrs0;
1211                 break;

1213             case ACPI_IVRS_TYPE_MEMORY1:
1214             case ACPI_IVRS_TYPE_MEMORY2:
1215             case ACPI_IVRS_TYPE_MEMORY3:

1217                 InfoTable = AcpiDmTableInfoIvrs1;
1218                 break;

1220             default:

1222                 DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "IVRS");
1223                 return (AE_ERROR);
1224         }

1226         Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
1227         if (ACPI_FAILURE (Status))
1228         {
1229             return (Status);
1230         }

1232         ParentTable = DtPeekSubtable ();
1233         DtInsertSubtable (ParentTable, Subtable);

1235         if (IvrsHeader->Type == ACPI_IVRS_TYPE_HARDWARE)
1236         {
1237             while (*PFieldList &&
1238                 !ACPI_STRCMP ((*PFieldList)->Name, "Entry Type"))
1239             {
1240                 SubtableStart = *PFieldList;
1241                 DtCompileInteger (&EntryType, *PFieldList, 1, 0);

1243                 switch (EntryType)
1244                 {
1245                     /* 4-byte device entries */

```

```

1247         case ACPI_IVRS_TYPE_PAD4:
1248         case ACPI_IVRS_TYPE_ALL:
1249         case ACPI_IVRS_TYPE_SELECT:
1250         case ACPI_IVRS_TYPE_START:
1251         case ACPI_IVRS_TYPE_END:

1253             InfoTable = AcpiDmTableInfoIvrs4;
1254             break;

1256         /* 8-byte entries, type A */

1258         case ACPI_IVRS_TYPE_ALIAS_SELECT:
1259         case ACPI_IVRS_TYPE_ALIAS_START:

1261             InfoTable = AcpiDmTableInfoIvrs8a;
1262             break;

1264         /* 8-byte entries, type B */

1266         case ACPI_IVRS_TYPE_PAD8:
1267         case ACPI_IVRS_TYPE_EXT_SELECT:
1268         case ACPI_IVRS_TYPE_EXT_START:

1270             InfoTable = AcpiDmTableInfoIvrs8b;
1271             break;

1273         /* 8-byte entries, type C */

1275         case ACPI_IVRS_TYPE_SPECIAL:

1277             InfoTable = AcpiDmTableInfoIvrs8c;
1278             break;

1280         default:

1282             DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart,
1283                     "IVRS Device Entry");
1284             return (AE_ERROR);
1285         }

1287         Status = DtCompileTable (PFieldList, InfoTable,
1288                                 &Subtable, TRUE);
1289         if (ACPI_FAILURE (Status))
1290         {
1291             return (Status);
1292         }

1294         DtInsertSubtable (ParentTable, Subtable);
1295     }
1296 }

1298     DtPopSubtable ();
1299 }

1301     return (AE_OK);
1302 }

1305 /*****
1306  *
1307  * FUNCTION:      DtCompileMadt
1308  *
1309  * PARAMETERS:    List                - Current field list pointer
1310  *
1311  * RETURN:        Status
1312  */

```

```

1313 * DESCRIPTION: Compile MADT.
1314 *
1315 *****/

1317 ACPI_STATUS
1318 DtCompileMadt (
1319     void                **List)
1320 {
1321     ACPI_STATUS          Status;
1322     DT_SUBTABLE          *Subtable;
1323     DT_SUBTABLE          *ParentTable;
1324     DT_FIELD             **PFieldList = (DT_FIELD **) List;
1325     DT_FIELD             *SubtableStart;
1326     ACPI_SUBTABLE_HEADER *MadtHeader;
1327     ACPI_DM_TABLE_INFO   *InfoTable;

1330     Status = DtCompileTable (PFieldList, AcpiDmTableInfoMadt,
1331                             &Subtable, TRUE);
1332     if (ACPI_FAILURE (Status))
1333     {
1334         return (Status);
1335     }

1337     ParentTable = DtPeekSubtable ();
1338     DtInsertSubtable (ParentTable, Subtable);

1340     while (*PFieldList)
1341     {
1342         SubtableStart = *PFieldList;
1343         Status = DtCompileTable (PFieldList, AcpiDmTableInfoMadtHdr,
1344                                 &Subtable, TRUE);
1345         if (ACPI_FAILURE (Status))
1346         {
1347             return (Status);
1348         }

1350         ParentTable = DtPeekSubtable ();
1351         DtInsertSubtable (ParentTable, Subtable);
1352         DtPushSubtable (Subtable);

1354         MadtHeader = ACPI_CAST_PTR (ACPI_SUBTABLE_HEADER, Subtable->Buffer);

1356         switch (MadtHeader->Type)
1357         {
1358             case ACPI_MADT_TYPE_LOCAL_APIC:

1360                 InfoTable = AcpiDmTableInfoMadt0;
1361                 break;

1363             case ACPI_MADT_TYPE_IO_APIC:

1365                 InfoTable = AcpiDmTableInfoMadt1;
1366                 break;

1368             case ACPI_MADT_TYPE_INTERRUPT_OVERRIDE:

1370                 InfoTable = AcpiDmTableInfoMadt2;
1371                 break;

1373             case ACPI_MADT_TYPE_NMI_SOURCE:

1375                 InfoTable = AcpiDmTableInfoMadt3;
1376                 break;

1378             case ACPI_MADT_TYPE_LOCAL_APIC_NMI:

```

```

1380         InfoTable = AcpiDmTableInfoMadt4;
1381         break;

1383     case ACPI_MADT_TYPE_LOCAL_APIC_OVERRIDE:

1385         InfoTable = AcpiDmTableInfoMadt5;
1386         break;

1388     case ACPI_MADT_TYPE_IO_SAPIC:

1390         InfoTable = AcpiDmTableInfoMadt6;
1391         break;

1393     case ACPI_MADT_TYPE_LOCAL_SAPIC:

1395         InfoTable = AcpiDmTableInfoMadt7;
1396         break;

1398     case ACPI_MADT_TYPE_INTERRUPT_SOURCE:

1400         InfoTable = AcpiDmTableInfoMadt8;
1401         break;

1403     case ACPI_MADT_TYPE_LOCAL_X2APIC:

1405         InfoTable = AcpiDmTableInfoMadt9;
1406         break;

1408     case ACPI_MADT_TYPE_LOCAL_X2APIC_NMI:

1410         InfoTable = AcpiDmTableInfoMadt10;
1411         break;

1413     case ACPI_MADT_TYPE_GENERIC_INTERRUPT:

1415         InfoTable = AcpiDmTableInfoMadt11;
1416         break;

1418     case ACPI_MADT_TYPE_GENERIC_DISTRIBUTOR:

1420         InfoTable = AcpiDmTableInfoMadt12;
1421         break;

1423     default:

1425         DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "MADT");
1426         return (AE_ERROR);
1427     }

1429     Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
1430     if (ACPI_FAILURE (Status))
1431     {
1432         return (Status);
1433     }

1435     ParentTable = DtPeekSubtable ();
1436     DtInsertSubtable (ParentTable, Subtable);
1437     DtPopSubtable ();
1438 }

1440 return (AE_OK);
1441 }

1444 /*****

```

```

1445 *
1446 * FUNCTION:      DtCompileMcfg
1447 *
1448 * PARAMETERS:    List          - Current field list pointer
1449 *
1450 * RETURN:        Status
1451 *
1452 * DESCRIPTION:    Compile MCFG.
1453 *
1454 *****/
1456 ACPI_STATUS
1457 DtCompileMcfg (
1458     void                **List)
1459 {
1460     ACPI_STATUS          Status;

1463     Status = DtCompileTwoSubtables (List,
1464                                     AcpiDmTableInfoMcfg, AcpiDmTableInfoMcfg0);
1465     return (Status);
1466 }

1469 /*****
1470 *
1471 * FUNCTION:      DtCompileMpst
1472 *
1473 * PARAMETERS:    List          - Current field list pointer
1474 *
1475 * RETURN:        Status
1476 *
1477 * DESCRIPTION:    Compile MPST.
1478 *
1479 *****/
1481 ACPI_STATUS
1482 DtCompileMpst (
1483     void                **List)
1484 {
1485     ACPI_STATUS          Status;
1486     DT_SUBTABLE          *Subtable;
1487     DT_SUBTABLE          *ParentTable;
1488     DT_FIELD              **PFieldList = (DT_FIELD **) List;
1489     ACPI_MPST_CHANNEL     *MpstChannelInfo;
1490     ACPI_MPST_POWER_NODE *MpstPowerNode;
1491     ACPI_MPST_DATA_HDR    *MpstDataHeader;
1492     UINT16                SubtableCount;
1493     UINT32                PowerStateCount;
1494     UINT32                ComponentCount;

1497     /* Main table */

1499     Status = DtCompileTable (PFieldList, AcpiDmTableInfoMpst, &Subtable, TRUE);
1500     if (ACPI_FAILURE (Status))
1501     {
1502         return (Status);
1503     }

1505     ParentTable = DtPeekSubtable ();
1506     DtInsertSubtable (ParentTable, Subtable);
1507     DtPushSubtable (Subtable);

1509     MpstChannelInfo = ACPI_CAST_PTR (ACPI_MPST_CHANNEL, Subtable->Buffer);
1510     SubtableCount = MpstChannelInfo->PowerNodeCount;

```

```

1512     while (*PFieldList && SubtableCount)
1513     {
1514         /* Subtable: Memory Power Node(s) */

1516         Status = DtCompileTable (PFieldList, AcpiDmTableInfoMpst0,
1517                                 &Subtable, TRUE);
1518         if (ACPI_FAILURE (Status))
1519         {
1520             return (Status);
1521         }

1523         ParentTable = DtPeekSubtable ();
1524         DtInsertSubtable (ParentTable, Subtable);
1525         DtPushSubtable (Subtable);

1527         MpstPowerNode = ACPI_CAST_PTR (ACPI_MPST_POWER_NODE, Subtable->Buffer);
1528         PowerStateCount = MpstPowerNode->NumPowerStates;
1529         ComponentCount = MpstPowerNode->NumPhysicalComponents;

1531         ParentTable = DtPeekSubtable ();

1533         /* Sub-subtables - Memory Power State Structure(s) */

1535         while (*PFieldList && PowerStateCount)
1536         {
1537             Status = DtCompileTable (PFieldList, AcpiDmTableInfoMpst0A,
1538                                     &Subtable, TRUE);
1539             if (ACPI_FAILURE (Status))
1540             {
1541                 return (Status);
1542             }

1544             DtInsertSubtable (ParentTable, Subtable);
1545             PowerStateCount--;
1546         }

1548         /* Sub-subtables - Physical Component ID Structure(s) */

1550         while (*PFieldList && ComponentCount)
1551         {
1552             Status = DtCompileTable (PFieldList, AcpiDmTableInfoMpst0B,
1553                                     &Subtable, TRUE);
1554             if (ACPI_FAILURE (Status))
1555             {
1556                 return (Status);
1557             }

1559             DtInsertSubtable (ParentTable, Subtable);
1560             ComponentCount--;
1561         }

1563         SubtableCount--;
1564         DtPopSubtable ();
1565     }

1567     /* Subtable: Count of Memory Power State Characteristic structures */

1569     DtPopSubtable ();

1571     Status = DtCompileTable (PFieldList, AcpiDmTableInfoMpst1, &Subtable, TRUE);
1572     if (ACPI_FAILURE (Status))
1573     {
1574         return (Status);
1575     }

```

```

1577 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:
1711         InfoTable = AcpiDmTableInfoPcct0;
1712         break;
1714     default:
1716         DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "PCCT");
1717         return (AE_ERROR);
1718     }
1720     Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
1721     if (ACPI_FAILURE (Status))
1722     {
1723         return (Status);
1724     }
1726     ParentTable = DtPeekSubtable ();
1727     DtInsertSubtable (ParentTable, Subtable);
1728     DtPopSubtable ();
1729 }
1731 return (AE_OK);
1732 }

1735 /*****
1736  *
1737  * FUNCTION:      DtCompilePmtt
1738  *
1739  * PARAMETERS:    List                - Current field list pointer
1740  *
1741  * RETURN:        Status
1742  *
1743  * DESCRIPTION:   Compile PMTT.
1744  *
1745  *****/

1747 ACPI_STATUS
1748 DtCompilePmtt (
1749     void                **List)
1750 {
1751     ACPI_STATUS          Status;
1752     DT_SUBTABLE          *Subtable;
1753     DT_SUBTABLE          *ParentTable;
1754     DT_FIELD             **PFieldList = (DT_FIELD **) List;
1755     DT_FIELD             *SubtableStart;
1756     ACPI_PMTT_HEADER     *PmttHeader;
1757     ACPI_PMTT_CONTROLLER *PmttController;
1758     UINT16               DomainCount;
1759     UINT8                PrevType = ACPI_PMTT_TYPE_SOCKET;

1762     /* Main table */

1764     Status = DtCompileTable (PFieldList, AcpiDmTableInfoPmtt, &Subtable, TRUE);
1765     if (ACPI_FAILURE (Status))
1766     {
1767         return (Status);
1768     }

1770     ParentTable = DtPeekSubtable ();
1771     DtInsertSubtable (ParentTable, Subtable);
1772     DtPushSubtable (Subtable);

1774     while (*PFieldList)

```

```

1775     {
1776         SubtableStart = *PFieldList;
1777         Status = DtCompileTable (PFieldList, AcpiDmTableInfoPmttHdr,
1778                                 &Subtable, TRUE);
1779         if (ACPI_FAILURE (Status))
1780         {
1781             return (Status);
1782         }

1784         PmttHeader = ACPI_CAST_PTR (ACPI_PMTT_HEADER, Subtable->Buffer);
1785         while (PrevType >= PmttHeader->Type)
1786         {
1787             DtPopSubtable ();

1789             if (PrevType == ACPI_PMTT_TYPE_SOCKET)
1790             {
1791                 break;
1792             }
1793             PrevType--;
1794         }
1795         PrevType = PmttHeader->Type;

1797         ParentTable = DtPeekSubtable ();
1798         DtInsertSubtable (ParentTable, Subtable);
1799         DtPushSubtable (Subtable);

1801         switch (PmttHeader->Type)
1802         {
1803             case ACPI_PMTT_TYPE_SOCKET:

1805                 /* Subtable: Socket Structure */

1807                 Status = DtCompileTable (PFieldList, AcpiDmTableInfoPmtt0,
1808                                         &Subtable, TRUE);
1809                 if (ACPI_FAILURE (Status))
1810                 {
1811                     return (Status);
1812                 }

1814                 ParentTable = DtPeekSubtable ();
1815                 DtInsertSubtable (ParentTable, Subtable);
1816                 break;

1818             case ACPI_PMTT_TYPE_CONTROLLER:

1820                 /* Subtable: Memory Controller Structure */

1822                 Status = DtCompileTable (PFieldList, AcpiDmTableInfoPmtt1,
1823                                         &Subtable, TRUE);
1824                 if (ACPI_FAILURE (Status))
1825                 {
1826                     return (Status);
1827                 }

1829                 ParentTable = DtPeekSubtable ();
1830                 DtInsertSubtable (ParentTable, Subtable);

1832                 PmttController = ACPI_CAST_PTR (ACPI_PMTT_CONTROLLER,
1833                                                  (Subtable->Buffer - sizeof (ACPI_PMTT_HEADER)));
1834                 DomainCount = PmttController->DomainCount;

1836                 while (DomainCount)
1837                 {
1838                     Status = DtCompileTable (PFieldList, AcpiDmTableInfoPmtt1a,
1839                                             &Subtable, TRUE);
1840                     if (ACPI_FAILURE (Status))

```

```

1841     {
1842         return (Status);
1843     }

1845     DtInsertSubtable (ParentTable, Subtable);
1846     DomainCount--;
1847 }
1848 break;

1850 case ACPI_PMTT_TYPE_DIMM:

1852     /* Subtable: Physical Component Structure */

1854     Status = DtCompileTable (PFieldList, AcpiDmTableInfoPmtt2,
1855                             &Subtable, TRUE);
1856     if (ACPI_FAILURE (Status))
1857     {
1858         return (Status);
1859     }

1861     ParentTable = DtPeekSubtable ();
1862     DtInsertSubtable (ParentTable, Subtable);
1863     break;

1865 default:

1867     DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "PMTT");
1868     return (AE_ERROR);
1869 }
1870 }

1872 return (Status);
1873 }

1876 /*****
1877 *
1878 * FUNCTION:      DtCompileRsdt
1879 *
1880 * PARAMETERS:    List                - Current field list pointer
1881 *
1882 * RETURN:        Status
1883 *
1884 * DESCRIPTION:    Compile RSDT.
1885 *
1886 *****/

1888 ACPI_STATUS
1889 DtCompileRsdt (
1890     void                **List)
1891 {
1892     DT_SUBTABLE          *Subtable;
1893     DT_SUBTABLE          *ParentTable;
1894     DT_FIELD             *FieldList = *(DT_FIELD **) List;
1895     UINT32               Address;

1898     ParentTable = DtPeekSubtable ();

1900     while (FieldList)
1901     {
1902         DtCompileInteger ((UINT8 *) &Address, FieldList, 4, DT_NON_ZERO);

1904         DtCreateSubtable ((UINT8 *) &Address, 4, &Subtable);
1905         DtInsertSubtable (ParentTable, Subtable);
1906         FieldList = FieldList->Next;

```

```

1907     }

1909     return (AE_OK);
1910 }

1913 /*****
1914 *
1915 * FUNCTION:      DtCompileS3pt
1916 *
1917 * PARAMETERS:    PFieldList          - Current field list pointer
1918 *
1919 * RETURN:        Status
1920 *
1921 * DESCRIPTION:    Compile S3PT (Pointed to by FPDT)
1922 *
1923 *****/

1925 ACPI_STATUS
1926 DtCompileS3pt (
1927     DT_FIELD             **PFieldList)
1928 {
1929     ACPI_STATUS          Status;
1930     ACPI_S3PT_HEADER     *S3ptHeader;
1931     DT_SUBTABLE          *Subtable;
1932     DT_SUBTABLE          *ParentTable;
1933     ACPI_DMTABLE_INFO    *InfoTable;
1934     DT_FIELD             *SubtableStart;

1937     Status = DtCompileTable (PFieldList, AcpiDmTableInfoS3pt,
1938                             &Gbl_RootTable, TRUE);
1939     if (ACPI_FAILURE (Status))
1940     {
1941         return (Status);
1942     }

1944     DtPushSubtable (Gbl_RootTable);

1946     while (*PFieldList)
1947     {
1948         SubtableStart = *PFieldList;
1949         Status = DtCompileTable (PFieldList, AcpiDmTableInfoS3ptHdr,
1950                                 &Subtable, TRUE);
1951         if (ACPI_FAILURE (Status))
1952         {
1953             return (Status);
1954         }

1956         ParentTable = DtPeekSubtable ();
1957         DtInsertSubtable (ParentTable, Subtable);
1958         DtPushSubtable (Subtable);

1960         S3ptHeader = ACPI_CAST_PTR (ACPI_S3PT_HEADER, Subtable->Buffer);

1962         switch (S3ptHeader->Type)
1963         {
1964             case ACPI_S3PT_TYPE_RESUME:

1966                 InfoTable = AcpiDmTableInfoS3pt0;
1967                 break;

1969             case ACPI_S3PT_TYPE_SUSPEND:

1971                 InfoTable = AcpiDmTableInfoS3pt1;
1972                 break;

```

```

1974     default:
1975
1976         DtFatal (ASL_MSG_UNKNOWN_SUBTABLE, SubtableStart, "S3PT");
1977         return (AE_ERROR);
1978     }
1979
1980     Status = DtCompileTable (PFieldList, InfoTable, &Subtable, TRUE);
1981     if (ACPI_FAILURE (Status))
1982     {
1983         return (Status);
1984     }
1985
1986     ParentTable = DtPeekSubtable ();
1987     DtInsertSubtable (ParentTable, Subtable);
1988     DtPopSubtable ();
1989 }
1990
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, AcpiDmTableInfoSlic,
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     **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 *
2370 * DESCRIPTION: Compile XSDT.
2371 *
2372 *****/
2374 ACPI_STATUS
2375 DtCompileXsdt (
2376     void                **List)
2377 {
2378     DT_SUBTABLE          *Subtable;
2379     DT_SUBTABLE          *ParentTable;
2380     DT_FIELD             *FieldList = *(DT_FIELD **) List;
2381     UINT64               Address;
2383     ParentTable = DtPeekSubtable ();
2385     while (FieldList)
2386     {
2387         DtCompileInteger ((UINT8 *) &Address, FieldList, 8, DT_NON_ZERO);
2389         DtCreateSubtable ((UINT8 *) &Address, 8, &Subtable);
2390         DtInsertSubtable (ParentTable, Subtable);
2391         FieldList = FieldList->Next;
2392     }
2394     return (AE_OK);
2395 }
2398 /*****
2399 *
2400 * FUNCTION:    DtCompileGeneric
2401 *
2402 * PARAMETERS:  List                - Current field list pointer
2403 *
2404 * RETURN:      Status
2405 *
2406 * DESCRIPTION: Compile generic unknown table.
2407 *
2408 *****/
2410 ACPI_STATUS
2411 DtCompileGeneric (
2412     void                **List)
2413 {
2414     ACPI_STATUS          Status;
2415     DT_SUBTABLE          *Subtable;
2416     DT_SUBTABLE          *ParentTable;
2417     DT_FIELD             *PFieldList = (DT_FIELD **) List;
2418     ACPI_DMTABLE_INFO    *Info;
2421     ParentTable = DtPeekSubtable ();
2423     /*
2424     * Compile the "generic" portion of the table. This
2425     * part of the table is not predefined and any of the generic
2426     * operators may be used.
2427     */
2429     /* Find any and all labels in the entire generic portion */
2431     DtDetectAllLabels (*PFieldList);
2433     /* Now we can actually compile the parse tree */

```

```

2435     while (*PFieldList)
2436     {
2437         Info = DtGetGenericTableInfo ((*PFieldList)->Name);
2438         if (!Info)
2439         {
2440             sprintf (MsgBuffer, "Generic data type \"%s\" not found",
2441                     (*PFieldList)->Name);
2442             DtNameError (ASL_ERROR, ASL_MSG_INVALID_FIELD_NAME,
2443                         (*PFieldList), MsgBuffer);
2445             *PFieldList = (*PFieldList)->Next;
2446             continue;
2447         }
2449         Status = DtCompileTable (PFieldList, Info,
2450                                 &Subtable, TRUE);
2451         if (ACPI_SUCCESS (Status))
2452         {
2453             DtInsertSubtable (ParentTable, Subtable);
2454         }
2455         else
2456         {
2457             *PFieldList = (*PFieldList)->Next;
2459             if (Status == AE_NOT_FOUND)
2460             {
2461                 sprintf (MsgBuffer, "Generic data type \"%s\" not found",
2462                         (*PFieldList)->Name);
2463                 DtNameError (ASL_ERROR, ASL_MSG_INVALID_FIELD_NAME,
2464                             (*PFieldList), MsgBuffer);
2465             }
2466         }
2467     }
2469     return (AE_OK);
2470 }

```

new/usr/src/common/acpica/compiler/dttemplate.c

1

```
*****
11365 Fri Jan 17 07:29:44 2014
new/usr/src/common/acpica/compiler/dttemplate.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dttemplate - ACPI table template generation
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "dtcompiler.h"
47 #include "dttemplate.h" /* Contains the hex ACPI table templates */

49 #define _COMPONENT DT_COMPILER
50 ACPI_MODULE_NAME ("dttemplate")

53 /* Local prototypes */

55 static BOOLEAN
56 AcpiUtIsSpecialTable (
57     char *Signature);

59 static ACPI_STATUS
```

new/usr/src/common/acpica/compiler/dttemplate.c

2

```
60 DtCreateOneTemplate (
61     char *Signature,
62     ACPI_DMTABLE_DATA *TableData);

64 static ACPI_STATUS
65 DtCreateAllTemplates (
66     void);

69 /*****
70 *
71 * FUNCTION: AcpiUtIsSpecialTable
72 *
73 * PARAMETERS: Signature - ACPI table signature
74 *
75 * RETURN: TRUE if signature is a special ACPI table
76 *
77 * DESCRIPTION: Check for valid ACPI tables that are not in the main ACPI
78 * table data structure (AcpiDmTableData).
79 *
80 *****/

82 static BOOLEAN
83 AcpiUtIsSpecialTable (
84     char *Signature)
85 {
87     if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_DSDT) ||
88         ACPI_COMPARE_NAME (Signature, ACPI_SIG_SSDDT) ||
89         ACPI_COMPARE_NAME (Signature, ACPI_SIG_FACS) ||
90         ACPI_COMPARE_NAME (Signature, ACPI_SIG_RSDP_NAME))
91     {
92         return (TRUE);
93     }

95     return (FALSE);
96 }

99 /*****
100 *
101 * FUNCTION: DtCreateTemplates
102 *
103 * PARAMETERS: Signature - ACPI table signature
104 *
105 * RETURN: Status
106 *
107 * DESCRIPTION: Create one or more template files.
108 *
109 *****/

111 ACPI_STATUS
112 DtCreateTemplates (
113     char *Signature)
114 {
115     ACPI_DMTABLE_DATA *TableData;
116     ACPI_STATUS Status;

119     AslInitializeGlobals ();

121     /* Default (no signature) is DSDT */

123     if (!Signature)
124     {
125         Signature = "DSDT";
```

```

126     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_DM_TABLE_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_SSMT, 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 (TemplateDsdtd, 1, sizeof (TemplateDsdtd) - 1, File);
355             if (Actual != sizeof (TemplateDsdtd) - 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 (TemplateSsdtd, 1, sizeof (TemplateSsdtd) - 1, File);
366             if (Actual != sizeof (TemplateSsdtd) - 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_SIG_RSDP))
380         {
381             AcpiDmDumpDataTable (ACPI_CAST_PTR (ACPI_TABLE_HEADER,
382                 TemplateRsdpd));
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 }
392
393 fprintf (stderr,
394         "Created ACPI table template for [%4.4s], written to \"%s\\n\",
395         Signature, DisasmFilename);
396
397 Cleanup:
398     fclose (File);
399     AcpiOsRedirectOutput (stdout);
400     ACPI_FREE (DisasmFilename);
401     return (Status);
402 }
```

new/usr/src/common/acpica/compiler/dttemplate.h

1

```
*****
63848 Fri Jan 17 07:29:44 2014
new/usr/src/common/acpica/compiler/dttemplate.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dttemplate.h - ACPI table template definitions
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #ifndef __DTTEMPLATE_H
45 #define __DTTEMPLATE_H
46
47 /* Special templates for DSDT and SSDT (AML byte-code tables) */
48
49 const char TemplateDsdtt[] =
50 {
51     "DefinitionBlock (\"dsdt.aml\", \"DSDT\", 2, \"Intel\", \"Template\", 0x0000
52     \"\n\"
53     \"    Method (MAIN, 0, NotSerialized)\n\"
54     \"    {\n\"
55     \"        Return (Zero)\n\"
56     \"    }\n\"
57     \"}\n\n\";

```

new/usr/src/common/acpica/compiler/dttemplate.h

2

```
59 const char TemplateSsdt[] =
60 {
61     "DefinitionBlock (\"ssdt.aml\", \"SSDT\", 2, \"Intel\", \"Template\", 0x0000
62     \"\n\"
63     \"    Method (MAIN, 0, NotSerialized)\n\"
64     \"    {\n\"
65     \"        Return (Zero)\n\"
66     \"    }\n\"
67     \"}\n\n\";
68
69 /* Templates for ACPI data tables */
70
71 const unsigned char TemplateAsf[] =
72 {
73     0x41,0x53,0x46,0x21,0x72,0x00,0x00,0x00, /* 00000000 "ASF!r..." */
74     0x10,0x0B,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
75     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
76     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
77     0x28,0x05,0x10,0x20,0x00,0x00,0x10,0x00, /* 00000020 "(...)" */
78     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "..." */
79     0x00,0x00,0x00,0x00,0x01,0x00,0x14,0x00, /* 00000030 "..." */
80     0x00,0x00,0x01,0x0C,0x00,0x00,0x00,0x00, /* 00000038 "..." */
81     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000040 "..." */
82     0x02,0x00,0x0C,0x00,0x01,0x04,0x00,0x00, /* 00000048 "..." */
83     0x00,0x00,0x00,0x00,0x03,0x00,0x17,0x00, /* 00000050 "..." */
84     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000058 "..." */
85     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000060 "..." */
86     0x00,0x00,0x00,0x84,0x00,0x07,0x00,0x00, /* 00000068 "..." */
87     0x01,0x00 /* 00000070 "..." */
88 };
89
90 const unsigned char TemplateBgrt[] =
91 {
92     0x42,0x47,0x52,0x54,0x38,0x00,0x00,0x00, /* 00000000 "BGRt8..." */
93     0x01,0x0D,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
94     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
95     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
96     0x23,0x06,0x11,0x20,0x01,0x00,0x00,0x00, /* 00000020 "#..." */
97     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "..." */
98     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00 /* 00000030 "..." */
99 };
100
101 const unsigned char TemplateBert[] =
102 {
103     0x42,0x45,0x52,0x54,0x30,0x00,0x00,0x00, /* 00000000 "BERT0..." */
104     0x01,0x15,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
105     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
106     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
107     0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00, /* 00000020 "(...)" */
108     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00 /* 00000028 "..." */
109 };
110
111 const unsigned char TemplateBoot[] =
112 {
113     0x42,0x4F,0x4F,0x54,0x28,0x00,0x00,0x00, /* 00000000 "BOOT(...)" */
114     0x01,0x0D,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
115     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
116     0x00,0x00,0x04,0x06,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
117     0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00 /* 00000020 "(...)" */
118 };
119
120 const unsigned char TemplateCpep[] =
121 {
122     0x43,0x50,0x45,0x50,0x34,0x00,0x00,0x00, /* 00000000 "CPEP4..." */
123     0x01,0x0F,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
124     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */

```

```

125     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 TemplateCsr[] =
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     0x00,0x00,0x00,0xB3,0x00,0x00,0x00,0x00, /* 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     0x43,0x48,0x41,0x35,0x0A,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 "...MyDev" */
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 "..._SE..." */
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 "....." */
258 0x00,0x00,0x00,0x00,0x00,0x00,0x40,0x00,0x04, /* 00000030 ".....@.." */
259 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "....." */
260 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000040 "....." */
261 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000048 "....." */
262 0x01,0x00,0x00,0x00,0x00,0x40,0x00,0x04, /* 00000050 ".....@.." */
263 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000058 "....." */
264 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000060 "....." */
265 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000068 "....." */
266 0x02,0x02,0x01,0x00,0x00,0x40,0x00,0x04, /* 00000070 ".....@.." */
267 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000078 "....." */
268 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000080 "....." */
269 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000088 "....." */
270 0x03,0x00,0x00,0x00,0x00,0x00,0x40,0x00,0x04, /* 00000090 ".....@.." */
271 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000098 "....." */
272 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000A0 "....." */
273 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 000000A8 "....." */
274 0x04,0x03,0x01,0x00,0x00,0x40,0x00,0x04, /* 000000B0 ".....@.." */
275 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000B8 "....." */
276 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000C0 "....." */
277 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 000000C8 "....." */
278 0x05,0x03,0x01,0x00,0x01,0x10,0x00,0x02, /* 000000D0 "....." */
279 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000D8 "....." */
280 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000E0 "....." */
281 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 000000E8 "....." */
282 0x06,0x01,0x00,0x00,0x00,0x00,0x40,0x00,0x04, /* 000000F0 ".....@.." */
283 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000F8 "....." */
284 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000100 "....." */
285 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000108 "....." */
286 0x07,0x00,0x01,0x00,0x00,0x40,0x00,0x04, /* 00000110 ".....@.." */
287 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000118 "....." */
288 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000120 "....." */
289 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000128 "....." */
290 };

```

```

292 const unsigned char TemplateErst[] =
293 {
294 0x45,0x52,0x53,0x54,0x30,0x02,0x00,0x00, /* 00000000 "ERST0..." */
295 0x01,0xAB,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
296 0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* "TEMPLATE" */
297 0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* "...INTL" */
298 0x28,0x05,0x10,0x20,0x30,0x00,0x00,0x00, /* "{...0..." */
299 0x00,0x00,0x00,0x00,0x10,0x00,0x00,0x00, /* "....." */
300 0x00,0x03,0x00,0x00,0x00,0x40,0x00,0x04, /* ".....@.." */
301 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* "....." */
302 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000040 "....." */
303 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000048 "....." */
304 0x01,0x03,0x00,0x00,0x00,0x40,0x00,0x04, /* 00000050 ".....@.." */
305 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000058 "....." */
306 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000060 "....." */
307 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000068 "....." */
308 0x02,0x03,0x00,0x00,0x00,0x40,0x00,0x04, /* 00000070 ".....@.." */
309 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000078 "....." */
310 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000080 "....." */
311 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000088 "....." */
312 0x03,0x04,0x01,0x00,0x00,0x40,0x00,0x04, /* 00000090 ".....@.." */
313 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000098 "....." */
314 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000A0 "....." */
315 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 000000A8 "....." */
316 0x04,0x02,0x00,0x00,0x00,0x40,0x00,0x04, /* 000000B0 ".....@.." */
317 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000B8 "....." */
318 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000C0 "....." */
319 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 000000C8 "....." */
320 0x05,0x03,0x00,0x00,0x01,0x08,0x00,0x01, /* 000000D0 "....." */
321 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000D8 "....." */
322 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000E0 "....." */

```

```

323 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 000000E8 "....." */
324 0x06,0x01,0x00,0x00,0x00,0x40,0x00,0x04, /* 000000F0 ".....@.." */
325 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000F8 "....." */
326 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000100 "....." */
327 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000108 "....." */
328 0x07,0x00,0x00,0x00,0x00,0x40,0x00,0x04, /* 00000110 ".....@.." */
329 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000118 "....." */
330 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000120 "....." */
331 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000128 "....." */
332 0x08,0x00,0x00,0x00,0x00,0x40,0x00,0x04, /* 00000130 ".....@.." */
333 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000138 "....." */
334 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000140 "....." */
335 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000148 "....." */
336 0x09,0x02,0x00,0x00,0x00,0x40,0x00,0x04, /* 00000150 ".....@.." */
337 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000158 "....." */
338 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000160 "....." */
339 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000168 "....." */
340 0x0A,0x00,0x00,0x00,0x00,0x40,0x00,0x04, /* 00000170 ".....@.." */
341 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000178 "....." */
342 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000180 "....." */
343 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000188 "....." */
344 0x0B,0x03,0x00,0x00,0x00,0x40,0x00,0x04, /* 00000190 ".....@.." */
345 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000198 "....." */
346 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000001A0 "....." */
347 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 000001A8 "....." */
348 0x0C,0x00,0x00,0x00,0x00,0x40,0x00,0x04, /* 000001B0 ".....@.." */
349 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000001B8 "....." */
350 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000001C0 "....." */
351 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 000001C8 "....." */
352 0x0D,0x00,0x00,0x00,0x00,0x40,0x00,0x04, /* 000001D0 ".....@.." */
353 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000001D8 "....." */
354 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000001E0 "....." */
355 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 000001E8 "....." */
356 0x0E,0x00,0x00,0x00,0x00,0x40,0x00,0x04, /* 000001F0 ".....@.." */
357 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000001F8 "....." */
358 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000200 "....." */
359 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000208 "....." */
360 0x0F,0x00,0x00,0x00,0x00,0x40,0x00,0x04, /* 00000210 ".....@.." */
361 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000218 "....." */
362 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000220 "....." */
363 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF, /* 00000228 "....." */
364 };

```

```

366 const unsigned char TemplateFacs[] =
367 {
368 0x46,0x41,0x43,0x53,0x40,0x00,0x00,0x00, /* 00000000 "FACS@..." */
369 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000008 "....." */
370 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000010 "....." */
371 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000018 "....." */
372 0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000020 "....." */
373 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
374 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000030 "....." */
375 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "....." */
376 };

```

```
378 /* Version 5 FADT */
```

```

380 const unsigned char TemplateFadt[] =
381 {
382 0x46,0x41,0x43,0x50,0x0C,0x01,0x00,0x00, /* 00000000 "FACP..." */
383 0x05,0x64,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...dINTEL" */
384 0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
385 0x00,0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
386 0x15,0x11,0x13,0x20,0x01,0x00,0x00,0x00, /* 00000020 "....." */
387 0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
388 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000030 "....." */

```

```

389 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 0x00,0x00,0x00,0x00,0x08,0x00,0x00,0x00, /* 00000118 "....." */
487 0x00,0x00,0x00,0x01,0x01,0x00,0x00,0x00, /* 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,0x01,0x01,0x00,0x00,0x00,0x00, /* 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 };

523 const unsigned char TemplateIvrs[] =
524 {
525     0x49,0x56,0x52,0x53,0xBC,0x00,0x00,0x00, /* 00000000 "IVRS..." */
526     0x01,0x87,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
527     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
528     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
529     0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00, /* 00000020 "(...)" */
530     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
531     0x10,0x14,0x34,0x00,0x00,0x00,0x00,0x00, /* 00000030 "...4...." */
532     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "....." */
533     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000040 "....." */
534     0x00,0x00,0x00,0x00,0x40,0x00,0x00,0x00, /* 00000048 "....@" */
535     0x00,0x00,0x00,0x00,0x42,0x00,0x00,0x00, /* 00000050 "....B..." */
536     0x00,0x00,0x00,0x00,0x48,0x00,0x00,0x00, /* 00000058 "....H..." */
537     0x00,0x00,0x00,0x00,0x20,0x08,0x20,0x00, /* 00000060 "....." */
538     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000068 "....." */
539     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000070 "....." */
540     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000078 "....." */
541     0x00,0x00,0x00,0x00,0x21,0x04,0x20,0x00, /* 00000080 "....!..." */
542     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000088 "....." */
543     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000090 "....." */
544     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000098 "....." */
545     0x00,0x00,0x00,0x00,0x10,0x14,0x18,0x00, /* 000000A0 "....." */
546     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000A8 "....." */
547     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000B0 "....." */
548     0x00,0x00,0x00,0x00, /* 000000B8 "...." */
549 };

551 /* MADT with ACPI 5.0 subtables */

553 const unsigned char TemplateMadt[] =
554 {
555     0x41,0x50,0x49,0x43,0xF6,0x00,0x00,0x00, /* 00000000 "APIC..." */
556     0x01,0xB0,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
557     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
558     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
559     0x23,0x06,0x11,0x20,0x00,0x00,0x00,0x00, /* 00000020 "#....." */
560     0x01,0x00,0x00,0x00,0x00,0x08,0x00,0x00, /* 00000028 "....." */
561     0x01,0x00,0x00,0x00,0x01,0x0C,0x01,0x00, /* 00000030 "....." */
562     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "....." */
563     0x02,0x0A,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000040 "....." */
564     0x00,0x00,0x03,0x08,0x0D,0x00,0x01,0x00, /* 00000048 "....." */
565     0x00,0x00,0x04,0x06,0x00,0x05,0x00,0x01, /* 00000050 "....." */
566     0x05,0x0C,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000058 "....." */
567     0x00,0x00,0x00,0x00,0x06,0x10,0x00,0x00, /* 00000060 "....." */
568     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000068 "....." */
569     0x00,0x00,0x00,0x00,0x07,0x16,0x00,0x00, /* 00000070 "....." */
570     0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00, /* 00000078 "....." */
571     0x00,0x00,0x00,0x00,0x5C,0x43,0x50,0x55, /* 00000080 "....CPU" */
572     0x30,0x00,0x08,0x10,0x05,0x00,0x00,0x00, /* 00000088 "0....." */
573     0x00,0x00,0x01,0x00,0x00,0x00,0x01,0x00, /* 00000090 "....." */
574     0x00,0x00,0x09,0x10,0x00,0x00,0x00,0x00, /* 00000098 "....." */
575     0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00, /* 000000A0 "....." */
576     0x00,0x00,0x0A,0x0C,0x05,0x00,0x00,0x00, /* 000000A8 "....." */
577     0x00,0x00,0x00,0x00,0x00,0x00,0x0B,0x28, /* 000000B0 ".....(" */
578     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000B8 "....." */
579     0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00, /* 000000C0 "....." */
580     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000C8 "....." */
581     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000D0 "....." */
582     0x00,0x00,0x00,0x00,0x00,0x00,0x0C,0x18, /* 000000D8 "....." */
583     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000E0 "....." */
584     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000E8 "....." */
585     0x00,0x00,0x00,0x00,0x00,0x00, /* 000000F0 "....." */
586 };

```

```

588 const unsigned char TemplateMcfg[] =
589 {
590     0x4D,0x43,0x46,0x47,0x3C,0x00,0x00,0x00, /* 00000000 "MCFG<..." */
591     0x01,0x19,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
592     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
593     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
594     0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00, /* 00000020 "(...)" */
595     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
596     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000030 "....." */
597     0x00,0x00,0x00,0x00, /* 00000038 "...." */
598 };

600 const unsigned char TemplateMchi[] =
601 {
602     0x4D,0x43,0x48,0x49,0x45,0x00,0x00,0x00, /* 00000000 "MCHIE..." */
603     0x01,0xE4,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
604     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
605     0x15,0x07,0x00,0x02,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
606     0x28,0x05,0x10,0x20,0x01,0x00,0x00,0x00, /* 00000020 "(...)" */
607     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
608     0x00,0x00,0x00,0x00,0x02,0x08,0x00, /* 00000030 "....." */
609     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "....." */
610     0x00,0x00,0x00,0x00,0x00, /* 00000040 "....." */
611 };

613 const unsigned char TemplateMpst[] =
614 {
615     0x4D,0x50,0x53,0x54,0xB6,0x00,0x00,0x00, /* 00000000 "MPST..." */
616     0x01,0x77,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...wINTEL" */
617     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
618     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
619     0x13,0x09,0x12,0x20,0x00,0x00,0x00,0x00, /* 00000020 "....." */
620     0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
621     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000030 "....." */
622     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "....." */
623     0x00,0x00,0x00,0x00,0x02,0x00,0x00,0x00, /* 00000040 "....." */
624     0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000048 "....." */
625     0x00,0x00,0x01,0x00,0x02,0x00,0x00,0x00, /* 00000050 "....." */
626     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000058 "....." */
627     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000060 "....." */
628     0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x00, /* 00000068 "....." */
629     0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00, /* 00000070 "....." */
630     0x00,0x00,0x02,0x00,0x00,0x00,0x01,0x00, /* 00000078 "....." */
631     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000080 "....." */
632     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000088 "....." */
633     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000090 "....." */
634     0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00, /* 00000098 "....." */
635     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000A0 "....." */
636     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000A8 "....." */
637     0x00,0x00,0x00,0x00,0x00,0x00, /* 000000B0 "....." */
638 };

640 const unsigned char TemplateMsct[] =
641 {
642     0x4D,0x53,0x43,0x54,0x90,0x00,0x00,0x00, /* 00000000 "MSCT..." */
643     0x01,0xB7,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
644     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
645     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
646     0x28,0x05,0x10,0x20,0x38,0x00,0x00,0x00, /* 00000020 "(...8..." */
647     0x03,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
648     0xFF,0xFF,0xFF,0xFF,0xFF,0x00,0x00, /* 00000030 "....." */
649     0x01,0x16,0x00,0x00,0x00,0x00,0x03,0x00, /* 00000038 "....." */
650     0x00,0x00,0x10,0x00,0x00,0x00,0x00,0x00, /* 00000040 "....." */
651     0x00,0x00,0x40,0x00,0x00,0x00,0x01,0x16, /* 00000048 "...@" */
652     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000050 "....." */

```

```

653 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000058 "....." */
654 0x00,0x00,0x00,0x00,0x01,0x16,0x00,0x00, /* 00000060 "....." */
655 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000068 "....." */
656 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000070 "....." */
657 0x00,0x00,0x01,0x16,0x00,0x00,0x00,0x00, /* 00000078 "....." */
658 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000080 "....." */
659 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000088 "....." */
660 };

662 const unsigned char TemplateMtmr[] =
663 {
664 0x4D,0x54,0x4D,0x52,0x4C,0x00,0x00,0x00, /* 00000000 "MTMRL..." */
665 0x01,0xB0,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
666 0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
667 0x03,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
668 0x17,0x01,0x13,0x20,0x00,0x20,0x00,0x03, /* 00000020 "....." */
669 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
670 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000030 "....." */
671 0x00,0x20,0x00,0x03,0x00,0x00,0x00,0x00, /* 00000038 "....." */
672 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000040 "....." */
673 0x00,0x00,0x00,0x00, /* 00000048 "...." */
674 };

676 const unsigned char TemplatePcct[] =
677 {
678 0x50,0x43,0x43,0x54,0xAC,0x00,0x00,0x00, /* 00000000 "PCCT..." */
679 0x01,0x97,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
680 0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
681 0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
682 0x15,0x11,0x13,0x20,0x01,0x00,0x00,0x00, /* 00000020 "....." */
683 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
684 0x00,0x3E,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000030 ">....." */
685 0x11,0x11,0x11,0x11,0x11,0x11,0x11,0x11, /* 00000038 "UUUUffff" */
686 0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22, /* 00000040 "UUUUffff" */
687 0x01,0x32,0x00,0x03,0x33,0x33,0x33,0x33, /* 00000048 ".2..3333" */
688 0x33,0x33,0x33,0x33,0x44,0x44,0x44,0x44, /* 00000050 "3333DDDD" */
689 0x44,0x44,0x44,0x44,0x55,0x55,0x55,0x55, /* 00000058 "DDDDUUUU" */
690 0x55,0x55,0x55,0x55,0x66,0x66,0x66,0x66, /* 00000060 "UUUUffff" */
691 0x77,0x77,0x77,0x77,0x88,0x88,0x00,0x3E, /* 00000068 "www...>" */
692 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xFF, /* 00000070 "....." */
693 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xEE,0xEE, /* 00000078 ".....2" */
694 0xEE,0xEE,0xEE,0xEE,0xEE,0xEE,0x01,0x32, /* 00000080 "....." */
695 0x00,0x03,0xDD,0xDD,0xDD,0xDD,0xDD,0xDD, /* 00000088 "....." */
696 0xDD,0xDD,0xCC,0xCC,0xCC,0xCC,0xCC,0xCC, /* 00000090 "....." */
697 0xCC,0xCC,0xBB,0xBB,0xBB,0xBB,0xBB,0xBB, /* 00000098 "....." */
698 0xBB,0xBB,0xAA,0xAA,0xAA,0xAA,0x99,0x99, /* 000000A0 "....." */
699 0x99,0x99,0x88,0x88 /* 000000A8 "...." */
700 };

702 const unsigned char TemplatePmtt[] =
703 {
704 0x50,0x4D,0x54,0x54,0xB4,0x00,0x00,0x00, /* 00000000 "PMTT..." */
705 0x01,0x3A,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
706 0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
707 0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
708 0x26,0x08,0x11,0x20,0x00,0x00,0x00,0x00, /* 00000020 "&....." */
709 0x00,0x00,0x80,0x00,0x01,0x00,0x00,0x00, /* 00000028 ".....T" */
710 0x00,0x00,0x00,0x00,0x01,0x00,0x54,0x00, /* 00000030 "....." */
711 0x05,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "....." */
712 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000040 "....." */
713 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000048 "....." */
714 0x00,0x00,0x03,0x00,0x00,0x00,0x00,0x00, /* 00000050 "....." */
715 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000058 "....." */
716 0x02,0x00,0x14,0x00,0x02,0x00,0x00,0x00, /* 00000060 "....." */
717 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000068 "....." */
718 0x00,0x00,0x00,0x00,0x02,0x00,0x14,0x00, /* 00000070 "....." */

```

```

719 0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000078 "....." */
720 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000080 "....." */
721 0x01,0x00,0x20,0x00,0x01,0x00,0x00,0x00, /* 00000088 "....." */
722 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000090 "....." */
723 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000098 "....." */
724 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000A0 "....." */
725 0x00,0x00,0x0C,0x00,0x01,0x00,0x00,0x00, /* 000000A8 "....." */
726 0x00,0x00,0x00,0x00 /* 000000B0 "...." */
727 };

729 const unsigned char TemplateRsdP[] =
730 {
731 0x52,0x53,0x44,0x20,0x50,0x54,0x52,0x20, /* 00000000 "RSD PTR " */
732 0x43,0x49,0x4E,0x54,0x45,0x4C,0x20,0x02, /* 00000008 "CINTEL " */
733 0x00,0x00,0x00,0x00,0x24,0x00,0x00,0x00, /* 00000010 "....$..." */
734 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000018 "....." */
735 0xDC,0x00,0x00,0x00 /* 00000020 "...." */
736 };

738 const unsigned char TemplateRsdT[] =
739 {
740 0x52,0x53,0x44,0x54,0x44,0x00,0x00,0x00, /* 00000000 "RSDTD..." */
741 0x01,0xB1,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
742 0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
743 0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
744 0x28,0x05,0x10,0x20,0x10,0x00,0x00,0x00, /* 00000020 "(....." */
745 0x20,0x00,0x00,0x00,0x30,0x00,0x00,0x00, /* 00000028 "...0..." */
746 0x40,0x00,0x00,0x00,0x50,0x00,0x00,0x00, /* 00000030 "@...P..." */
747 0x60,0x00,0x00,0x00,0x70,0x00,0x00,0x00, /* 00000038 "'...p..." */
748 0x80,0x00,0x00,0x00 /* 00000040 "...." */
749 };

751 const unsigned char TemplateS3pt[] =
752 {
753 0x53,0x33,0x50,0x54,0x34,0x00,0x00,0x00, /* 00000000 "S3PT4..." */
754 0x00,0x00,0x18,0x01,0x00,0x00,0x00,0x00, /* 00000008 "....." */
755 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000010 "....." */
756 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000018 "....." */
757 0x01,0x00,0x14,0x01,0x00,0x00,0x00,0x00, /* 00000020 "....." */
758 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
759 0x00,0x00,0x00,0x00 /* 00000030 "...." */
760 };

762 const unsigned char TemplateSbst[] =
763 {
764 0x53,0x42,0x53,0x54,0x30,0x00,0x00,0x00, /* 00000000 "SBST0..." */
765 0x01,0x06,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
766 0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
767 0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
768 0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00, /* 00000020 "(....." */
769 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
770 };

772 const unsigned char TemplateSlic[] =
773 {
774 0x53,0x4C,0x49,0x43,0x76,0x01,0x00,0x00, /* 00000000 "SLICv..." */
775 0x01,0x07,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
776 0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
777 0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
778 0x11,0x02,0x11,0x20,0x00,0x00,0x00,0x00, /* 00000020 "....." */
779 0x9C,0x00,0x00,0x00,0x06,0x02,0x00,0x00, /* 00000028 "....." */
780 0x00,0x24,0x00,0x00,0x52,0x53,0x41,0x31, /* 00000030 "$..RSAI" */
781 0x00,0x04,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "....." */
782 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000040 "....." */
783 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000048 "....." */
784 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000050 "....." */

```

```

785 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 0x01,0x00,0x00,0x00,0x00,0xB6,0x00,0x00,0x00, /* 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 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 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,0x1F,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 "....&$#!" */
876 0x22,0x21,0x20,0x1E,0x1D,0x1C,0x1B, /* 00000198 "!!....." */
877 0x1A,0x19,0x18,0x17,0x16,0x15,0x0A,0x10, /* 000001A0 "....." */
878 0x27,0x26,0x25,0x24,0x23,0x22,0x21,0x20, /* 000001A8 " '&$#!..." */
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 "....." */
918 0x00,0x10,0x00,0x00,0x01,0x00,0x00,0x00, /* 00000030 "....." */
919 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "....." */
920 0x01,0x28,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000040 "....." */
921 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000048 "....." */
922 0x00,0xFC,0x09,0x00,0x00,0x00,0x00,0x00, /* 00000050 "....." */
923 0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00, /* 00000058 "....." */
924 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000060 "....." */
925 0x02,0x18,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000068 "....." */
926 0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00, /* 00000070 "....." */
927 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000078 "....." */
928 };

930 const unsigned char TemplateTcpc[] =
931 {
932     0x54,0x43,0x50,0x41,0x32,0x00,0x00,0x00, /* 00000000 "TCPA2..." */
933     0x01,0x67,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 ".gINTEL" */
934     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
935     0x80,0x31,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 ".1..INTL" */
936     0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00, /* 00000020 "(\....." */
937     0x00,0x00,0x01,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
938     0x00,0x00, /* 00000030 "..." */
939 };

941 const unsigned char TemplateTpm2[] =
942 {
943     0x54,0x50,0x4D,0x32,0x34,0x00,0x00,0x00, /* 00000000 "TPM24..." */
944     0x03,0x42,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 ".BINTEL" */
945     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
946     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
947     0x14,0x11,0x12,0x20,0x00,0x00,0x00,0x00, /* 00000020 "....." */
948     0x77,0x66,0x55,0x44,0x33,0x22,0x11,0x00, /* 00000028 "wFUD3..." */
949     0x01,0x00,0x00,0x00, /* 00000030 "...." */
950 };

952 const unsigned char TemplateUefi[] =
953 {
954     0x55,0x45,0x46,0x49,0x36,0x00,0x00,0x00, /* 00000000 "UEFI6..." */
955     0x01,0x9B,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 ".INTEL" */
956     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
957     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
958     0x28,0x05,0x10,0x20,0x00,0x01,0x02,0x03, /* 00000020 "(\....." */
959     0x04,0x05,0x06,0x07,0x08,0x09,0x0A,0x0B, /* 00000028 "....." */
960     0x0C,0x0D,0x0E,0x0F,0x00,0x00, /* 00000030 "....." */
961 };

963 const unsigned char TemplateVrtc[] =
964 {
965     0x56,0x52,0x54,0x43,0x44,0x00,0x00,0x00, /* 00000000 "VRTCD..." */
966     0x01,0xEF,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 ".INTEL" */
967     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
968     0x03,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
969     0x17,0x01,0x13,0x20,0x00,0x08,0x00,0x00, /* 00000020 "....." */
970     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
971     0x00,0x00,0x00,0x00,0x00,0x08,0x00,0x00, /* 00000030 "....." */
972     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "....." */
973     0x00,0x00,0x00,0x00, /* 00000040 "...." */
974 };

976 const unsigned char TemplateWaet[] =
977 {
978     0x57,0x41,0x45,0x54,0x28,0x00,0x00,0x00, /* 00000000 "WAET(...)" */
979     0x01,0x19,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 ".INTEL" */
980     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
981     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
982     0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00, /* 00000020 "(\....." */

```

```

983 };

985 const unsigned char TemplateWdat[] =
986 {
987     0x57,0x44,0x41,0x54,0x5C,0x00,0x00,0x00, /* 00000000 "WDAT\..." */
988     0x01,0xE3,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
989     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
990     0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
991     0x28,0x05,0x10,0x20,0x20,0x00,0x00,0x00, /* 00000020 "(\....." */
992     0xFF,0x00,0xFF,0xFF,0xFF,0x00,0x00,0x00, /* 00000028 "....." */
993     0x58,0x02,0x00,0x00,0xFF,0x03,0x00,0x00, /* 00000030 "X....." */
994     0x02,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "....." */
995     0x0E,0x00,0x00,0x00,0x01,0x02,0x00,0x00, /* 00000040 "....." */
996     0x01,0x10,0x00,0x02,0x60,0x04,0x00,0x00, /* 00000048 "....." */
997     0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00, /* 00000050 "....." */
998     0x01,0x00,0x00,0x00, /* 00000058 "...." */
999 };

1001 const unsigned char TemplateWddt[] =
1002 {
1003     0x57,0x44,0x44,0x54,0x40,0x00,0x00,0x00, /* 00000000 "WDDT@..." */
1004     0x01,0x00,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
1005     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
1006     0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
1007     0x28,0x05,0x10,0x20,0x00,0x00,0x00,0x00, /* 00000020 "(\....." */
1008     0x00,0x00,0x01,0xFF,0x00,0x00,0x00,0x00, /* 00000028 "....." */
1009     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000030 "....." */
1010     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "....." */
1011 };

1013 const unsigned char TemplateWdrt[] =
1014 {
1015     0x57,0x44,0x52,0x54,0x47,0x00,0x00,0x00, /* 00000000 "WDRTG..." */
1016     0x01,0xB0,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
1017     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
1018     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
1019     0x28,0x05,0x10,0x20,0x00,0x20,0x00,0x00, /* 00000020 "(\....." */
1020     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
1021     0x00,0x20,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000030 "....." */
1022     0x00,0x00,0x00,0x00,0xFF,0xFF,0xFF,0xFF, /* 00000038 "....." */
1023     0x00,0x00,0x00,0x00,0xFF,0xFF,0x00, /* 00000040 "....." */
1024 };

1026 const unsigned char TemplateXsdt[] =
1027 {
1028     0x58,0x53,0x44,0x54,0x64,0x00,0x00,0x00, /* 00000000 "XSDTD..." */
1029     0x01,0x8B,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
1030     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
1031     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
1032     0x28,0x05,0x10,0x20,0x10,0x00,0x00,0x00, /* 00000020 "(\....." */
1033     0x00,0x00,0x00,0x00,0x20,0x00,0x00,0x00, /* 00000028 "....." */
1034     0x00,0x00,0x00,0x00,0x30,0x00,0x00,0x00, /* 00000030 "....0..." */
1035     0x00,0x00,0x00,0x00,0x40,0x00,0x00,0x00, /* 00000038 "....@..." */
1036     0x00,0x00,0x00,0x00,0x50,0x00,0x00,0x00, /* 00000040 "....P..." */
1037     0x00,0x00,0x00,0x00,0x60,0x00,0x00,0x00, /* 00000048 "....." */
1038     0x00,0x00,0x00,0x00,0x70,0x00,0x00,0x00, /* 00000050 "....p..." */
1039     0x00,0x00,0x00,0x00,0x80,0x00,0x00,0x00, /* 00000058 "....." */
1040     0x00,0x00,0x00,0x00, /* 00000060 "...." */
1041 };

1043 #endif

```

new/usr/src/common/acpica/compiler/dtutils.c

1

```
*****
21549 Fri Jan 17 07:29:45 2014
new/usr/src/common/acpica/compiler/dtutils.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dtutils.c - Utility routines for the data table compiler
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define __DTUTILS_C__
45
46 #include "aslcompiler.h"
47 #include "dtcompiler.h"
48 #include "actables.h"
49
50 #define _COMPONENT DT_COMPILER
51 #define ACPI_MODULE_NAME ("dtutils")
52
53 /* Local prototypes */
54
55 static void
56 DtSum (
57     DT_SUBTABLE      *Subtable,
58     void              *Context,
```

new/usr/src/common/acpica/compiler/dtutils.c

2

```
59     void              *ReturnValue);
60
61 /*****
62 *
63 * FUNCTION:      DtError
64 *
65 * PARAMETERS:    Level              - Seriousness (Warning/error, etc.)
66 *                MessageId          - Index into global message buffer
67 *                Op                  - Parse node where error happened
68 *                ExtraMessage        - additional error message
69 *
70 * RETURN:        None
71 *
72 * DESCRIPTION:   Common error interface for data table compiler
73 *
74 *****/
75
76 void
77 DtError (
78     UINT8          Level,
79     UINT8          MessageId,
80     DT_FIELD       *FieldObject,
81     char           *ExtraMessage)
82 {
83     /* Check if user wants to ignore this exception */
84
85     if (AslIsExceptionDisabled (Level, MessageId))
86     {
87         return;
88     }
89
90     if (FieldObject)
91     {
92         AslCommonError (Level, MessageId,
93             FieldObject->Line,
94             FieldObject->Line,
95             FieldObject->ByteOffset,
96             FieldObject->Column,
97             Gbl_Files[ASL_FILE_INPUT].Filename, ExtraMessage);
98     }
99     else
100     {
101         AslCommonError (Level, MessageId, 0,
102             0, 0, 0, 0, ExtraMessage);
103     }
104 }
105
106 /*****
107 *
108 * FUNCTION:      DtNameError
109 *
110 * PARAMETERS:    Level              - Seriousness (Warning/error, etc.)
111 *                MessageId          - Index into global message buffer
112 *                Op                  - Parse node where error happened
113 *                ExtraMessage        - additional error message
114 *
115 * RETURN:        None
116 *
117 * DESCRIPTION:   Error interface for named objects
118 *
119 *****/
120
121 void
```

```

125 DtNameError (
126     UINT8          Level,
127     UINT8          MessageId,
128     DT_FIELD       *FieldObject,
129     char           *ExtraMessage)
130 {
131
132     switch (Level)
133     {
134     case ASL_WARNING2:
135     case ASL_WARNING3:
136
137         if (Gbl_WarningLevel < Level)
138         {
139             return;
140         }
141         break;
142
143     default:
144
145         break;
146     }
147
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 }
163
164
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  *****/
177
178 void
179 DtFatal (
180     UINT8          MessageId,
181     DT_FIELD       *FieldObject,
182     char           *ExtraMessage)
183 {
184
185     DtError (ASL_ERROR, MessageId, FieldObject, ExtraMessage);
186
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 #ifndef __OBSOLETE
194     CmCleanupAndExit ();
195     exit (1);
196 #endif
197 }
198
199
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  *****/
216
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;
226
227
228     /* Skip over any white space in the buffer */
229
230     while ((*ThisChar == ' ') || (*ThisChar == '\t'))
231     {
232         ThisChar++;
233     }
234
235     /* Skip leading zeros */
236
237     while ((*ThisChar) == '0')
238     {
239         ThisChar++;
240     }
241
242     /* Convert character-by-character */
243
244     while (*ThisChar)
245     {
246         if (ACPI_IS_DIGIT (*ThisChar))
247         {
248             /* Convert ASCII 0-9 to Decimal value */
249
250             ThisDigit = ((UINT8) *ThisChar) - '0';
251
252             else /* Letter */
253             {
254                 ThisDigit = (UINT32) ACPI_TOUPPER (*ThisChar);
255                 if (!ACPI_IS_XDIGIT ((char) ThisDigit))
256                     break;

```



```

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             Buffer++;
475         }
476     }
477     return (++ByteLength);
478 }

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     }
786     else
787     {
788         ChildTable = ParentTable;

790         if (ChildTable == Gbl_RootTable)
791         {
792             break;
793         }

795         ParentTable = DtGetParentSubtable (ParentTable);

797         ParentTable->TotalLength += ChildTable->TotalLength;
798         if (ParentTable->LengthField)
799         {
800             DtSetSubtableLength (ParentTable);
801         }
802     }
803 }
804 }

807 /*****
808 *
809 * FUNCTION:      DtWalkTableTree
810 *
811 * PARAMETERS:   StartTable      - Subtable in the tree where walking begins
812 *               UserFunction    - Called during the walk
813 *               Context         - Passed to user function
814 *               ReturnValue     - The return value of UserFunction
815 *
816 * RETURN:       None
817 *
818 * DESCRIPTION:  Performs a depth-first walk of the subtable tree
819 *
820 *****/

822 void
823 DtWalkTableTree (
824     DT_SUBTABLE      *StartTable,
825     DT_WALK_CALLBACK UserFunction,
826     void             *Context,
827     void             *ReturnValue)
828 {
829     DT_SUBTABLE      *ParentTable;
830     DT_SUBTABLE      *ChildTable;

833     ParentTable = StartTable;
834     ChildTable = NULL;

836     if (!ParentTable)
837     {
838         return;
839     }

841     UserFunction (ParentTable, Context, ReturnValue);

843     while (1)
844     {
845         ChildTable = DtGetNextSubtable (ParentTable, ChildTable);
846         if (ChildTable)
847         {
848             UserFunction (ChildTable, Context, ReturnValue);

850             if (ChildTable->Child)

```

```

851     {
852         ParentTable = ChildTable;
853         ChildTable = NULL;
854     }
855     }
856     else
857     {
858         ChildTable = ParentTable;
859         if (ChildTable == Gbl_RootTable)
860         {
861             break;
862         }

864         ParentTable = DtGetParentSubtable (ParentTable);

866         if (ChildTable->Peer == StartTable)
867         {
868             break;
869         }
870     }
871 }
872 }

875 /*****
876 *
877 * FUNCTION:      DtFreeFieldList
878 *
879 * PARAMETERS:   None
880 *
881 * RETURN:       None
882 *
883 * DESCRIPTION:  Free the field list
884 *
885 *****/

887 void
888 DtFreeFieldList (
889     void)
890 {
891     DT_FIELD      *Field = Gbl_FieldList;
892     DT_FIELD      *NextField;

895     /* Walk and free entire field list */

897     while (Field)
898     {
899         NextField = Field->Next; /* Save link */

901         if (!(Field->Flags & DT_FIELD_NOT_ALLOCATED))
902         {
903             ACPI_FREE (Field->Name);
904             ACPI_FREE (Field->Value);
905         }

907         ACPI_FREE (Field);
908         Field = NextField;
909     }
910 }

```

new/usr/src/common/acpica/compiler/new\_table.txt

1

\*\*\*\*\*

3581 Fri Jan 17 07:29:45 2014  
new/usr/src/common/acpica/compiler/new\_table.txt  
acpica-unix2-20130823  
\*\*\*\*\*

```
1 How to add a new ACPI table to ACPICA and the iASL compiler.
2 -----

4 There are four main tasks that are needed to provide support for a
5 new ACPI table:
6 1) Create a full definition of the table and any subtables
7   in the ACPICA headers.
8 2) Add disassembler support for the new table
9 3) Add iASL table compiler support for the new table
10 4) Create a default template for the new table for iASL -T
11   option.

13 Notes for each of these tasks provided below.

16 1) Header Support
17 -----

19 New tables should be added to the appropriate header:
20 actbl2.h: Used for new tables that are not defined in the ACPI spec.
21 actbl3.h: Used for new tables that are defined in the ACPI spec.

23 Use ACPI_TABLE_HEADER for the common ACPI table header.
24 Subtables should be defined separately from the main table.
25 Don't add placeholder fields for subtables and other multiple data items.
26 (Don't use xxxxx[1] for a field that can have multiple items.)
27 The disassembler and data table compiler depends on this.
28 For tables not defined in the ACPI spec, add a comment to indicate where
29 the table came from.
30 Use other table definitions for additional guidance.

33 2) iASL Disassembler Support
34 -----

36 Add definition of the table (and subtables) in common/dmtdbinfo.c
37 Add table access macro(s) of the form ACPI_xxxx_OFFSET
38 Add ACPI_DMT_TERMINATOR at the end of every table/subtable definition

40 Add externals for the table/subtable definitions in acdisasm.h
41 Add an entry for the new table in the AcpiDmTableData in common/dmtdbtable.c

43 If there are no subtables, add the AcpiDmTableInfoXXXX name to the
44 AcpiDmTableData and it will automatically be disassembled.

46 If there are subtables, a dump routine must be written:
47 Add an AcpiDmDumpXXXX function to dmtdbdump.c -- note, code for another
48 similar table can often be ported for the new table.
49 Add an external for this function to acdisasm.h
50 Add this function to the AcpiDmTableData entry for the new ACPI table

52 Debug/Test: Either find an existing example of the new ACPI table, or
53 create one using the "generic ACPI table support" included in the
54 iASL data table compiler. Use the -G option to force a
55 generic compile. It is often best to create the table from scratch,
56 since this clearly exposes the dependencies (lengths, offsets, etc.)
57 that the Table Compiler support will need to generate.

60 3) iASL Table Compiler Support
61 -----
```

new/usr/src/common/acpica/compiler/new\_table.txt

2

```
63 Simple tables do not require a compile routine. The definition of the
64 table in common/dmtdbinfo.c (created in step 2 above) will suffice.

66 Complex tables with subtables will require a compile routine with a name
67 of the form DtCompileXXXX.
68 Add a DtCompileXXXX function to the dtdtable.c module.
69 Add an external for this function in dtcompiler.h
70 Add this function to the AcpiDmTableData entry for the new ACPI table
71 in common/dmtdbtable.c

74 4) Template Support (-T iASL option)
75 -----

77 Create an example of the new ACPI table. This example should create
78 multiple subtables (if supported), and multiple instances of any
79 variable length data.

81 Compile the example file with the -sc option. This will create a C
82 array that contains the table contents.

84 Add this array to the dtdtemplate.h file. Name the array TemplateXXXX.
85 Add this array name to the AcpiDmTableData entry for the new ACPI table

87 Debug/Test: Create the template file. Compile the file. Disassemble the file.
88 Compile the disassembly file.
```

new/usr/src/common/acpica/compiler/preprocess.h

1

```
*****
7820 Fri Jan 17 07:29:45 2014
new/usr/src/common/acpica/compiler/preprocess.h
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: preprocess.h - header for iASL Preprocessor
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __PREPROCESS_H__

46 #ifndef _PREPROCESS
47 #define _PREPROCESS

49 #undef PR_EXTERN

51 #ifdef _DECLARE_PR_GLOBALS
52 #define PR_EXTERN
53 #define PR_INIT_GLOBAL(a,b)      (a)=(b)
54 #else
55 #define PR_EXTERN      extern
56 #define PR_INIT_GLOBAL(a,b)      (a)
57 #endif

60 /*
```

new/usr/src/common/acpica/compiler/preprocess.h

2

```
61 * Configuration
62 */
63 #define PR_MAX_MACRO_ARGS      32      /* Max number of macro args */
64 #define PR_MAX_ARG_INSTANCES  24      /* Max instances of any one arg
65 #define PR_LINES_PER_BLOCK    4096    /* Max input source lines per bl

68 /*
69 * Local defines and macros
70 */
71 #define PR_TOKEN_SEPARATORS    " , ( ) { } \t \n"
72 #define PR_MACRO_SEPARATORS    " , ( ) { } ~ ! * / % + - < > = & ^ | \" \t \n"
73 #define PR_MACRO_ARGUMENTS     " , \t \n"
74 #define PR_EXPR_SEPARATORS     " , ( ) { } ~ ! * / % + - < > = & ^ | \" \t \n"

76 #define PR_PREFIX_ID           "Pr(%.4u) - "      /* Used for debug outp

78 #define THIS_TOKEN_OFFSET(t)  ((t-Gbl_MainTokenBuffer) + 1)

81 /*
82 * Preprocessor structures
83 */
84 typedef struct pr_macro_arg
85 {
86     char                *Name;
87     UINT32              Offset[PR_MAX_ARG_INSTANCES];
88     UINT16              UseCount;
89 } PR_MACRO_ARG;

92 typedef struct pr_define_info
93 {
94     struct pr_define_info *Previous;
95     struct pr_define_info *Next;
96     char                  *Identifier;
97     char                  *Replacement;
98     char                  *Body;      /* Macro body */
99     PR_MACRO_ARG          *Args;      /* Macro arg list */
100     UINT16               ArgCount;    /* Macro arg count */
101     BOOLEAN               Persist;    /* Keep for entire compiler run

103 } PR_DEFINE_INFO;

105 typedef struct pr_directive_info
106 {
107     char                  *Name;      /* Directive name */
108     UINT8                 ArgCount;   /* Required # of args */

110 } PR_DIRECTIVE_INFO;

112 typedef struct pr_operator_info
113 {
114     char                  *Op;

116 } PR_OPERATOR_INFO;

118 typedef struct pr_file_node
119 {
120     struct pr_file_node  *Next;
121     FILE                 *File;
122     char                  *Filename;
123     UINT32               CurrentLineNumber;

125 } PR_FILE_NODE;
```

```

127 #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 - Preprocessor 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
```

new/usr/src/common/acpica/compiler/prexpress.c

1

```
*****
9317 Fri Jan 17 07:29:45 2014
new/usr/src/common/acpica/compiler/prexpress.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: prexpress - Preprocessor #if expression support
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "dtcompiler.h"

48 #define _COMPONENT          ASL_PREPROCESSOR
49         ACPI_MODULE_NAME    ("prexpress")

51 /* Local prototypes */

53 static char *
54 PrExpandMacros (
55     char                *Line);

58 #ifndef _UNDER_DEVELOPMENT
59 /*****
60 *
```

new/usr/src/common/acpica/compiler/prexpress.c

2

```
61 * FUNCTION:      PrUnTokenize
62 *
63 * PARAMETERS:    Buffer          - Token Buffer
64 *                Next           - "Next" buffer from GetNextToken
65 *
66 * RETURN:        None
67 *
68 * DESCRIPTION:    Un-tokenized the current token buffer. The implementation is
69 *                 to simply set the null inserted by GetNextToken to a blank.
70 *                 If Next is NULL, there were no tokens found in the Buffer,
71 *                 so there is nothing to do.
72 *
73 *****/

75 static void
76 PrUnTokenize (
77     char                *Buffer,
78     char                *Next)
79 {
80     UINT32              Length = strlen (Buffer);

83     if (!Next)
84     {
85         return;
86     }
87     if (Buffer[Length] != '\n')
88     {
89         Buffer[strlen(Buffer)] = ' ';
90     }
91 }
92 #endif

95 /*****
96 *
97 * FUNCTION:      PrExpandMacros
98 *
99 * PARAMETERS:    Line          - Pointer into the current line
100 *
101 * RETURN:        Updated pointer into the current line
102 *
103 * DESCRIPTION:    Expand any macros found in the current line buffer.
104 *
105 *****/

107 static char *
108 PrExpandMacros (
109     char                *Line)
110 {
111     char                *Token;
112     char                *ReplaceString;
113     PR_DEFINE_INFO      *DefineInfo;
114     ACPI_SIZE           TokenOffset;
115     char                *Next;
116     int                 OffsetAdjust;

119     strcpy (Gbl_ExpressionTokenBuffer, Gbl_CurrentLineBuffer);
120     Token = PrGetNextToken (Gbl_ExpressionTokenBuffer, PR_EXPR_SEPARATORS, &Next
121     OffsetAdjust = 0;

123     while (Token)
124     {
125         DefineInfo = PrMatchDefine (Token);
126         if (DefineInfo)
```

```

127 {
128     if (DefineInfo->Body)
129     {
130         /* This is a macro. TBD: Is this allowed? */
131
132         DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
133             "Matched Macro: %s->%s\n",
134             Gbl_CurrentLineNumber, DefineInfo->Identifier,
135             DefineInfo->Replacement);
136
137         PrDoMacroInvocation (Gbl_ExpressionTokenBuffer, Token,
138             DefineInfo, &Next);
139     }
140     else
141     {
142         ReplaceString = DefineInfo->Replacement;
143
144         /* Replace the name in the original line buffer */
145
146         TokenOffset = Token - Gbl_ExpressionTokenBuffer + OffsetAdjust;
147         PrReplaceData (
148             &Gbl_CurrentLineBuffer[TokenOffset], strlen (Token),
149             ReplaceString, strlen (ReplaceString));
150
151         /* Adjust for length difference between old and new name length
152
153         OffsetAdjust += strlen (ReplaceString) - strlen (Token);
154
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 }
161
162 Token = PrGetNextToken (NULL, PR_EXPR_SEPARATORS, &Next);
163 }
164
165 return (Line);
166 }
167
168 /*****
169 *
170 * FUNCTION:      PrIsDefined
171 *
172 * PARAMETERS:    Identifier          - Name to be resolved
173 *
174 * RETURN:        64-bit boolean integer value
175 *
176 * DESCRIPTION:   Returns TRUE if the name is defined, FALSE otherwise (0).
177 *
178 * *****/
179
180 UINT64
181 PrIsDefined (
182     char          *Identifier)
183 {
184     UINT64      Value;
185     PR_DEFINE_INFO *DefineInfo;
186
187     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
188         "**** Is defined?: %s\n", Gbl_CurrentLineNumber, Identifier);
189
190     Value = 0; /* Default is "Not defined" -- FALSE */

```

```

194     DefineInfo = PrMatchDefine (Identifier);
195     if (DefineInfo)
196     {
197         Value = ACPI_UINT64_MAX; /* TRUE */
198     }
199
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));
203
204     return (Value);
205 }
206
207 /*****
208 *
209 * FUNCTION:      PrResolveDefine
210 *
211 * PARAMETERS:    Identifier          - Name to be resolved
212 *
213 * RETURN:        A 64-bit boolean integer value
214 *
215 * DESCRIPTION:   Returns TRUE if the name is defined, FALSE otherwise (0).
216 *
217 * *****/
218
219 UINT64
220 PrResolveDefine (
221     char          *Identifier)
222 {
223     UINT64      Value;
224     PR_DEFINE_INFO *DefineInfo;
225
226     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
227         "**** Resolve #define: %s\n", Gbl_CurrentLineNumber, Identifier);
228
229     Value = 0; /* Default is "Not defined" -- FALSE */
230
231     DefineInfo = PrMatchDefine (Identifier);
232     if (DefineInfo)
233     {
234         Value = ACPI_UINT64_MAX; /* TRUE */
235     }
236
237     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
238         "[#if defined %s] resolved to: %8.8X%8.8X\n",
239         Gbl_CurrentLineNumber, Identifier, ACPI_FORMAT_UINT64 (Value));
240
241     return (Value);
242 }
243
244 /*****
245 *
246 * FUNCTION:      PrResolveIntegerExpression
247 *
248 * PARAMETERS:    Line              - Pointer to integer expression
249 *                 ReturnValue      - Where the resolved 64-bit integer is
250 *                                     returned.
251 *
252 * RETURN:        Status
253 *
254 * DESCRIPTION:   Resolve an integer expression to a single value. Supports
255 *                 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 }
```

new/usr/src/common/acpica/compiler/prmacros.c

1

```
*****
16160 Fri Jan 17 07:29:45 2014
new/usr/src/common/acpica/compiler/prmacros.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: prmacros - Preprocessor #define macro support
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "dtcompiler.h"

48 #define _COMPONENT          ASL_PREPROCESSOR
49 ACPI_MODULE_NAME          ("prmacros")

52 /*****
53 *
54 * FUNCTION:      PrDumpPredefinedNames
55 *
56 * PARAMETERS:    None
57 *
58 * RETURN:        None
59 *
60 * DESCRIPTION:   Dump the list of #defines. Used as the preprocessor starts, to
```

new/usr/src/common/acpica/compiler/prmacros.c

2

```
61 *          display the names that were defined on the command line.
62 *          Debug information only.
63 *
64 *****/

66 void
67 PrDumpPredefinedNames (
68     void)
69 {
70     PR_DEFINE_INFO      *DefineInfo;

73     DefineInfo = Gbl_DefineList;
74     while (DefineInfo)
75     {
76         DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
77             "Predefined #define: %s->%s\n",
78             0, DefineInfo->Identifier, DefineInfo->Replacement);

80         DefineInfo = DefineInfo->Next;
81     }
82 }

85 /*****
86 *
87 * FUNCTION:      PrAddDefine
88 *
89 * PARAMETERS:    Identifier          - Name to be replaced
90 *                Replacement        - Replacement for Identifier
91 *                Persist            - Keep define across multiple compiles?
92 *
93 * RETURN:        A new define_info struct. NULL on error.
94 *
95 * DESCRIPTION:   Add a new #define to the global list
96 *
97 *****/

99 PR_DEFINE_INFO *
100 PrAddDefine (
101     char                *Identifier,
102     char                *Replacement,
103     BOOLEAN              Persist)
104 {
105     char                *IdentifierString;
106     char                *ReplacementString;
107     PR_DEFINE_INFO      *DefineInfo;

110     if (!Replacement)
111     {
112         Replacement = "";
113     }

115     /* Check for already-defined first */

117     DefineInfo = PrMatchDefine (Identifier);
118     if (DefineInfo)
119     {
120         DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID,
121             "#define: name already exists: %s\n",
122             Gbl_CurrentLineNumber, Identifier);

124     /*
125      * Name already exists. This is only an error if the target name
126      * is different.
```

```

127     /*
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     /* Match name and delete the node */
186
187     DefineInfo = Gbl_DefineList;
188     while (DefineInfo)
189     {
190         if (!strcmp (DefineName, DefineInfo->Identifier))
191         {

```

```

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     DefineInfo = Gbl_DefineList;
245     while (DefineInfo)
246     {
247         if (!strcmp (MatchString, DefineInfo->Identifier))
248         {
249             return (DefineInfo);
250         }
251         DefineInfo = DefineInfo->Next;
252     }
253
254     return (NULL);
255 }

```

```

260 /*****
261 *
262 * FUNCTION:      PrAddMacro
263 *
264 * PARAMETERS:   Name          - Start of the macro definition
265 *               Next          - "Next" buffer from GetNextToken
266 *
267 * RETURN:       None
268 *
269 * DESCRIPTION:  Add a new macro to the list of #defines. Handles argument
270 *               processing.
271 *
272 *****/
274 void
275 PrAddMacro (
276     char          *Name,
277     char          **Next)
278 {
279     char          *Token = NULL;
280     ACPI_SIZE     TokenOffset;
281     ACPI_SIZE     MacroBodyOffset;
282     PR_DEFINE_INFO *DefineInfo;
283     PR_MACRO_ARG  *Args;
284     char          *Body;
285     char          *BodyInSource;
286     UINT32        i;
287     UINT16        UseCount = 0;
288     UINT16        ArgCount = 0;
289     UINT32        Depth = 1;
290     UINT32        EndOfArgList;
291     char          BufferChar;
292
293     /* Find the end of the arguments list */
294
295     TokenOffset = Name - Gbl_MainTokenBuffer + strlen (Name) + 1;
296     while (1)
297     {
298         BufferChar = Gbl_CurrentLineBuffer[TokenOffset];
299         if (BufferChar == '(')
300         {
301             Depth++;
302         }
303         else if (BufferChar == ')')
304         {
305             Depth--;
306         }
307         else if (BufferChar == 0)
308         {
309             PrError (ASL_ERROR, ASL_MSG_MACRO_SYNTAX, TokenOffset);
310             return;
311         }
312
313         if (Depth == 0)
314         {
315             /* Found arg list end */
316
317             EndOfArgList = TokenOffset;
318             break;
319         }
320
321         TokenOffset++;
322     }
323 }

```

```

325 /* At this point, we know that we have a reasonable argument list */
326
327 Args = UtLocalCalloc (sizeof (PR_MACRO_ARG) * PR_MAX_MACRO_ARGS);
328
329 /* Get the macro argument names */
330
331 for (i = 0; i < PR_MAX_MACRO_ARGS; i++)
332 {
333     Token = PrGetNextToken (NULL, PR_MACRO_SEPARATORS, Next);
334     if (!Token)
335     {
336         /* This is the case for a NULL macro body */
337
338         BodyInSource = "";
339         goto AddMacroToList;
340     }
341
342     /* Don't go beyond the argument list */
343
344     TokenOffset = Token - Gbl_MainTokenBuffer + strlen (Token);
345     if (TokenOffset > EndOfArgList)
346     {
347         break;
348     }
349
350     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
351         "Macro arg: %s \n",
352         Gbl_CurrentLineNumber, Token);
353
354     Args[i].Name = UtLocalCalloc (strlen (Token) + 1);
355     strcpy (Args[i].Name, Token);
356
357     Args[i].UseCount = 0;
358
359     ArgCount++;
360     if (ArgCount >= PR_MAX_MACRO_ARGS)
361     {
362         PrError (ASL_ERROR, ASL_MSG_TOO_MANY_ARGUMENTS, TokenOffset);
363         return;
364     }
365 }
366
367 /* Get the macro body. Token now points to start of body */
368
369 MacroBodyOffset = Token - Gbl_MainTokenBuffer;
370
371 /* Match each method arg in the macro body for later use */
372
373 Token = PrGetNextToken (NULL, PR_MACRO_SEPARATORS, Next);
374 while (Token)
375 {
376     /* Search the macro arg list for matching arg */
377
378     for (i = 0; Args[i].Name && (i < PR_MAX_MACRO_ARGS); i++)
379     {
380         /*
381          * Save argument offset within macro body. This is the mechanism
382          * used to expand the macro upon invocation.
383          *
384          * Handles multiple instances of the same argument
385          */
386         if (!strcmp (Token, Args[i].Name))
387         {
388             UseCount = Args[i].UseCount;
389
390             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]);

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));

403             return;
404         }
405         break;
406     }
407 }

409     Token = PrGetNextToken (NULL, PR_MACRO_SEPARATORS, Next);
410 }

412     BodyInSource = &Gbl_CurrentLineBuffer[MacroBodyOffset];

415 AddMacroToList:

417     /* Check if name is already defined first */

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);

426         /* Error only if not exactly the same macro */

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         }

435         return;
436     }

438     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
439     "Macro body: %s \n",
440     Gbl_CurrentLineNumber, BodyInSource);

442     /* Add macro to the #define list */

444     DefineInfo = PrAddDefine (Name, BodyInSource, FALSE);
445     if (DefineInfo)
446     {
447         Body = UtLocalCalloc (strlen (BodyInSource) + 1);
448         strcpy (Body, BodyInSource);

450         DefineInfo->Body = Body;
451         DefineInfo->Args = Args;
452         DefineInfo->ArgCount = ArgCount;
453     }
454 }

```

```

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 *****/

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;

487     /* Take a copy of the macro body for expansion */

489     strcpy (Gbl_MacroTokenBuffer, DefineInfo->Body);

491     /* Replace each argument within the prototype body */

493     Args = DefineInfo->Args;
494     if (!Args->Name)
495     {
496         /* This macro has no arguments */

498         Token = PrGetNextToken (NULL, PR_MACRO_ARGUMENTS, Next);
499         if (!Token)
500         {
501             goto BadInvocation;
502         }

504         TokenOffset = (MacroStart - TokenBuffer);
505         Length = Token - MacroStart + strlen (Token) + 1;

507         PrReplaceData (
508             &Gbl_CurrentLineBuffer[TokenOffset], Length,
509             Gbl_MacroTokenBuffer, strlen (Gbl_MacroTokenBuffer));
510         return;
511     }

513     while (Args->Name)
514     {
515         /* Get the next argument from macro invocation */

517         Token = PrGetNextToken (NULL, PR_MACRO_SEPARATORS, Next);
518         if (!Token)
519         {
520             goto BadInvocation;
521         }

```



```
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 */
530         if (Args->Offset[i] == 0)
531         {
532             break;
533         }
535         PrReplaceData (
536             &Gbl_MacroTokenBuffer[Args->Offset[i]], strlen (Args->Name),
537             Token, strlen (Token));
539         DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
540             "ExpandArg: %s \n",
541             Gbl_CurrentLineNumber, Gbl_MacroTokenBuffer);
542     }
544     Args++;
545 }
547 /* TBD: need to make sure macro was not invoked with too many arguments */
549 if (!Token)
550 {
551     return;
552 }
554 /* Replace the entire macro invocation with the expanded macro */
556 TokenOffset = (MacroStart - TokenBuffer);
557 Length = Token - MacroStart + strlen (Token) + 1;
559 PrReplaceData (
560     &Gbl_CurrentLineBuffer[TokenOffset], Length,
561     Gbl_MacroTokenBuffer, strlen (Gbl_MacroTokenBuffer));
563 return;
566 BadInvocation:
567     PrError (ASL_ERROR, ASL_MSG_INVALID_INVOCATION,
568         THIS_TOKEN_OFFSET (MacroStart));
570     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
571         "Bad macro invocation: %s \n",
572         Gbl_CurrentLineNumber, Gbl_MacroTokenBuffer);
573     return;
574 }
```

new/usr/src/common/acpica/compiler/prparser.1

1

```
*****
4947 Fri Jan 17 07:29:46 2014
new/usr/src/common/acpica/compiler/prparser.1
update to acpica-unix2-20140114
acpica-unix2-20130823
*****

1 %{
2 /*****
3 *
4 * Module Name: prparser.1 - Flex input file for preprocessor lexer
5 *
6 *****/

8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "aslcompiler.h"
46 #include "prparser.y.h"

48 /* Buffer to pass strings to the parser */

50 #define STRING_SETUP strcpy (StringBuffer, PrParseText); \
51 PrParserIval.str = StringBuffer

53 #define YY_NO_INPUT /* No file input, we use strings only */

55 #define _COMPONENT ACPI_COMPILER
56 ACPI_MODULE_NAME ("prscanner")
57 %}

59 %option noyywrap
60 %option nounput
```

new/usr/src/common/acpica/compiler/prparser.1

2

```
62 Number [0-9a-fA-F]+
63 HexNumber 0[xX][0-9a-fA-F]+
64 WhiteSpace [ \t\v\r]+
65 NewLine [\n]
66 Identifier [a-zA-Z][0-9a-zA-Z]*

68 %%

70 \(\ return (EXPOP_PAREN_OPEN);
71 \) return (EXPOP_PAREN_CLOSE);
72 \~ return (EXPOP_ONES_COMPLIMENT);
73 \! return (EXPOP_LOGICAL_NOT);
74 \* return (EXPOP_MULTIPLY);
75 \/ return (EXPOP_DIVIDE);
76 \% return (EXPOP_MODULO);
77 \+ return (EXPOP_ADD);
78 \- return (EXPOP_SUBTRACT);
79 ">>" return (EXPOP_SHIFT_RIGHT);
80 "<<" return (EXPOP_SHIFT_LEFT);
81 "<" return (EXPOP_LESS);
82 ">" return (EXPOP_GREATER);
83 "<=" return (EXPOP_LESS_EQUAL);
84 ">=" return (EXPOP_GREATER_EQUAL);
85 "==" return (EXPOP_EQUAL);
86 "!=" return (EXPOP_NOT_EQUAL);
87 "&" return (EXPOP_AND);
88 "\^" return (EXPOP_XOR);
89 "\|" return (EXPOP_OR);
90 "&&" return (EXPOP_LOGICAL_AND);
91 "||" return (EXPOP_LOGICAL_OR);

93 "defined" return (EXPOP_DEFINE);
94 {Identifier} {STRING_SETUP; return (EXPOP_IDENTIFIER);}

96 <<EOF>> return (EXPOP_EOF); /* null end-of-string */

98 {Number} return (EXPOP_NUMBER);
99 {HexNumber} return (EXPOP_HEX_NUMBER);
100 {NewLine} return (EXPOP_NEW_LINE);
101 {WhiteSpace} /* Ignore */

103 . return (EXPOP_EOF);
104 %%

106 /*
107 * Local support functions
108 */
109 YY_BUFFER_STATE LexBuffer;

112 /*****
113 *
114 * FUNCTION: PrInitLexer
115 *
116 * PARAMETERS: String - Input string to be parsed
117 *
118 * RETURN: TRUE if parser returns NULL. FALSE otherwise.
119 *
120 * DESCRIPTION: Initialization routine for lexer. The lexer needs
121 * a buffer to handle strings instead of a file.
122 *
123 *****/

125 int
126 PrInitLexer (
```

```
127     char                *String)
128 {
129
130     LexBuffer = yy_scan_string (String);
131     return (LexBuffer == NULL);
132 }
133
134
135 /*****
136  *
137  * FUNCTION:      PrTerminateLexer
138  *
139  * PARAMETERS:    None
140  *
141  * RETURN:        None
142  *
143  * DESCRIPTION:   Termination routine for thelexer.
144  *
145  *****/
146
147 void
148 PrTerminateLexer (
149     void)
150 {
151
152     yy_delete_buffer (LexBuffer);
153 }
```

new/usr/src/common/acpica/compiler/prparser.y

1

```
*****
10174 Fri Jan 17 07:29:46 2014
new/usr/src/common/acpica/compiler/prparser.y
update to acpica-unix2-20140114
acpica-unix2-20130823
*****

1 %{
2 /*****
3 *
4 * Module Name: prparser.y - Bison input file for preprocessor parser
5 *
6 *****/

8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "aslcompiler.h"
46 #include "dtcompiler.h"

48 #define _COMPONENT          ASL_PREPROCESSOR
49         ACPI_MODULE_NAME    ("prparser")

51 int          PrParserlex (void);
52 int          PrParserparse (void);
53 void         PrParsererror (char const *msg);
54 extern char   *PrParseertext;

56 UINT64       PrParserResult; /* Expression return value */

58 /* Bison/yacc configuration */

60 #define yyname             PrParsername
```

new/usr/src/common/acpica/compiler/prparser.y

2

```
61 #define YYDEBUG           1          /* Enable debug output */
62 #define YYERROR_VERBOSE   1          /* Verbose error messages */
63 #define YYFLAG             -32768

65 /* Define YYMALLOC/YYFREE to prevent redefinition errors */

67 #define YYMALLOC           malloc
68 #define YYFREE             free
69 %}

71 union
72 {
73     UINT64         value;
74     UINT32         op;
75     char           *str;
76 }

78 /*! [Begin] no source code translation */

80 %type <value> Expression

82 %token <op>         EXPOP_EOF
83 %token <op>         EXPOP_NEW_LINE
84 %token <op>         EXPOP_NUMBER
85 %token <op>         EXPOP_HEX_NUMBER
86 %token <op>         EXPOP_RESERVED1
87 %token <op>         EXPOP_RESERVED2
88 %token <op>         EXPOP_PAREN_OPEN
89 %token <op>         EXPOP_PAREN_CLOSE

91 %left <op>          EXPOP_LOGICAL_OR
92 %left <op>          EXPOP_LOGICAL_AND
93 %left <op>          EXPOP_OR
94 %left <op>          EXPOP_XOR
95 %left <op>          EXPOP_AND
96 %left <op>          EXPOP_EQUAL EXPOP_NOT_EQUAL
97 %left <op>          EXPOP_GREATER EXPOP_LESS EXPOP_GREATER_EQUAL EXPOP_LESS_EQUAL
98 %left <op>          EXPOP_SHIFT_RIGHT EXPOP_SHIFT_LEFT
99 %left <op>          EXPOP_ADD EXPOP_SUBTRACT
100 %left <op>          EXPOP_MULTIPLY EXPOP_DIVIDE EXPOP_MODULO
101 %right <op>         EXPOP_ONES_COMPLIMENT EXPOP_LOGICAL_NOT

103 /* Tokens above must be kept in synch with dtparser.y */

105 %token <op>         EXPOP_DEFINE
106 %token <op>         EXPOP_IDENTIFIER

108 %%

110 /*
111 * Operator precedence rules (from K&R)
112 *
113 * 1)      ( )
114 * 2)      ! ~ (unary operators that are supported here)
115 * 3)      * / %
116 * 4)      + -
117 * 5)      >> <<
118 * 6)      < > <= >=
119 * 7)      == !=
120 * 8)      &
121 * 9)      ^
122 * 10)     |
123 * 11)     &&
124 * 12)     ||
125 */
```

```

127 /*! [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 {
261
262     DbgPrint (ASL_DEBUG_OUTPUT,
263     "**** Input expression: %s\n", ExprString);
264
265     /* Point lexer to the input string */
266
267     if (PrInitLexer (ExprString))
268     {
269         DtError (ASL_ERROR, ASL_MSG_COMPILER_INTERNAL,
270         NULL, "Could not initialize lexer");
271         return (0);
272     }
273
274     /* Parse/Evaluate the input string (value returned in PrParserResult) */
275
276     PrParserparse ();
277     PrTerminateLexer ();
278
279     DbgPrint (ASL_DEBUG_OUTPUT,
280     "**** Parser returned value: %u (%8.8X%8.8X)\n",
281     (UINT32) PrParserResult, ACPI_FORMAT_UINT64 (PrParserResult));
282
283     return (PrParserResult);
284 }
```

new/usr/src/common/acpica/compiler/prscan.c

1

```
*****
26485 Fri Jan 17 07:29:46 2014
new/usr/src/common/acpica/compiler/prscan.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: prscan - Preprocessor start-up and file scan module
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define _DECLARE_PR_GLOBALS
45
46 #include "aslcompiler.h"
47 #include "dtcompiler.h"
48
49 /*
50 * TBDs:
51 *
52 * No nested macros, maybe never
53 * Implement ASL "Include" as well as "#include" here?
54 */
55 #define _COMPONENT ASL_PREPROCESSOR
56 ACPI_MODULE_NAME ("prscan")
57
58
59 /* Local prototypes */
```

new/usr/src/common/acpica/compiler/prscan.c

2

```
61 static void
62 PrPreprocessInputFile (
63     void);
64
65 static void
66 PrDoDirective (
67     char *DirectiveToken,
68     char **Next);
69
70 static int
71 PrMatchDirective (
72     char *Directive);
73
74 static void
75 PrPushDirective (
76     int Directive,
77     char *Argument);
78
79 static ACPI_STATUS
80 PrPopDirective (
81     void);
82
83 static void
84 PrDbgPrint (
85     char *Action,
86     char *DirectiveName);
87
88 /*
89 * Supported preprocessor directives
90 */
91
92 static const PR_DIRECTIVE_INFO Gbl_DirectiveInfo[] =
93 {
94     {"define", 1},
95     {"elif", 0}, /* Converted to #else..#if internally */
96     {"else", 0},
97     {"endif", 0},
98     {"error", 1},
99     {"if", 1},
100     {"ifdef", 1},
101     {"ifndef", 1},
102     {"include", 0}, /* Argument is not standard format, so 0 */
103     {"line", 1},
104     {"pragma", 1},
105     {"undef", 1},
106     {"warning", 1},
107     {NULL, 0}
108 };
109
110 enum Gbl_DirectiveIndexes
111 {
112     PR_DIRECTIVE_DEFINE = 0,
113     PR_DIRECTIVE_ELIF,
114     PR_DIRECTIVE_ELSE,
115     PR_DIRECTIVE_ENDIF,
116     PR_DIRECTIVE_ERROR,
117     PR_DIRECTIVE_IF,
118     PR_DIRECTIVE_IFDEF,
119     PR_DIRECTIVE_IFNDEF,
120     PR_DIRECTIVE_INCLUDE,
121     PR_DIRECTIVE_LINE,
122     PR_DIRECTIVE_PRAGMA,
123     PR_DIRECTIVE_UNDEF,
124     PR_DIRECTIVE_WARNING,
125 };
```

```

127 #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:
524
525         PrDbgPrint ("Executing", "endif");
526
527         /* Pop the owning #if/#ifdef/#ifndef */
528
529         if (ACPI_FAILURE (PrPopDirective ()))
530         {
531             PrError (ASL_ERROR, ASL_MSG_ENDIF_MISMATCH,
532                 THIS_TOKEN_OFFSET (DirectiveToken));
533         }
534         return;
535
536     default:
537         break;
538 }
539
540 /* Most directives have at least one argument */
541
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 }
550
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:
564
565         PrPushDirective (Directive, Token);
566         PrDbgPrint ("Ignoring", Gbl_DirectiveInfo[Directive].Name);
567         break;
568
569     default:
570         break;
571     }
572
573     return;
574 }
575
576 /*
577  * Execute the directive
578  */
579 PrDbgPrint ("Begin execution", Gbl_DirectiveInfo[Directive].Name);
580
581 switch (Directive)
582 {
583 case PR_DIRECTIVE_IF:
584
585     TokenOffset = Token - Gbl_MainTokenBuffer;
586
587     /* Need to expand #define macros in the expression string first */

```

```

588     Status = PrResolveIntegerExpression (
589         &Gbl_CurrentLineBuffer[TokenOffset-1], &Value);
590     if (ACPI_FAILURE (Status))
591     {
592         return;
593     }
594
595     PrPushDirective (Directive, Token);
596     if (!Value)
597     {
598         Gbl_IgnoringThisCodeBlock = TRUE;
599     }
600
601     DbgPrint (ASL_DEBUG_OUTPUT, PR_PREFIX_ID
602         "Resolved #if: %8.8X%8.8X %s\n",
603         Gbl_CurrentLineNumber, ACPI_FORMAT_UINT64 (Value),
604         Gbl_IgnoringThisCodeBlock ? "<Skipping Block>" : "<Executing Block>");
605     break;
606
607 case PR_DIRECTIVE_IFDEF:
608
609     PrPushDirective (Directive, Token);
610     if (!PrMatchDefine (Token))
611     {
612         Gbl_IgnoringThisCodeBlock = TRUE;
613     }
614
615     PrDbgPrint ("Evaluated", "ifdef");
616     break;
617
618 case PR_DIRECTIVE_IFNDEF:
619
620     PrPushDirective (Directive, Token);
621     if (PrMatchDefine (Token))
622     {
623         Gbl_IgnoringThisCodeBlock = TRUE;
624     }
625
626     PrDbgPrint ("Evaluated", "ifndef");
627     break;
628
629 case PR_DIRECTIVE_DEFINE:
630     /*
631      * By definition, if first char after the name is a paren,
632      * this is a function macro.
633      */
634     TokenOffset = Token - Gbl_MainTokenBuffer + strlen (Token);
635     if ((*(&Gbl_CurrentLineBuffer[TokenOffset]) == '('))
636     {
637         #ifndef MACROS_SUPPORTED
638             AcpiOsPrintf ("%s ERROR - line %u: #define macros are not supported\n",
639                 Gbl_CurrentLineBuffer, Gbl_CurrentLineNumber);
640             exit(1);
641         #else
642             PrAddMacro (Token, Next);
643         #endif
644     }
645     else
646     {
647         /* Use the remainder of the line for the #define */
648
649         Token2 = *Next;
650         if (Token2)
651         {
652             while ((*Token2 == ' ') || (*Token2 == '\t'))
653                 Token2++;
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 */

```

```

920     Gbl_IfDepth--;
921     Gbl_IgnoringThisCodeBlock = Info->IgnoringThisCodeBlock;
922     Gbl_DirectiveStack = Info->Next;

924     DbgPrint (ASL_DEBUG_OUTPUT,
925         "Pr(%.4u) - [%u %s] %*s Popped [%s %s]: IgnoreFlag now = %s\n",
926         Gbl_CurrentLineNumber, Gbl_IfDepth,
927         Gbl_IgnoringThisCodeBlock ? "I" : "E",
928         Gbl_IfDepth * 4, " ",
929         Gbl_DirectiveInfo[Info->Directive].Name,
930         Info->Argument, Gbl_IgnoringThisCodeBlock ? "TRUE" : "FALSE");

932     ACPI_FREE (Info);
933     return (AE_OK);
934 }

937 /*****
938  *
939  * FUNCTION:      PrDbgPrint
940  *
941  * PARAMETERS:    Action              - Action being performed
942  *                DirectiveName        - Directive being processed
943  *
944  * RETURN:        None
945  *
946  * DESCRIPTION:   Special debug print for directive processing.
947  *
948  *****/

950 static void
951 PrDbgPrint (
952     char          *Action,
953     char          *DirectiveName)
954 {
956     DbgPrint (ASL_DEBUG_OUTPUT, "Pr(%.4u) - [%u %s] "
957         "%*s %s #s, Depth %u\n",
958         Gbl_CurrentLineNumber, Gbl_IfDepth,
959         Gbl_IgnoringThisCodeBlock ? "I" : "E",
960         Gbl_IfDepth * 4, " ",
961         Action, DirectiveName, Gbl_IfDepth);
962 }

```

```

*****
13064 Fri Jan 17 07:29:46 2014
new/usr/src/common/acpica/compiler/prutils.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: prutils - Preprocessor utilities
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "dtcompiler.h"

48 #define _COMPONENT          ASL_PREPROCESSOR
49 ACPI_MODULE_NAME("prutils")

52 /*****
53 *
54 * FUNCTION:      PrGetNextToken
55 *
56 * PARAMETERS:    Buffer          - Current line buffer
57 *                MatchString    - String with valid token delimiters
58 *                Next           - Set to next possible token in buffer
59 *
60 * RETURN:        Next token (null-terminated). Modifies the input line.

```

```

61 *          Remainder of line is stored in *Next.
62 *
63 * DESCRIPTION: Local implementation of strtok() with local storage for the
64 * next pointer. Not only thread-safe, but allows multiple
65 * parsing of substrings such as expressions.
66 *
67 *****/

69 char *
70 PrGetNextToken (
71     char          *Buffer,
72     char          *MatchString,
73     char          **Next)
74 {
75     char          *TokenStart;

78     if (!Buffer)
79     {
80         /* Use Next if it is valid */

82         Buffer = *Next;
83         if (!(*Next))
84         {
85             return (NULL);
86         }
87     }

89     /* Skip any leading delimiters */

91     while (*Buffer)
92     {
93         if (strchr (MatchString, *Buffer))
94         {
95             Buffer++;
96         }
97         else
98         {
99             break;
100         }
101     }

103     /* Anything left on the line? */

105     if (!(*Buffer))
106     {
107         *Next = NULL;
108         return (NULL);
109     }

111     TokenStart = Buffer;

113     /* Find the end of this token */

115     while (*Buffer)
116     {
117         if (strchr (MatchString, *Buffer))
118         {
119             *Buffer = 0;
120             *Next = Buffer+1;
121             if (!**Next)
122             {
123                 *Next = NULL;
124             }
125             return (TokenStart);
126         }

```

```

127     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 }

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 }

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     }

295     NextDir = NextDir->Next;
296 }

298 /* We could not open the include file after trying very hard */

300 ErrorExit:
301     sprintf (Gbl_MainTokenBuffer, "%s, %s", Filename, strerror (errno));
302     PrError (ASL_ERROR, ASL_MSG_INCLUDE_FILE_OPEN, 0);
303 }

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  *****/

320 FILE *
321 PrOpenIncludeWithPrefix (
322     char      *PrefixDir,
323     char      *Filename)
324 {

```

```

325     FILE      *IncludeFile;
326     char      *Pathname;

329     /* Build the full pathname to the file */

331     Pathname = FlMergePathnames (PrefixDir, Filename);

333     DbgPrint (ASL_PARSE_OUTPUT, PR_PREFIX_ID
334         "Include: Opening file - \"%s\"\n",
335         Gbl_CurrentLineNumber, Pathname);

337     /* Attempt to open the file, push if successful */

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     }

347     /* Push the include file on the open input file stack */

349     PrPushInputFileStack (IncludeFile, Pathname);
350     return (IncludeFile);
351 }

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  *****/

369 void
370 PrPushInputFileStack (
371     FILE      *InputFile,
372     char      *Filename)
373 {
374     PR_FILE_NODE *Fnode;

377     /* Save the current state in an Fnode */

379     Fnode = UtLocalCalloc (sizeof (PR_FILE_NODE));

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;

386     /* Push it on the stack */

388     Gbl_InputFileList = Fnode;

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 }

```

new/usr/src/common/acpica/compiler/readme.txt

1

\*\*\*\*\*

4383 Fri Jan 17 07:29:46 2014  
new/usr/src/common/acpica/compiler/readme.txt  
update to acpica-unix2-20131218  
acpica-unix2-20130823  
\*\*\*\*\*

```
1 /*
2  * Miscellaneous instructions for building and using the iASL compiler.
3  */
4 Last update 9 December 2013.
```

```
7 1) Generating iASL from source
8 -----
```

10 Generation of the ASL compiler from source code requires these items:

- 12 1) The ACPICA source code tree.
- 13 2) An ANSI C compiler.
- 14 3) The Flex (or Lex) lexical analyzer generator.
- 15 4) The Bison (or Yacc) parser generator.

17 There are three major ACPICA source code components that are required to  
18 generate the compiler (Basically, the entire ACPICA source tree should  
19 be installed):

- 21 1) The ASL compiler source.
- 22 2) The ACPICA Core Subsystem source. In particular, the Namespace  
23 Manager component is used to create an internal ACPI namespace  
24 and symbol table, and the AML Interpreter is used to evaluate  
25 constant expressions.
- 26 3) The "common" source directory that is used for all ACPI components.

```
29 1a) Notes for Linux/Unix generation
30 -----
```

32 iASL has been generated with these versions of Flex/Bison:

```
34 flex: Version 2.5.32
35 bison: Version 2.6.2
```

37 Other required packages:

```
39 make
40 gcc C compiler
41 m4 (macro processor required by bison)
```

43 On Linux/Unix systems, the following commands will build the compiler:

```
45 cd acpica (or cd acpica/generate/unix)
46 make clean
47 make iasl
```

```
50 1b) Notes for Windows generation
51 -----
```

53 On Windows, the Visual Studio 2008 project file appears in this directory:

```
55 generate/msvc9/AcpiComponents.sln
```

57 The Windows versions of GNU Flex/Bison must be installed, and they must  
58 be installed in a directory that contains no embedded spaces in the  
59 pathname. They cannot be installed in the default "c:\Program Files"  
60 directory. This is a bug in Bison. The default Windows project file for

new/usr/src/common/acpica/compiler/readme.txt

2

61 iASL assumes that these tools are installed at this location:

```
63 c:\GnuWin32
```

65 Once the tools are installed, ensure that this path is added to the  
66 default system \$Path environment variable:

```
68 c:\GnuWin32\bin
```

70 Goto: ControlPanel/System/AdvancedSystemSettings/EnvironmentVariables

72 Important: Now Windows must be rebooted to make the system aware of  
73 the updated \$Path. Otherwise, Bison will not be able to find the M4  
74 interpreter library and will fail.

76 iASL has been generated with these versions of Flex/Bison for Windows:

```
78 Flex for Windows: V2.5.4a
79 Bison for Windows: V2.4.1
```

81 Flex is available at: <http://gnuwin32.sourceforge.net/packages/flex.htm>  
82 Bison is available at: <http://gnuwin32.sourceforge.net/packages/bison.htm>

```
86 2) Integration as a custom tool for Visual Studio
87 -----
```

89 This procedure adds the iASL compiler as a custom tool that can be used  
90 to compile ASL source files. The output is sent to the VC output  
91 window.

93 a) Select Tools->Customize.

95 b) Select the "Tools" tab.

97 c) Scroll down to the bottom of the "Menu Contents" window. There you  
98 will see an empty rectangle. Click in the rectangle to enter a  
99 name for this tool.

101 d) Type "iASL Compiler" in the box and hit enter. You can now edit  
102 the other fields for this new custom tool.

104 e) Enter the following into the fields:

```
106 Command: C:\Acpi\iasl.exe
107 Arguments: -vi "$(FilePath)"
108 Initial Directory "$(FileDir)"
109 Use Output Window <Check this option>
```

111 "Command" must be the path to wherever you copied the compiler.  
112 "-vi" instructs the compiler to produce messages appropriate for VC.  
113 Quotes around FilePath and FileDir enable spaces in filenames.

115 f) Select "Close".

117 These steps will add the compiler to the tools menu as a custom tool.  
118 By enabling "Use Output Window", you can click on error messages in  
119 the output window and the source file and source line will be  
120 automatically displayed by VC. Also, you can use F4 to step through  
121 the messages and the corresponding source line(s).

```
125 3) Integrating iASL into a Visual Studio ASL project build
126 -----
```

```
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 Fri Jan 17 07:29:46 2014
new/usr/src/common/acpica/components/debugger/dbcmds.c
update to acpica-unix2-20140114
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dbcmds - Miscellaneous debug commands and output routines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acevents.h"
48 #include "acdebug.h"
49 #include "acnamesp.h"
50 #include "acresrc.h"
51 #include "actables.h"

53 #ifdef ACPI_DEBUGGER

55 #define _COMPONENT          ACPI_CA_DEBUGGER
56 #define ACPI_MODULE_NAME    ("dbcmds")

```

```

59 /* Local prototypes */

61 static void
62 AcpiDmCompareAmlResources (
63     UINT8                *Aml1Buffer,
64     ACPI_RSDESC_SIZE      Aml1BufferLength,
65     UINT8                *Aml2Buffer,
66     ACPI_RSDESC_SIZE      Aml2BufferLength);

68 static ACPI_STATUS
69 AcpiDmTestResourceConversion (
70     ACPI_NAMESPACE_NODE   *Node,
71     char                  *Name);

73 static ACPI_STATUS
74 AcpiDbResourceCallback (
75     ACPI_RESOURCE          *Resource,
76     void                  *Context);

78 static ACPI_STATUS
79 AcpiDbDeviceResources (
80     ACPI_HANDLE            ObjHandle,
81     UINT32                NestingLevel,
82     void                  *Context,
83     void                  **ReturnValue);

85 static void
86 AcpiDbDoOneSleepState (
87     UINT8                  SleepState);

90 /*****
91 *
92 * FUNCTION:      AcpiDbConvertToNode
93 *
94 * PARAMETERS:    InString          - String to convert
95 *
96 * RETURN:        Pointer to a NS node
97 *
98 * DESCRIPTION:   Convert a string to a valid NS pointer. Handles numeric or
99 * alphanumeric strings.
100 *
101 *****/

103 ACPI_NAMESPACE_NODE *
104 AcpiDbConvertToNode (
105     char                  *InString)
106 {
107     ACPI_NAMESPACE_NODE   *Node;
108     ACPI_SIZE              Address;

111     if ((*InString >= 0x30) && (*InString <= 0x39))
112     {
113         /* Numeric argument, convert */

115         Address = ACPI_STRTOUL (InString, NULL, 16);
116         Node = ACPI_TO_POINTER (Address);
117         if (!AcpiOsReadable (Node, sizeof (ACPI_NAMESPACE_NODE)))
118         {
119             AcpiOsPrintf ("Address %p is invalid in this address space\n",
120                 Node);
121             return (NULL);
122         }
124         /* 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);
258     if (ACPI_FAILURE (Status))
259     {
260         goto ErrorExit;
261     }

263     AcpiOsPrintf ("**** Wake: Prepare to return from sleep (S%d) ****\n",
264         SleepState);
265     Status = AcpiLeaveSleepStatePrep (SleepState);
266     if (ACPI_FAILURE (Status))
267     {
268         goto ErrorExit;
269     }

271     AcpiOsPrintf ("**** Wake: Return from sleep (S%d) ****\n",
272         SleepState);
273     Status = AcpiLeaveSleepState (SleepState);
274     if (ACPI_FAILURE (Status))
275     {
276         goto ErrorExit;
277     }

279     return;

282 ErrorExit:
283     ACPI_EXCEPTION ((AE_INFO, Status, "During invocation of sleep state S%d",
284         SleepState));
285 }

288 /*****
289  *
290  * FUNCTION:      AcpiDbDisplayLocks
291  *
292  * PARAMETERS:    None
293  *
294  * RETURN:        None
295  *
296  * DESCRIPTION:   Display information about internal mutexes.
297  *
298  *****/

300 void
301 AcpiDbDisplayLocks (
302     void)
303 {
304     UINT32          i;

307     for (i = 0; i < ACPI_MAX_MUTEX; i++)
308     {
309         AcpiOsPrintf ("%26s : %s\n", AcpiUtGetMutexName (i),
310             AcpiGbl_MutexInfo[i].ThreadId == ACPI_MUTEX_NOT_ACQUIRED
311             ? "Locked" : "Unlocked");
312     }
313 }

316 /*****
317  *
318  * FUNCTION:      AcpiDbDisplayTableInfo
319  *
320  * PARAMETERS:    TableArg          - Name of table to be displayed
321  *
322  * RETURN:        None

```

```

323  *
324  * DESCRIPTION:   Display information about loaded tables. Current
325  *               implementation displays all loaded tables.
326  *
327  *****/

329 void
330 AcpiDbDisplayTableInfo (
331     char            *TableArg)
332 {
333     UINT32          i;
334     ACPI_TABLE_DESC *TableDesc;
335     ACPI_STATUS      Status;

338     /* Header */
340     AcpiOsPrintf ("Idx ID Status      Type              Sig Address Len  Header\n");

342     /* Walk the entire root table list */

344     for (i = 0; i < AcpiGbl_RootTableList.CurrentTableCount; i++)
345     {
346         TableDesc = &AcpiGbl_RootTableList.Tables[i];

348         /* Index and Table ID */

350         AcpiOsPrintf ("%3u %2u ", i, TableDesc->OwnerId);

352         /* Decode the table flags */

354         if (!(TableDesc->Flags & ACPI_TABLE_IS_LOADED))
355         {
356             AcpiOsPrintf ("NotLoaded ");
357         }
358         else
359         {
360             AcpiOsPrintf ("    Loaded ");
361         }

363         switch (TableDesc->Flags & ACPI_TABLE_ORIGIN_MASK)
364         {
365             case ACPI_TABLE_ORIGIN_UNKNOWN:

367                 AcpiOsPrintf ("Unknown  ");
368                 break;

370             case ACPI_TABLE_ORIGIN_MAPPED:

372                 AcpiOsPrintf ("Mapped   ");
373                 break;

375             case ACPI_TABLE_ORIGIN_ALLOCATED:

377                 AcpiOsPrintf ("Allocated ");
378                 break;

380             case ACPI_TABLE_ORIGIN_OVERRIDE:

382                 AcpiOsPrintf ("Override  ");
383                 break;

385             default:

387                 AcpiOsPrintf ("INVALID   ");
388                 break;

```

```

389     }

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\n",
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
522 *
523 *****/

525 void
526 AcpiDbDisplayInterfaces (
527     char                *ActionArg,
528     char                *InterfaceNameArg)
529 {
530     ACPI_INTERFACE_INFO *NextInterface;
531     char                *SubString;
532     ACPI_STATUS          Status;

535     /* If no arguments, just display current interface list */

537     if (!ActionArg)
538     {
539         (void) AcpiOsAcquireMutex (AcpiGbl_OsiMutex,
540             ACPI_WAIT_FOREVER);

542         NextInterface = AcpiGbl_SupportedInterfaces;
543         while (NextInterface)
544         {
545             if (!(NextInterface->Flags & ACPI_OSI_INVALID))
546             {
547                 AcpiOsPrintf ("%s\n", NextInterface->Name);
548             }
549             NextInterface = NextInterface->Next;
550         }

552         AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
553         return;
554     }

556     /* If ActionArg exists, so must InterfaceNameArg */

558     if (!InterfaceNameArg)
559     {
560         AcpiOsPrintf ("Missing Interface Name argument\n");
561         return;
562     }

564     /* Uppercase the action for match below */

566     AcpiUtStrupr (ActionArg);

568     /* Install - install an interface */

570     SubString = ACPI_STRSTR ("INSTALL", ActionArg);
571     if (SubString)
572     {
573         Status = AcpiInstallInterface (InterfaceNameArg);
574         if (ACPI_FAILURE (Status))
575         {
576             AcpiOsPrintf ("%s, while installing \"%s\"\n",
577                 AcpiFormatException (Status), InterfaceNameArg);
578         }
579         return;
580     }

582     /* Remove - remove an interface */

584     SubString = ACPI_STRSTR ("REMOVE", ActionArg);
585     if (SubString)
586     {

```

```

587     Status = AcpiRemoveInterface (InterfaceNameArg);
588     if (ACPI_FAILURE (Status))
589     {
590         AcpiOsPrintf ("%s, while removing \"%s\"\n",
591             AcpiFormatException (Status), InterfaceNameArg);
592     }
593     return;
594 }

596 /* Invalid ActionArg */

598     AcpiOsPrintf ("Invalid action argument: %s\n", ActionArg);
599     return;
600 }

603 *****/
604 *
605 * FUNCTION:      AcpiDbDisplayTemplate
606 *
607 * PARAMETERS:    BufferArg          - Buffer name or address
608 *
609 * RETURN:        None
610 *
611 * DESCRIPTION:    Dump a buffer that contains a resource template
612 *
613 *****/

615 void
616 AcpiDbDisplayTemplate (
617     char                *BufferArg)
618 {
619     ACPI_NAMESPACE_NODE *Node;
620     ACPI_STATUS          Status;
621     ACPI_BUFFER          ReturnBuffer;

624     /* Translate BufferArg to an Named object */

626     Node = AcpiDbConvertToNode (BufferArg);
627     if (!Node || (Node == AcpiGbl_RootNode))
628     {
629         AcpiOsPrintf ("Invalid argument: %s\n", BufferArg);
630         return;
631     }

633     /* We must have a buffer object */

635     if (Node->Type != ACPI_TYPE_BUFFER)
636     {
637         AcpiOsPrintf ("Not a Buffer object, cannot be a template: %s\n",
638             BufferArg);
639         return;
640     }

642     ReturnBuffer.Length = ACPI_DEBUG_BUFFER_SIZE;
643     ReturnBuffer.Pointer = AcpiGbl_DbBuffer;

645     /* Attempt to convert the raw buffer to a resource list */

647     Status = AcpiRsCreateResourceList (Node->Object, &ReturnBuffer);

649     AcpiDbSetOutputDestination (ACPI_DB_REDIRECTABLE_OUTPUT);
650     AcpiDbgLevel |= ACPI_LV_RESOURCES;

652     if (ACPI_FAILURE (Status))

```

```

653 {
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 *
786 * RETURN:      Status
787 *
788 * DESCRIPTION: Compare the original AML with a conversion of the AML to
789 *              internal resource list, then back to AML.
790 *
791 *****/

793 static ACPI_STATUS
794 AcpiDmTestResourceConversion (
795     ACPI_NAMESPACE_NODE *Node,
796     char *Name)
797 {
798     ACPI_STATUS Status;
799     ACPI_BUFFER ReturnBuffer;
800     ACPI_BUFFER ResourceBuffer;
801     ACPI_BUFFER NewAml;
802     ACPI_OBJECT *OriginalAml;

805     AcpiOsPrintf ("Resource Conversion Comparison:\n");

807     NewAml.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
808     ReturnBuffer.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
809     ResourceBuffer.Length = ACPI_ALLOCATE_LOCAL_BUFFER;

811     /* Get the original _CRS AML resource template */

813     Status = AcpiEvaluateObject (Node, Name, NULL, &ReturnBuffer);
814     if (ACPI_FAILURE (Status))
815     {
816         AcpiOsPrintf ("Could not obtain %s: %s\n",
817             Name, AcpiFormatException (Status));
818         return (Status);
819     }

821     /* Get the AML resource template, converted to internal resource structs */

823     Status = AcpiGetCurrentResources (Node, &ResourceBuffer);
824     if (ACPI_FAILURE (Status))
825     {
826         AcpiOsPrintf ("AcpiGetCurrentResources failed: %s\n",
827             AcpiFormatException (Status));
828         goto Exit1;
829     }

831     /* Convert internal resource list to external AML resource template */

833     Status = AcpiRsCreateAmlResources (&ResourceBuffer, &NewAml);
834     if (ACPI_FAILURE (Status))
835     {
836         AcpiOsPrintf ("AcpiRsCreateAmlResources failed: %s\n",
837             AcpiFormatException (Status));
838         goto Exit2;
839     }

841     /* Compare original AML to the newly created AML resource list */

843     OriginalAml = ReturnBuffer.Pointer;

845     AcpiDmCompareAmlResources (OriginalAml->Buffer.Pointer,
846         (ACPI_RSDESC_SIZE) OriginalAml->Buffer.Length,
847         NewAml.Pointer, (ACPI_RSDESC_SIZE) NewAml.Length);

849     /* Cleanup and exit */

```

```

851     ACPI_FREE (NewAml.Pointer);
852 Exit2:
853     ACPI_FREE (ResourceBuffer.Pointer);
854 Exit1:
855     ACPI_FREE (ReturnBuffer.Pointer);
856     return (Status);
857 }

860 /*****
861 *
862 * FUNCTION:      AcpiDbResourceCallback
863 *
864 * PARAMETERS:    ACPI_WALK_RESOURCE_CALLBACK
865 *
866 * RETURN:        Status
867 *
868 * DESCRIPTION:    Simple callback to exercise AcpiWalkResources and
869 *                 AcpiWalkResourceBuffer.
870 *
871 *****/

873 static ACPI_STATUS
874 AcpiDbResourceCallback (
875     ACPI_RESOURCE *Resource,
876     void *Context)
877 {
879     return (AE_OK);
880 }

883 /*****
884 *
885 * FUNCTION:      AcpiDbDeviceResources
886 *
887 * PARAMETERS:    ACPI_WALK_CALLBACK
888 *
889 * RETURN:        Status
890 *
891 * DESCRIPTION:    Display the _PRT/_CRS/_PRS resources for a device object.
892 *
893 *****/

895 static ACPI_STATUS
896 AcpiDbDeviceResources (
897     ACPI_HANDLE ObjHandle,
898     UINT32 NestingLevel,
899     void *Context,
900     void **ReturnValue)
901 {
902     ACPI_NAMESPACE_NODE *Node;
903     ACPI_NAMESPACE_NODE *PrtNode = NULL;
904     ACPI_NAMESPACE_NODE *CrsNode = NULL;
905     ACPI_NAMESPACE_NODE *PrsNode = NULL;
906     ACPI_NAMESPACE_NODE *AeiNode = NULL;
907     char *ParentPath;
908     ACPI_BUFFER ReturnBuffer;
909     ACPI_STATUS Status;

912     Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, ObjHandle);
913     ParentPath = AcpiNsGetExternalPathname (Node);
914     if (!ParentPath)
915     {
916         return (AE_NO_MEMORY);

```

```

917     }

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
1182 *                               0 or 1 for FADT GPE blocks
1183 *
1184 * RETURN:      None
1185 *
1186 * DESCRIPTION: Simulate firing of a GPE
1187 *
1188 *****/
1190 void
1191 AcpiDbGenerateGpe (
1192     char          *GpeArg,
1193     char          *BlockArg)
1194 {
1195     UINT32        BlockNumber;
1196     UINT32        GpeNumber;
1197     ACPI_GPE_EVENT_INFO *GpeEventInfo;
1198
1200     GpeNumber = ACPI_STRTOUL (GpeArg, NULL, 0);
1201     BlockNumber = ACPI_STRTOUL (BlockArg, NULL, 0);
1202
1203     GpeEventInfo = AcpiEvGetGpeEventInfo (ACPI_TO_POINTER (BlockNumber),
1204     GpeNumber);
1205     if (!GpeEventInfo)
1206     {
1207         AcpiOsPrintf ("Invalid GPE\n");
1208         return;
1209     }
1210
1211     (void) AcpiEvGpeDispatch (NULL, GpeEventInfo, GpeNumber);
1212 }
1213
1214 void
1215 AcpiDbGenerateSci (
1216     void)
1217 {
1218     AcpiEvSciDispatch ();
1219 }
1220
1221 #endif /* !ACPI_REduced_HARDWARE */
1222
1223 #endif /* ACPI_DEBUGGER */
```

new/usr/src/common/acpica/components/debugger/dbconvert.c

1

```
*****
15819 Fri Jan 17 07:29:47 2014
new/usr/src/common/acpica/components/debugger/dbconvert.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: dbconvert - debugger miscellaneous conversion routines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"

49 #ifdef ACPI_DEBUGGER

51 #define _COMPONENT          ACPI_CA_DEBUGGER
52 #define ACPI_MODULE_NAME    ("dbconvert")

55 #define DB_DEFAULT_PKG_ELEMENTS    33

58 /*****
59 *
60 * FUNCTION:      AcpiDbHexCharToValue
```

new/usr/src/common/acpica/components/debugger/dbconvert.c

2

```
61 *
62 * PARAMETERS:   HexChar          - Ascii Hex digit, 0-9|a-f|A-F
63 *              ReturnValue        - Where the converted value is returned
64 *
65 * RETURN:       Status
66 *
67 * DESCRIPTION:   Convert a single hex character to a 4-bit number (0-16).
68 *
69 *****/

71 ACPI_STATUS
72 AcpiDbHexCharToValue (
73     int             HexChar,
74     UINT8           *ReturnValue)
75 {
76     UINT8           Value;

79     /* Digit must be ascii [0-9a-fA-F] */

81     if (!ACPI_IS_XDIGIT (HexChar))
82     {
83         return (AE_BAD_HEX_CONSTANT);
84     }

86     if (HexChar <= 0x39)
87     {
88         Value = (UINT8) (HexChar - 0x30);
89     }
90     else
91     {
92         Value = (UINT8) (ACPI_TOUPPER (HexChar) - 0x37);
93     }

95     *ReturnValue = Value;
96     return (AE_OK);
97 }

100 /*****
101 *
102 * FUNCTION:      AcpiDbHexByteToBinary
103 *
104 * PARAMETERS:   HexByte          - Double hex digit (0x00 - 0xFF) in format:
105 *              HiByte then LoByte.
106 *              ReturnValue        - Where the converted value is returned
107 *
108 * RETURN:       Status
109 *
110 * DESCRIPTION:   Convert two hex characters to an 8 bit number (0 - 255).
111 *
112 *****/

114 static ACPI_STATUS
115 AcpiDbHexByteToBinary (
116     char           *HexByte,
117     UINT8          *ReturnValue)
118 {
119     UINT8          Local0;
120     UINT8          Local1;
121     ACPI_STATUS    Status;

124     /* High byte */

126     Status = AcpiDbHexCharToValue (HexByte[0], &Local0);
```

```

127     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  * FUNCTION:      AcpiDbConvertToBuffer
147  *
148  * PARAMETERS:    String          - Input string to be converted
149  *                Object          - Where the buffer object is returned
150  *
151  * RETURN:        Status
152  *
153  * DESCRIPTION:    Convert a string to a buffer object. String is treated a list
154  *                of buffer elements, each separated by a space or comma.
155  *
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  * FUNCTION:      AcpiDbConvertToPackage
220  *
221  * PARAMETERS:    String          - Input string to be converted
222  *                Object          - Where the package object is returned
223  *
224  * RETURN:        Status
225  *
226  * DESCRIPTION:    Convert a string to a package object. Handles nested packages
227  *                via recursion with AcpiDbConvertToObject.
228  *
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     }

267     This = Next;
268 }

270 Object->Type = ACPI_TYPE_PACKAGE;
271 Object->Package.Count = i;
272 Object->Package.Elements = Elements;
273 return (AE_OK);
274 }

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 *****/

295 ACPI_STATUS
296 AcpiDbConvertToObject (
297     ACPI_OBJECT_TYPE    Type,
298     char                *String,
299     ACPI_OBJECT          *Object)
300 {
301     ACPI_STATUS          Status = AE_OK;

304     switch (Type)
305     {
306     case ACPI_TYPE_STRING:

308         Object->Type = ACPI_TYPE_STRING;
309         Object->String.Pointer = String;
310         Object->String.Length = (UINT32) ACPI_STRLen (String);
311         break;

313     case ACPI_TYPE_BUFFER:

315         Status = AcpiDbConvertToBuffer (String, Object);
316         break;

318     case ACPI_TYPE_PACKAGE:

320         Status = AcpiDbConvertToPackage (String, Object);
321         break;

323     default:

```

```

325     Object->Type = ACPI_TYPE_INTEGER;
326     Status = AcpiUtStrtoul64 (String, 16, &Object->Integer.Value);
327     break;
328 }

330     return (Status);
331 }

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 *****/

346 UINT8 *
347 AcpiDbEncodePldBuffer (
348     ACPI_PLD_INFO        *PldInfo)
349 {
350     UINT32                *Buffer;
351     UINT32                Dword;

354     Buffer = ACPI_ALLOCATE_ZEROED (ACPI_PLD_BUFFER_SIZE);
355     if (!Buffer)
356     {
357         return (NULL);
358     }

360     /* First 32 bits */

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);

368     /* Second 32 bits */

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);

375     /* Third 32 bits */

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);

```

```

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 */
```

new/usr/src/common/acpica/components/debugger/dbdisply.c

1

```
*****
32871 Fri Jan 17 07:29:47 2014
new/usr/src/common/acpica/components/debugger/dbdisply.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dbdisply - debug display commands
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "amlcode.h"
48 #include "acdispat.h"
49 #include "acnamesp.h"
50 #include "acparser.h"
51 #include "acinterp.h"
52 #include "acdebug.h"
53 #include "acdisasm.h"

56 #ifdef ACPI_DEBUGGER

58 #define _COMPONENT ACPI_CA_DEBUGGER
```

new/usr/src/common/acpica/components/debugger/dbdisply.c

2

```
59 ACPI_MODULE_NAME ("dbdisply")

61 /* Local prototypes */

63 static void
64 AcpiDbDumpParserDescriptor (
65     ACPI_PARSE_OBJECT *Op);

67 static void *
68 AcpiDbGetPointer (
69     void *Target);

71 static ACPI_STATUS
72 AcpiDbDisplayNonRootHandlers (
73     ACPI_HANDLE ObjHandle,
74     UINT32 NestingLevel,
75     void *Context,
76     void **ReturnValue);

78 /*
79 * System handler information.
80 * Used for Handlers command, in AcpiDbDisplayHandlers.
81 */
82 #define ACPI_PREDEFINED_PREFIX "%25s (%.2X) : "
83 #define ACPI_HANDLER_NAME_STRING "%30s : "
84 #define ACPI_HANDLER_PRESENT_STRING "%-9s (%p)\n"
85 #define ACPI_HANDLER_PRESENT_STRING2 "%-9s (%p)"
86 #define ACPI_HANDLER_NOT_PRESENT_STRING "%-9s\n"

88 /* All predefined Address Space IDs */

90 static ACPI_ADR_SPACE_TYPE AcpiGbl_SpaceIdList[] =
91 {
92     ACPI_ADR_SPACE_SYSTEM_MEMORY,
93     ACPI_ADR_SPACE_SYSTEM_IO,
94     ACPI_ADR_SPACE_PCI_CONFIG,
95     ACPI_ADR_SPACE_EC,
96     ACPI_ADR_SPACE_SMBUS,
97     ACPI_ADR_SPACE_CMOS,
98     ACPI_ADR_SPACE_PCI_BAR_TARGET,
99     ACPI_ADR_SPACE_IPMI,
100     ACPI_ADR_SPACE_GPIO,
101     ACPI_ADR_SPACE_GSBUS,
102     ACPI_ADR_SPACE_DATA_TABLE,
103     ACPI_ADR_SPACE_FIXED_HARDWARE
104 };
105 #ifndef _unchanged_portion_omitted_
106
115 static ACPI_HANDLER_INFO AcpiGbl_HandlerList[] =
116 {
117     {&AcpiGbl_GlobalNotify[0].Handler, "System Notifications"},
118     {&AcpiGbl_GlobalNotify[1].Handler, "Device Notifications"},
119     {&AcpiGbl_SystemNotify.Handler, "System Notifications"},
120     {&AcpiGbl_DeviceNotify.Handler, "Device Notifications"},
121     {&AcpiGbl_TableHandler, "ACPI Table Events"},
122     {&AcpiGbl_ExceptionHandler, "Control Method Exceptions"},
123     {&AcpiGbl_InterfaceHandler, "OSI Invocations"}
124 };

125 /*****
126 *
127 * FUNCTION: AcpiDbGetPointer
128 *
129 * PARAMETERS: Target - Pointer to string to be converted
130 */
```

```

131 * RETURN:      Converted pointer
132 *
133 * DESCRIPTION: Convert an ascii pointer value to a real value
134 *
135 *****/
136
137 static void *
138 AcpiDbGetPointer (
139     void                *Target)
140 {
141     void                *ObjPtr;
142     ACPI_SIZE           Address;
143
144     Address = ACPI_STRTOL (Target, NULL, 16);
145     ObjPtr = ACPI_TO_POINTER (Address);
146     ObjPtr = ACPI_TO_POINTER (ACPI_STRTOL (Target, NULL, 16));
147     return (ObjPtr);
148 }
149
150 unchanged_portion_omitted
151
152
153
154 /*****
155 *
156 * FUNCTION:      AcpiDbDecodeAndDisplayObject
157 *
158 * PARAMETERS:    Target          - String with object to be displayed. Names
159 *                  and hex pointers are supported.
160 *                  OutputType     - Byte, Word, Dword, or Qword (B|W|D|Q)
161 *
162 * RETURN:        None
163 *
164 * DESCRIPTION:    Display a formatted ACPI object
165 *
166 *****/
167
168 void
169 AcpiDbDecodeAndDisplayObject (
170     char                *Target,
171     char                *OutputType)
172 {
173     void                *ObjPtr;
174     ACPI_NAMESPACE_NODE *Node;
175     ACPI_OPERAND_OBJECT *ObjDesc;
176     UINT32              Display = DB_BYTE_DISPLAY;
177     char                Buffer[80];
178     ACPI_BUFFER          RetBuf;
179     ACPI_STATUS          Status;
180     UINT32              Size;
181
182     if (!Target)
183     {
184         return;
185     }
186
187     /* Decode the output type */
188
189     if (OutputType)
190     {
191         AcpiUtStrupr (OutputType);
192         if (OutputType[0] == 'W')
193         {
194             Display = DB_WORD_DISPLAY;
195         }
196         else if (OutputType[0] == 'D')
197         {
198             Display = DB_DWORD_DISPLAY;
199         }
200         else if (OutputType[0] == 'Q')
201         {
202             Display = DB_QWORD_DISPLAY;
203         }
204     }
205
206     ObjPtr = AcpiDbGetPointer (Target);
207     if (!ObjPtr)
208     {
209         return;
210     }
211
212     Node = AcpiNsGetNode (ObjPtr);
213     if (!Node)
214     {
215         return;
216     }
217
218     ObjDesc = AcpiNsGetObjDesc (Node);
219     if (!ObjDesc)
220     {
221         return;
222     }
223
224     Status = AcpiOsRead (ObjPtr, &RetBuf, sizeof (RetBuf));
225     if (ACPI_FAILURE (Status))
226     {
227         return;
228     }
229
230     Size = RetBuf.Length;
231     if (Size > 0)
232     {
233         AcpiOsPrint (ACPI_OUTPUT_FILE, "Object at address %p is %s\n",
234             ObjPtr, Display);
235         AcpiOsPrint (ACPI_OUTPUT_FILE, "Size: %d\n", Size);
236         AcpiOsPrint (ACPI_OUTPUT_FILE, "Data: ");
237         AcpiOsPrint (ACPI_OUTPUT_FILE, RetBuf.Pointer);
238         AcpiOsPrint (ACPI_OUTPUT_FILE, "\n");
239     }
240 }

```

```

228     {
229         Display = DB_DWORD_DISPLAY;
230     }
231     else if (OutputType[0] == 'Q')
232     {
233         Display = DB_QWORD_DISPLAY;
234     }
235 }
236
237 RetBuf.Length = sizeof (Buffer);
238 RetBuf.Pointer = Buffer;
239
240 /* Differentiate between a number and a name */
241
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     }
251 }
252
253 /* Decode the object type */
254
255 switch (ACPI_GET_DESCRIPTOR_TYPE (ObjPtr))
256 {
257     case ACPI_DESC_TYPE_NAMED:
258
259         /* This is a namespace Node */
260
261         if (!AcpiOsReadable (ObjPtr, sizeof (ACPI_NAMESPACE_NODE)))
262         {
263             AcpiOsPrintf (
264                 "Cannot read entire Named object at address %p\n", ObjPtr);
265             return;
266         }
267
268         Node = ObjPtr;
269         goto DumpNode;
270
271     case ACPI_DESC_TYPE_OPERAND:
272
273         /* This is a ACPI OPERAND OBJECT */
274
275         if (!AcpiOsReadable (ObjPtr, sizeof (ACPI_OPERAND_OBJECT)))
276         {
277             AcpiOsPrintf ("Cannot read entire ACPI object at address %p\n",
278                 ObjPtr);
279             return;
280         }
281
282         AcpiUtDebugDumpBuffer (ObjPtr, sizeof (ACPI_OPERAND_OBJECT), Display,
283             ACPI_UINT32_MAX);
284         AcpiExDumpObjectDescriptor (ObjPtr, 1);
285         break;
286
287     case ACPI_DESC_TYPE_PARSER:
288
289         /* This is a Parser Op object */
290
291         if (!AcpiOsReadable (ObjPtr, sizeof (ACPI_PARSE_OBJECT)))
292         {
293             AcpiOsPrintf ("Cannot read entire Parser object at address %p\n",
294                 ObjPtr);
295             return;
296         }
297
298         AcpiExDumpObjectDescriptor (ObjPtr, 1);
299         break;
300 }

```

```

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,
298         AcpiUtDumpBuffer (ObjPtr, sizeof (ACPI_PARSE_OBJECT), Display,
299             ACPI_UINT32_MAX);
300     AcpiDbDumpParserDescriptor ((ACPI_PARSE_OBJECT *) ObjPtr);
301     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);
315         AcpiUtDumpBuffer (ObjPtr, Size, Display, ACPI_UINT32_MAX);
316         break;

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),
352     AcpiUtDumpBuffer ((void *) Node, sizeof (ACPI_NAMESPACE_NODE),
353         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),
367             AcpiUtDumpBuffer ((void *) ObjDesc, sizeof (ACPI_OPERAND_OBJECT),
368                 Display, ACPI_UINT32_MAX);
369         AcpiExDumpObjectDescriptor (ObjDesc, 1);
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                 NumOperators++;
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
656  *                ObjectArg     - User entered NS node handle
657  *
658  * RETURN:        None
659  *
660  * DESCRIPTION:   Display type of an arbitrary NS node
661  *****/

663 void
664 AcpiDbDisplayObjectType (
665     char          *Name)
666 {
667     char          *ObjectArg)
668 {
669     ACPI_NAMESPACE_NODE *Node;
670     ACPI_HANDLE Handle;
671     ACPI_DEVICE_INFO *Info;
672     ACPI_STATUS Status;
673     UINT32 i;

674     Node = AcpiDbConvertToNode (Name);
675     if (!Node)
676     {
677         return;
678     }
679     Handle = ACPI_TO_POINTER (ACPI_STRTOL (ObjectArg, NULL, 16));

680     Status = AcpiGetObjectInfo (ACPI_CAST_PTR (ACPI_HANDLE, Node), &Info);
681     if (ACPI_FAILURE (Status))
682     {
683         AcpiOsPrintf ("Could not get object info, %s\n",
684             AcpiFormatException (Status));
685         return;
686     }

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)
708     {
709         AcpiOsPrintf ("HID: %s\n", Info->HardwareId.String);
710     }
711     if (Info->Valid & ACPI_VALID_UID)
712     {
713         AcpiOsPrintf ("UID: %s\n", Info->UniqueId.String);
714     }
715     if (Info->Valid & ACPI_VALID_SUB)
716     {
717         AcpiOsPrintf ("SUB: %s\n", Info->SubsystemId.String);
718     }
719     if (Info->Valid & ACPI_VALID_CID)
720     {
721         for (i = 0; i < Info->CompatibleIdList.Count; i++)
722         {
723             AcpiOsPrintf ("CID %u: %s\n", i,
724                 Info->CompatibleIdList.Ids[i].String);
725         }
726     }
728     ACPI_FREE (Info);
729 }

```

unchanged\_portion\_omitted

```

797 #if (!ACPI_REDUCED_HARDWARE)
798 /*****
799  *
800  * FUNCTION:      AcpiDbDisplayGpes
801  *
802  * PARAMETERS:    None
803  *
804  * RETURN:        None
805  *
806  * DESCRIPTION:    Display the current GPE structures
807  *
808  *****/

```

```

810 void
811 AcpiDbDisplayGpes (
812     void)
813 {
814     ACPI_GPE_BLOCK_INFO      *GpeBlock;
815     ACPI_GPE_XRPT_INFO       *GpeXruptInfo;
816     ACPI_GPE_EVENT_INFO      *GpeEventInfo;
817     ACPI_GPE_REGISTER_INFO    *GpeRegisterInfo;
818     char                      *GpeType;
819     ACPI_GPE_NOTIFY_INFO      *Notify;
820     UINT32                    GpeIndex;
821     UINT32                    Block = 0;
822     UINT32                    i;
823     UINT32                    j;
824     UINT32                    Count;
825     char                      Buffer[80];
826     ACPI_BUFFER               RetBuf;
827     ACPI_STATUS               Status;
828
830     RetBuf.Length = sizeof (Buffer);
831     RetBuf.Pointer = Buffer;
832
833     Block = 0;
834
835     /* Walk the GPE lists */

```

```

837     GpeXruptInfo = AcpiGbl_GpeXruptListHead;
838     while (GpeXruptInfo)
839     {
840         GpeBlock = GpeXruptInfo->GpeBlockListHead;
841         while (GpeBlock)
842         {
843             Status = AcpiGetName (GpeBlock->Node, ACPI_FULL_PATHNAME, &RetBuf);
844             if (ACPI_FAILURE (Status))
845             {
846                 AcpiOsPrintf ("Could not convert name to pathname\n");
847             }
848
849             if (GpeBlock->Node == AcpiGbl_FadtGpeDevice)
850             {
851                 GpeType = "FADT-defined GPE block";
852             }
853             else
854             {
855                 GpeType = "GPE Block Device";
856             }
857
858             AcpiOsPrintf ("\nBlock %u - Info %p DeviceNode %p [%s] - %s\n",
859                 Block, GpeBlock, GpeBlock->Node, Buffer, GpeType);
860
861             AcpiOsPrintf ("    Registers:      %u (%u GPEs)\n",
862                 GpeBlock->RegisterCount, GpeBlock->GpeCount);
863
864             AcpiOsPrintf ("    GPE range:      0x%X to 0x%X on interrupt %u\n",
865                 GpeBlock->BlockBaseNumber,
866                 GpeBlock->BlockBaseNumber + (GpeBlock->GpeCount - 1),
867                 GpeXruptInfo->InterruptNumber);
868
869             AcpiOsPrintf (
870                 "    RegisterInfo: %p Status %8.8X%8.8X Enable %8.8X%8.8X\n",
871                 GpeBlock->RegisterInfo,
872                 ACPI_FORMAT_UINT64 (GpeBlock->RegisterInfo->StatusAddress.Address),
873                 ACPI_FORMAT_UINT64 (GpeBlock->RegisterInfo->EnableAddress.Address)
874             );
875             AcpiOsPrintf ("    EventInfo:      %p\n", GpeBlock->EventInfo);
876
877             /* Examine each GPE Register within the block */
878
879             for (i = 0; i < GpeBlock->RegisterCount; i++)
880             {
881                 GpeRegisterInfo = &GpeBlock->RegisterInfo[i];
882
883                 AcpiOsPrintf (
884                     "    Reg %u: (GPE %2X-%2X) RunEnable %2.2X WakeEnable %2.2X\n",
885                     i, GpeRegisterInfo->BaseGpeNumber,
886                     GpeRegisterInfo->BaseGpeNumber + (ACPI_GPE_REGISTER_WIDTH -
887                         GpeRegisterInfo->EnableForRun,
888                         GpeRegisterInfo->EnableForWake,
889                         ACPI_FORMAT_UINT64 (GpeRegisterInfo->StatusAddress.Address),
890                         ACPI_FORMAT_UINT64 (GpeRegisterInfo->EnableAddress.Address))
891                 );
892
893                 /* Now look at the individual GPEs in this byte register */
894
895                 for (j = 0; j < ACPI_GPE_REGISTER_WIDTH; j++)
896                 {
897                     GpeIndex = (i * ACPI_GPE_REGISTER_WIDTH) + j;
898                     GpeEventInfo = &GpeBlock->EventInfo[GpeIndex];
899
900                     if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) ==
901                         ACPI_GPE_DISPATCH_NONE)

```



```

902     {
903         /* This GPE is not used (no method or handler), ignore it
904
905         continue;
906     }
907
908     AcpiOsPrintf (
909         "    GPE %.2X: %p RunRefs %.2X Flags %.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:
950
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
962     default:
963
964         AcpiOsPrintf ("UNKNOWN: %X",
965             GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK);
966         break;
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 * FUNCTION:      AcpiDbDisplayHandlers
982 * PARAMETERS:    None
983 * RETURN:        None
984 * DESCRIPTION:   Display the currently installed global handlers
985 *
986 *****/
987 void
988 AcpiDbDisplayHandlers (
989     void)
990 {
991     ACPI_OPERAND_OBJECT *ObjDesc;
992     ACPI_OPERAND_OBJECT *HandlerObj;
993     ACPI_ADR_SPACE_TYPE SpaceId;
994     UINT32 i;
995
996     /* Operation region handlers */
997
998     AcpiOsPrintf ("\nOperation Region Handlers at the namespace root:\n");
999     AcpiOsPrintf ("Operation Region Handlers:\n");
1000
1001     ObjDesc = AcpiNsGetAttachedObject (AcpiGbl_RootNode);
1002     if (ObjDesc)
1003     {
1004         for (i = 0; i < ACPI_ARRAY_LENGTH (AcpiGbl_SpaceIdList); i++)
1005         {
1006             SpaceId = AcpiGbl_SpaceIdList[i];
1007             HandlerObj = ObjDesc->Device.Handler;
1008
1009             AcpiOsPrintf (ACPI_PREDEFINED_PREFIX,
1010                 AcpiUtGetRegionName ((UINT8) SpaceId), SpaceId);
1011
1012             while (HandlerObj)
1013             {
1014                 if (AcpiGbl_SpaceIdList[i] == HandlerObj->AddressSpace.SpaceId)
1015                 {
1016                     if (i == HandlerObj->AddressSpace.SpaceId)
1017                     {
1018                         AcpiOsPrintf (ACPI_HANDLER_PRESENT_STRING,
1019                             (HandlerObj->AddressSpace.HandlerFlags &
1020                                 ACPI_ADDR_HANDLER_DEFAULT_INSTALLED) ? "Default" : "
1021                             HandlerObj->AddressSpace.Handler);
1022                         goto FoundHandler;
1023                     }
1024                     HandlerObj = HandlerObj->AddressSpace.Next;
1025                 }
1026             }
1027         }
1028     }
1029 }

```

```

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      * FUNCTION:      AcpiDbDisplayNonRootHandlers
1110      * PARAMETERS:    ACPI_WALK_CALLBACK
1111      * RETURN:        Status
1112      * DESCRIPTION:   Display information about all handlers installed for a
1113                      device object.
1114      *****/
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, ObjHandle);
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 */
```

new/usr/src/common/acpica/components/debugger/dbexec.c

1

```
*****
23658 Fri Jan 17 07:29:48 2014
new/usr/src/common/acpica/components/debugger/dbexec.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dbexec - debugger control method execution
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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    ("dbexec")

56 static ACPI_DB_METHOD_INFO      AcpiGbl_DbMethodInfo;

58 /* Local prototypes */
```

new/usr/src/common/acpica/components/debugger/dbexec.c

2

```
60 static ACPI_STATUS
61 AcpiDbExecuteMethod (
62     ACPI_DB_METHOD_INFO      *Info,
63     ACPI_BUFFER               *ReturnObj);

65 static ACPI_STATUS
66 AcpiDbExecuteSetup (
67     ACPI_DB_METHOD_INFO      *Info);

69 static UINT32
70 AcpiDbGetOutstandingAllocations (
71     void);

73 static void ACPI_SYSTEM_XFACE
74 AcpiDbMethodThread (
75     void                      *Context);

77 static ACPI_STATUS
78 AcpiDbExecutionWalk (
79     ACPI_HANDLE               ObjHandle,
80     UINT32                    NestingLevel,
81     void                      *Context,
82     void                      **ReturnValue);

85 /*****
86 *
87 * FUNCTION:      AcpiDbDeleteObjects
88 *
89 * PARAMETERS:    Count          - Count of objects in the list
90 *                Objects        - Array of ACPI_OBJECTS to be deleted
91 *
92 * RETURN:        None
93 *
94 * DESCRIPTION:   Delete a list of ACPI_OBJECTS. Handles packages and nested
95 *                packages via recursion.
96 *
97 *****/

99 void
100 AcpiDbDeleteObjects (
101     UINT32                Count,
102     ACPI_OBJECT            *Objects)
103 {
104     UINT32                i;

107     for (i = 0; i < Count; i++)
108     {
109         switch (Objects[i].Type)
110         {
111             case ACPI_TYPE_BUFFER:

113                 ACPI_FREE (Objects[i].Buffer.Pointer);
114                 break;

116             case ACPI_TYPE_PACKAGE:

118                 /* Recursive call to delete package elements */

120                 AcpiDbDeleteObjects (Objects[i].Package.Count,
121                                     Objects[i].Package.Elements);

123                 /* Free the elements array */
```

```

125         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     ParamObjects;
156     ACPI_OBJECT_LIST        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 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\n",
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  *****/
542
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         }
599     }

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;
669
671     /* Get the arguments */
672
673     NumThreads = ACPI_STRTOUL (NumThreadsArg, NULL, 0);
674     NumLoops   = ACPI_STRTOUL (NumLoopsArg, NULL, 0);
675
676     if (!NumThreads || !NumLoops)
677     {
678         AcpiOsPrintf ("Bad argument: Threads %X, Loops %X\n",
679             NumThreads, NumLoops);
680         return;
681     }
682
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     }
694
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     }
707
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     }
717
718     ACPI_MEMSET (&AcpiGbl_DbMethodInfo, 0, sizeof (ACPI_DB_METHOD_INFO));

```

```

720     /* Array to store IDs of threads */
721
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);
734
735     /* Setup the context to be passed to each thread */
736
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;
743
744     /* Init arguments to be passed to method */
745
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;
752
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;
757
758     AcpiDbUint32ToHexString (NumThreads, AcpiGbl_DbMethodInfo.NumThreadsStr);
759
760     Status = AcpiDbExecuteSetup (&AcpiGbl_DbMethodInfo);
761     if (ACPI_FAILURE (Status))
762     {
763         goto CleanupAndExit;
764     }
765
766     /* Get the NS node, determines existence also */
767
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     }
776
777     /* Create the threads */
778
779     AcpiOsPrintf ("Creating %X threads to execute %X times each\n",
780         NumThreads, NumLoops);
781
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 */
```

new/usr/src/common/acpica/components/debugger/dbfileio.c

1

```
*****
16106 Fri Jan 17 07:29:48 2014
new/usr/src/common/acpica/components/debugger/dbfileio.c
update to acpica-unix2-20140114
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 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
47 #include "accommon.h"
48 #include "acdebug.h"

50 #ifdef ACPI_APPLICATION
51 #include "actables.h"
52 #endif

54 #ifdef ACPI_AS_L_COMPILER
55 #include "aslcompiler.h"
56 #endif
```

new/usr/src/common/acpica/components/debugger/dbfileio.c

2

```
58 #if (defined ACPI_DEBUGGER || defined ACPI_DISASSEMBLER)

60 #define _COMPONENT          ACPI_CA_DEBUGGER
61 #define 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 {
123
124 #ifdef ACPI_APPLICATION
125
126     AcpiDbCloseDebugFile ();
127     AcpiGbl_DebugFile = fopen (Name, "w+");
128     if (!AcpiGbl_DebugFile)
129     {
130         AcpiOsPrintf ("Debug output file %s opened\n", Name);
131         ACPI_STRCPY (AcpiGbl_DbDebugFilename, Name);
132         AcpiGbl_DbOutputToFile = TRUE;
133     }
134     else
135     {
136         AcpiOsPrintf ("Could not open debug file %s\n", Name);
137         return;
138     }
139
140     AcpiOsPrintf ("Debug output file %s opened\n", Name);
141     ACPI_STRNCPY (AcpiGbl_DbDebugFilename, Name,
142                 sizeof (AcpiGbl_DbDebugFilename));
143     AcpiGbl_DbOutputToFile = TRUE;
144
145 #endif
146 }
147
148 _____
149
150 /*****
151 *
152 * FUNCTION:      AcpiDbReadTable
153 *
154 * PARAMETERS:    fp          - File that contains table
155 *                Table       - Return value, buffer with table
156 *                TableLength  - Return value, length of table
157 *
158 * RETURN:        Status
159 *
160 * DESCRIPTION:   Load the DSDT from the file pointer
161 *
162 *****/
163
164 static ACPI_STATUS
165 AcpiDbReadTable (
166     FILE            *fp,
167     ACPI_TABLE_HEADER **Table,
168     UINT32          *TableLength)
169 {
170     ACPI_TABLE_HEADER TableHeader;
171     UINT32 Actual;
172     ACPI_STATUS Status;
173     UINT32 FileSize;
174     BOOLEAN StandardHeader = TRUE;
175
176     /* Get the file size */

```

```

249     fseek (fp, 0, SEEK_END);
250     FileSize = (UINT32) ftell (fp);
251     fseek (fp, 0, SEEK_SET);
252
253     if (FileSize < 4)
254     {
255         return (AE_BAD_HEADER);
256     }
257
258     /* Read the signature */
259
260     if (fread (&TableHeader, 1, 4, fp) != 4)
261     {
262         AcpiOsPrintf ("Could not read the table signature\n");
263         return (AE_BAD_HEADER);
264     }
265
266     fseek (fp, 0, SEEK_SET);
267
268     /* The RSDP table does not have standard ACPI header */
269     /* The RSDT and FACS tables do not have standard ACPI headers */
270
271     if (ACPI_COMPARE_NAME (TableHeader.Signature, "RSD ") ||
272         if (ACPI_COMPARE_NAME (TableHeader.Signature, "RSD ") ||
273             ACPI_COMPARE_NAME (TableHeader.Signature, "FACS"))
274     {
275         *TableLength = FileSize;
276         StandardHeader = FALSE;
277     }
278     else
279     {
280         /* Read the table header */
281
282         if (fread (&TableHeader, 1, sizeof (ACPI_TABLE_HEADER), fp) !=
283             sizeof (ACPI_TABLE_HEADER))
284         {
285             AcpiOsPrintf ("Could not read the table header\n");
286             return (AE_BAD_HEADER);
287         }
288
289         #if 0
290         /* Validate the table header/length */
291
292         Status = AcpiTbValidateTableHeader (&TableHeader);
293         if (ACPI_FAILURE (Status))
294         {
295             AcpiOsPrintf ("Table header is invalid!\n");
296             return (Status);
297         }
298     #endif
299
300     /* File size must be at least as long as the Header-specified length */
301
302     if (TableHeader.Length > FileSize)
303     {
304         AcpiOsPrintf (
305             "TableHeader length [0x%X] greater than the input file size [0x%
306             TableHeader.Length, FileSize);
307
308     #ifdef ACPI_ASX_COMPILER
309     Status = FlCheckForAscii (fp, NULL, FALSE);
310     if (ACPI_SUCCESS (Status))
311     {
312         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))
322     if (ACPI_STRNCMP ((char *) TableHeader.Signature, DSDT_SIG, 4) &&
323         ACPI_STRNCMP ((char *) TableHeader.Signature, PSDT_SIG, 4) &&
324         ACPI_STRNCMP ((char *) TableHeader.Signature, SSDT_SIG, 4))
325     {
326         AcpiOsPrintf ("Table signature [%4.4s] is invalid or not supported\n",
327             (char *) TableHeader.Signature);
328         ACPI_DUMP_BUFFER (&TableHeader, sizeof (ACPI_TABLE_HEADER));
329         return (AE_ERROR);
330     }
331 #endif

333     *TableLength = TableHeader.Length;

335     /* Allocate a buffer for the table */

337     *Table = AcpiOsAllocate ((size_t) FileSize);
338     if (!*Table)
339     {
340         AcpiOsPrintf (
341             "Could not allocate memory for ACPI table %4.4s (size=0x%X)\n",
342             TableHeader.Signature, *TableLength);
343         return (AE_NO_MEMORY);
344     }

346     /* Get the rest of the table */

348     fseek (fp, 0, SEEK_SET);
349     Actual = fread (*Table, 1, (size_t) FileSize, fp);
350     if (Actual == FileSize)
351     {
352         if (StandardHeader)
353         {
354             /* Now validate the checksum */

356             Status = AcpiTbVerifyChecksum ((void *) *Table,
357                 ACPI_CAST_PTR (ACPI_TABLE_HEADER, *Table)->Length);

359             if (Status == AE_BAD_CHECKSUM)
360             {
361                 Status = AcpiDbCheckTextModeCorruption ((UINT8 *) *Table,
362                     FileSize, (*Table)->Length);
363                 return (Status);
364             }
365         }
366         return (AE_OK);
367     }

369     if (Actual > 0)
370     {
371         AcpiOsPrintf ("Warning - reading table, asked for %X got %X\n",
372             FileSize, Actual);
373         return (AE_OK);
374     }

```

```

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;
476     UINT32              FileSize;
477     FILE                *fp;
478     UINT32              TableLength;
479     ACPI_STATUS          Status;

481     /* Open the file */

483     File = fopen (Filename, "rb");
484     if (!File)
485     {
486         fp = fopen (Filename, "rb");
487         if (!fp)
488         {
489             perror ("Could not open input file");
490             AcpiOsPrintf ("Could not open input file %s\n", Filename);
491             return (AE_ERROR);
492         }
493     }

495     /* Get the file size */

497     fseek (File, 0, SEEK_END);
498     FileSize = (UINT32) ftell (File);
499     fseek (File, 0, SEEK_SET);

501     /* Get the entire file */

503     fprintf (stderr, "Loading Acpi table from file %10s - Length %.8u (%06X)\n",
504         Filename, FileSize, FileSize);
505     fprintf (stderr, "Loading Acpi table from file %s\n", Filename);
506     Status = AcpiDbReadTable (fp, Table, &TableLength);
507     fclose (fp);

509     Status = AcpiDbReadTable (File, Table, &TableLength);
510     fclose (File);

512     if (ACPI_FAILURE (Status))

```

```

505     {
506         AcpiOsPrintf ("Could not get table from the file\n");
507         return (Status);
508     }

510     return (AE_OK);
511 }
512 #endif

515 /*****
516 *
517 * FUNCTION:      AcpiDbGetTableFromFile
518 *
519 * PARAMETERS:    Filename          - File where table is located
520 *                ReturnTable       - Where a pointer to the table is returned
521 *
522 * RETURN:        Status
523 *
524 * DESCRIPTION:   Load an ACPI table from a file
525 *
526 *****/

528 ACPI_STATUS
529 AcpiDbGetTableFromFile (
530     char                *Filename,
531     ACPI_TABLE_HEADER **ReturnTable)
532 {
533 #ifdef ACPI_APPLICATION
534     ACPI_STATUS          Status;
535     ACPI_TABLE_HEADER    *Table;
536     BOOLEAN              IsAmlTable = TRUE;

539     Status = AcpiDbReadTableFromFile (Filename, &Table);
540     if (ACPI_FAILURE (Status))
541     {
542         return (Status);
543     }

545 #ifdef ACPI_DATA_TABLE_DISASSEMBLY
546     IsAmlTable = AcpiUtIsAmlTable (Table);
547 #endif

549     if (IsAmlTable)
550     {
551         /* Attempt to recognize and install the table */

553         Status = AeLocalLoadTable (Table);
554         if (ACPI_FAILURE (Status))
555         {
556             if (Status == AE_ALREADY_EXISTS)
557             {
558                 AcpiOsPrintf ("Table %4.4s is already installed\n",
559                     Table->Signature);
560             }
561             else
562             {
563                 AcpiOsPrintf ("Could not install table, %s\n",
564                     AcpiFormatException (Status));
565             }

567             return (Status);
568         }

570         fprintf (stderr,

```

```

571         "Acpi table [%4.4s] successfully installed and loaded\n",
572         Table->Signature);
573     }

575     AcpiGbl_AcpiHardwarePresent = FALSE;
576     if (ReturnTable)
577     {
578         *ReturnTable = Table;
579     }

582 #endif /* ACPI_APPLICATION */
583     return (AE_OK);
584 }

586 #endif /* ACPI_DEBUGGER */

```

new/usr/src/common/acpica/components/debugger/dbhistry.c

1

```
*****
7672 Fri Jan 17 07:29:48 2014
new/usr/src/common/acpica/components/debugger/dbhistry.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dbhistry - debugger HISTORY command
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "acpi.h"
45 #include "accommon.h"
46 #include "acdebug.h"
47
48 #ifdef ACPI_DEBUGGER
49
50 #define _COMPONENT          ACPI_CA_DEBUGGER
51 #define ACPI_MODULE_NAME    ("dbhistry")
52
53
54 #define HI_NO_HISTORY      0
55 #define HI_RECORD_HISTORY 1
56 #define HISTORY_SIZE      40
```

new/usr/src/common/acpica/components/debugger/dbhistry.c

2

```
60 typedef struct HistoryInfo
61 {
62     char          *Command;
63     UINT32         CmdNum;
64
65 } HISTORY_INFO;
66
67 static HISTORY_INFO  AcpiGbl_HistoryBuffer[HISTORY_SIZE];
68 static UINT16        AcpiGbl_LoHistory = 0;
69 static UINT16        AcpiGbl_NumHistory = 0;
70 static UINT16        AcpiGbl_NextHistoryIndex = 0;
71 static UINT16        AcpiGbl_NextCmdNum = 1;
72
73
74 /*****
75 *
76 * FUNCTION:      AcpiDbAddToHistory
77 *
78 * PARAMETERS:    CommandLine      - Command to add
79 *
80 * RETURN:        None
81 *
82 * DESCRIPTION:   Add a command line to the history buffer.
83 *
84 *****/
85
86 void
87 AcpiDbAddToHistory (
88     char          *CommandLine)
89 {
90     UINT16        CmdLen;
91     UINT16        BufferLen;
92
93     /* Put command into the next available slot */
94
95     CmdLen = (UINT16) ACPI_STRLEN (CommandLine);
96     if (!CmdLen)
97     {
98         return;
99     }
100
101     if (AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].Command != NULL)
102     {
103         BufferLen = (UINT16) ACPI_STRLEN (
104             AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].Command);
105         if (CmdLen > BufferLen)
106         {
107             AcpiOsFree (AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].
108                 Command);
109             AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].Command =
110                 AcpiOsAllocate (CmdLen + 1);
111         }
112     }
113     else
114     {
115         AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].Command =
116             AcpiOsAllocate (CmdLen + 1);
117     }
118
119     ACPI_STRCPY (AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].Command,
120         CommandLine);
121
122     AcpiGbl_HistoryBuffer[AcpiGbl_NextHistoryIndex].CmdNum =
123         AcpiGbl_NextCmdNum;
124 }
```

```

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 */
```

new/usr/src/common/acpica/components/debugger/dbinput.c

1

```
*****
35254 Fri Jan 17 07:29:48 2014
new/usr/src/common/acpica/components/debugger/dbinput.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dbinput - user front-end to the AML debugger
4 *
5 *****/
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"

50 #ifdef ACPI_DEBUGGER

52 #define _COMPONENT          ACPI_CA_DEBUGGER
53       ACPI_MODULE_NAME    ("dbinput")

55 /* Local prototypes */

57 static UINT32
```

new/usr/src/common/acpica/components/debugger/dbinput.c

2

```
58 AcpiDbGetLine (
59     char                                *InputBuffer);

61 static UINT32
62 AcpiDbMatchCommand (
63     char                                *UserCommand);

65 static void
66 AcpiDbSingleThread (
67     void);

69 static void
70 AcpiDbDisplayCommandInfo (
71     char                                *Command,
72     BOOLEAN                             DisplayAll);

74 static void
75 AcpiDbDisplayHelp (
76     char                                *Command);

78 static BOOLEAN
79 AcpiDbMatchCommandHelp (
80     char                                *Command,
81     const ACPI_DB_COMMAND_HELP         *Help);

84 /*
85 * Top-level debugger commands.
86 *
87 * This list of commands must match the string table below it
88 */
89 enum AcpiExDebuggerCommands
90 {
91     CMD_NOT_FOUND = 0,
92     CMD_NULL,
93     CMD_ALLOCATIONS,
94     CMD_ARGS,
95     CMD_ARGUMENTS,
96     CMD_BREAKPOINT,
97     CMD_BUSINFO,
98     CMD_CALL,
99     CMD_CLOSE,
100     CMD_DEBUG,
101     CMD_DISASSEMBLE,
102     CMD_DISASM,
103     CMD_DUMP,
104     CMD_ENABLEACPI,
105     CMD_EVALUATE,
106     CMD_EVENT,
107     CMD_EXECUTE,
108     CMD_EXIT,
109     CMD_FIND,
110     CMD_GO,
111     CMD_GPE,
112     CMD_GPES,
113     CMD_HANDLERS,
114     CMD_HELP,
115     CMD_HELP2,
116     CMD_HISTORY,
117     CMD_HISTORY_EXE,
118     CMD_HISTORY_LAST,
119     CMD_INFORMATION,
120     CMD_INTEGRITY,
121     CMD_INT0,
122     CMD_LEVEL,
123     CMD_LIST,
```

```

124  CMD_LOAD,
125  CMD_LOCALS,
126  CMD_LOCKS,
127  CMD_METHODS,
128  CMD_NAMESPACE,
129  CMD_NOTIFY,
130  CMD_OBJECTS,
131  CMD_OPEN,
132  CMD_OSI,
133  CMD_OWNER,
134  CMD_PATHS,
135  CMD_PREDEFINED,
136  CMD_PREFIX,
137  CMD_QUIT,
138  CMD_REFERENCES,
139  CMD_RESOURCES,
140  CMD_RESULTS,
141  CMD_SCI,
142  CMD_SET,
143  CMD_SLEEP,
144  CMD_STATS,
145  CMD_STOP,
146  CMD_TABLES,
147  CMD_TEMPLATE,
148  CMD_TERMINATE,
149  CMD_TEST,
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     {"BREAKPOINT", 1},
170     {"BUSINFO", 0},
171     {"CALL", 0},
172     {"CLOSE", 0},
173     {"DEBUG", 1},
174     {"DISASSEMBLE", 1},
175     {"DISASM", 1},
176     {"DUMP", 1},
177     {"ENABLEACPI", 0},
178     {"EVALUATE", 1},
179     {"EVENT", 1},
180     {"EXECUTE", 1},
181     {"EXIT", 0},
182     {"FIND", 1},
183     {"GO", 0},
184     {"GPE", 2},
185     {"GPES", 0},
186     {"HANDLERS", 0},
187     {"HELP", 0},
188     {"?", 0},
189     {"HISTORY", 0},

```

```

190     {"!", 1},
191     {"!!", 0},
192     {"INFORMATION", 0},
193     {"INTEGRITY", 0},
194     {"INTO", 0},
195     {"LEVEL", 0},
196     {"LIST", 0},
197     {"LOAD", 1},
198     {"LOCALS", 0},
199     {"LOCKS", 0},
200     {"METHODS", 0},
201     {"NAMESPACE", 0},
202     {"NOTIFY", 2},
203     {"OBJECTS", 1},
204     {"OPEN", 1},
205     {"OSI", 0},
206     {"OWNER", 1},
207     {"PATHS", 0},
208     {"PREDEFINED", 0},
209     {"PREFIX", 0},
210     {"QUIT", 0},
211     {"REFERENCES", 1},
212     {"RESOURCES", 0},
213     {"RESULTS", 0},
214     {"SCI", 0},
215     {"SET", 3},
216     {"SLEEP", 0},
217     {"STATS", 1},
218     {"STOP", 0},
219     {"TABLES", 0},
220     {"TEMPLATE", 1},
221     {"TERMINATE", 0},
222     {"TEST", 1},
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     {8, "Stats <SubCommand>", "Display namespace and memory sta"},
250     {1, "Allocations", "Display list of current memory a"},
251     {1, "Memory", "Dump internal memory lists\n"},
252     {1, "Misc", "Namespace search and mutex stats"},
253     {1, "Objects", "Summary of namespace objects\n"},
254     {1, "Sizes", "Sizes for each of the internal o"},
255     {1, "Stack", "Display CPU stack usage\n"},

```

```

256 {1, " Tables", "Info about current ACPI table(s)",
257 {1, " Tables", "Display info about loaded ACPI t",
258 {1, " Unload <Namepath>", "Unload an ACPI table via namespa",
259 {1, " ! <CommandNumber>", "Execute command from history buf",
260 {1, " !!", "Execute last command again\n"},

262 {0, "\nNamespace Access Commands:", "\n"},
263 {1, " Businfo", "Display system bus info\n"},
264 {1, " Disassemble <Method>", "Disassemble a control method\n"},
265 {1, " Find <AcpiName> (? is wildcard)", "Find ACPI name(s) with wildcards",
266 {1, " Integrity", "Validate namespace integrity\n"},
267 {1, " Methods", "Display list of loaded control m",
268 {1, " Namespace [Object] [Depth]", "Display loaded namespace tree/su",
269 {1, " Notify <Object> <Value>", "Send a notification on Object\n"},
270 {1, " Objects <ObjectType>", "Display all objects of the given",
271 {1, " Owner <OwnerId> [Depth]", "Display loaded namespace by obje",
272 {1, " Paths", "Display full pathnames of namesp",
273 {1, " Predefined", "Check all predefined names\n"},
274 {1, " Prefix [<NamePath>]", "Set or Get current execution pre",
275 {1, " References <Addr>", "Find all references to object at",
276 {1, " Resources [DeviceName]", "Display Device resources (no arg",
277 {1, " Set N <NamedObject> <Value>", "Set value for named integer\n"},
278 {1, " Template <Object>", "Format/dump a Buffer/ResourceTem",
279 {1, " Terminate", "Delete namespace and all interna",
280 {1, " Type <Object>", "Display object type\n"},

282 {0, "\nControl Method Execution Commands:", "\n"},
283 {1, " Arguments (or Args)", "Display method arguments\n"},
284 {1, " Breakpoint <AmlOffset>", "Set an AML execution breakpoint\",
285 {1, " Call", "Run to next control method invoc",
286 {1, " Debug <Namepath> [Arguments]", "Single Step a control method\n"},
287 {6, " Evaluate", "Synonym for Execute\n"},
288 {5, " Execute <Namepath> [Arguments]", "Execute control method\n"},
289 {1, " Hex Integer", "Integer method argument\n"},
290 {1, " \"Ascii String\"", "String method argument\n"},
291 {1, " (Hex Byte List)", "Buffer method argument\n"},
292 {1, " [Package Element List]", "Package method argument\n"},
293 {1, " Go", "Allow method to run to completio",
294 {1, " Information", "Display info about the current m",
295 {1, " Into", "Step into (not over) a method ca",
296 {1, " List [# of Aml Opcodes]", "Display method ASL statements\n"},
297 {1, " Locals", "Display method local variables\n"},
298 {1, " Results", "Display method result stack\n"},
299 {1, " Set <A|L> <#> <Value>", "Set method data (Arguments/Local",
300 {1, " Stop", "Terminate control method\n"},
301 {1, " Thread <Threads><Loops><NamePath>", "Spawn threads to execute method(",
302 {1, " Trace <method name>", "Trace method execution\n"},
303 {1, " Tree", "Display control method calling t",
304 {1, " <Enter>", "Single step next AML opcode (ove",

306 {0, "\nHardware Related Commands:", "\n"},
307 {1, " Event <F|G> <Value>", "Generate AcpiEvent (Fixed/GPE)\n",
308 {1, " Gpe <GpeNum> <GpeBlock>", "Simulate a GPE\n"},
309 {1, " Gpes", "Display info on all GPEs\n"},
310 {1, " Sci", "Generate an SCI\n"},
311 {1, " Sleep [SleepState]", "Simulate sleep/wake sequence(s)",

313 {0, "\nFile I/O Commands:", "\n"},
314 {1, " Close", "Close debug output file\n"},
315 {1, " Load <Input Filename>", "Load ACPI table from a file\n"},
316 {1, " Open <Output Filename>", "Open a file for debug output\n"},

318 {0, "\nDebug Test Commands:", "\n"},
319 {3, " Test <TestName>", "Invoke a debug test\n"},
320 {1, " Objects", "Read/write/compare all namespace",
321 {1, " Predefined", "Execute all ACPI predefined name

```

```

322 {0, NULL, NULL}
323 };

326 /*****
327 *
328 * FUNCTION: AcpiDbMatchCommandHelp
329 *
330 * PARAMETERS: Command - Command string to match
331 *             Help - Help table entry to attempt match
332 *
333 * RETURN: TRUE if command matched, FALSE otherwise
334 *
335 * DESCRIPTION: Attempt to match a command in the help table in order to
336 *              print help information for a single command.
337 *
338 *****/

340 static BOOLEAN
341 AcpiDbMatchCommandHelp (
342     char *Command,
343     const ACPI_DB_COMMAND_HELP *Help)
344 {
345     char *Invocation = Help->Invocation;
346     UINT32 LineCount;

349     /* Valid commands in the help table begin with a couple of spaces */

351     if (*Invocation != ' ')
352     {
353         return (FALSE);
354     }

356     while (*Invocation == ' ')
357     {
358         Invocation++;
359     }

361     /* Match command name (full command or substring) */

363     while ((*Command) && (*Invocation) && (*Invocation != ' '))
364     {
365         if (ACPI_TOLOWER (*Command) != ACPI_TOLOWER (*Invocation))
366         {
367             return (FALSE);
368         }

370         Invocation++;
371         Command++;
372     }

374     /* Print the appropriate number of help lines */

376     LineCount = Help->LineCount;
377     while (LineCount)
378     {
379         AcpiOsPrintf ("%s\n", Help->Invocation, Help->Description);
380         Help++;
381         LineCount--;
382     }

384     return (TRUE);
385 }

```

```

388 /*****
389 *
390 * FUNCTION:      AcpiDbDisplayCommandInfo
391 *
392 * PARAMETERS:    Command          - Command string to match
393 *                DisplayAll       - Display all matching commands, or just
394 *                                the first one (substring match)
395 *
396 * RETURN:        None
397 *
398 * DESCRIPTION:   Display help information for a Debugger command.
399 *
400 *****/

402 static void
403 AcpiDbDisplayCommandInfo (
404     char          *Command,
405     BOOLEAN       DisplayAll)
406 {
407     const ACPI_DB_COMMAND_HELP *Next;
408     BOOLEAN       Matched;

411     Next = AcpiGbl_DbCommandHelp;
412     while (Next->Invocation)
413     {
414         Matched = AcpiDbMatchCommandHelp (Command, Next);
415         if (!DisplayAll && Matched)
416         {
417             return;
418         }

420         Next++;
421     }
422 }

425 /*****
426 *
427 * FUNCTION:      AcpiDbDisplayHelp
428 *
429 * PARAMETERS:    Command          - Optional command string to display help.
430 *                                if not specified, all debugger command
431 *                                help strings are displayed
432 *
433 * RETURN:        None
434 *
435 * DESCRIPTION:   Display help for a single debugger command, or all of them.
436 *
437 *****/

439 static void
440 AcpiDbDisplayHelp (
441     char          *Command)
442 {
443     const ACPI_DB_COMMAND_HELP *Next = AcpiGbl_DbCommandHelp;

446     if (!Command)
447     {
448         /* No argument to help, display help for all commands */

450         while (Next->Invocation)
451         {
452             AcpiOsPrintf ("%38s%s", Next->Invocation, Next->Description);
453             Next++;

```

```

454     }
455 }
456 else
457 {
458     /* Display help for all commands that match the substring */

460     AcpiDbDisplayCommandInfo (Command, TRUE);
461 }
462 }

465 /*****
466 *
467 * FUNCTION:      AcpiDbGetNextToken
468 *
469 * PARAMETERS:    String          - Command buffer
470 *                Next            - Return value, end of next token
471 *
472 * RETURN:        Pointer to the start of the next token.
473 *
474 * DESCRIPTION:   Command line parsing. Get the next token on the command line
475 *
476 *****/

478 char *
479 AcpiDbGetNextToken (
480     char          *String,
481     char          **Next,
482     ACPI_OBJECT_TYPE *ReturnType)
483 {
484     char          *Start;
485     UINT32       Depth;
486     ACPI_OBJECT_TYPE Type = ACPI_TYPE_INTEGER;

489     /* At end of buffer? */

491     if (!String || !(*String))
492     {
493         return (NULL);
494     }

496     /* Remove any spaces at the beginning */

498     if (*String == ' ')
499     {
500         while (*String && (*String == ' '))
501         {
502             String++;
503         }

505         if (!(*String))
506         {
507             return (NULL);
508         }
509     }

511     switch (*String)
512     {
513     case '"':

515         /* This is a quoted string, scan until closing quote */

517         String++;
518         Start = String;
519         Type = ACPI_TYPE_STRING;

```

```

521     /* Find end of string */
522     while (*String && (*String != '"'))
523     {
524         String++;
525     }
526     break;
527
528 case '(':
529     /* This is the start of a buffer, scan until closing paren */
530     String++;
531     Start = String;
532     Type = ACPI_TYPE_BUFFER;
533     /* Find end of buffer */
534     while (*String && (*String != ')'))
535     {
536         String++;
537     }
538     break;
539
540 case '[':
541     /* This is the start of a package, scan until closing bracket */
542     String++;
543     Depth = 1;
544     Start = String;
545     Type = ACPI_TYPE_PACKAGE;
546     /* Find end of package (closing bracket) */
547     while (*String)
548     {
549         /* Handle String package elements */
550         if (*String == '"')
551         {
552             /* Find end of string */
553             String++;
554             while (*String && (*String != '"'))
555             {
556                 String++;
557             }
558             if (!(*String))
559             {
560                 break;
561             }
562         }
563         else if (*String == '[')
564         {
565             Depth++; /* A nested package declaration */
566         }
567         else if (*String == ']')
568         {
569             Depth--;
570             if (Depth == 0) /* Found final package closing bracket */
571             {
572                 break;
573             }
574         }
575     }
576 }

```

```

587         String++;
588     }
589     break;
590
591 default:
592     Start = String;
593     /* Find end of token */
594     while (*String && (*String != ' '))
595     {
596         String++;
597     }
598     break;
599 }
600
601 if (!(*String))
602 {
603     *Next = NULL;
604 }
605 else
606 {
607     *String = 0;
608     *Next = String + 1;
609 }
610
611 *ReturnType = Type;
612 return (Start);
613 }
614
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
627 static UINT32
628 AcpiDbGetLine (
629     char *InputBuffer)
630 {
631     UINT32 i;
632     UINT32 Count;
633     char *Next;
634     char *This;
635
636     if (AcpiUtSafeStrcpy (AcpiGbl_DbParsedBuf, sizeof (AcpiGbl_DbParsedBuf),
637         InputBuffer))
638     {
639         AcpiOsPrintf ("Buffer overflow while parsing input line (max %u characters)\n",
640             sizeof (AcpiGbl_DbParsedBuf));
641         return (0);
642     }
643
644     This = AcpiGbl_DbParsedBuf;
645     for (i = 0; i < ACPI_DEBUGGER_MAX_ARGS; i++)

```

```

652 {
653     AcpiGbl_DbArgs[i] = AcpiDbGetNextToken (This, &Next,
654         &AcpiGbl_DbArgTypes[i]);
655     if (!AcpiGbl_DbArgs[i])
656     {
657         break;
658     }
659
660     This = Next;
661 }
662
663 /* Uppercase the actual command */
664
665 if (AcpiGbl_DbArgs[0])
666 {
667     AcpiUtStrupr (AcpiGbl_DbArgs[0]);
668 }
669
670 Count = i;
671 if (Count)
672 {
673     Count--; /* Number of args only */
674 }
675
676 return (Count);
677 }
678
679 /*****
680 *
681 * FUNCTION:      AcpiDbMatchCommand
682 *
683 * PARAMETERS:    UserCommand          - User command line
684 *
685 * RETURN:        Index into command array, -1 if not found
686 *
687 * DESCRIPTION:    Search command array for a command match
688 *
689 * *****/
690
691 static UINT32
692 AcpiDbMatchCommand (
693     char                *UserCommand)
694 {
695     UINT32              i;
696
697     if (!UserCommand || UserCommand[0] == 0)
698     {
699         return (CMD_NULL);
700     }
701
702     for (i = CMD_FIRST_VALID; AcpiGbl_DbCommands[i].Name; i++)
703     {
704         if (ACPI_STRSTR (AcpiGbl_DbCommands[i].Name, UserCommand) ==
705             AcpiGbl_DbCommands[i].Name)
706         {
707             return (i);
708         }
709     }
710
711     /* Command not recognized */
712
713     return (CMD_NOT_FOUND);
714 }

```

```

719 /*****
720 *
721 * FUNCTION:      AcpiDbCommandDispatch
722 *
723 * PARAMETERS:    InputBuffer          - Command line buffer
724 *                WalkState            - Current walk
725 *                Op                      - Current (executing) parse op
726 *
727 * RETURN:        Status
728 *
729 * DESCRIPTION:    Command dispatcher.
730 *
731 * *****/
732
733 ACPI_STATUS
734 AcpiDbCommandDispatch (
735     char                *InputBuffer,
736     ACPI_WALK_STATE     *WalkState,
737     ACPI_PARSE_OBJECT   *Op)
738 {
739     UINT32              Temp;
740     UINT32              CommandIndex;
741     UINT32              ParamCount;
742     char                *CommandLine;
743     ACPI_STATUS          Status = AE_CTRL_TRUE;
744
745     /* If AcpiTerminate has been called, terminate this thread */
746
747     if (AcpiGbl_DbTerminateThreads)
748     {
749         return (AE_CTRL_TERMINATE);
750     }
751
752     /* Add all commands that come here to the history buffer */
753
754     AcpiDbAddToHistory (InputBuffer);
755
756     ParamCount = AcpiDbGetLine (InputBuffer);
757     CommandIndex = AcpiDbMatchCommand (AcpiGbl_DbArgs[0]);
758     Temp = 0;
759
760     /* Verify that we have the minimum number of params */
761
762     if (ParamCount < AcpiGbl_DbCommands[CommandIndex].MinArgs)
763     {
764         AcpiOsPrintf ("%u parameters entered, [%s] requires %u parameters\n",
765             ParamCount, AcpiGbl_DbCommands[CommandIndex].Name,
766             AcpiGbl_DbCommands[CommandIndex].MinArgs);
767
768         AcpiDbDisplayCommandInfo (AcpiGbl_DbCommands[CommandIndex].Name, FALSE);
769         return (AE_CTRL_TRUE);
770     }
771
772     /* Decode and dispatch the command */
773
774     switch (CommandIndex)
775     {
776     case CMD_NULL:
777
778         if (Op)
779         {
780             return (AE_OK);
781         }
782     }

```

```

784     break;

786     case CMD_ALLOCATIONS:

788 #ifdef ACPI_DBG_TRACK_ALLOCATIONS
789     AcpiUtDumpAllocations ((UINT32) -1, NULL);
790 #endif
791     break;

793     case CMD_ARGS:
794     case CMD_ARGUMENTS:

796         AcpiDbDisplayArguments ();
797         break;

799     case CMD_BREAKPOINT:

801         AcpiDbSetMethodBreakpoint (AcpiGbl_DbArgs[1], WalkState, Op);
802         break;

804     case CMD_BUSINFO:

806         AcpiDbGetBusInfo ();
807         break;

809     case CMD_CALL:

811         AcpiDbSetMethodCallBreakpoint (Op);
812         Status = AE_OK;
813         break;

815     case CMD_CLOSE:

817         AcpiDbCloseDebugFile ();
818         break;

820     case CMD_DEBUG:

822         AcpiDbExecute (AcpiGbl_DbArgs[1],
823             &AcpiGbl_DbArgs[2], &AcpiGbl_DbArgTypes[2], EX_SINGLE_STEP);
824         break;

826     case CMD_DISASSEMBLE:
827     case CMD_DISASM:

829         (void) AcpiDbDisassembleMethod (AcpiGbl_DbArgs[1]);
830         break;

832     case CMD_DUMP:

834         AcpiDbDecodeAndDisplayObject (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2]);
835         break;

837     case CMD_ENABLEACPI:
838     #if (!ACPI_REduced_HARDWARE)

840         Status = AcpiEnable();
841         if (ACPI_FAILURE(Status))
842         {
843             AcpiOsPrintf("AcpiEnable failed (Status=%X)\n", Status);
844             return (Status);
845         }
846 #endif /* !ACPI_REduced_HARDWARE */
847         break;

849     case CMD_EVENT:

```

```

851         AcpiOsPrintf ("Event command not implemented\n");
852         break;

854     case CMD_EVALUATE:
855     case CMD_EXECUTE:

857         AcpiDbExecute (AcpiGbl_DbArgs[1],
858             &AcpiGbl_DbArgs[2], &AcpiGbl_DbArgTypes[2], EX_NO_SINGLE_STEP);
859         break;

861     case CMD_FIND:

863         Status = AcpiDbFindNameInNamespace (AcpiGbl_DbArgs[1]);
864         break;

866     case CMD_GO:

868         AcpiGbl_CmSingleStep = FALSE;
869         return (AE_OK);

871     case CMD_GPE:

873         AcpiDbGenerateGpe (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2]);
874         break;

876     case CMD_GPES:

878         AcpiDbDisplayGpes ();
879         break;

881     case CMD_HANDLERS:

883         AcpiDbDisplayHandlers ();
884         break;

886     case CMD_HELP:
887     case CMD_HELP2:

889         AcpiDbDisplayHelp (AcpiGbl_DbArgs[1]);
890         break;

892     case CMD_HISTORY:

894         AcpiDbDisplayHistory ();
895         break;

897     case CMD_HISTORY_EXE: /* ! command */

899         CommandLine = AcpiDbGetFromHistory (AcpiGbl_DbArgs[1]);
900         if (!CommandLine)
901         {
902             return (AE_CTRL_TRUE);
903         }

905         Status = AcpiDbCommandDispatch (CommandLine, WalkState, Op);
906         return (Status);

908     case CMD_HISTORY_LAST: /* !! command */

910         CommandLine = AcpiDbGetFromHistory (NULL);
911         if (!CommandLine)
912         {
913             return (AE_CTRL_TRUE);
914         }

```



```

916     Status = AcpiDbCommandDispatch (CommandLine, WalkState, Op);
917     return (Status);

919     case CMD_INFORMATION:

921         AcpiDbDisplayMethodInfo (Op);
922         break;

924     case CMD_INTEGRITY:

926         AcpiDbCheckIntegrity ();
927         break;

929     case CMD_INTTO:

931         if (Op)
932         {
933             AcpiGbl_CmSingleStep = TRUE;
934             return (AE_OK);
935         }
936         break;

938     case CMD_LEVEL:

940         if (ParamCount == 0)
941         {
942             AcpiOsPrintf ("Current debug level for file output is:  %8.8lX\n",
943                 AcpiGbl_DbDebugLevel);
944             AcpiOsPrintf ("Current debug level for console output is: %8.8lX\n",
945                 AcpiGbl_DbConsoleDebugLevel);
946         }
947         else if (ParamCount == 2)
948         {
949             Temp = AcpiGbl_DbConsoleDebugLevel;
950             AcpiGbl_DbConsoleDebugLevel = ACPI_STRTOUL (AcpiGbl_DbArgs[1],
951                 NULL, 16);
952             AcpiOsPrintf (
953                 "Debug Level for console output was %8.8lX, now %8.8lX\n",
954                 Temp, AcpiGbl_DbConsoleDebugLevel);
955         }
956         else
957         {
958             Temp = AcpiGbl_DbDebugLevel;
959             AcpiGbl_DbDebugLevel = ACPI_STRTOUL (AcpiGbl_DbArgs[1], NULL, 16);
960             AcpiOsPrintf (
961                 "Debug Level for file output was %8.8lX, now %8.8lX\n",
962                 Temp, AcpiGbl_DbDebugLevel);
963         }
964         break;

966     case CMD_LIST:

968         AcpiDbDisassembleAml (AcpiGbl_DbArgs[1], Op);
969         break;

971     case CMD_LOAD:

973         Status = AcpiDbGetTableFromFile (AcpiGbl_DbArgs[1], NULL);
974         break;

976     case CMD_LOCKS:

978         AcpiDbDisplayLocks ();
979         break;

981     case CMD_LOCALS:

```

```

983         AcpiDbDisplayLocals ();
984         break;

986     case CMD_METHODS:

988         Status = AcpiDbDisplayObjects ("METHOD", AcpiGbl_DbArgs[1]);
989         break;

991     case CMD_NAMESPACE:

993         AcpiDbDumpNamespace (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2]);
994         break;

996     case CMD_NOTIFY:

998         Temp = ACPI_STRTOUL (AcpiGbl_DbArgs[2], NULL, 0);
999         AcpiDbSendNotify (AcpiGbl_DbArgs[1], Temp);
1000         break;

1002     case CMD_OBJECTS:

1004         AcpiUtStrupr (AcpiGbl_DbArgs[1]);
1005         Status = AcpiDbDisplayObjects (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2]);
1006         break;

1008     case CMD_OPEN:

1010         AcpiDbOpenDebugFile (AcpiGbl_DbArgs[1]);
1011         break;

1013     case CMD_OSI:

1015         AcpiDbDisplayInterfaces (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2]);
1016         break;

1018     case CMD_OWNER:

1020         AcpiDbDumpNamespaceByOwner (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2]);
1021         break;

1023     case CMD_PATHS:

1025         AcpiDbDumpNamespacePaths ();
1026         break;

1028     case CMD_PREDEFINED:

1030         AcpiDbCheckPredefinedNames ();
1031         break;

1033     case CMD_PREFIX:

1035         AcpiDbSetScope (AcpiGbl_DbArgs[1]);
1036         break;

1038     case CMD_REFERENCES:

1040         AcpiDbFindReferences (AcpiGbl_DbArgs[1]);
1041         break;

1043     case CMD_RESOURCES:

1045         AcpiDbDisplayResources (AcpiGbl_DbArgs[1]);
1046         break;

```

```

1048     case CMD_RESULTS:
1050         AcpiDbDisplayResults ();
1051         break;
1053     case CMD_SCI:
1055         AcpiDbGenerateSci ();
1056         break;
1058     case CMD_SET:
1060         AcpiDbSetMethodData (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2],
1061             AcpiGbl_DbArgs[3]);
1062         break;
1064     case CMD_SLEEP:
1066         Status = AcpiDbSleep (AcpiGbl_DbArgs[1]);
1067         break;
1069     case CMD_STATS:
1071         Status = AcpiDbDisplayStatistics (AcpiGbl_DbArgs[1]);
1072         break;
1074     case CMD_STOP:
1076         return (AE_NOT_IMPLEMENTED);
1078     case CMD_TABLES:
1080         AcpiDbDisplayTableInfo (AcpiGbl_DbArgs[1]);
1081         break;
1083     case CMD_TEMPLATE:
1085         AcpiDbDisplayTemplate (AcpiGbl_DbArgs[1]);
1086         break;
1088     case CMD_TERMINATE:
1090         AcpiDbSetOutputDestination (ACPI_DB_REDIRECTABLE_OUTPUT);
1091         AcpiUtSubsystemShutdown ();
1093         /*
1094          * TBD: [Restructure] Need some way to re-initialize without
1095          * re-creating the semaphores!
1096          */
1098         /* AcpiInitialize (NULL); */
1099         break;
1101     case CMD_TEST:
1103         AcpiDbExecuteTest (AcpiGbl_DbArgs[1]);
1104         break;
1106     case CMD_THREADS:
1108         AcpiDbCreateExecutionThreads (AcpiGbl_DbArgs[1], AcpiGbl_DbArgs[2],
1109             AcpiGbl_DbArgs[3]);
1110         break;
1112     case CMD_TRACE:

```

```

1114         (void) AcpiDebugTrace (AcpiGbl_DbArgs[1], 0, 0, 1);
1115         break;
1117     case CMD_TREE:
1119         AcpiDbDisplayCallingTree ();
1120         break;
1122     case CMD_TYPE:
1124         AcpiDbDisplayObjectType (AcpiGbl_DbArgs[1]);
1125         break;
1127     case CMD_UNLOAD:
1129         AcpiDbUnloadAcpiTable (AcpiGbl_DbArgs[1]);
1130         break;
1132     case CMD_EXIT:
1133     case CMD_QUIT:
1135         if (Op)
1136         {
1137             AcpiOsPrintf ("Method execution terminated\n");
1138             return (AE_CTRL_TERMINATE);
1139         }
1141         if (!AcpiGbl_DbOutputToFile)
1142         {
1143             AcpiDbgLevel = ACPI_DEBUG_DEFAULT;
1144         }
1146         AcpiDbCloseDebugFile ();
1147         AcpiGbl_DbTerminateThreads = TRUE;
1148         return (AE_CTRL_TERMINATE);
1150     case CMD_NOT_FOUND:
1151     default:
1153         AcpiOsPrintf ("%s: unknown command\n", AcpiGbl_DbArgs[0]);
1154         return (AE_CTRL_TRUE);
1155     }
1157     if (ACPI_SUCCESS (Status))
1158     {
1159         Status = AE_CTRL_TRUE;
1160     }
1162     return (Status);
1163 }
1166 /*****
1167  *
1168  * FUNCTION:      AcpiDbExecuteThread
1169  *
1170  * PARAMETERS:    Context          - Not used
1171  *
1172  * RETURN:        None
1173  *
1174  * DESCRIPTION:   Debugger execute thread. Waits for a command line, then
1175  *               simply dispatches it.
1176  *
1177  *****/
1179 void ACPI_SYSTEM_XFACE

```

```

1180 AcpiDbExecuteThread (
1181     void                *Context)
1182 {
1183     ACPI_STATUS          Status = AE_OK;
1184     ACPI_STATUS          MStatus;

    while (Status != AE_CTRL_TERMINATE)
    {
        AcpiGbl_MethodExecuting = FALSE;
        AcpiGbl_StepToNextCall = FALSE;

        MStatus = AcpiUtAcquireMutex (ACPI_MTX_DEBUG_CMD_READY);
        if (ACPI_FAILURE (MStatus))
        {
            return;
        }

        Status = AcpiDbCommandDispatch (AcpiGbl_DbLineBuf, NULL, NULL);

        MStatus = AcpiUtReleaseMutex (ACPI_MTX_DEBUG_CMD_COMPLETE);
        if (ACPI_FAILURE (MStatus))
        {
            return;
        }
    }
}

1209 /*****
1210  *
1211  * FUNCTION:      AcpiDbSingleThread
1212  *
1213  * PARAMETERS:    None
1214  *
1215  * RETURN:        None
1216  *
1217  * DESCRIPTION:   Debugger execute thread. Waits for a command line, then
1218  *               simply dispatches it.
1219  *
1220  *****/

1222 static void
1223 AcpiDbSingleThread (
1224     void)
1225 {
    AcpiGbl_MethodExecuting = FALSE;
    AcpiGbl_StepToNextCall = FALSE;

    (void) AcpiDbCommandDispatch (AcpiGbl_DbLineBuf, NULL, NULL);
}

1234 /*****
1235  *
1236  * FUNCTION:      AcpiDbUserCommands
1237  *
1238  * PARAMETERS:    Prompt          - User prompt (depends on mode)
1239  *               Op              - Current executing parse op
1240  *
1241  * RETURN:        None
1242  *
1243  * DESCRIPTION:   Command line execution for the AML debugger. Commands are
1244  *               matched and dispatched here.
1245  *

```

```

1246 *****/

1248 ACPI_STATUS
1249 AcpiDbUserCommands (
1250     char                Prompt,
1251     ACPI_PARSE_OBJECT   *Op)
1252 {
1253     ACPI_STATUS          Status = AE_OK;

    AcpiOsPrintf ("\n");

    /* TBD: [Restructure] Need a separate command line buffer for step mode */

    while (!AcpiGbl_DbTerminateThreads)
    {
        /* Force output to console until a command is entered */

        AcpiDbSetOutputDestination (ACPI_DB_CONSOLE_OUTPUT);

        /* Different prompt if method is executing */

        if (!AcpiGbl_MethodExecuting)
        {
            AcpiOsPrintf ("%1c ", ACPI_DEBUGGER_COMMAND_PROMPT);
        }
        else
        {
            AcpiOsPrintf ("%1c ", ACPI_DEBUGGER_EXECUTE_PROMPT);
        }

        /* Get the user input line */

        Status = AcpiOsGetLine (AcpiGbl_DbLineBuf,
                                ACPI_DB_LINE_BUFFER_SIZE, NULL);
        if (ACPI_FAILURE (Status))
        {
            ACPI_EXCEPTION ((AE_INFO, Status, "While parsing command line"));
            return (Status);
        }

        /* Check for single or multithreaded debug */

        if (AcpiGbl_DebuggerConfiguration & DEBUGGER_MULTI_THREADED)
        {
            /*
             * Signal the debug thread that we have a command to execute,
             * and wait for the command to complete.
             */
            Status = AcpiUtReleaseMutex (ACPI_MTX_DEBUG_CMD_READY);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }

            Status = AcpiUtAcquireMutex (ACPI_MTX_DEBUG_CMD_COMPLETE);
            if (ACPI_FAILURE (Status))
            {
                return (Status);
            }
        }
        else
        {
            /* Just call to the command line interpreter */

            AcpiDbSingleThread ();
        }
    }
}

```

```
1312     }
1313 }
1314
1315 /*
1316  * Only this thread (the original thread) should actually terminate the
1317  * subsystem, because all the semaphores are deleted during termination
1318  */
1319 Status = AcpiTerminate ();
1320 return (Status);
1321 }
1322
1323 #endif /* ACPI_DEBUGGER */
```

```

*****
11003 Fri Jan 17 07:29:49 2014
new/usr/src/common/acpica/components/debugger/dbmethod.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dbmethod - Debug commands for control methods
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acdispat.h"
48 #include "acnamesp.h"
49 #include "acdebug.h"
50 #include "acdisasm.h"
51 #include "acparser.h"
52 #include "acpredef.h"

55 #ifdef ACPI_DEBUGGER

57 #define _COMPONENT          ACPI_CA_DEBUGGER
58     ACPI_MODULE_NAME    ("dbmethod")

```

```

61 /*****
62 *
63 * FUNCTION:      AcpiDbSetMethodBreakpoint
64 *
65 * PARAMETERS:    Location          - AML offset of breakpoint
66 *                WalkState         - Current walk info
67 *                Op                - Current Op (from parse walk)
68 *
69 * RETURN:        None
70 *
71 * DESCRIPTION:   Set a breakpoint in a control method at the specified
72 *                AML offset
73 *
74 *****/

76 void
77 AcpiDbSetMethodBreakpoint (
78     char                *Location,
79     ACPI_WALK_STATE     *WalkState,
80     ACPI_PARSE_OBJECT   *Op)
81 {
82     UINT32              Address;

85     if (!Op)
86     {
87         AcpiOsPrintf ("There is no method currently executing\n");
88         return;
89     }

91     /* Get and verify the breakpoint address */

93     Address = ACPI_STRToul (Location, NULL, 16);
94     if (Address <= Op->Common.AmlOffset)
95     {
96         AcpiOsPrintf ("Breakpoint %X is beyond current address %X\n",
97             Address, Op->Common.AmlOffset);
98     }

100     /* Save breakpoint in current walk */

102     WalkState->UserBreakpoint = Address;
103     AcpiOsPrintf ("Breakpoint set at AML offset %X\n", Address);
104 }

107 /*****
108 *
109 * FUNCTION:      AcpiDbSetMethodCallBreakpoint
110 *
111 * PARAMETERS:    Op                - Current Op (from parse walk)
112 *
113 * RETURN:        None
114 *
115 * DESCRIPTION:   Set a breakpoint in a control method at the specified
116 *                AML offset
117 *
118 *****/

120 void
121 AcpiDbSetMethodCallBreakpoint (
122     ACPI_PARSE_OBJECT   *Op)
123 {

```

```

126     if (!Op)
127     {
128         AcpiOsPrintf ("There is no method currently executing\n");
129         return;
130     }

132     AcpiGbl_StepToNextCall = TRUE;
133 }

136 /*****
137  *
138  * FUNCTION:      AcpiDbSetMethodData
139  *
140  * PARAMETERS:    TypeArg      - L for local, A for argument
141  *                IndexArg     - which one
142  *                ValueArg      - Value to set.
143  *
144  * RETURN:        None
145  *
146  * DESCRIPTION:   Set a local or argument for the running control method.
147  *                NOTE: only object supported is Number.
148  *
149  *****/

151 void
152 AcpiDbSetMethodData (
153     char          *TypeArg,
154     char          *IndexArg,
155     char          *ValueArg)
156 {
157     char          Type;
158     UINT32        Index;
159     UINT32        Value;
160     ACPI_WALK_STATE *WalkState;
161     ACPI_OPERAND_OBJECT *ObjDesc;
162     ACPI_STATUS    Status;
163     ACPI_NAMESPACE_NODE *Node;

166     /* Validate TypeArg */

168     AcpiUtStrupr (TypeArg);
169     Type = TypeArg[0];
170     if ((Type != 'L') &&
171         (Type != 'A') &&
172         (Type != 'N'))
173     {
174         AcpiOsPrintf ("Invalid SET operand: %s\n", TypeArg);
175         return;
176     }

178     Value = ACPI_STRTOUL (ValueArg, NULL, 16);

180     if (Type == 'N')
181     {
182         Node = AcpiDbConvertToNode (IndexArg);
183         if (Node->Type != ACPI_TYPE_INTEGER)
184         {
185             AcpiOsPrintf ("Can only set Integer nodes\n");
186             return;
187         }
188         ObjDesc = Node->Object;
189         ObjDesc->Integer.Value = Value;
190         return;
191     }

```

```

193     /* Get the index and value */

195     Index = ACPI_STRTOUL (IndexArg, NULL, 16);

197     WalkState = AcpiDsGetCurrentWalkState (AcpiGbl_CurrentWalkList);
198     if (!WalkState)
199     {
200         AcpiOsPrintf ("There is no method currently executing\n");
201         return;
202     }

204     /* Create and initialize the new object */

206     ObjDesc = AcpiUtCreateIntegerObject ((UINT64) Value);
207     if (!ObjDesc)
208     {
209         AcpiOsPrintf ("Could not create an internal object\n");
210         return;
211     }

213     /* Store the new object into the target */

215     switch (Type)
216     {
217     case 'A':

219         /* Set a method argument */

221         if (Index > ACPI_METHOD_MAX_ARG)
222         {
223             AcpiOsPrintf ("Arg%u - Invalid argument name\n", Index);
224             goto Cleanup;
225         }

227         Status = AcpiDsStoreObjectToLocal (ACPI_REFCLASS_ARG, Index, ObjDesc,
228             WalkState);
229         if (ACPI_FAILURE (Status))
230         {
231             goto Cleanup;
232         }

234         ObjDesc = WalkState->Arguments[Index].Object;

236         AcpiOsPrintf ("Arg%u: ", Index);
237         AcpiDmDisplayInternalObject (ObjDesc, WalkState);
238         break;

240     case 'L':

242         /* Set a method local */

244         if (Index > ACPI_METHOD_MAX_LOCAL)
245         {
246             AcpiOsPrintf ("Local%u - Invalid local variable name\n", Index);
247             goto Cleanup;
248         }

250         Status = AcpiDsStoreObjectToLocal (ACPI_REFCLASS_LOCAL, Index, ObjDesc,
251             WalkState);
252         if (ACPI_FAILURE (Status))
253         {
254             goto Cleanup;
255         }

257         ObjDesc = WalkState->LocalVariables[Index].Object;

```

```

259     AcpiOsPrintf ("Local%u: ", Index);
260     AcpiDmDisplayInternalObject (ObjDesc, WalkState);
261     break;

263     default:

265         break;
266     }

268 Cleanup:
269     AcpiUtRemoveReference (ObjDesc);
270 }

273 /*****
274  *
275  * FUNCTION:     AcpiDbDisassembleAml
276  *
277  * PARAMETERS:  Statements      - Number of statements to disassemble
278  *              Op              - Current Op (from parse walk)
279  *
280  * RETURN:      None
281  *
282  * DESCRIPTION: Display disassembled AML (ASL) starting from Op for the number
283  *              of statements specified.
284  *
285  *****/

287 void
288 AcpiDbDisassembleAml (
289     char          *Statements,
290     ACPI_PARSE_OBJECT *Op)
291 {
292     UINT32          NumStatements = 8;

295     if (!Op)
296     {
297         AcpiOsPrintf ("There is no method currently executing\n");
298         return;
299     }

301     if (Statements)
302     {
303         NumStatements = ACPI_STRTOUL (Statements, NULL, 0);
304     }

306     AcpiDmDisassemble (NULL, Op, NumStatements);
307 }

310 /*****
311  *
312  * FUNCTION:     AcpiDbDisassembleMethod
313  *
314  * PARAMETERS:  Name          - Name of control method
315  *
316  * RETURN:      None
317  *
318  * DESCRIPTION: Display disassembled AML (ASL) starting from Op for the number
319  *              of statements specified.
320  *
321  *****/

323 ACPI_STATUS

```

```

324 AcpiDbDisassembleMethod (
325     char          *Name)
326 {
327     ACPI_STATUS      Status;
328     ACPI_PARSE_OBJECT *Op;
329     ACPI_WALK_STATE  *WalkState;
330     ACPI_OPERAND_OBJECT *ObjDesc;
331     ACPI_NAMESPACE_NODE *Method;

334     Method = AcpiDbConvertToNode (Name);
335     if (!Method)
336     {
337         return (AE_BAD_PARAMETER);
338     }

340     if (Method->Type != ACPI_TYPE_METHOD)
341     {
342         ACPI_ERROR ((AE_INFO, "%s (%s): Object must be a control method",
343             Name, AcpiUtGetTypeName (Method->Type)));
344         return (AE_BAD_PARAMETER);
345     }

347     ObjDesc = Method->Object;

349     Op = AcpiPsCreateScopeOp ();
350     if (!Op)
351     {
352         return (AE_NO_MEMORY);
353     }

355     /* Create and initialize a new walk state */

357     WalkState = AcpiDsCreateWalkState (0, Op, NULL, NULL);
358     if (!WalkState)
359     {
360         return (AE_NO_MEMORY);
361     }

363     Status = AcpiDsInitAmlWalk (WalkState, Op, NULL,
364         ObjDesc->Method.AmlStart,
365         ObjDesc->Method.AmlLength, NULL, ACPI_IMODE_LOAD_PASS1);
366     if (ACPI_FAILURE (Status))
367     {
368         return (Status);
369     }

371     Status = AcpiUtAllocateOwnerId (&ObjDesc->Method.OwnerId);
372     WalkState->OwnerId = ObjDesc->Method.OwnerId;

374     /* Push start scope on scope stack and make it current */

376     Status = AcpiDsScopeStackPush (Method,
377         Method->Type, WalkState);
378     if (ACPI_FAILURE (Status))
379     {
380         return (Status);
381     }

383     /* Parse the entire method AML including deferred operators */

385     WalkState->ParseFlags &= ~ACPI_PARSE_DELETE_TREE;
386     WalkState->ParseFlags |= ACPI_PARSE_DISASSEMBLE;

388     Status = AcpiPsParseAml (WalkState);
389     (void) AcpiDmParseDeferredOps (Op);

```

```
391     /* Now we can disassemble the method */
393     AcpiGbl_DbOpt_verbose = FALSE;
394     AcpiDmDisassemble (NULL, Op, 0);
395     AcpiGbl_DbOpt_verbose = TRUE;
397     AcpiPsDeleteParseTree (Op);
399     /* Method cleanup */
401     AcpiNsDeleteNamespaceSubtree (Method);
402     AcpiNsDeleteNamespaceByOwner (ObjDesc->Method.OwnerId);
403     AcpiUtReleaseOwnerId (&ObjDesc->Method.OwnerId);
404     return (AE_OK);
405 }
407 #endif /* ACPI_DEBUGGER */
```



new/usr/src/common/acpica/components/debugger/dbnames.c

1

```
*****
26977 Fri Jan 17 07:29:49 2014
new/usr/src/common/acpica/components/debugger/dbnames.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dbnames - Debugger commands for the acpi namespace
4 *
5 *****/
6 /
7 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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")
```

new/usr/src/common/acpica/components/debugger/dbnames.c

2

```
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 [] =
107 static ARGUMENT_INFO AcpiDbObjectTypes [] =
108 {
109     {"ANY"},
110     {"INTEGERS"},
111     {"STRINGS"},
112     {"BUFFERS"},
113     {"PACKAGES"},
114     {"FIELDS"},
115     {"DEVICES"},
116     {"EVENTS"},
117     {"METHODS"},
118     {"MUTEXES"},
119     {"REGIONS"},
120     {"POWERRESOURCES"},
121     {"PROCESSORS"},
122     {"THERMALZONES"},
123     {"BUFFERFIELDS"},
```

```

123     {"DDBHANDLES"},
124     {"DEBUG"},
125     {"REGIONFIELDS"},
126     {"BANKFIELDS"},
127     {"INDEXFIELDS"},
128     {"REFERENCES"},
129     {"ALIASES"},
130     {NULL} /* Must be null terminated */
131 };

134 /*****
135  *
136  * FUNCTION:      AcpiDbSetScope
137  *
138  * PARAMETERS:    Name                - New scope path
139  *
140  * RETURN:        Status
141  *
142  * DESCRIPTION:   Set the "current scope" as maintained by this utility.
143  *               The scope is used as a prefix to ACPI paths.
144  *
145  *****/

147 void
148 AcpiDbSetScope (
149     char                *Name)
150 {
151     ACPI_STATUS          Status;
152     ACPI_NAMESPACE_NODE *Node;

155     if (!Name || Name[0] == 0)
156     {
157         AcpiOsPrintf ("Current scope: %s\n", AcpiGbl_DbScopeBuf);
158         return;
159     }

161     AcpiDbPrepNamestring (Name);

163     if (ACPI_IS_ROOT_PREFIX (Name[0]))
164     {
165         /* Validate new scope from the root */

167         Status = AcpiNsGetNode (AcpiGbl_RootNode, Name, ACPI_NS_NO_UPSEARCH,
168                                 &Node);
169         if (ACPI_FAILURE (Status))
170         {
171             goto ErrorExit;
172         }

174         AcpiGbl_DbScopeBuf[0] = 0;
175         ACPI_STRCPY (AcpiGbl_DbScopeBuf, Name);
176         ACPI_STRCAT (AcpiGbl_DbScopeBuf, "\\");
177     }
178     else
179     {
180         /* Validate new scope relative to old scope */

181         Status = AcpiNsGetNode (AcpiGbl_DbScopeNode, Name, ACPI_NS_NO_UPSEARCH,
182                                 &Node);
183         if (ACPI_FAILURE (Status))
184         {
185             goto ErrorExit;
186         }

```

```

186     }

188     /* Build the final pathname */

190     if (AcpiUtSafeStrcat (AcpiGbl_DbScopeBuf, sizeof (AcpiGbl_DbScopeBuf),
191                           Name))
192     {
193         Status = AE_BUFFER_OVERFLOW;
194         goto ErrorExit;
195     }

197     if (AcpiUtSafeStrcat (AcpiGbl_DbScopeBuf, sizeof (AcpiGbl_DbScopeBuf),
198                           "\\"))
199     {
200         Status = AE_BUFFER_OVERFLOW;
201         goto ErrorExit;
202     }

204     AcpiGbl_DbScopeNode = Node;
205     AcpiOsPrintf ("New scope: %s\n", AcpiGbl_DbScopeBuf);
206     return;

208 ErrorExit:

210     AcpiOsPrintf ("Could not attach scope: %s, %s\n",
211                   Name, AcpiFormatException (Status));
212 }

    unchanged_portion_omitted

269 /*****
270  *
271  * FUNCTION:      AcpiDbDumpNamespacePaths
272  *
273  * PARAMETERS:    None
274  *
275  * RETURN:        None
276  *
277  * DESCRIPTION:   Dump entire namespace with full object pathnames and object
278  *               type information. Alternative to "namespace" command.
279  *
280  *****/

282 void
283 AcpiDbDumpNamespacePaths (
284     void)
285 {
287     AcpiDbSetOutputDestination (ACPI_DB_DUPLICATE_OUTPUT);
288     AcpiOsPrintf ("ACPI Namespace (from root):\n");

290     /* Display the entire namespace */

292     AcpiDbSetOutputDestination (ACPI_DB_REDIRECTABLE_OUTPUT);
293     AcpiNsDumpObjectPaths (ACPI_TYPE_ANY, ACPI_DISPLAY_SUMMARY,
294                             ACPI_UINT32_MAX, ACPI_OWNER_ID_MAX, AcpiGbl_RootNode);

296     AcpiDbSetOutputDestination (ACPI_DB_CONSOLE_OUTPUT);
297 }

300 /*****
301  *
302  * FUNCTION:      AcpiDbDumpNamespaceByOwner

```

```

303 *
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 *****/
312
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;
321
322     OwnerId = (ACPI_OWNER_ID) ACPI_STRTOL (OwnerArg, NULL, 0);
323
324     /* Now we can check for the depth argument */
325
326     if (DepthArg)
327     {
328         MaxDepth = ACPI_STRTOL (DepthArg, NULL, 0);
329     }
330
331     AcpiDbSetOutputDestination (ACPI_DB_DUPLICATE_OUTPUT);
332     AcpiOsPrintf ("ACPI Namespace by owner %X:\n", OwnerId);
333
334     /* Display the subtree */
335
336     AcpiDbSetOutputDestination (ACPI_DB_REDIRECTABLE_OUTPUT);
337     AcpiNsDumpObjects (ACPI_TYPE_ANY, ACPI_DISPLAY_SUMMARY, MaxDepth, OwnerId,
338         SubtreeEntry);
339     AcpiDbSetOutputDestination (ACPI_DB_CONSOLE_OUTPUT);
340 }
341
342 _____unchanged_portion_omitted_____
343
344
345 /*****
346 *
347 * FUNCTION:      AcpiDbWalkForPredefinedNames
348 *
349 * PARAMETERS:    Callback from WalkNamespace
350 *
351 * RETURN:        Status
352 *
353 * DESCRIPTION:   Detect and display predefined ACPI names (names that start with
354 *               an underscore)
355 *
356 *****/
357
358 static ACPI_STATUS
359 AcpiDbWalkForPredefinedNames (
360     ACPI_HANDLE    ObjHandle,
361     UINT32         NestingLevel,
362     void           *Context,
363     void           **ReturnValue)
364 {
365     ACPI_NAMESPACE_NODE *Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
366     UINT32              *Count = (UINT32 *) Context;
367     const ACPI_PREDEFINED_INFO *Predefined;
368     const ACPI_PREDEFINED_INFO *Package = NULL;

```

```

480     char          *Pathname;
481     char          StringBuffer[48];
482
483
484     Predefined = AcpiUtMatchPredefinedMethod (Node->Name.Ascii);
485     if (!Predefined)
486     {
487         return (AE_OK);
488     }
489
490     Pathname = AcpiNsGetExternalPathname (Node);
491     if (!Pathname)
492     {
493         return (AE_OK);
494     }
495
496     /* If method returns a package, the info is in the next table entry */
497
498     if (Predefined->Info.ExpectedBtypes & ACPI_RTYPE_PACKAGE)
499     {
500         Package = Predefined + 1;
501     }
502
503     AcpiUtGetExpectedReturnTypes (StringBuffer,
504         Predefined->Info.ExpectedBtypes);
505     AcpiOsPrintf ("%32s arg %X ret %2.2X", Pathname,
506         Predefined->Info.ParamCount, Predefined->Info.ExpectedBtypes);
507
508     AcpiOsPrintf ("%32s Arguments %X, Return Types: %s", Pathname,
509         METHOD_GET_ARG_COUNT (Predefined->Info.ArgumentList),
510         StringBuffer);
511
512     if (Package)
513     {
514         AcpiOsPrintf (" (PkgType %2.2X, ObjType %2.2X, Count %2.2X)",
515             PkgType %2.2X ObjType %2.2X Count %2.2X",
516             Package->RetInfo.Type, Package->RetInfo.ObjectType1,
517             Package->RetInfo.Count1);
518     }
519
520     AcpiOsPrintf("\n");
521
522     /* Check that the declared argument count matches the ACPI spec */
523
524     AcpiNsCheckAcpiCompliance (Pathname, Node, Predefined);
525
526     AcpiNsCheckParameterCount (Pathname, Node, ACPI_UINT32_MAX, Predefined);
527     ACPI_FREE (Pathname);
528     (*Count)++;
529
530     return (AE_OK);
531 }
532
533 _____unchanged_portion_omitted_____
534
535
536 /*****
537 *
538 * FUNCTION:      AcpiDbIntegrityWalk
539 *
540 * PARAMETERS:    Callback from WalkNamespace
541 *
542 * RETURN:        Status
543 *
544 * 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))
718     if (!AcpiUtValidAcpiName (Node->Name.Integer))
719     {
720         AcpiOsPrintf ("Invalid AcpiName for Node %p\n", Node);
721         return (AE_OK);
722     }

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 _____

```

new/usr/src/common/acpica/components/debugger/dbstats.c

1

```
*****
17067 Fri Jan 17 07:29:49 2014
new/usr/src/common/acpica/components/debugger/dbstats.c
update to acpica-unix2-20140114
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 - 2014, 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 MERCHANTABILITY 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 #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 (
```

new/usr/src/common/acpica/components/debugger/dbstats.c

2

```
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 };
93
94 unchanged_portion_omitted
95 #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]);
232         AcpiDbEnumerateObject (ObjDesc->Device.SystemNotify);
233         AcpiDbEnumerateObject (ObjDesc->Device.DeviceNotify);
234         AcpiDbEnumerateObject (ObjDesc->Device.Handler);
235         break;

237     case ACPI_TYPE_BUFFER_FIELD:

239         if (AcpiNsGetSecondaryObject (ObjDesc))
240         {
241             AcpiGbl_ObjTypeCount [ACPI_TYPE_BUFFER_FIELD]++;
242         }
243         break;

245     case ACPI_TYPE_REGION:

247         AcpiGbl_ObjTypeCount [ACPI_TYPE_LOCAL_REGION_FIELD]++;
248         AcpiDbEnumerateObject (ObjDesc->Region.Handler);
249         break;

251     case ACPI_TYPE_POWER:

253         AcpiDbEnumerateObject (ObjDesc->PowerResource.NotifyList[0]);
254         AcpiDbEnumerateObject (ObjDesc->PowerResource.NotifyList[1]);
255         AcpiDbEnumerateObject (ObjDesc->PowerResource.SystemNotify);
256         AcpiDbEnumerateObject (ObjDesc->PowerResource.DeviceNotify);
257         break;

259     case ACPI_TYPE_PROCESSOR:

261         AcpiDbEnumerateObject (ObjDesc->Processor.NotifyList[0]);
262         AcpiDbEnumerateObject (ObjDesc->Processor.NotifyList[1]);
263         AcpiDbEnumerateObject (ObjDesc->Processor.SystemNotify);
264         AcpiDbEnumerateObject (ObjDesc->Processor.DeviceNotify);
265         AcpiDbEnumerateObject (ObjDesc->Processor.Handler);
266         break;

268     case ACPI_TYPE_THERMAL:

```

```

264         AcpiDbEnumerateObject (ObjDesc->ThermalZone.NotifyList[0]);
265         AcpiDbEnumerateObject (ObjDesc->ThermalZone.NotifyList[1]);
266         AcpiDbEnumerateObject (ObjDesc->ThermalZone.SystemNotify);
267         AcpiDbEnumerateObject (ObjDesc->ThermalZone.DeviceNotify);
268         AcpiDbEnumerateObject (ObjDesc->ThermalZone.Handler);
269         break;

271     default:

273         break;
274     }
275 }

277 /*****
278 * FUNCTION:      AcpiDbClassifyOneObject
279 * PARAMETERS:    Callback for WalkNamespace
280 * RETURN:        Status
281 * DESCRIPTION:   Enumerate both the object descriptor (including subobjects) and
282 *                the parent namespace node.
283 *****/

285 static ACPI_STATUS
286 AcpiDbClassifyOneObject (
287     ACPI_HANDLE          ObjHandle,
288     UINT32                NestingLevel,
289     void                  *Context,
290     void                  **ReturnValue)
291 {
292     ACPI_NAMESPACE_NODE *Node;
293     ACPI_OPERAND_OBJECT *ObjDesc;
294     ACPI_TYPE Type;

296     Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
297     ObjDesc = AcpiNsGetAttachedObject (Node);

299     AcpiGbl_NumNodes++;

301     Type = Node->Type;
302     if (Type > ACPI_TYPE_NS_NODE_MAX)
303     {
304         AcpiGbl_NodeTypeCountMisc++;
305     }
306     else
307     {
308         AcpiGbl_NodeTypeCount [Type]++;
309     }

311     return (AE_OK);
312 }

314 #ifdef ACPI_FUTURE_IMPLEMENTATION

316     /* TBD: These need to be counted during the initial parsing phase */

318     if (AcpiPsIsNamedOp (Op->Opcode))
319     {

```

```

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 == ACPI_TYPE_NOT_FOUND)
405     if (Temp == (UINT32) -1)
406     {
407         AcpiOsPrintf ("Invalid or unsupported argument\n");
408         return (AE_OK);
409     }

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:.. %10ld\n",
469             AcpiGbl_PsFindCount);
470         AcpiOsPrintf ("Calls to AcpiNsLookup:.....%10ld\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 ("%10s: %10ld\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_RESOURCE));
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_HANDLER));
506     AcpiOsPrintf ("AddressSpace    %3d\n", sizeof (ACPI_OBJECT_ADDRESS_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_EntryStackPointer);
540         AcpiOsPrintf ("Lowest Stack Pointer      %p\n", AcpiGbl_LowestStackPointer);
541         AcpiOsPrintf ("Stack Use                 %X (%u)\n", Temp, Temp);
542         AcpiOsPrintf ("Deepest Procedure Nesting %u\n", AcpiGbl_Deepnest);
543     #endif
544         break;

546     default:

548         break;
549 }

```

```

551     AcpiOsPrintf ("\n");
552     return (AE_OK);
553 }
_____unchanged_portion_omitted_____

```



new/usr/src/common/acpica/components/debugger/dbtest.c

1

```
*****
31953 Fri Jan 17 07:29:50 2014
new/usr/src/common/acpica/components/debugger/dbtest.c
update to acpica-unix2-20140114
*****
1 /*****
2 *
3 * Module Name: dbtest - Various debug-related tests
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acdebug.h"
47 #include "acnamesp.h"
48 #include "acpredef.h"

50 #ifdef ACPI_DEBUGGER

52 #define _COMPONENT          ACPI_CA_DEBUGGER
53 #define ACPI_MODULE_NAME    ("dbtest")

56 /* Local prototypes */

58 static void
59 AcpiDbTestAllObjects (
60     void);
```

new/usr/src/common/acpica/components/debugger/dbtest.c

2

```
62 static ACPI_STATUS
63 AcpiDbTestOneObject (
64     ACPI_HANDLE             ObjHandle,
65     UINT32                  NestingLevel,
66     void                    *Context,
67     void                    **ReturnValue);

69 static ACPI_STATUS
70 AcpiDbTestIntegerType (
71     ACPI_NAMESPACE_NODE     *Node,
72     UINT32                  BitLength);

74 static ACPI_STATUS
75 AcpiDbTestBufferType (
76     ACPI_NAMESPACE_NODE     *Node,
77     UINT32                  BitLength);

79 static ACPI_STATUS
80 AcpiDbTestStringType (
81     ACPI_NAMESPACE_NODE     *Node,
82     UINT32                  ByteLength);

84 static ACPI_STATUS
85 AcpiDbReadFromObject (
86     ACPI_NAMESPACE_NODE     *Node,
87     ACPI_OBJECT_TYPE         ExpectedType,
88     ACPI_OBJECT              **Value);

90 static ACPI_STATUS
91 AcpiDbWriteToObject (
92     ACPI_NAMESPACE_NODE     *Node,
93     ACPI_OBJECT              *Value);

95 static void
96 AcpiDbEvaluateAllPredefinedNames (
97     char                    *CountArg);

99 static ACPI_STATUS
100 AcpiDbEvaluateOnePredefinedName (
101     ACPI_HANDLE             ObjHandle,
102     UINT32                  NestingLevel,
103     void                    *Context,
104     void                    **ReturnValue);

106 /*
107 * Test subcommands
108 */
109 static ACPI_DB_ARGUMENT_INFO    AcpiDbTestTypes [] =
110 {
111     {"OBJECTS"},
112     {"PREDEFINED"},
113     {NULL}          /* Must be null terminated */
114 };

116 #define CMD_TEST_OBJECTS          0
117 #define CMD_TEST_PREDEFINED       1

119 #define BUFFER_FILL_VALUE        0xFF

121 /*
122 * Support for the special debugger read/write control methods.
123 * These methods are installed into the current namespace and are
124 * used to read and write the various namespace objects. The point
125 * is to force the AML interpreter do all of the work.
126 */
127 #define          ACPI_DB_READ_METHOD    "\\_T98"
```

```

128 #define ACPI_DB_WRITE_METHOD "\\_T99"

130 static ACPI_HANDLE ReadHandle = NULL;
131 static ACPI_HANDLE WriteHandle = NULL;

133 /* ASL Definitions of the debugger read/write control methods */

135 #if 0
136 DefinitionBlock ("ssdt.aml", "SSDT", 2, "Intel", "DEBUG", 0x00000001)
137 {
138     Method (_T98, 1, NotSerialized) /* Read */
139     {
140         Return (DeRefOf (Arg0))
141     }
142 }
143 DefinitionBlock ("ssdt2.aml", "SSDT", 2, "Intel", "DEBUG", 0x00000001)
144 {
145     Method (_T99, 2, NotSerialized) /* Write */
146     {
147         Store (Arg1, Arg0)
148     }
149 }
150 #endif

152 static unsigned char ReadMethodCode[] =
153 {
154     0x53,0x53,0x44,0x54,0x2E,0x00,0x00,0x00, /* 00000000 "SSDT..." */
155     0x02,0xC9,0x49,0x6E,0x74,0x65,0x6C,0x00, /* 00000008 "...Intel." */
156     0x44,0x45,0x42,0x55,0x47,0x00,0x00,0x00, /* 00000010 "DEBUG..." */
157     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
158     0x18,0x12,0x13,0x20,0x14,0x09,0x5F,0x54, /* 00000020 "..._T" */
159     0x39,0x38,0x01,0xA4,0x83,0x68 /* 00000028 "98...h" */
160 };

162 static unsigned char WriteMethodCode[] =
163 {
164     0x53,0x53,0x44,0x54,0x2E,0x00,0x00,0x00, /* 00000000 "SSDT..." */
165     0x02,0x15,0x49,0x6E,0x74,0x65,0x6C,0x00, /* 00000008 "...Intel." */
166     0x44,0x45,0x42,0x55,0x47,0x00,0x00,0x00, /* 00000010 "DEBUG..." */
167     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
168     0x18,0x12,0x13,0x20,0x14,0x09,0x5F,0x54, /* 00000020 "..._T" */
169     0x39,0x39,0x02,0x70,0x69,0x68 /* 00000028 "99.pih" */
170 };

173 /*****
174 *
175 * FUNCTION: AcpiDbExecuteTest
176 *
177 * PARAMETERS: TypeArg - Subcommand
178 *
179 * RETURN: None
180 *
181 * DESCRIPTION: Execute various debug tests.
182 *
183 * Note: Code is prepared for future expansion of the TEST command.
184 *
185 *****/

187 void
188 AcpiDbExecuteTest (
189     char *TypeArg)
190 {
191     UINT32 Temp;

```

```

194     AcpiUtStrupr (TypeArg);
195     Temp = AcpiDbMatchArgument (TypeArg, AcpiDbTestTypes);
196     if (Temp == ACPI_TYPE_NOT_FOUND)
197     {
198         AcpiOsPrintf ("Invalid or unsupported argument\n");
199         return;
200     }

202     switch (Temp)
203     {
204     case CMD_TEST_OBJECTS:

206         AcpiDbTestAllObjects ();
207         break;

209     case CMD_TEST_PREDEFINED:

211         AcpiDbEvaluateAllPredefinedNames (NULL);
212         break;

214     default:
215         break;
216     }
217 }

220 /*****
221 *
222 * FUNCTION: AcpiDbTestAllObjects
223 *
224 * PARAMETERS: None
225 *
226 * RETURN: None
227 *
228 * DESCRIPTION: This test implements the OBJECTS subcommand. It exercises the
229 *               namespace by reading/writing/comparing all data objects such
230 *               as integers, strings, buffers, fields, buffer fields, etc.
231 *
232 *****/

234 static void
235 AcpiDbTestAllObjects (
236     void)
237 {
238     ACPI_STATUS Status;

241     /* Install the debugger read-object control method if necessary */

243     if (!ReadHandle)
244     {
245         Status = AcpiInstallMethod (ReadMethodCode);
246         if (ACPI_FAILURE (Status))
247         {
248             AcpiOsPrintf ("%s, Could not install debugger read method\n",
249                 AcpiFormatException (Status));
250             return;
251         }

253         Status = AcpiGetHandle (NULL, ACPI_DB_READ_METHOD, &ReadHandle);
254         if (ACPI_FAILURE (Status))
255         {
256             AcpiOsPrintf ("Could not obtain handle for debug method %s\n",
257                 ACPI_DB_READ_METHOD);
258             return;
259         }

```

```

260     }

262     /* Install the debugger write-object control method if necessary */

264     if (!WriteHandle)
265     {
266         Status = AcpiInstallMethod (WriteMethodCode);
267         if (ACPI_FAILURE (Status))
268         {
269             AcpiOsPrintf ("%s, Could not install debugger write method\n",
270                 AcpiFormatException (Status));
271             return;
272         }

274         Status = AcpiGetHandle (NULL, ACPI_DB_WRITE_METHOD, &WriteHandle);
275         if (ACPI_FAILURE (Status))
276         {
277             AcpiOsPrintf ("Could not obtain handle for debug method %s\n",
278                 ACPI_DB_WRITE_METHOD);
279             return;
280         }
281     }

283     /* Walk the entire namespace, testing each supported named data object */

285     (void) AcpiWalkNamespace (ACPI_TYPE_ANY, ACPI_ROOT_OBJECT,
286         ACPI_UINT32_MAX, AcpiDbTestOneObject, NULL, NULL, NULL);
287 }

290 /*****
291  *
292  * FUNCTION:      AcpiDbTestOneObject
293  *
294  * PARAMETERS:    ACPI_WALK_CALLBACK
295  *
296  * RETURN:        Status
297  *
298  * DESCRIPTION:   Test one namespace object. Supported types are Integer,
299  *               String, Buffer, BufferField, and FieldUnit. All other object
300  *               types are simply ignored.
301  *
302  *               Note: Support for Packages is not implemented.
303  *
304  *****/

306 static ACPI_STATUS
307 AcpiDbTestOneObject (
308     ACPI_HANDLE          ObjHandle,
309     UINT32               NestingLevel,
310     void                 *Context,
311     void                 **ReturnValue)
312 {
313     ACPI_NAMESPACE_NODE *Node;
314     ACPI_OPERAND_OBJECT *ObjDesc;
315     ACPI_OPERAND_OBJECT *RegionObj;
316     ACPI_OBJECT_TYPE LocalType;
317     UINT32 BitLength = 0;
318     UINT32 ByteLength = 0;
319     ACPI_STATUS Status = AE_OK;

322     Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, ObjHandle);
323     ObjDesc = Node->Object;

325     /*

```

```

326     * For the supported types, get the actual bit length or
327     * byte length. Map the type to one of Integer/String/Buffer.
328     */
329     switch (Node->Type)
330     {
331     case ACPI_TYPE_INTEGER:

333         /* Integer width is either 32 or 64 */

335         LocalType = ACPI_TYPE_INTEGER;
336         BitLength = AcpiGbl_IntegerBitWidth;
337         break;

339     case ACPI_TYPE_STRING:

341         LocalType = ACPI_TYPE_STRING;
342         ByteLength = ObjDesc->String.Length;
343         break;

345     case ACPI_TYPE_BUFFER:

347         LocalType = ACPI_TYPE_BUFFER;
348         ByteLength = ObjDesc->Buffer.Length;
349         BitLength = ByteLength * 8;
350         break;

352     case ACPI_TYPE_FIELD_UNIT:
353     case ACPI_TYPE_BUFFER_FIELD:
354     case ACPI_TYPE_LOCAL_REGION_FIELD:
355     case ACPI_TYPE_LOCAL_INDEX_FIELD:
356     case ACPI_TYPE_LOCAL_BANK_FIELD:

358         LocalType = ACPI_TYPE_INTEGER;
359         if (ObjDesc)
360         {
361             /*
362              * Returned object will be a Buffer if the field length
363              * is larger than the size of an Integer (32 or 64 bits
364              * depending on the DSDT version).
365              */
366             BitLength = ObjDesc->CommonField.BitLength;
367             ByteLength = ACPI_ROUND_BITS_UP_TO_BYTES (BitLength);
368             if (BitLength > AcpiGbl_IntegerBitWidth)
369             {
370                 LocalType = ACPI_TYPE_BUFFER;
371             }
372         }
373         break;

375     default:

377         /* Ignore all other types */

379         return (AE_OK);
380     }

382     /* Emit the common prefix: Type:Name */

384     AcpiOsPrintf ("%14s: %4.4s",
385         AcpiUtGetTypeName (Node->Type), Node->Name.Ascii);
386     if (!ObjDesc)
387     {
388         AcpiOsPrintf (" Ignoring, no attached object\n");
389         return (AE_OK);
390     }

```

```

392  /*
393  * Check for unsupported region types. Note: AcpiExec simulates
394  * access to SystemMemory, SystemIO, PCI_Config, and EC.
395  */
396  switch (Node->Type)
397  {
398  case ACPI_TYPE_LOCAL_REGION_FIELD:

400      RegionObj = ObjDesc->Field.RegionObj;
401      switch (RegionObj->Region.SpaceId)
402      {
403      case ACPI_ADR_SPACE_SYSTEM_MEMORY:
404      case ACPI_ADR_SPACE_SYSTEM_IO:
405      case ACPI_ADR_SPACE_PCI_CONFIG:
406      case ACPI_ADR_SPACE_EC:

408          break;

410      default:

412          AcpiOsPrintf ("      %s space is not supported [%4.4s]\n",
413              AcpiUtGetRegionName (RegionObj->Region.SpaceId),
414              RegionObj->Region.Node->Name.Ascii);
415          return (AE_OK);
416      }
417      break;

419  default:
420      break;
421  }

423  /* At this point, we have resolved the object to one of the major types */

425  switch (LocalType)
426  {
427  case ACPI_TYPE_INTEGER:

429      Status = AcpiDbTestIntegerType (Node, BitLength);
430      break;

432  case ACPI_TYPE_STRING:

434      Status = AcpiDbTestStringType (Node, ByteLength);
435      break;

437  case ACPI_TYPE_BUFFER:

439      Status = AcpiDbTestBufferType (Node, BitLength);
440      break;

442  default:

444      AcpiOsPrintf (" Ignoring, type not implemented (%2.2X)",
445          LocalType);
446      break;
447  }

449  switch (Node->Type)
450  {
451  case ACPI_TYPE_LOCAL_REGION_FIELD:

453      RegionObj = ObjDesc->Field.RegionObj;
454      AcpiOsPrintf (" (%s)",
455          AcpiUtGetRegionName (RegionObj->Region.SpaceId));
456      break;

```

```

458  default:
459      break;
460  }

462  AcpiOsPrintf ("\n");
463  return (Status);
464  }

467  /*****
468  *
469  * FUNCTION:      AcpiDbTestIntegerType
470  *
471  * PARAMETERS:    Node          - Parent NS node for the object
472  *                BitLength     - Actual length of the object. Used for
473  *                               support of arbitrary length FieldUnit
474  *                               and BufferField objects.
475  *
476  * RETURN:        Status
477  *
478  * DESCRIPTION:   Test read/write for an Integer-valued object. Performs a
479  *               write/read/compare of an arbitrary new value, then performs
480  *               a write/read/compare of the original value.
481  *
482  *****/

484  static ACPI_STATUS
485  AcpiDbTestIntegerType (
486      ACPI_NAMESPACE_NODE *Node,
487      UINT32 BitLength)
488  {
489      ACPI_OBJECT *Temp1 = NULL;
490      ACPI_OBJECT *Temp2 = NULL;
491      ACPI_OBJECT *Temp3 = NULL;
492      WriteValue;
493      ValueToWrite;
494      ACPI_STATUS Status;

497      if (BitLength > 64)
498      {
499          AcpiOsPrintf (" Invalid length for an Integer: %u", BitLength);
500          return (AE_OK);
501      }

503      /* Read the original value */

505      Status = AcpiDbReadFromObject (Node, ACPI_TYPE_INTEGER, &Temp1);
506      if (ACPI_FAILURE (Status))
507      {
508          return (Status);
509      }

511      AcpiOsPrintf (" (%4.4X/%3.3X) %8.8X%8.8X",
512          BitLength, ACPI_ROUND_BITS_UP_TO_BYTES (BitLength),
513          ACPI_FORMAT_UINT64 (Temp1->Integer.Value));

515      ValueToWrite = ACPI_UINT64_MAX >> (64 - BitLength);
516      if (Temp1->Integer.Value == ValueToWrite)
517      {
518          ValueToWrite = 0;
519      }

521      /* Write a new value */

523      WriteValue.Type = ACPI_TYPE_INTEGER;

```

```

524 WriteValue.Integer.Value = ValueToWrite;
525 Status = AcpiDbWriteToObject (Node, &WriteValue);
526 if (ACPI_FAILURE (Status))
527 {
528     goto Exit;
529 }

531 /* Ensure that we can read back the new value */

533 Status = AcpiDbReadFromObject (Node, ACPI_TYPE_INTEGER, &Temp2);
534 if (ACPI_FAILURE (Status))
535 {
536     goto Exit;
537 }

539 if (Temp2->Integer.Value != ValueToWrite)
540 {
541     AcpiOsPrintf (" MISMATCH 2: %8.8X%8.8X, expecting %8.8X%8.8X",
542         ACPI_FORMAT_UINT64 (Temp2->Integer.Value),
543         ACPI_FORMAT_UINT64 (ValueToWrite));
544 }

546 /* Write back the original value */

548 WriteValue.Integer.Value = Temp1->Integer.Value;
549 Status = AcpiDbWriteToObject (Node, &WriteValue);
550 if (ACPI_FAILURE (Status))
551 {
552     goto Exit;
553 }

555 /* Ensure that we can read back the original value */

557 Status = AcpiDbReadFromObject (Node, ACPI_TYPE_INTEGER, &Temp3);
558 if (ACPI_FAILURE (Status))
559 {
560     goto Exit;
561 }

563 if (Temp3->Integer.Value != Temp1->Integer.Value)
564 {
565     AcpiOsPrintf (" MISMATCH 3: %8.8X%8.8X, expecting %8.8X%8.8X",
566         ACPI_FORMAT_UINT64 (Temp3->Integer.Value),
567         ACPI_FORMAT_UINT64 (Temp1->Integer.Value));
568 }

570 Exit:
571 if (Temp1) {AcpiOsFree (Temp1);}
572 if (Temp2) {AcpiOsFree (Temp2);}
573 if (Temp3) {AcpiOsFree (Temp3);}
574 return (AE_OK);
575 }

578 /*****
579 *
580 * FUNCTION:    AcpiDbTestBufferType
581 *
582 * PARAMETERS:  Node           - Parent NS node for the object
583 *              BitLength      - Actual length of the object.
584 *
585 * RETURN:      Status
586 *
587 * DESCRIPTION: Test read/write for an Buffer-valued object. Performs a
588 *              write/read/compare of an arbitrary new value, then performs
589 *              a write/read/compare of the original value.

```

```

590 *
591 *****/

593 static ACPI_STATUS
594 AcpiDbTestBufferType (
595     ACPI_NAMESPACE_NODE *Node,
596     UINT32 BitLength)
597 {
598     ACPI_OBJECT *Temp1 = NULL;
599     ACPI_OBJECT *Temp2 = NULL;
600     ACPI_OBJECT *Temp3 = NULL;
601     UINT8 *Buffer;
602     ACPI_OBJECT WriteValue;
603     ACPI_STATUS Status;
604     UINT32 ByteLength;
605     UINT32 i;
606     UINT8 ExtraBits;

609     ByteLength = ACPI_ROUND_BITS_UP_TO_BYTES (BitLength);
610     if (ByteLength == 0)
611     {
612         AcpiOsPrintf (" Ignoring zero length buffer");
613         return (AE_OK);
614     }

616     /* Allocate a local buffer */

618     Buffer = ACPI_ALLOCATE_ZEROED (ByteLength);
619     if (!Buffer)
620     {
621         return (AE_NO_MEMORY);
622     }

624     /* Read the original value */

626     Status = AcpiDbReadFromObject (Node, ACPI_TYPE_BUFFER, &Temp1);
627     if (ACPI_FAILURE (Status))
628     {
629         goto Exit;
630     }

632     /* Emit a few bytes of the buffer */

634     AcpiOsPrintf (" (%4.4X/%3.3X)", BitLength, Temp1->Buffer.Length);
635     for (i = 0; ((i < 4) && (i < ByteLength)); i++)
636     {
637         AcpiOsPrintf (" %2.2X", Temp1->Buffer.Pointer[i]);
638     }
639     AcpiOsPrintf ("... ");

641     /*
642     * Write a new value.
643     *
644     * Handle possible extra bits at the end of the buffer. Can
645     * happen for FieldUnits larger than an integer, but the bit
646     * count is not an integral number of bytes. Zero out the
647     * unused bits.
648     */
649     ACPI_MEMSET (Buffer, BUFFER_FILL_VALUE, ByteLength);
650     ExtraBits = BitLength % 8;
651     if (ExtraBits)
652     {
653         Buffer [ByteLength - 1] = ACPI_MASK_BITS_ABOVE (ExtraBits);
654     }

```

```

656 WriteValue.Type = ACPI_TYPE_BUFFER;
657 WriteValue.Buffer.Length = ByteLength;
658 WriteValue.Buffer.Pointer = Buffer;

660 Status = AcpiDbWriteToObject (Node, &WriteValue);
661 if (ACPI_FAILURE (Status))
662 {
663     goto Exit;
664 }

666 /* Ensure that we can read back the new value */

668 Status = AcpiDbReadFromObject (Node, ACPI_TYPE_BUFFER, &Temp2);
669 if (ACPI_FAILURE (Status))
670 {
671     goto Exit;
672 }

674 if (ACPI_MEMCMP (Temp2->Buffer.Pointer, Buffer, ByteLength))
675 {
676     AcpiOsPrintf (" MISMATCH 2: New buffer value");
677 }

679 /* Write back the original value */

681 WriteValue.Buffer.Length = ByteLength;
682 WriteValue.Buffer.Pointer = Temp1->Buffer.Pointer;

684 Status = AcpiDbWriteToObject (Node, &WriteValue);
685 if (ACPI_FAILURE (Status))
686 {
687     goto Exit;
688 }

690 /* Ensure that we can read back the original value */

692 Status = AcpiDbReadFromObject (Node, ACPI_TYPE_BUFFER, &Temp3);
693 if (ACPI_FAILURE (Status))
694 {
695     goto Exit;
696 }

698 if (ACPI_MEMCMP (Temp1->Buffer.Pointer, Temp3->Buffer.Pointer, ByteLength))
699 {
700     AcpiOsPrintf (" MISMATCH 3: While restoring original buffer");
701 }

703 Exit:
704     ACPI_FREE (Buffer);
705     if (Temp1) {AcpiOsFree (Temp1);}
706     if (Temp2) {AcpiOsFree (Temp2);}
707     if (Temp3) {AcpiOsFree (Temp3);}
708     return (Status);
709 }

712 /*****
713 *
714 * FUNCTION:      AcpiDbTestStringType
715 *
716 * PARAMETERS:    Node          - Parent NS node for the object
717 *                ByteLength    - Actual length of the object.
718 *
719 * RETURN:        Status
720 *
721 * DESCRIPTION:   Test read/write for an String-valued object. Performs a

```

```

722 *                write/read/compare of an arbitrary new value, then performs
723 *                a write/read/compare of the original value.
724 *
725 *****/

727 static ACPI_STATUS
728 AcpiDbTestStringType (
729     ACPI_NAMESPACE_NODE *Node,
730     UINT32 ByteLength)
731 {
732     ACPI_OBJECT *Temp1 = NULL;
733     ACPI_OBJECT *Temp2 = NULL;
734     ACPI_OBJECT *Temp3 = NULL;
735     char *ValueToWrite = "Test String from AML Debugger";
736     ACPI_OBJECT WriteValue;
737     ACPI_STATUS Status;

740 /* Read the original value */

742 Status = AcpiDbReadFromObject (Node, ACPI_TYPE_STRING, &Temp1);
743 if (ACPI_FAILURE (Status))
744 {
745     return (Status);
746 }

748 AcpiOsPrintf (" (%4.4X/%3.3X) \"%s\"", (Temp1->String.Length * 8),
749     Temp1->String.Length, Temp1->String.Pointer);

751 /* Write a new value */

753 WriteValue.Type = ACPI_TYPE_STRING;
754 WriteValue.String.Length = ACPI_STRLEN (ValueToWrite);
755 WriteValue.String.Pointer = ValueToWrite;

757 Status = AcpiDbWriteToObject (Node, &WriteValue);
758 if (ACPI_FAILURE (Status))
759 {
760     goto Exit;
761 }

763 /* Ensure that we can read back the new value */

765 Status = AcpiDbReadFromObject (Node, ACPI_TYPE_STRING, &Temp2);
766 if (ACPI_FAILURE (Status))
767 {
768     goto Exit;
769 }

771 if (ACPI_STRCMP (Temp2->String.Pointer, ValueToWrite))
772 {
773     AcpiOsPrintf (" MISMATCH 2: %s, expecting %s",
774         Temp2->String.Pointer, ValueToWrite);
775 }

777 /* Write back the original value */

779 WriteValue.String.Length = ACPI_STRLEN (Temp1->String.Pointer);
780 WriteValue.String.Pointer = Temp1->String.Pointer;

782 Status = AcpiDbWriteToObject (Node, &WriteValue);
783 if (ACPI_FAILURE (Status))
784 {
785     goto Exit;
786 }

```

```

788  /* Ensure that we can read back the original value */
790  Status = AcpiDbReadFromObject (Node, ACPI_TYPE_STRING, &Temp3);
791  if (ACPI_FAILURE (Status))
792  {
793      goto Exit;
794  }

796  if (ACPI_STRCMP (Temp1->String.Pointer, Temp3->String.Pointer))
797  {
798      AcpiOsPrintf (" MISMATCH 3: %s, expecting %s",
799                  Temp3->String.Pointer, Temp1->String.Pointer);
800  }

802 Exit:
803     if (Temp1) {AcpiOsFree (Temp1);}
804     if (Temp2) {AcpiOsFree (Temp2);}
805     if (Temp3) {AcpiOsFree (Temp3);}
806     return (Status);
807 }

810 /*****
811  *
812  * FUNCTION:      AcpiDbReadFromObject
813  *
814  * PARAMETERS:    Node          - Parent NS node for the object
815  *                ExpectedType  - Object type expected from the read
816  *                Value         - Where the value read is returned
817  *
818  * RETURN:        Status
819  *
820  * DESCRIPTION:   Performs a read from the specified object by invoking the
821  *                special debugger control method that reads the object. Thus,
822  *                the AML interpreter is doing all of the work, increasing the
823  *                validity of the test.
824  *
825  *****/

827 static ACPI_STATUS
828 AcpiDbReadFromObject (
829     ACPI_NAMESPACE_NODE *Node,
830     ACPI_OBJECT_TYPE ExpectedType,
831     ACPI_OBJECT **Value)
832 {
833     ACPI_OBJECT *RetVal;
834     ACPI_OBJECT_LIST ParamObjects;
835     ACPI_OBJECT Params[2];
836     ACPI_BUFFER ReturnObj;
837     ACPI_STATUS Status;

840     Params[0].Type = ACPI_TYPE_LOCAL_REFERENCE;
841     Params[0].Reference.ActualType = Node->Type;
842     Params[0].Reference.Handle = ACPI_CAST_PTR (ACPI_HANDLE, Node);

844     ParamObjects.Count = 1;
845     ParamObjects.Pointer = Params;

847     ReturnObj.Length = ACPI_ALLOCATE_BUFFER;

849     AcpiGbl_MethodExecuting = TRUE;
850     Status = AcpiEvaluateObject (ReadHandle, NULL, &ParamObjects, &ReturnObj);
851     AcpiGbl_MethodExecuting = FALSE;

853     if (ACPI_FAILURE (Status))

```

```

854     {
855         AcpiOsPrintf ("Could not read from object, %s",
856                     AcpiFormatException (Status));
857         return (Status);
858     }

860     RetValue = (ACPI_OBJECT *) ReturnObj.Pointer;

862     switch (RetVal->Type)
863     {
864     case ACPI_TYPE_INTEGER:
865     case ACPI_TYPE_BUFFER:
866     case ACPI_TYPE_STRING:
867         /*
868          * Did we receive the type we wanted? Most important for the
869          * Integer/Buffer case (when a field is larger than an Integer,
870          * it should return a Buffer).
871          */
872         if (RetVal->Type != ExpectedType)
873         {
874             AcpiOsPrintf (" Type mismatch: Expected %s, Received %s",
875                         AcpiUtGetTypeName (ExpectedType),
876                         AcpiUtGetTypeName (RetVal->Type));

878             return (AE_TYPE);
879         }

881         *Value = RetValue;
882         break;

884     default:

886         AcpiOsPrintf (" Unsupported return object type, %s",
887                     AcpiUtGetTypeName (RetVal->Type));
888         AcpiOsFree (ReturnObj.Pointer);

890         return (AE_TYPE);
891     }

893     return (Status);
894 }

897 /*****
898  *
899  * FUNCTION:      AcpiDbWriteToObject
900  *
901  * PARAMETERS:    Node          - Parent NS node for the object
902  *                Value         - Value to be written
903  *
904  * RETURN:        Status
905  *
906  * DESCRIPTION:   Performs a write to the specified object by invoking the
907  *                special debugger control method that writes the object. Thus,
908  *                the AML interpreter is doing all of the work, increasing the
909  *                validity of the test.
910  *
911  *****/

913 static ACPI_STATUS
914 AcpiDbWriteToObject (
915     ACPI_NAMESPACE_NODE *Node,
916     ACPI_OBJECT *Value)
917 {
918     ACPI_OBJECT_LIST ParamObjects;
919     ACPI_OBJECT Params[2];

```

```

920     ACPI_STATUS      Status;

923     Params[0].Type = ACPI_TYPE_LOCAL_REFERENCE;
924     Params[0].Reference.ActualType = Node->Type;
925     Params[0].Reference.Handle = ACPI_CAST_PTR (ACPI_HANDLE, Node);

927     /* Copy the incoming user parameter */

929     ACPI_MEMCPY (&Params[1], Value, sizeof (ACPI_OBJECT));

931     ParamObjects.Count = 2;
932     ParamObjects.Pointer = Params;

934     AcpiGbl_MethodExecuting = TRUE;
935     Status = AcpiEvaluateObject (WriteHandle, NULL, &ParamObjects, NULL);
936     AcpiGbl_MethodExecuting = FALSE;

938     if (ACPI_FAILURE (Status))
939     {
940         AcpiOsPrintf ("Could not write to object, %s",
941             AcpiFormatException (Status));
942     }

944     return (Status);
945 }

948 /*****
949  *
950  * FUNCTION:      AcpiDbEvaluateAllPredefinedNames
951  *
952  * PARAMETERS:    CountArg          - Max number of methods to execute
953  *
954  * RETURN:        None
955  *
956  * DESCRIPTION:   Namespace batch execution. Execute predefined names in the
957  *               namespace, up to the max count, if specified.
958  *
959  *****/

961 static void
962 AcpiDbEvaluateAllPredefinedNames (
963     char          *CountArg)
964 {
965     ACPI_DB_EXECUTE_WALK    Info;

968     Info.Count = 0;
969     Info.MaxCount = ACPI_UINT32_MAX;

971     if (CountArg)
972     {
973         Info.MaxCount = ACPI_STRTOUL (CountArg, NULL, 0);
974     }

976     /* Search all nodes in namespace */

978     (void) AcpiWalkNamespace (ACPI_TYPE_ANY, ACPI_ROOT_OBJECT, ACPI_UINT32_MAX,
979         AcpiDbEvaluateOnePredefinedName, NULL, (void *) &Info, NULL);

981     AcpiOsPrintf ("Evaluated %u predefined names in the namespace\n", Info.Count
982 }

985 /*****

```

```

986  *
987  * FUNCTION:      AcpiDbEvaluateOnePredefinedName
988  *
989  * PARAMETERS:    Callback from WalkNamespace
990  *
991  * RETURN:        Status
992  *
993  * DESCRIPTION:   Batch execution module. Currently only executes predefined
994  *               ACPI names.
995  *
996  *****/

998 static ACPI_STATUS
999 AcpiDbEvaluateOnePredefinedName (
1000     ACPI_HANDLE      ObjHandle,
1001     UINT32            NestingLevel,
1002     void              *Context,
1003     void              **ReturnValue)
1004 {
1005     ACPI_NAMESPACE_NODE *Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
1006     ACPI_DB_EXECUTE_WALK *Info = (ACPI_DB_EXECUTE_WALK *) Context;
1007     char                  *Pathname;
1008     const ACPI_PREDEFINED_INFO *Predefined;
1009     ACPI_DEVICE_INFO      *ObjInfo;
1010     ACPI_OBJECT_LIST      ParamObjects;
1011     ACPI_OBJECT           *ThisParam;
1012     ACPI_BUFFER            ReturnObj;
1013     ACPI_STATUS            Status;
1014     UINT16                 ArgTypeList;
1015     UINT8                  ArgCount;
1016     UINT8                  ArgType;
1017     i;

1021     /* The name must be a predefined ACPI name */

1023     Predefined = AcpiUtMatchPredefinedMethod (Node->Name.Ascii);
1024     if (!Predefined)
1025     {
1026         return (AE_OK);
1027     }

1029     if (Node->Type == ACPI_TYPE_LOCAL_SCOPE)
1030     {
1031         return (AE_OK);
1032     }

1034     Pathname = AcpiNsGetExternalPathname (Node);
1035     if (!Pathname)
1036     {
1037         return (AE_OK);
1038     }

1040     /* Get the object info for number of method parameters */

1042     Status = AcpiGetObjectInfo (ObjHandle, &ObjInfo);
1043     if (ACPI_FAILURE (Status))
1044     {
1045         return (Status);
1046     }

1048     ParamObjects.Count = 0;
1049     ParamObjects.Pointer = NULL;

1051     if (ObjInfo->Type == ACPI_TYPE_METHOD)

```



```

1052 {
1053     /* Setup default parameters (with proper types) */
1054
1055     ArgTypeList = Predefined->Info.ArgumentList;
1056     ArgCount = METHOD_GET_ARG_COUNT (ArgTypeList);
1057
1058     /*
1059      * Setup the ACPI-required number of arguments, regardless of what
1060      * the actual method defines. If there is a difference, then the
1061      * method is wrong and a warning will be issued during execution.
1062      */
1063     ThisParam = Params;
1064     for (i = 0; i < ArgCount; i++)
1065     {
1066         ArgType = METHOD_GET_NEXT_TYPE (ArgTypeList);
1067         ThisParam->Type = ArgType;
1068
1069         switch (ArgType)
1070         {
1071             case ACPI_TYPE_INTEGER:
1072
1073                 ThisParam->Integer.Value = 1;
1074                 break;
1075
1076             case ACPI_TYPE_STRING:
1077
1078                 ThisParam->String.Pointer = "This is the default argument string
1079                 ThisParam->String.Length = ACPI_STRLen (ThisParam->String.Pointer);
1080                 break;
1081
1082             case ACPI_TYPE_BUFFER:
1083
1084                 ThisParam->Buffer.Pointer = (UINT8 *) Params; /* just a garbage
1085                 ThisParam->Buffer.Length = 48;
1086                 break;
1087
1088             case ACPI_TYPE_PACKAGE:
1089
1090                 ThisParam->Package.Elements = NULL;
1091                 ThisParam->Package.Count = 0;
1092                 break;
1093
1094             default:
1095
1096                 AcpiOsPrintf ("%s: Unsupported argument type: %u\n",
1097                 Pathname, ArgType);
1098                 break;
1099             }
1100
1101             ThisParam++;
1102         }
1103
1104         ParamObjects.Count = ArgCount;
1105         ParamObjects.Pointer = Params;
1106     }
1107
1108     ACPI_FREE (ObjInfo);
1109     ReturnObj.Pointer = NULL;
1110     ReturnObj.Length = ACPI_ALLOCATE_BUFFER;
1111
1112     /* Do the actual method execution */
1113
1114     AcpiGbl_MethodExecuting = TRUE;
1115
1116     Status = AcpiEvaluateObject (Node, NULL, &ParamObjects, &ReturnObj);

```

```

1118     AcpiOsPrintf ("%32s returned %s\n", Pathname, AcpiFormatException (Status))
1119     AcpiGbl_MethodExecuting = FALSE;
1120     ACPI_FREE (Pathname);
1121
1122     /* Ignore status from method execution */
1123
1124     Status = AE_OK;
1125
1126     /* Update count, check if we have executed enough methods */
1127
1128     Info->Count++;
1129     if (Info->Count >= Info->MaxCount)
1130     {
1131         Status = AE_CTRL_TERMINATE;
1132     }
1133
1134     return (Status);
1135 }
1136
1137 #endif /* ACPI_DEBUGGER */

```

new/usr/src/common/acpica/components/debugger/dbutils.c

1

```
*****
13449 Fri Jan 17 07:29:50 2014
new/usr/src/common/acpica/components/debugger/dbutils.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dbutils - AML debugger utilities
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "acnamesp.h"
48 #include "acdebug.h"
49 #include "acdisasm.h"

52 #ifdef ACPI_DEBUGGER

54 #define _COMPONENT ACPI_CA_DEBUGGER
55     ACPI_MODULE_NAME ("dbutils")

57 /* Local prototypes */
```

new/usr/src/common/acpica/components/debugger/dbutils.c

2

```
59 #ifdef ACPI_OBSOLETE_FUNCTIONS
60 ACPI_STATUS
61 AcpiDbSecondPassParse (
62     ACPI_PARSE_OBJECT *Root);

64 void
65 AcpiDbDumpBuffer (
66     UINT32 Address);
67 #endif

69 static char *Converter = "0123456789ABCDEF";

72 /*****
73 *
74 * FUNCTION: AcpiDbMatchArgument
75 *
76 * PARAMETERS: UserArgument - User command line
77 *             Arguments - Array of commands to match against
78 *
79 * RETURN: Index into command array or ACPI_TYPE_NOT_FOUND if not found
80 *
81 * DESCRIPTION: Search command array for a command match
82 *
83 *****/

85 ACPI_OBJECT_TYPE
86 AcpiDbMatchArgument (
87     char *UserArgument,
88     ACPI_DB_ARGUMENT_INFO *Arguments)
89 {
90     UINT32 i;

93     if (!UserArgument || UserArgument[0] == 0)
94     {
95         return (ACPI_TYPE_NOT_FOUND);
96     }

98     for (i = 0; Arguments[i].Name; i++)
99     {
100         if (ACPI_STRSTR (Arguments[i].Name, UserArgument) == Arguments[i].Name)
101         {
102             return (i);
103         }
104     }

106     /* Argument not recognized */

108     return (ACPI_TYPE_NOT_FOUND);
109 }

unchanged_portion_omitted

143 /*****
144 *
145 * FUNCTION: AcpiDbDumpExternalObject
146 *
147 * PARAMETERS: ObjDesc - External ACPI object to dump
148 *             Level - Nesting level.
149 *
150 * RETURN: None
151 *
152 * DESCRIPTION: Dump the contents of an ACPI external object
153 */
```

```

154  *****/
155 void
156 AcpiDbDumpExternalObject (
157     ACPI_OBJECT      *ObjDesc,
158     UINT32            Level)
159 {
160     UINT32            i;
161
162     if (!ObjDesc)
163     {
164         AcpiOsPrintf ("[Null Object]\n");
165         return;
166     }
167
168     for (i = 0; i < Level; i++)
169     {
170         AcpiOsPrintf (" ");
171     }
172
173     switch (ObjDesc->Type)
174     {
175     case ACPI_TYPE_ANY:
176
177         AcpiOsPrintf ("[Null Object] (Type=0)\n");
178         break;
179
180     case ACPI_TYPE_INTEGER:
181
182         AcpiOsPrintf ("[Integer] = %8.8X%8.8X\n",
183             ACPI_FORMAT_UINT64 (ObjDesc->Integer.Value));
184         break;
185
186     case ACPI_TYPE_STRING:
187
188         AcpiOsPrintf ("[String] Length %.2X = ", ObjDesc->String.Length);
189         AcpiUtPrintString (ObjDesc->String.Pointer, ACPI_UINT8_MAX);
190         for (i = 0; i < ObjDesc->String.Length; i++)
191         {
192             AcpiOsPrintf ("%c", ObjDesc->String.Pointer[i]);
193         }
194         AcpiOsPrintf ("\n");
195         break;
196
197     case ACPI_TYPE_BUFFER:
198
199         AcpiOsPrintf ("[Buffer] Length %.2X = ", ObjDesc->Buffer.Length);
200         if (ObjDesc->Buffer.Length)
201         {
202             if (ObjDesc->Buffer.Length > 16)
203             {
204                 AcpiOsPrintf ("\n");
205                 AcpiUtDebugDumpBuffer (ACPI_CAST_PTR (UINT8, ObjDesc->Buffer.Pointer),
206                     ObjDesc->Buffer.Length, DB_BYTE_DISPLAY, _COMPONENT);
207                 AcpiUtDumpBuffer (ACPI_CAST_PTR (UINT8, ObjDesc->Buffer.Pointer),
208                     ObjDesc->Buffer.Length, DB_DWORD_DISPLAY, _COMPONENT);
209             }
210             else
211             {
212                 AcpiOsPrintf ("\n");
213             }
214         }
215     }

```

```

211     break;
212
213     case ACPI_TYPE_PACKAGE:
214
215         AcpiOsPrintf ("[Package] Contains %u Elements:\n",
216             ObjDesc->Package.Count);
217
218         for (i = 0; i < ObjDesc->Package.Count; i++)
219         {
220             AcpiDbDumpExternalObject (&ObjDesc->Package.Elements[i], Level+1);
221         }
222         break;
223
224     case ACPI_TYPE_LOCAL_REFERENCE:
225
226         AcpiOsPrintf ("[Object Reference] = ");
227         AcpiDmDisplayInternalObject (ObjDesc->Reference.Handle, NULL);
228         break;
229
230     case ACPI_TYPE_PROCESSOR:
231
232         AcpiOsPrintf ("[Processor]\n");
233         break;
234
235     case ACPI_TYPE_POWER:
236
237         AcpiOsPrintf ("[Power Resource]\n");
238         break;
239
240     default:
241
242         AcpiOsPrintf ("[Unknown Type] %X\n", ObjDesc->Type);
243         break;
244     }
245 }
246
247
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  *****/
259
260 void
261 AcpiDbPrepNamestring (
262     char              *Name)
263 {
264     if (!Name)
265     {
266         return;
267     }
268
269     AcpiUtStrupr (Name);

```

```
new/usr/src/common/acpica/components/debugger/dbutils.c
```

```

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 {
504
505     AcpiOsPrintf ("\nLocation %X:\n", Address);
506
507     AcpiDbgLevel |= ACPI_LV_TABLES;
508     AcpiUtDebugDumpBuffer (ACPI_TO_POINTER (Address), 64, DB_BYTE_DISPLAY,
509         AcpiUtDumpBuffer (ACPI_TO_POINTER (Address), 64, DB_BYTE_DISPLAY,
510             ACPI_UINT32_MAX);
511 }
512 #endif
513 #endif /* ACPI_DEBUGGER */

```

new/usr/src/common/acpica/components/debugger/dbxface.c

1

```
*****
15621 Fri Jan 17 07:29:51 2014
new/usr/src/common/acpica/components/debugger/dbxface.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 / *****
2 *
3 * Module Name: dbxface - AML Debugger external interfaces
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "amlcode.h"
48 #include "acdebug.h"
49 #include "acdisasm.h"

52 #ifdef ACPI_DEBUGGER

54 #define _COMPONENT ACPI_CA_DEBUGGER
55 #define ACPI_MODULE_NAME ("dbxface")
```

new/usr/src/common/acpica/components/debugger/dbxface.c

2

```
58 /* Local prototypes */

60 static ACPI_STATUS
61 AcpiDbStartCommand (
62     ACPI_WALK_STATE      *WalkState,
63     ACPI_PARSE_OBJECT     *Op);

65 #ifdef ACPI_OBSOLETE_FUNCTIONS
66 void
67 AcpiDbMethodEnd (
68     ACPI_WALK_STATE      *WalkState);
69 #endif

72 /*****
73 *
74 * FUNCTION:      AcpiDbStartCommand
75 *
76 * PARAMETERS:    WalkState      - Current walk
77 *                Op              - Current executing Op, from AML interpreter
78 *
79 * RETURN:        Status
80 *
81 * DESCRIPTION:   Enter debugger command loop
82 *
83 *****/

85 static ACPI_STATUS
86 AcpiDbStartCommand (
87     ACPI_WALK_STATE      *WalkState,
88     ACPI_PARSE_OBJECT     *Op)
89 {
90     ACPI_STATUS          Status;

93     /* TBD: [Investigate] are there namespace locking issues here? */

95     /* AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE); */

97     /* Go into the command loop and await next user command */

100     AcpiGbl_MethodExecuting = TRUE;
101     Status = AE_CTRL_TRUE;
102     while (Status == AE_CTRL_TRUE)
103     {
104         if (AcpiGbl_DebuggerConfiguration == DEBUGGER_MULTI_THREADED)
105         {
106             /* Handshake with the front-end that gets user command lines */

108             Status = AcpiUtReleaseMutex (ACPI_MTX_DEBUG_CMD_COMPLETE);
109             if (ACPI_FAILURE (Status))
110             {
111                 return (Status);
112             }
113             Status = AcpiUtAcquireMutex (ACPI_MTX_DEBUG_CMD_READY);
114             if (ACPI_FAILURE (Status))
115             {
116                 return (Status);
117             }
118         }
119         else
120         {
121             /* Single threaded, we must get a command line ourselves */

123             /* 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 * FUNCTION:     AcpiDbSingleStep
160 *
161 * PARAMETERS:   WalkState      - Current walk
162 *               Op             - Current executing op (from aml interpreter)
163 *               OpcodeClass     - Class of the current AML Opcode
164 *
165 * RETURN:       Status
166 *
167 * DESCRIPTION:  Called just before execution of an AML opcode.
168 *
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);
436         return (AE_NO_MEMORY);
437     }
438     ACPI_MEMSET (AcpiGbl_DbBuffer, 0, ACPI_DEBUG_BUFFER_SIZE);

439     /* Initial scope is the root */

441     AcpiGbl_DbScopeBuf [0] = AML_ROOT_PREFIX;
442     AcpiGbl_DbScopeBuf [0] = '\\';
443     AcpiGbl_DbScopeBuf [1] = 0;
444     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);
459         return (Status);
460     }

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);
466         return (Status);
467     }

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);
475         AcpiOsPrintf ("Could not start debugger thread\n");
476         return (Status);
477     }

478     if (!AcpiGbl_DbOpt_verbose)
479     {
480         AcpiGbl_DbOpt_disasm = TRUE;
481         AcpiGbl_DbOpt_stats = FALSE;
482     }

484     return ACPI_STATUS (AE_OK);
485     return (AE_OK);
486 }

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 {
504     if (AcpiGbl_DbBuffer)
505     {
506         AcpiOsFree (AcpiGbl_DbBuffer);
507         AcpiGbl_DbBuffer = NULL;
508     }

511     /* Ensure that debug output is now disabled */

```



```
new/usr/src/common/acpica/components/debugger/dbxface.c
```

9

```
513     AcpiGbl_DbOutputFlags = ACPI_DB_DISABLE_OUTPUT;
514 }
_____unchanged_portion_omitted_____
```

new/usr/src/common/acpica/components/disassembler/dmbuffer.c

1

```
*****
18848 Fri Jan 17 07:29:51 2014
new/usr/src/common/acpica/components/disassembler/dmbuffer.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dmbuffer - AML disassembler, buffer and string support
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acdisasm.h"
48 #include "acparser.h"
49 #include "amlcode.h"

52 #ifndef ACPI_DISASSEMBLER

54 #define _COMPONENT ACPI_CA_DEBUGGER
55     ACPI_MODULE_NAME ("dmbuffer")

57 /* Local prototypes */

59 static void
```

new/usr/src/common/acpica/components/disassembler/dmbuffer.c

2

```
60 AcpiDmUnicode (
61     ACPI_PARSE_OBJECT *Op);

63 static void
64 AcpiDmIsEisaIdElement (
65     ACPI_PARSE_OBJECT *Op);

67 static void
68 AcpiDmPldBuffer (
69     UINT32 Level,
70     UINT8 *ByteData,
71     UINT32 ByteCount);

74 /*****
75 *
76 * FUNCTION: AcpiDmDisasmByteList
77 *
78 * PARAMETERS: Level - Current source code indentation level
79 *              ByteData - Pointer to the byte list
80 *              ByteCount - Length of the byte list
81 *
82 * RETURN: None
83 *
84 * DESCRIPTION: Dump an AML "ByteList" in Hex format. 8 bytes per line, prefixed
85 *              with the hex buffer offset.
86 *
87 *****/

89 void
90 AcpiDmDisasmByteList (
91     UINT32 Level,
92     UINT8 *ByteData,
93     UINT32 ByteCount)
94 {
95     UINT32 i;

98     if (!ByteCount)
99     {
100         return;
101     }

103     /* Dump the byte list */

105     for (i = 0; i < ByteCount; i++)
106     {
107         /* New line every 8 bytes */

109         if (((i % 8) == 0) && (i < ByteCount))
110         {
111             if (i > 0)
112             {
113                 AcpiOsPrintf ("\n");
114             }

116             AcpiDmIndent (Level);
117             if (ByteCount > 8)
118             {
119                 AcpiOsPrintf ("/* %04X */", i);
120             }
121         }

123         AcpiOsPrintf (" 0x%2.2X", (UINT32) ByteData[i]);

125         /* Add comma if there are more bytes to display */
```

```

127     if (i < (ByteCount - 1))
128     {
129         AcpiOsPrintf (",");
130     }
131 }

133 if (Level)
134 {
135     AcpiOsPrintf ("\n");
136 }
137 }

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 *****/

154 void
155 AcpiDmByteList (
156     ACPI_OP_WALK_INFO      *Info,
157     ACPI_PARSE_OBJECT      *Op)
158 {
159     UINT8                  *ByteData;
160     UINT32                  ByteCount;

163     ByteData = Op->Named.Data;
164     ByteCount = (UINT32) Op->Common.Value.Integer;

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:

174         AcpiDmResourceTemplate (Info, Op->Common.Parent, ByteData, ByteCount);
175         break;

177     case ACPI_DASM_STRING:

179         AcpiDmIndent (Info->Level);
180         AcpiUtPrintString ((char *) ByteData, ACPI_UINT16_MAX);
181         AcpiOsPrintf ("\n");
182         break;

184     case ACPI_DASM_UNICODE:

186         AcpiDmUnicode (Op);
187         break;

189     case ACPI_DASM_PLD_METHOD:

191         AcpiDmDisasmByteList (Info->Level, ByteData, ByteCount);

```

```

192     AcpiDmPldBuffer (Info->Level, ByteData, ByteCount);
193     break;

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 }

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 *****/

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;

231     /* Buffer size is the buffer argument */

233     SizeOp = Op->Common.Value.Arg;

235     /* Next, the initializer byte list to examine */

237     NextOp = SizeOp->Common.Next;
238     if (!NextOp)
239     {
240         return (FALSE);
241     }

243     /* Extract the byte list info */

245     ByteData = NextOp->Named.Data;
246     ByteCount = (UINT32) NextOp->Common.Value.Integer;
247     WordCount = ACPI_DIV_2 (ByteCount);

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);
259 }

261 /* For each word, 1st byte must be ascii, 2nd byte must be zero */

263 for (i = 0; i < (ByteCount - 2); i += 2)
264 {
265     if ((!ACPI_IS_PRINT (ByteData[i])) ||
266         (ByteData[(ACPI_SIZE) i + 1] != 0))
267     {
268         return (FALSE);
269     }
270 }

272 /* Ignore the Size argument in the disassembly of this buffer op */

274 SizeOp->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;
275 return (TRUE);
276 }

279 /*****
280 *
281 * FUNCTION:      AcpiDmIsStringBuffer
282 *
283 * PARAMETERS:    Op                - Buffer Object to be examined
284 *
285 * RETURN:        TRUE if buffer contains a ASCII string, FALSE otherwise
286 *
287 * DESCRIPTION:   Determine if a buffer Op contains a ASCII string
288 *
289 *****/

291 BOOLEAN
292 AcpiDmIsStringBuffer (
293     ACPI_PARSE_OBJECT    *Op)
294 {
295     UINT8                *ByteData;
296     UINT32                ByteCount;
297     ACPI_PARSE_OBJECT    *SizeOp;
298     ACPI_PARSE_OBJECT    *NextOp;
299     UINT32                i;

302     /* Buffer size is the buffer argument */

304     SizeOp = Op->Common.Value.Arg;

306     /* Next, the initializer byte list to examine */

308     NextOp = SizeOp->Common.Next;
309     if (!NextOp)
310     {
311         return (FALSE);
312     }

314     /* Extract the byte list info */

316     ByteData = NextOp->Named.Data;
317     ByteCount = (UINT32) NextOp->Common.Value.Integer;

319     /* Last byte must be the null terminator */

321     if ((!ByteCount)
322         || (ByteCount < 2)
323         || (ByteData[ByteCount-1] != 0))

```

```

324     {
325         return (FALSE);
326     }

328     for (i = 0; i < (ByteCount - 1); i++)
329     {
330         /* TBD: allow some escapes (non-ascii chars).
331          * they will be handled in the string output routine
332          */

334         if (!ACPI_IS_PRINT (ByteData[i]))
335         {
336             return (FALSE);
337         }
338     }

340     return (TRUE);
341 }

344 /*****
345 *
346 * FUNCTION:      AcpiDmIsPldBuffer
347 *
348 * PARAMETERS:    Op                - Buffer Object to be examined
349 *
350 * RETURN:        TRUE if buffer contains a ASCII string, FALSE otherwise
351 *
352 * DESCRIPTION:   Determine if a buffer Op contains a _PLD structure
353 *
354 *****/

356 BOOLEAN
357 AcpiDmIsPldBuffer (
358     ACPI_PARSE_OBJECT    *Op)
359 {
360     ACPI_NAMESPACE_NODE    *Node;
361     ACPI_PARSE_OBJECT    *ParentOp;

364     ParentOp = Op->Common.Parent;
365     if (!ParentOp)
366     {
367         return (FALSE);
368     }

370     /* Check for form: Name(_PLD, Buffer() {}). Not legal, however */

372     if (ParentOp->Common.AmlOpcode == AML_NAME_OP)
373     {
374         Node = ParentOp->Common.Node;

376         if (ACPI_COMPARE_NAME (Node->Name.Ascii, METHOD_NAME__PLD))
377         {
378             return (TRUE);
379         }

381         return (FALSE);
382     }

384     /* Check for proper form: Name(_PLD, Package() {Buffer() {}}) */

386     if (ParentOp->Common.AmlOpcode == AML_PACKAGE_OP)
387     {
388         ParentOp = ParentOp->Common.Parent;
389         if (!ParentOp)

```

```

390     {
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 * FUNCTION:      AcpiDmPldBuffer
411 *
412 * PARAMETERS:    Level              - Current source code indentation level
413 *                ByteData           - Pointer to the byte list
414 *                ByteCount          - Length of the byte list
415 *
416 * RETURN:        None
417 *
418 * DESCRIPTION:   Dump and format the contents of a _PLD buffer object
419 *
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 * FUNCTION:      AcpiDmUnicode
501 *
502 * PARAMETERS:    Op              - Byte List op containing Unicode string
503 *
504 * RETURN:        None
505 *
506 * DESCRIPTION:   Dump Unicode string as a standard ASCII string. (Remove
507 *                the extra zero bytes).
508 *
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 ("\n");
529 for (i = 0; i < (WordCount - 1); i++)
530 {
531     AcpiOsPrintf ("%c", (int) WordData[i]);
532 }

534 AcpiOsPrintf ("\n");
535 }

538 /*****
539 * FUNCTION:      AcpiDmIsEisaIdElement
540 *
541 * PARAMETERS:    Op                - Op to be examined
542 *
543 * RETURN:        None
544 *
545 * DESCRIPTION:   Determine if an Op (argument to _HID or _CID) can be converted
546 *               to an EISA ID.
547 *
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 convertible 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);

655     /* Don't need to use the length, just walk the peer list */

656     NextOp = NextOp->Common.Next;
657     while (NextOp)
658     {
659         AcpiDmIsEisaIdElement (NextOp);
660         NextOp = NextOp->Common.Next;
661     }
662 }

663
664
665
666
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  *****/

678 void
679 AcpiDmEisaId (
680     UINT32          EncodedId)
681 {
682     UINT32          BigEndianId;

683     /* Swap from little-endian to big-endian to simplify conversion */

684     BigEndianId = AcpiUtDwordByteSwap (EncodedId);

685     /* Split to form "AAANNNN" string */

686     AcpiOsPrintf ("EisaId (\"%c%c%c%4.4X\")",
687         /* Three Alpha characters (AAA), 5 bits each */
688         (int) ((BigEndianId >> 26) & 0x1F) + 0x40,
689         (int) ((BigEndianId >> 21) & 0x1F) + 0x40,
690         (int) ((BigEndianId >> 16) & 0x1F) + 0x40,
691         /* Numeric part (NNNN) is simply the lower 16 bits */
692         (UINT32) (BigEndianId & 0xFFFF));
693 }

694
695 #endif
```

new/usr/src/common/acpica/components/disassembler/dmdeferred.c

1

```
*****
7921 Fri Jan 17 07:29:51 2014
new/usr/src/common/acpica/components/disassembler/dmdeferred.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: dmdeferred - Disassembly of deferred AML opcodes
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acdispat.h"
48 #include "amlcode.h"
49 #include "acdisasm.h"
50 #include "acparser.h"

52 #define _COMPONENT          ACPI_CA_DISASSEMBLER
53 #define ACPI_MODULE_NAME    ("dmdeferred")

56 /* Local prototypes */

58 static ACPI_STATUS
59 AcpiDmDeferredParse (
60     ACPI_PARSE_OBJECT *Op,
```

new/usr/src/common/acpica/components/disassembler/dmdeferred.c

2

```
61     UINT8 *Aml,
62     UINT32 AmlLength);

65 /*****
66 *
67 * FUNCTION: AcpiDmDeferredOps
68 *
69 * PARAMETERS: Root - Root of the parse tree
70 *
71 * RETURN: Status
72 *
73 * DESCRIPTION: Parse the deferred opcodes (Methods, regions, etc.)
74 *
75 *****/

77 ACPI_STATUS
78 AcpiDmDeferredOps (
79     ACPI_PARSE_OBJECT *Root)
80 {
81     const ACPI_OPCODE_INFO *OpInfo;
82     ACPI_PARSE_OBJECT *Op = Root;
83     ACPI_STATUS Status;

86     ACPI_FUNCTION_ENTRY ();

89     /* Traverse the entire parse tree */

91     while (Op)
92     {
93         OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
94         if (!(OpInfo->Flags & AML_DEFER))
95         {
96             Op = AcpiPsGetDepthNext (Root, Op);
97             continue;
98         }

100         /* Now we know we have a deferred opcode */

102         switch (Op->Common.AmlOpcode)
103         {
104             case AML_METHOD_OP:
105             case AML_BUFFER_OP:
106             case AML_PACKAGE_OP:
107             case AML_VAR_PACKAGE_OP:

109                 Status = AcpiDmDeferredParse (Op, Op->Named.Data, Op->Named.Length);
110                 if (ACPI_FAILURE (Status))
111                 {
112                     return (Status);
113                 }
114                 break;

116         /* We don't need to do anything for these deferred opcodes */

118             case AML_REGION_OP:
119             case AML_DATA_REGION_OP:
120             case AML_CREATE_QWORD_FIELD_OP:
121             case AML_CREATE_DWORD_FIELD_OP:
122             case AML_CREATE_WORD_FIELD_OP:
123             case AML_CREATE_BYTE_FIELD_OP:
124             case AML_CREATE_BIT_FIELD_OP:
125             case AML_CREATE_FIELD_OP:
126             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  * FUNCTION:      AcpiDmDeferredParse
146  *
147  * PARAMETERS:    Op          - Root Op of the deferred opcode
148  *                Aml         - Pointer to the raw AML
149  *                AmlLength    - Length of the AML
150  *
151  * RETURN:        Status
152  *
153  * DESCRIPTION:    Parse one deferred opcode
154  *                (Methods, operation regions, etc.)
155  *
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 }
```

new/usr/src/common/acpica/components/disassembler/dmnames.c

1

```
*****
11854 Fri Jan 17 07:29:51 2014
new/usr/src/common/acpica/components/disassembler/dmnames.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dmnames - AML disassembler, names, namestrings, pathnames
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "acparser.h"
48 #include "amlcode.h"
49 #include "acnamesp.h"
50 #include "acdisasm.h"

52 #ifndef ACPI_DISASSEMBLER

54 #define _COMPONENT ACPI_CA_DEBUGGER
55 #define ACPI_MODULE_NAME ("dmnames")

57 /* Local prototypes */
```

new/usr/src/common/acpica/components/disassembler/dmnames.c

2

```
59 #ifdef ACPI_OBSOLETE_FUNCTIONS
60 void
61 AcpiDmDisplayPath (
62     ACPI_PARSE_OBJECT *Op);
63 #endif

66 /*****
67 *
68 * FUNCTION: AcpiDmDumpName
69 *
70 * PARAMETERS: Name - 4 character ACPI name
71 *
72 * RETURN: Final length of name
73 *
74 * DESCRIPTION: Dump an ACPI name, minus any trailing underscores.
75 *
76 *****/

78 UINT32
79 AcpiDmDumpName (
80     UINT32 Name)
81 {
82     UINT32 i;
83     UINT32 Length;
84     char NewName[4];

87     /* Copy name locally in case the original name is not writeable */
89     *ACPI_CAST_PTR (UINT32, &NewName[0]) = Name;

91     /* Ensure that the name is printable, even if we have to fix it */

93     AcpiUtRepairName (NewName);

95     /* Remove all trailing underscores from the name */

97     Length = ACPI_NAME_SIZE;
98     for (i = (ACPI_NAME_SIZE - 1); i != 0; i--)
99     {
100         if (NewName[i] == '_')
101         {
102             Length--;
103         }
104         else
105         {
106             break;
107         }
108     }

110     /* Dump the name, up to the start of the trailing underscores */

112     for (i = 0; i < Length; i++)
113     {
114         AcpiOsPrintf ("%c", NewName[i]);
115     }

117     return (Length);
118 }

_____unchanged_portion_omitted_____

202 /*****
203 *
```

```

204 * 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])))
331             (NamePath->Common.Value.String[0] == '\\'))
332         {
333             AcpiDmNamestring (NamePath->Common.Value.String);
334             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

```

new/usr/src/common/acpica/components/disassembler/dmobject.c

1

```
*****
15696 Fri Jan 17 07:29:52 2014
new/usr/src/common/acpica/components/disassembler/dmobject.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dmobject - ACPI object decode and display
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "acnamesp.h"
48 #include "acdisasm.h"

51 #ifndef ACPI_DISASSEMBLER

53 #define _COMPONENT ACPI_CA_DEBUGGER
54 ACPI_MODULE_NAME ("dmnames")

56 /* Local prototypes */

58 static void
```

new/usr/src/common/acpica/components/disassembler/dmobject.c

2

```
59 AcpiDmDecodeNode (
60     ACPI_NAMESPACE_NODE *Node);

63 /*****
64 *
65 * FUNCTION: AcpiDmDumpMethodInfo
66 *
67 * PARAMETERS: Status - Method execution status
68 *             WalkState - Current state of the parse tree walk
69 *             Op - Executing parse op
70 *
71 * RETURN: None
72 *
73 * DESCRIPTION: Called when a method has been aborted because of an error.
74 *             Dumps the method execution stack, and the method locals/args,
75 *             and disassembles the AML opcode that failed.
76 *
77 *****/

79 void
80 AcpiDmDumpMethodInfo (
81     ACPI_STATUS Status,
82     ACPI_WALK_STATE *WalkState,
83     ACPI_PARSE_OBJECT *Op)
84 {
85     ACPI_PARSE_OBJECT *Next;
86     ACPI_THREAD_STATE *Thread;
87     ACPI_WALK_STATE *NextWalkState;
88     ACPI_NAMESPACE_NODE *PreviousMethod = NULL;

91     /* Ignore control codes, they are not errors */

93     if ((Status & AE_CODE_MASK) == AE_CODE_CONTROL)
94     {
95         return;
96     }

98     /* We may be executing a deferred opcode */

100     if (WalkState->DeferredNode)
101     {
102         AcpiOsPrintf ("Executing subtree for Buffer/Package/Region\n");
103         return;
104     }

106     /*
107      * If there is no Thread, we are not actually executing a method.
108      * This can happen when the iASL compiler calls the interpreter
109      * to perform constant folding.
110      */
111     Thread = WalkState->Thread;
112     if (!Thread)
113     {
114         return;
115     }

117     /* Display exception and method name */

119     AcpiOsPrintf ("\n*** Exception %s during execution of method ",
120         AcpiFormatException (Status));
121     AcpiNsPrintNodePathname (WalkState->MethodNode, NULL);

123     /* Display stack of executing methods */
```

```

125     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 *
255 * RETURN:      None
256 *
257 * DESCRIPTION: Short display of a namespace node
258 *
259 *****/
261 static void
262 AcpiDmDecodeNode (
263     ACPI_NAMESPACE_NODE *Node)
264 {
265     AcpiOsPrintf ("<Node>          Name %4.4s",
266         AcpiUtGetNodeName (Node));
267
268     if (Node->Flags & ANOBJ_METHOD_ARG)
269     {
270         AcpiOsPrintf (" [Method Arg]");
271     }
272     if (Node->Flags & ANOBJ_METHOD_LOCAL)
273     {
274         AcpiOsPrintf (" [Method Local]");
275     }
276
277     switch (Node->Type)
278     {
279     /* These types have no attached object */
280
281     case ACPI_TYPE_DEVICE:
282
283         AcpiOsPrintf (" Device");
284         break;
285
286     case ACPI_TYPE_THERMAL:
287
288         AcpiOsPrintf (" Thermal Zone");
289         break;
290
291     default:
292
293         AcpiDmDecodeInternalObject (AcpiNsGetAttachedObject (Node));
294         break;
295     }
296 }
297
300 *****/
301 *
302 * FUNCTION:      AcpiDmDisplayInternalObject
303 *
304 * PARAMETERS:    ObjDesc      - Object to be displayed
305 *                WalkState     - Current walk state
306 *
307 * RETURN:        None
308 *
309 * DESCRIPTION:   Short display of an internal object
310 *
311 *****/
313 void
314 AcpiDmDisplayInternalObject (
315     ACPI_OPERAND_OBJECT *ObjDesc,
316     ACPI_WALK_STATE *WalkState)
317 {
318     UINT8      Type;

```

```

321     AcpiOsPrintf ("%p ", ObjDesc);
322
323     if (!ObjDesc)
324     {
325         AcpiOsPrintf ("<Null Object>\n");
326         return;
327     }
328
329     /* Decode the object type */
330
331     switch (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc))
332     {
333     case ACPI_DESC_TYPE_PARSER:
334
335         AcpiOsPrintf ("<Parser>  ");
336         break;
337
338     case ACPI_DESC_TYPE_NAMED:
339
340         AcpiDmDecodeNode ((ACPI_NAMESPACE_NODE *) ObjDesc);
341         break;
342
343     case ACPI_DESC_TYPE_OPERAND:
344
345         Type = ObjDesc->Common.Type;
346         if (Type > ACPI_TYPE_LOCAL_MAX)
347         {
348             AcpiOsPrintf (" Type %X [Invalid Type]", (UINT32) Type);
349             return;
350         }
351
352         /* Decode the ACPI object type */
353
354         switch (ObjDesc->Common.Type)
355         {
356         case ACPI_TYPE_LOCAL_REFERENCE:
357
358             AcpiOsPrintf ("[%s] ", AcpiUtGetReferenceName (ObjDesc));
359
360             /* Decode the reference */
361
362             switch (ObjDesc->Reference.Class)
363             {
364             case ACPI_REFCLASS_LOCAL:
365
366                 AcpiOsPrintf ("%X ", ObjDesc->Reference.Value);
367                 if (WalkState)
368                 {
369                     ObjDesc = WalkState->LocalVariables
370                         [ObjDesc->Reference.Value].Object;
371                     AcpiOsPrintf ("%p", ObjDesc);
372                     AcpiDmDecodeInternalObject (ObjDesc);
373                 }
374                 break;
375
376             case ACPI_REFCLASS_ARG:
377
378                 AcpiOsPrintf ("%X ", ObjDesc->Reference.Value);
379                 if (WalkState)
380                 {
381                     ObjDesc = WalkState->Arguments
382                         [ObjDesc->Reference.Value].Object;

```



```

383         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 Fri Jan 17 07:29:52 2014
new/usr/src/common/acpica/components/disassembler/dmopcode.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dmopcode - AML disassembler, specific AML opcodes
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acparser.h"
47 #include "amlcode.h"
48 #include "acdisasm.h"

50 #ifdef ACPI_DISASSEMBLER

52 #define _COMPONENT          ACPI_CA_DEBUGGER
53       ACPI_MODULE_NAME    ("dmopcode")

55 /* Local prototypes */

57 static void
58 AcpiDmMatchKeyword (
59     ACPI_PARSE_OBJECT      *Op);

```

```

62 /*****
63 *
64 * FUNCTION:      AcpiDmPredefinedDescription
65 *
66 * PARAMETERS:    Op              - Name() parse object
67 *
68 * RETURN:        None
69 *
70 * DESCRIPTION:    Emit a description comment for a predefined ACPI name.
71 *                  Used for iASL compiler only.
72 *
73 *****/

75 void
76 AcpiDmPredefinedDescription (
77     ACPI_PARSE_OBJECT      *Op)
78 {
79     #ifdef ACPI_AS_L_COMPILER
80         const AH_PREDEFINED_NAME *Info;
81         char *NameString;
82         int LastCharIsDigit;
83         int LastCharsAreHex;

86         if (!Op)
87         {
88             return;
89         }

91         /* Ensure that the comment field is emitted only once */

93         if (Op->Common.DisasmFlags & ACPI_PARSEOP_PREDEF_CHECKED)
94         {
95             return;
96         }
97         Op->Common.DisasmFlags |= ACPI_PARSEOP_PREDEF_CHECKED;

99         /* Predefined name must start with an underscore */

101         NameString = ACPI_CAST_PTR (char, &Op->Named.Name);
102         if (NameString[0] != '_')
103         {
104             return;
105         }

107         /*
108          * Check for the special ACPI names:
109          * _ACd, _ALd, _EJd, _Exx, _Lxx, _Qxx, _Wxx, _Ta
110          * (where d=decimal_digit, x=hex_digit, a=anything)
111          *
112          * Convert these to the generic name for table lookup.
113          * Note: NameString is guaranteed to be upper case here.
114          */
115         LastCharIsDigit =
116             (ACPI_IS_DIGIT (NameString[3])); /* d */
117         LastCharsAreHex =
118             (ACPI_IS_XDIGIT (NameString[2]) && /* xx */
119              ACPI_IS_XDIGIT (NameString[3]));

121         switch (NameString[1])
122         {
123             case 'A':

125                 if ((NameString[2] == 'C') && (LastCharIsDigit))

```

```

126     {
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
206  *                - Parse object
207  *
208  * RETURN:        None
209  *
210  * DESCRIPTION:    Emit a description comment for a resource descriptor tag
211  *                (which is a predefined ACPI name.) Used for iASL compiler only.
212  *****/

214 void
215 AcpiDmFieldPredefinedDescription (
216     ACPI_PARSE_OBJECT *Op)
217 {
218     #ifndef 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)
523 {
524     const ACPI_OPCODE_INFO *OpInfo = NULL;
525     UINT32 Offset;
526     UINT32 Length;
527     ACPI_PARSE_OBJECT *Child;
528     ACPI_STATUS Status;
529     UINT8 *Aml;

532     if (!Op)
533     {
534         AcpiOsPrintf("<NULL OP PTR>");
535         return;
536     }

538     switch (Op->Common.DisasmOpcode)
539     {
540     case ACPI_DASM_MATCHOP:

542         AcpiDmMatchKeyword (Op);
543         return;

545     case ACPI_DASM_LNOT_SUFFIX:

547         switch (Op->Common.AmlOpcode)
548         {
549         case AML_LEQUAL_OP:

551             AcpiOsPrintf ("LNotEqual");
552             break;

554         case AML_LGREATER_OP:

556             AcpiOsPrintf ("LLessEqual");
557             break;

559         case AML_LLESS_OP:

561             AcpiOsPrintf ("LGreaterEqual");
562             break;

564         default:

566             break;
567         }
568         Op->Common.DisasmOpcode = 0;
569         Op->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;
570         return;

572     default:
573         break;
574     }

577     OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);

579     /* The op and arguments */

581     switch (Op->Common.AmlOpcode)
582     {
583     case AML_LNOT_OP:

585         Child = Op->Common.Value.Arg;
586         if ((Child->Common.AmlOpcode == AML_LEQUAL_OP) ||
587             (Child->Common.AmlOpcode == AML_LGREATER_OP) ||

```

```

588         (Child->Common.AmlOpcode == AML_LLESS_OP))
589     {
590         Child->Common.DisasmOpcode = ACPI_DASM_LNOT_SUFFIX;
591         Op->Common.DisasmOpcode = ACPI_DASM_LNOT_PREFIX;
592     }
593     else
594     {
595         AcpiOsPrintf ("%s", OpInfo->Name);
596     }
597     break;

599     case AML_BYTE_OP:

601         AcpiOsPrintf ("0x%2.2X", (UINT32) Op->Common.Value.Integer);
602         break;

604     case AML_WORD_OP:

606         if (Op->Common.DisasmOpcode == ACPI_DASM_EISAID)
607         {
608             AcpiDmEisaId ((UINT32) Op->Common.Value.Integer);
609         }
610         else
611         {
612             AcpiOsPrintf ("0x%4.4X", (UINT32) Op->Common.Value.Integer);
613         }
614         break;

616     case AML_DWORD_OP:

618         if (Op->Common.DisasmOpcode == ACPI_DASM_EISAID)
619         {
620             AcpiDmEisaId ((UINT32) Op->Common.Value.Integer);
621         }
622         else
623         {
624             AcpiOsPrintf ("0x%8.8X", (UINT32) Op->Common.Value.Integer);
625         }
626         break;

628     case AML_QWORD_OP:

630         AcpiOsPrintf ("0x%8.8X%8.8X",
631             ACPI_FORMAT_UINT64 (Op->Common.Value.Integer));
632         break;

634     case AML_STRING_OP:

636         AcpiUtPrintString (Op->Common.Value.String, ACPI_UINT16_MAX);
637         break;

639     case AML_BUFFER_OP:
640     /*
641      * Determine the type of buffer. We can have one of the following:
642      *
643      * 1) ResourceTemplate containing Resource Descriptors.
644      * 2) Unicode String buffer
645      * 3) ASCII String buffer
646      * 4) Raw data buffer (if none of the above)
647      *
648      * Since there are no special AML opcodes to differentiate these
649      * types of buffers, we have to closely look at the data in the
650      * buffer to determine the type.
651      */
652     if (!AcpiGbl_NoResourceDisassembly)
653     {

```

```

654     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%.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 */
```



new/usr/src/common/acpica/components/disassembler/dmresrc.c

1

```
*****
13510 Fri Jan 17 07:29:53 2014
new/usr/src/common/acpica/components/disassembler/dmresrc.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dmresrc.c - Resource Descriptor disassembly
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "amlcode.h"
48 #include "acdisasm.h"

50 #ifdef ACPI_DISASSEMBLER

52 #define _COMPONENT ACPI_CA_DEBUGGER
53 ACPI_MODULE_NAME ("dbresrc")

56 /* Dispatch tables for Resource disassembly functions */

58 typedef
```

new/usr/src/common/acpica/components/disassembler/dmresrc.c

2

```
59 void (*ACPI_RESOURCE_HANDLER) (
60     AML_RESOURCE *Resource,
61     UINT32 Length,
62     UINT32 Level);

58 static ACPI_RESOURCE_HANDLER AcpiGbl_DmResourceDispatch [] =
59 {
60     /* Small descriptors */

62     NULL, /* 0x00, Reserved */
63     NULL, /* 0x01, Reserved */
64     NULL, /* 0x02, Reserved */
65     NULL, /* 0x03, Reserved */
66     AcpiDmIrqDescriptor, /* 0x04, ACPI_RESOURCE_NAME_IRQ_FORMAT */
67     AcpiDmDmaDescriptor, /* 0x05, ACPI_RESOURCE_NAME_DMA_FORMAT */
68     AcpiDmStartDependentDescriptor, /* 0x06, ACPI_RESOURCE_NAME_START_DEPENDENT */
69     AcpiDmEndDependentDescriptor, /* 0x07, ACPI_RESOURCE_NAME_END_DEPENDENT */
70     AcpiDmIoDescriptor, /* 0x08, ACPI_RESOURCE_NAME_IO_PORT */
71     AcpiDmFixedIoDescriptor, /* 0x09, ACPI_RESOURCE_NAME_FIXED_IO_PORT */
72     AcpiDmFixedDmaDescriptor, /* 0x0A, ACPI_RESOURCE_NAME_FIXED_DMA */
73     NULL, /* 0x0A, Reserved */
74     NULL, /* 0x0B, Reserved */
75     NULL, /* 0x0C, Reserved */
76     NULL, /* 0x0D, Reserved */
77     AcpiDmVendorSmallDescriptor, /* 0x0E, ACPI_RESOURCE_NAME_SMALL_VENDOR */
78     NULL, /* 0x0F, ACPI_RESOURCE_NAME_END_TAG (not use) */

79     /* Large descriptors */

81     NULL, /* 0x00, Reserved */
82     AcpiDmMemory24Descriptor, /* 0x01, ACPI_RESOURCE_NAME_MEMORY_24 */
83     AcpiDmGenericRegisterDescriptor, /* 0x02, ACPI_RESOURCE_NAME_GENERIC_REGISTER */
84     NULL, /* 0x03, Reserved */
85     AcpiDmVendorLargeDescriptor, /* 0x04, ACPI_RESOURCE_NAME_LARGE_VENDOR */
86     AcpiDmMemory32Descriptor, /* 0x05, ACPI_RESOURCE_NAME_MEMORY_32 */
87     AcpiDmFixedMemory32Descriptor, /* 0x06, ACPI_RESOURCE_NAME_FIXED_MEMORY_32 */
88     AcpiDmDwordDescriptor, /* 0x07, ACPI_RESOURCE_NAME_DWORD_ADDRESS_SPA */
89     AcpiDmWordDescriptor, /* 0x08, ACPI_RESOURCE_NAME_WORD_ADDRESS_SPA */
90     AcpiDmInterruptDescriptor, /* 0x09, ACPI_RESOURCE_NAME_EXTENDED_XRUP */
91     AcpiDmQwordDescriptor, /* 0x0A, ACPI_RESOURCE_NAME_QWORD_ADDRESS_SPA */
92     AcpiDmExtendedDescriptor, /* 0x0B, ACPI_RESOURCE_NAME_EXTENDED_ADDRESS */
93     AcpiDmGpioDescriptor, /* 0x0C, ACPI_RESOURCE_NAME_GPIO */
94     NULL, /* 0x0D, Reserved */
95     AcpiDmSerialBusDescriptor, /* 0x0E, ACPI_RESOURCE_NAME_SERIAL_BUS */
96     AcpiDmExtendedDescriptor, /* 0x0B, ACPI_RESOURCE_NAME_EXTENDED_ADDRESS */
97 };

/* unchanged portion omitted */

230 /*****
231 *
232 * FUNCTION: AcpiDmResourceTemplate
233 *
234 * PARAMETERS: Info - Current parse tree walk info
235 * ByteData - Pointer to the byte list data
236 * ByteCount - Length of the byte list
237 *
238 * RETURN: None
239 *
240 * DESCRIPTION: Dump the contents of a Resource Template containing a set of
241 * Resource Descriptors.
242 *
243 *****/

245 void
246 AcpiDmResourceTemplate (
```

```

247     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);
283         Status = AcpiUtValidateResource (Aml, &ResourceIndex);
284         if (ACPI_FAILURE (Status))
285         {
286             AcpiOsPrintf ("/*** Could not validate Resource, type (%X) %s***\n"
287                 ResourceType, AcpiFormatException (Status));
288             return;
289         }

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 () */\n");
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  *                Op              - Buffer Op to be examined
373  *
374  * RETURN:        Status. AE_OK if valid template
375  *
376  * DESCRIPTION:    Walk a byte list to determine if it consists of a valid set
377                    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));
421     Status = AcpiUtWalkAmlResources (Aml, Length, NULL, &EndAml);
422     if (ACPI_FAILURE (Status))
423     {
424         return (AE_TYPE);
425     }

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_

```

new/usr/src/common/acpica/components/disassembler/dmresrcl.c

1

```
*****
29253 Fri Jan 17 07:29:53 2014
new/usr/src/common/acpica/components/disassembler/dmresrcl.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dmresrcl.c - "Large" Resource Descriptor disassembly
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "acdisasm.h"

50 #ifdef ACPI_DISASSEMBLER

52 #define _COMPONENT ACPI_CA_DEBUGGER
53 #define ACPI_MODULE_NAME ("dbresrcl")

56 /* Common names for address and memory descriptors */

58 static char *AcpiDmAddressNames[] =
```

new/usr/src/common/acpica/components/disassembler/dmresrcl.c

2

```
59 {
60     "Granularity",
61     "Range Minimum",
62     "Range Maximum",
63     "Translation Offset",
64     "Length"
65 };
66
67 unchanged_portion_omitted
68
69 /* Local prototypes */
70
71 static void
72 AcpiDmSpaceFlags (
73     UINT8 Flags);
74
75 static void
76 AcpiDmIoFlags (
77     UINT8 Flags);
78
79 static void
80 AcpiDmIoFlags2 (
81     UINT8 SpecificFlags);
82
83 static void
84 AcpiDmMemoryFlags (
85     UINT8 Flags,
86     UINT8 SpecificFlags);
87
88 static void
89 AcpiDmMemoryFlags2 (
90     UINT8 SpecificFlags);
91
92 static void
93 AcpiDmResourceSource (
94     AML_RESOURCE *Resource,
95     ACPI_SIZE MinimumLength,
96     UINT32 Length);
97
98 static void
99 AcpiDmAddressFields (
100     void *Source,
101     UINT8 Type,
102     UINT32 Level);
103
104 static void
105 AcpiDmAddressPrefix (
106     UINT8 Type);
107
108 static void
109 AcpiDmAddressCommon (
110     AML_RESOURCE *Resource,
111     UINT8 Type,
112     UINT32 Level);
113
114 static void
115 AcpiDmAddressFlags (
116     AML_RESOURCE *Resource);
117
118
119 /*****
120 *
121 * FUNCTION: AcpiDmMemoryFields
122 *
123 * PARAMETERS: Source - Pointer to the contiguous data fields
124 *              Type - 16 or 32 (bit)
125 */
```

```

132 *           Level           - Current source code indentation level
133 *
134 * RETURN:      None
135 *
136 * DESCRIPTION: Decode fields common to Memory24 and Memory32 descriptors
137 *
138 *****/

140 static void
141 AcpiDmMemoryFields (
142     void                *Source,
143     UINT8               Type,
144     UINT32              Level)
145 {
146     UINT32              i;

149     for (i = 0; i < 4; i++)
150     {
151         AcpiDmIndent (Level + 1);

153         switch (Type)
154         {
155             case 16:

157                 AcpiDmDumpInteger16 (ACPI_CAST_PTR (UINT16, Source)[i],
158                     AcpiDmMemoryNames[i]);
159                 break;

161             case 32:

163                 AcpiDmDumpInteger32 (ACPI_CAST_PTR (UINT32, Source)[i],
164                     AcpiDmMemoryNames[i]);
165                 break;

167             default:

169                 return;
170         }
171     }
172 }

175 /*****
176 *
177 * FUNCTION:      AcpiDmAddressFields
178 *
179 * PARAMETERS:    Source          - Pointer to the contiguous data fields
180 *                Type            - 16, 32, or 64 (bit)
181 *                Level          - Current source code indentation level
182 *
183 * RETURN:        None
184 *
185 * DESCRIPTION: Decode fields common to address descriptors
186 *
187 *****/

189 static void
190 AcpiDmAddressFields (
191     void                *Source,
192     UINT8               Type,
193     UINT32              Level)
194 {
195     UINT32              i;

```

```

198     AcpiOsPrintf ("\n");

200     for (i = 0; i < 5; i++)
201     {
202         AcpiDmIndent (Level + 1);

204         switch (Type)
205         {
206             case 16:

208                 AcpiDmDumpInteger16 (ACPI_CAST_PTR (UINT16, Source)[i],
209                     AcpiDmAddressNames[i]);
210                 break;

212             case 32:

214                 AcpiDmDumpInteger32 (ACPI_CAST_PTR (UINT32, Source)[i],
215                     AcpiDmAddressNames[i]);
216                 break;

218             case 64:

220                 AcpiDmDumpInteger64 (ACPI_CAST_PTR (UINT64, Source)[i],
221                     AcpiDmAddressNames[i]);
222                 break;

224             default:

226                 return;
227         }
228     }
229 }

232 /*****
233 *
234 * FUNCTION:      AcpiDmAddressPrefix
235 *
236 * PARAMETERS:    Type            - Descriptor type
237 *
238 * RETURN:        None
239 *
240 * DESCRIPTION: Emit name prefix representing the address descriptor type
241 *
242 *****/

244 static void
245 AcpiDmAddressPrefix (
246     UINT8              Type)
247 {
249     switch (Type)
250     {
251         case ACPI_RESOURCE_TYPE_ADDRESS16:

253             AcpiOsPrintf ("Word");
254             break;

256         case ACPI_RESOURCE_TYPE_ADDRESS32:

258             AcpiOsPrintf ("DWord");
259             break;

261         case ACPI_RESOURCE_TYPE_ADDRESS64:

263             AcpiOsPrintf ("QWord");

```

```

264     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 (SpecificFlags & 0x3)]);
347             AcpiOsPrintf (" %s", AcpiGbl_RngDecode [SpecificFlags & 0x3]);
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)]);
426     AcpiGbl_ConsumeDecode [(Flags & 1)],
427     AcpiGbl_MinDecode [(Flags & 0x4) >> 2],
428     AcpiGbl_MaxDecode [(Flags & 0x8) >> 3],
429     AcpiGbl_DecDecode [(Flags & 0x2) >> 1]);
430 }
431
432 /*****
433  * FUNCTION:      AcpiDmIoFlags2
434  *
435  * PARAMETERS:    SpecificFlags      - "Specific" flag byte to be decoded
436  *
437  * RETURN:        None
438  *
439  * DESCRIPTION:   Decode the flags specific to IO Address space descriptors
440  *****/
441
442 static void
443 AcpiDmIoFlags2 (
444     UINT8                SpecificFlags)
445 {
446     AcpiOsPrintf (" ", %s",
447         AcpiGbl_TtpDecode [ACPI_EXTRACT_1BIT_FLAG (SpecificFlags, 4)]);
448     AcpiGbl_TtpDecode [(SpecificFlags & 0x10) >> 4]);
449
450     /* TRS is only used if TTP is TypeTranslation */
451
452     if (SpecificFlags & 0x10)
453     {
454         AcpiOsPrintf (" ", %s",
455             AcpiGbl_TrnsDecode [ACPI_EXTRACT_1BIT_FLAG (SpecificFlags, 5)]);
456         AcpiGbl_TrnsDecode [(SpecificFlags & 0x20) >> 5]);
457     }
458 }
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
471 static void
472 AcpiDmMemoryFlags (
473     UINT8                Flags,
474     UINT8                SpecificFlags)
475 {

```

```

476     AcpiOsPrintf ("%s, %s, %s, %s, %s, %s,",
477         AcpiGbl_ConsumeDecode [ACPI_GET_1BIT_FLAG (Flags)],
478         AcpiGbl_DecDecode [ACPI_EXTRACT_1BIT_FLAG (Flags, 1)],
479         AcpiGbl_MinDecode [ACPI_EXTRACT_1BIT_FLAG (Flags, 2)],
480         AcpiGbl_MaxDecode [ACPI_EXTRACT_1BIT_FLAG (Flags, 3)],
481         AcpiGbl_MemDecode [ACPI_EXTRACT_2BIT_FLAG (SpecificFlags, 1)],
482         AcpiGbl_RwDecode [ACPI_GET_1BIT_FLAG (SpecificFlags)]);
483     AcpiGbl_ConsumeDecode [(Flags & 1)],
484     AcpiGbl_DecDecode [(Flags & 0x2) >> 1],
485     AcpiGbl_MinDecode [(Flags & 0x4) >> 2],
486     AcpiGbl_MaxDecode [(Flags & 0x8) >> 3],
487     AcpiGbl_MemDecode [(SpecificFlags & 0x6) >> 1],
488     AcpiGbl_RwDecode [(SpecificFlags & 0x1)]);
489 }
490
491 /*****
492  * FUNCTION:      AcpiDmMemoryFlags2
493  *
494  * PARAMETERS:    SpecificFlags      - "Specific" flag byte to be decoded
495  *
496  * RETURN:        None
497  *
498  * DESCRIPTION:   Decode flags specific to Memory Address Space descriptors
499  *****/
500
501 static void
502 AcpiDmMemoryFlags2 (
503     UINT8                SpecificFlags)
504 {
505     AcpiOsPrintf (" ", %s, %s",
506         AcpiGbl_MtpDecode [ACPI_EXTRACT_2BIT_FLAG (SpecificFlags, 3)],
507         AcpiGbl_TtpDecode [ACPI_EXTRACT_1BIT_FLAG (SpecificFlags, 5)]);
508     AcpiGbl_MtpDecode [(SpecificFlags & 0x18) >> 3],
509     AcpiGbl_TtpDecode [(SpecificFlags & 0x20) >> 5]);
510 }
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
524 static void
525 AcpiDmResourceSource (
526     *Resource,
527     ACPI_SIZE      MinimumTotalLength,
528     ACPI_SIZE      ResourceLength)
529 {
530     UINT8          *AmlResourceSource;
531     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);
569         AcpiUtPrintString ((char *) &AmlResourceSource[1], ACPI_UINT8_MAX);
570     }

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 {
855     /* Dump name and read/write flag */
856
857     AcpiDmIndent (Level);
858     AcpiOsPrintf ("Memory32Fixed (%s,\n",
859         AcpiGbl_RwDecode [ACPI_GET_1BIT_FLAG (Resource->FixedMemory32.Flags)];
860     AcpiGbl_RwDecode [Resource->FixedMemory32.Flags & 1]);
861
862     AcpiDmIndent (Level + 1);
863     AcpiDmDumpInteger32 (Resource->FixedMemory32.Address, "Address Base");
864
865     AcpiDmIndent (Level + 1);
866     AcpiDmDumpInteger32 (Resource->FixedMemory32.AddressLength, "Address Length")
867
868     /* Insert a descriptor name */
869
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;
953
954     AcpiDmIndent (Level);
955     AcpiOsPrintf ("Interrupt (%s, %s, %s, %s, ",
956         AcpiGbl_ConsumeDecode [ACPI_GET_1BIT_FLAG (Resource->ExtendedIrq.Flags)]
957         AcpiGbl_HeDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->ExtendedIrq.Flags, 1)
958         AcpiGbl_LlDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->ExtendedIrq.Flags, 2)
959         AcpiGbl_ShrDecode [ACPI_EXTRACT_2BIT_FLAG (Resource->ExtendedIrq.Flags,
960         AcpiGbl_ConsumeDecode [(Resource->ExtendedIrq.Flags & 1)],
961         AcpiGbl_HeDecode [(Resource->ExtendedIrq.Flags >> 1) & 1],
962         AcpiGbl_LlDecode [(Resource->ExtendedIrq.Flags >> 2) & 1],
963         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);
971
972 /* Insert a descriptor name */
973
974 AcpiDmDescriptorName ();
975 AcpiOsPrintf (")\n");
976
977 /* Dump the interrupt list */
978
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 }
987
988 AcpiDmIndent (Level);
989 AcpiOsPrintf ("}\n");
990 }

```

unchanged portion omitted

1064 #endif

new/usr/src/common/acpica/components/disassembler/dmresrc12.c

1

```
*****
20667 Fri Jan 17 07:29:53 2014
new/usr/src/common/acpica/components/disassembler/dmresrc12.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: dmresrc12.c - "Large" Resource Descriptor disassembly (#2)
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "acpi.h"
45 #include "accommon.h"
46 #include "acdisasm.h"
47
48 #ifdef ACPI_DISASSEMBLER
49
50 #define _COMPONENT ACPI_CA_DEBUGGER
51 #define ACPI_MODULE_NAME ("dbresrc12")
52
53 /* Local prototypes */
54
55 static void
56 AcpiDmI2cSerialBusDescriptor (
57     AML_RESOURCE *Resource,
58     UINT32 Length,
```

new/usr/src/common/acpica/components/disassembler/dmresrc12.c

2

```
61     UINT32 Level);
62
63 static void
64 AcpiDmSpiSerialBusDescriptor (
65     AML_RESOURCE *Resource,
66     UINT32 Length,
67     UINT32 Level);
68
69 static void
70 AcpiDmUartSerialBusDescriptor (
71     AML_RESOURCE *Resource,
72     UINT32 Length,
73     UINT32 Level);
74
75 static void
76 AcpiDmGpioCommon (
77     AML_RESOURCE *Resource,
78     UINT32 Level);
79
80 static void
81 AcpiDmDumpRawDataBuffer (
82     UINT8 *Buffer,
83     UINT32 Length,
84     UINT32 Level);
85
86 /* Dispatch table for the serial bus descriptors */
87
88 static ACPI_RESOURCE_HANDLER SerialBusResourceDispatch [] =
89 {
90     NULL,
91     AcpiDmI2cSerialBusDescriptor,
92     AcpiDmSpiSerialBusDescriptor,
93     AcpiDmUartSerialBusDescriptor
94 };
95
96
97
98 /*****
99 *
100 * FUNCTION: AcpiDmDumpRawDataBuffer
101 *
102 * PARAMETERS: Buffer - Pointer to the data bytes
103 *             Length - Length of the descriptor in bytes
104 *             Level - Current source code indentation level
105 *
106 * RETURN: None
107 *
108 * DESCRIPTION: Dump a data buffer as a RawDataBuffer() object. Used for
109 *             vendor data bytes.
110 *
111 *****/
112
113 static void
114 AcpiDmDumpRawDataBuffer (
115     UINT8 *Buffer,
116     UINT32 Length,
117     UINT32 Level)
118 {
119     UINT32 Index;
120     i;
121     j;
122
123     if (!Length)
124     {
125         return;
126     }
```

```

127     }
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; i++)
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%.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%.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%.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
260 *
261 *****/

263 static void
264 AcpiDmGpioIntDescriptor (
265     AML_RESOURCE      *Resource,
266     UINT32             Length,
267     UINT32             Level)
268 {
269     /* Dump the GpioInt-specific portion of the descriptor */
270
271     /* EdgeLevel, ActiveLevel, Shared */
272
273     AcpiDmIndent (Level);
274     AcpiOsPrintf ("GpioInt (%s, %s, %s, ",
275         AcpiGbl_HeDecode [ACPI_GET_1BIT_FLAG (Resource->Gpio.IntFlags)],
276         AcpiGbl_LlDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->Gpio.IntFlags, 1)],
277         AcpiGbl_ShrDecode [ACPI_EXTRACT_2BIT_FLAG (Resource->Gpio.IntFlags, 3)])
278
279     /* PinConfig, DebounceTimeout */
280
281     if (Resource->Gpio.PinConfig <= 3)
282     {
283         AcpiOsPrintf ("%s, ",
284             AcpiGbl_PpcDecode [Resource->Gpio.PinConfig]);
285     }
286     else
287     {
288         AcpiOsPrintf ("0x%2.2X, ", Resource->Gpio.PinConfig);
289     }
290     AcpiOsPrintf ("0x%4.4X,\n", Resource->Gpio.DebounceTimeout);
291
292     /* Dump the GpioInt/GpioIo common portion of the descriptor */
293
294     AcpiDmGpioCommon (Resource, Level);
295 }

299 /*****
300 *
301 * FUNCTION:      AcpiDmGpioIoDescriptor
302 *
303 * PARAMETERS:    Resource      - Pointer to the resource descriptor
304 *                Length        - Length of the descriptor in bytes
305 *                Level         - Current source code indentation level
306 *
307 * RETURN:        None
308 *
309 * DESCRIPTION:   Decode a GPIO I/O descriptor
310 *
311 *****/

313 static void
314 AcpiDmGpioIoDescriptor (
315     AML_RESOURCE      *Resource,
316     UINT32             Length,
317     UINT32             Level)
318 {
319     /* Dump the GpioIo-specific portion of the descriptor */
320
321     /* Shared, PinConfig */
322
323     AcpiDmIndent (Level);

```

```

325     AcpiOsPrintf ("GpioIo (%s, ",
326         AcpiGbl_ShrDecode [ACPI_EXTRACT_2BIT_FLAG (Resource->Gpio.IntFlags, 3)])
327
328     if (Resource->Gpio.PinConfig <= 3)
329     {
330         AcpiOsPrintf ("%s, ",
331             AcpiGbl_PpcDecode [Resource->Gpio.PinConfig]);
332     }
333     else
334     {
335         AcpiOsPrintf ("0x%2.2X, ", Resource->Gpio.PinConfig);
336     }
337
338     /* DebounceTimeout, DriveStrength, IoRestriction */
339
340     AcpiOsPrintf ("0x%4.4X, ", Resource->Gpio.DebounceTimeout);
341     AcpiOsPrintf ("0x%4.4X, ", Resource->Gpio.DriveStrength);
342     AcpiOsPrintf ("%s,\n",
343         AcpiGbl_IorDecode [ACPI_GET_2BIT_FLAG (Resource->Gpio.IntFlags)]);
344
345     /* Dump the GpioInt/GpioIo common portion of the descriptor */
346
347     AcpiDmGpioCommon (Resource, Level);
348 }

351 /*****
352 *
353 * FUNCTION:      AcpiDmGpioDescriptor
354 *
355 * PARAMETERS:    Resource      - Pointer to the resource descriptor
356 *                Length        - Length of the descriptor in bytes
357 *                Level         - Current source code indentation level
358 *
359 * RETURN:        None
360 *
361 * DESCRIPTION:   Decode a GpioInt/GpioIo GPIO Interrupt/IO descriptor
362 *
363 *****/

365 void
366 AcpiDmGpioDescriptor (
367     AML_RESOURCE      *Resource,
368     UINT32             Length,
369     UINT32             Level)
370 {
371     UINT8             ConnectionType;
372
373     ConnectionType = Resource->Gpio.ConnectionType;
374
375     switch (ConnectionType)
376     {
377     case AML_RESOURCE_GPIO_TYPE_INT:
378
379         AcpiDmGpioIntDescriptor (Resource, Length, Level);
380         break;
381
382     case AML_RESOURCE_GPIO_TYPE_IO:
383
384         AcpiDmGpioIoDescriptor (Resource, Length, Level);
385         break;
386
387     default:
388
389         AcpiOsPrintf ("Unknown GPIO type\n");

```

```

391     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 *****/
407
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 *****/
472
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 }

```

```

525 /*****
526 *
527 * FUNCTION:      AcpiDmSpiSerialBusDescriptor
528 *
529 * PARAMETERS:   Resource          - Pointer to the resource descriptor
530 *               Length            - Length of the descriptor in bytes
531 *               Level             - Current source code indentation level
532 *
533 * RETURN:       None
534 *
535 * DESCRIPTION:  Decode a SPI serial bus descriptor
536 *
537 *****/

539 static void
540 AcpiDmSpiSerialBusDescriptor (
541     AML_RESOURCE *Resource,
542     UINT32 Length,
543     UINT32 Level)
544 {
545     UINT32 ResourceSourceOffset;

548     /* DeviceSelection, DeviceSelectionPolarity, WireMode, DataBitLength */

550     AcpiDmIndent (Level);
551     AcpiOsPrintf ("SpiSerialBus (0x%4.4X, %s, %s, 0x%2.2X,\n",
552         Resource->SpiSerialBus.DeviceSelection,
553         AcpiGbl_DpDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->SpiSerialBus.TypeSpecifi
554         AcpiGbl_WmDecode [ACPI_GET_1BIT_FLAG (Resource->SpiSerialBus.TypeSpecifi
555         Resource->SpiSerialBus.DataBitLength);

557     /* SlaveMode, ConnectionSpeed, ClockPolarity, ClockPhase */

559     AcpiDmIndent (Level + 1);
560     AcpiOsPrintf ("%s, 0x%8.8X, %s,\n",
561         AcpiGbl_SmDecode [ACPI_GET_1BIT_FLAG (Resource->SpiSerialBus.Flags)],
562         Resource->SpiSerialBus.ConnectionSpeed,
563         AcpiGbl_CpoDecode [ACPI_GET_1BIT_FLAG (Resource->SpiSerialBus.ClockPolar

565     AcpiDmIndent (Level + 1);
566     AcpiOsPrintf ("%s, ",
567         AcpiGbl_CphDecode [ACPI_GET_1BIT_FLAG (Resource->SpiSerialBus.ClockPhase

569     /* ResourceSource is a required field */

571     ResourceSourceOffset = sizeof (AML_RESOURCE_COMMON_SERIALBUS) +
572         Resource->CommonSerialBus.TypeDataLength;

574     AcpiUtPrintString (
575         ACPI_ADD_PTR (char, Resource, ResourceSourceOffset),
576         ACPI_UINT16_MAX);

578     /* ResourceSourceIndex, ResourceUsage */

580     AcpiOsPrintf ("\n");
581     AcpiDmIndent (Level + 1);
582     AcpiOsPrintf ("0x%2.2X, ", Resource->SpiSerialBus.ResSourceIndex);

584     AcpiOsPrintf ("%s, ",
585         AcpiGbl_ConsumeDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->SpiSerialBus.Fl

587     /* Insert a descriptor name */

```

```

589     AcpiDmDescriptorName ();
590     AcpiOsPrintf ("\n");

592     /* Dump the vendor data */

594     AcpiDmIndent (Level + 1);
595     AcpiDmDumpSerialBusVendorData (Resource, Level);
596     AcpiOsPrintf ("\n");
597 }

600 /*****
601 *
602 * FUNCTION:      AcpiDmUartSerialBusDescriptor
603 *
604 * PARAMETERS:   Resource          - Pointer to the resource descriptor
605 *               Length            - Length of the descriptor in bytes
606 *               Level             - Current source code indentation level
607 *
608 * RETURN:       None
609 *
610 * DESCRIPTION:  Decode a UART serial bus descriptor
611 *
612 *****/

614 static void
615 AcpiDmUartSerialBusDescriptor (
616     AML_RESOURCE *Resource,
617     UINT32 Length,
618     UINT32 Level)
619 {
620     UINT32 ResourceSourceOffset;

623     /* ConnectionSpeed, BitsPerByte, StopBits */

625     AcpiDmIndent (Level);
626     AcpiOsPrintf ("UartSerialBus (0x%8.8X, %s, %s,\n",
627         Resource->UartSerialBus.DefaultBaudRate,
628         AcpiGbl_BpbDecode [ACPI_EXTRACT_3BIT_FLAG (Resource->UartSerialBus.TypeSp
629         AcpiGbl_SbDecode [ACPI_EXTRACT_2BIT_FLAG (Resource->UartSerialBus.TypeSp

631     /* LinesInUse, IsBigEndian, Parity, FlowControl */

633     AcpiDmIndent (Level + 1);
634     AcpiOsPrintf ("0x%2.2X, %s, %s, %s,\n",
635         Resource->UartSerialBus.LinesEnabled,
636         AcpiGbl_EdDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->UartSerialBus.TypeSp
637         AcpiGbl_PtDecode [ACPI_GET_3BIT_FLAG (Resource->UartSerialBus.Parity)],
638         AcpiGbl_FcDecode [ACPI_GET_2BIT_FLAG (Resource->UartSerialBus.TypeSpecifi

640     /* ReceiveBufferSize, TransmitBufferSize */

642     AcpiDmIndent (Level + 1);
643     AcpiOsPrintf ("0x%4.4X, 0x%4.4X, ",
644         Resource->UartSerialBus.RxFifoSize,
645         Resource->UartSerialBus.TxFifoSize);

647     /* ResourceSource is a required field */

649     ResourceSourceOffset = sizeof (AML_RESOURCE_COMMON_SERIALBUS) +
650         Resource->CommonSerialBus.TypeDataLength;

652     AcpiUtPrintString (
653         ACPI_ADD_PTR (char, Resource, ResourceSourceOffset),
654         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
```

new/usr/src/common/acpica/components/disassembler/dmresrcs.c

1

```
*****
10814 Fri Jan 17 07:29:54 2014
new/usr/src/common/acpica/components/disassembler/dmresrcs.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dmresrcs.c - "Small" Resource Descriptor disassembly
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 * 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 MERCHANTABILITY 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 "acdisasm.h"

50 #ifdef ACPI_DISASSEMBLER

52 #define _COMPONENT          ACPI_CA_DEBUGGER
53 #define ACPI_MODULE_NAME    ("dbresrcs")

56 /*****
57 *
58 * FUNCTION:      AcpiDmIrqDescriptor
```

new/usr/src/common/acpica/components/disassembler/dmresrcs.c

2

```
59 *
60 * PARAMETERS:    Resource          - Pointer to the resource descriptor
61 *                Length            - Length of the descriptor in bytes
62 *                Level             - Current source code indentation level
63 *
64 * RETURN:        None
65 *
66 * DESCRIPTION:    Decode a IRQ descriptor, either Irq() or IrqNoFlags()
67 *
68 *****/

70 void
71 AcpiDmIrqDescriptor (
72     AML_RESOURCE          *Resource,
73     UINT32                Length,
74     UINT32                Level)
75 {
77     AcpiDmIndent (Level);
78     AcpiOsPrintf ("%s (",
79         AcpiGbl_IrqDecode [ACPI_GET_1BIT_FLAG (Length)]);
80     AcpiGbl_IrqDecode [Length & 1]);

81     /* Decode flags byte if present */

82     if (Length & 1)
83     {
84         AcpiOsPrintf ("%s, %s, %s, ",
85             AcpiGbl_HeDecode [ACPI_GET_1BIT_FLAG (Resource->Irq.Flags)],
86             AcpiGbl_LlDecode [ACPI_EXTRACT_1BIT_FLAG (Resource->Irq.Flags, 3)],
87             AcpiGbl_ShrDecode [ACPI_EXTRACT_2BIT_FLAG (Resource->Irq.Flags, 4)],
88             AcpiGbl_HeDecode [(Resource->Irq.Flags & 1)],
89             AcpiGbl_LlDecode [(Resource->Irq.Flags >> 3) & 1],
90             AcpiGbl_ShrDecode [(Resource->Irq.Flags >> 4) & 1]);
91     }

92     /* Insert a descriptor name */

93     AcpiDmDescriptorName ();
94     AcpiOsPrintf ("")\n");

95     AcpiDmIndent (Level + 1);
96     AcpiDmBitList (Resource->Irq.IrqMask);
97 }

101 /*****
102 *
103 * FUNCTION:      AcpiDmDmaDescriptor
104 *
105 * PARAMETERS:    Resource          - Pointer to the resource descriptor
106 *                Length            - Length of the descriptor in bytes
107 *                Level             - Current source code indentation level
108 *
109 * RETURN:        None
110 *
111 * DESCRIPTION:    Decode a DMA descriptor
112 *
113 *****/

115 void
116 AcpiDmDmaDescriptor (
117     AML_RESOURCE          *Resource,
118     UINT32                Length,
119     UINT32                Level)
120 {
```



```

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)]);
205     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 }

224 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 Dependent 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
```

new/usr/src/common/acpica/components/disassembler/dmutils.c

1

```
*****
7799 Fri Jan 17 07:29:54 2014
new/usr/src/common/acpica/components/disassembler/dmutils.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dmutils - AML disassembler utilities
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #include "acpi.h"
46 #include "accommon.h"
47 #include "amlcode.h"
48 #include "acdisasm.h"
49
50 #ifdef ACPI_ASL_COMPILER
51 #include <acnamesp.h>
52 #endif
53
54 #ifdef ACPI_DISASSEMBLER
55
56 #define _COMPONENT ACPI_CA_DEBUGGER
57 #define ACPI_MODULE_NAME ("dmutils")
```

new/usr/src/common/acpica/components/disassembler/dmutils.c

2

```
60 /* Data used in keeping track of fields */
61 #if 0
62 const char *AcpiGbl_FENames[] =
63 {
64     "skip",
65     "?access?"
66 };
67 /* FE = Field Element */
68 #endif
69 #ifndef unchanged_portion_omitted
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130 /*****
131 *
132 * FUNCTION: AcpiDmDecodeAttribute
133 *
134 * PARAMETERS: Attribute - Attribute field of AccessAs keyword
135 *
136 * RETURN: None
137 *
138 * DESCRIPTION: Decode the AccessAs attribute byte. (Mostly SMBus and
139 * GenericSerialBus stuff.)
140 * DESCRIPTION: Decode the AccessAs attribute byte. (Mostly SMBus stuff)
141 *****/
142
143 void
144 AcpiDmDecodeAttribute (
145     UINT8 Attribute)
146 {
147     switch (Attribute)
148     {
149     case AML_FIELD_ATTRIB_QUICK:
150     case AML_FIELD_ATTRIB_SMB_QUICK:
151
152         AcpiOsPrintf ("AttribQuick");
153         AcpiOsPrintf ("SMBQuick");
154         break;
155
156     case AML_FIELD_ATTRIB_SEND_RCV:
157     case AML_FIELD_ATTRIB_SMB_SEND_RCV:
158
159         AcpiOsPrintf ("AttribSendReceive");
160         AcpiOsPrintf ("SMBSendReceive");
161         break;
162
163     case AML_FIELD_ATTRIB_BYTE:
164     case AML_FIELD_ATTRIB_SMB_BYTE:
165
166         AcpiOsPrintf ("AttribByte");
167         AcpiOsPrintf ("SMBByte");
168         break;
169
170     case AML_FIELD_ATTRIB_WORD:
171     case AML_FIELD_ATTRIB_SMB_WORD:
172
173         AcpiOsPrintf ("AttribWord");
174         AcpiOsPrintf ("SMBWord");
175         break;
176
177     case AML_FIELD_ATTRIB_BLOCK:
178     case AML_FIELD_ATTRIB_SMB_WORD_CALL:
179
180         AcpiOsPrintf ("AttribBlock");
181         AcpiOsPrintf ("SMBProcessCall");
182         break;
183     }
```

```

175     case AML_FIELD_ATTRIB_MULTIBYTE:
176     case AML_FIELD_ATTRIB_SMB_BLOCK:
177         AcpiOsPrintf ("AttribBytes");
178         AcpiOsPrintf ("SMBBlock");
179         break;
180     case AML_FIELD_ATTRIB_WORD_CALL:
181     case AML_FIELD_ATTRIB_SMB_BLOCK_CALL:
182         AcpiOsPrintf ("AttribProcessCall");
183         AcpiOsPrintf ("SMBBlockProcessCall");
184         break;
185     case AML_FIELD_ATTRIB_BLOCK_CALL:
186         AcpiOsPrintf ("AttribBlockProcessCall");
187         break;
188     case AML_FIELD_ATTRIB_RAW_BYTES:
189         AcpiOsPrintf ("AttribRawBytes");
190         break;
191     case AML_FIELD_ATTRIB_RAW_PROCESS:
192         AcpiOsPrintf ("AttribRawProcessBytes");
193         break;
194     default:
195         /* A ByteConst is allowed by the grammar */
196         AcpiOsPrintf ("0x%2.2X", Attribute);
197         AcpiOsPrintf ("0x%.2X", Attribute);
198         break;
199     }
200 }
201
202
203
204
205
206
207 }
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

259     {
260         /* Check for a NULL target operand */
261
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);
274                 return FALSE;
275             }
276
277             if ((Op->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST) &&
278                 (!Op->Common.Next->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST))
279             {
280                 return (FALSE);
281                 return FALSE;
282             }
283
284             AcpiOsPrintf (" ");
285             return (TRUE);
286         }
287
288         else if ((Op->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST) &&
289                 (Op->Common.Next->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST))
290         {
291             AcpiOsPrintf (" ");
292             return (TRUE);
293         }
294
295         return (FALSE);
296     }
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

new/usr/src/common/acpica/components/disassembler/dmwalk.c

1

```
*****
26408 Fri Jan 17 07:29:55 2014
new/usr/src/common/acpica/components/disassembler/dmwalk.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dmwalk - AML disassembly tree walk
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "acparser.h"
48 #include "amlcode.h"
49 #include "acdisasm.h"
50 #include "acdebug.h"

53 #ifdef ACPI_DISASSEMBLER

55 #define _COMPONENT ACPI_CA_DEBUGGER
56 #define ACPI_MODULE_NAME ("dmwalk")
```

new/usr/src/common/acpica/components/disassembler/dmwalk.c

2

```
59 #define DB_FULL_OP_INFO "[%4.4s] @%5.5X #%4.4X: "

61 /* Stub for non-compiler code */

63 #ifndef ACPI_AS_L_COMPILER
64 void
65 AcpiDmEmitExternals (
66     void)
67 {
68     return;
69 }

    unchanged_portion_omitted

378 /*****
379 *
380 * FUNCTION: AcpiDmDescendingOp
381 *
382 * PARAMETERS: ASL_WALK_CALLBACK
383 *
384 * RETURN: Status
385 *
386 * DESCRIPTION: First visitation of a parse object during tree descent.
387 * Decode opcode name and begin parameter list(s), if any.
388 *
389 *****/

391 static ACPI_STATUS
392 AcpiDmDescendingOp (
393     ACPI_PARSE_OBJECT *Op,
394     UINT32 Level,
395     void *Context)
396 {
397     ACPI_OP_WALK_INFO *Info = Context;
398     const ACPI_OPCODE_INFO *OpInfo;
399     UINT32 Name;
400     ACPI_PARSE_OBJECT *NextOp;

403     if (Op->Common.DisasmFlags & ACPI_PARSEOP_IGNORE)
404     {
405         /* Ignore this op -- it was handled elsewhere */

407         return (AE_CTRL_DEPTH);
408     }

410     /* Level 0 is at the Definition Block level */

412     if (Level == 0)
413     {
414         /* In verbose mode, print the AML offset, opcode and depth count */

416         if (Info->WalkState)
417         {
418             VERBOSE_PRINT ((DB_FULL_OP_INFO,
419                 (Info->WalkState->MethodNode ?
420                     Info->WalkState->MethodNode->Name.Ascii : " "),
421                     Op->Common.AmlOffset, (UINT32) Op->Common.AmlOpcode));
422         }

424         if (Op->Common.AmlOpcode == AML_SCOPE_OP)
425         {
426             /* This is the beginning of the Definition Block */

428             AcpiOsPrintf ("{\n");
```

```

430      /* Emit all External() declarations here */
431
432      AcpiDmEmitExternals ();
433      return (AE_OK);
434  }
435  }
436  else if ((AcpiDmBlockType (Op->Common.Parent) & BLOCK_BRACE) &&
437      (! (Op->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST)) &&
438      (Op->Common.AmlOpcode != AML_INT_BYTELIST_OP))
439  {
440      /*
441       * This is a first-level element of a term list,
442       * indent a new line
443       */
444      switch (Op->Common.AmlOpcode)
445      {
446      case AML_NOOP_OP:
447          /*
448           * Optionally just ignore this opcode. Some tables use
449           * NoOp opcodes for "padding" out packages that the BIOS
450           * changes dynamically. This can leave hundreds or
451           * thousands of NoOp opcodes that if disassembled,
452           * cannot be compiled because they are syntactically
453           * incorrect.
454           */
455          if (AcpiGbl_IgnoreNoopOperator)
456          {
457              Op->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;
458              return (AE_OK);
459          }
460
461          /* Fallthrough */
462
463      default:
464          AcpiDmIndent (Level);
465          break;
466      }
467
468      Info->LastLevel = Level;
469      Info->Count = 0;
470  }
471  }
472
473  /*
474   * This is an inexpensive mechanism to try and keep lines from getting
475   * too long. When the limit is hit, start a new line at the previous
476   * indent plus one. A better but more expensive mechanism would be to
477   * keep track of the current column.
478   */
479  Info->Count++;
480  if (Info->Count /* +Info->LastLevel */ > 10)
481  if (Info->Count /* +Info->LastLevel */ > 10)
482  {
483      Info->Count = 0;
484      AcpiOsPrintf ("\n");
485      AcpiDmIndent (Info->LastLevel + 1);
486  }
487
488  /* Print the opcode name */
489  AcpiDmDisassembleOneOp (NULL, Info, Op);
490
491  if ((Op->Common.DisasmOpcode == ACPI_DASM_LNOT_PREFIX) ||
492      (Op->Common.AmlOpcode == AML_INT_CONNECTION_OP))
493  if (Op->Common.DisasmOpcode == ACPI_DASM_LNOT_PREFIX)
494  {

```

```

494      return (AE_OK);
495  }
496
497  if ((Op->Common.AmlOpcode == AML_NAME_OP) ||
498      (Op->Common.AmlOpcode == AML_RETURN_OP))
499  {
500      Info->Level--;
501  }
502
503  /* Start the opcode argument list if necessary */
504
505  OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
506
507  if ((OpInfo->Flags & AML_HAS_ARGS) ||
508      (Op->Common.AmlOpcode == AML_EVENT_OP))
509  {
510      /* This opcode has an argument list */
511
512      if (AcpiDmBlockType (Op) & BLOCK_PAREN)
513      {
514          AcpiOsPrintf (" (");
515      }
516
517      /* If this is a named opcode, print the associated name value */
518
519      if (OpInfo->Flags & AML_NAMED)
520      {
521          switch (Op->Common.AmlOpcode)
522          {
523          case AML_ALIAS_OP:
524
525              NextOp = AcpiPsGetDepthNext (NULL, Op);
526              NextOp->Common.DisasmFlags |= ACPI_PARSEOP_IGNORE;
527              AcpiDmNamestring (NextOp->Common.Value.Name);
528              AcpiOsPrintf (" ", " ");
529
530              /*lint -fallthrough */
531
532          default:
533
534              Name = AcpiPsGetName (Op);
535              if (Op->Named.Path)
536              {
537                  AcpiDmNamestring ((char *) Op->Named.Path);
538              }
539              else
540              {
541                  AcpiDmDumpName (Name);
542              }
543
544              if (Op->Common.AmlOpcode != AML_INT_NAMEDFIELD_OP)
545              {
546                  if (AcpiGbl_DbOpt_verbose)
547                  {
548                      (void) AcpiPsDisplayObjectPathname (NULL, Op);
549                  }
550              }
551              break;
552          }
553
554      switch (Op->Common.AmlOpcode)
555      {
556      case AML_METHOD_OP:
557
558          AcpiDmMethodFlags (Op);
559          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) */
819
820         AcpiOsPrintf ("\n\n");
821         return (AE_OK);
822     }
823
824     switch (AcpiDmBlockType (Op))
825     {
826     case BLOCK_PAREN:
827
828         /* Completed an op that has arguments, add closing paren */
829
830         AcpiOsPrintf (");");
831
832         if (Op->Common.AmlOpcode == AML_NAME_OP)
833         {
834             /* Emit description comment for Name() with a predefined ACPI name */
835
836             AcpiDmPredefinedDescription (Op);
837         }
838         else
839         {
840             /* For Create* operators, attempt to emit resource tag description */
841
842             AcpiDmFieldPredefinedDescription (Op);
843         }
844
845         /* Could be a nested operator, check if comma required */
846
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;
864
865     case BLOCK_BRACE:
866     case (BLOCK_BRACE | BLOCK_PAREN):
867
868         /* Completed an op that has a term list, add closing brace */
869
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);
881
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             }
893
894             if ((AcpiDmBlockType (Op->Common.Parent) & BLOCK_BRACE) &&
895                 (!Op->Common.Next))
896             {
897                 break;
898             }
899             AcpiOsPrintf ("\n");
900         }
901     }
902     break;
903
904     case BLOCK_NONE:
905     default:
906
907         /* Could be a nested operator, check if comma required */
908
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:
928
929                 if (!(Op->Common.DisasmFlags & ACPI_PARSEOP_PARAMLIST))
930                 {
931                     AcpiOsPrintf ("\n");
932                 }
933                 break;
934
935             default:
936
937                 break;
938             }
939         }
940         break;
941     }
942
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             AcpiOsPrintf ("\n");
970             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

new/usr/src/common/acpica/components/dispatcher/dsargs.c

1

```
*****
12480 Fri Jan 17 07:29:55 2014
new/usr/src/common/acpica/components/dispatcher/dsargs.c
update to acpica-unix2-20140114
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 - 2014, 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 MERCHANTABILITY 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 */
```

new/usr/src/common/acpica/components/dispatcher/dsargs.c

2

```
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

new/usr/src/common/acpica/components/dispatcher/dscontrol.c

1

```
*****
12954 Fri Jan 17 07:29:55 2014
new/usr/src/common/acpica/components/dispatcher/dscontrol.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dscontrol - Support for execution control opcodes -
4 *                               if/else/while/return
5 *
6 *****/
7
8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
45 #define __DSCONTROL_C__
46
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "amlcode.h"
50 #include "acdispat.h"
51 #include "acinterp.h"
52
53 #define _COMPONENT ACPI_DISPATCHER
54         ACPI_MODULE_NAME("dscontrol")
55
56 /*****
57 *
58 */
```

new/usr/src/common/acpica/components/dispatcher/dscontrol.c

2

```
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 *****/
70
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;
78
79     ACPI_FUNCTION_NAME (DsExecBeginControlOp);
80
81     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Op=%p Opcode=%2.2X State=%p\n",
82         Op, Op->Common.AmlOpcode, WalkState));
83
84     switch (Op->Common.AmlOpcode)
85     {
86     case AML_WHILE_OP:
87
88         /*
89          * If this is an additional iteration of a while loop, continue.
90          * There is no need to allocate a new control state.
91          */
92         if (WalkState->ControlState)
93         {
94             if (WalkState->ControlState->Control.AmlPredicateStart ==
95                 (WalkState->ParserState.Aml - 1))
96             {
97                 /* Reset the state to start-of-loop */
98
99                 WalkState->ControlState->Common.State =
100                     ACPI_CONTROL_CONDITIONAL_EXECUTING;
101                 break;
102             }
103         }
104
105         /*lint -fallthrough */
106
107     case AML_IF_OP:
108
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     }
```

```

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:
253
254         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
255             "[RETURN_OP] Op=%p Arg=%p\n", Op, Op->Common.Value.Arg));
256
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 */
265
266             AcpiDsClearImplicitReturn (WalkState);
267
268             /* Return statement has an immediate operand */
269
270             Status = AcpiDsCreateOperands (WalkState, Op->Common.Value.Arg);
271             if (ACPI_FAILURE (Status))
272             {
273                 return (Status);
274             }
275
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             }
286
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 */
297
298             AcpiDsClearImplicitReturn (WalkState);
299
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     }
320
321     WalkState->ReturnDesc = WalkState->Results->Results.ObjDesc [0];
322 }
323 else
324 {
325     /* No return operand */
326
327     if (WalkState->NumOperands)
328     {
329         AcpiUtRemoveReference (WalkState->Operands [0]);
330     }
331
332     WalkState->Operands [0] = NULL;
333     WalkState->NumOperands = 0;
334     WalkState->ReturnDesc = NULL;
335 }
336
337 ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
338     "Completed RETURN_OP State=%p, RetVal=%p\n",
339     WalkState, WalkState->ReturnDesc));
340
341 /* End the control method execution right now */
342
343 Status = AE_CTRL_TERMINATE;
344 break;
345
346 case AML_NOOP_OP:
347
348     /* Just do nothing! */
349
350     break;
351
352 case AML_BREAK_POINT_OP:
353
354     /*
355      * Set the single-step flag. This will cause the debugger (if present)
356      * to break to the console within the AML debugger at the start of the
357      * next AML instruction.
358      */
359     ACPI_DEBUGGER_EXEC (
360         AcpiGbl_CmSingleStep = TRUE);
361     ACPI_DEBUGGER_EXEC (
362         AcpiOsPrintf ("**break** Executed AML BreakPoint opcode\n"));
363
364     /* Call to the OSL in case OS wants a piece of the action */
365
366     Status = AcpiOsSignal (ACPI_SIGNAL_BREAKPOINT,
367         "Executed AML Breakpoint opcode");
368     break;
369
370 case AML_BREAK_OP:
371 case AML_CONTINUE_OP: /* ACPI 2.0 */
372
373     /* Pop and delete control states until we find a while */
374
375     while (WalkState->ControlState &&
376         (WalkState->ControlState->Control.Opcode != AML_WHILE_OP))
377     {
378         ControlState = AcpiUtPopGenericState (&WalkState->ControlState);

```

```
380         AcpiUtDeleteGenericState (ControlState);
381     }
382
383     /* No while found? */
384
385     if (!WalkState->ControlState)
386     {
387         return (AE_AML_NO_WHILE);
388     }
389
390     /* Was: WalkState->AmlLastWhile = WalkState->ControlState->Control.AmlPr
391
392     WalkState->AmlLastWhile = WalkState->ControlState->Control.PackageEnd;
393
394     /* Return status depending on opcode */
395
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;
405
406     default:
407
408         ACPI_ERROR ((AE_INFO, "Unknown control opcode=0x%X Op=%p",
409             Op->Common.AmlOpcode, Op));
410
411         Status = AE_AML_BAD_OPCODE;
412         break;
413     }
414
415     return (Status);
416 }
```

unchanged\_portion\_omitted



new/usr/src/common/acpica/components/dispatcher/dsfield.c

1

```
*****
26047 Fri Jan 17 07:29:56 2014
new/usr/src/common/acpica/components/dispatcher/dsfield.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dsfield - Dispatcher field routines
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __DSFIELD_C__
45
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"
53
54 #define _COMPONENT ACPI_DISPATCHER
55 #define ACPI_MODULE_NAME ("dsfield")
```

new/usr/src/common/acpica/components/dispatcher/dsfield.c

2

```
58 /* Local prototypes */
59
60 #ifdef ACPI_AS_L_COMPILER
61 #include "acdisasm.h"
62
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
71
72 static ACPI_STATUS
73 AcpiDsGetFieldNames (
74     ACPI_CREATE_FIELD_INFO *Info,
75     ACPI_WALK_STATE  *WalkState,
76     ACPI_PARSE_OBJECT *Arg);
77
78 #ifdef ACPI_AS_L_COMPILER
79 /*****
80 *
81 * FUNCTION: AcpiDsCreateExternalRegion (iASL Disassembler only)
82 *
83 * PARAMETERS: LookupStatus - Status from NsLookup operation
84 *              Op           - Op containing the Field definition and args
85 *              Path         - Pathname of the region
86 *              WalkState    - Current method state
87 *              Node         - Where the new region node is returned
88 *
89 * RETURN:      Status
90 *
91 * DESCRIPTION: Add region to the external list if NOT_FOUND. Create a new
92 *              region node/object.
93 *
94 * *****/
95
96 static ACPI_STATUS
97 AcpiDsCreateExternalRegion (
98     ACPI_STATUS      LookupStatus,
99     ACPI_PARSE_OBJECT *Op,
100     char             *Path,
101     ACPI_WALK_STATE  *WalkState,
102     ACPI_NAMESPACE_NODE **Node)
103 {
104     ACPI_STATUS      Status;
105     ACPI_OPERAND_OBJECT *ObjDesc;
106
107     if (LookupStatus != AE_NOT_FOUND)
108     {
109         return (LookupStatus);
110     }
111
112     /*
113      * Table disassembly:
114      * OperationRegion not found. Generate an External for it, and
115      * insert the name into the namespace.
116      */
117     AcpiDmAddOpToExternalList (Op, Path, ACPI_TYPE_REGION, 0, 0);
118     Status = AcpiNsLookup (WalkState->ScopeInfo, Path, ACPI_TYPE_REGION,
119         ACPI_IMODE_LOAD_PASS1, ACPI_NS_SEARCH_PARENT, WalkState, Node);
120     if (ACPI_FAILURE (Status))
121     {
122     }
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  * FUNCTION:      AcpiDsCreateBufferField
144  *
145  * PARAMETERS:    Op                - Current parse op (CreateXXField)
146  *                WalkState          - Current state
147  *
148  * RETURN:        Status
149  *
150  * DESCRIPTION:    Execute the CreateField operators:
151  *                CreateBitFieldOp,
152  *                CreateByteFieldOp,
153  *                CreateWordFieldOp,
154  *                CreateDwordFieldOp,
155  *                CreateQwordFieldOp,
156  *                CreateDwordFieldOp,
157  *                CreateQwordFieldOp,
158  *                CreateFieldOp      (all of which define a field in a buffer)
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 CreateXXField 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
341          * Three types of field elements are handled:
342          * 1) Offset - specifies a bit offset
343          * 2) AccessAs - changes the access mode
344          * 3) Name - Enters a new named field into the namespace
345          */
346         switch (Arg->Common.AmlOpcode)
347         {
348             case AML_INT_RESERVEDFIELD_OP:

349                 Position = (UINT64) Info->FieldBitPosition
350                     + (UINT64) Arg->Common.Value.Size;

351                 if (Position > ACPI_UINT32_MAX)
352                 {
353                     ACPI_ERROR ((AE_INFO,
354                         "Bit offset within field too large (> 0xFFFFFFFF)"));
355                     return ACPI_STATUS (AE_SUPPORT);
356                 }

357                 Info->FieldBitPosition = (UINT32) Position;
358                 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.
367                  * Get a new AccessType and AccessAttribute -- to be used for all
368                  * field units that follow, until field end or another AccessAs
369                  * keyword.
370                  */
371                 In FieldFlags, preserve the flag bits other than the
372                 ACCESS_TYPE bits.
373                 ACCESS_TYPE bits
374                 */

375                 /* AccessType (ByteAcc, WordAcc, etc.) */

376                 Info->FieldFlags = (UINT8)
377                     ((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      * FUNCTION:      AcpiDsCreateField
487      *
488      * PARAMETERS:    Op          - Op containing the Field definition and args
489      *                  RegionNode - Object for the containing Operation Region
490      *                  WalkState  - Current method state
491      *
492      * RETURN:        Status
493      *
494      * DESCRIPTION:    Create a new field in the specified operation region
495      *
496      *****/

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;
506     ACPI_PARSE_OBJECT *Arg;
507     ACPI_CREATE_FIELD_INFO Info;

510     ACPI_FUNCTION_TRACE_PTR (DsCreateField, Op);

513     /* First arg is the name of the parent OpRegion (must already exist) */
515     Arg = Op->Common.Value.Arg;

517     if (!RegionNode)
518     {
519         Status = AcpiNsLookup (WalkState->ScopeInfo, Arg->Common.Value.Name,
520                               ACPI_TYPE_REGION, ACPI_IMODE_EXECUTE,
521                               ACPI_NS_SEARCH_PARENT, WalkState, &RegionNode);
522 #ifndef ACPI_ASL_COMPILER
523         Status = AcpiDsCreateExternalRegion (Status, Arg,
524                                              Arg->Common.Value.Name, WalkState, &RegionNode);
525 #endif
526         if (ACPI_FAILURE (Status))
527         {
528             ACPI_ERROR_NAMESPACE (Arg->Common.Value.Name, Status);
529             return_ACPI_STATUS (Status);
530         }
531     }

533     ACPI_MEMSET (&Info, 0, sizeof (ACPI_CREATE_FIELD_INFO));

535     /* Second arg is the field flags */

537     Arg = Arg->Common.Next;
538     Info.FieldFlags = (UINT8) Arg->Common.Value.Integer;
539     Info.Attribute = 0;

541     /* Each remaining arg is a Named Field */

543     Info.FieldType = ACPI_TYPE_LOCAL_REGION_FIELD;
544     Info.RegionNode = RegionNode;

546     Status = AcpiDsGetFieldNames (&Info, WalkState, Arg->Common.Next);

547     return_ACPI_STATUS (Status);
548 }

551 /*****
552  *
553  * FUNCTION:      AcpiDsInitFieldObjects
554  *
555  * PARAMETERS:    Op          - Op containing the Field definition and args
556  *                WalkState    - Current method state
557  *
558  * RETURN:        Status
559  *
560  * DESCRIPTION:    For each "Field Unit" name in the argument list that is
561  *                part of the field declaration, enter the name into the
562  *                namespace.
563  *
564  *****/

566 ACPI_STATUS
567 AcpiDsInitFieldObjects (
568     ACPI_PARSE_OBJECT *Op,
569     ACPI_WALK_STATE *WalkState)

```

```

570 {
571     ACPI_STATUS      Status;
572     ACPI_PARSE_OBJECT *Arg = NULL;
573     ACPI_NAMESPACE_NODE *Node;
574     UINT8             Type = 0;
575     UINT32            Flags;

578     ACPI_FUNCTION_TRACE_PTR (DsInitFieldObjects, Op);

581     /* Execute flag should always be set when this function is entered */

583     if (!(WalkState->ParseFlags & ACPI_PARSE_EXECUTE))
584     {
585         if (WalkState->ParseFlags & ACPI_PARSE_DEFERRED_OP)
586         {
587             /* BankField Op is deferred, just return OK */

589             return_ACPI_STATUS (AE_OK);
590         }

592         return_ACPI_STATUS (AE_AML_INTERNAL);
593     }

595     /*
596     * Get the FieldList argument for this opcode. This is the start of the
597     * list of field elements.
598     */
599     switch (WalkState->Opcode)
600     {
601     case AML_FIELD_OP:

603         Arg = AcpiPsGetArg (Op, 2);
604         Type = ACPI_TYPE_LOCAL_REGION_FIELD;
605         break;

607     case AML_BANK_FIELD_OP:

609         Arg = AcpiPsGetArg (Op, 4);
610         Type = ACPI_TYPE_LOCAL_BANK_FIELD;
611         break;

613     case AML_INDEX_FIELD_OP:

615         Arg = AcpiPsGetArg (Op, 3);
616         Type = ACPI_TYPE_LOCAL_INDEX_FIELD;
617         break;

619     default:

621         return_ACPI_STATUS (AE_BAD_PARAMETER);
622     }

624     /* Creating new namespace node(s), should not already exist */

626     Flags = ACPI_NS_NO_UPSEARCH | ACPI_NS_DONT_OPEN_SCOPE |
627            ACPI_NS_ERROR_IF_FOUND;

629     /*
630     * Mark node(s) temporary if we are executing a normal control
631     * method. (Don't mark if this is a module-level code method)
632     */
633     if (WalkState->MethodNode &&
634         !(WalkState->ParseFlags & ACPI_PARSE_MODULE_LEVEL))
635     {

```

```

636     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.
648      * Ignore OFFSET and ACCESSAS terms here; we are only interested in the
649      * field names in order to enter them into the namespace.
650      */
651     if (Arg->Common.AmlOpcode == AML_INT_NAMEDFIELD_OP)
652     {
653         Status = AcpiNsLookup (WalkState->ScopeInfo,
654             (char *) &Arg->Named.Name, Type, ACPI_IMODE_LOAD_PASS1,
655             Flags, WalkState, &Node);
656         if (ACPI_FAILURE (Status))
657         {
658             ACPI_ERROR_NAMESPACE ((char *) &Arg->Named.Name, Status);
659             if (Status != AE_ALREADY_EXISTS)
660             {
661                 return ACPI_STATUS (Status);
662             }
663             /* Name already exists, just ignore this error */
664             Status = AE_OK;
665         }
666         Arg->Common.Node = Node;
667     }
668     /* Get the next field element in the list */
669     Arg = Arg->Common.Next;
670 }
671 return ACPI_STATUS (AE_OK);
672 }

673 /*****
674  * FUNCTION:      AcpiDsCreateBankField
675  * PARAMETERS:    Op          - Op containing the Field definition and args
676  *                RegionNode  - Object for the containing Operation Region
677  *                WalkState   - Current method state
678  * RETURN:        Status
679  * DESCRIPTION:   Create a new bank field in the specified operation region
680  *****/
681
682 ACPI_STATUS
683 AcpiDsCreateBankField (
684     ACPI_PARSE_OBJECT *Op,
685     ACPI_NAMESPACE_NODE *RegionNode,
686     ACPI_WALK_STATE *WalkState)
687 {
688     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_AS_L_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 }

```

new/usr/src/common/acpica/components/dispatcher/dsinit.c

1

```
*****
7212 Fri Jan 17 07:29:56 2014
new/usr/src/common/acpica/components/dispatcher/dsinit.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dsinit - Object initialization namespace walk
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __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 (
```

new/usr/src/common/acpica/components/dispatcher/dsinit.c

2

```
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]",
```



new/usr/src/common/acpica/components/dispatcher/dsinit.c

3

```
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

new/usr/src/common/acpica/components/dispatcher/dsmethod.c

1

```
*****
23030 Fri Jan 17 07:29:57 2014
new/usr/src/common/acpica/components/dispatcher/dsmethod.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dsmethod - Parser/Interpreter interface - control method parsing
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __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 */
```

new/usr/src/common/acpica/components/dispatcher/dsmethod.c

2

```
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 #ifndef 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;
415     }
416     return ACPI_STATUS (AE_NO_MEMORY);
417
418     Info->Parameters = &ThisWalkState->Operands[0];
419
420     Status = AcpiDsInitAmlWalk (NextWalkState, NULL, MethodNode,
421                               ObjDesc->Method.AmlStart, ObjDesc->Method.AmlLength,
422                               Info, ACPI_IMODE_EXECUTE);
423
424     ACPI_FREE (Info);
425     if (ACPI_FAILURE (Status))
426     {
427         goto Cleanup;
428     }
429
430     /*
431     * Delete the operands on the previous walkstate operand stack
432     * (they were copied to new objects)
433     */
434     for (i = 0; i < ObjDesc->Method.ParamCount; i++)
435     {
436         AcpiUtRemoveReference (ThisWalkState->Operands [i]);
437         ThisWalkState->Operands [i] = NULL;
438     }
439
440     /* Clear the operand stack */
441     ThisWalkState->NumOperands = 0;
442
443     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
444                       "**** Begin nested execution of [%4.4s] **** WalkState=%p\n",
445                       MethodNode->Name.Ascii, NextWalkState));
446
447     /* Invoke an internal method if necessary */
448
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     }
457
458     return ACPI_STATUS (Status);
459
460 Cleanup:
461
462     /* On error, we must terminate the method properly */
463
464     AcpiDsTerminateControlMethod (ObjDesc, NextWalkState);
465     if (NextWalkState)
466     {
467         AcpiDsDeleteWalkState (NextWalkState);
468     }
469
470     return ACPI_STATUS (Status);
471 }
472 }

```

unchanged portion omitted

new/usr/src/common/acpica/components/dispatcher/dsmthdat.c

1

```
*****
23466 Fri Jan 17 07:29:57 2014
new/usr/src/common/acpica/components/dispatcher/dsmthdat.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dsmthdat - control method arguments and local variables
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __DSMTHDAT_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acdispat.h"
49 #include "acnamesp.h"
50 #include "acinterp.h"

53 #define _COMPONENT          ACPI_DISPATCHER
54       ACPI_MODULE_NAME    ("dsmthdat")

56 /* Local prototypes */

58 static void
```

new/usr/src/common/acpica/components/dispatcher/dsmthdat.c

2

```
59 AcpiDsMethodDataDeleteValue (
60     UINT8                Type,
61     UINT32                Index,
62     ACPI_WALK_STATE       *WalkState);

64 static ACPI_STATUS
65 AcpiDsMethodDataSetValue (
66     UINT8                Type,
67     UINT32                Index,
68     ACPI_OPERAND_OBJECT  *Object,
69     ACPI_WALK_STATE       *WalkState);

71 #ifdef ACPI_OBSOLETE_FUNCTIONS
72 ACPI_OBJECT_TYPE
73 AcpiDsMethodDataGetType (
74     UINT16                Opcode,
75     UINT32                Index,
76     ACPI_WALK_STATE       *WalkState);
77 #endif

80 /*****
81 *
82 * FUNCTION:      AcpiDsMethodDataInit
83 *
84 * PARAMETERS:    WalkState          - Current walk state object
85 *
86 * RETURN:        Status
87 *
88 * DESCRIPTION:   Initialize the data structures that hold the method's arguments
89 * and locals. The data struct is an array of namespace nodes for
90 * each - this allows RefOf and DeRefOf to work properly for these
91 * special data types.
92 *
93 * NOTES:         WalkState fields are initialized to zero by the
94 * ACPI_ALLOCATE_ZEROED().
95 *
96 *               A pseudo-Namespace Node is assigned to each argument and local
97 *               so that RefOf() can return a pointer to the Node.
98 *
99 *****/

101 void
102 AcpiDsMethodDataInit (
103     ACPI_WALK_STATE       *WalkState)
104 {
105     UINT32                i;

108     ACPI_FUNCTION_TRACE (DsMethodDataInit);

111     /* Init the method arguments */

113     for (i = 0; i < ACPI_METHOD_NUM_ARGS; i++)
114     {
115         ACPI_MOVE_32_TO_32 (&WalkState->Arguments[i].Name, NAMEOF_ARG_NTE);
116         WalkState->Arguments[i].Name.Integer |= (i << 24);
117         WalkState->Arguments[i].DescriptorType = ACPI_DESC_TYPE_NAMED;
118         WalkState->Arguments[i].Type = ACPI_TYPE_ANY;
119         WalkState->Arguments[i].Flags = ANOBJ_METHOD_ARG;
120     }

122     /* Init the method locals */

124     for (i = 0; i < ACPI_METHOD_NUM_LOCALS; i++)
```

```

125 {
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     }
433
434     /* Get the object from the node */
435
436     Object = Node->Object;
437
438     /* Examine the returned object, it must be valid. */
439
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          */
449
450         /* If slack enabled, init the LocalX/ArgX to an Integer of value zero */
451
452         if (AcpiGbl_EnableInterpreterSlack)
453         {
454             Object = AcpiUtCreateIntegerObject ((UINT64) 0);
455             if (!Object)
456             {
457                 return_ACPI_STATUS (AE_NO_MEMORY);
458             }
459
460             Node->Object = Object;
461         }
462
463         /* Otherwise, return the error */
464
465         else switch (Type)
466         {
467             case ACPI_REFCLASS_ARG:
468
469                 ACPI_ERROR ((AE_INFO,
470                     "Uninitialized Arg[%u] at node %p",
471                     Index, Node));
472
473                 return_ACPI_STATUS (AE_AML_UNINITIALIZED_ARG);
474
475             case ACPI_REFCLASS_LOCAL:
476
477                 /*
478                  * No error message for this case, will be trapped again later to
479                  * detect and ignore cases of Store(LocalX,LocalX)
480                  */
481                 return_ACPI_STATUS (AE_AML_UNINITIALIZED_LOCAL);
482
483             default:
484
485                 ACPI_ERROR ((AE_INFO, "Not a Arg/Local opcode: 0x%X", Type));
486                 return_ACPI_STATUS (AE_AML_INTERNAL);
487         }
488
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);
495
496         return_ACPI_STATUS (AE_OK);

```

```

497 }
    unchanged_portion_omitted
770 #endif

```

new/usr/src/common/acpica/components/dispatcher/dsobject.c

1

```
*****
28342 Fri Jan 17 07:29:58 2014
new/usr/src/common/acpica/components/dispatcher/dsobject.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dsobject - Dispatcher object management routines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __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 */
```

new/usr/src/common/acpica/components/dispatcher/dsobject.c

2

```
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->Common
111                                     if (ACPI_FAILURE (Status))
112                                     {
113                                         /* Check if we are resolving a named reference within a package
114
115                                         if ((Status == AE_NOT_FOUND) && (AcpiGbl_EnableInterpreterSlack)
116
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
126         * can be overwritten by subsequent ASL code - this is
127         * typically the case.
128         */
129         ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
130             "Ignoring unresolved reference in package [%4.4s]\n",
131             WalkState->ScopeInfo->Scope.Node->Name.Ascii));

133         return ACPI_STATUS (AE_OK);
134     }
135     else
136     {
137         ACPI_ERROR_NAMESPACE (Op->Common.Value.String, Status);
138     }

140     return ACPI_STATUS (Status);
141 }
142 }

144 /* Special object resolution for elements of a package */

146 if ((Op->Common.Parent->Common.AmlOpcode == AML_PACKAGE_OP) ||
147     (Op->Common.Parent->Common.AmlOpcode == AML_VAR_PACKAGE_OP))
148 {
149     /*
150     * Attempt to resolve the node to a value before we insert it into
151     * the package. If this is a reference to a common data type,
152     * resolve it immediately. According to the ACPI spec, package
153     * elements can only be "data objects" or method references.
154     * Attempt to resolve to an Integer, Buffer, String or Package.
155     * If cannot, return the named reference (for things like Devices,
156     * Methods, etc.) Buffer Fields and Fields will resolve to simple
157     * objects (int/buf/str/pkg).
158     *
159     * NOTE: References to things like Devices, Methods, Mutexes, etc.
160     * will remain as named references. This behavior is not described
161     * in the ACPI spec, but it appears to be an oversight.
162     */
163     ObjDesc = ACPI_CAST_PTR (ACPI_OPERAND_OBJECT, Op->Common.Node);

165     Status = AcpiExResolveNodeToValue (
166         ACPI_CAST_INDIRECT_PTR (ACPI_NAMESPACE_NODE, &ObjDesc),
167         WalkState);
168     if (ACPI_FAILURE (Status))
169     {
170         return ACPI_STATUS (Status);
171     }

173     /*
174     * Special handling for Alias objects. We need to setup the type
175     * and the Op->Common.Node to point to the Alias target. Note,
176     * Alias has at most one level of indirection internally.
177     */
178     Type = Op->Common.Node->Type;
179     if (Type == ACPI_TYPE_LOCAL_ALIAS)
180     {
181         Type = ObjDesc->Common.Type;
182         Op->Common.Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE,
183             Op->Common.Node->Object);
184     }

186     switch (Type)
187     {
188     /*
189     * For these types, we need the actual node, not the subobject.
190     * However, the subobject did not get an extra reference count above

```

```

191         *
192         * TBD: should ExResolveNodeToValue be changed to fix this?
193         */
194         case ACPI_TYPE_DEVICE:
195         case ACPI_TYPE_THERMAL:

197             AcpiUtAddReference (Op->Common.Node->Object);

199             /*lint -fallthrough */
200         /*
201         * For these types, we need the actual node, not the subobject..
202         * The subobject got an extra reference count in ExResolveNodeToValu
203         */
204         case ACPI_TYPE_MUTEX:
205         case ACPI_TYPE_METHOD:
206         case ACPI_TYPE_POWER:
207         case ACPI_TYPE_PROCESSOR:
208         case ACPI_TYPE_EVENT:
209         case ACPI_TYPE_REGION:

211             /* We will create a reference object for these types below */
212             break;

214         default:
215             /*
216             * All other types - the node was resolved to an actual
217             * object, we are done.
218             */
219             goto Exit;
220         }
221     }
222 }

224 /* Create and init a new internal ACPI object */

226 ObjDesc = AcpiUtCreateInternalObject (
227     (AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode))->ObjectType);
228 if (!ObjDesc)
229 {
230     return ACPI_STATUS (AE_NO_MEMORY);
231 }

233 Status = AcpiDsInitObjectFromOp (WalkState, Op, Op->Common.AmlOpcode,
234     &ObjDesc);
235 if (ACPI_FAILURE (Status))
236 {
237     AcpiUtRemoveReference (ObjDesc);
238     return ACPI_STATUS (Status);
239 }

241 Exit:
242 *ObjDescPtr = ObjDesc;
243 return ACPI_STATUS (Status);
244 }

    unchanged_portion_omitted

364 /*****
365  *
366  * FUNCTION:      AcpiDsBuildInternalPackageObj
367  *
368  * PARAMETERS:    WalkState      - Current walk state
369  *                Op              - Parser object to be translated
370  *                ElementCount    - Number of elements in the package - this is
371  *                                the NumElements argument to Package()
372  *                ObjDescPtr      - Where the ACPI internal object is returned

```

```

373 *
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 *****/
392
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;
407
408     ACPI_FUNCTION_TRACE (DsBuildInternalPackageObj);
409
410     /* Find the parent of a possibly nested package */
411
412     Parent = Op->Common.Parent;
413     while ((Parent->Common.AmlOpcode == AML_PACKAGE_OP) ||
414           (Parent->Common.AmlOpcode == AML_VAR_PACKAGE_OP))
415     {
416         Parent = Parent->Common.Parent;
417     }
418
419     /*
420     * If we are evaluating a Named package object "Name (xxxx, Package)",
421     * the package object already exists, otherwise it must be created.
422     */
423     ObjDesc = *ObjDescPtr;
424     if (!ObjDesc)
425     {
426         ObjDesc = AcpiUtCreateInternalObject (ACPI_TYPE_PACKAGE);
427         *ObjDescPtr = ObjDesc;
428         if (!ObjDesc)
429         {
430             return_ACPI_STATUS (AE_NO_MEMORY);
431         }
432     }
433
434     ObjDesc->Package.Node = Parent->Common.Node;
435 }
436
437 /*

```

```

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 *));
445
446 if (!ObjDesc->Package.Elements)
447 {
448     AcpiUtDeleteObjectDesc (ObjDesc);
449     return_ACPI_STATUS (AE_NO_MEMORY);
450 }
451
452 ObjDesc->Package.Count = ElementCount;
453
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 */
479
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     }
489 }
490
491 if (*ObjDescPtr)
492 {
493     /* Existing package, get existing reference count */
494
495     ReferenceCount = (*ObjDescPtr)->Common.ReferenceCount;
496     if (ReferenceCount > 1)
497     {
498         /* Make new element ref count match original ref count */
499
500         for (Index = 0; Index < (ReferenceCount - 1); Index++)
501         {
502             AcpiUtAddReference ((ObjDesc->Package.Elements[i]));
503         }
504     }
505 }

```

```

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;
714                 switch (Opcode)
715                 {
716                     case AML_ZERO_OP:
718                         ObjDesc->Integer.Value = 0;
719                         break;
721                     case AML_ONE_OP:
723                         ObjDesc->Integer.Value = 1;
724                         break;
726                     case AML_ONES_OP:
728                         ObjDesc->Integer.Value = ACPI_UINT64_MAX;
730                         /* Truncate value if we are executing from a 32-bit ACPI table */
732 #ifndef ACPI_NO_METHOD_EXECUTION
733                         (void) AcpiExTruncateFor32bitTable (ObjDesc);
734 #endif
735                         break;
737                     case AML_REVISION_OP:
739                         ObjDesc->Integer.Value = ACPI_CA_VERSION;
740                         break;
742                     default:
744                         ACPI_ERROR ((AE_INFO,
745                                     "Unknown constant opcode 0x%X", Opcode));
746                         Status = AE_AML_OPERAND_TYPE;
747                         break;
748                 }
749                 break;
751             case AML_TYPE_LITERAL:
753                 ObjDesc->Integer.Value = Op->Common.Value.Integer;
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 */
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     }
765     AcpiExTruncateFor32bitTable (ObjDesc);
766 #endif
767     break;
768
769 default:
770
771     default:
772         ACPI_ERROR ((AE_INFO, "Unknown Integer type 0x%X",
773             OpInfo->Type));
774         Status = AE_AML_OPERAND_TYPE;
775         break;
776     }
777     break;
778
779 case ACPI_TYPE_STRING:
780
781     ObjDesc->String.Pointer = Op->Common.Value.String;
782     ObjDesc->String.Length = (UINT32) ACPI_STRLen (Op->Common.Value.String);
783
784     /*
785      * The string is contained in the ACPI table, don't ever try
786      * to delete it
787      */
788     ObjDesc->Common.Flags |= APOBJ_STATIC_POINTER;
789     break;
790
791 case ACPI_TYPE_METHOD:
792     break;
793
794 case ACPI_TYPE_LOCAL_REFERENCE:
795
796     switch (OpInfo->Type)
797     {
798     case AML_TYPE_LOCAL_VARIABLE:
799
800         /* Local ID (0-7) is (AML opcode - base AML_LOCAL_OP) */
801
802         ObjDesc->Reference.Value = ((UINT32) Opcode) - AML_LOCAL_OP;
803         ObjDesc->Reference.Class = ACPI_REFCLASS_LOCAL;
804
805 #ifndef ACPI_NO_METHOD_EXECUTION
806         Status = AcpiDsMethodDataGetNode (ACPI_REFCLASS_LOCAL,
807             ObjDesc->Reference.Value, WalkState,
808             ACPI_CAST_INDIRECT_PTR (ACPI_NAMESPACE_NODE,
809                 &ObjDesc->Reference.Object));
810 #endif
811         break;
812
813 case AML_TYPE_METHOD_ARGUMENT:
814
815         /* Arg ID (0-6) is (AML opcode - base AML_ARG_OP) */
816
817         ObjDesc->Reference.Value = ((UINT32) Opcode) - AML_ARG_OP;
818         ObjDesc->Reference.Class = ACPI_REFCLASS_ARG;
819
820 #ifndef ACPI_NO_METHOD_EXECUTION
821         Status = AcpiDsMethodDataGetNode (ACPI_REFCLASS_ARG,
822             ObjDesc->Reference.Value, WalkState,
823             ACPI_CAST_INDIRECT_PTR (ACPI_NAMESPACE_NODE,
824                 &ObjDesc->Reference.Object));
825 #endif
826         break;
827     }
828 }
829
830 return (Status);
831 }
832
833 /*****************************************************************************
834 *
835 *  Function: AcpiDsMethodDataGetNode
836 *
837 *  Purpose: Get the node for a method argument or local variable.
838 *
839 *  Parameters:
840 *    RefClass - Reference class (ACPI_REFCLASS_LOCAL or ACPI_REFCLASS_ARG)
841 *    Value - Reference value (local ID or argument ID)
842 *    WalkState - Current walk state
843 *    Object - Pointer to the object (NULL if not found)
844 *
845 *  Returns:
846 *    Status - AE_OK if successful, AE_NOT_FOUND if not found,
847 *    or other error code.
848 *
849 *****************************************************************************/
850
851 ACPI_STATUS AcpiDsMethodDataGetNode (ACPI_REFCLASS ref_class,
852     UINT32 value, ACPI_WALK_STATE *walk_state,
853     ACPI_OBJECT **object)
854 {
855     ACPI_STATUS status;
856     ACPI_OBJECT *obj;
857     ACPI_NAMESPACE_NODE *node;
858     ACPI_METHOD_OBJECT *method_obj;
859     ACPI_LOCAL_VARIABLE *local_var;
860     ACPI_METHOD_ARGUMENT *method_arg;
861
862     /*
863      * Local variables are stored in the namespace, while
864      * method arguments are stored in the method object.
865      */
866     if (ref_class == ACPI_REFCLASS_LOCAL)
867     {
868         node = AcpiNsFindLocalVariable (value);
869         if (!node)
870             return (AE_NOT_FOUND);
871         local_var = (ACPI_LOCAL_VARIABLE *) node;
872         obj = local_var->Object;
873     }
874     else
875     {
876         method_obj = (ACPI_METHOD_OBJECT *) walk_state->MethodObj;
877         method_arg = (ACPI_METHOD_ARGUMENT *) method_obj->Arguments + value;
878         obj = method_arg->Object;
879     }
880
881     if (!obj)
882         return (AE_NOT_FOUND);
883
884     *object = obj;
885     return (AE_OK);
886 }
887
888 /*****************************************************************************
889 *
890 *  Function: AcpiDsMethodDataGetNode
891 *
892 *  Purpose: Get the node for a method argument or local variable.
893 *
894 *  Parameters:
895 *    RefClass - Reference class (ACPI_REFCLASS_LOCAL or ACPI_REFCLASS_ARG)
896 *    Value - Reference value (local ID or argument ID)
897 *    WalkState - Current walk state
898 *    Object - Pointer to the object (NULL if not found)
899 *
900 *  Returns:
901 *    Status - AE_OK if successful, AE_NOT_FOUND if not found,
902 *    or other error code.
903 *
904 *****************************************************************************/
905
906 ACPI_STATUS AcpiDsMethodDataGetNode (ACPI_REFCLASS ref_class,
907     UINT32 value, ACPI_WALK_STATE *walk_state,
908     ACPI_OBJECT **object)
909 {
910     ACPI_STATUS status;
911     ACPI_OBJECT *obj;
912     ACPI_NAMESPACE_NODE *node;
913     ACPI_METHOD_OBJECT *method_obj;
914     ACPI_LOCAL_VARIABLE *local_var;
915     ACPI_METHOD_ARGUMENT *method_arg;
916
917     /*
918      * Local variables are stored in the namespace, while
919      * method arguments are stored in the method object.
920      */
921     if (ref_class == ACPI_REFCLASS_LOCAL)
922     {
923         node = AcpiNsFindLocalVariable (value);
924         if (!node)
925             return (AE_NOT_FOUND);
926         local_var = (ACPI_LOCAL_VARIABLE *) node;
927         obj = local_var->Object;
928     }
929     else
930     {
931         method_obj = (ACPI_METHOD_OBJECT *) walk_state->MethodObj;
932         method_arg = (ACPI_METHOD_ARGUMENT *) method_obj->Arguments + value;
933         obj = method_arg->Object;
934     }
935
936     if (!obj)
937         return (AE_NOT_FOUND);
938
939     *object = obj;
940     return (AE_OK);
941 }
942
943 /*****************************************************************************
944 *
945 *  Function: AcpiDsMethodDataGetNode
946 *
947 *  Purpose: Get the node for a method argument or local variable.
948 *
949 *  Parameters:
950 *    RefClass - Reference class (ACPI_REFCLASS_LOCAL or ACPI_REFCLASS_ARG)
951 *    Value - Reference value (local ID or argument ID)
952 *    WalkState - Current walk state
953 *    Object - Pointer to the object (NULL if not found)
954 *
955 *  Returns:
956 *    Status - AE_OK if successful, AE_NOT_FOUND if not found,
957 *    or other error code.
958 *
959 *****************************************************************************/
960
961 ACPI_STATUS AcpiDsMethodDataGetNode (ACPI_REFCLASS ref_class,
962     UINT32 value, ACPI_WALK_STATE *walk_state,
963     ACPI_OBJECT **object)
964 {
965     ACPI_STATUS status;
966     ACPI_OBJECT *obj;
967     ACPI_NAMESPACE_NODE *node;
968     ACPI_METHOD_OBJECT *method_obj;
969     ACPI_LOCAL_VARIABLE *local_var;
970     ACPI_METHOD_ARGUMENT *method_arg;
971
972     /*
973      * Local variables are stored in the namespace, while
974      * method arguments are stored in the method object.
975      */
976     if (ref_class == ACPI_REFCLASS_LOCAL)
977     {
978         node = AcpiNsFindLocalVariable (value);
979         if (!node)
980             return (AE_NOT_FOUND);
981         local_var = (ACPI_LOCAL_VARIABLE *) node;
982         obj = local_var->Object;
983     }
984     else
985     {
986         method_obj = (ACPI_METHOD_OBJECT *) walk_state->MethodObj;
987         method_arg = (ACPI_METHOD_ARGUMENT *) method_obj->Arguments + value;
988         obj = method_arg->Object;
989     }
990
991     if (!obj)
992         return (AE_NOT_FOUND);
993
994     *object = obj;
995     return (AE_OK);
996 }
997
998 /*****************************************************************************
999 *
1000 *  Function: AcpiDsMethodDataGetNode
1001 *
1002 *  Purpose: Get the node for a method argument or local variable.
1003 *
1004 *  Parameters:
1005 *    RefClass - Reference class (ACPI_REFCLASS_LOCAL or ACPI_REFCLASS_ARG)
1006 *    Value - Reference value (local ID or argument ID)
1007 *    WalkState - Current walk state
1008 *    Object - Pointer to the object (NULL if not found)
1009 *
1010 *  Returns:
1011 *    Status - AE_OK if successful, AE_NOT_FOUND if not found,
1012 *    or other error code.
1013 *
1014 *****************************************************************************/
1015
1016 ACPI_STATUS AcpiDsMethodDataGetNode (ACPI_REFCLASS ref_class,
1017     UINT32 value, ACPI_WALK_STATE *walk_state,
1018     ACPI_OBJECT **object)
1019 {
1020     ACPI_STATUS status;
1021     ACPI_OBJECT *obj;
1022     ACPI_NAMESPACE_NODE *node;
1023     ACPI_METHOD_OBJECT *method_obj;
1024     ACPI_LOCAL_VARIABLE *local_var;
1025     ACPI_METHOD_ARGUMENT *method_arg;
1026
1027     /*
1028      * Local variables are stored in the namespace, while
1029      * method arguments are stored in the method object.
1030      */
1031     if (ref_class == ACPI_REFCLASS_LOCAL)
1032     {
1033         node = AcpiNsFindLocalVariable (value);
1034         if (!node)
1035             return (AE_NOT_FOUND);
1036         local_var = (ACPI_LOCAL_VARIABLE *) node;
1037         obj = local_var->Object;
1038     }
1039     else
1040     {
1041         method_obj = (ACPI_METHOD_OBJECT *) walk_state->MethodObj;
1042         method_arg = (ACPI_METHOD_ARGUMENT *) method_obj->Arguments + value;
1043         obj = method_arg->Object;
1044     }
1045
1046     if (!obj)
1047         return (AE_NOT_FOUND);
1048
1049     *object = obj;
1050     return (AE_OK);
1051 }
1052
1053 /*****************************************************************************
1054 *
1055 *  Function: AcpiDsMethodDataGetNode
1056 *
1057 *  Purpose: Get the node for a method argument or local variable.
1058 *
1059 *  Parameters:
1060 *    RefClass - Reference class (ACPI_REFCLASS_LOCAL or ACPI_REFCLASS_ARG)
1061 *    Value - Reference value (local ID or argument ID)
1062 *    WalkState - Current walk state
1063 *    Object - Pointer to the object (NULL if not found)
1064 *
1065 *  Returns:
1066 *    Status - AE_OK if successful, AE_NOT_FOUND if not found,
1067 *    or other error code.
1068 *
1069 *****************************************************************************/
1070
1071 ACPI_STATUS AcpiDsMethodDataGetNode (ACPI_REFCLASS ref_class,
1072     UINT32 value, ACPI_WALK_STATE *walk_state,
1073     ACPI_OBJECT **object)
1074 {
1075     ACPI_STATUS status;
1076     ACPI_OBJECT *obj;
1077     ACPI_NAMESPACE_NODE *node;
1078     ACPI_METHOD_OBJECT *method_obj;
1079     ACPI_LOCAL_VARIABLE *local_var;
1080     ACPI_METHOD_ARGUMENT *method_arg;
1081
1082     /*
1083      * Local variables are stored in the namespace, while
1084      * method arguments are stored in the method object.
1085      */
1086     if (ref_class == ACPI_REFCLASS_LOCAL)
1087     {
1088         node = AcpiNsFindLocalVariable (value);
1089         if (!node)
1090             return (AE_NOT_FOUND);
1091         local_var = (ACPI_LOCAL_VARIABLE *) node;
1092         obj = local_var->Object;
1093     }
1094     else
1095     {
1096         method_obj = (ACPI_METHOD_OBJECT *) walk_state->MethodObj;
1097         method_arg = (ACPI_METHOD_ARGUMENT *) method_obj->Arguments + value;
1098         obj = method_arg->Object;
1099     }
1100
1101     if (!obj)
1102         return (AE_NOT_FOUND);
1103
1104     *object = obj;
1105     return (AE_OK);
1106 }
1107
1108 /*****************************************************************************
1109 *
1110 *  Function: AcpiDsMethodDataGetNode
1111 *
1112 *  Purpose: Get the node for a method argument or local variable.
1113 *
1114 *  Parameters:
1115 *    RefClass - Reference class (ACPI_REFCLASS_LOCAL or ACPI_REFCLASS_ARG)
1116 *    Value - Reference value (local ID or argument ID)
1117 *    WalkState - Current walk state
1118 *    Object - Pointer to the object (NULL if not found)
1119 *
1120 *  Returns:
1121 *    Status - AE_OK if successful, AE_NOT_FOUND if not found,
1122 *    or other error code.
1123 *
1124 *****************************************************************************/
1125
1126 ACPI_STATUS AcpiDsMethodDataGetNode (ACPI_REFCLASS ref_class,
1127     UINT32 value, ACPI_WALK_STATE *walk_state,
1128     ACPI_OBJECT **object)
1129 {
1130     ACPI_STATUS status;
1131     ACPI_OBJECT *obj;
1132     ACPI_NAMESPACE_NODE *node;
1133     ACPI_METHOD_OBJECT *method
```

```
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 }
```

new/usr/src/common/acpica/components/dispatcher/dsopcode.c

1

```
*****
22489 Fri Jan 17 07:29:58 2014
new/usr/src/common/acpica/components/dispatcher/dsopcode.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dsopcode - Dispatcher support for regions and fields
4 * Module Name: dsopcode - Dispatcher support for regions and fields
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __DSOPCODE_C__
45
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 "acevents.h"
54 #include "actables.h"
55
56 #define _COMPONENT ACPI_DISPATCHER
57 #define ACPI_MODULE_NAME ("dsopcode")
```

new/usr/src/common/acpica/components/dispatcher/dsopcode.c

2

```
59 /* Local prototypes */
60
61 static ACPI_STATUS
62 AcpiDsInitBufferField (
63     UINT16          AmlOpcode,
64     ACPI_OPERAND_OBJECT *ObjDesc,
65     ACPI_OPERAND_OBJECT *BufferDesc,
66     ACPI_OPERAND_OBJECT *OffsetDesc,
67     ACPI_OPERAND_OBJECT *LengthDesc,
68     ACPI_OPERAND_OBJECT *ResultDesc);
69
70
71 /*****
72 *
73 * FUNCTION:      AcpiDsInitializeRegion
74 *
75 * PARAMETERS:    ObjHandle      - Region namespace node
76 *
77 * RETURN:        Status
78 *
79 * DESCRIPTION:    Front end to EvInitializeRegion
80 *
81 *****/
82
83 ACPI_STATUS
84 AcpiDsInitializeRegion (
85     ACPI_HANDLE      ObjHandle)
86 {
87     ACPI_OPERAND_OBJECT *ObjDesc;
88     ACPI_STATUS          Status;
89
90     ObjDesc = AcpiNsGetAttachedObject (ObjHandle);
91
92     /* Namespace is NOT locked */
93
94     Status = AcpiEvInitializeRegion (ObjDesc, FALSE);
95     return (Status);
96 }
97
98 _unchanged_portion_omitted_
99
100 /*****
101 *
102 * FUNCTION:      AcpiDsEvalTableRegionOperands
103 *
104 * PARAMETERS:    WalkState      - Current walk
105 *                Op              - A valid region Op object
106 *
107 * RETURN:        Status
108 *
109 * DESCRIPTION:    Get region address and length.
110 *                Called from AcpiDsExecEndOp during DataTableRegion parse
111 *                tree walk.
112 *****/
113
114 ACPI_STATUS
115 AcpiDsEvalTableRegionOperands (
116     ACPI_WALK_STATE *WalkState,
117     ACPI_PARSE_OBJECT *Op)
118 {
119     ACPI_STATUS          Status;
120     ACPI_OPERAND_OBJECT *ObjDesc;
121     ACPI_OPERAND_OBJECT **Operand;
122 }
```

```

516     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
528     * This is where we evaluate the SignatureString and OemIDString
529     * and OemTableIDString of the DataTableRegion declaration
530     */
531     Node = Op->Common.Node;

532     /* NextOp points to Signature string op */
533     /* NextOp points to SignatureString op */

534     NextOp = Op->Common.Value.Arg;

535     /*
536     * Evaluate/create the Signature string, OemId string,
537     * and OemTableId string operands
538     * Evaluate/create the SignatureString and OemIDString
539     * and OemTableIDString operands
540     */
541     Status = AcpiDsCreateOperands (WalkState, NextOp);
542     if (ACPI_FAILURE (Status))
543     {
544         return_ACPI_STATUS (Status);
545     }

546     /*
547     * Resolve the Signature string, OemId string,
548     * and OemTableId string operands
549     * Resolve the SignatureString and OemIDString
550     * and OemTableIDString operands
551     */
552     Status = AcpiExResolveOperands (Op->Common.AmlOpcode,
553     ACPI_WALK_OPERANDS, WalkState);
554     if (ACPI_FAILURE (Status))
555     {
556         return_ACPI_STATUS (Status);
557     }

558     Operand = &WalkState->Operands[0];

559     /* 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     }

567     AcpiUtRemoveReference (Operand[0]);
568     AcpiUtRemoveReference (Operand[1]);
569     AcpiUtRemoveReference (Operand[2]);

570

571     Status = AcpiGetTableByIndex (TableIndex, &Table);
572     if (ACPI_FAILURE (Status))
573     {
574

```

```

575         return_ACPI_STATUS (Status);
576     }

577     ObjDesc = AcpiNsGetAttachedObject (Node);
578     if (!ObjDesc)
579     {
580         return_ACPI_STATUS (AE_NOT_EXIST);
581     }

582

583     ObjDesc->Region.Address = (ACPI_PHYSICAL_ADDRESS) ACPI_TO_INTEGER (Table);
584     ObjDesc->Region.Length = Table->Length;

585     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "RgnObj %p Addr %8.8X%8.8X Len %X\n",
586     ObjDesc,
587     ACPI_FORMAT_NATIVE_UINT (ObjDesc->Region.Address),
588     ObjDesc->Region.Length));

589     /* Now the address and length are valid for this opregion */

590     ObjDesc->Region.Flags |= APOBJ_DATA_VALID;

591     return_ACPI_STATUS (Status);
592 }

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 *****/

614 ACPI_STATUS
615 AcpiDsEvalDataObjectOperands (
616     ACPI_WALK_STATE      *WalkState,
617     ACPI_PARSE_OBJECT     *Op,
618     ACPI_OPERAND_OBJECT   *ObjDesc)
619 {
620     ACPI_STATUS      Status;
621     ACPI_OPERAND_OBJECT *ArgDesc;
622     UINT32            Length;

623

624     ACPI_FUNCTION_TRACE (DsEvalDataObjectOperands);

625

626     /* The first operand (for all of these data objects) is the length */

627

628     /*
629     * Set proper index into operand stack for AcpiDsObjStackPush
630     * invoked inside AcpiDsCreateOperand.
631     */
632     WalkState->OperandIndex = WalkState->NumOperands;

633     Status = AcpiDsCreateOperand (WalkState, Op->Common.Value.Arg, 1);
634     if (ACPI_FAILURE (Status))
635     {
636         return_ACPI_STATUS (Status);
637     }

```

```

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



new/usr/src/common/acpica/components/dispatcher/dsutils.c

1

```
*****
28491 Fri Jan 17 07:29:59 2014
new/usr/src/common/acpica/components/dispatcher/dsutils.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dsutils - Dispatcher utilities
4 *
5 *****/
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __DSUTILS_C__
45
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"
54
55 #define _COMPONENT ACPI_DISPATCHER
56 #define ACPI_MODULE_NAME ("dsutils")
```

new/usr/src/common/acpica/components/dispatcher/dsutils.c

2

```
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 *****/
73
74 void
75 AcpiDsClearImplicitReturn (
76     ACPI_WALK_STATE *WalkState)
77 {
78     ACPI_FUNCTION_NAME (DsClearImplicitReturn);
79
80     /*
81      * Slack must be enabled for this feature
82      */
83     if (!AcpiGbl_EnableInterpreterSlack)
84     {
85         return;
86     }
87
88     if (WalkState->ImplicitReturnObj)
89     {
90         /*
91          * Delete any "stale" implicit return. However, in
92          * complex statements, the implicit return value can be
93          * bubbled up several levels.
94          */
95         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
96             "Removing reference on stale implicit return obj %p\n",
97             WalkState->ImplicitReturnObj));
98
99         AcpiUtRemoveReference (WalkState->ImplicitReturnObj);
100         WalkState->ImplicitReturnObj = NULL;
101     }
102 }
103
104 _____ unchanged portion omitted _____
105
106 /*****
107 *
108 * FUNCTION: AcpiDsIsResultUsed
109 *
110 * PARAMETERS: Op - Current Op
111 * WalkState - Current State
112 *
113 * RETURN: TRUE if result is used, FALSE otherwise
114 *
115 * DESCRIPTION: Check if a result object will be used by the parent
116 *
117 *****/
118
119 BOOLEAN
120 AcpiDsIsResultUsed (
121     ACPI_PARSE_OBJECT *Op,
122     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     *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--;

788     /* Create the interpreter arguments, in reverse order */
789     /* 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_____

```

new/usr/src/common/acpica/components/dispatcher/dswexec.c

1

```
*****
23264 Fri Jan 17 07:29:59 2014
new/usr/src/common/acpica/components/dispatcher/dswexec.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dswexec - Dispatcher method execution callbacks;
4 *                               dispatch to interpreter.
5 *
6 *****/
8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
```

new/usr/src/common/acpica/components/dispatcher/dswexec.c

2

```
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         }
252     }

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_REFERENCE) &&
483             (WalkState->Operands[1]->Common.Type == ACPI_TYPE_LOCAL_REFERENCE) &&
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;

```

```
new/usr/src/common/acpica/components/dispatcher/dswexec.c
```

```

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:
642             Status = AcpiDsEvalDataObjectOperands (WalkState, Op, NULL);
643             break;
644         }
645
646         /*
647          * If a result object was returned from above, push it on the
648          * current result stack
649          */
650         if (WalkState->ResultObj)
651         {
652             Status = AcpiDsResultPush (WalkState->ResultObj, WalkState);
653         }
654         break;
655
656     case AML_TYPE_NAMED_FIELD:
657     case AML_TYPE_NAMED_COMPLEX:
658     case AML_TYPE_NAMED_SIMPLE:
659     case AML_TYPE_NAMED_NO_OBJ:
660
661         Status = AcpiDsLoad2EndOp (WalkState);
662         if (ACPI_FAILURE (Status))
663         {
664             break;
665         }
666
667         if (Op->Common.AmlOpcode == AML_REGION_OP)
668         {
669             ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
670                 "Executing OpRegion Address/Length Op=%p\n", Op));
671
672             Status = AcpiDsEvalRegionOperands (WalkState, Op);
673             if (ACPI_FAILURE (Status))
674             {
675                 break;
676             }
677         }
678         else if (Op->Common.AmlOpcode == AML_DATA_REGION_OP)
679         {
680             ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
681                 "Executing DataTableRegion Strings Op=%p\n", Op));
682
683             Status = AcpiDsEvalTableRegionOperands (WalkState, Op);
684             if (ACPI_FAILURE (Status))
685             {
686                 break;
687             }
688         }
689         else if (Op->Common.AmlOpcode == AML_BANK_FIELD_OP)
690         {
691             ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
692                 "Executing BankField Op=%p\n", Op));
693
694             Status = AcpiDsEvalBankFieldOperands (WalkState, Op);
695             if (ACPI_FAILURE (Status))
696             {
697                 break;
698             }
699         }
700         break;
701
702     case AML_TYPE_UNDEFINED:

```

```

703     case AML_TYPE_UNDEFINED:

```

```

705         ACPI_ERROR ((AE_INFO,
706             "Undefined opcode type Op=%p", Op));
707         return ACPI_STATUS (AE_NOT_IMPLEMENTED);
708
709     case AML_TYPE_BOGUS:
710
711         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
712             "Internal opcode=%X type Op=%p\n",
713             WalkState->Opcode, Op));
714         break;
715
716     default:
717
718         ACPI_ERROR ((AE_INFO,
719             "Unimplemented opcode, class=0x%X type=0x%X Opcode=0x%X Op=%p",
720             "Unimplemented opcode, class=0x%X type=0x%X Opcode=-0x%X Op=%p",
721             OpClass, OpType, Op->Common.AmlOpcode, Op));
722
723         Status = AE_NOT_IMPLEMENTED;
724         break;
725     }
726
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);
732     AcpiExTruncateFor32bitTable (WalkState->ResultObj);
733
734     /*
735      * Check if we just completed the evaluation of a
736      * conditional predicate
737      */
738     if ((ACPI_SUCCESS (Status)) &&
739         (WalkState->ControlState) &&
740         (WalkState->ControlState->Common.State ==
741             ACPI_CONTROL_PREDICATE_EXECUTING) &&
742         (WalkState->ControlState->Control.PredicateOp == Op))
743     {
744         Status = AcpiDsGetPredicateValue (WalkState, WalkState->ResultObj);
745         WalkState->ResultObj = NULL;
746     }
747
748 Cleanup:
749
750     if (WalkState->ResultObj)
751     {
752         /* Break to debugger to display result */
753
754         ACPI_DEBUGGER_EXEC (AcpiDbDisplayResultObject (WalkState->ResultObj,
755             WalkState));
756
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     }
764
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 }
```

new/usr/src/common/acpica/components/dispatcher/dswload.c

1

```
*****
17940 Fri Jan 17 07:30:00 2014
new/usr/src/common/acpica/components/dispatcher/dswload.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: dswload - Dispatcher first pass namespace load callbacks
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __DSWLOAD_C__
45
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
54 #ifdef ACPI_AS_L_COMPILER
55 #include "acdisasm.h"
56 #endif
```

new/usr/src/common/acpica/components/dispatcher/dswload.c

2

```
58 #define _COMPONENT          ACPI_DISPATCHER
59 #define ACPI_MODULE_NAME    ("dswload")
60
61 /*****
62 *
63 * FUNCTION:      AcpiDsInitCallbacks
64 *
65 * PARAMETERS:    WalkState      - Current state of the parse tree walk
66 *                PassNumber     - 1, 2, or 3
67 *
68 * RETURN:       Status
69 *
70 * DESCRIPTION:   Init walk state callbacks
71 *
72 * *****/
73
74
75 ACPI_STATUS
76 AcpiDsInitCallbacks (
77     ACPI_WALK_STATE *WalkState,
78     UINT32 PassNumber)
79 {
80
81     switch (PassNumber)
82     {
83     case 1:
84
85         WalkState->ParseFlags      = ACPI_PARSE_LOAD_PASS1 |
86                                     ACPI_PARSE_DELETE_TREE;
87         WalkState->DescendingCallback = AcpiDsLoad1BeginOp;
88         WalkState->AscendingCallback  = AcpiDsLoad1EndOp;
89         break;
90
91     case 2:
92
93         WalkState->ParseFlags      = ACPI_PARSE_LOAD_PASS1 |
94                                     ACPI_PARSE_DELETE_TREE;
95         WalkState->DescendingCallback = AcpiDsLoad2BeginOp;
96         WalkState->AscendingCallback  = AcpiDsLoad2EndOp;
97         break;
98
99     case 3:
100
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;
108
109     default:
110
111         return (AE_BAD_PARAMETER);
112     }
113
114     return (AE_OK);
115 }
116
117 /*****
118 *
119 * FUNCTION:      AcpiDsLoad1BeginOp
120 *
121 * PARAMETERS:    WalkState      - Current state of the parse tree walk
122 *                OutOp          - Where to return op if a new one is created
123 */
```

```

124 *
125 * RETURN:      Status
126 *
127 * DESCRIPTION: Descending callback used during the loading of ACPI tables.
128 *
129 *****/
131 ACPI_STATUS
132 AcpiDsLoadlBeginOp (
133     ACPI_WALK_STATE      *WalkState,
134     ACPI_PARSE_OBJECT    **OutOp)
135 {
136     ACPI_PARSE_OBJECT    *Op;
137     ACPI_NAMESPACE_NODE  *Node;
138     ACPI_STATUS           Status;
139     ACPI_OBJECT_TYPE      ObjectType;
140     char                  *Path;
141     UINT32                Flags;
142
144     ACPI_FUNCTION_TRACE (DsLoadlBeginOp);
145
147     Op = WalkState->Op;
148     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Op=%p State=%p\n", Op, WalkState));
149
150     /* We are only interested in opcodes that have an associated name */
151
152     if (Op)
153     {
154         if (!(WalkState->OpInfo->Flags & AML_NAMED))
155         {
156             *OutOp = Op;
157             return ACPI_STATUS (AE_OK);
158         }
159
160         /* Check if this object has already been installed in the namespace */
161
162         if (Op->Common.Node)
163         {
164             *OutOp = Op;
165             return ACPI_STATUS (AE_OK);
166         }
167     }
168
169     Path = AcpiPsGetNextNamestring (&WalkState->ParserState);
170
171     /* Map the raw opcode into an internal object type */
172
173     ObjectType = WalkState->OpInfo->ObjectType;
174
175     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
176         "State=%p Op=%p [%s]\n", WalkState, Op, AcpiUtGetTypeName (ObjectType)));
177
178     switch (WalkState->Opcode)
179     {
180     case AML_SCOPE_OP:
181
182         /*
183          * The target name of the Scope() operator must exist at this point so
184          * that we can actually open the scope to enter new names underneath it.
185          * Allow search-to-root for single namesegs.
186          */
187         Status = AcpiNsLookup (WalkState->ScopeInfo, Path, ObjectType,
188             ACPI_IMODE_EXECUTE, ACPI_NS_SEARCH_PARENT, WalkState, &
189             #ifdef ACPI_ASL_COMPILER

```

```

189     if (Status == AE_NOT_FOUND)
190     {
191         /*
192          * Table disassembly:
193          * Target of Scope() not found. Generate an External for it, and
194          * insert the name into the namespace.
195          */
196         AcpiDmAddOpToExternalList (Op, Path, ACPI_TYPE_DEVICE, 0, 0);
197         AcpiDmAddToExternalList (Op, Path, ACPI_TYPE_DEVICE, 0);
198         Status = AcpiNsLookup (WalkState->ScopeInfo, Path, ObjectType,
199             ACPI_IMODE_LOAD_PASS1, ACPI_NS_SEARCH_PARENT,
200             WalkState, &Node);
201     }
202     #endif
203     if (ACPI_FAILURE (Status))
204     {
205         ACPI_ERROR_NAMESPACE (Path, Status);
206         return ACPI_STATUS (Status);
207     }
208
209     /*
210     * Check to make sure that the target is
211     * one of the opcodes that actually opens a scope
212     */
213     switch (Node->Type)
214     {
215     case ACPI_TYPE_ANY:
216     case ACPI_TYPE_LOCAL_SCOPE:          /* Scope */
217     case ACPI_TYPE_DEVICE:
218     case ACPI_TYPE_POWER:
219     case ACPI_TYPE_PROCESSOR:
220     case ACPI_TYPE_THERMAL:
221
222         /* These are acceptable types */
223         break;
224
225     case ACPI_TYPE_INTEGER:
226     case ACPI_TYPE_STRING:
227     case ACPI_TYPE_BUFFER:
228
229         /*
230          * These types we will allow, but we will change the type.
231          * This enables some existing code of the form:
232          *
233          * Name (DEB, 0)
234          * Scope (DEB) { ... }
235          *
236          * Note: silently change the type here. On the second pass,
237          * we will report a warning
238          */
239         ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
240             "Type override - [%4.4s] had invalid type (%s) "
241             "for Scope operator, changed to type ANY\n",
242             AcpiUtGetNodeName (Node), AcpiUtGetTypeName (Node->Type)));
243
244         Node->Type = ACPI_TYPE_ANY;
245         WalkState->ScopeInfo->Common.Value = ACPI_TYPE_ANY;
246         break;
247
248     case ACPI_TYPE_METHOD:
249
250         /*
251          * Allow scope change to root during execution of module-level
252          * code. Root is typed METHOD during this time.
253          */
254         if ((Node == AcpiGbl_RootNode) &&
255             (WalkState->ParseFlags & ACPI_PARSE_MODULE_LEVEL))

```

```

253     {
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 */

386 #if (defined (ACPI_NO_METHOD_EXECUTION) || defined (ACPI_CONSTANT_EVAL_ONLY))
387     Op->Named.Path = ACPI_CAST_PTR (UINT8, Path);
388 #endif

390     if (Node)
391     {
392         /*
393          * Put the Node in the "op" object that the parser uses, so we
394          * can get it again quickly when this scope is closed
395          */
396         Op->Common.Node = Node;
397         Op->Named.Name = Node->Name.Integer;
398     }

400     AcpiPsAppendArg (AcpiPsGetParentScope (&WalkState->ParserState), Op);
401     *OutOp = Op;
402     return ACPI_STATUS (Status);
403 }
```

unchanged\_portion\_omitted

new/usr/src/common/acpica/components/dispatcher/dswload2.c

1

```
*****
21930 Fri Jan 17 07:30:00 2014
new/usr/src/common/acpica/components/dispatcher/dswload2.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dswload2 - Dispatcher second pass namespace load callbacks
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __DSWLOAD2_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 "acevents.h"

55 #define _COMPONENT          ACPI_DISPATCHER
56 #define ACPI_MODULE_NAME    ("dswload2")
```

new/usr/src/common/acpica/components/dispatcher/dswload2.c

2

```
59 /*****
60 *
61 * FUNCTION:      AcpiDsLoad2BeginOp
62 *
63 * PARAMETERS:    WalkState      - Current state of the parse tree walk
64 *                OutOp          - Where to return op if a new one is created
65 *
66 * RETURN:        Status
67 *
68 * DESCRIPTION:    Descending callback used during the loading of ACPI tables.
69 *
70 *****/

72 ACPI_STATUS
73 AcpiDsLoad2BeginOp (
74     ACPI_WALK_STATE      *WalkState,
75     ACPI_PARSE_OBJECT     **OutOp)
76 {
77     ACPI_PARSE_OBJECT     *Op;
78     ACPI_NAMESPACE_NODE    *Node;
79     ACPI_STATUS            Status;
80     ACPI_OBJECT_TYPE        ObjectType;
81     char                   *BufferPtr;
82     UINT32                  Flags;

85     ACPI_FUNCTION_TRACE (DsLoad2BeginOp);

88     Op = WalkState->Op;
89     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH, "Op=%p State=%p\n", Op, WalkState));

91     if (Op)
92     {
93         if ((WalkState->ControlState) &&
94             (WalkState->ControlState->Common.State ==
95              ACPI_CONTROL_CONDITIONAL_EXECUTING))
96         {
97             /* We are executing a while loop outside of a method */

99             Status = AcpiDsExecBeginOp (WalkState, OutOp);
100             return ACPI_STATUS (Status);
101         }

103         /* We only care about Namespace opcodes here */

105         if (((WalkState->OpInfo->Flags & AML_NSOPCODE) &&
106             (WalkState->Opcode != AML_INT_NAMEPATH_OP)) ||
107             ((WalkState->OpInfo->Flags & AML_NAMED)))
108         {
109             return ACPI_STATUS (AE_OK);
110         }

112         /* Get the name we are going to enter or lookup in the namespace */

114         if (WalkState->Opcode == AML_INT_NAMEPATH_OP)
115         {
116             /* For Namepath op, get the path string */

118             BufferPtr = Op->Common.Value.String;
119             if (!BufferPtr)
120             {
121                 /* No name, just exit */

123                 return ACPI_STATUS (AE_OK);
124             }
125         }
126     }
127 }
```

```

125     }
126     else
127     {
128         /* Get name from the op */
129
130         BufferPtr = ACPI_CAST_PTR (char, &Op->Named.Name);
131     }
132 }
133 else
134 {
135     /* Get the namestring from the raw AML */
136
137     BufferPtr = AcpiPsGetNextNamestring (&WalkState->ParserState);
138 }
139
140 /* Map the opcode into an internal object type */
141
142 ObjectType = WalkState->OpInfo->ObjectType;
143
144 ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
145     "State=%p Op=%p Type=%X\n", WalkState, Op, ObjectType));
146
147 switch (WalkState->Opcode)
148 {
149 case AML_FIELD_OP:
150 case AML_BANK_FIELD_OP:
151 case AML_INDEX_FIELD_OP:
152
153     Node = NULL;
154     Status = AE_OK;
155     break;
156
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;
167
168 case AML_SCOPE_OP:
169
170     /* Special case for Scope(\) -> refers to the Root node */
171
172     if (Op && (Op->Named.Node == AcpiGbl_RootNode))
173     {
174         Node = Op->Named.Node;
175
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         }
204         #else
205             ACPI_ERROR_NAMESPACE (BufferPtr, Status);
206         #endif
207         return_ACPI_STATUS (Status);
208     }
209 }
210
211 /*
212  * We must check to make sure that the target is
213  * one of the opcodes that actually opens a scope
214  */
215 switch (Node->Type)
216 {
217 case ACPI_TYPE_ANY:
218 case ACPI_TYPE_LOCAL_SCOPE: /* Scope */
219 case ACPI_TYPE_DEVICE:
220 case ACPI_TYPE_POWER:
221 case ACPI_TYPE_PROCESSOR:
222 case ACPI_TYPE_THERMAL:
223
224     /* These are acceptable types */
225     break;
226
227 case ACPI_TYPE_INTEGER:
228 case ACPI_TYPE_STRING:
229 case ACPI_TYPE_BUFFER:
230
231     /*
232      * These types we will allow, but we will change the type.
233      * This enables some existing code of the form:
234      *
235      * Name (DEB, 0)
236      * Scope (DEB) { ... }
237      */
238     ACPI_WARNING ((AE_INFO,
239         "Type override - [%4.4s] had invalid type (%s) ",
240         "for Scope operator, changed to type ANY",
241         AcpiUtGetNodeName (Node), AcpiUtGetTypeName (Node->Type)));
242
243     Node->Type = ACPI_TYPE_ANY;
244     WalkState->ScopeInfo->Common.Value = ACPI_TYPE_ANY;
245     break;
246
247 case ACPI_TYPE_METHOD:
248
249     /*
250      * Allow scope change to root during execution of module-level
251      * code. Root is typed METHOD during this time.
252      */
253     if ((Node == AcpiGbl_RootNode) &&
254         (WalkState->ParseFlags & ACPI_PARSE_MODULE_LEVEL))
255     {
256         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);
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);
628             return (Status);
629         }

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)
648         {
649             Status = AE_OK;
650         }
651     }
652     break;

654     case AML_NAME_OP:

656         Status = AcpiDsCreateNode (WalkState, Node, Op);
657         break;

659     case AML_METHOD_OP:
660         /*
661          * MethodOp PkgLength NameString MethodFlags TermList
662          *
663          * Note: We must create the method node/object pair as soon as we
664          * see the method declaration. This allows later pass1 parsing
665          * of invocations of the method (need to know the number of
666          * arguments.)
667          */
668         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
669             "LOADING-Method: State=%p Op=%p NamedObj=%p\n",
670             WalkState, Op, Op->Named.Node));

672         if (!AcpiNsGetAttachedObject (Op->Named.Node))
673         {
674             WalkState->Operands[0] = ACPI_CAST_PTR (void, Op->Named.Node);
675             WalkState->NumOperands = 1;

677             Status = AcpiDsCreateOperands (WalkState, Op->Common.Value.Arg);
678             if (ACPI_SUCCESS (Status))
679             {
680                 Status = AcpiExCreateMethod (Op->Named.Data,
681                     Op->Named.Length, WalkState);
682             }
683             WalkState->Operands[0] = NULL;
684             WalkState->NumOperands = 0;

686             if (ACPI_FAILURE (Status))
687             {
688                 return ACPI_STATUS (Status);
689             }
690         }
691         break;

693 #endif /* ACPI_NO_METHOD_EXECUTION */

695     default:

697         /* All NAMED_COMPLEX opcodes must be handled above */
698         break;
699     }
700     break;

702     case AML_CLASS_INTERNAL:

704         /* case AML_INT_NAMEPATH_OP: */
705         break;

707     case AML_CLASS_METHOD_CALL:

```

```

709         ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
710             "RESOLVING-MethodCall: State=%p Op=%p NamedObj=%p\n",
711             WalkState, Op, Node));

713         /*
714          * Lookup the method name and save the Node
715          */
716         Status = AcpiNsLookup (WalkState->ScopeInfo, Arg->Common.Value.String,
717             ACPI_TYPE_ANY, ACPI_IMODE_LOAD_PASS2,
718             ACPI_NS_SEARCH_PARENT | ACPI_NS_DONT_OPEN_SCOPE,
719             WalkState, &(NewNode));
720         if (ACPI_SUCCESS (Status))
721         {
722             /*
723              * Make sure that what we found is indeed a method
724              * We didn't search for a method on purpose, to see if the name
725              * would resolve
726              */
727             if (NewNode->Type != ACPI_TYPE_METHOD)
728             {
729                 Status = AE_AML_OPERAND_TYPE;
730             }

732             /* We could put the returned object (Node) on the object stack for
733              * later, but for now, we will put it in the "op" object that the
734              * parser uses, so we can get it again at the end of this scope
735              */
736             Op->Common.Node = NewNode;
737         }
738         else
739         {
740             ACPI_ERROR_NAMESPACE (Arg->Common.Value.String, Status);
741         }
742         break;

745     default:

747         break;
748     }

750 Cleanup:

752     /* Remove the Node pushed at the very beginning */

754     WalkState->Operands[0] = NULL;
755     WalkState->NumOperands = 0;
756     return ACPI_STATUS (Status);
757 }

```

new/usr/src/common/acpica/components/dispatcher/dswscope.c

1

```
*****
6978 Fri Jan 17 07:30:01 2014
new/usr/src/common/acpica/components/dispatcher/dswscope.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dswscope - Scope stack manipulation
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __DSWScope_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acdispat.h"

51 #define _COMPONENT          ACPI_DISPATCHER
52         ACPI_MODULE_NAME    ("dswscope")

55 /*****
56 *
57 * FUNCTION:      AcpiDsScopeStackClear
58 *
```

new/usr/src/common/acpica/components/dispatcher/dswscope.c

2

```
59 * PARAMETERS:  WalkState      - Current state
60 *
61 * RETURN:       None
62 *
63 * DESCRIPTION:  Pop (and free) everything on the scope stack except the
64 *               root scope object (which remains at the stack top.)
65 *
66 *****/

68 void
69 AcpiDsScopeStackClear (
70     ACPI_WALK_STATE      *WalkState)
71 {
72     ACPI_GENERIC_STATE    *ScopeInfo;

74     ACPI_FUNCTION_NAME (DsScopeStackClear);

77     while (WalkState->ScopeInfo)
78     {
79         /* Pop a scope off the stack */

81         ScopeInfo = WalkState->ScopeInfo;
82         WalkState->ScopeInfo = ScopeInfo->Scope.Next;

84         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
85             "Popped object type (%s)\n",
86             AcpiUtGetTypeName (ScopeInfo->Common.Value)));
87         AcpiUtDeleteGenericState (ScopeInfo);
88     }
89 }

    unchanged_portion_omitted_
```

new/usr/src/common/acpica/components/dispatcher/dswstate.c

1

```
*****
22835 Fri Jan 17 07:30:01 2014
new/usr/src/common/acpica/components/dispatcher/dswstate.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: dswstate - Dispatcher parse tree walk management routines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __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
```

new/usr/src/common/acpica/components/dispatcher/dswstate.c

2

```
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;
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;
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 }

```

new/usr/src/common/acpica/components/events/evevent.c

1

```
*****
9694 Fri Jan 17 07:30:02 2014
new/usr/src/common/acpica/components/events/evevent.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: evevent - Fixed Event handling and dispatch
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
45 #include "accommon.h"
46 #include "acevents.h"

48 #define _COMPONENT          ACPI_EVENTS
49         ACPI_MODULE_NAME    ("evevent")

51 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */

53 /* Local prototypes */

55 static ACPI_STATUS
56 AcpiEvFixedEventInitialize (
57     void);
```

new/usr/src/common/acpica/components/events/evevent.c

2

```
59 static UINT32
60 AcpiEvFixedEventDispatch (
61     UINT32                Event);

64 /*****
65 *
66 * FUNCTION:      AcpiEvInitializeEvents
67 *
68 * PARAMETERS:    None
69 *
70 * RETURN:        Status
71 *
72 * DESCRIPTION:    Initialize global data structures for ACPI events (Fixed, GPE)
73 *
74 *****/

76 ACPI_STATUS
77 AcpiEvInitializeEvents (
78     void)
79 {
80     ACPI_STATUS          Status;

83     ACPI_FUNCTION_TRACE (EvInitializeEvents);

86     /* If Hardware Reduced flag is set, there are no fixed events */

88     if (AcpiGbl_ReducedHardware)
89     {
90         return_ACPI_STATUS (AE_OK);
91     }

93     /*
94     * Initialize the Fixed and General Purpose Events. This is done prior to
95     * enabling SCIs to prevent interrupts from occurring before the handlers
96     * are installed.
97     */
98     Status = AcpiEvFixedEventInitialize ();
99     if (ACPI_FAILURE (Status))
100     {
101         ACPI_EXCEPTION ((AE_INFO, Status,
102             "Unable to initialize fixed events"));
103         return_ACPI_STATUS (Status);
104     }

106     Status = AcpiEvGpeInitialize ();
107     if (ACPI_FAILURE (Status))
108     {
109         ACPI_EXCEPTION ((AE_INFO, Status,
110             "Unable to initialize general purpose events"));
111         return_ACPI_STATUS (Status);
112     }

114     return_ACPI_STATUS (Status);
115 }

118 /*****
119 *
120 * FUNCTION:      AcpiEvInstallXruptHandlers
121 *
122 * PARAMETERS:    None
123 *
124 * RETURN:        Status
```



```

125 *
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.
316      * Make sure we've got a handler. If not, report an error. The event is
317      * disabled to prevent further interrupts.
318      */
319     if (!AcpiGbl_FixedEventHandlers[Event].Handler)
320     {
321         (void) AcpiWriteBitRegister (
322             AcpiGbl_FixedEventInfo[Event].EnableRegisterId,
323             ACPI_DISABLE_EVENT);
324         ACPI_ERROR ((AE_INFO,
325             "No installed handler for fixed event - %s (%u), disabling",
326             AcpiUtGetEventName (Event), Event));
327         "No installed handler for fixed event [0x%08X]",
328         Event));

329     return (ACPI_INTERRUPT_NOT_HANDLED);

330     /* Invoke the Fixed Event handler */

332     return ((AcpiGbl_FixedEventHandlers[Event].Handler)(
333         AcpiGbl_FixedEventHandlers[Event].Context));
334 }

336 #endif /* !ACPI_REDUCED_HARDWARE */

```

new/usr/src/common/acpica/components/events/evglock.c

1

```
*****
11425 Fri Jan 17 07:30:02 2014
new/usr/src/common/acpica/components/events/evglock.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: evglock - Global Lock support
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
45 #include "accommon.h"
46 #include "acevents.h"
47 #include "acinterp.h"

49 #define _COMPONENT          ACPI_EVENTS
50 #define ACPI_MODULE_NAME    ("evglock")

52 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */

54 /* Local prototypes */

56 static UINT32
57 AcpiEvGlobalLockHandler (
58     void                *Context);
```

new/usr/src/common/acpica/components/events/evglock.c

2

```
61 /*****
62 *
63 * FUNCTION:      AcpiEvInitGlobalLockHandler
64 *
65 * PARAMETERS:    None
66 *
67 * RETURN:        Status
68 *
69 * DESCRIPTION:   Install a handler for the global lock release event
70 *
71 *****/

73 ACPI_STATUS
74 AcpiEvInitGlobalLockHandler (
75     void)
76 {
77     ACPI_STATUS          Status;

80     ACPI_FUNCTION_TRACE (EvInitGlobalLockHandler);

83     /* If Hardware Reduced flag is set, there is no global lock */

85     if (AcpiGbl_ReducedHardware)
86     {
87         return_ACPI_STATUS (AE_OK);
88     }

90     /* Attempt installation of the global lock handler */

92     Status = AcpiInstallFixedEventHandler (ACPI_EVENT_GLOBAL,
93         AcpiEvGlobalLockHandler, NULL);

95     /*
96     * If the global lock does not exist on this platform, the attempt to
97     * enable GBL_STATUS will fail (the GBL_ENABLE bit will not stick).
98     * Map to AE_OK, but mark global lock as not present. Any attempt to
99     * actually use the global lock will be flagged with an error.
100     */
101     AcpiGbl_GlobalLockPresent = FALSE;
102     if (Status == AE_NO_HARDWARE_RESPONSE)
103     {
104         ACPI_ERROR ((AE_INFO,
105             "No response from Global Lock hardware, disabling lock"));

107         return_ACPI_STATUS (AE_OK);
108     }

110     Status = AcpiOsCreateLock (&AcpiGbl_GlobalLockPendingLock);
111     if (ACPI_FAILURE (Status))
112     {
113         return_ACPI_STATUS (Status);
114     }

116     AcpiGbl_GlobalLockPending = FALSE;
117     AcpiGbl_GlobalLockPresent = TRUE;
118     return_ACPI_STATUS (Status);
119 }

122 /*****
123 *
124 * FUNCTION:      AcpiEvRemoveGlobalLockHandler
```

```
125 *
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 */
```

new/usr/src/common/acpica/components/events/evgpe.c

1

```
*****
25182 Fri Jan 17 07:30:02 2014
new/usr/src/common/acpica/components/events/evgpe.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: evgpe - General Purpose Event handling and dispatch
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
45 #include "accommon.h"
46 #include "acevents.h"
47 #include "acnamesp.h"
48
49 #define _COMPONENT          ACPI_EVENTS
50 #define ACPI_MODULE_NAME    ("evgpe")
51
52 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */
53
54 /* Local prototypes */
55
56 static void ACPI_SYSTEM_XFACE
57 AcpiEvAsynchExecuteGpeMethod (
```

new/usr/src/common/acpica/components/events/evgpe.c

2

```
58     void                      *Context);
59
60 static void ACPI_SYSTEM_XFACE
61 AcpiEvAsynchEnableGpe (
62     void                      *Context);
63
64
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 *****/
77
78 ACPI_STATUS
79 AcpiEvUpdateGpeEnableMask (
80     ACPI_GPE_EVENT_INFO      *GpeEventInfo)
81 {
82     ACPI_GPE_REGISTER_INFO    *GpeRegisterInfo;
83     UINT32                    RegisterBit;
84
85     ACPI_FUNCTION_TRACE (EvUpdateGpeEnableMask);
86
87     GpeRegisterInfo = GpeEventInfo->RegisterInfo;
88     if (!GpeRegisterInfo)
89     {
90         return_ACPI_STATUS (AE_NOT_EXIST);
91     }
92
93     RegisterBit = AcpiHwGetGpeRegisterBit (GpeEventInfo, GpeRegisterInfo);
94
95     /* Clear the run bit up front */
96
97     ACPI_CLEAR_BIT (GpeRegisterInfo->EnableForRun, RegisterBit);
98
99     /* Set the mask bit only if there are references to this GPE */
100
101     if (GpeEventInfo->RuntimeCount)
102     {
103         ACPI_SET_BIT (GpeRegisterInfo->EnableForRun, (UINT8) RegisterBit);
104     }
105
106     return_ACPI_STATUS (AE_OK);
107 }
108
109 unchanged_portion_omitted
110
111
112 /*****
113 *
114 * FUNCTION:      AcpiEvGpeDetect
115 *
116 * PARAMETERS:    GpeXruptList      - Interrupt block for this interrupt.
117 *                                     Can have multiple GPE blocks attached.
118 *
119 * RETURN:        INTERRUPT_HANDLED or INTERRUPT_NOT_HANDLED
120 *
121 * DESCRIPTION:    Detect if any GP events have occurred. This function is
```

```

376 *          executed at interrupt level.
377 *
378 *****/

380 UINT32
381 AcpiEvGpeDetect (
382     ACPI_GPE_XRPT_INFO    *GpeXruptList)
383 {
384     ACPI_STATUS             Status;
385     ACPI_GPE_BLOCK_INFO    *GpeBlock;
386     ACPI_GPE_REGISTER_INFO *GpeRegisterInfo;
387     UINT32                  IntStatus = ACPI_INTERRUPT_NOT_HANDLED;
388     UINT8                   EnabledStatusByte;
389     UINT32                  StatusReg;
390     UINT32                  EnableReg;
391     ACPI_CPU_FLAGS          Flags;
392     UINT32                  i;
393     UINT32                  j;

396     ACPI_FUNCTION_NAME (EvGpeDetect);

398     /* Check for the case where there are no GPEs */

400     if (!GpeXruptList)
401     {
402         return (IntStatus);
403     }

405     /*
406      * We need to obtain the GPE lock for both the data structs and registers
407      * Note: Not necessary to obtain the hardware lock, since the GPE
408      * registers are owned by the GpeLock.
409      */
410     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);

412     /* Examine all GPE blocks attached to this interrupt level */

414     GpeBlock = GpeXruptList->GpeBlockListHead;
415     while (GpeBlock)
416     {
417         /*
418          * Read all of the 8-bit GPE status and enable registers in this GPE
419          * block, saving all of them. Find all currently active GP events.
420          */
421         for (i = 0; i < GpeBlock->RegisterCount; i++)
422         {
423             /* Get the next status/enable pair */

425             GpeRegisterInfo = &GpeBlock->RegisterInfo[i];

427             /*
428              * Optimization: If there are no GPEs enabled within this
429              * register, we can safely ignore the entire register.
430              */
431             if (!(GpeRegisterInfo->EnableForRun |
432                 GpeRegisterInfo->EnableForWake))
433             {
434                 ACPI_DEBUG_PRINT ((ACPI_DB_INTERRUPTS,
435                     "Ignore disabled registers for GPE%02X-GPE%02X: "
436                     "RunEnable=%02X, WakeEnable=%02X\n",
437                     GpeRegisterInfo->BaseGpeNumber,
438                     GpeRegisterInfo->BaseGpeNumber + (ACPI_GPE_REGISTER_WIDTH - 1),
439                     GpeRegisterInfo->EnableForRun,
440                     GpeRegisterInfo->EnableForWake));
441                 continue;

```

```

442     }

444     /* Read the Status Register */

446     Status = AcpiHwRead (&StatusReg, &GpeRegisterInfo->StatusAddress);
447     if (ACPI_FAILURE (Status))
448     {
449         goto UnlockAndExit;
450     }

452     /* Read the Enable Register */

454     Status = AcpiHwRead (&EnableReg, &GpeRegisterInfo->EnableAddress);
455     if (ACPI_FAILURE (Status))
456     {
457         goto UnlockAndExit;
458     }

460     ACPI_DEBUG_PRINT ((ACPI_DB_INTERRUPTS,
461         "Read registers for GPE%02X-GPE%02X: Status=%02X, Enable=%02X, "
462         "RunEnable=%02X, WakeEnable=%02X\n",
463         GpeRegisterInfo->BaseGpeNumber,
464         GpeRegisterInfo->BaseGpeNumber + (ACPI_GPE_REGISTER_WIDTH - 1),
465         StatusReg, EnableReg,
466         GpeRegisterInfo->EnableForRun,
467         GpeRegisterInfo->EnableForWake));
468     /* Read GPE Register at GPE%02X: Status=%02X, Enable=%02X\n",
469     GpeRegisterInfo->BaseGpeNumber, StatusReg, EnableReg));

469     /* Check if there is anything active at all in this register */

471     EnabledStatusByte = (UINT8) (StatusReg & EnableReg);
472     if (!EnabledStatusByte)
473     {
474         /* No active GPEs in this register, move on */

476         continue;
477     }

479     /* Now look at the individual GPEs in this byte register */

481     for (j = 0; j < ACPI_GPE_REGISTER_WIDTH; j++)
482     {
483         /* Examine one GPE bit */

485         if (EnabledStatusByte & (1 << j))
486         {
487             /*
488              * Found an active GPE. Dispatch the event to a handler
489              * or method.
490              */
491             IntStatus |= AcpiEvGpeDispatch (GpeBlock->Node,
492                 &GpeBlock->EventInfo[(ACPI_SIZE) i *
493                     ACPI_GPE_REGISTER_WIDTH] + j],
494                 j + GpeRegisterInfo->BaseGpeNumber);
495         }
496     }

497     }

499     GpeBlock = GpeBlock->Next;
500 }

502 UnlockAndExit:

504     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
505     return (IntStatus);

```

```

506 }

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,
599                 Status = AcpiEvQueueNotifyRequest (
600                     LocalGpeEventInfo->Dispatch.DeviceNode,
601                     ACPI_NOTIFY_DEVICE_WAKE));

602             Notify = Notify->Next;
603         }
604         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:
636
637         return_VOID; /* Should never happen */
638     }
639
640     /* Defer enabling of GPE until all notify handlers are done */
641
642     Status = AcpiOsExecute (OSL_NOTIFY_HANDLER,
643         AcpiEvAsynchEnableGpe, LocalGpeEventInfo);
644     if (ACPI_FAILURE (Status))
645     {
646         ACPI_FREE (LocalGpeEventInfo);
647     }
648     return_VOID;
649 }
650
651 unchanged portion omitted
652
653
654 /*****
655  * FUNCTION:      AcpiEvGpeDispatch
656  *
657  * PARAMETERS:    GpeDevice          - Device node. NULL for GPE0/GPE1
658  *                GpeEventInfo       - Info for this GPE
659  *                GpeNumber          - Number relative to the parent GPE block
660  *
661  * RETURN:        INTERRUPT_HANDLED or INTERRUPT_NOT_HANDLED
662  *
663  * DESCRIPTION:    Dispatch a General Purpose Event to either a function (e.g. EC)
664  *                or method (e.g. _Lxx/_Exx) handler.
665  *
666  *                This function executes at interrupt level.
667  *****/
668
669 UINT32
670 AcpiEvGpeDispatch (
671     ACPI_NAMESPACE_NODE *GpeDevice,
672     ACPI_GPE_EVENT_INFO *GpeEventInfo,
673     UINT32 GpeNumber)
674 {
675     ACPI_STATUS Status;
676     UINT32 ReturnValue;
677
678     ACPI_FUNCTION_TRACE (EvGpeDispatch);
679
680     /* Invoke global event handler if present */
681
682     AcpiGpeCount++;
683     if (AcpiGbl_GlobalEventHandler)
684     {
685         AcpiGbl_GlobalEventHandler (ACPI_EVENT_TYPE_GPE, GpeDevice,
686             GpeNumber, AcpiGbl_GlobalEventHandlerContext);
687     }
688
689     /*
690      * If edge-triggered, clear the GPE status bit now. Note that
691      * level-triggered events are cleared after the GPE is serviced.
692      */
693     if ((GpeEventInfo->Flags & ACPI_GPE_XRUPT_TYPE_MASK) ==
694         ACPI_GPE_EDGE_TRIGGERED)
695     {
696         Status = AcpiHwClearGpe (GpeEventInfo);
697     }

```

```

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 }
778
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 }
795
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:
806
807         /* Invoke the installed handler (at interrupt level) */
808
809         ReturnValue = GpeEventInfo->Dispatch.Handler->Address (
810             GpeDevice, GpeNumber,
811             GpeEventInfo->Dispatch.Handler->Context);
812
813         /* If requested, clear (if level-triggered) and reenble the GPE */
814
815         if (ReturnValue & ACPI_REENABLE_GPE)
816         {
817             (void) AcpiEvFinishGpe (GpeEventInfo);
818         }
819         break;
820
821     case ACPI_GPE_DISPATCH_METHOD:
822     case ACPI_GPE_DISPATCH_NOTIFY:
823
824         /*
825          * Execute the method associated with the GPE
826          * NOTE: Level-triggered GPEs are cleared after the method completes.
827          */
828         Status = AcpiOsExecute (OSL_GPE_HANDLER,
829             AcpiEvAsynchExecuteGpeMethod, GpeEventInfo);
830         if (ACPI_FAILURE (Status))
831         {
832             ACPI_EXCEPTION ((AE_INFO, Status,
833                 "Unable to queue handler for GPE%02X - event disabled",
834                 GpeNumber));
835         }
836         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     }
848
849     return_UINT32 (ACPI_INTERRUPT_HANDLED);
850 }
851
852 #endif /* !ACPI_REDUCED_HARDWARE */
```



new/usr/src/common/acpica/components/events/evgpeblk.c

1

```
*****
16556 Fri Jan 17 07:30:03 2014
new/usr/src/common/acpica/components/events/evgpeblk.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: evgpeblk - GPE block creation and initialization.
4 *
5 *****/
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
45 #include "accommon.h"
46 #include "acevents.h"
47 #include "acnamesp.h"
48
49 #define _COMPONENT          ACPI_EVENTS
50 #define ACPI_MODULE_NAME    ("evgpeblk")
51
52 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */
53
54 /* Local prototypes */
55
56 static ACPI_STATUS
57 AcpiEvInstallGpeBlock (
```

new/usr/src/common/acpica/components/events/evgpeblk.c

2

```
58     ACPI_GPE_BLOCK_INFO    *GpeBlock,
59     UINT32                  InterruptNumber);
60
61 static ACPI_STATUS
62 AcpiEvCreateGpeInfoBlocks (
63     ACPI_GPE_BLOCK_INFO    *GpeBlock);
64
65
66 /*****
67 *
68 * FUNCTION:      AcpiEvInstallGpeBlock
69 *
70 * PARAMETERS:    GpeBlock          - New GPE block
71 *                InterruptNumber    - Xrupt to be associated with this
72 *                                GPE block
73 *
74 * RETURN:        Status
75 *
76 * DESCRIPTION:   Install new GPE block with mutex support
77 *
78 *****/
79
80 static ACPI_STATUS
81 AcpiEvInstallGpeBlock (
82     ACPI_GPE_BLOCK_INFO    *GpeBlock,
83     UINT32                  InterruptNumber)
84 {
85     ACPI_GPE_BLOCK_INFO    *NextGpeBlock;
86     ACPI_GPE_XRUPT_INFO    *GpeXruptBlock;
87     ACPI_STATUS             Status;
88     ACPI_CPU_FLAGS          Flags;
89
90     ACPI_FUNCTION_TRACE (EvInstallGpeBlock);
91
92
93     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
94     if (ACPI_FAILURE (Status))
95     {
96         return_ACPI_STATUS (Status);
97     }
98
99     Status = AcpiEvGetGpeXruptBlock (InterruptNumber, &GpeXruptBlock);
100     if (ACPI_FAILURE (Status))
101     {
102         GpeXruptBlock = AcpiEvGetGpeXruptBlock (InterruptNumber);
103         if (!GpeXruptBlock)
104         {
105             Status = AE_NO_MEMORY;
106             goto UnlockAndExit;
107         }
108     }
109
110     /* Install the new block at the end of the list with lock */
111
112     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);
113     if (GpeXruptBlock->GpeBlockListHead)
114     {
115         NextGpeBlock = GpeXruptBlock->GpeBlockListHead;
116         while (NextGpeBlock->Next)
117         {
118             NextGpeBlock = NextGpeBlock->Next;
119         }
120         NextGpeBlock->Next = GpeBlock;
121         GpeBlock->Previous = NextGpeBlock;
122     }
123     else
```

```

121 {
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",
449                             ACPI_DEBUG_PRINT ((ACPI_DB_INIT,
450                                                 "GPE %02X to %02X [%4.4s] %u regs on int 0x%X\n",
451                                                 (UINT32) GpeBlock->BlockBaseNumber,
452                                                 (UINT32) (GpeBlock->BlockBaseNumber + (GpeBlock->GpeCount - 1)),
453                                                 GpeDevice->Name.Ascii, GpeBlock->RegisterCount,
454                                                 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)));
547     ACPI_DEBUG_PRINT ((ACPI_DB_INIT,
548         "Enabled %u GPEs in this block\n", GpeEnabledCount));
549 }

549     GpeBlock->Initialized = TRUE;
550     return_ACPI_STATUS (AE_OK);
551 }

553 #endif /* !ACPI_REduced_HARDWARE */

```

new/usr/src/common/acpica/components/events/evgpeinit.c

1

```
*****
14987 Fri Jan 17 07:30:03 2014
new/usr/src/common/acpica/components/events/evgpeinit.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: evgpeinit - System GPE initialization and update
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "acevents.h"
48 #include "acnamesp.h"

50 #define _COMPONENT          ACPI_EVENTS
51         ACPI_MODULE_NAME    ("evgpeinit")

53 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */

55 /*
56 * Note: History of _PRW support in ACPIA
57 *
58 * Originally (2000 - 2010), the GPE initialization code performed a walk of
```

new/usr/src/common/acpica/components/events/evgpeinit.c

2

```
59 * the entire namespace to execute the _PRW methods and detect all GPEs
60 * capable of waking the system.
61 *
62 * As of 10/2010, the _PRW method execution has been removed since it is
63 * actually unnecessary. The host OS must in fact execute all _PRW methods
64 * in order to identify the device/power-resource dependencies. We now put
65 * the onus on the host OS to identify the wake GPEs as part of this process
66 * and to inform ACPIA of these GPEs via the AcpiSetupGpeForWake interface. Thi
67 * not only reduces the complexity of the ACPIA initialization code, but in
68 * some cases (on systems with very large namespaces) it should reduce the
69 * kernel boot time as well.
70 */

72 /*****
73 *
74 * FUNCTION:      AcpiEvGpeInitialize
75 *
76 * PARAMETERS:    None
77 *
78 * RETURN:        Status
79 *
80 * DESCRIPTION:   Initialize the GPE data structures and the FADT GPE 0/1 blocks
81 *
82 *****/

84 ACPI_STATUS
85 AcpiEvGpeInitialize (
86     void)
87 {
88     UINT32          RegisterCount0 = 0;
89     UINT32          RegisterCount1 = 0;
90     UINT32          GpeNumberMax = 0;
91     ACPI_STATUS      Status;

94     ACPI_FUNCTION_TRACE (EvGpeInitialize);

97     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT,
98         "Initializing General Purpose Events (GPEs):\n"));

100     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
101     if (ACPI_FAILURE (Status))
102     {
103         return ACPI_STATUS (Status);
104     }

106     /*
107     * Initialize the GPE Block(s) defined in the FADT
108     *
109     * Why the GPE register block lengths are divided by 2: From the ACPI
110     * Spec, section "General-Purpose Event Registers", we have:
111     *
112     * "Each register block contains two registers of equal length
113     * GPEx_STS and GPEx_EN (where x is 0 or 1). The length of the
114     * GPE0_STS and GPE0_EN registers is equal to half the GPE0_LEN
115     * The length of the GPE1_STS and GPE1_EN registers is equal to
116     * half the GPE1_LEN. If a generic register block is not supported
117     * then its respective block pointer and block length values in the
118     * FADT table contain zeros. The GPE0_LEN and GPE1_LEN do not need
119     * to be the same size."
120     */

122     /*
123     * Determine the maximum GPE number for this machine.
124     */
```

```

125  * 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) */
135
136      RegisterCount0 = (UINT16) (AcpiGbl_FADT.Gpe0BlockLength / 2);
137
138      GpeNumberMax = (RegisterCount0 * ACPI_GPE_REGISTER_WIDTH) - 1;
139
140      /* Install GPE Block 0 */
141
142      Status = AcpiEvCreateGpeBlock (AcpiGbl_FadtGpeDevice,
143                                     &AcpiGbl_FADT.XGpe0Block, RegisterCount0, 0,
144                                     AcpiGbl_FADT.SciInterrupt, &AcpiGbl_GpeFadtBlocks[0]);
145
146      if (ACPI_FAILURE (Status))
147      {
148          ACPI_EXCEPTION ((AE_INFO, Status,
149                          "Could not create GPE Block 0"));
150      }
151
152      if (AcpiGbl_FADT.Gpe1BlockLength &&
153          AcpiGbl_FADT.XGpe1Block.Address)
154      {
155          /* GPE block 1 exists (has both length and address > 0) */
156
157          RegisterCount1 = (UINT16) (AcpiGbl_FADT.Gpe1BlockLength / 2);
158
159          /* Check for GPE0/GPE1 overlap (if both banks exist) */
160
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));
170
171              /* Ignore GPE1 block by setting the register count to zero */
172
173              RegisterCount1 = 0;
174          }
175          else
176          {
177              /* Install GPE Block 1 */
178
179              Status = AcpiEvCreateGpeBlock (AcpiGbl_FadtGpeDevice,
180                                             &AcpiGbl_FADT.XGpe1Block, RegisterCount1,
181                                             AcpiGbl_FADT.Gpe1Base,
182                                             AcpiGbl_FADT.SciInterrupt, &AcpiGbl_GpeFadtBlocks[1]);
183
184              if (ACPI_FAILURE (Status))
185              {
186                  ACPI_EXCEPTION ((AE_INFO, Status,
187                                  "Could not create GPE Block 1"));
188              }
189          }
190      }
191  }

```

```

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  }
198
199  /* Exit if there are no GPE registers */
200
201  if ((RegisterCount0 + RegisterCount1) == 0)
202  {
203      /* GPEs are not required by ACPI, this is OK */
204
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  }
210
211  /* Check for Max GPE number out-of-range */
212
213  if (GpeNumberMax > ACPI_GPE_MAX)
214  {
215      ACPI_ERROR ((AE_INFO,
216                  "Maximum GPE number from FADT is too large: 0x%X",
217                  GpeNumberMax));
218      Status = AE_BAD_VALUE;
219      goto Cleanup;
220  }
221
222  Cleanup:
223  (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
224  return ACPI_STATUS (AE_OK);
225  }
226
227  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_STRTOL (&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%.2X and _E%.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 */

```

new/usr/src/common/acpica/components/events/evgpeutil.c

1

```
*****
12952 Fri Jan 17 07:30:04 2014
new/usr/src/common/acpica/components/events/evgpeutil.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: evgpeutil - GPE utilities
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
45 #include "accommon.h"
46 #include "acevents.h"
47
48 #define _COMPONENT          ACPI_EVENTS
49 #define ACPI_MODULE_NAME    ("evgpeutil")
50
51
52 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */
53 /*****
54 *
55 * FUNCTION:      AcpiEvWalkGpeList
56 *****/
```

new/usr/src/common/acpica/components/events/evgpeutil.c

2

```
57 * PARAMETERS:    GpeWalkCallback      - Routine called for each GPE block
58 *                Context                - Value passed to callback
59 *
60 * RETURN:         Status
61 *
62 * DESCRIPTION:    Walk the GPE lists.
63 *
64 *****/
65
66 ACPI_STATUS
67 AcpiEvWalkGpeList (
68     ACPI_GPE_CALLBACK    GpeWalkCallback,
69     void                  *Context)
70 {
71     ACPI_GPE_BLOCK_INFO  *GpeBlock;
72     ACPI_GPE_XRPT_INFO    *GpeXruptInfo;
73     ACPI_STATUS            Status = AE_OK;
74     ACPI_CPU_FLAGS        Flags;
75
76
77     ACPI_FUNCTION_TRACE (EvWalkGpeList);
78
79
80     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);
81
82     /* Walk the interrupt level descriptor list */
83
84     GpeXruptInfo = AcpiGbl_GpeXruptListHead;
85     while (GpeXruptInfo)
86     {
87         /* Walk all Gpe Blocks attached to this interrupt level */
88
89         GpeBlock = GpeXruptInfo->GpeBlockListHead;
90         while (GpeBlock)
91         {
92             /* One callback per GPE block */
93
94             Status = GpeWalkCallback (GpeXruptInfo, GpeBlock, Context);
95             if (ACPI_FAILURE (Status))
96             {
97                 if (Status == AE_CTRL_END) /* Callback abort */
98                 {
99                     Status = AE_OK;
100                 }
101                 goto UnlockAndExit;
102             }
103
104             GpeBlock = GpeBlock->Next;
105         }
106
107         GpeXruptInfo = GpeXruptInfo->Next;
108     }
109
110     UnlockAndExit:
111     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
112     return_ACPI_STATUS (Status);
113 }
114
115 _____unchanged_portion_omitted_____
116
117
118
119
120
121 /*****
122 *
123 * FUNCTION:      AcpiEvGetGpeXruptBlock
124 *
125 * PARAMETERS:    InterruptNumber        - Interrupt for a GPE block
126 *                GpeXruptBlock          - Where the block is returned
127 *****/
```

```

221 *
222 * RETURN:      Status
223 * RETURN:      A GPE interrupt block
224 * DESCRIPTION: Get or Create a GPE interrupt block. There is one interrupt
225 *              block per unique interrupt level used for GPEs. Should be
226 *              called only when the GPE lists are semaphore locked and not
227 *              subject to change.
228 *
229 *****/
231 ACPI_STATUS
232 AcpiEvGetGpeXruptBlock (
233     UINT32          InterruptNumber,
234     **GpeXruptBlock)
235 {
236     ACPI_GPE_XRUPT_INFO *NextGpeXrupt;
237     ACPI_GPE_XRUPT_INFO *GpeXrupt;
238     ACPI_STATUS Status;
239     ACPI_CPU_FLAGS Flags;
242     ACPI_FUNCTION_TRACE (EvGetGpeXruptBlock);
245     /* No need for lock since we are not changing any list elements here */
247     NextGpeXrupt = AcpiGbl_GpeXruptListHead;
248     while (NextGpeXrupt)
249     {
250         if (NextGpeXrupt->InterruptNumber == InterruptNumber)
251         {
252             *GpeXruptBlock = NextGpeXrupt;
253             return ACPI_STATUS (AE_OK);
254             return_PTR (NextGpeXrupt);
255         }
256         NextGpeXrupt = NextGpeXrupt->Next;
257     }
259     /* Not found, must allocate a new xrupt descriptor */
261     GpeXrupt = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_GPE_XRUPT_INFO));
262     if (!GpeXrupt)
263     {
264         return ACPI_STATUS (AE_NO_MEMORY);
265         return_PTR (NULL);
266     }
267     GpeXrupt->InterruptNumber = InterruptNumber;
269     /* Install new interrupt descriptor with spin lock */
271     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);
272     if (AcpiGbl_GpeXruptListHead)
273     {
274         NextGpeXrupt = AcpiGbl_GpeXruptListHead;
275         while (NextGpeXrupt->Next)
276         {
277             NextGpeXrupt = NextGpeXrupt->Next;
278         }
280         NextGpeXrupt->Next = GpeXrupt;
281         GpeXrupt->Previous = NextGpeXrupt;

```

```

282     }
283     else
284     {
285         AcpiGbl_GpeXruptListHead = GpeXrupt;
286     }
288     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
290     /* Install new interrupt handler if not SCI_INT */
292     if (InterruptNumber != AcpiGbl_FADT.SciInterrupt)
293     {
294         Status = AcpiOsInstallInterruptHandler (InterruptNumber,
295         AcpiEvGpeXruptHandler, GpeXrupt);
296         if (ACPI_FAILURE (Status))
297         {
298             ACPI_EXCEPTION ((AE_INFO, Status,
299             ACPI_ERROR ((AE_INFO,
300             "Could not install GPE interrupt handler at level 0x%X",
301             InterruptNumber));
302             return ACPI_STATUS (Status);
303             return_PTR (NULL);
304         }
305     }
306     *GpeXruptBlock = GpeXrupt;
307     return ACPI_STATUS (AE_OK);
308     return_PTR (GpeXrupt);
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```



```
412      /* Now look at the individual GPEs in this byte register */
413
414      for (j = 0; j < ACPI_GPE_REGISTER_WIDTH; j++)
415      {
416          GpeEventInfo = &GpeBlock->EventInfo[(ACPI_SIZE) i *
417              ACPI_GPE_REGISTER_WIDTH + j];
418
419          if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) ==
420              ACPI_GPE_DISPATCH_HANDLER)
421          {
422              /* Delete an installed handler block */
423
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 */
432
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
445      return_ACPI_STATUS (AE_OK);
446  }
447
448 #endif /* !ACPI_REDUCED_HARDWARE */
```

new/usr/src/common/acpica/components/events/evhandler.c

1

```
*****
17268 Fri Jan 17 07:30:04 2014
new/usr/src/common/acpica/components/events/evhandler.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: evhandler - Support for Address Space handlers
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EVHANDLER_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acevents.h"
50 #include "acnamesp.h"
51 #include "acinterp.h"

53 #define _COMPONENT          ACPI_EVENTS
54         ACPI_MODULE_NAME    ("evhandler")

57 /* Local prototypes */

59 static ACPI_STATUS
60 AcpiEvInstallHandler (
```

new/usr/src/common/acpica/components/events/evhandler.c

2

```
61     ACPI_HANDLE            ObjHandle,
62     UINT32                 Level,
63     void                   *Context,
64     void                   **ReturnValue);

66 /* These are the address spaces that will get default handlers */

68 UINT8                      AcpiGbl_DefaultAddressSpaces[ACPI_NUM_DEFAULT_SPACES] =
69 {
70     ACPI_ADR_SPACE_SYSTEM_MEMORY,
71     ACPI_ADR_SPACE_SYSTEM_IO,
72     ACPI_ADR_SPACE_PCI_CONFIG,
73     ACPI_ADR_SPACE_DATA_TABLE
74 };

77 /*****
78 *
79 * FUNCTION:      AcpiEvInstallRegionHandlers
80 *
81 * PARAMETERS:    None
82 *
83 * RETURN:        Status
84 *
85 * DESCRIPTION:   Installs the core subsystem default address space handlers.
86 *
87 *****/

89 ACPI_STATUS
90 AcpiEvInstallRegionHandlers (
91     void)
92 {
93     ACPI_STATUS            Status;
94     UINT32                 i;

97     ACPI_FUNCTION_TRACE (EvInstallRegionHandlers);

100     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
101     if (ACPI_FAILURE (Status))
102     {
103         return_ACPI_STATUS (Status);
104     }

106     /*
107     * All address spaces (PCI Config, EC, SMBus) are scope dependent and
108     * registration must occur for a specific device.
109     *
110     * In the case of the system memory and IO address spaces there is
111     * currently no device associated with the address space. For these we
112     * use the root.
113     *
114     * We install the default PCI config space handler at the root so that
115     * this space is immediately available even though the we have not
116     * enumerated all the PCI Root Buses yet. This is to conform to the ACPI
117     * specification which states that the PCI config space must be always
118     * available -- even though we are nowhere near ready to find the PCI root
119     * buses at this point.
120     *
121     * NOTE: We ignore AE_ALREADY_EXISTS because this means that a handler
122     * has already been installed (via AcpiInstallAddressSpaceHandler).
123     * Similar for AE_SAME_HANDLER.
124     */
125     for (i = 0; i < ACPI_NUM_DEFAULT_SPACES; i++)
126     {
```

```

127     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;
372     ACPI_FUNCTION_TRACE (EvInstallSpaceHandler);
374     /*
375      * This registration is valid for only the types below and the root. This
376      * is where the default handlers get placed.
377      */
378     if ((Node->Type != ACPI_TYPE_DEVICE) &&
379         (Node->Type != ACPI_TYPE_PROCESSOR) &&
380         (Node->Type != ACPI_TYPE_THERMAL) &&
381         (Node != AcpiGbl_RootNode))
382     {
383         Status = AE_BAD_PARAMETER;
384         goto UnlockAndExit;
385     }
386     if (Handler == ACPI_DEFAULT_HANDLER)
387     {

```

```

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                 goto UnlockAndExit;
480             }
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 }
```

new/usr/src/common/acpica/components/events/evmisc.c

1

```
*****
9419 Fri Jan 17 07:30:04 2014
new/usr/src/common/acpica/components/events/evmisc.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: evmisc - Miscellaneous event manager support functions
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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    ("evmisc")

53 /* Local prototypes */

55 static void ACPI_SYSTEM_XFACE
56 AcpiEvNotifyDispatch (
57     void                *Context);
```

new/usr/src/common/acpica/components/events/evmisc.c

2

```
60 /*****
61 *
62 * FUNCTION:      AcpiEvIsNotifyObject
63 *
64 * PARAMETERS:    Node                - Node to check
65 *
66 * RETURN:        TRUE if notifies allowed on this object
67 *
68 * DESCRIPTION:    Check type of node for a object that supports notifies.
69 *
70 *                TBD: This could be replaced by a flag bit in the node.
71 *
72 *****/

74 BOOLEAN
75 AcpiEvIsNotifyObject (
76     ACPI_NAMESPACE_NODE *Node)
77 {
78     switch (Node->Type)
79     {
80     case ACPI_TYPE_DEVICE:
81     case ACPI_TYPE_PROCESSOR:
82     case ACPI_TYPE_THERMAL:
83         /*
84          * These are the ONLY objects that can receive ACPI notifications
85          */
86         return (TRUE);

88     default:

90         return (FALSE);
91     }
92 }

95 /*****
96 *
97 * FUNCTION:      AcpiEvQueueNotifyRequest
98 *
99 * PARAMETERS:    Node                - NS node for the notified object
100 *                NotifyValue         - Value from the Notify() request
101 *
102 * RETURN:        Status
103 *
104 * DESCRIPTION:    Dispatch a device notification event to a previously
105 *                installed handler.
106 *
107 *****/

109 ACPI_STATUS
110 AcpiEvQueueNotifyRequest (
111     ACPI_NAMESPACE_NODE *Node,
112     UINT32               NotifyValue)
113 {
114     ACPI_OPERAND_OBJECT *ObjDesc;
115     ACPI_OPERAND_OBJECT *HandlerListHead = NULL;
116     ACPI_GENERIC_STATE *Info;
117     UINT8               HandlerListId = 0;
118     ACPI_STATUS          Status = AE_OK;

121     ACPI_FUNCTION_NAME (EvQueueNotifyRequest);

124     /* Are Notifies allowed on this object? */
```

```

126     if (!AcpiEvIsNotifyObject (Node))
127     {
128         return (AE_TYPE);
129     }

131     /* Get the correct notify list type (System or Device) */

133     if (NotifyValue <= ACPI_MAX_SYS_NOTIFY)
134     {
135         HandlerListId = ACPI_SYSTEM_HANDLER_LIST;
136     }
137     else
138     {
139         HandlerListId = ACPI_DEVICE_HANDLER_LIST;
140     }

142     /* Get the notify object attached to the namespace Node */

144     ObjDesc = AcpiNsGetAttachedObject (Node);
145     if (ObjDesc)
146     {
147         /* We have an attached object, Get the correct handler list */

149         HandlerListHead = ObjDesc->CommonNotify.NotifyList[HandlerListId];
150     }

152     /*
153      * If there is no notify handler (Global or Local)
154      * for this object, just ignore the notify
155      */
156     if (!AcpiGbl_GlobalNotify[HandlerListId].Handler && !HandlerListHead)
157     {
158         ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
159             "No notify handler for Notify, ignoring (%4.4s, %X) node %p\n",
160             AcpiUtGetNodeName (Node), NotifyValue, Node));

162         return (AE_OK);
163     }

165     /* Setup notify info and schedule the notify dispatcher */

167     Info = AcpiUtCreateGenericState ();
168     if (!Info)
169     {
170         return (AE_NO_MEMORY);
171     }

173     Info->Common.DescriptorType = ACPI_DESC_TYPE_STATE_NOTIFY;

175     Info->Notify.Node = Node;
176     Info->Notify.Value = (UINT16) NotifyValue;
177     Info->Notify.HandlerListId = HandlerListId;
178     Info->Notify.HandlerListHead = HandlerListHead;
179     Info->Notify.Global = &AcpiGbl_GlobalNotify[HandlerListId];

181     ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
182         "Dispatching Notify on [%4.4s] (%s) Value 0x%2.2X (%s) Node %p\n",
183         AcpiUtGetNodeName (Node), AcpiUtGetTypeName (Node->Type),
184         NotifyValue, AcpiUtGetNotifyName (NotifyValue), Node));

186     Status = AcpiOsExecute (OSL_NOTIFY_HANDLER, AcpiEvNotifyDispatch,
187         Info);
188     if (ACPI_FAILURE (Status))
189     {
190         AcpiUtDeleteGenericState (Info);
191     }

```

```

193     return (Status);
194 }

197 /*****
198  *
199  * FUNCTION:      AcpiEvNotifyDispatch
200  *
201  * PARAMETERS:    Context          - To be passed to the notify handler
202  *
203  * RETURN:        None.
204  *
205  * DESCRIPTION:   Dispatch a device notification event to a previously
206  *                installed handler.
207  *
208  *****/

210 static void ACPI_SYSTEM_XFACE
211 AcpiEvNotifyDispatch (
212     void                *Context)
213 {
214     ACPI_GENERIC_STATE  *Info = (ACPI_GENERIC_STATE *) Context;
215     ACPI_OPERAND_OBJECT *HandlerObj;

218     ACPI_FUNCTION_ENTRY ();

221     /* Invoke a global notify handler if installed */

223     if (Info->Notify.Global->Handler)
224     {
225         Info->Notify.Global->Handler (Info->Notify.Node,
226             Info->Notify.Value,
227             Info->Notify.Global->Context);
228     }

230     /* Now invoke the local notify handler(s) if any are installed */

232     HandlerObj = Info->Notify.HandlerListHead;
233     while (HandlerObj)
234     {
235         HandlerObj->Notify.Handler (Info->Notify.Node,
236             Info->Notify.Value,
237             HandlerObj->Notify.Context);

239         HandlerObj = HandlerObj->Notify.Next[Info->Notify.HandlerListId];
240     }

242     /* All done with the info object */

244     AcpiUtDeleteGenericState (Info);
245 }

248 #if (!ACPI_REDUCED_HARDWARE)
249 /*****
250  *
251  * FUNCTION:      AcpiEvTerminate
252  *
253  * PARAMETERS:    none
254  *
255  * RETURN:        none
256  *
257  * DESCRIPTION:   Disable events and free memory allocated for table storage.

```



```

258 *
259 *****/

261 void
262 AcpiEvTerminate (
263     void)
264 {
265     UINT32          i;
266     ACPI_STATUS      Status;

269     ACPI_FUNCTION_TRACE (EvTerminate);

272     if (AcpiGbl_EventsInitialized)
273     {
274         /*
275          * Disable all event-related functionality. In all cases, on error,
276          * print a message but obviously we don't abort.
277          */

279         /* Disable all fixed events */

281         for (i = 0; i < ACPI_NUM_FIXED_EVENTS; i++)
282         {
283             Status = AcpiDisableEvent (i, 0);
284             if (ACPI_FAILURE (Status))
285             {
286                 ACPI_ERROR ((AE_INFO,
287                     "Could not disable fixed event %u", (UINT32) i));
288             }
289         }

291         /* Disable all GPEs in all GPE blocks */

293         Status = AcpiEvWalkGpeList (AcpiHwDisableGpeBlock, NULL);

295         Status = AcpiEvRemoveGlobalLockHandler ();
296         if (ACPI_FAILURE (Status))
297         {
298             ACPI_ERROR ((AE_INFO,
299                 "Could not remove Global Lock handler"));
300         }

302         AcpiGbl_EventsInitialized = FALSE;
303     }

305     /* Remove SCI handlers */

307     Status = AcpiEvRemoveAllSciHandlers ();
308     if (ACPI_FAILURE (Status))
309     {
310         ACPI_ERROR ((AE_INFO,
311             "Could not remove SCI handler"));
312     }

314     /* Deallocate all handler objects installed within GPE info structs */

316     Status = AcpiEvWalkGpeList (AcpiEvDeleteGpeHandlers, NULL);

318     /* Return to original mode if necessary */

320     if (AcpiGbl_OriginalMode == ACPI_SYS_MODE_LEGACY)
321     {
322         Status = AcpiDisable ();
323         if (ACPI_FAILURE (Status))

```

```

324     {
325         ACPI_WARNING ((AE_INFO, "AcpiDisable failed"));
326     }
327 }
328     return_VOID;
329 }

331 #endif /* !ACPI_REduced_HARDWARE */

```

new/usr/src/common/acpica/components/events/evregion.c

1

```
*****
24892 Fri Jan 17 07:30:05 2014
new/usr/src/common/acpica/components/events/evregion.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: evregion - Operation Region support
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EVREGION_C__
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acevents.h"
50 #include "acnamesp.h"
51 #include "acinterp.h"

53 #define _COMPONENT          ACPI_EVENTS
54         ACPI_MODULE_NAME    ("evregion")

57 extern UINT8              AcpiGbl_DefaultAddressSpaces[];

59 /* Local prototypes */
```

new/usr/src/common/acpica/components/events/evregion.c

2

```
61 static void
62 AcpiEvOrphanEcRegMethod (
63     ACPI_NAMESPACE_NODE    *EcDeviceNode);

65 static ACPI_STATUS
66 AcpiEvRegRun (
67     ACPI_HANDLE             ObjHandle,
68     UINT32                  Level,
69     void                    *Context,
70     void                    **ReturnValue);

73 /*****
74 *
75 * FUNCTION:      AcpiEvInitializeOpRegions
76 *
77 * PARAMETERS:    None
78 *
79 * RETURN:        Status
80 *
81 * DESCRIPTION:   Execute _REG methods for all Operation Regions that have
82 *                an installed default region handler.
83 *
84 *****/

86 ACPI_STATUS
87 AcpiEvInitializeOpRegions (
88     void)
89 {
90     ACPI_STATUS              Status;
91     UINT32                   i;

94     ACPI_FUNCTION_TRACE (EvInitializeOpRegions);

97     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
98     if (ACPI_FAILURE (Status))
99     {
100         return_ACPI_STATUS (Status);
101     }

103     /* Run the _REG methods for OpRegions in each default address space */

105     for (i = 0; i < ACPI_NUM_DEFAULT_SPACES; i++)
106     {
107         /*
108          * Make sure the installed handler is the DEFAULT handler. If not the
109          * default, the _REG methods will have already been run (when the
110          * handler was installed)
111          */
112         if (AcpiEvHasDefaultHandler (AcpiGbl_RootNode,
113             AcpiGbl_DefaultAddressSpaces[i]))
114         {
115             Status = AcpiEvExecuteRegMethods (AcpiGbl_RootNode,
116                 AcpiGbl_DefaultAddressSpaces[i]);
117         }
118     }

120     AcpiGbl_RegMethodsExecuted = TRUE;

122     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
123     return_ACPI_STATUS (Status);
124 }
```

```

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 */
272
273     Context->Connection = FieldObj->Field.ResourceBuffer;
274     Context->Length = FieldObj->Field.ResourceLength;
275     Context->AccessLength = FieldObj->Field.AccessLength;
276 }
277
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 interpreter.
285     */
286     AcpiExExitInterpreter();
287 }
288
289 /* Call the handler */
290
291 Status = Handler (Function,
292                  (RegionObj->Region.Address + RegionOffset), BitWidth, Value,
293                  Context, RegionObj2->Extra.RegionContext);
294
295 if (ACPI_FAILURE (Status))
296 {
297     ACPI_EXCEPTION ((AE_INFO, Status, "Returned by Handler for [%s]",
298                    AcpiUtGetRegionName (RegionObj->Region.SpaceId)));
299 }
300
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 }
310
311 return ACPI_STATUS (Status);
312 }
313
314 /*****
315  *
316  * FUNCTION:      AcpiEvDetachRegion
317  *
318  * PARAMETERS:    RegionObj          - Region Object
319  *                AcpiNsIsLocked     - Namespace Region Already Locked?
320  *
321  * RETURN:        None
322  */
323

```

```

324 * DESCRIPTION: Break the association between the handler and the region
325 *              this is a two way association.
326 *
327 *****/
328
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;
341
342
343     ACPI_FUNCTION_TRACE (EvDetachRegion);
344
345     RegionObj2 = AcpiNsGetSecondaryObject (RegionObj);
346     if (!RegionObj2)
347     {
348         return_VOID;
349     }
350     RegionContext = &RegionObj2->Extra.RegionContext;
351
352     /* Get the address handler from the region object */
353
354     HandlerObj = RegionObj->Region.Handler;
355     if (!HandlerObj)
356     {
357         /* This region has no handler, all done */
358         return_VOID;
359     }
360
361     /* Find this region in the handler's list */
362
363     ObjDesc = HandlerObj->AddressSpace.RegionList;
364     LastObjPtr = &HandlerObj->AddressSpace.RegionList;
365
366     while (ObjDesc)
367     {
368         /* Is this the correct Region? */
369
370         if (ObjDesc == RegionObj)
371         {
372             ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION,
373                               "Removing Region %p from address handler %p\n",
374                               RegionObj, HandlerObj));
375
376             /* This is it, remove it from the handler's list */
377
378             *LastObjPtr = ObjDesc->Region.Next;
379             ObjDesc->Region.Next = NULL; /* Must clear field */
380
381             if (AcpiNsIsLocked)
382             {
383                 Status = AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
384                 if (ACPI_FAILURE (Status))
385                 {
386                     return_VOID;
387                 }
388             }
389         }
390     }
391 }

```

```

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 /*****
523 *
524 * FUNCTION:      AcpiEvExecuteRegMethod
525 *
526 * PARAMETERS:    RegionObj      - Region object
527 *                Function        - Passed to _REG: On (1) or Off (0)
528 *
529 * RETURN:        Status
530 *
531 * DESCRIPTION:    Execute _REG method for a region
532 *
533 *****/

535 ACPI_STATUS
536 AcpiEvExecuteRegMethod (
537     ACPI_OPERAND_OBJECT *RegionObj,
538     UINT32                Function)
539 {
540     ACPI_EVALUATE_INFO *Info;
541     ACPI_OPERAND_OBJECT *Args[3];
542     ACPI_OPERAND_OBJECT *RegionObj2;
543     ACPI_STATUS          Status;

546     ACPI_FUNCTION_TRACE (EvExecuteRegMethod);

549     RegionObj2 = AcpiNsGetSecondaryObject (RegionObj);
550     if (!RegionObj2)
551     {
552         return_ACPI_STATUS (AE_NOT_EXIST);
553     }

555     if (RegionObj2->Extra.Method_REG == NULL)
556     {
557         return_ACPI_STATUS (AE_OK);
558     }

560     /* Allocate and initialize the evaluation information block */

562     Info = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_EVALUATE_INFO));
563     if (!Info)
564     {
565         return_ACPI_STATUS (AE_NO_MEMORY);
566     }

568     Info->PrefixNode = RegionObj2->Extra.Method_REG;
569     Info->RelativePathname = NULL;
570     Info->Parameters = Args;
571     Info->Flags = ACPI_IGNORE_RETURN_VALUE;

573     /*
574     * The _REG method has two arguments:
575     *
576     * Arg0 - Integer:
577     *   Operation region space ID Same value as RegionObj->Region.SpaceId
578     *
579     * Arg1 - Integer:
580     *   connection status 1 for connecting the handler, 0 for disconnecting
581     *   the handler (Passed as a parameter)
582     */
583     Args[0] = AcpiUtCreateIntegerObject ((UINT64) RegionObj->Region.SpaceId);
584     if (!Args[0])
585     {
586         Status = AE_NO_MEMORY;
587         goto Cleanup1;

```

```

588     }

590     Args[1] = AcpiUtCreateIntegerObject ((UINT64) Function);
591     if (!Args[1])
592     {
593         Status = AE_NO_MEMORY;
594         goto Cleanup2;
595     }

597     Args[2] = NULL; /* Terminate list */

599     /* Execute the method, no return value */

601     ACPI_DEBUG_EXEC (
602         AcpiUtDisplayInitPathname (ACPI_TYPE_METHOD, Info->PrefixNode, NULL));

604     Status = AcpiNsEvaluate (Info);
605     AcpiUtRemoveReference (Args[1]);

607 Cleanup2:
608     AcpiUtRemoveReference (Args[0]);

610 Cleanup1:
611     ACPI_FREE (Info);
612     return_ACPI_STATUS (Status);
613 }

616 /*****
617 *
618 * FUNCTION:      AcpiEvExecuteRegMethods
619 *
620 * PARAMETERS:    Node          - Namespace node for the device
621 *                SpaceId       - The address space ID
622 *
623 * RETURN:        Status
624 *
625 * DESCRIPTION:    Run all _REG methods for the input Space ID;
626 *                Note: assumes namespace is locked, or system init time.
627 *
628 *****/

630 ACPI_STATUS
631 AcpiEvExecuteRegMethods (
632     ACPI_NAMESPACE_NODE *Node,
633     ACPI_ADR_SPACE_TYPE SpaceId)
634 {
635     ACPI_STATUS          Status;

638     ACPI_FUNCTION_TRACE (EvExecuteRegMethods);

641     /*
642     * Run all _REG methods for all Operation Regions for this space ID. This
643     * is a separate walk in order to handle any interdependencies between
644     * regions and _REG methods. (i.e. handlers must be installed for all
645     * regions of this Space ID before we can run any _REG methods)
646     */
647     Status = AcpiNsWalkNamespace (ACPI_TYPE_ANY, Node, ACPI_UINT32_MAX,
648         ACPI_NS_WALK_UNLOCK, AcpiEvRegRun, NULL,
649         &SpaceId, NULL);

651     /* Special case for EC: handle "orphan" _REG methods with no region */
653     if (SpaceId == ACPI_ADR_SPACE_EC)

```

```

654 {
655     AcpiEvOrphanEcRegMethod (Node);
656 }

658 return ACPI_STATUS (Status);
659 }

662 /*****
663 *
664 * FUNCTION:      AcpiEvRegRun
665 * PARAMETERS:    WalkNamespace callback
666 * DESCRIPTION:   Run _REG method for region objects of the requested spaceID
667 *
668 * *****/
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 */
711         return (AE_OK);
712     }

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      }
801
802      NextNode = AcpiNsGetNextNode (EcDeviceNode, NextNode);
803  }
804
805  /* Evaluate the _REG(EmbeddedControl,Connect) method */
806
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;
813
814  Status = AcpiEvaluateObject (RegMethod, NULL, &Args, NULL);
815
816 Exit:
817  /* We ignore all errors from above, don't care */
818
819  Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
820  return_VOID;
821 }
```



new/usr/src/common/acpica/components/events/evrgnini.c

1

```
*****
22020 Fri Jan 17 07:30:05 2014
new/usr/src/common/acpica/components/events/evrgnini.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: evrgnini- ACPI AddressSpace (OpRegion) init
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EVGRNINI_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acevents.h"
50 #include "acnamesp.h"

52 #define _COMPONENT ACPI_EVENTS
53 ACPI_MODULE_NAME ("evrgnini")

55 /* Local prototypes */

57 static BOOLEAN
58 AcpiEvIsPciRootBridge (
```

new/usr/src/common/acpica/components/events/evrgnini.c

2

```
59 ACPI_NAMESPACE_NODE *Node);

62 /*****
63 *
64 * FUNCTION: AcpiEvSystemMemoryRegionSetup
65 *
66 * PARAMETERS: Handle - Region we are interested in
67 * Function - Start or stop
68 * HandlerContext - Address space handler context
69 * RegionContext - Region specific context
70 *
71 * RETURN: Status
72 *
73 * DESCRIPTION: Setup a SystemMemory operation region
74 *
75 *****/

77 ACPI_STATUS
78 AcpiEvSystemMemoryRegionSetup (
79 ACPI_HANDLE Handle,
80 UINT32 Function,
81 void *HandlerContext,
82 void **RegionContext)
83 {
84 ACPI_OPERAND_OBJECT *RegionDesc = (ACPI_OPERAND_OBJECT *) Handle;
85 ACPI_MEM_SPACE_CONTEXT *LocalRegionContext;

88 ACPI_FUNCTION_TRACE (EvSystemMemoryRegionSetup);

91 if (Function == ACPI_REGION_DEACTIVATE)
92 {
93 if (*RegionContext)
94 {
95 LocalRegionContext = (ACPI_MEM_SPACE_CONTEXT *) *RegionContext;

97 /* Delete a cached mapping if present */

99 if (LocalRegionContext->MappedLength)
100 {
101 AcpiOsUnmapMemory (LocalRegionContext->MappedLogicalAddress,
102 LocalRegionContext->MappedLength);
103 }
104 ACPI_FREE (LocalRegionContext);
105 *RegionContext = NULL;
106 }
107 return_ACPI_STATUS (AE_OK);
108 }

110 /* Create a new context */

112 LocalRegionContext = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_MEM_SPACE_CONTEXT));
113 if (!LocalRegionContext)
114 {
115 return_ACPI_STATUS (AE_NO_MEMORY);
116 }

118 /* Save the region length and address for use in the handler */

120 LocalRegionContext->Length = RegionDesc->Region.Length;
121 LocalRegionContext->Address = RegionDesc->Region.Address;

123 *RegionContext = LocalRegionContext;
124 return_ACPI_STATUS (AE_OK);
```

```

125 }
    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                   * See AcpiNsExecModuleCode
658                   */
659                  if (ObjDesc->Method.InfoFlags & ACPI_METHOD_MODULE_LEVEL)
660                  {
661                      HandlerObj = ObjDesc->Method.Dispatch.Handler;
662                  }
663                  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));

```

`new/usr/src/common/acpica/components/events/evrgnini.c`

7

```
732     return_ACPI_STATUS (AE_NOT_EXIST);  
733 }
```

new/usr/src/common/acpica/components/events/evsci.c

1

```
*****
8237 Fri Jan 17 07:30:05 2014
new/usr/src/common/acpica/components/events/evsci.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: evsci - System Control Interrupt configuration and
4 *                 legacy to ACPI mode state transition functions
5 *
6 *****/

8 /*
9  * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "acevents.h"

50 #define _COMPONENT          ACPI_EVENTS
51     ACPI_MODULE_NAME    ("evsci")

53 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */

55 /* Local prototypes */

57 static UINT32 ACPI_SYSTEM_XFACE
58 AcpiEvSciXruptHandler (
59     void                *Context);
```

new/usr/src/common/acpica/components/events/evsci.c

2

```
62 /*****
63 *
64 * FUNCTION:        AcpiEvSciDispatch
65 *
66 * PARAMETERS:      None
67 *
68 * RETURN:          Status code indicates whether interrupt was handled.
69 *
70 * DESCRIPTION:     Dispatch the SCI to all host-installed SCI handlers.
71 *
72 *****/

74 UINT32
75 AcpiEvSciDispatch (
76     void)
77 {
78     ACPI_SCI_HANDLER_INFO    *SciHandler;
79     ACPI_CPU_FLAGS            Flags;
80     UINT32                    IntStatus = ACPI_INTERRUPT_NOT_HANDLED;

83     ACPI_FUNCTION_NAME (EvSciDispatch);

86     /* Are there any host-installed SCI handlers? */

88     if (!AcpiGbl_SciHandlerList)
89     {
90         return (IntStatus);
91     }

93     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);

95     /* Invoke all host-installed SCI handlers */

97     SciHandler = AcpiGbl_SciHandlerList;
98     while (SciHandler)
99     {
100         /* Invoke the installed handler (at interrupt level) */

102         IntStatus |= SciHandler->Address (
103             SciHandler->Context);

105         SciHandler = SciHandler->Next;
106     }

108     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
109     return (IntStatus);
110 }

113 /*****
114 *
115 * FUNCTION:        AcpiEvSciXruptHandler
116 *
117 * PARAMETERS:      Context - Calling Context
118 *
119 * RETURN:          Status code indicates whether interrupt was handled.
120 *
121 * DESCRIPTION:     Interrupt handler that will figure out what function or
122 *                  control method to call to deal with a SCI.
123 *
124 *****/
```

```

126 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  * PARAMETERS:    Context - Calling Context
167  * RETURN:        Status code indicates whether interrupt was handled.
168  * DESCRIPTION:   Handler for GPE Block Device interrupts
169  *****/
170
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  * PARAMETERS:    none
202  * RETURN:        Status
203  * DESCRIPTION:   Installs SCI handler.
204  *****/
205
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  * PARAMETERS:    none
230  * RETURN:        AE_OK if handler uninstalled, AE_ERROR if handler was not
231                  installed to begin with
232  * DESCRIPTION:   Remove the SCI interrupt handler. No further SCIs will be
233                  taken. Remove all host-installed SCI handlers.
234  * Note:         It doesn't seem important to disable all events or set the event
235                  enable registers to their original values. The OS should disable
236                  the SCI interrupt level when the handler is removed, so no more
237                  events will come in.
238  *****/
239
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);  
  
262     if (!AcpiGbl_SciHandlerList)  
263     {  
264         return (Status);  
265     }  
  
267     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);  
  
269     /* Free all host-installed SCI handlers */  
  
271     while (AcpiGbl_SciHandlerList)  
272     {  
273         SciHandler = AcpiGbl_SciHandlerList;  
274         AcpiGbl_SciHandlerList = SciHandler->Next;  
275         ACPI_FREE (SciHandler);  
276     }  
  
278     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);  
279     return_ACPI_STATUS (Status);  
280 }  
  
282 #endif /* !ACPI_REduced_HARDWARE */
```

new/usr/src/common/acpica/components/events/evxface.c

1

```
*****
30979 Fri Jan 17 07:30:05 2014
new/usr/src/common/acpica/components/events/evxface.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: evxface - External interfaces for ACPI events
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define __EVXFACE_C__
45 #define EXPORT_ACPI_INTERFACES
46
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acnamesp.h"
50 #include "acevents.h"
51 #include "acinterp.h"
52
53 #define _COMPONENT          ACPI_EVENTS
54 #define ACPI_MODULE_NAME    ("evxface")
55
56 /*****
```

new/usr/src/common/acpica/components/events/evxface.c

2

```
59 *
60 * FUNCTION:      AcpiInstallNotifyHandler
61 *
62 * PARAMETERS:    Device          - The device for which notifies will be handled
63 *               HandlerType      - The type of handler:
64 *                               ACPI_SYSTEM_NOTIFY: System Handler (00-7F)
65 *                               ACPI_DEVICE_NOTIFY: Device Handler (80-FF)
66 *                               ACPI_ALL_NOTIFY:   Both System and Device
67 *               Handler          - Address of the handler
68 *               Context          - Value passed to the handler on each GPE
69 *
70 * RETURN:        Status
71 *
72 * DESCRIPTION:    Install a handler for notifications on an ACPI Device,
73 *                 ThermalZone, or Processor object.
74 *
75 * NOTES:          The Root namespace object may have only one handler for each
76 *                 type of notify (System/Device). Device/Thermal/Processor objects
77 *                 may have one device notify handler, and multiple system notify
78 *                 handlers.
79 *
80 *****/
81
82 ACPI_STATUS
83 AcpiInstallNotifyHandler (
84     ACPI_HANDLE      Device,
85     UINT32            HandlerType,
86     ACPI_NOTIFY_HANDLER Handler,
87     void              *Context)
88 {
89     ACPI_NAMESPACE_NODE *Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Device);
90     ACPI_OPERAND_OBJECT *ObjDesc;
91     ACPI_OPERAND_OBJECT *HandlerObj;
92     ACPI_STATUS Status;
93     UINT32 i;
94
95     ACPI_FUNCTION_TRACE (AcpiInstallNotifyHandler);
96
97     /* Parameter validation */
98
99     if ((!Device) || (!Handler) || (!HandlerType) ||
100         (HandlerType > ACPI_MAX_NOTIFY_HANDLER_TYPE))
101     {
102         return_ACPI_STATUS (AE_BAD_PARAMETER);
103     }
104
105     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
106     if (ACPI_FAILURE (Status))
107     {
108         return_ACPI_STATUS (Status);
109     }
110
111     /*
112     * Root Object:
113     * Registering a notify handler on the root object indicates that the
114     * caller wishes to receive notifications for all objects. Note that
115     * only one global handler can be registered per notify type.
116     * Ensure that a handler is not already installed.
117     */
118     if (Device == ACPI_ROOT_OBJECT)
119     {
120         for (i = 0; i < ACPI_NUM_NOTIFY_TYPES; i++)
121         {
122             if (HandlerType & (1<i>))
123             {
124                 if (HandlerType & (1<i>))
```



```

125     {
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 UnlockAndExit:
234 (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
235 return ACPI_STATUS (Status);
236 }
237
238 ACPI_EXPORT_SYMBOL (AcpiInstallNotifyHandler)
239
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 *****/
259 ACPI_STATUS
260 AcpiRemoveNotifyHandler (
261     ACPI_HANDLE          Device,
262     UINT32               HandlerType,
263     ACPI_NOTIFY_HANDLER   Handler)
264 {
265     ACPI_NAMESPACE_NODE *Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Device);
266     ACPI_OPERAND_OBJECT *ObjDesc;
267     ACPI_OPERAND_OBJECT *HandlerObj;
268     ACPI_OPERAND_OBJECT *PreviousHandlerObj;
269     ACPI_STATUS          Status;
270     UINT32               i;
271
272
273     ACPI_FUNCTION_TRACE (AcpiRemoveNotifyHandler);
274
275     /* Parameter validation */
276
277     if ((!Device) || (!Handler) || (!HandlerType) ||
278         (HandlerType > ACPI_MAX_NOTIFY_HANDLER_TYPE))
279     {
280         return_ACPI_STATUS (AE_BAD_PARAMETER);
281     }
282
283     /* Make sure all deferred notify tasks are completed */
284
285     AcpiOsWaitEventsComplete ();
286
287     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
288     if (ACPI_FAILURE (Status))
289     {
290         return_ACPI_STATUS (Status);
291     }
292
293     /* Root Object. Global handlers are removed here */
294
295     if (Device == ACPI_ROOT_OBJECT)
296     {
297         for (i = 0; i < ACPI_NUM_NOTIFY_TYPES; i++)
298         {
299             if (HandlerType & (i+1))
300             {
301                 if (!AcpiGbl_GlobalNotify[i].Handler ||
302                     (AcpiGbl_GlobalNotify[i].Handler != Handler))
303                 {
304                     Status = AE_NOT_EXIST;
305                     goto UnlockAndExit;
306                 }
307             }
308
309             ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
310                               "Removing global notify handler\n"));
311
312             AcpiGbl_GlobalNotify[i].Handler = NULL;
313             AcpiGbl_GlobalNotify[i].Context = NULL;
314         }
315     }
316
317     goto UnlockAndExit;
318 }
319
320 /* All other objects: Are Notifies allowed on this object? */
321
322 if (!AcpiEvIsNotifyObject (Node))

```

```

323 {
324     Status = AE_TYPE;
325     goto UnlockAndExit;
326 }
327
328 /* Must have an existing internal object */
329
330 ObjDesc = AcpiNsGetAttachedObject (Node);
331 if (!ObjDesc)
332 {
333     Status = AE_NOT_EXIST;
334     goto UnlockAndExit;
335 }
336
337 /* Internal object exists. Find the handler and remove it */
338
339 for (i = 0; i < ACPI_NUM_NOTIFY_TYPES; i++)
340 {
341     if (HandlerType & (i+1))
342     {
343         HandlerObj = ObjDesc->CommonNotify.NotifyList[i];
344         PreviousHandlerObj = NULL;
345
346         /* Attempt to find the handler in the handler list */
347
348         while (HandlerObj &&
349             (HandlerObj->Notify.Handler != Handler))
350         {
351             PreviousHandlerObj = HandlerObj;
352             HandlerObj = HandlerObj->Notify.Next[i];
353         }
354
355         if (!HandlerObj)
356         {
357             Status = AE_NOT_EXIST;
358             goto UnlockAndExit;
359         }
360
361         /* Remove the handler object from the list */
362
363         if (PreviousHandlerObj) /* Handler is not at the list head */
364         {
365             PreviousHandlerObj->Notify.Next[i] =
366                 HandlerObj->Notify.Next[i];
367         }
368         else /* Handler is at the list head */
369         {
370             ObjDesc->CommonNotify.NotifyList[i] =
371                 HandlerObj->Notify.Next[i];
372         }
373
374         AcpiUtRemoveReference (HandlerObj);
375     }
376 }
377
378 UnlockAndExit:
379 (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
380 return_ACPI_STATUS (Status);
381 }
382
383 ACPI_EXPORT_SYMBOL (AcpiRemoveNotifyHandler)
384
385
386
387 /*****
388 *

```

```

389 * FUNCTION:      AcpiInstallExceptionHandler
390 *
391 * PARAMETERS:    Handler          - Pointer to the handler function for the
392 *                  event
393 *
394 * RETURN:        Status
395 *
396 * DESCRIPTION:    Saves the pointer to the handler function
397 *
398 *****/
400 ACPI_STATUS
401 AcpiInstallExceptionHandler (
402     ACPI_EXCEPTION_HANDLER Handler)
403 {
404     ACPI_STATUS      Status;
405
406     ACPI_FUNCTION_TRACE (AcpiInstallExceptionHandler);
407
408     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
409     if (ACPI_FAILURE (Status))
410     {
411         return ACPI_STATUS (Status);
412     }
413
414     /* Don't allow two handlers. */
415
416     if (AcpiGbl_ExceptionHandler)
417     {
418         Status = AE_ALREADY_EXISTS;
419         goto Cleanup;
420     }
421
422     /* Install the handler */
423
424     AcpiGbl_ExceptionHandler = Handler;
425
426 Cleanup:
427     (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
428     return ACPI_STATUS (Status);
429 }
430
431
432
433 ACPI_EXPORT_SYMBOL (AcpiInstallExceptionHandler)
434
435 #if (!ACPI_REDUCED_HARDWARE)
436 /*****
437 *
438 * FUNCTION:      AcpiInstallSciHandler
439 *
440 * PARAMETERS:    Address          - Address of the handler
441 *                  Context        - Value passed to the handler on each SCI
442 *
443 * RETURN:        Status
444 *
445 * DESCRIPTION:    Install a handler for a System Control Interrupt.
446 *
447 *****/
448
449
450 ACPI_STATUS
451 AcpiInstallSciHandler (
452     ACPI_SCI_HANDLER      Address,
453     void                   *Context)
454 {

```

```

455     ACPI_SCI_HANDLER_INFO *NewSciHandler;
456     ACPI_SCI_HANDLER_INFO *SciHandler;
457     ACPI_CPU_FLAGS        Flags;
458     ACPI_STATUS            Status;
459
460     ACPI_FUNCTION_TRACE (AcpiInstallSciHandler);
461
462     if (!Address)
463     {
464         return ACPI_STATUS (AE_BAD_PARAMETER);
465     }
466
467     /* Allocate and init a handler object */
468
469     NewSciHandler = ACPI_ALLOCATE (sizeof (ACPI_SCI_HANDLER_INFO));
470     if (!NewSciHandler)
471     {
472         return ACPI_STATUS (AE_NO_MEMORY);
473     }
474
475     NewSciHandler->Address = Address;
476     NewSciHandler->Context = Context;
477
478     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
479     if (ACPI_FAILURE (Status))
480     {
481         goto Exit;
482     }
483
484     /* Lock list during installation */
485
486     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);
487     SciHandler = AcpiGbl_SciHandlerList;
488
489     /* Ensure handler does not already exist */
490
491     while (SciHandler)
492     {
493         if (Address == SciHandler->Address)
494         {
495             Status = AE_ALREADY_EXISTS;
496             goto UnlockAndExit;
497         }
498         SciHandler = SciHandler->Next;
499     }
500
501     /* Install the new handler into the global list (at head) */
502
503     NewSciHandler->Next = AcpiGbl_SciHandlerList;
504     AcpiGbl_SciHandlerList = NewSciHandler;
505
506 UnlockAndExit:
507
508     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
509     (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
510
511 Exit:
512     if (ACPI_FAILURE (Status))
513     {
514         ACPI_FREE (NewSciHandler);
515     }
516     return ACPI_STATUS (Status);

```

```

521 }

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_XRUP_T_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_XRUP_T_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_XRUP_T_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_XRUP_T_TYPE_MASK | ACPI_GPE_DISPATCH_MASK);
918     GpeEventInfo->Flags |= (UINT8) (Type | ACPI_GPE_DISPATCH_HANDLER);

920     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);

923 UnlockAndExit:
924     (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
925     return ACPI_STATUS (Status);

927 FreeAndExit:
928     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
929     ACPI_FREE (Handler);
930     goto UnlockAndExit;
931 }

933 ACPI_EXPORT_SYMBOL (AcpiInstallGpeHandler)

936 /*****
937  * FUNCTION:      AcpiRemoveGpeHandler
938  *
939  * PARAMETERS:    GpeDevice      - Namespace node for the GPE (NULL for FADT
940  *                                defined GPEs)
941  *                                GpeNumber    - The event to remove a handler
942  *                                Address      - Address of the handler
943  *
944  * RETURN:        Status
945  *
946  * DESCRIPTION:   Remove a handler for a General Purpose AcpiEvent.
947  *
948  *
949  *****/

951 ACPI_STATUS
952 AcpiRemoveGpeHandler (
953     ACPI_HANDLE             GpeDevice,
954     UINT32                  GpeNumber,
955     ACPI_GPE_HANDLER        Address)
956 {
957     ACPI_GPE_EVENT_INFO     *GpeEventInfo;
958     ACPI_GPE_HANDLER_INFO   *Handler;
959     ACPI_STATUS              Status;
960     ACPI_CPU_FLAGS          Flags;

963     ACPI_FUNCTION_TRACE (AcpiRemoveGpeHandler);

966     /* Parameter validation */

968     if (!Address)
969     {
970         return ACPI_STATUS (AE_BAD_PARAMETER);
971     }

973     /* Make sure all deferred GPE tasks are completed */

975     AcpiOsWaitEventsComplete ();

977     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
978     if (ACPI_FAILURE (Status))
979     {
980         return ACPI_STATUS (Status);
981     }

```

```

983     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);

985     /* Ensure that we have a valid GPE number */

987     GpeEventInfo = AcpiEvGetGpeEventInfo (GpeDevice, GpeNumber);
988     if (!GpeEventInfo)
989     {
990         Status = AE_BAD_PARAMETER;
991         goto UnlockAndExit;
992     }

994     /* Make sure that a handler is indeed installed */

996     if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) !=
997         ACPI_GPE_DISPATCH_HANDLER)
998     {
999         Status = AE_NOT_EXIST;
1000         goto UnlockAndExit;
1001     }

1003     /* Make sure that the installed handler is the same */

1005     if (GpeEventInfo->Dispatch.Handler->Address != Address)
1006     {
1007         Status = AE_BAD_PARAMETER;
1008         goto UnlockAndExit;
1009     }

1011     /* Remove the handler */

1013     Handler = GpeEventInfo->Dispatch.Handler;

1015     /* Restore Method node (if any), set dispatch flags */

1017     GpeEventInfo->Dispatch.MethodNode = Handler->MethodNode;
1018     GpeEventInfo->Flags &=
1019         ~(ACPI_GPE_XRUP_T_TYPE_MASK | ACPI_GPE_DISPATCH_MASK);
1020     GpeEventInfo->Flags |= Handler->OriginalFlags;

1022     /*
1023     * If the GPE was previously associated with a method and it was
1024     * enabled, it should be enabled at this point to restore the
1025     * post-initialization configuration.
1026     */
1027     if ((Handler->OriginalFlags & ACPI_GPE_DISPATCH_METHOD) &&
1028         Handler->OriginallyEnabled)
1029     {
1030         (void) AcpiEvAddGpeReference (GpeEventInfo);
1031     }

1033     /* Now we can free the handler object */

1035     ACPI_FREE (Handler);

1038 UnlockAndExit:
1039     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
1040     (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
1041     return ACPI_STATUS (Status);
1042 }

1044 ACPI_EXPORT_SYMBOL (AcpiRemoveGpeHandler)

1047 /*****
1048  *

```

```

1049 * FUNCTION:      AcpiAcquireGlobalLock
1050 *
1051 * PARAMETERS:    Timeout      - How long the caller is willing to wait
1052 *                Handle       - Where the handle to the lock is returned
1053 *                                (if acquired)
1054 *
1055 * RETURN:        Status
1056 *
1057 * DESCRIPTION:    Acquire the ACPI Global Lock
1058 *
1059 * Note: Allows callers with the same thread ID to acquire the global lock
1060 * multiple times. In other words, externally, the behavior of the global lock
1061 * is identical to an AML mutex. On the first acquire, a new handle is
1062 * returned. On any subsequent calls to acquire by the same thread, the same
1063 * handle is returned.
1064 *
1065 *****/
1067 ACPI_STATUS
1068 AcpiAcquireGlobalLock (
1069     UINT16      Timeout,
1070     UINT32      *Handle)
1071 {
1072     ACPI_STATUS      Status;
1073
1074     if (!Handle)
1075     {
1076         return (AE_BAD_PARAMETER);
1077     }
1078
1079     /* Must lock interpreter to prevent race conditions */
1080
1081     AcpiExEnterInterpreter ();
1082
1083     Status = AcpiExAcquireMutexObject (Timeout,
1084         AcpiGbl_GlobalLockMutex, AcpiOsGetThreadId ());
1085
1086     if (ACPI_SUCCESS (Status))
1087     {
1088         /* Return the global lock handle (updated in AcpiEvAcquireGlobalLock) */
1089
1090         *Handle = AcpiGbl_GlobalLockHandle;
1091     }
1092
1093     AcpiExExitInterpreter ();
1094     return (Status);
1095 }
1096
1098 ACPI_EXPORT_SYMBOL (AcpiAcquireGlobalLock)
1099
1101 /*****
1102 *
1103 * FUNCTION:      AcpiReleaseGlobalLock
1104 *
1105 * PARAMETERS:    Handle      - Returned from AcpiAcquireGlobalLock
1106 *
1107 * RETURN:        Status
1108 *
1109 * DESCRIPTION:    Release the ACPI Global Lock. The handle must be valid.
1110 *
1111 *****/
1113 ACPI_STATUS
1114 AcpiReleaseGlobalLock (

```

```

1115     UINT32      Handle)
1116 {
1117     ACPI_STATUS      Status;
1118
1119     if (!Handle || (Handle != AcpiGbl_GlobalLockHandle))
1120     {
1121         return (AE_NOT_ACQUIRED);
1122     }
1123
1124     Status = AcpiExReleaseMutexObject (AcpiGbl_GlobalLockMutex);
1125     return (Status);
1126 }
1127
1129 ACPI_EXPORT_SYMBOL (AcpiReleaseGlobalLock)
1131 #endif /* !ACPI_REduced_HARDWARE */

```



new/usr/src/common/acpica/components/events/evxfevnt.c

1

```
*****
10127 Fri Jan 17 07:30:06 2014
new/usr/src/common/acpica/components/events/evxfevnt.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: evxfevnt - External Interfaces, ACPI event disable/enable
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #define __EVXFEVNT_C__
46 #define EXPORT_ACPI_INTERFACES
47
48 #include "acpi.h"
49 #include "accommon.h"
50 #include "actables.h"
51
52 #define _COMPONENT ACPI_EVENTS
53 #define ACPI_MODULE_NAME ("evxfevnt")
54
55 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */
56
57 /*****
```

new/usr/src/common/acpica/components/events/evxfevnt.c

2

```
58 *
59 * FUNCTION: AcpiEnable
60 *
61 * PARAMETERS: None
62 *
63 * RETURN: Status
64 *
65 * DESCRIPTION: Transfers the system into ACPI mode.
66 *
67 *****/
68
69 ACPI_STATUS
70 AcpiEnable (
71     void)
72 {
73     ACPI_STATUS Status = AE_OK;
74
75     ACPI_FUNCTION_TRACE (AcpiEnable);
76
77     /* ACPI tables must be present */
78
79     if (!AcpiTbTablesLoaded ())
80     {
81         return_ACPI_STATUS (AE_NO_ACPI_TABLES);
82     }
83
84     /* If the Hardware Reduced flag is set, machine is always in acpi mode */
85
86     if (AcpiGbl_ReducedHardware)
87     {
88         return_ACPI_STATUS (AE_OK);
89     }
90
91     /* Check current mode */
92
93     if (AcpiHwGetMode() == ACPI_SYS_MODE ACPI)
94     {
95         ACPI_DEBUG_PRINT ((ACPI_DB_INIT, "System is already in ACPI mode\n"));
96     }
97     else
98     {
99         /* Transition to ACPI mode */
100
101         Status = AcpiHwSetMode (ACPI_SYS_MODE ACPI);
102         if (ACPI_FAILURE (Status))
103         {
104             ACPI_ERROR ((AE_INFO, "Could not transition to ACPI mode"));
105             return_ACPI_STATUS (Status);
106         }
107
108         ACPI_DEBUG_PRINT ((ACPI_DB_INIT,
109             "Transition to ACPI mode successful\n"));
110     }
111
112     return_ACPI_STATUS (Status);
113 }
114
115
116 ACPI_EXPORT_SYMBOL (AcpiEnable)
117
118
119 /*****
120 *
121 * FUNCTION: AcpiDisable
122 *
123 *
```

```
124 * 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 */
```

new/usr/src/common/acpica/components/events/evxfge.c

1

```
*****
26471 Fri Jan 17 07:30:06 2014
new/usr/src/common/acpica/components/events/evxfge.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 / *****
2 *
3 * Module Name: evxfge - External Interfaces for General Purpose Events (GPEs)
4 *
5 *****/
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #define __EVXFGPE_C__
46 #define EXPORT_ACPI_INTERFACES
47
48 #include "acpi.h"
49 #include "accommon.h"
50 #include "acevents.h"
51 #include "acnamesp.h"
52
53 #define _COMPONENT          ACPI_EVENTS
54         ACPI_MODULE_NAME    ("evxfge")
55
56
57 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */
```

new/usr/src/common/acpica/components/events/evxfge.c

2

```
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 *****/
79
80 ACPI_STATUS
81 AcpiUpdateAllGpes (
82     void)
83 {
84     ACPI_STATUS          Status;
85
86     /*
87      * ACPI_FUNCTION_TRACE (AcpiUpdateAllGpes);
88      * ACPI_FUNCTION_TRACE (AcpiUpdateGpes);
89      */
90
91     Status = AcpiUtAcquireMutex (ACPI_MTX_EVENTS);
92     if (ACPI_FAILURE (Status))
93     {
94         return ACPI_STATUS (Status);
95     }
96
97     if (AcpiGbl_AllGpesInitialized)
98     {
99         goto UnlockAndExit;
100     }
101
102     Status = AcpiEvWalkGpeList (AcpiEvInitializeGpeBlock, NULL);
103     if (ACPI_SUCCESS (Status))
104     {
105         AcpiGbl_AllGpesInitialized = TRUE;
106     }
107
108     UnlockAndExit:
109     (void) AcpiUtReleaseMutex (ACPI_MTX_EVENTS);
110     return ACPI_STATUS (Status);
111 }
112
113 /*
114  * unchanged portion omitted
115  */
116
117 201 ACPI_EXPORT_SYMBOL (AcpiDisableGpe)
118
119
120 /*****
121 *
122 * FUNCTION:      AcpiSetGpe
123 *
124 * PARAMETERS:    GpeDevice      - Parent GPE Device. NULL for GPE0/GPE1
125 *                GpeNumber      - GPE level within the GPE block
126 *                Action          - ACPI_GPE_ENABLE or ACPI_GPE_DISABLE
127 *****/
```

```

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 *****/
223
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;
233
234     ACPI_FUNCTION_TRACE (AcpiSetGpe);
235
236     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);
237
238     /* Ensure that we have a valid GPE number */
239
240     GpeEventInfo = AcpiEvGetGpeEventInfo (GpeDevice, GpeNumber);
241     if (!GpeEventInfo)
242     {
243         Status = AE_BAD_PARAMETER;
244         goto UnlockAndExit;
245     }
246
247     /* Perform the action */
248
249     switch (Action)
250     {
251     case ACPI_GPE_ENABLE:
252
253         Status = AcpiEvEnableGpe (GpeEventInfo);
254         break;
255
256     case ACPI_GPE_DISABLE:
257
258         Status = AcpiHwLowSetGpe (GpeEventInfo, ACPI_GPE_DISABLE);
259         break;
260
261     default:
262
263         Status = AE_BAD_PARAMETER;
264         break;
265     }
266
267     UnlockAndExit:
268     AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);
269     return ACPI_STATUS (Status);
270 }
271
272 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 *****/
295
296 ACPI_STATUS
297 AcpiSetupGpeForWake (
298     ACPI_HANDLE      WakeDevice,
299     ACPI_HANDLE      GpeDevice,
300     UINT32           GpeNumber)
301 {
302     ACPI_STATUS          Status;
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;
308
309     ACPI_FUNCTION_TRACE (AcpiSetupGpeForWake);
310
311     /* Parameter Validation */
312
313     if (!WakeDevice)
314     {
315         /*
316          * By forcing WakeDevice to be valid, we automatically enable the
317          * implicit notify feature on all hosts.
318          */
319         return ACPI_STATUS (AE_BAD_PARAMETER);
320     }
321
322     /* Handle root object case */
323
324     if (WakeDevice == ACPI_ROOT_OBJECT)
325     {
326         DeviceNode = AcpiGbl_RootNode;
327     }
328     else
329     {
330         DeviceNode = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, WakeDevice);
331     }
332
333     /* Validate WakeDevice is of type Device */
334
335     if (DeviceNode->Type != ACPI_TYPE_DEVICE)
336     {
337         return ACPI_STATUS (AE_BAD_PARAMETER);
338     }
339 }

```

```

342  /*
343  * Allocate a new notify object up front, in case it is needed.
344  * Memory allocation while holding a spinlock is a big no-no
345  * on some hosts.
346  */
347  NewNotify = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_GPE_NOTIFY_INFO));
348  if (!NewNotify)
349  {
350      return ACPI_STATUS (AE_NO_MEMORY);
351  }

353  Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);

355  /* Ensure that we have a valid GPE number */

357  GpeEventInfo = AcpiEvGetGpeEventInfo (GpeDevice, GpeNumber);
358  if (!GpeEventInfo)
359  {
360      Status = AE_BAD_PARAMETER;
361      goto UnlockAndExit;
362  }

364  /*
365  * If there is no method or handler for this GPE, then the
366  * WakeDevice will be notified whenever this GPE fires. This is
367  * known as an "implicit notify". Note: The GPE is assumed to be
368  * "WakeDevice will be notified whenever this GPE fires (aka
369  * "implicit notify") Note: The GPE is assumed to be
370  * level-triggered (for windows compatibility).
371  */
372  if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) ==
373      ACPI_GPE_DISPATCH_NONE)
374  {
375      /*
376      * This is the first device for implicit notify on this GPE.
377      * Just set the flags here, and enter the NOTIFY block below.
378      */
379      GpeEventInfo->Flags =
380          (ACPI_GPE_DISPATCH_NOTIFY | ACPI_GPE_LEVEL_TRIGGERED);
381      GpeEventInfo->Dispatch.DeviceNode = DeviceNode;
382  }

384  /*
385  * If we already have an implicit notify on this GPE, add
386  * this device to the notify list.
387  */
388  if ((GpeEventInfo->Flags & ACPI_GPE_DISPATCH_MASK) ==
389      ACPI_GPE_DISPATCH_NOTIFY)
390  {
391      /* Ensure that the device is not already in the list */
392      Notify = GpeEventInfo->Dispatch.NotifyList;
393      while (Notify)
394      {
395          if (Notify->DeviceNode == DeviceNode)
396          {
397              Status = AE_ALREADY_EXISTS;
398              goto UnlockAndExit;
399          }
400          Notify = Notify->Next;
401      }

402      /* Add this device to the notify list for this GPE */
403      NewNotify->DeviceNode = DeviceNode;

```

```

404      NewNotify->Next = GpeEventInfo->Dispatch.NotifyList;
405      GpeEventInfo->Dispatch.NotifyList = NewNotify;
406      NewNotify = NULL;
407  }

409  /* Mark the GPE as a possible wake event */

411  GpeEventInfo->Flags |= ACPI_GPE_CAN_WAKE;
412  Status = AE_OK;
413  }

415 UnlockAndExit:
416  AcpiOsReleaseLock (AcpiGbl_GpeLock, Flags);

418  /* Delete the notify object if it was not used above */

420  if (NewNotify)
421  {
422      ACPI_FREE (NewNotify);
423  }
424  return ACPI_STATUS (Status);
425  }

427 ACPI_EXPORT_SYMBOL (AcpiSetupGpeForWake)

430 /*****
431  *
432  * FUNCTION:      AcpiSetGpeWakeMask
433  *
434  * PARAMETERS:    GpeDevice          - Parent GPE Device. NULL for GPE0/GPE1
435  *                GpeNumber          - GPE level within the GPE block
436  *                Action              - Enable or Disable
437  *
438  * RETURN:        Status
439  *
440  * DESCRIPTION:   Set or clear the GPE's wakeup enable mask bit. The GPE must
441  *                already be marked as a WAKE GPE.
442  *
443  *****/

445 ACPI_STATUS
446 AcpiSetGpeWakeMask (
447     ACPI_HANDLE      GpeDevice,
448     UINT32            GpeNumber,
449     UINT8             Action)
450 {
451     ACPI_STATUS      Status = AE_OK;
452     ACPI_GPE_EVENT_INFO *GpeEventInfo;
453     ACPI_GPE_REGISTER_INFO *GpeRegisterInfo;
454     ACPI_CPU_FLAGS    Flags;
455     UINT32             RegisterBit;

458     ACPI_FUNCTION_TRACE (AcpiSetGpeWakeMask);

461     Flags = AcpiOsAcquireLock (AcpiGbl_GpeLock);

463     /*
464     * Ensure that we have a valid GPE number and that this GPE is in
465     * fact a wake GPE
466     */
467     GpeEventInfo = AcpiEvGetGpeEventInfo (GpeDevice, GpeNumber);
468     if (!GpeEventInfo)

```

```

469 {
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);
785     }

787     Node = AcpiNsValidateHandle (GpeDevice);
788     if (!Node)
789     {
790         Status = AE_BAD_PARAMETER;
791         goto UnlockAndExit;
792     }

794     /* Validate the parent device */

796     if (Node->Type != ACPI_TYPE_DEVICE)
797     {
798         Status = AE_TYPE;
799         goto UnlockAndExit;
800     }

802     if (Node->Object)
803     {
804         Status = AE_ALREADY_EXISTS;
805         goto UnlockAndExit;
806     }

808     /*
809     * For user-installed GPE Block Devices, the GpeBlockBaseNumber
810     * is always zero
811     */
812     Status = AcpiEvCreateGpeBlock (Node, GpeBlockAddress, RegisterCount,
813                                     0, InterruptNumber, &GpeBlock);
814     if (ACPI_FAILURE (Status))
815     {
816         goto UnlockAndExit;
817     }

819     /* Install block in the DeviceObject attached to the node */

821     ObjDesc = AcpiNsGetAttachedObject (Node);

```

```

822     if (!ObjDesc)
823     {
824         /*
825          * No object, create a new one (Device nodes do not always have
826          * an attached object)
827          */
828         ObjDesc = AcpiUtCreateInternalObject (ACPI_TYPE_DEVICE);
829         if (!ObjDesc)
830         {
831             Status = AE_NO_MEMORY;
832             goto UnlockAndExit;
833         }
835         Status = AcpiNsAttachObject (Node, ObjDesc, ACPI_TYPE_DEVICE);
837         /* Remove local reference to the object */
839         AcpiUtRemoveReference (ObjDesc);
840         if (ACPI_FAILURE (Status))
841         {
842             goto UnlockAndExit;
843         }
844     }
846     /* Now install the GPE block in the DeviceObject */
848     ObjDesc->Device.GpeBlock = GpeBlock;
851 UnlockAndExit:
852     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
853     return ACPI_STATUS (Status);
854 }
856 ACPI_EXPORT_SYMBOL (AcpiInstallGpeBlock)
859 /*****
860 *
861 * FUNCTION:      AcpiRemoveGpeBlock
862 *
863 * PARAMETERS:    GpeDevice          - Handle to the parent GPE Block Device
864 *
865 * RETURN:        Status
866 *
867 * DESCRIPTION:   Remove a previously installed block of GPE registers
868 *
869 *****/
871 ACPI_STATUS
872 AcpiRemoveGpeBlock (
873     ACPI_HANDLE          GpeDevice)
874 {
875     ACPI_OPERAND_OBJECT *ObjDesc;
876     ACPI_STATUS          Status;
877     ACPI_NAMESPACE_NODE *Node;
880     ACPI_FUNCTION_TRACE (AcpiRemoveGpeBlock);
883     if (!GpeDevice)
884     {
885         return ACPI_STATUS (AE_BAD_PARAMETER);
886     }

```

```

888     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
889     if (ACPI_FAILURE (Status))
890     {
891         return ACPI_STATUS (Status);
892     }
894     Node = AcpiNsValidateHandle (GpeDevice);
895     if (!Node)
896     {
897         Status = AE_BAD_PARAMETER;
898         goto UnlockAndExit;
899     }
901     /* Validate the parent device */
903     if (Node->Type != ACPI_TYPE_DEVICE)
904     {
905         Status = AE_TYPE;
906         goto UnlockAndExit;
907     }
909     /* Get the DeviceObject attached to the node */
911     ObjDesc = AcpiNsGetAttachedObject (Node);
912     if (!ObjDesc ||
913         !ObjDesc->Device.GpeBlock)
914     {
915         return ACPI_STATUS (AE_NULL_OBJECT);
916     }
918     /* Delete the GPE block (but not the DeviceObject) */
920     Status = AcpiEvDeleteGpeBlock (ObjDesc->Device.GpeBlock);
921     if (ACPI_SUCCESS (Status))
922     {
923         ObjDesc->Device.GpeBlock = NULL;
924     }
926 UnlockAndExit:
927     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
928     return ACPI_STATUS (Status);
929 }
931 unchanged_portion_omitted
933 ACPI_EXPORT_SYMBOL (AcpiGetGpeDevice)
935 #endif /* !ACPI_REDUCED_HARDWARE */

```

new/usr/src/common/acpica/components/events/evxfregn.c

1

```
*****
10049 Fri Jan 17 07:30:07 2014
new/usr/src/common/acpica/components/events/evxfregn.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: evxfregn - External Interfaces, ACPI Operation Regions and
4 * Address Spaces.
5 *
6 *****/
8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EVXFREGN_C__
46 #define EXPORT_ACPI_INTERFACES
48 #include "acpi.h"
49 #include "accommon.h"
50 #include "acnamesp.h"
51 #include "acevents.h"
53 #define _COMPONENT ACPI_EVENTS
54 ACPI_MODULE_NAME ("evxfregn")
57 /*****
```

new/usr/src/common/acpica/components/events/evxfregn.c

2

```
58 *
59 * FUNCTION: AcpiInstallAddressSpaceHandler
60 *
61 * PARAMETERS: Device - Handle for the device
62 * SpaceId - The address space ID
63 * Handler - Address of the handler
64 * Setup - Address of the setup function
65 * Context - Value passed to the handler on each access
66 *
67 * RETURN: Status
68 *
69 * DESCRIPTION: Install a handler for all OpRegions of a given SpaceId.
70 *
71 * NOTE: This function should only be called after AcpiEnableSubsystem has
72 * been called. This is because any _REG methods associated with the Space ID
73 * are executed here, and these methods can only be safely executed after
74 * the default handlers have been installed and the hardware has been
75 * initialized (via AcpiEnableSubsystem.)
76 *
77 *****/
79 ACPI_STATUS
80 AcpiInstallAddressSpaceHandler (
81     ACPI_HANDLE Device,
82     ACPI_ADR_SPACE_TYPE SpaceId,
83     ACPI_ADR_SPACE_HANDLER Handler,
84     ACPI_ADR_SPACE_SETUP Setup,
85     void *Context)
86 {
87     ACPI_NAMESPACE_NODE *Node;
88     ACPI_STATUS Status;
89
91     ACPI_FUNCTION_TRACE (AcpiInstallAddressSpaceHandler);
92
93     /* Parameter validation */
94
95     if (!Device)
96     {
97         return_ACPI_STATUS (AE_BAD_PARAMETER);
98     }
99
101     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
102     if (ACPI_FAILURE (Status))
103     {
104         return_ACPI_STATUS (Status);
105     }
106
107     /* Convert and validate the device handle */
108
109     Node = AcpiNsValidateHandle (Device);
110     if (!Node)
111     {
112         Status = AE_BAD_PARAMETER;
113         goto UnlockAndExit;
114     }
115
116     /* Install the handler for all Regions for this Space ID */
117
118     Status = AcpiEvInstallSpaceHandler (Node, SpaceId, Handler, Setup, Context);
119     if (ACPI_FAILURE (Status))
120     {
121         goto UnlockAndExit;
122     }
123 }
```



```
124  /*
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:
144
145      if (!AcpiGbl_RegMethodsExecuted)
146      {
147          /* We will defer execution of the _REG methods for this space */
148          goto UnlockAndExit;
149      }
150      break;
151
152  default:
153
154      break;
155  }
156
157  /* Run all _REG methods for this address space */
158
159  Status = AcpiEvExecuteRegMethods (Node, SpaceId);
160
161
162  UnlockAndExit:
163      (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
164      return ACPI_STATUS (Status);
165  }
166
167  unchanged portion omitted
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314  ACPI_EXPORT_SYMBOL (AcpiRemoveAddressSpaceHandler)
```

new/usr/src/common/acpica/components/executer/exconfig.c

1

```
*****
20453 Fri Jan 17 07:30:07 2014
new/usr/src/common/acpica/components/executer/exconfig.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: exconfig - Namespace reconfiguration (Load/Unload opcodes)
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __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 #define ACPI_MODULE_NAME    ("exconfig")
```

new/usr/src/common/acpica/components/executer/exconfig.c

2

```
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 |= APOBJ_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);
189 }

192 /* Find the ACPI table in the RSDT/XSDT */

194 Status = AcpiTbFindTable (
195     Operand[0]->String.Pointer,
196     Status = AcpiTbFindTable (Operand[0]->String.Pointer,
197     Operand[1]->String.Pointer,
198     Operand[2]->String.Pointer, &TableIndex);
199 if (ACPI_FAILURE (Status))
200 {
201     if (Status != AE_NOT_FOUND)
202     {
203         return_ACPI_STATUS (Status);
204     }

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))
244     if ((Operand[4]->String.Pointer[0] != '\\') &&
245         (Operand[4]->String.Pointer[0] != '^'))
246     {
247         /*
248         * Path is not absolute, so it will be relative to the node
249         * referenced by the RootPathString (or the NS root if omitted)
250         */
251         StartNode = ParentNode;
252     }

```

```

252      /* Find the node referenced by the ParameterPathString */
253
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  }
261
262  /* Load the table into the namespace */
263
264  Status = AcpiExAddTable (TableIndex, ParentNode, &DdbHandle);
265  if (ACPI_FAILURE (Status))
266  {
267      return_ACPI_STATUS (Status);
268  }
269
270  /* Parameter Data (optional) */
271
272  if (ParameterNode)
273  {
274      /* Store the parameter data into the optional parameter object */
275
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);
282
283          AcpiUtRemoveReference (DdbHandle);
284          return_ACPI_STATUS (Status);
285      }
286  }
287
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  }
294
295  /* Invoke table handler if present */
296
297  if (AcpiGbl_TableHandler)
298  {
299      (void) AcpiGbl_TableHandler (ACPI_TABLE_EVENT_LOAD, Table,
300                                  AcpiGbl_TableHandlerContext);
301  }
302
303  *ReturnDesc = DdbHandle;
304  return_ACPI_STATUS (Status);
305 }
306
307 /*****
308 *
309 * FUNCTION:      AcpiExRegionRead
310 *
311 * PARAMETERS:    ObjDesc      - Region descriptor
312 *                Length       - Number of bytes to read
313 *                Buffer        - Pointer to where to put the data
314 *
315 * RETURN:        Status
316 */
317

```

```

318 * DESCRIPTION:  Read data from an operation region. The read starts from the
319 *              beginning of the region.
320 *
321 *****/
322
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;
333
334     /* Bytewise reads */
335
336     for (i = 0; i < Length; i++)
337     {
338         Status = AcpiEvAddressSpaceDispatch (ObjDesc, NULL, ACPI_READ,
339                                               Status = AcpiEvAddressSpaceDispatch (ObjDesc, ACPI_READ,
340                                               RegionOffset, 8, &Value);
341         if (ACPI_FAILURE (Status))
342         {
343             return (Status);
344         }
345
346         *Buffer = (UINT8) Value;
347         Buffer++;
348         RegionOffset++;
349     }
350
351     return (AE_OK);
352 }
353
354 /*****
355 *
356 * FUNCTION:      AcpiExLoadOp
357 *
358 * PARAMETERS:    ObjDesc      - Region or Buffer/Field where the table will be
359 *                Target        - Where a handle to the table will be stored
360 *                WalkState     - Current state
361 *
362 * RETURN:        Status
363 *
364 * DESCRIPTION:   Load an ACPI table from a field or operation region
365 *
366 * NOTE: Region Fields (Field, BankField, IndexFields) are resolved to buffer
367 *        objects before this code is reached.
368 *
369 *        If source is an operation region, it must refer to SystemMemory, as
370 *        per the ACPI specification.
371 *
372 *****/
373
374
375
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

```

```

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:
529         return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
530     }

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 & APOBJ_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 }
```

new/usr/src/common/acpica/components/executer/exconvrt.c

1

```
*****
21068 Fri Jan 17 07:30:08 2014
new/usr/src/common/acpica/components/executer/exconvrt.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: exconvrt - Object conversion routines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXCONVRT_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acinterp.h"
50 #include "amlcode.h"

53 #define _COMPONENT          ACPI_EXECUTER
54         ACPI_MODULE_NAME    ("exconvrt")

56 /* Local prototypes */

58 static UINT32
```

new/usr/src/common/acpica/components/executer/exconvrt.c

2

```
59 AcpiExConvertToAscii (
60     UINT64                      Integer,
61     UINT16                      Base,
62     UINT8                       *String,
63     UINT8                       MaxLength);

66 /*****
67 *
68 * FUNCTION:      AcpiExConvertToInteger
69 *
70 * PARAMETERS:    ObjDesc        - Object to be converted. Must be an
71 *                                     Integer, Buffer, or String
72 *                                     ResultDesc - Where the new Integer object is returned
73 *                                     Flags      - Used for string conversion
74 *
75 * RETURN:        Status
76 *
77 * DESCRIPTION:    Convert an ACPI Object to an integer.
78 *
79 *****/

81 ACPI_STATUS
82 AcpiExConvertToInteger (
83     ACPI_OPERAND_OBJECT *ObjDesc,
84     **ResultDesc,
85     UINT32              Flags)
86 {
87     ACPI_OPERAND_OBJECT *ReturnDesc;
88     UINT8                *Pointer;
89     UINT64               Result;
90     UINT32               i;
91     UINT32               Count;
92     ACPI_STATUS           Status;

95     ACPI_FUNCTION_TRACE_PTR (ExConvertToInteger, ObjDesc);

98     switch (ObjDesc->Common.Type)
99     {
100     case ACPI_TYPE_INTEGER:

102         /* No conversion necessary */

104         *ResultDesc = ObjDesc;
105         return ACPI_STATUS (AE_OK);

107     case ACPI_TYPE_BUFFER:
108     case ACPI_TYPE_STRING:

110         /* Note: Takes advantage of common buffer/string fields */

112         Pointer = ObjDesc->Buffer.Pointer;
113         Count   = ObjDesc->Buffer.Length;
114         break;

116     default:

118         return ACPI_STATUS (AE_TYPE);
119     }

121     /*
122     * Convert the buffer/string to an integer. Note that both buffers and
123     * strings are treated as raw data - we don't convert ascii to hex for
124     * strings.
125     */
```



```

125      *
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);
203      *ResultDesc = ReturnDesc;
204      return ACPI_STATUS (AE_OK);
205 }

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;

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;

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;

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;

558     default:
559         return_ACPI_STATUS (AE_BAD_PARAMETER);
560 }

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 }

572 ReturnDesc = AcpiUtCreateStringObject ((ACPI_SIZE) StringLength);
573 if (!ReturnDesc)
574 {
575     return_ACPI_STATUS (AE_NO_MEMORY);
576 }

```

```

578     NewBuf = ReturnDesc->Buffer.Pointer;

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     }

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;

603     default:

605         return_ACPI_STATUS (AE_TYPE);
606     }

608     *ResultDesc = ReturnDesc;
609     return_ACPI_STATUS (AE_OK);
610 }

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 *****/

628 ACPI_STATUS
629 AcpiExConvertToTargetType (
630     ACPI_OBJECT_TYPE      DestinationType,
631     *SourceDesc,
632     ACPI_OPERAND_OBJECT **ResultDesc,
633     ACPI_WALK_STATE *WalkState)
634 {
635     ACPI_STATUS      Status = AE_OK;

638     ACPI_FUNCTION_TRACE (ExConvertToTargetType);

641     /* Default behavior */

```

```

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)\n",
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:
717         ACPI_ERROR ((AE_INFO, "Bad destination type during conversion: 0x%X"
718         DestinationType));
719         Status = AE_AML_INTERNAL;
720         break;
721     }
722     break;

724     case ARGI_REFERENCE:
725         /*
726         * CreateXxxxField cases - we are storing the field object into the name
727         */
728         break;

729     default:

732     default:
733         ACPI_ERROR ((AE_INFO,
734         "Unknown Target type ID 0x%X AmlOpcode 0x%X DestType %s",
735         GET_CURRENT_ARG_TYPE (WalkState->OpInfo->RuntimeArgs),
736         WalkState->Opcode, AcpiUtGetTypeName (DestinationType)));
737         Status = AE_AML_INTERNAL;
738     }

739     /*
740     * Source-to-Target conversion semantics:
741     * If conversion to the target type cannot be performed, then simply
742     * overwrite the target with the new object and type.
743     */
744     if (Status == AE_TYPE)
745     {
746         Status = AE_OK;
747     }

748     return ACPI_STATUS (Status);
749 }

```

new/usr/src/common/acpica/components/executer/excreate.c

1

```
*****
16473 Fri Jan 17 07:30:08 2014
new/usr/src/common/acpica/components/executer/excreate.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: excreate - Named object creation
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __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 /*****
```

new/usr/src/common/acpica/components/executer/excreate.c

2

```
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 */
144
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 }
160
161 _____ 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
285                  RegionSpace       - SpaceID for the region
286                  WalkState         - Current state
287  *
288  * RETURN:        Status
289  *
290  * DESCRIPTION:   Create a new operation region object
291  *****/

```

```

293 ACPI_STATUS
294 AcpiExCreateRegion (
295     UINT8          *AmlStart,
296     UINT32         AmlLength,
297     UINT8          SpaceId,
298     UINT8          RegionSpace,
299     ACPI_WALK_STATE *WalkState)
300 {
301     ACPI_STATUS Status;

```

```

301     ACPI_OPERAND_OBJECT *ObjDesc;
302     ACPI_NAMESPACE_NODE *Node;
303     ACPI_OPERAND_OBJECT *RegionObj2;
304
305     ACPI_FUNCTION_TRACE (ExCreateRegion);
306
307     /* Get the Namespace Node */
308
309     Node = WalkState->Op->Common.Node;
310
311     /*
312      * If the region object is already attached to this node,
313      * just return
314      */
315     if (AcpiNsGetAttachedObject (Node))
316     {
317         return ACPI_STATUS (AE_OK);
318     }
319
320     /*
321      * Space ID must be one of the predefined IDs, or in the user-defined
322      * range
323      */
324     if (!AcpiIsValidSpaceId (SpaceId))
325     {
326         if ((RegionSpace >= ACPI_NUM_PREDEFINED_REGIONS) &&
327             (RegionSpace < ACPI_USER_REGION_BEGIN) &&
328             (RegionSpace != ACPI_ADR_SPACE_DATA_TABLE))
329         {
330             /*
331              * Print an error message, but continue. We don't want to abort
332              * a table load for this exception. Instead, if the region is
333              * actually used at runtime, abort the executing method.
334              */
335             ACPI_ERROR ((AE_INFO, "Invalid/unknown Address Space ID: 0x%2.2X", SpaceId));
336             ACPI_ERROR ((AE_INFO, "Invalid AddressSpace type 0x%X", RegionSpace));
337             return ACPI_STATUS (AE_AML_INVALID_SPACE_ID);
338         }
339     }
340
341     ACPI_DEBUG_PRINT ((ACPI_DB_LOAD, "Region Type - %s (0x%X)\n",
342                      AcpiUtGetRegionName (SpaceId), SpaceId));
343     AcpiUtGetRegionName (RegionSpace, RegionSpace));
344
345     /* Create the region descriptor */
346
347     ObjDesc = AcpiUtCreateInternalObject (ACPI_TYPE_REGION);
348     if (!ObjDesc)
349     {
350         Status = AE_NO_MEMORY;
351         goto Cleanup;
352     }
353
354     /*
355      * Remember location in AML stream of address & length
356      * operands since they need to be evaluated at run time.
357      */
358     RegionObj2 = ObjDesc->Common.NextObject;
359     RegionObj2->Extra.AmlStart = AmlStart;
360     RegionObj2->Extra.AmlLength = AmlLength;
361     if (WalkState->ScopeInfo)
362     {
363         RegionObj2->Extra.ScopeNode = WalkState->ScopeInfo->Scope.Node;
364     }
365     else
366     {

```

```
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_
```



new/usr/src/common/acpica/components/executer/exdebug.c

1

```
*****
8374 Fri Jan 17 07:30:08 2014
new/usr/src/common/acpica/components/executer/exdebug.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: exdebug - Support for stores to the AML Debug Object
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXDEBUG_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acinterp.h"

51 #define _COMPONENT          ACPI_EXECUTER
52 #define ACPI_MODULE_NAME    ("exdebug")

55 #ifndef ACPI_NO_ERROR_MESSAGES
56 /*****
57 *
58 * FUNCTION:      AcpiExDoDebugObject
```

new/usr/src/common/acpica/components/executer/exdebug.c

2

```
59 *
60 * PARAMETERS:    SourceDesc      - Object to be output to "Debug Object"
61 *               Level           - Indentation level (used for packages)
62 *               Index           - Current package element, zero if not pkg
63 *
64 * RETURN:        None
65 *
66 * DESCRIPTION:   Handles stores to the AML Debug Object. For example:
67 *               Store(INT1, Debug)
68 *
69 * This function is not compiled if ACPI_NO_ERROR_MESSAGES is set.
70 *
71 * This function is only enabled if AcpiGbl_EnableAmlDebugObject is set, or
72 * if ACPI_LV_DEBUG_OBJECT is set in the AcpiDbgLevel. Thus, in the normal
73 * operational case, stores to the debug object are ignored but can be easily
74 * enabled if necessary.
75 *
76 *****/

78 void
79 AcpiExDoDebugObject (
80     ACPI_OPERAND_OBJECT *SourceDesc,
81     UINT32 Level,
82     UINT32 Index)
83 {
84     UINT32 i;

87     ACPI_FUNCTION_TRACE_PTR (ExDoDebugObject, SourceDesc);

90     /* Output must be enabled via the DebugObject global or the DbgLevel */

92     if (!AcpiGbl_EnableAmlDebugObject &&
93         !(AcpiDbgLevel & ACPI_LV_DEBUG_OBJECT))
94     {
95         return_VOID;
96     }

98     /*
99     * Print line header as long as we are not in the middle of an
100     * object display
101     */
102     if (((Level > 0) && Index == 0))
103     {
104         AcpiOsPrintf ("[ACPI Debug] %s", Level, " ");
105     }

107     /* Display the index for package output only */

109     if (Index > 0)
110     {
111         AcpiOsPrintf ("(%.2u ", Index-1);
112     }

114     if (!SourceDesc)
115     {
116         AcpiOsPrintf ("[Null Object]\n");
117         return_VOID;
118     }

120     if (ACPI_GET_DESCRIPTOR_TYPE (SourceDesc) == ACPI_DESC_TYPE_OPERAND)
121     {
122         AcpiOsPrintf ("%s ", AcpiUtGetObjectTypeName (SourceDesc));

124         if (!AcpiUtValidInternalObject (SourceDesc))
```

```

125     {
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 }
141
142 /* SourceDesc is of type ACPI_DESC_TYPE_OPERAND */
143
144 switch (SourceDesc->Common.Type)
145 {
146 case ACPI_TYPE_INTEGER:
147
148     /* Output correct integer width */
149
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;
161
162 case ACPI_TYPE_BUFFER:
163
164     AcpiOsPrintf ("0x%.2X\n", (UINT32) SourceDesc->Buffer.Length);
165     AcpiUtDumpBuffer (SourceDesc->Buffer.Pointer,
166         AcpiUtDumpBuffer2 (SourceDesc->Buffer.Pointer,
167             (SourceDesc->Buffer.Length < 256) ?
168                 SourceDesc->Buffer.Length : 256, DB_BYTE_DISPLAY, 0);
169     break;
170
171 case ACPI_TYPE_STRING:
172
173     AcpiOsPrintf ("0x%.2X] \"%s\"\n",
174         SourceDesc->String.Length, SourceDesc->String.Pointer);
175     break;
176
177 case ACPI_TYPE_PACKAGE:
178
179     AcpiOsPrintf ("Contains 0x%.2X Elements]\n",
180         SourceDesc->Package.Count);
181
182     /* Output the entire contents of the package */
183
184     for (i = 0; i < SourceDesc->Package.Count; i++)
185     {
186         AcpiExDoDebugObject (SourceDesc->Package.Elements[i],
187             Level+4, i+1);
188     }
189     break;

```

```

190 case ACPI_TYPE_LOCAL_REFERENCE:
191
192     AcpiOsPrintf ("[%s] ", AcpiUtGetReferenceName (SourceDesc));
193
194     /* Decode the reference */
195
196     switch (SourceDesc->Reference.Class)
197     {
198     case ACPI_REFCLASS_INDEX:
199
200         AcpiOsPrintf ("0x%X\n", SourceDesc->Reference.Value);
201         break;
202
203     case ACPI_REFCLASS_TABLE:
204
205         /* Case for DdbHandle */
206
207         AcpiOsPrintf ("Table Index 0x%X\n", SourceDesc->Reference.Value);
208         return_VOID;
209         return;
210
211     default:
212         break;
213     }
214
215     AcpiOsPrintf (" ");
216
217     /* Check for valid node first, then valid object */
218
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));
231
232             switch ((SourceDesc->Reference.Node)->Type)
233             {
234             /* These types have no attached object */
235
236             case ACPI_TYPE_DEVICE:
237                 AcpiOsPrintf ("Device\n");
238                 break;
239
240             case ACPI_TYPE_THERMAL:
241                 AcpiOsPrintf ("Thermal Zone\n");
242                 break;
243
244             default:
245
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
```

new/usr/src/common/acpica/components/executer/exdump.c

1

```
*****
34334 Fri Jan 17 07:30:09 2014
new/usr/src/common/acpica/components/executer/exdump.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: exdump - Interpreter debug output routines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXDUMP_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 #define ACPI_MODULE_NAME    ("exdump")

56 /*
57 * The following routines are used for debug output only
58 */
```

new/usr/src/common/acpica/components/executer/exdump.c

2

```
59 #if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)

61 /* Local prototypes */

63 static void
64 AcpiExOutString (
65     char                *Title,
66     char                *Value);

68 static void
69 AcpiExOutPointer (
70     char                *Title,
71     void                *Value);

73 static void
74 AcpiExDumpObject (
75     ACPI_OPERAND_OBJECT *ObjDesc,
76     ACPI_EXDUMP_INFO    *Info);

78 static void
79 AcpiExDumpReferenceObj (
80     ACPI_OPERAND_OBJECT *ObjDesc);

82 static void
83 AcpiExDumpPackageObj (
84     ACPI_OPERAND_OBJECT *ObjDesc,
85     UINT32               Level,
86     UINT32               Index);

89 /*****
90 *
91 * Object Descriptor info tables
92 *
93 * Note: The first table entry must be an INIT opcode and must contain
94 * the table length (number of table entries)
95 *
96 *****/

98 static ACPI_EXDUMP_INFO    AcpiExDumpInteger[2] =
99 {
100     {ACPI_EXD_INIT,      ACPI_EXD_TABLE_SIZE (AcpiExDumpInteger),    NULL},
101     {ACPI_EXD_UINT64,    ACPI_EXD_OFFSET (Integer.Value),            "Value"}
102 };
    unchanged_portion_omitted

130 static ACPI_EXDUMP_INFO    AcpiExDumpDevice[4] =
131 {
132     {ACPI_EXD_INIT,      ACPI_EXD_TABLE_SIZE (AcpiExDumpDevice),    NULL},
133     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (Device.Handler),          "Handler"},
134     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (Device.NotifyList[0]),    "System"},
135     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (Device.NotifyList[1]),    "Device"},
136     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (Device.SystemNotify),     "System"},
137     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (Device.DeviceNotify),     "Device"}
138 };
    unchanged_portion_omitted

177 static ACPI_EXDUMP_INFO    AcpiExDumpPower[5] =
178 {
179     {ACPI_EXD_INIT,      ACPI_EXD_TABLE_SIZE (AcpiExDumpPower),    NULL},
180     {ACPI_EXD_UINT32,     ACPI_EXD_OFFSET (PowerResource.SystemLevel), "System"},
181     {ACPI_EXD_UINT32,     ACPI_EXD_OFFSET (PowerResource.ResourceOrder), "Resource"},
182     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (PowerResource.NotifyList[0]), "System"},
183     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (PowerResource.NotifyList[1]), "Device"},
184     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (PowerResource.SystemNotify), "System"},
185     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (PowerResource.DeviceNotify), "Device"}
186 }
```

```

184 };

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"},
194     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (Processor.SystemNotify),        "System"},
195     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (Processor.DeviceNotify),        "Device"},
196     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (Processor.Handler),            "Handler"},
197 };

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"},
202     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (ThermalZone.SystemNotify),        "System"},
203     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (ThermalZone.DeviceNotify),        "Device"},
204     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (ThermalZone.Handler),            "Handler"},
205 };
    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"},
219     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (Field.RegionObj),              "Region"},
220 };
    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"},
270     {ACPI_EXD_POINTER,    ACPI_EXD_OFFSET (Notify.Context),                "Context"},
271 };
    unchanged_portion_omitted

339 /*****
340 *
341 * FUNCTION:      AcpiExDumpObject
342 *
343 * PARAMETERS:    ObjDesc      - Descriptor to dump
344 *                Info         - Info table corresponding to this object
345 *
346 * RETURN:        None
347 *
348 * DESCRIPTION:   Walk the info table for this object
349 *
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));
447         AcpiExOutString ("Class Name",
448             ACPI_CAST_PTR (char, AcpiUtGetReferenceName (ObjDesc)));
449         AcpiExDumpReferenceObj (ObjDesc);
450         break;

452     default:

453         AcpiOsPrintf ("**** Invalid table opcode [%X] ****\n",
454             Info->Opcode);
455         return;

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))
491     if (!((ACPI_LV_EXEC & AcpiDbgLevel) && (_COMPONENT & AcpiDbgLayer)))
492     {
493         return;
494     }

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:
547
548         AcpiOsPrintf ("%p\n", ObjDesc->Reference.Object);
549         break;
550
551     case ACPI_REFCLASS_TABLE:
552
553         AcpiOsPrintf ("Table Index %X\n", ObjDesc->Reference.Value);
554         break;
555
556     case ACPI_REFCLASS_REFOF:
557
558         AcpiOsPrintf ("%p [%s]\n", ObjDesc->Reference.Object,
559             AcpiUtGetTypeName (((ACPI_OPERAND_OBJECT *)
560                 ObjDesc->Reference.Object)->Common.Type));
561         break;
562
563     case ACPI_REFCLASS_NAME:
564
565         AcpiOsPrintf ("- [%4.4s]\n", ObjDesc->Reference.Node->Name.Ascii);
566         break;
567
568     case ACPI_REFCLASS_ARG:
569     case ACPI_REFCLASS_LOCAL:
570
571         AcpiOsPrintf ("%X\n", ObjDesc->Reference.Value);
572         break;
573
574     default:      /* Unknown reference class */
575
576         AcpiOsPrintf ("%2.2X\n", ObjDesc->Reference.Class);
577         break;
578     }
579     break;
580
581     case ACPI_TYPE_BUFFER:
582
583         AcpiOsPrintf ("Buffer length %.2X @ %p\n",
584             ObjDesc->Buffer.Length, ObjDesc->Buffer.Pointer);
585
586         /* Debug only -- dump the buffer contents */
587
588         if (ObjDesc->Buffer.Pointer)
589         {
590             Length = ObjDesc->Buffer.Length;
591             if (Length > 128)
592             {
593                 Length = 128;
594             }
595
596             AcpiOsPrintf ("Buffer Contents: (displaying length 0x%.2X)\n",
597                 Length);
598             ACPI_DUMP_BUFFER (ObjDesc->Buffer.Pointer, Length);
599         }
600         break;
601
602     case ACPI_TYPE_INTEGER:
603
604         AcpiOsPrintf ("Integer %8.8X%8.8X\n",

```

```

605         ACPI_FORMAT_UINT64 (ObjDesc->Integer.Value));
606         break;
607
608     case ACPI_TYPE_PACKAGE:
609
610         AcpiOsPrintf ("Package [Len %X] ElementArray %p\n",
611             ObjDesc->Package.Count, ObjDesc->Package.Elements);
612
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;
627
628     case ACPI_TYPE_REGION:
629
630         AcpiOsPrintf ("Region %s (%X)",
631             AcpiUtGetRegionName (ObjDesc->Region.SpaceId),
632             ObjDesc->Region.SpaceId);
633
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;
649
650     case ACPI_TYPE_STRING:
651
652         AcpiOsPrintf ("String length %X @ %p ",
653             ObjDesc->String.Length,
654             ObjDesc->String.Pointer);
655
656         AcpiUtPrintString (ObjDesc->String.Pointer, ACPI_UINT8_MAX);
657         AcpiOsPrintf ("\n");
658         break;
659
660     case ACPI_TYPE_LOCAL_BANK_FIELD:
661
662         AcpiOsPrintf ("BankField\n");
663         break;
664
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))
861             if (!((ACPI_LV_OBJECTS & AcpiDbgLevel) && (_COMPONENT & AcpiDbgLayer)))
862             {
863                 return;
864             }
865     }

```



```

866     AcpiOsPrintf ("%20s : %4.4s\n", "Name", AcpiUtGetNodeName (Node));
867     AcpiExOutString ("Type", AcpiUtGetTypeNames (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     }
1006     AcpiOsPrintf ("\n");
1007     break;

1095     case ACPI_TYPE_BUFFER:

1097         AcpiOsPrintf ("[Buffer] Length %2X = ", ObjDesc->Buffer.Length);
1098         if (ObjDesc->Buffer.Length)
1099         {
1100             AcpiUtDebugDumpBuffer (ACPI_CAST_PTR (UINT8, ObjDesc->Buffer.Pointer),
1101                                   AcpiUtDumpBuffer (ACPI_CAST_PTR (UINT8, ObjDesc->Buffer.Pointer),
1102                                                       ObjDesc->Buffer.Length, DB_DWORD_DISPLAY, _COMPONENT));
1103         }
1104         else
1105         {
1106             AcpiOsPrintf ("\n");
1107         }
1108         break;

1209     case ACPI_TYPE_PACKAGE:

1211         AcpiOsPrintf ("[Package] Contains %u Elements:\n",
1212                       ObjDesc->Package.Count);

1214         for (i = 0; i < ObjDesc->Package.Count; i++)
1215         {
1216             AcpiExDumpPackageObj (ObjDesc->Package.Elements[i], Level+1, i);
1217         }
1218         break;

1220     case ACPI_TYPE_LOCAL_REFERENCE:

1222         AcpiOsPrintf ("[Object Reference] Type [%s] %2.2X",
1223                       AcpiUtGetReferenceName (ObjDesc),
1224                       ObjDesc->Reference.Class);
1225         AcpiExDumpReferenceObj (ObjDesc);
1226         break;

1228     default:

1230         AcpiOsPrintf ("[Unknown Type] %X\n", ObjDesc->Common.Type);
1231         break;
1232     }
1233 }

1236 /*****
1237  *
1238  * FUNCTION:      AcpiExDumpObjectDescriptor
1239  *
1240  * PARAMETERS:    ObjDesc          - Descriptor to dump
1241  *                Flags            - Force display if TRUE
1242  *
1243  * DESCRIPTION:   Dumps the members of the object descriptor given.
1244  *
1245  *****/

```

```
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
```

new/usr/src/common/acpica/components/executer/exfield.c

1

```
*****
13428 Fri Jan 17 07:30:09 2014
new/usr/src/common/acpica/components/executer/exfield.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: exfield - ACPI AML (p-code) execution - field manipulation
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXFIELD_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acdispat.h"
50 #include "acinterp.h"

53 #define _COMPONENT          ACPI_EXECUTER
54         ACPI_MODULE_NAME    ("exfield")

57 /*****
58 *
```

new/usr/src/common/acpica/components/executer/exfield.c

2

```
59 * FUNCTION:      AcpiExReadDataFromField
60 *
61 * PARAMETERS:    WalkState      - Current execution state
62 *                ObjDesc        - The named field
63 *                RetBufferDesc   - Where the return data object is stored
64 *
65 * RETURN:        Status
66 *
67 * DESCRIPTION:    Read from a named field. Returns either an Integer or a
68 *                Buffer, depending on the size of the field.
69 *
70 *****/

72 ACPI_STATUS
73 AcpiExReadDataFromField (
74     ACPI_WALK_STATE      *WalkState,
75     ACPI_OPERAND_OBJECT  *ObjDesc,
76     ACPI_OPERAND_OBJECT  **RetBufferDesc)
77 {
78     ACPI_STATUS      Status;
79     ACPI_OPERAND_OBJECT *BufferDesc;
80     ACPI_SIZE         Length;
81     void              *Buffer;
82     UINT32             Function;

85     ACPI_FUNCTION_TRACE_PTR (ExReadDataFromField, ObjDesc);

88     /* Parameter validation */

90     if (!ObjDesc)
91     {
92         return_ACPI_STATUS (AE_AML_NO_OPERAND);
93     }
94     if (!RetBufferDesc)
95     {
96         return_ACPI_STATUS (AE_BAD_PARAMETER);
97     }

99     if (ObjDesc->Common.Type == ACPI_TYPE_BUFFER_FIELD)
100     {
101         /*
102          * If the BufferField arguments have not been previously evaluated,
103          * evaluate them now and save the results.
104          */
105         if (!(ObjDesc->Common.Flags & AOPOBJ_DATA_VALID))
106         {
107             Status = AcpiDsGetBufferFieldArguments (ObjDesc);
108             if (ACPI_FAILURE (Status))
109             {
110                 return_ACPI_STATUS (Status);
111             }
112         }
113     }
114     else if ((ObjDesc->Common.Type == ACPI_TYPE_LOCAL_REGION_FIELD) &&
115             (ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_SMBUS |
116              ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_GSBUS |
117              ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_IPMI))
118     {
119         /*
120          * This is an SMBus, GSBUS or IPMI read. We must create a buffer to hold
121          * the data and then directly access the region handler.
122          */
123         * Note: SMBus and GSBUS protocol value is passed in upper 16-bits of Fu
```

```

122  * Note: Smbus protocol value is passed in upper 16-bits of Function
123  */
124  if (ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_SMBUS)
125  {
126      Length = ACPI_SMBUS_BUFFER_SIZE;
127      Function = ACPI_READ | (ObjDesc->Field.Attribute << 16);
128  }
129  else if (ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_GSBUS)
130  {
131      Length = ACPI_GSBUS_BUFFER_SIZE;
132      Function = ACPI_READ | (ObjDesc->Field.Attribute << 16);
133  }
134  else /* IPMI */
135  {
136      Length = ACPI_IPMI_BUFFER_SIZE;
137      Function = ACPI_READ;
138  }
139
140  BufferDesc = AcpiUtCreateBufferObject (Length);
141  if (!BufferDesc)
142  {
143      return_ACPI_STATUS (AE_NO_MEMORY);
144  }
145
146  /* Lock entire transaction if requested */
147
148  AcpiExAcquireGlobalLock (ObjDesc->CommonField.FieldFlags);
149
150  /* Call the region handler for the read */
151
152  Status = AcpiExAccessRegion (ObjDesc, 0,
153      ACPI_CAST_PTR (UINT64, BufferDesc->Buffer.Pointer),
154      Function);
155  AcpiExReleaseGlobalLock (ObjDesc->CommonField.FieldFlags);
156  goto Exit;
157
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  Exit:
215  if (ACPI_FAILURE (Status))
216  {
217      AcpiUtRemoveReference (BufferDesc);
218  }
219  else
220  {
221      *RetBufferDesc = BufferDesc;
222  }
223
224  return_ACPI_STATUS (Status);
225  }
226
227
228  /*****
229  * FUNCTION:      AcpiExWriteDataToField
230  *
231  * PARAMETERS:    SourceDesc      - Contains data to write
232  *                ObjDesc         - The named field
233  *                ResultDesc      - Where the return value is returned, if any
234  *
235  * RETURN:        Status
236  *
237  * DESCRIPTION:   Write to a named field
238  *****/
239
240  ACPI_STATUS
241  AcpiExWriteDataToField (
242      ACPI_OPERAND_OBJECT *SourceDesc,
243      ACPI_OPERAND_OBJECT *ObjDesc,
244      ACPI_OPERAND_OBJECT **ResultDesc)
245  {
246      ACPI_STATUS Status;
247      UINT32 Length;
248      void *Buffer;
249      ACPI_OPERAND_OBJECT *BufferDesc;
250      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 & APOBJ_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
288          * This is an SMBus or IPMI write. We will bypass the entire field
289          * mechanism and handoff the buffer directly to the handler. For
290          * these address spaces, the buffer is bi-directional; on a write,
291          * return data is returned in the same buffer.
292          *
293          * Source must be a buffer of sufficient size:
294          * ACPI_SMBUS_BUFFER_SIZE, ACPI_GSBUS_BUFFER_SIZE, or ACPI_IPMI_BUFFER_S
295          * ACPI_SMBUS_BUFFER_SIZE or ACPI_IPMI_BUFFER_SIZE.
296          *
297          * Note: SMBus and GSBUS protocol type is passed in upper 16-bits of Fun
298          * Note: SMBus protocol type is passed in upper 16-bits of Function
299          */
300         if (SourceDesc->Common.Type != ACPI_TYPE_BUFFER)
301         {
302             ACPI_ERROR ((AE_INFO,
303                          "SMBus/IPMI/GenericSerialBus write requires Buffer, found type %",
304                          "SMBus or IPMI write requires Buffer, found type %s",
305                          AcpiUtGetObjectTypeName (SourceDesc)));
306         }
307         return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
308     }
309     if (ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_SMBUS)
310     {
311         Length = ACPI_SMBUS_BUFFER_SIZE;
312         Function = ACPI_WRITE | (ObjDesc->Field.Attribute << 16);
313     }
314     else if (ObjDesc->Field.RegionObj->Region.SpaceId == ACPI_ADR_SPACE_GSBUS)
315     {
316         Length = ACPI_GSBUS_BUFFER_SIZE;
317         Function = ACPI_WRITE | (ObjDesc->Field.Attribute << 16);
318     }
319     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,
326                      "SMBus or IPMI write requires Buffer of length %u, found length
327                      Length, SourceDesc->Buffer.Length));
328     }
329     return_ACPI_STATUS (AE_AML_BUFFER_LIMIT);

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 }
384
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));
389
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));
397
398 /* Lock entire transaction if requested */
399
400 AcpiExAcquireGlobalLock (ObjDesc->CommonField.FieldFlags);
401
402 /* Write to the field */
403
404 Status = AcpiExInsertIntoField (ObjDesc, Buffer, Length);
405 AcpiExReleaseGlobalLock (ObjDesc->CommonField.FieldFlags);
406
407 return ACPI_STATUS (Status);
408 }
```

new/usr/src/common/acpica/components/executer/exfldio.c

1

```
*****
33386 Fri Jan 17 07:30:10 2014
new/usr/src/common/acpica/components/executer/exfldio.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: exfldio - Aml Field I/O
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXFLDIO_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acinterp.h"
50 #include "amlcode.h"
51 #include "acevents.h"
52 #include "acdispat.h"

55 #define _COMPONENT ACPI_EXECUTER
56 #define ACPI_MODULE_NAME ("exfldio")

58 /* Local prototypes */
```

new/usr/src/common/acpica/components/executer/exfldio.c

2

```
60 static ACPI_STATUS
61 AcpiExFieldDatumIo (
62     ACPI_OPERAND_OBJECT *ObjDesc,
63     UINT32 FieldDatumByteOffset,
64     UINT64 *Value,
65     UINT32 ReadWrite);

67 static BOOLEAN
68 AcpiExRegisterOverflow (
69     ACPI_OPERAND_OBJECT *ObjDesc,
70     UINT64 Value);

72 static ACPI_STATUS
73 AcpiExSetupRegion (
74     ACPI_OPERAND_OBJECT *ObjDesc,
75     UINT32 FieldDatumByteOffset);

78 /*****
79 *
80 * FUNCTION: AcpiExSetupRegion
81 *
82 * PARAMETERS: ObjDesc - Field to be read or written
83 *              FieldDatumByteOffset - Byte offset of this datum within the
84 *              parent field
85 *
86 * RETURN: Status
87 *
88 * DESCRIPTION: Common processing for AcpiExExtractFromField and
89 *              AcpiExInsertIntoField. Initialize the Region if necessary and
90 *              validate the request.
91 *
92 *****/

94 static ACPI_STATUS
95 AcpiExSetupRegion (
96     ACPI_OPERAND_OBJECT *ObjDesc,
97     UINT32 FieldDatumByteOffset)
98 {
99     ACPI_STATUS Status = AE_OK;
100     ACPI_OPERAND_OBJECT *RgnDesc;
101     UINT8 SpaceId;

104     ACPI_FUNCTION_TRACE_U32 (ExSetupRegion, FieldDatumByteOffset);

107     RgnDesc = ObjDesc->CommonField.RegionObj;

109     /* We must have a valid region */

111     if (RgnDesc->Common.Type != ACPI_TYPE_REGION)
112     {
113         ACPI_ERROR ((AE_INFO, "Needed Region, found type 0x%X (%s)",
114             RgnDesc->Common.Type,
115             AcpiUtGetObjectTypeName (RgnDesc)));
117         return ACPI_STATUS (AE_AML_OPERAND_TYPE);
118     }

120     SpaceId = RgnDesc->Region.SpaceId;

122     /* Validate the Space ID */

124     if (!AcpiIsValidSpaceId (SpaceId))
```

```

125 {
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
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)
150 {
151     if (RgnDesc->Region.SpaceId == ACPI_ADR_SPACE_SMBUS ||
152         RgnDesc->Region.SpaceId == ACPI_ADR_SPACE_IPMI)
153     {
154         /* SMBus or IPMI has a non-linear address space */
155         return ACPI_STATUS (AE_OK);
156     }
157 }

158 #ifdef ACPI_UNDER_DEVELOPMENT
159 /*
160  * If the Field access is AnyAcc, we can now compute the optimal
161  * access (because we know know the length of the parent region)
162  */
163 if (!(ObjDesc->Common.Flags & AOPOBJ_DATA_VALID))
164 {
165     if (ACPI_FAILURE (Status))
166     {
167         return ACPI_STATUS (Status);
168     }
169 }
170 #endif

171 /*
172  * Validate the request. The entire request from the byte offset for a
173  * length of one field datum (access width) must fit within the region.
174  * (Region length is specified in bytes)
175  */
176 if (RgnDesc->Region.Length <
177     (ObjDesc->CommonField.BaseByteOffset + FieldDatumByteOffset +
178     ObjDesc->CommonField.AccessByteWidth))
179 {
180     if (AcpiGbl_EnableInterpreterSlack)
181     {
182         /*
183          * Slack mode only: We will go ahead and allow access to this
184          * field if it is within the region length rounded up to the next
185          * access width boundary. ACPI_SIZE cast for 64-bit compile.
186          */
187         if (ACPI_ROUND_UP (RgnDesc->Region.Length,
188             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 [%4.
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  *                Value              - parent field
237  *                Function            - Where to store value (must at least
238  *                64 bits)
239  *                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,
307         Status = AcpiEvAddressSpaceDispatch (RgnDesc, Function, RegionOffset,
308             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 {
353     if (ObjDesc->CommonField.BitLength >= ACPI_INTEGER_BIT_SIZE)
354     {
355         /*
356         * The field is large enough to hold the maximum integer, so we can
357         * never overflow it.
358         */
359         return (FALSE);
360     }

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 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     * BufferField - Read/write from/to a Buffer
433     * RegionField - Read/write from/to a Operation Region.
434     * BankField - Write to a Bank Register, then read/write from/to an
435     *              OperationRegion
436     * IndexField - Write to an Index Register, then read/write from/to a
437     *              Data Register
438     */
439     switch (ObjDesc->Common.Type)
440     {
441     case ACPI_TYPE_BUFFER_FIELD:
442         /*
443         * If the BufferField arguments have not been previously evaluated,
444         * evaluate them now and save the results.
445         */
446         if (!(ObjDesc->Common.Flags & AOPOBJ_DATA_VALID))
447         {
448             Status = AcpiDsGetBufferFieldArguments (ObjDesc);
449             if (ACPI_FAILURE (Status))
450

```

```

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      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));
751     return_ACPI_STATUS (AE_BUFFER_OVERFLOW);
752 }

754 ACPI_MEMSET (Buffer, 0, BufferLength);
755 AccessBitWidth = ACPI_MUL_8 (ObjDesc->CommonField.AccessByteWidth);

757 /* Handle the simple case here */

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 */
770         Status = AcpiExFieldDatumIo (ObjDesc, 0, &RawDatum, ACPI_READ);
771         ACPI_MEMCPY (Buffer, &RawDatum, BufferLength);
772     }
774     return_ACPI_STATUS (Status);
775 }

777 /* TBD: Move to common setup code */

779 /* Field algorithm is limited to sizeof(UINT64), truncate if needed */

781 if (ObjDesc->CommonField.AccessByteWidth > sizeof (UINT64))
782 {
783     ObjDesc->CommonField.AccessByteWidth = sizeof (UINT64);
784     AccessBitWidth = sizeof (UINT64) * 8;
785 }

787 /* Compute the number of datums (access width data items) */

789 DatumCount = ACPI_ROUND_UP_TO (
790     ObjDesc->CommonField.BitLength, AccessBitWidth);

792 FieldDatumCount = ACPI_ROUND_UP_TO (
793     ObjDesc->CommonField.BitLength +
794     ObjDesc->CommonField.StartFieldBitOffset, AccessBitWidth);

796 /* Priming read from the field */

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;

805 /* Read the rest of the field */

807 for (i = 1; i < FieldDatumCount; i++)
808 {
809     /* Get next input datum from the field */

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     }

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     }

834     if (i == DatumCount)
835     {
836         break;
837     }

839     /* Write merged datum to target buffer */

841     ACPI_MEMCPY (((char *) Buffer) + BufferOffset, &MergedDatum,
842         ACPI_MIN(ObjDesc->CommonField.AccessByteWidth,
843             BufferLength - BufferOffset));

845     BufferOffset += ObjDesc->CommonField.AccessByteWidth;
846     MergedDatum = RawDatum >> ObjDesc->CommonField.StartFieldBitOffset;
847 }

849 /* Mask off any extra bits in the last datum */

851 BufferTailBits = ObjDesc->CommonField.BitLength % AccessBitWidth;
852 if (BufferTailBits)
853 {
854     MergedDatum &= ACPI_MASK_BITS_ABOVE (BufferTailBits);
855 }

857 /* Write the last datum to the buffer */

859 ACPI_MEMCPY (((char *) Buffer) + BufferOffset, &MergedDatum,
860     ACPI_MIN(ObjDesc->CommonField.AccessByteWidth,
861         BufferLength - BufferOffset));

863     return_ACPI_STATUS (AE_OK);
864 }

```

unchanged portion omitted

new/usr/src/common/acpica/components/executer/exmisc.c

1

\*\*\*\*\*

23075 Fri Jan 17 07:30:10 2014

new/usr/src/common/acpica/components/executer/exmisc.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: exmisc - ACPI AML (p-code) execution - specific opcodes
4 *
5 *****/
```

```
7 /*
```

```
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXMISC_C__
```

```
46 #include "acpi.h"
```

```
47 #include "accommon.h"
```

```
48 #include "acinterp.h"
```

```
49 #include "amlcode.h"
```

```
50 #include "amlresrc.h"
```

```
53 #define _COMPONENT      ACPI_EXECUTER
```

```
54     ACPI_MODULE_NAME    ("exmisc")
```

```
57 /*****
```

new/usr/src/common/acpica/components/executer/exmisc.c

2

```
58 *
```

```
59 * FUNCTION:      AcpiExGetObjectReference
```

```
60 *
```

```
61 * PARAMETERS:    ObjDesc          - Create a reference to this object
```

```
62 *               ReturnDesc        - Where to store the reference
```

```
63 *               WalkState          - Current state
```

```
64 *
```

```
65 * RETURN:        Status
```

```
66 *
```

```
67 * DESCRIPTION:    Obtain and return a "reference" to the target object
```

```
68 *               Common code for the RefOfOp and the CondRefOfOp.
```

```
69 *
```

```
70 *****/
```

```
72 ACPI_STATUS
```

```
73 AcpiExGetObjectReference (
```

```
74     ACPI_OPERAND_OBJECT *ObjDesc,
```

```
75     ACPI_OPERAND_OBJECT **ReturnDesc,
```

```
76     ACPI_WALK_STATE *WalkState)
```

```
77 {
```

```
78     ACPI_OPERAND_OBJECT *ReferenceObj;
```

```
79     ACPI_OPERAND_OBJECT *ReferencedObj;
```

```
82     ACPI_FUNCTION_TRACE_PTR (ExGetObjectReference, ObjDesc);
```

```
85     *ReturnDesc = NULL;
```

```
87     switch (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc))
```

```
88     {
```

```
89     case ACPI_DESC_TYPE_OPERAND:
```

```
91         if (ObjDesc->Common.Type != ACPI_TYPE_LOCAL_REFERENCE)
```

```
92         {
```

```
93             return ACPI_STATUS (AE_AML_OPERAND_TYPE);
```

```
94         }
```

```
96     /*
```

```
97      * Must be a reference to a Local or Arg
```

```
98      */
```

```
99     switch (ObjDesc->Reference.Class)
```

```
100     {
```

```
101     case ACPI_REFCLASS_LOCAL:
```

```
102     case ACPI_REFCLASS_ARG:
```

```
103     case ACPI_REFCLASS_DEBUG:
```

```
105         /* The referenced object is the pseudo-node for the local/arg */
```

```
107         ReferencedObj = ObjDesc->Reference.Object;
```

```
108         break;
```

```
110     default:
```

```
112         ACPI_ERROR ((AE_INFO, "Unknown Reference Class 0x%2.2X",
```

```
113             ObjDesc->Reference.Class));
```

```
114         return ACPI_STATUS (AE_AML_INTERNAL);
```

```
115     }
```

```
116     break;
```

```
118     case ACPI_DESC_TYPE_NAMED:
```

```
119     /*
```

```
120      * A named reference that has already been resolved to a Node
```

```
121      */
```

```

122     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     *ReturnDesc = LocalOperand1;
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);

348     /* Copy the second integer (LSB first) after the first */

350     ACPI_MEMCPY (NewBuf + AcpiGbl_IntegerByteWidth,
351                  &LocalOperand1->Integer.Value,
352                  AcpiGbl_IntegerByteWidth);
353     break;

355     case ACPI_TYPE_STRING:

357         /* Result of two Strings is a String */

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         }

368         NewBuf = ReturnDesc->String.Pointer;

370         /* Concatenate the strings */

372         ACPI_STRCPY (NewBuf, Operand0->String.Pointer);
373         ACPI_STRCPY (NewBuf + Operand0->String.Length,
374                     LocalOperand1->String.Pointer);
375         break;

377     case ACPI_TYPE_BUFFER:

379         /* Result of two Buffers is a Buffer */

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         }

390         NewBuf = (char *) ReturnDesc->Buffer.Pointer;

392         /* Concatenate the buffers */

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;

401     default:

403         /* Invalid object type, should not happen here */

405         ACPI_ERROR ((AE_INFO, "Invalid object type: 0x%X",
406                     Operand0->Common.Type));
407         Status = AE_AML_INTERNAL;
408         goto Cleanup;
409 }

```

```

411     *ActualReturnDesc = ReturnDesc;

413 Cleanup:
414     if (LocalOperand1 != Operand1)
415     {
416         AcpiUtRemoveReference (LocalOperand1);
417     }
418     return_ACPI_STATUS (Status);
419 }

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  *****/

438 UINT64
439 AcpiExDoMathOp (
440     UINT16      Opcode,
441     UINT64      Integer0,
442     UINT64      Integer1)
443 {
444     ACPI_FUNCTION_ENTRY ();

448     switch (Opcode)
449     {
450     case AML_ADD_OP:                /* Add (Integer0, Integer1, Result) */

452         return (Integer0 + Integer1);

454     case AML_BIT_AND_OP:            /* And (Integer0, Integer1, Result) */

456         return (Integer0 & Integer1);

458     case AML_BIT_NAND_OP:           /* NAnd (Integer0, Integer1, Result) */

460         return (~(Integer0 & Integer1));

462     case AML_BIT_OR_OP:             /* Or (Integer0, Integer1, Result) */

464         return (Integer0 | Integer1);

466     case AML_BIT_NOR_OP:            /* NOR (Integer0, Integer1, Result) */

468         return (~(Integer0 | Integer1));

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:
600 *           Result = (Operand0 == Operand1);
601 *
602 *****/
604 ACPI_STATUS
605 AcpiExDoLogicalOp (
606     UINT16          Opcode,
607     ACPI_OPERAND_OBJECT *Operand0,
608     ACPI_OPERAND_OBJECT *Operand1,
609     BOOLEAN          *LogicalResult)
610 {
611     ACPI_OPERAND_OBJECT *LocalOperand1 = Operand1;
612     UINT64               Integer0;
613     UINT64               Integer1;
614     UINT32               Length0;
615     UINT32               Length1;
616     ACPI_STATUS          Status = AE_OK;
617     BOOLEAN              LocalResult = FALSE;
618     int                  Compare;

621     ACPI_FUNCTION_TRACE (ExDoLogicalOp);

624     /*
625     * Convert the second operand if necessary. The first operand
626     * determines the type of the second operand, (See the Data Types
627     * section of the ACPI 3.0+ specification.) Both object types are
628     * guaranteed to be either Integer/String/Buffer by the operand
629     * resolution mechanism.
630     */
631     switch (Operand0->Common.Type)
632     {
633     case ACPI_TYPE_INTEGER:

635         Status = AcpiExConvertToInteger (Operand1, &LocalOperand1, 16);
636         break;

638     case ACPI_TYPE_STRING:

640         Status = AcpiExConvertToString (Operand1, &LocalOperand1,
641             ACPI_IMPLICIT_CONVERT_HEX);
642         break;

644     case ACPI_TYPE_BUFFER:

646         Status = AcpiExConvertToBuffer (Operand1, &LocalOperand1);
647         break;

649     default:

651         Status = AE_AML_INTERNAL;
652         break;
653     }

655     if (ACPI_FAILURE (Status))
656     {
657         goto Cleanup;
658     }

660     /*
661     * Two cases: 1) Both Integers, 2) Both Strings or Buffers
662     */
663     if (Operand0->Common.Type == ACPI_TYPE_INTEGER)
664     {

```

```

665     /*
666     * 1) Both operands are of type integer
667     *   Note: LocalOperand1 may have changed above
668     */
669     Integer0 = Operand0->Integer.Value;
670     Integer1 = LocalOperand1->Integer.Value;

672     switch (Opcode)
673     {
674     case AML_LEQUAL_OP:           /* LEqual (Operand0, Operand1) */

676         if (Integer0 == Integer1)
677         {
678             LocalResult = TRUE;
679         }
680         break;

682     case AML_LGREATER_OP:        /* LGreater (Operand0, Operand1) */

684         if (Integer0 > Integer1)
685         {
686             LocalResult = TRUE;
687         }
688         break;

690     case AML_LLESS_OP:          /* LLess (Operand0, Operand1) */

692         if (Integer0 < Integer1)
693         {
694             LocalResult = TRUE;
695         }
696         break;

698     default:

700         Status = AE_AML_INTERNAL;
701         break;
702     }
703     }
704     else
705     {
706     /*
707     * 2) Both operands are Strings or both are Buffers
708     *   Note: Code below takes advantage of common Buffer/String
709     *   object fields. LocalOperand1 may have changed above. Use
710     *   memcmp to handle nulls in buffers.
711     */
712     Length0 = Operand0->Buffer.Length;
713     Length1 = LocalOperand1->Buffer.Length;

715     /* Lexicographic compare: compare the data bytes */

717     Compare = ACPI_MEMCMP (Operand0->Buffer.Pointer,
718         LocalOperand1->Buffer.Pointer,
719         (Length0 > Length1) ? Length1 : Length0);

721     switch (Opcode)
722     {
723     case AML_LEQUAL_OP:           /* LEqual (Operand0, Operand1) */

725         /* Length and all bytes must be equal */

727         if ((Length0 == Length1) &&
728             (Compare == 0))
729         {
730             /* 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 }

```

new/usr/src/common/acpica/components/executer/exmutex.c

1

\*\*\*\*\*

16339 Fri Jan 17 07:30:11 2014

new/usr/src/common/acpica/components/executer/exmutex.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: exmutex - ASL Mutex Acquire/Release functions
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #define __EXMUTEX_C__
46
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acinterp.h"
50 #include "acevents.h"
51
52 #define _COMPONENT ACPI_EXECUTER
53 #define ACPI_MODULE_NAME ("exmutex")
54
55 /* Local prototypes */
56
57 static void
58 AcpiExLinkMutex (
```

new/usr/src/common/acpica/components/executer/exmutex.c

2

```
58 ACPI_OPERAND_OBJECT *ObjDesc,
59 ACPI_THREAD_STATE *Thread);
60
61 /*****
62 *
63 * FUNCTION: AcpiExUnlinkMutex
64 *
65 * PARAMETERS: ObjDesc - The mutex to be unlinked
66 *
67 * RETURN: None
68 *
69 * DESCRIPTION: Remove a mutex from the "AcquiredMutex" list
70 *
71 *****/
72
73 void
74 AcpiExUnlinkMutex (
75     ACPI_OPERAND_OBJECT *ObjDesc)
76 {
77     ACPI_THREAD_STATE *Thread = ObjDesc->Mutex.OwnerThread;
78
79     if (!Thread)
80     {
81         return;
82     }
83
84     /* Doubly linked list */
85
86     if (ObjDesc->Mutex.Next)
87     {
88         (ObjDesc->Mutex.Next)->Mutex.Prev = ObjDesc->Mutex.Prev;
89     }
90
91     if (ObjDesc->Mutex.Prev)
92     {
93         (ObjDesc->Mutex.Prev)->Mutex.Next = ObjDesc->Mutex.Next;
94     }
95
96     /*
97     * Migrate the previous sync level associated with this mutex to
98     * the previous mutex on the list so that it may be preserved.
99     * This handles the case where several mutexes have been acquired
100     * at the same level, but are not released in opposite order.
101     */
102     (ObjDesc->Mutex.Prev)->Mutex.OriginalSyncLevel =
103         ObjDesc->Mutex.OriginalSyncLevel;
104 }
105
106 else
107 {
108     Thread->AcquiredMutexList = ObjDesc->Mutex.Next;
109 }
110 }
111
112 /* unchanged portion omitted */
113
114 /*****
115 *
116 * FUNCTION: AcpiExReleaseMutexObject
117 *
118 * PARAMETERS: ObjDesc - The object descriptor for this op
119 *
120 * RETURN: Status
121 *
122 * DESCRIPTION: Release a previously acquired Mutex, low level interface.
123 * Provides a common path that supports multiple releases (after
```

```

319 *           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;
336
337     ACPI_FUNCTION_TRACE (ExReleaseMutexObject);
338
339     if (ObjDesc->Mutex.AcquisitionDepth == 0)
340     {
341         return ACPI_STATUS (AE_NOT_ACQUIRED);
342         return (AE_NOT_ACQUIRED);
343     }
344
345     /* Match multiple Acquires with multiple Releases */
346
347     ObjDesc->Mutex.AcquisitionDepth--;
348     if (ObjDesc->Mutex.AcquisitionDepth != 0)
349     {
350         /* Just decrement the depth and return */
351
352         return ACPI_STATUS (AE_OK);
353     }
354
355     if (ObjDesc->Mutex.OwnerThread)
356     {
357         /* Unlink the mutex from the owner's list */
358
359         AcpiExUnlinkMutex (ObjDesc);
360         ObjDesc->Mutex.OwnerThread = NULL;
361     }
362
363     /* Release the mutex, special case for Global Lock */
364
365     if (ObjDesc == AcpiGbl_GlobalLockMutex)
366     {
367         Status = AcpiEvReleaseGlobalLock ();
368     }
369     else
370     {
371         AcpiOsReleaseMutex (ObjDesc->Mutex.OsMutex);
372     }
373
374     /* Clear mutex info */
375
376     ObjDesc->Mutex.ThreadId = 0;
377     return ACPI_STATUS (Status);
378 }
379
380 _____ 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;
515
516     ACPI_FUNCTION_NAME (ExReleaseAllMutexes);
517     ACPI_FUNCTION_ENTRY ();
518
519     /* Traverse the list of owned mutexes, releasing each one */
520
521     while (Next)
522     {
523         ObjDesc = Next;
524         Next = ObjDesc->Mutex.Next;
525
526         ObjDesc->Mutex.Prev = NULL;
527         ObjDesc->Mutex.Next = NULL;
528         ObjDesc->Mutex.AcquisitionDepth = 0;
529
530         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
531             "Force-releasing held mutex: %p\n", ObjDesc));
532
533         /* Release the mutex, special case for Global Lock */
534
535         if (ObjDesc == AcpiGbl_GlobalLockMutex)
536         {
537             /* Ignore errors */
538
539             (void) AcpiEvReleaseGlobalLock ();
540         }
541         else
542         {
543             AcpiOsReleaseMutex (ObjDesc->Mutex.OsMutex);
544         }
545
546         /* Mark mutex unowned */
547
548         ObjDesc->Mutex.OwnerThread = NULL;
549         ObjDesc->Mutex.ThreadId = 0;
550
551         /* Update Thread SyncLevel (Last mutex is the important one) */
552
553         Thread->CurrentSyncLevel = ObjDesc->Mutex.OriginalSyncLevel;
554     }
555 }
556

```

new/usr/src/common/acpica/components/executer/exnames.c

1

\*\*\*\*\*

13641 Fri Jan 17 07:30:11 2014

new/usr/src/common/acpica/components/executer/exnames.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: exnames - interpreter/scanner name load/execute
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #define __EXNAMES_C__
46
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acinterp.h"
50 #include "amlcode.h"
51
52 #define _COMPONENT ACPI_EXECUTER
53 #define ACPI_MODULE_NAME ("exnames")
54
55 /* Local prototypes */
56
57 static char *
```

new/usr/src/common/acpica/components/executer/exnames.c

2

```
58     UINT32 PrefixCount,
59     UINT32 NumNameSegs);
60
61 static ACPI_STATUS
62 AcpiExNameSegment (
63     UINT8 **InAmlAddress,
64     char *NameString);
65
66 /*****
67 *
68 * FUNCTION: AcpiExAllocateNameString
69 *
70 * PARAMETERS: PrefixCount - Count of parent levels. Special cases:
71 * (-1)==root, 0==none
72 * NumNameSegs - count of 4-character name segments
73 *
74 * RETURN: A pointer to the allocated string segment. This segment must
75 * be deleted by the caller.
76 *
77 * DESCRIPTION: Allocate a buffer for a name string. Ensure allocated name
78 * string is long enough, and set up prefix if any.
79 *
80 *****/
81
82 static char *
83 AcpiExAllocateNameString (
84     UINT32 PrefixCount,
85     UINT32 NumNameSegs)
86 {
87     char *TempPtr;
88     char *NameString;
89     SizeNeeded;
90
91     ACPI_FUNCTION_TRACE (ExAllocateNameString);
92
93     /*
94     * Allow room for all \ and ^ prefixes, all segments and a MultiNamePrefix.
95     * Also, one byte for the null terminator.
96     * This may actually be somewhat longer than needed.
97     */
98     if (PrefixCount == ACPI_UINT32_MAX)
99     {
100         /* Special case for root */
101
102         SizeNeeded = 1 + (ACPI_NAME_SIZE * NumNameSegs) + 2 + 1;
103     }
104     else
105     {
106         SizeNeeded = PrefixCount + (ACPI_NAME_SIZE * NumNameSegs) + 2 + 1;
107     }
108
109     /*
110     * Allocate a buffer for the name.
111     * This buffer must be deleted by the caller!
112     */
113     NameString = ACPI_ALLOCATE (SizeNeeded);
114     if (!NameString)
115     {
116         ACPI_ERROR ((AE_INFO,
117             "Could not allocate size %u", SizeNeeded));
118         return_PTR (NULL);
119     }
120
121     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 }



new/usr/src/common/acpica/components/executer/exoparg1.c

1

\*\*\*\*\*

34340 Fri Jan 17 07:30:12 2014

new/usr/src/common/acpica/components/executer/exoparg1.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: exoparg1 - AML execution - opcodes with 1 argument
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXOPARG1_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acparser.h"
49 #include "acdispat.h"
50 #include "acinterp.h"
51 #include "amlcode.h"
52 #include "acnamesp.h"
```

```
55 #define _COMPONENT      ACPI_EXECUTER
56 #define ACPI_MODULE_NAME    ("exoparg1")
```

new/usr/src/common/acpica/components/executer/exoparg1.c

2

```
59 /*!
60 * Naming convention for AML interpreter execution routines.
61 *
62 * The routines that begin execution of AML opcodes are named with a common
63 * convention based upon the number of arguments, the number of target operands,
64 * and whether or not a value is returned:
65 *
66 *      AcpiExOpcode_xA_yT_zR
67 *
68 * Where:
69 *
70 * xA - ARGUMENTS:   The number of arguments (input operands) that are
71 *                   required for this opcode type (0 through 6 args).
72 * yT - TARGETS:     The number of targets (output operands) that are required
73 *                   for this opcode type (0, 1, or 2 targets).
74 * zR - RETURN VALUE: Indicates whether this opcode type returns a value
75 *                   as the function return (0 or 1).
76 *
77 * The AcpiExOpcode* functions are called via the Dispatcher component with
78 * fully resolved operands.
79 !*/
80
81 /*****
82 *
83 * FUNCTION:      AcpiExOpcode_0A_0T_1R
84 *
85 * PARAMETERS:    WalkState           - Current state (contains AML opcode)
86 *
87 * RETURN:        Status
88 *
89 * DESCRIPTION:   Execute operator with no operands, one return value
90 *
91 *****/
92
93 ACPI_STATUS
94 AcpiExOpcode_0A_0T_1R (
95     ACPI_WALK_STATE      *WalkState)
96 {
97     ACPI_STATUS           Status = AE_OK;
98     ACPI_OPERAND_OBJECT  *ReturnDesc = NULL;
99
100     ACPI_FUNCTION_TRACE_STR (ExOpcode_0A_0T_1R,
101                             AcpiPsGetOpcodeName (WalkState->Opcode));
102
103     /* Examine the AML opcode */
104
105     switch (WalkState->Opcode)
106     {
107     case AML_TIMER_OP:      /* Timer () */
108
109         /* Create a return object of type Integer */
110
111         ReturnDesc = AcpiUtCreateIntegerObject (AcpiOsGetTimer ());
112         if (!ReturnDesc)
113         {
114             Status = AE_NO_MEMORY;
115             goto Cleanup;
116         }
117         goto Cleanup;
118     }
119     break;
120
121     default:
122         /* Unknown opcode */
123
124         ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
```

```

124     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) */
551
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;
560
561     case AML_SHIFT_LEFT_BIT_OP:      /* ShiftLeftBit (Source, BitNum) */
562     case AML_SHIFT_RIGHT_BIT_OP:    /* ShiftRightBit (Source, BitNum) */
563
564         /* These are two obsolete opcodes */
565
566         ACPI_ERROR ((AE_INFO,
567             "%s is obsolete and not implemented",
568             AcpiPsGetOpcodeName (WalkState->Opcode)));
569         Status = AE_SUPPORT;
570         goto Cleanup;
571
572     default:                          /* Unknown opcode */
573
574         ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
575             WalkState->Opcode));
576         Status = AE_AML_BAD_OPCODE;
577         goto Cleanup;
578     }
579
580     if (ACPI_SUCCESS (Status))
581     {
582         /* Store the return value computed above into the target object */
583
584         Status = AcpiExStore (ReturnDesc, Operand[1], WalkState);
585     }
586
587 Cleanup:
588
589     /* Delete return object on error */
590
591     if (ACPI_FAILURE (Status))
592     {
593         AcpiUtRemoveReference (ReturnDesc);
594     }
595
596     /* Save return object on success */
597
598     else if (!WalkState->ResultObj)
599     {
600         WalkState->ResultObj = ReturnDesc;
601     }
602
603     return ACPI_STATUS (Status);
604 }
605
606 /*****
607  *
608  * FUNCTION:      AcpiExOpcode_1A_0T_1R
609  *
610  */

```

```

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  *****/
619
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;
630
631     ACPI_FUNCTION_TRACE_STR (ExOpcode_1A_0T_1R,
632         AcpiPsGetOpcodeName (WalkState->Opcode));
633
634     /* Examine the AML opcode */
635
636     switch (WalkState->Opcode)
637     {
638     case AML_LNOT_OP:                /* LNot (Operand) */
639
640         ReturnDesc = AcpiUtCreateIntegerObject ((UINT64) 0);
641         if (!ReturnDesc)
642         {
643             Status = AE_NO_MEMORY;
644             goto Cleanup;
645         }
646
647         /*
648          * Set result to ONES (TRUE) if Value == 0. Note:
649          * ReturnDesc->Integer.Value is initially == 0 (FALSE) from above.
650          */
651         if (!Operand[0]->Integer.Value)
652         {
653             ReturnDesc->Integer.Value = ACPI_UINT64_MAX;
654         }
655         break;
656
657     case AML_DECREMENT_OP:           /* Decrement (Operand) */
658     case AML_INCREMENT_OP:           /* Increment (Operand) */
659
660         /*
661          * Create a new integer. Can't just get the base integer and
662          * increment it because it may be an Arg or Field.
663          */
664         ReturnDesc = AcpiUtCreateInternalObject (ACPI_TYPE_INTEGER);
665         if (!ReturnDesc)
666         {
667             Status = AE_NO_MEMORY;
668             goto Cleanup;
669         }
670
671         /*
672          * Since we are expecting a Reference operand, it can be either a
673          * NS Node or an internal object.
674          */

```

```

676     TempDesc = Operand[0];
677     if (ACPI_GET_DESCRIPTOR_TYPE (TempDesc) == ACPI_DESC_TYPE_OPERAND)
678     {
679         /* Internal reference object - prevent deletion */
680
681         AcpiUtAddReference (TempDesc);
682     }
683
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)));
697
698         goto Cleanup;
699     }
700
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     }
713
714     /* Finished with this Integer object */
715
716     AcpiUtRemoveReference (TempDesc);
717
718     /*
719     * Store the result back (indirectly) through the original
720     * Reference object
721     */
722     Status = AcpiExStore (ReturnDesc, Operand[0], WalkState);
723     break;
724
725     case AML_TYPE_OP:                /* ObjectType (SourceObject) */
726
727     /*
728     * Note: The operand is not resolved at this point because we want to
729     * get the associated object, not its value. For example, we don't
730     * want to resolve a FieldUnit to its value, we want the actual
731     * FieldUnit object.
732     */
733
734     /* Get the type of the base object */
735
736     Status = AcpiExResolveMultiple (WalkState, Operand[0], &Type, NULL);
737     if (ACPI_FAILURE (Status))
738     {
739         goto Cleanup;
740     }

```

```

741     /* Allocate a descriptor to hold the type. */
742
743     ReturnDesc = AcpiUtCreateIntegerObject ((UINT64) Type);
744     if (!ReturnDesc)
745     {
746         Status = AE_NO_MEMORY;
747         goto Cleanup;
748     }
749     break;
750
751     case AML_SIZE_OF_OP:            /* SizeOf (SourceObject) */
752
753     /*
754     * Note: The operand is not resolved at this point because we want to
755     * get the associated object, not its value.
756     */
757
758     /* Get the base object */
759
760     Status = AcpiExResolveMultiple (WalkState,
761         Operand[0], &Type, &TempDesc);
762     if (ACPI_FAILURE (Status))
763     {
764         goto Cleanup;
765     }
766
767     /*
768     * The type of the base object must be integer, buffer, string, or
769     * package. All others are not supported.
770     *
771     * NOTE: Integer is not specifically supported by the ACPI spec,
772     * but is supported implicitly via implicit operand conversion.
773     * rather than bother with conversion, we just use the byte width
774     * global (4 or 8 bytes).
775     */
776     switch (Type)
777     {
778     case ACPI_TYPE_INTEGER:
779         Value = AcpiGbl_IntegerByteWidth;
780         break;
781
782     case ACPI_TYPE_STRING:
783
784         Value = TempDesc->String.Length;
785         break;
786
787     case ACPI_TYPE_BUFFER:
788
789         /* Buffer arguments may not be evaluated at this point */
790
791         Status = AcpiDsGetBufferArguments (TempDesc);
792         Value = TempDesc->Buffer.Length;
793         break;
794
795     case ACPI_TYPE_PACKAGE:
796
797         /* Package arguments may not be evaluated at this point */
798
799         Status = AcpiDsGetPackageArguments (TempDesc);
800         Value = TempDesc->Package.Count;
801         break;
802
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                  * 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,
937             ACPI_CAST_INDIRECT_PTR (
938                 ACPI_NAMESPACE_NODE, &ReturnDesc));
939         if (ACPI_FAILURE (Status))
940         {
941             goto Cleanup;
942         }
943
944         Status = AcpiExResolveNodeToValue (
945             ACPI_CAST_INDIRECT_PTR (
946                 ACPI_NAMESPACE_NODE, &ReturnDesc),
947             WalkState);
948         goto Cleanup;
949     }
950 }
951
952 /* Operand[0] may have changed from the code above */
953
954 if (ACPI_GET_DESCRIPTOR_TYPE (Operand[0]) == ACPI_DESC_TYPE_NAMED)
955 {
956     /*
957     * This is a DerefOf (ObjectReference)
958     * Get the actual object from the Node (This is the dereference).
959     * This case may only happen when a LocalX or ArgX is
960     * dereferenced above.
961     */
962     ReturnDesc = AcpiNsGetAttachedObject (
963         (ACPI_NAMESPACE_NODE *) Operand[0]);
964     AcpiUtAddReference (ReturnDesc);
965 }
966 else
967 {
968     /*
969     * This must be a reference object produced by either the
970     * Index() or RefOf() operator
971     */
972     switch (Operand[0]->Reference.Class)
973     {
974     case ACPI_REFCLASS_INDEX:
975
976         /*
977         * The target type for the Index operator must be
978         * either a Buffer or a Package
979         */
980         switch (Operand[0]->Reference.TargetType)
981         {
982         case ACPI_TYPE_BUFFER_FIELD:
983
984             TempDesc = Operand[0]->Reference.Object;
985
986             /*
987             * Create a new object that contains one element of the
988             * buffer -- the element pointed to by the index.
989             *
990             * NOTE: index into a buffer is NOT a pointer to a
991             * sub-buffer of the main buffer, it is only a pointer to a
992             * single element (byte) of the buffer!
993             *
994             * Since we are returning the value of the buffer at the
995             * indexed location, we don't need to add an additional
996             * reference to the buffer itself.
997             */
998             ReturnDesc = AcpiUtCreateIntegerObject ((UINT64)
999                 TempDesc->Buffer.Pointer[Operand[0]->Reference.Value]);
1000             if (!ReturnDesc)
1001             {

```

```

1001             Status = AE_NO_MEMORY;
1002             goto Cleanup;
1003         }
1004         break;
1005
1006     case ACPI_TYPE_PACKAGE:
1007
1008         /*
1009         * Return the referenced element of the package. We must
1010         * add another reference to the referenced object, however.
1011         */
1012         ReturnDesc = *(Operand[0]->Reference.Where);
1013         if (!ReturnDesc)
1014         {
1015             /*
1016             * Element is NULL, do not allow the dereference.
1017             * This provides compatibility with other ACPI
1018             * implementations.
1019             */
1020             return ACPI_STATUS (AE_AML_UNINITIALIZED_ELEMENT);
1021         }
1022         AcpiUtAddReference (ReturnDesc);
1023         break;
1024
1025     default:
1026
1027         ACPI_ERROR ((AE_INFO,
1028             "Unknown Index TargetType 0x%X in reference object %p",
1029             Operand[0]->Reference.TargetType, Operand[0]));
1030         Status = AE_AML_OPERAND_TYPE;
1031         goto Cleanup;
1032     }
1033     break;
1034
1035     case ACPI_REFCLASS_REFOF:
1036
1037         ReturnDesc = Operand[0]->Reference.Object;
1038
1039         if (ACPI_GET_DESCRIPTOR_TYPE (ReturnDesc) ==
1040             ACPI_DESC_TYPE_NAMED)
1041         {
1042             ReturnDesc = AcpiNsGetAttachedObject (
1043                 (ACPI_NAMESPACE_NODE *) ReturnDesc);
1044             if (!ReturnDesc)
1045             {
1046                 break;
1047             }
1048
1049             /*
1050             * June 2013:
1051             * BufferFields/FieldUnits require additional resolution
1052             */
1053             switch (ReturnDesc->Common.Type)
1054             {
1055             case ACPI_TYPE_BUFFER_FIELD:
1056             case ACPI_TYPE_LOCAL_REGION_FIELD:
1057             case ACPI_TYPE_LOCAL_BANK_FIELD:
1058             case ACPI_TYPE_LOCAL_INDEX_FIELD:
1059                 /* Add another reference to the object! */

```



```
1060         Status = AcpiExReadDataFromField (WalkState,
1061         ReturnDesc, &TempDesc);
1062         if (ACPI_FAILURE (Status))
1063         {
1064             goto Cleanup;
1065         }
1067         ReturnDesc = TempDesc;
1068         break;
1070     default:
1072         /* Add another reference to the object */
1074         AcpiUtAddReference (ReturnDesc);
1075         break;
1076     }
1077 }
1078 break;
1080 default:
1072     default:
1082         ACPI_ERROR ((AE_INFO,
1083         "Unknown class in reference(%p) - 0x%2.2X",
1084         Operand[0], Operand[0]->Reference.Class));
1086         Status = AE_TYPE;
1087         goto Cleanup;
1088     }
1089 }
1090 break;
1092 default:
1094     ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
1095     WalkState->Opcode));
1096     Status = AE_AML_BAD_OPCODE;
1097     goto Cleanup;
1098 }
1101 Cleanup:
1103     /* Delete return object on error */
1105     if (ACPI_FAILURE (Status))
1106     {
1107         AcpiUtRemoveReference (ReturnDesc);
1108     }
1110     /* Save return object on success */
1112     else
1113     {
1114         WalkState->ResultObj = ReturnDesc;
1115     }
1117     return ACPI_STATUS (Status);
1118 }
```

new/usr/src/common/acpica/components/executer/exoparg2.c

1

```
*****
17928 Fri Jan 17 07:30:12 2014
new/usr/src/common/acpica/components/executer/exoparg2.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: exoparg2 - AML execution - opcodes with 2 arguments
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXOPARG2_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acparser.h"
50 #include "acinterp.h"
51 #include "acevents.h"
52 #include "amlcode.h"

55 #define _COMPONENT ACPI_EXECUTER
56 #define ACPI_MODULE_NAME ("exoparg2")
```

new/usr/src/common/acpica/components/executer/exoparg2.c

2

```
59 /*!
60 * Naming convention for AML interpreter execution routines.
61 *
62 * The routines that begin execution of AML opcodes are named with a common
63 * convention based upon the number of arguments, the number of target operands,
64 * and whether or not a value is returned:
65 *
66 * AcpiExOpcode_xA_yT_zR
67 *
68 * Where:
69 *
70 * xA - ARGUMENTS: The number of arguments (input operands) that are
71 * required for this opcode type (1 through 6 args).
72 * yT - TARGETS: The number of targets (output operands) that are required
73 * for this opcode type (0, 1, or 2 targets).
74 * zR - RETURN VALUE: Indicates whether this opcode type returns a value
75 * as the function return (0 or 1).
76 *
77 * The AcpiExOpcode* functions are called via the Dispatcher component with
78 * fully resolved operands.
79 !*/

82 /*****
83 *
84 * FUNCTION: AcpiExOpcode_2A_0T_0R
85 *
86 * PARAMETERS: WalkState - Current walk state
87 *
88 * RETURN: Status
89 *
90 * DESCRIPTION: Execute opcode with two arguments, no target, and no return
91 * value.
92 *
93 * ALLOCATION: Deletes both operands
94 *
95 *****/

97 ACPI_STATUS
98 AcpiExOpcode_2A_0T_0R (
99     ACPI_WALK_STATE *WalkState)
100 {
101     ACPI_OPERAND_OBJECT **Operand = &WalkState->Operands[0];
102     ACPI_NAMESPACE_NODE *Node;
103     UINT32 Value;
104     ACPI_STATUS Status = AE_OK;

107     ACPI_FUNCTION_TRACE_STR (ExOpcode_2A_0T_0R,
108                             AcpiPsGetOpcodeName (WalkState->Opcode));

111     /* Examine the opcode */

113     switch (WalkState->Opcode)
114     {
115     case AML_NOTIFY_OP: /* Notify (NotifyObject, NotifyValue) */

117         /* The first operand is a namespace node */

119         Node = (ACPI_NAMESPACE_NODE *) Operand[0];

121         /* Second value is the notify value */

123         Value = (UINT32) Operand[1]->Integer.Value;
```

```

125      /* 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  * FUNCTION:      AcpiExOpcode_2A_1T_1R
268  *
269  * PARAMETERS:    WalkState          - Current walk state
270  *
271  * RETURN:        Status
272  *
273  * DESCRIPTION:    Execute opcode with two arguments, one target, and a return
274  *                  value.
275  *
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          */
354         Length = 0;
355         while ((Length < Operand[0]->Buffer.Length) &&
356             (Length < Operand[1]->Integer.Value) &&
357             (Operand[0]->Buffer.Pointer[Length]))
358         {
359             Length++;
360         }

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))
510     {
511         AcpiUtRemoveReference (ReturnDesc);
512         WalkState->ResultObj = NULL;
513     }

515     return_ACPI_STATUS (Status);
516 }

519 /*****
520 *
521 * FUNCTION:      AcpiExOpcode_2A_0T_1R
522 *
523 * PARAMETERS:    WalkState          - Current walk state
524 *
525 * RETURN:        Status
526 *
527 * DESCRIPTION:   Execute opcode with 2 arguments, no target, and a return value
528 *
529 *****/

531 ACPI_STATUS
532 AcpiExOpcode_2A_0T_1R (
533     ACPI_WALK_STATE      *WalkState)
534 {
535     ACPI_OPERAND_OBJECT  **Operand = &WalkState->Operands[0];
536     ACPI_OPERAND_OBJECT  *ReturnDesc = NULL;
537     ACPI_STATUS           Status = AE_OK;
538     BOOLEAN               LogicalResult = FALSE;

541     ACPI_FUNCTION_TRACE_STR (ExOpcode_2A_0T_1R,
542         AcpiPsGetOpcodeName (WalkState->Opcode));

545     /* Create the internal return object */

547     ReturnDesc = AcpiUtCreateInternalObject (ACPI_TYPE_INTEGER);
548     if (!ReturnDesc)
549     {
550         Status = AE_NO_MEMORY;
551         goto Cleanup;
552     }

554     /* Execute the Opcode */

556     if (WalkState->OpInfo->Flags & AML_LOGICAL_NUMERIC)
557     {
558         /* LogicalOp (Operand0, Operand1) */

560         Status = AcpiExDoLogicalNumericOp (WalkState->Opcode,
561             Operand[0]->Integer.Value, Operand[1]->Integer.Value,
562             &LogicalResult);
563         goto StoreLogicalResult;
564     }
565     else if (WalkState->OpInfo->Flags & AML_LOGICAL)
566     {
567         /* LogicalOp (Operand0, Operand1) */

569         Status = AcpiExDoLogicalOp (WalkState->Opcode, Operand[0],
570             Operand[1], &LogicalResult);
571         goto StoreLogicalResult;
572     }

574     switch (WalkState->Opcode)

```

```

575     {
576         case AML_ACQUIRE_OP:          /* Acquire (MutexObject, Timeout) */

578         Status = AcpiExAcquireMutex (Operand[1], Operand[0], WalkState);
579         if (Status == AE_TIME)
580         {
581             LogicalResult = TRUE;      /* TRUE = Acquire timed out */
582             Status = AE_OK;
583         }
584         break;

587         case AML_WAIT_OP:              /* Wait (EventObject, Timeout) */

589         Status = AcpiExSystemWaitEvent (Operand[1], Operand[0]);
590         if (Status == AE_TIME)
591         {
592             LogicalResult = TRUE;      /* TRUE, Wait timed out */
593             Status = AE_OK;
594         }
595         break;

597         default:

599         ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
600             WalkState->Opcode));
601         Status = AE_AML_BAD_OPCODE;
602         goto Cleanup;
603     }

606 StoreLogicalResult:
607     /*
608     * Set return value to according to LogicalResult. logical TRUE (all ones)
609     * Default is FALSE (zero)
610     */
611     if (LogicalResult)
612     {
613         ReturnDesc->Integer.Value = ACPI_UINT64_MAX;
614     }

616 Cleanup:

618     /* Delete return object on error */

620     if (ACPI_FAILURE (Status))
621     {
622         AcpiUtRemoveReference (ReturnDesc);
623     }

625     /* Save return object on success */

627     else
628     {
629         WalkState->ResultObj = ReturnDesc;
630     }

632     return_ACPI_STATUS (Status);
633 }

```



```

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  * FUNCTION:      AcpiExOpcode_3A_1T_1R
149  *
150  * PARAMETERS:    WalkState          - Current walk state
151  *
152  * RETURN:        Status
153  *
154  * DESCRIPTION:    Execute Triadic operator (3 operands)
155  *
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 }
```

new/usr/src/common/acpica/components/executer/exoparg6.c

1

\*\*\*\*\*

11135 Fri Jan 17 07:30:13 2014

new/usr/src/common/acpica/components/executer/exoparg6.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: exoparg6 - AML execution - opcodes with 6 arguments
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXOPARG6_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acinterp.h"
49 #include "acparser.h"
50 #include "amlcode.h"
51
52
53 #define _COMPONENT          ACPI_EXECUTER
54 #define ACPI_MODULE_NAME    ("exoparg6")
55
56
57 /*!
```

new/usr/src/common/acpica/components/executer/exoparg6.c

2

```
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 */
78
79 /* Local prototypes */
80
81 static BOOLEAN
82 AcpiExDoMatch (
83     UINT32                      MatchOp,
84     ACPI_OPERAND_OBJECT         *PackageObj,
85     ACPI_OPERAND_OBJECT         *MatchObj);
86
87
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 *****/
103
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;
112
113
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:
127
128         /* Always true */
129
130         break;
131
132     case MATCH_MEQ:
133
134         /*
135          * True if equal: (P[i] == M)
136          * Change to:      (M == P[i])
137          */
138         Status = AcpiExDoLogicalOp (AML_LEQUAL_OP, MatchObj, PackageObj,
139                                     &LogicalResult);
140         if (ACPI_FAILURE (Status))
141         {
142             return (FALSE);
143         }
144         break;
145
146     case MATCH_MLE:
147
148         /*
149          * True if less than or equal: (P[i] <= M) (P[i] NotGreater than M)
150          * Change to:                  (M >= P[i]) (M NotLess than P[i])
151          */
152         Status = AcpiExDoLogicalOp (AML_LLESS_OP, MatchObj, PackageObj,
153                                     &LogicalResult);
154         if (ACPI_FAILURE (Status))
155         {
156             return (FALSE);
157         }
158         LogicalResult = (BOOLEAN) !LogicalResult;
159         break;
160
161     case MATCH_MLT:
162
163         /*
164          * True if less than: (P[i] < M)
165          * Change to:          (M > P[i])
166          */
167         Status = AcpiExDoLogicalOp (AML_LGREATER_OP, MatchObj, PackageObj,
168                                     &LogicalResult);
169         if (ACPI_FAILURE (Status))
170         {
171             return (FALSE);
172         }
173         break;
174
175     case MATCH_MGE:
176
177         /*
178          * True if greater than or equal: (P[i] >= M) (P[i] NotLess than M)
179          * Change to:                    (M <= P[i]) (M NotGreater than P[i])
180          */
181         Status = AcpiExDoLogicalOp (AML_LGREATER_OP, MatchObj, PackageObj,
182                                     &LogicalResult);
183         if (ACPI_FAILURE (Status))
184         {
185             return (FALSE);
186         }
187         LogicalResult = (BOOLEAN) !LogicalResult;
188         break;

```

```

186     case MATCH_MGT:
187
188         /*
189          * True if greater than: (P[i] > M)
190          * Change to:            (M < P[i])
191          */
192         Status = AcpiExDoLogicalOp (AML_LLESS_OP, MatchObj, PackageObj,
193                                     &LogicalResult);
194         if (ACPI_FAILURE (Status))
195         {
196             return (FALSE);
197         }
198         break;
199
200     default:
201
202         /* Undefined */
203
204         return (FALSE);
205
206     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  *****/
221
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;
231
232     ACPI_FUNCTION_TRACE_STR (ExOpcode_6A_0T_1R,
233                               AcpiPsGetOpcodeName (WalkState->Opcode));
234
235     switch (WalkState->Opcode)
236     {
237     case AML_MATCH_OP:
238         /*
239          * Match (SearchPkg[0], MatchOp1[1], MatchObj1[2],
240          *          MatchOp2[3], MatchObj2[4], StartIndex[5])
241          */
242
243         /* Validate both Match Term Operators (MTR, MEQ, etc.) */
244
245         if ((Operand[1]->Integer.Value > MAX_MATCH_OPERATOR) ||
246             (Operand[3]->Integer.Value > MAX_MATCH_OPERATOR))
247         {

```

```

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;
275 }

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\_\_\_\_\_

new/usr/src/common/acpica/components/executer/exprep.c

1

\*\*\*\*\*

22243 Fri Jan 17 07:30:14 2014

new/usr/src/common/acpica/components/executer/exprep.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: exprep - ACPI AML (p-code) execution - field prep utilities
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXPREP_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acinterp.h"
49 #include "amlcode.h"
50 #include "acnamesp.h"
51 #include "acdispat.h"
52
53 #define _COMPONENT ACPI_EXECUTER
54 #define ACPI_MODULE_NAME ("exprep")
55
56 /* Local prototypes */
```

new/usr/src/common/acpica/components/executer/exprep.c

2

```
59 static UINT32
60 AcpiExDecodeFieldAccess (
61     ACPI_OPERAND_OBJECT *ObjDesc,
62     UINT8 FieldFlags,
63     UINT32 *ReturnByteAlignment);
64
65 #ifdef ACPI_UNDER_DEVELOPMENT
66
67 static UINT32
68 AcpiExGenerateAccess (
69     UINT32 FieldBitOffset,
70     UINT32 FieldBitLength,
71     UINT32 RegionLength);
72
73 /*****
74 *
75 * FUNCTION: AcpiExGenerateAccess
76 *
77 * PARAMETERS: FieldBitOffset - Start of field within parent region/buffer
78 *              FieldBitLength - Length of field in bits
79 *              RegionLength - Length of parent in bytes
80 *
81 * RETURN: Field granularity (8, 16, 32 or 64) and
82 *         ByteAlignment (1, 2, 3, or 4)
83 *
84 * DESCRIPTION: Generate an optimal access width for fields defined with the
85 *             AnyAcc keyword.
86 *
87 * NOTE: Need to have the RegionLength in order to check for boundary
88 *       conditions (end-of-region). However, the RegionLength is a deferred
89 *       operation. Therefore, to complete this implementation, the generation
90 *       of this access width must be deferred until the region length has
91 *       been evaluated.
92 *
93 *
94 *****/
95
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;
111
112     ACPI_FUNCTION_TRACE (ExGenerateAccess);
113
114     /* Round Field start offset and length to "minimal" byte boundaries */
115
116     FieldByteOffset = ACPI_DIV_8 (ACPI_ROUND_DOWN (FieldBitOffset, 8));
117     FieldByteEndOffset = ACPI_DIV_8 (ACPI_ROUND_UP (FieldBitLength +
118         FieldBitOffset, 8));
119     FieldByteLength = FieldByteEndOffset - FieldByteOffset;
120
121     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     }
186     else
187     {
188         ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD,
189         "AccessWidth %u end is NOT within region\n", AccessByteWidth));
190         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.
427 *
428 * DESCRIPTION: Construct an ACPI_OPERAND_OBJECT of type DefField and
429 *               connect it to the parent Node.
430 *
431 *****/
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)));
459         }

460         return ACPI_STATUS (AE_AML_OPERAND_TYPE);
461     }

462     /* Allocate a new field object */

464     ObjDesc = AcpiUtCreateInternalObject (Info->FieldType);
465     if (!ObjDesc)
466     {
467         return ACPI_STATUS (AE_NO_MEMORY);
468     }

470     /* Initialize areas of the object that are common to all fields */

472     ObjDesc->CommonField.Node = Info->FieldNode;
473     Status = AcpiExPrepCommonFieldObject (ObjDesc,
474                                           Info->FieldFlags, Info->Attribute,
475                                           Info->FieldBitPosition, Info->FieldBitLength);
476     if (ACPI_FAILURE (Status))
477     {
478         AcpiUtDeleteObjectDesc (ObjDesc);
479         return ACPI_STATUS (Status);
480     }
481 }

```

```

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 & AOBJ_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 < 255)
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);

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;

630      default:

632          /* No other types should get here */

634      break;
635  }

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));

644  ACPI_DEBUG_PRINT ((ACPI_DB_BFIELD, "Set NamedObj %p [%4.4s], ObjDesc %p\n",
645      Info->FieldNode, AcpiUtGetNodeName (Info->FieldNode), ObjDesc));

647  /* Remove local reference to the object */

649  AcpiUtRemoveReference (ObjDesc);
650  return_ACPI_STATUS (Status);
651 }
```

new/usr/src/common/acpica/components/executer/exregion.c

1

\*\*\*\*\*

17443 Fri Jan 17 07:30:14 2014

new/usr/src/common/acpica/components/executer/exregion.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: exregion - ACPI default OpRegion (address space) handlers
4 *
5 *****/
```

```
7 /*
```

```
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 *
```

new/usr/src/common/acpica/components/executer/exregion.c

2

```
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",
```

new/usr/src/common/acpica/components/executer/exregion.c

3

```
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     {
```

new/usr/src/common/acpica/components/executer/exregion.c

4

```
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

```

new/usr/src/common/acpica/components/executer/exresnte.c

1

```
*****
9366 Fri Jan 17 07:30:15 2014
new/usr/src/common/acpica/components/executer/exresnte.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOVs restructure
*****

1 /*****
2 *
3 * Module Name: exresnte - AML Interpreter object resolution
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXRESNTE_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acdispat.h"
49 #include "acinterp.h"
50 #include "acnamesp.h"

53 #define _COMPONENT ACPI_EXECUTER
54 ACPI_MODULE_NAME ("exresnte")
```

new/usr/src/common/acpica/components/executer/exresnte.c

2

```
57 /*****
58 *
59 * FUNCTION: AcpiExResolveNodeToValue
60 *
61 * PARAMETERS: ObjectPtr - Pointer to a location that contains
62 * a pointer to a NS node, and will receive a
63 * pointer to the resolved object.
64 * WalkState - Current state. Valid only if executing AML
65 * code. NULL if simply resolving an object
66 *
67 * RETURN: Status
68 *
69 * DESCRIPTION: Resolve a Namespace node to a valued object
70 *
71 * Note: for some of the data types, the pointer attached to the Node
72 * can be either a pointer to an actual internal object or a pointer into the
73 * AML stream itself. These types are currently:
74 *
75 * ACPI_TYPE_INTEGER
76 * ACPI_TYPE_STRING
77 * ACPI_TYPE_BUFFER
78 * ACPI_TYPE_MUTEX
79 * ACPI_TYPE_PACKAGE
80 *
81 *****/

83 ACPI_STATUS
84 AcpiExResolveNodeToValue (
85     ACPI_NAMESPACE_NODE **ObjectPtr,
86     ACPI_WALK_STATE *WalkState)
87 {
88     ACPI_STATUS Status = AE_OK;
89     ACPI_OPERAND_OBJECT *SourceDesc;
90     ACPI_OPERAND_OBJECT *ObjDesc = NULL;
91     ACPI_NAMESPACE_NODE *Node;
92     ACPI_OBJECT_TYPE EntryType;

96     ACPI_FUNCTION_TRACE (ExResolveNodeToValue);

99     /*
100     * The stack pointer points to a ACPI_NAMESPACE_NODE (Node). Get the
101     * object that is attached to the Node.
102     */
103     Node = *ObjectPtr;
104     SourceDesc = AcpiNsGetAttachedObject (Node);
105     EntryType = AcpiNsGetType ((ACPI_HANDLE) Node);

107     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Entry=%p SourceDesc=%p [%s]\n",
108         Node, SourceDesc, AcpiUtGetTypeName (EntryType)));

110     if ((EntryType == ACPI_TYPE_LOCAL_ALIAS) ||
111         (EntryType == ACPI_TYPE_LOCAL_METHOD_ALIAS))
112     {
113         /* There is always exactly one level of indirection */

115         Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Node->Object);
116         SourceDesc = AcpiNsGetAttachedObject (Node);
117         EntryType = AcpiNsGetType ((ACPI_HANDLE) Node);
118         *ObjectPtr = Node;
119     }

121     /*
122     * Several object types require no further processing:
```

```

123  * 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  }
134
135  if (!SourceDesc)
136  {
137      ACPI_ERROR ((AE_INFO, "No object attached to node [%4.4s] %p",
138                  Node->Name.Ascii, Node));
139      ACPI_ERROR ((AE_INFO, "No object attached to node %p",
140                  Node));
141      return_ACPI_STATUS (AE_AML_NO_OPERAND);
142  }
143
144  /*
145   * Action is based on the type of the Node, which indicates the type
146   * of the attached object or pointer
147   */
148  switch (EntryType)
149  {
150  case ACPI_TYPE_PACKAGE:
151
152      if (SourceDesc->Common.Type != ACPI_TYPE_PACKAGE)
153      {
154          ACPI_ERROR ((AE_INFO, "Object not a Package, type %s",
155                          AcpiUtGetObjectTypeName (SourceDesc)));
156          return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
157      }
158
159      Status = AcpiDsGetPackageArguments (SourceDesc);
160      if (ACPI_SUCCESS (Status))
161      {
162          /* Return an additional reference to the object */
163
164          ObjDesc = SourceDesc;
165          AcpiUtAddReference (ObjDesc);
166          break;
167      }
168
169  case ACPI_TYPE_BUFFER:
170
171      if (SourceDesc->Common.Type != ACPI_TYPE_BUFFER)
172      {
173          ACPI_ERROR ((AE_INFO, "Object not a Buffer, type %s",
174                          AcpiUtGetObjectTypeName (SourceDesc)));
175          return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
176      }
177
178      Status = AcpiDsGetBufferArguments (SourceDesc);
179      if (ACPI_SUCCESS (Status))
180      {
181          /* Return an additional reference to the object */
182
183          ObjDesc = SourceDesc;
184          AcpiUtAddReference (ObjDesc);
185          break;
186      }

```

```

186  case ACPI_TYPE_STRING:
187
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      }
194
195      /* Return an additional reference to the object */
196
197      ObjDesc = SourceDesc;
198      AcpiUtAddReference (ObjDesc);
199      break;
200
201  case ACPI_TYPE_INTEGER:
202
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      }
209
210      /* Return an additional reference to the object */
211
212      ObjDesc = SourceDesc;
213      AcpiUtAddReference (ObjDesc);
214      break;
215
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:
220
221      ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
222                          "FieldRead Node=%p SourceDesc=%p Type=%X\n",
223                          Node, SourceDesc, EntryType));
224
225      Status = AcpiExReadDataFromField (WalkState, SourceDesc, &ObjDesc);
226      break;
227
228      /* For these objects, just return the object attached to the Node */
229
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:
235
236      /* Return an additional reference to the object */
237
238      ObjDesc = SourceDesc;
239      AcpiUtAddReference (ObjDesc);
240      break;
241
242      /* TYPE_ANY is untyped, and thus there is no object associated with it */
243
244  case ACPI_TYPE_ANY:
245
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 }
```

new/usr/src/common/acpica/components/executer/exresolv.c

1

\*\*\*\*\*

17048 Fri Jan 17 07:30:15 2014

new/usr/src/common/acpica/components/executer/exresolv.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: exresolv - AML Interpreter object resolution
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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__
45
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
53 #define _COMPONENT ACPI_EXECUTER
54 #define ACPI_MODULE_NAME ("exresolv")
55
56 /* Local prototypes */
```

new/usr/src/common/acpica/components/executer/exresolv.c

2

```
59 static ACPI_STATUS
60 AcpiExResolveObjectToValue (
61     ACPI_OPERAND_OBJECT **StackPtr,
62     ACPI_WALK_STATE *WalkState);
63
64
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 *****/
79
80 ACPI_STATUS
81 AcpiExResolveToValue (
82     ACPI_OPERAND_OBJECT **StackPtr,
83     ACPI_WALK_STATE *WalkState)
84 {
85     ACPI_STATUS Status;
86
87     ACPI_FUNCTION_TRACE_PTR (ExResolveToValue, StackPtr);
88
89     if (!StackPtr || !*StackPtr)
90     {
91         ACPI_ERROR ((AE_INFO, "Internal - null pointer"));
92         return ACPI_STATUS (AE_AML_NO_OPERAND);
93     }
94
95     /*
96     * The entity pointed to by the StackPtr can be either
97     * 1) A valid ACPI_OPERAND_OBJECT, or
98     * 2) A ACPI_NAMESPACE_NODE (NamedObj)
99     */
100     if (ACPI_GET_DESCRIPTOR_TYPE (*StackPtr) == ACPI_DESC_TYPE_OPERAND)
101     {
102         Status = AcpiExResolveObjectToValue (StackPtr, WalkState);
103         if (ACPI_FAILURE (Status))
104         {
105             return ACPI_STATUS (Status);
106         }
107     }
108
109     if (!*StackPtr)
110     {
111         ACPI_ERROR ((AE_INFO, "Internal - null pointer"));
112         return ACPI_STATUS (AE_AML_NO_OPERAND);
113     }
114
115     /*
116     * Object on the stack may have changed if AcpiExResolveObjectToValue()
117     * was called (i.e., we can't use an _else_ here.)
118     */
119     if (ACPI_GET_DESCRIPTOR_TYPE (*StackPtr) == ACPI_DESC_TYPE_NAMED)
120     {
121         Status = AcpiExResolveNodeToValue (
122             *StackPtr, WalkState, &WalkState->Obj);
123     }
```



```

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 *                ReturnTypes     - 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      *ReturnTypes,
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       * This could of course in turn be another reference object.
467       */
468      ObjDesc = *(ObjDesc->Reference.Where);
469      if (!ObjDesc)
470      {
471          /* 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 }

```

new/usr/src/common/acpica/components/executer/exresop.c

1

\*\*\*\*\*

22214 Fri Jan 17 07:30:16 2014

new/usr/src/common/acpica/components/executer/exresop.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: exresop - AML Interpreter operand/object resolution
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #define __EXRESOP_C__
46
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "amlcode.h"
50 #include "acparser.h"
51 #include "acinterp.h"
52 #include "acnamesp.h"
53
54 #define _COMPONENT ACPI_EXECUTER
55 #define ACPI_MODULE_NAME ("exresop")
56
57 /* Local prototypes */
```

new/usr/src/common/acpica/components/executer/exresop.c

2

```
59 static ACPI_STATUS
60 AcpiExCheckObjectType (
61     ACPI_OBJECT_TYPE      TypeNeeded,
62     ACPI_OBJECT_TYPE      ThisType,
63     void                   *Object);
64
65
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 *****/
79
80 static ACPI_STATUS
81 AcpiExCheckObjectType (
82     ACPI_OBJECT_TYPE      TypeNeeded,
83     ACPI_OBJECT_TYPE      ThisType,
84     void                   *Object)
85 {
86     ACPI_FUNCTION_ENTRY ();
87
88     if (TypeNeeded == ACPI_TYPE_ANY)
89     {
90         /* All types OK, so we don't perform any typechecks */
91
92         return (AE_OK);
93     }
94
95     if (TypeNeeded == ACPI_TYPE_LOCAL_REFERENCE)
96     {
97         /*
98          * Allow the AML "Constant" opcodes (Zero, One, etc.) to be reference
99          * objects and thus allow them to be targets. (As per the ACPI
100          * specification, a store to a constant is a noop.)
101          */
102         if ((ThisType == ACPI_TYPE_INTEGER) &&
103             (((ACPI_OPERAND_OBJECT *) Object)->Common.Flags & AOPOBJ_AML_CONSTAN
104             {
105                 return (AE_OK);
106             }
107         }
108     }
109
110     if (TypeNeeded != ThisType)
111     {
112         ACPI_ERROR ((AE_INFO,
113                     "Needed type [%s], found [%s] %p",
114                     AcpiUtGetTypeName (TypeNeeded),
115                     AcpiUtGetTypeName (ThisType), Object));
116
117         return (AE_AML_OPERAND_TYPE);
118     }
119
120     return (AE_OK);
121 }
```

```

124 /*****
125 *
126 * FUNCTION:      AcpiExResolveOperands
127 *
128 * PARAMETERS:    Opcode           - Opcode being interpreted
129 *                StackPtr         - Pointer to the operand stack to be
130 *                                resolved
131 *                WalkState        - Current state
132 *
133 * RETURN:        Status
134 *
135 * DESCRIPTION:    Convert multiple input operands to the types required by the
136 *                target operator.
137 *
138 *                Each 5-bit group in ArgTypes represents one required
139 *                operand and indicates the required Type. The corresponding operand
140 *                will be converted to the required type if possible, otherwise we
141 *                abort with an exception.
142 *
143 *****/
144
145 ACPI_STATUS
146 AcpiExResolveOperands (
147     UINT16          Opcode,
148     ACPI_OPERAND_OBJECT **StackPtr,
149     ACPI_WALK_STATE *WalkState)
150 {
151     ACPI_OPERAND_OBJECT *ObjDesc;
152     ACPI_STATUS          Status = AE_OK;
153     UINT8                ObjectType;
154     UINT32               ArgTypes;
155     const ACPI_OPCODE_INFO *OpInfo;
156     UINT32               ThisArgType;
157     ACPI_OBJECT_TYPE      TypeNeeded;
158     UINT16               TargetOp = 0;
159
160     ACPI_FUNCTION_TRACE_U32 (ExResolveOperands, Opcode);
161
162     OpInfo = AcpiPsGetOpcodeInfo (Opcode);
163     if (OpInfo->Class == AML_CLASS_UNKNOWN)
164     {
165         return ACPI_STATUS (AE_AML_BAD_OPCODE);
166     }
167
168     ArgTypes = OpInfo->RuntimeArgs;
169     if (ArgTypes == ARG_INVALID_OPCODE)
170     {
171         ACPI_ERROR ((AE_INFO, "Unknown AML opcode 0x%X",
172             Opcode));
173     }
174
175     return ACPI_STATUS (AE_AML_INTERNAL);
176 }
177
178 ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
179     "Opcode %X [%s] RequiredOperandTypes=%8.8X\n",
180     Opcode, OpInfo->Name, ArgTypes));
181
182 /*
183 * Normal exit is with (ArgTypes == 0) at end of argument list.
184 * Function will return an exception from within the loop upon
185 * finding an entry which is not (or cannot be converted
186 * to) the required type; if stack underflows; or upon
187 * finding a NULL stack entry (which should not happen).
188 */

```

```

190 while (GET_CURRENT_ARG_TYPE (ArgTypes))
191 {
192     if (!StackPtr || !*StackPtr)
193     {
194         ACPI_ERROR ((AE_INFO, "Null stack entry at %p",
195             StackPtr));
196     }
197     return ACPI_STATUS (AE_AML_INTERNAL);
198 }
199
200 /* Extract useful items */
201
202 ObjDesc = *StackPtr;
203
204 /* Decode the descriptor type */
205
206 switch (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc))
207 {
208     case ACPI_DESC_TYPE_NAMED:
209
210         /* Namespace Node */
211
212         ObjectType = ((ACPI_NAMESPACE_NODE *) ObjDesc)->Type;
213
214         /*
215          * Resolve an alias object. The construction of these objects
216          * guarantees that there is only one level of alias indirection;
217          * thus, the attached object is always the aliased namespace node
218          */
219         if (ObjectType == ACPI_TYPE_LOCAL_ALIAS)
220         {
221             ObjDesc = AcpiNsGetAttachedObject ((ACPI_NAMESPACE_NODE *) ObjDesc);
222             *StackPtr = ObjDesc;
223             ObjectType = ((ACPI_NAMESPACE_NODE *) ObjDesc)->Type;
224         }
225         break;
226
227     case ACPI_DESC_TYPE_OPERAND:
228
229         /* ACPI internal object */
230
231         ObjectType = ObjDesc->Common.Type;
232
233         /* Check for bad ACPI_OBJECT_TYPE */
234
235         if (!AcpiUtValidObjectType (ObjectType))
236         {
237             ACPI_ERROR ((AE_INFO,
238                 "Bad operand object type [0x%X]", ObjectType));
239         }
240         return ACPI_STATUS (AE_AML_OPERAND_TYPE);
241     }
242
243     if (ObjectType == (UINT8) ACPI_TYPE_LOCAL_REFERENCE)
244     {
245         /* Validate the Reference */
246
247         switch (ObjDesc->Reference.Class)
248         {
249             case ACPI_REFCLASS_DEBUG:
250
251                 TargetOp = AML_DEBUG_OP;
252
253                 /*lint -fallthrough */

```

```

255     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 ARG1_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 ARG1_REFERENCE: /* References: */
319     case ARG1_INTEGER_REF:

```

```

320     case ARG1_OBJECT_REF:
321     case ARG1_DEVICE_REF:
322     case ARG1_TARGETREF: /* Allows implicit conversion rules before stor
323     case ARG1_FIXED_TARGET: /* No implicit conversion before store to targe
324     case ARG1_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 ARG1_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 ARG1_MUTEX:

388         /* Need an operand of type ACPI_TYPE_MUTEX */

390         TypeNeeded = ACPI_TYPE_MUTEX;
391         break;

393     case ARG1_EVENT:

395         /* Need an operand of type ACPI_TYPE_EVENT */

397         TypeNeeded = ACPI_TYPE_EVENT;
398         break;

400     case ARG1_PACKAGE:    /* Package */

402         /* Need an operand of type ACPI_TYPE_PACKAGE */

404         TypeNeeded = ACPI_TYPE_PACKAGE;
405         break;

407     case ARG1_ANYTYPE:

409         /* Any operand type will do */

411         TypeNeeded = ACPI_TYPE_ANY;
412         break;

414     case ARG1_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 ARG1_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 ARG1_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 ARG1_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 ARG1_COMPUTEDATA:

```



```

510      /* Need an operand of type INTEGER, STRING or BUFFER */
511
512      switch (ObjDesc->Common.Type)
513      {
514      case ACPI_TYPE_INTEGER:
515      case ACPI_TYPE_STRING:
516      case ACPI_TYPE_BUFFER:
517
518          /* Valid operand */
519          break;
520
521      default:
522          ACPI_ERROR ((AE_INFO,
523                      "Needed [Integer/String/Buffer], found [%s] %p",
524                      AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));
525
526          return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
527      }
528      goto NextOperand;
529
530  case ARG1_BUFFER_OR_STRING:
531
532      /* Need an operand of type STRING or BUFFER */
533
534      switch (ObjDesc->Common.Type)
535      {
536      case ACPI_TYPE_STRING:
537      case ACPI_TYPE_BUFFER:
538
539          /* Valid operand */
540          break;
541
542      case ACPI_TYPE_INTEGER:
543
544          /* Highest priority conversion is to type Buffer */
545
546          Status = AcpiExConvertToBuffer (ObjDesc, StackPtr);
547          if (ACPI_FAILURE (Status))
548          {
549              return_ACPI_STATUS (Status);
550          }
551
552          if (ObjDesc != *StackPtr)
553          {
554              AcpiUtRemoveReference (ObjDesc);
555          }
556          break;
557
558      default:
559          ACPI_ERROR ((AE_INFO,
560                      "Needed [Integer/String/Buffer], found [%s] %p",
561                      AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));
562
563          return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
564      }
565      goto NextOperand;
566
567  case ARG1_DATAOBJECT:
568      /*
569       * ARG1_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:
581
582          /* Valid operand */
583          break;
584
585      default:
586
587          ACPI_ERROR ((AE_INFO,
588                      "Needed [Buffer/String/Package/Reference], found [%s] %p",
589                      AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));
590
591          return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
592      }
593      goto NextOperand;
594
595  case ARG1_COMPLEXOBJ:
596
597      /* Need a buffer or package or (ACPI 2.0) String */
598
599      switch (ObjDesc->Common.Type)
600      {
601      case ACPI_TYPE_PACKAGE:
602      case ACPI_TYPE_STRING:
603      case ACPI_TYPE_BUFFER:
604
605          /* Valid operand */
606          break;
607
608      default:
609
610          ACPI_ERROR ((AE_INFO,
611                      "Needed [Buffer/String/Package], found [%s] %p",
612                      AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));
613
614          return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
615      }
616      goto NextOperand;
617
618  case ARG1_REGION_OR_BUFFER: /* Used by Load() only */
619
620      /* Need an operand of type REGION or a BUFFER (which could be a reso
621
622      switch (ObjDesc->Common.Type)
623      {
624      case ACPI_TYPE_BUFFER:
625      case ACPI_TYPE_REGION:
626
627          /* Valid operand */
628          break;
629
630      default:
631
632          ACPI_ERROR ((AE_INFO,
633                      "Needed [Region/Buffer], found [%s] %p",
634                      AcpiUtGetObjectTypeName (ObjDesc), ObjDesc));
635
636          return_ACPI_STATUS (AE_AML_OPERAND_TYPE);
637      }

```

```

638         goto NextOperand;

640     case ARG1_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 ARG1 (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 }

```

new/usr/src/common/acpica/components/executer/exstore.c

1

```
*****
18686 Fri Jan 17 07:30:16 2014
new/usr/src/common/acpica/components/executer/exstore.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: exstore - AML Interpreter object store support
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXSTORE_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acdispat.h"
49 #include "acinterp.h"
50 #include "amlcode.h"
51 #include "acnamesp.h"
52
53 #define _COMPONENT ACPI_EXECUTER
54 #define ACPI_MODULE_NAME ("exstore")
55
56 /* Local prototypes */
```

new/usr/src/common/acpica/components/executer/exstore.c

2

```
59 static ACPI_STATUS
60 AcpiExStoreObjectToIndex (
61     ACPI_OPERAND_OBJECT *ValDesc,
62     ACPI_OPERAND_OBJECT *DestDesc,
63     ACPI_WALK_STATE *WalkState);
64
65 static ACPI_STATUS
66 AcpiExStoreDirectToNode (
67     ACPI_OPERAND_OBJECT *SourceDesc,
68     ACPI_NAMESPACE_NODE *Node,
69     ACPI_WALK_STATE *WalkState);
70
71 /*****
72 *
73 * FUNCTION: AcpiExStore
74 *
75 * PARAMETERS: *SourceDesc - Value to be stored
76 *              *DestDesc - Where to store it. Must be an NS node
77 *                      or ACPI_OPERAND_OBJECT of type
78 *                      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 *****/
91
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;
100
101     ACPI_FUNCTION_TRACE_PTR (ExStore, DestDesc);
102
103     /* Validate parameters */
104
105     if (!SourceDesc || !DestDesc)
106     {
107         ACPI_ERROR ((AE_INFO, "Null parameter"));
108         return ACPI_STATUS (AE_AML_NO_OPERAND);
109     }
110
111     /* DestDesc can be either a namespace node or an ACPI object */
112
113     if (ACPI_GET_DESCRIPTOR_TYPE (DestDesc) == ACPI_DESC_TYPE_NAMED)
114     {
115         /*
116          * Dest is a namespace node,
117          * Storing an object into a Named node.
118          */
119         Status = AcpiExStoreObjectToNode (SourceDesc,
120             (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 & APOBJ_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     *ObjDesc;
241     *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);

```

```
new/usr/src/common/acpica/components/executer/exstore.c
```

[illegible]

```

476     if (ACPI_FAILURE (Status))
477     {
478         return ACPI_STATUS (Status);
479     }

481     if (NewDesc != TargetDesc)
482     {
483         /*
484          * Store the new NewDesc as the new value of the Name, and set
485          * the Name's type to that of the value being stored in it.
486          * SourceDesc reference count is incremented by AttachObject.
487          *
488          * Note: This may change the type of the node if an explicit
489          * store has been performed such that the node/object type
490          * has been changed.
491          * Note: This may change the type of the node if an explicit store
492          * has been performed such that the node/object type has been
493          * changed.
494          */
495         Status = AcpiNsAttachObject (Node, NewDesc,
496                                     NewDesc->Common.Type);
497         Status = AcpiNsAttachObject (Node, NewDesc, NewDesc->Common.Type);

498         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
499                             "Store %s into %s via Convert/Attach\n",
500                             AcpiUtGetObjectTypeName (SourceDesc),
501                             AcpiUtGetObjectTypeName (NewDesc)));
502     }
503     break;

504 case ACPI_TYPE_BUFFER_FIELD:
505 case ACPI_TYPE_LOCAL_REGION_FIELD:
506 case ACPI_TYPE_LOCAL_BANK_FIELD:
507 case ACPI_TYPE_LOCAL_INDEX_FIELD:
508     /*
509      * For all fields, always write the source data to the target
510      * field. Any required implicit source operand conversion is
511      * performed in the function below as necessary. Note, field
512      * objects must retain their original type permanently.
513      */
514     Status = AcpiExWriteDataToField (SourceDesc, TargetDesc,
515                                     &WalkState->ResultObj);
516     break;

517 default:
518     /*
519      * No conversions for all other types. Directly store a copy of
520      * the source object. This is the ACPI spec-defined behavior for
521      * the CopyObject operator.
522      *
523      * NOTE: For the Store operator, this is a departure from the
524      * ACPI spec, which states "If conversion is impossible, abort
525      * the running control method". Instead, this code implements
526      * "If conversion is impossible, treat the Store operation as
527      * a CopyObject".
528      */
529     Status = AcpiExStoreDirectToNode (SourceDesc, Node,
530                                     WalkState);
531     break;

533     return ACPI_STATUS (Status);
534 }

```

```

537 /*****

```

```

538 *
539 * FUNCTION:      AcpiExStoreDirectToNode
540 *
541 * PARAMETERS:    SourceDesc          - Value to be stored
542 *               Node                  - Named object to receive the value
543 *               WalkState              - Current walk state
544 *
545 * RETURN:        Status
546 *
547 * DESCRIPTION:    "Store" an object directly to a node. This involves a copy
548 *                 and an attach.
549 *
550 *****/

552 static ACPI_STATUS
553 AcpiExStoreDirectToNode (
554     ACPI_OPERAND_OBJECT *SourceDesc,
555     ACPI_NAMESPACE_NODE *Node,
556     ACPI_WALK_STATE *WalkState)
557 {
558     ACPI_STATUS Status;
559     ACPI_OPERAND_OBJECT *NewDesc;

562     ACPI_FUNCTION_TRACE (ExStoreDirectToNode);

565     ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
566                         "Storing [%s] (%p) directly into node [%s] (%p)"
567                         " with no implicit conversion\n",
568                         AcpiUtGetObjectTypeName (SourceDesc), SourceDesc,
569                         AcpiUtGetTypeName (Node->Type), Node));
570     "Storing %s (%p) directly into node (%p) with no implicit conversion
571     AcpiUtGetObjectTypeName (SourceDesc), SourceDesc, Node));

572     /* Copy the source object to a new object */
573     /* No conversions for all other types. Just attach the source object */

574     Status = AcpiUtCopyIobjectToIobject (SourceDesc, &NewDesc, WalkState);
575     if (ACPI_FAILURE (Status))
576     {
577         return ACPI_STATUS (Status);
578         Status = AcpiNsAttachObject (Node, SourceDesc,
579                                     SourceDesc->Common.Type);
580         break;
581     }

582     /* Attach the new object to the node */

583     Status = AcpiNsAttachObject (Node, NewDesc, NewDesc->Common.Type);
584     AcpiUtRemoveReference (NewDesc);
585     return ACPI_STATUS (Status);
586 }

```

new/usr/src/common/acpica/components/executer/exstoren.c

1

\*\*\*\*\*

10281 Fri Jan 17 07:30:16 2014

new/usr/src/common/acpica/components/executer/exstoren.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOV's restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: exstoren - AML Interpreter object store support,
4 *               Store to Node (namespace object)
5 *
6 *****/
```

```
8 /*
9 * Copyright (C) 2000 - 2014, 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 * 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 MERCHANTABILITY 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 __EXSTOREN_C__
```

```
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acinterp.h"
50 #include "amlcode.h"
```

```
53 #define _COMPONENT      ACPI_EXECUTER
54 #define ACPI_MODULE_NAME    ("exstoren")
```

```
57 /*****
```

new/usr/src/common/acpica/components/executer/exstoren.c

2

```
58 *
59 * FUNCTION:      AcpiExResolveObject
60 *
61 * PARAMETERS:    SourceDescPtr      - Pointer to the source object
62 *               TargetType          - Current type of the target
63 *               WalkState            - Current walk state
64 *
65 * RETURN:        Status, resolved object in SourceDescPtr.
66 *
67 * DESCRIPTION:    Resolve an object. If the object is a reference, dereference
68 *               it and return the actual object in the SourceDescPtr.
69 *
70 *****/
```

```
72 ACPI_STATUS
73 AcpiExResolveObject (
74     ACPI_OPERAND_OBJECT **SourceDescPtr,
75     ACPI_OBJECT_TYPE TargetType,
76     ACPI_WALK_STATE *WalkState)
77 {
78     ACPI_OPERAND_OBJECT *SourceDesc = *SourceDescPtr;
79     ACPI_STATUS Status = AE_OK;
```

```
82     ACPI_FUNCTION_TRACE (ExResolveObject);
```

```
85     /* Ensure we have a Target that can be stored to */
```

```
87     switch (TargetType)
88     {
89     case ACPI_TYPE_BUFFER_FIELD:
90     case ACPI_TYPE_LOCAL_REGION_FIELD:
91     case ACPI_TYPE_LOCAL_BANK_FIELD:
92     case ACPI_TYPE_LOCAL_INDEX_FIELD:
93         /*
94          * These cases all require only Integers or values that
95          * can be converted to Integers (Strings or Buffers)
96          */
```

```
97     case ACPI_TYPE_INTEGER:
98     case ACPI_TYPE_STRING:
99     case ACPI_TYPE_BUFFER:
```

```
100     /*
101     * Stores into a Field/Region or into a Integer/Buffer/String
102     * are all essentially the same. This case handles the
103     * "interchangeable" types Integer, String, and Buffer.
104     */
```

```
105     if (SourceDesc->Common.Type == ACPI_TYPE_LOCAL_REFERENCE)
106     {
107         /* Resolve a reference object first */
```

```
109         Status = AcpiExResolveToValue (SourceDescPtr, WalkState);
110         if (ACPI_FAILURE (Status))
111         {
112             break;
113         }
114     }
```

```
116     /* For CopyObject, no further validation necessary */
```

```
118     if (WalkState->Opcode == AML_COPY_OP)
119     {
120         break;
121     }
```



```

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  *
186  *      When storing into an object the data is converted to the
187  *      target object type then stored in the object. This means
188  *      that the target object type (for an initialized target) will
189  *      not be changed by a store operation.
190  *
191  *      This module allows destination types of Number, String,
192  *      Buffer, and Package.
193  *
194  *      Assumes parameters are already validated. NOTE: SourceDesc
195  *      resolution (from a reference object) must be performed by
196  *      the caller if necessary.
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:
261
262         DestDesc->Integer.Value = ActualSrcDesc->Integer.Value;
263
264         /* Truncate value if we are executing from a 32-bit ACPI table */
265
266         (void) AcpiExTruncateFor32bitTable (DestDesc);
267         break;
268
269     case ACPI_TYPE_STRING:
270
271         Status = AcpiExStoreStringToString (ActualSrcDesc, DestDesc);
272         break;
273
274     case ACPI_TYPE_BUFFER:
275
276         Status = AcpiExStoreBufferToBuffer (ActualSrcDesc, DestDesc);
277         break;
278
279     case ACPI_TYPE_PACKAGE:
280
281         Status = AcpiUtCopyIobjectToIobject (ActualSrcDesc, &DestDesc,
282         WalkState);
283         break;
284
285     default:
286         /*
287          * All other types come here.
288          */
289         ACPI_WARNING ((AE_INFO, "Store into type %s not implemented",
290         AcpiUtGetObjectTypeName (DestDesc)));
291
292         Status = AE_NOT_IMPLEMENTED;
293         break;
294 }

296 if (ActualSrcDesc != SourceDesc)
297 {
298     /* Delete the intermediate (temporary) source object */
299
300     AcpiUtRemoveReference (ActualSrcDesc);
301 }

303 *NewDesc = DestDesc;
304 return_ACPI_STATUS (Status);
305 }
```

new/usr/src/common/acpica/components/executer/exstorob.c

1

\*\*\*\*\*

7845 Fri Jan 17 07:30:17 2014  
new/usr/src/common/acpica/components/executer/exstorob.c  
update to acpica-unix2-20140114

acpica-unix2-20130823  
PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: exstorob - AML Interpreter object store support, store to object
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXSTOROB_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acinterp.h"
49
50 #define _COMPONENT ACPI_EXECUTER
51 #define ACPI_MODULE_NAME ("exstorob")
52
53 /*****
54 *
55 * FUNCTION: AcpiExStoreBufferToBuffer
56 */
```

new/usr/src/common/acpica/components/executer/exstorob.c

2

```
58 *
59 * PARAMETERS: SourceDesc - Source object to copy
60 *              TargetDesc - Destination object of the copy
61 *
62 * RETURN: Status
63 *
64 * DESCRIPTION: Copy a buffer object to another buffer object.
65 *
66 *****/
67
68 ACPI_STATUS
69 AcpiExStoreBufferToBuffer (
70     ACPI_OPERAND_OBJECT *SourceDesc,
71     ACPI_OPERAND_OBJECT *TargetDesc)
72 {
73     UINT32 Length;
74     UINT8 *Buffer;
75
76     ACPI_FUNCTION_TRACE_PTR (ExStoreBufferToBuffer, SourceDesc);
77
78     /* If Source and Target are the same, just return */
79
80     if (SourceDesc == TargetDesc)
81     {
82         return ACPI_STATUS (AE_OK);
83     }
84
85     /* We know that SourceDesc is a buffer by now */
86
87     Buffer = ACPI_CAST_PTR (UINT8, SourceDesc->Buffer.Pointer);
88     Length = SourceDesc->Buffer.Length;
89
90     /*
91     * If target is a buffer of length zero or is a static buffer,
92     * allocate a new buffer of the proper length
93     */
94     if ((TargetDesc->Buffer.Length == 0) ||
95         (TargetDesc->Common.Flags & AOPOBJ_STATIC_POINTER))
96     {
97         TargetDesc->Buffer.Pointer = ACPI_ALLOCATE (Length);
98         if (!TargetDesc->Buffer.Pointer)
99         {
100             return ACPI_STATUS (AE_NO_MEMORY);
101         }
102         TargetDesc->Buffer.Length = Length;
103     }
104
105     /* Copy source buffer to target buffer */
106
107     if (Length <= TargetDesc->Buffer.Length)
108     {
109         /* Clear existing buffer and copy in the new one */
110
111         ACPI_MEMSET (TargetDesc->Buffer.Pointer, 0, TargetDesc->Buffer.Length);
112         ACPI_MEMCPY (TargetDesc->Buffer.Pointer, Buffer, Length);
113     }
114
115 #ifndef ACPI_OBSOLETE_BEHAVIOR
116     /*
117     * NOTE: ACPI versions up to 3.0 specified that the buffer must be
118     * truncated if the string is smaller than the buffer. However, "other"
119     * implementations of ACPI never did this and thus became the defacto
120     * standard. ACPI 3.0A changes this behavior such that the buffer
121     * is no longer truncated.
122     */
123 #endif
```

```
124      */
125
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 */
135
136          TargetDesc->Buffer.Length = Length;
137      }
138 #endif
139     }
140     else
141     {
142         /* Truncate the source, copy only what will fit */
143
144         ACPI_MEMCPY (TargetDesc->Buffer.Pointer, Buffer,
145                     TargetDesc->Buffer.Length);
146
147         ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
148                          "Truncating source buffer from %X to %X\n",
149                          Length, TargetDesc->Buffer.Length));
150     }
151
152     /* Copy flags */
153
154     TargetDesc->Buffer.Flags = SourceDesc->Buffer.Flags;
155     TargetDesc->Common.Flags &= ~AOPOBJ_STATIC_POINTER;
156     return_ACPI_STATUS (AE_OK);
157 }
158
159 unchanged portion omitted
```

new/usr/src/common/acpica/components/executer/exsystem.c

1

\*\*\*\*\*

9896 Fri Jan 17 07:30:17 2014  
new/usr/src/common/acpica/components/executer/exsystem.c  
update to acpica-unix2-20140114  
acpica-unix2-20130823  
PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: exsystem - Interface to OS services
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXSYSTEM_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acinterp.h"
49
50 #define _COMPONENT ACPI_EXECUTER
51 #define ACPI_MODULE_NAME ("exsystem")
52
53
54 /*****
55 *
56 * FUNCTION: AcpiExSystemWaitSemaphore
57 */
```

new/usr/src/common/acpica/components/executer/exsystem.c

2

```
58 * PARAMETERS: Semaphore - Semaphore to wait on
59 *              Timeout   - Max time to wait
60 *
61 * RETURN:      Status
62 *
63 * DESCRIPTION: Implements a semaphore wait with a check to see if the
64 * semaphore is available immediately. If it is not, the
65 * interpreter is released before waiting.
66 *
67 *****/
68
69 ACPI_STATUS
70 AcpiExSystemWaitSemaphore (
71     ACPI_SEMAPHORE Semaphore,
72     UINT16 Timeout)
73 {
74     ACPI_STATUS Status;
75
76     /* We must wait, so unlock the interpreter */
77     AcpiFunctionTrace (ExSystemWaitSemaphore);
78
79     Status = AcpiOsWaitSemaphore (Semaphore, 1, ACPI_DO_NOT_WAIT);
80     if (ACPI_SUCCESS (Status))
81     {
82         return ACPI_STATUS (Status);
83     }
84
85     if (Status == AE_TIME)
86     {
87         /* We must wait, so unlock the interpreter */
88         AcpiExRelinquishInterpreter ();
89
90         Status = AcpiOsWaitSemaphore (Semaphore, 1, Timeout);
91
92         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
93             "*** Thread awake after blocking, %s\n",
94             AcpiFormatException (Status)));
95
96         /* Reacquire the interpreter */
97         AcpiExReacquireInterpreter ();
98     }
99
100     return ACPI_STATUS (Status);
101 }
102
103 }
104 }
105
106 _____
107 unchanged_portion_omitted
108 _____
```

new/usr/src/common/acpica/components/executer/exutils.c

1

\*\*\*\*\*

15228 Fri Jan 17 07:30:18 2014

new/usr/src/common/acpica/components/executer/exutils.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: exutils - interpreter/scanner utilities
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXUTILS_C__
45
46 /*
47 * DEFINE_AML_GLOBALS is tested in amlcode.h
48 * to determine whether certain global names should be "defined" or only
49 * "declared" in the current compilation. This enhances maintainability
50 * by enabling a single header file to embody all knowledge of the names
51 * in question.
52 *
53 * Exactly one module of any executable should #define DEFINE_GLOBALS
54 * before #including the header files which use this convention. The
55 * names in question will be defined and initialized in that module,
56 * and declared as extern in all other modules which #include those
57 * header files.
```

new/usr/src/common/acpica/components/executer/exutils.c

2

```
58 */
59
60 #define DEFINE_AML_GLOBALS
61
62 #include "acpi.h"
63 #include "accommon.h"
64 #include "acinterp.h"
65 #include "amlcode.h"
66
67 #define _COMPONENT ACPI_EXECUTER
68 ACPI_MODULE_NAME ("exutils")
69
70 /* Local prototypes */
71
72 static UINT32
73 AcpiExDigitsNeeded (
74     UINT64      Value,
75     UINT32      Base);
76
77 #ifndef ACPI_NO_METHOD_EXECUTION
78 /*****
79 *
80 * FUNCTION:      AcpiExEnterInterpreter
81 *
82 * PARAMETERS:    None
83 *
84 * RETURN:        None
85 *
86 * DESCRIPTION:   Enter the interpreter execution region. Failure to enter
87 *                the interpreter region is a fatal system error. Used in
88 *                conjunction with ExitInterpreter.
89 *
90 * *****/
91
92 void
93 AcpiExEnterInterpreter (
94     void)
95 {
96     ACPI_STATUS      Status;
97
98     ACPI_FUNCTION_TRACE (ExEnterInterpreter);
99
100     Status = AcpiUtAcquireMutex (ACPI_MTX_INTERPRETER);
101     if (ACPI_FAILURE (Status))
102     {
103         ACPI_ERROR ((AE_INFO, "Could not acquire AML Interpreter mutex"));
104     }
105
106     return_VOID;
107 }
108
109
110
111
112 /*****
113 *
114 * FUNCTION:      AcpiExReacquireInterpreter
115 *
116 * PARAMETERS:    None
117 *
118 * RETURN:        None
119 *
120 * DESCRIPTION:   Reacquire the interpreter execution region from within the
121 *                interpreter code. Failure to enter the interpreter region is a
122 *                fatal system error. Used in conjunction with
123 * *****/
```

```

124 *          fatal system error. Used in conjunction with
124 *          RelinquishInterpreter
125 *
126 *****/
128 void
129 AcpiExReacquireInterpreter (
130     void)
131 {
132     ACPI_FUNCTION_TRACE (ExReacquireInterpreter);

135     /*
136      * If the global serialized flag is set, do not release the interpreter,
137      * since it was not actually released by AcpiExRelinquishInterpreter.
138      * This forces the interpreter to be single threaded.
139      */
140     if (!AcpiGbl_AllMethodsSerialized)
141     {
142         AcpiExEnterInterpreter ();
143     }

145     return_VOID;
146 }
    unchanged_portion_omitted_

226 /*****
227 *
228 * FUNCTION:      AcpiExTruncateFor32bitTable
229 *
230 * PARAMETERS:    ObjDesc          - Object to be truncated
231 *
232 * RETURN:        TRUE if a truncation was performed, FALSE otherwise.
233 * RETURN:        none
234 * DESCRIPTION:    Truncate an ACPI Integer to 32 bits if the execution mode is
235 *                 32-bit, as determined by the revision of the DSDT.
236 *
237 *****/
239 BOOLEAN
240 void
241 AcpiExTruncateFor32bitTable (
242     ACPI_OPERAND_OBJECT *ObjDesc)
243 {
244     ACPI_FUNCTION_ENTRY ();

247     /*
248      * Object must be a valid number and we must be executing
249      * a control method. Object could be NS node for AML_INT_NAMEPATH_OP.
250      * a control method. NS node could be there for AML_INT_NAMEPATH_OP.
251      */
252     if ((!ObjDesc) ||
253         (ACPI_GET_DESCRIPTOR_TYPE (ObjDesc) != ACPI_DESC_TYPE_OPERAND) ||
254         (ObjDesc->Common.Type != ACPI_TYPE_INTEGER))
255     {
256         return (FALSE);
257     }

258     if ((AcpiGbl_IntegerByteWidth == 4) &&
259         (ObjDesc->Integer.Value > (UINT64) ACPI_UINT32_MAX))
260     {
261         if (AcpiGbl_IntegerByteWidth == 4)

```

```

260     {
261         /*
262          * We are executing in a 32-bit ACPI table.
263          * We are running a method that exists in a 32-bit ACPI table.
264          * Truncate the value to 32 bits by zeroing out the upper 32-bit field
265          */
266         ObjDesc->Integer.Value &= (UINT64) ACPI_UINT32_MAX;
267         return (TRUE);
268     }

269     return (FALSE);
270 }
    unchanged_portion_omitted_

506 /*****
507 *
508 * FUNCTION:      AcpiIsValidSpaceId
509 *
510 * PARAMETERS:    SpaceId          - ID to be validated
511 *
512 * RETURN:        TRUE if valid/supported ID.
513 *
514 * DESCRIPTION:    Validate an operation region SpaceID.
515 *
516 *****/
518 BOOLEAN
519 AcpiIsValidSpaceId (
520     UINT8          SpaceId)
521 {
522     if ((SpaceId >= ACPI_NUM_PREDEFINED_REGIONS) &&
523         (SpaceId < ACPI_USER_REGION_BEGIN) &&
524         (SpaceId != ACPI_ADR_SPACE_DATA_TABLE) &&
525         (SpaceId != ACPI_ADR_SPACE_FIXED_HARDWARE))
526     {
527         return (FALSE);
528     }

531     return (TRUE);
532 }

535 #endif

```

new/usr/src/common/acpica/components/hardware/hwacpi.c

1

\*\*\*\*\*

6700 Fri Jan 17 07:30:18 2014  
new/usr/src/common/acpica/components/hardware/hwacpi.c  
update to acpica-unix2-20140114  
acpica-unix2-20130823  
PANKOVs restructure  
\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: hwacpi - ACPI Hardware Initialization/Mode Interface
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __HWACPI_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48
49 #define _COMPONENT ACPI_HARDWARE
50 #define ACPI_MODULE_NAME ("hwacpi")
51
52 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */
53 /*****
54 *
55 * FUNCTION: AcpiHwSetMode
56 */
57
```

new/usr/src/common/acpica/components/hardware/hwacpi.c

2

```
58 *
59 * PARAMETERS: Mode - SYS_MODE ACPI or SYS_MODE_LEGACY
60 *
61 * RETURN: Status
62 *
63 * DESCRIPTION: Transitions the system into the requested mode.
64 *
65 *****/
66
67 ACPI_STATUS
68 AcpiHwSetMode (
69     UINT32 Mode)
70 {
71     ACPI_STATUS Status;
72     UINT32 Retry;
73
74     ACPI_FUNCTION_TRACE (HwSetMode);
75
76     /* If the Hardware Reduced flag is set, machine is always in acpi mode */
77
78     if (AcpiGbl_ReducedHardware)
79     {
80         return ACPI_STATUS (AE_OK);
81     }
82
83     /*
84     * ACPI 2.0 clarified that if SMI_CMD in FADT is zero,
85     * system does not support mode transition.
86     */
87     if (!AcpiGbl_FADT.SmiCommand)
88     {
89         ACPI_ERROR ((AE_INFO, "No SMI_CMD in FADT, mode transition failed"));
90         return ACPI_STATUS (AE_NO_HARDWARE_RESPONSE);
91     }
92
93     /*
94     * ACPI 2.0 clarified the meaning of ACPI_ENABLE and ACPI_DISABLE
95     * in FADT: If it is zero, enabling or disabling is not supported.
96     * As old systems may have used zero for mode transition,
97     * we make sure both the numbers are zero to determine these
98     * transitions are not supported.
99     */
100     if (!AcpiGbl_FADT.AcpiEnable && !AcpiGbl_FADT.AcpiDisable)
101     {
102         ACPI_ERROR ((AE_INFO,
103             "No ACPI mode transition supported in this system "
104             "(enable/disable both zero)"));
105         return ACPI_STATUS (AE_OK);
106     }
107
108     switch (Mode)
109     {
110     case ACPI_SYS_MODE ACPI:
111
112         /* BIOS should have disabled ALL fixed and GP events */
113
114         Status = AcpiHwWritePort (AcpiGbl_FADT.SmiCommand,
115             (UINT32) AcpiGbl_FADT.AcpiEnable, 8);
116         ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "Attempting to enable ACPI mode\n"));
117         break;
118
119     case ACPI_SYS_MODE_LEGACY:
120
121
```



```

123      /*
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;
132
133  default:
134
135      return_ACPI_STATUS (AE_BAD_PARAMETER);
136  }
137
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  }
144
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)
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);
159      AcpiOsStall(1000);
160      Retry--;
161  }
162
163  ACPI_ERROR ((AE_INFO, "Hardware did not change modes"));
164  return_ACPI_STATUS (AE_NO_HARDWARE_RESPONSE);
165 }
166
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  *****/
179
180  UINT32
181  AcpiHwGetMode (
182      void)
183  {
184      ACPI_STATUS      Status;
185      UINT32            Value;

```

```

188      ACPI_FUNCTION_TRACE (HwGetMode);
189
190      /* If the Hardware Reduced flag is set, machine is always in acpi mode */
191
192      if (AcpiGbl_ReducedHardware)
193      {
194          return_UINT32 (ACPI_SYS_MODE_ACPI);
195      }
196
197      /*
198       * ACPI 2.0 clarified that if SMI_CMD in FADT is zero,
199       * system does not support mode transition.
200       */
201      if (!AcpiGbl_FADT.SmiCommand)
202      {
203          return_UINT32 (ACPI_SYS_MODE_ACPI);
204      }
205
206      Status = AcpiReadBitRegister (ACPI_BITREG_SCI_ENABLE, &Value);
207      if (ACPI_FAILURE (Status))
208      {
209          return_UINT32 (ACPI_SYS_MODE_LEGACY);
210      }
211
212      if (Value)
213      {
214          return_UINT32 (ACPI_SYS_MODE_ACPI);
215      }
216      else
217      {
218          return_UINT32 (ACPI_SYS_MODE_LEGACY);
219      }
220  }
221
222 #endif /* !ACPI_REDUCED_HARDWARE */

```

new/usr/src/common/acpica/components/hardware/hwesleep.c

1

```
*****
8029 Fri Jan 17 07:30:19 2014
new/usr/src/common/acpica/components/hardware/hwesleep.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Name: hwesleep.c - ACPI Hardware Sleep/Wake Support functions for the
4 * extended FADT-V5 sleep registers.
5 *
6 *****/

8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"

48 #define _COMPONENT ACPI_HARDWARE
49 ACPI_MODULE_NAME ("hwesleep")

52 /*****
53 *
54 * FUNCTION: AcpiHwExecutesSleepMethod
55 *
56 * PARAMETERS: MethodPathname - Pathname of method to execute
57 * IntegerArgument - Argument to pass to the method
58 *
59 * RETURN: None
60 */
```

new/usr/src/common/acpica/components/hardware/hwesleep.c

2

```
61 * DESCRIPTION: Execute a sleep/wake related method with one integer argument
62 * and no return value.
63 *
64 *****/

66 void
67 AcpiHwExecutesSleepMethod (
68     char *MethodPathname,
69     UINT32 IntegerArgument)
70 {
71     ACPI_OBJECT_LIST ArgList;
72     ACPI_OBJECT Arg;
73     ACPI_STATUS Status;

76     ACPI_FUNCTION_TRACE (HwExecutesSleepMethod);

79     /* One argument, IntegerArgument; No return value expected */

81     ArgList.Count = 1;
82     ArgList.Pointer = &Arg;
83     Arg.Type = ACPI_TYPE_INTEGER;
84     Arg.Integer.Value = (UINT64) IntegerArgument;

86     Status = AcpiEvaluateObject (NULL, MethodPathname, &ArgList, NULL);
87     if (ACPI_FAILURE (Status) && Status != AE_NOT_FOUND)
88     {
89         ACPI_EXCEPTION ((AE_INFO, Status, "While executing method %s",
90             MethodPathname));
91     }

93     return_VOID;
94 }

97 /*****
98 *
99 * FUNCTION: AcpiHwExtendedSleep
100 *
101 * PARAMETERS: SleepState - Which sleep state to enter
102 *
103 * RETURN: Status
104 *
105 * DESCRIPTION: Enter a system sleep state via the extended FADT sleep
106 * registers (V5 FADT).
107 * THIS FUNCTION MUST BE CALLED WITH INTERRUPTS DISABLED
108 *
109 *****/

111 ACPI_STATUS
112 AcpiHwExtendedSleep (
113     UINT8 SleepState)
114 {
115     ACPI_STATUS Status;
116     UINT8 SleepTypeValue;
117     UINT64 SleepStatus;

120     ACPI_FUNCTION_TRACE (HwExtendedSleep);

123     /* Extended sleep registers must be valid */

125     if (!AcpiGbl_FADT.SleepControl.Address ||
126         !AcpiGbl_FADT.SleepStatus.Address)
```

```

127 {
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 [%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 }

```

new/usr/src/common/acpica/components/hardware/hwesleep.c

new/usr/src/common/acpica/components/hardware/hwgpe.c

1

\*\*\*\*\*

14266 Fri Jan 17 07:30:19 2014

new/usr/src/common/acpica/components/hardware/hwgpe.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: hwgpe - Low level GPE enable/disable/clear functions
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #include "acpi.h"
46 #include "accommon.h"
47 #include "acevents.h"
48
49 #define _COMPONENT          ACPI_HARDWARE
50 #define ACPI_MODULE_NAME    ("hwgpe")
51
52 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */
53
54 /* Local prototypes */
55
56 static ACPI_STATUS
57 AcpiHwEnableWakeupGpeBlock (
58     ACPI_GPE_XRPT_INFO    *GpeXruptInfo,
```

new/usr/src/common/acpica/components/hardware/hwgpe.c

2

```
58     ACPI_GPE_BLOCK_INFO    *GpeBlock,
59     void                    *Context);
60
61
62 /*****
63 *
64 * FUNCTION:      AcpiHwGetGpeRegisterBit
65 *
66 * PARAMETERS:    GpeEventInfo      - Info block for the GPE
67 *                GpeRegisterInfo    - Info block for the GPE register
68 *
69 * RETURN:        Register mask with a one in the GPE bit position
70 *
71 * DESCRIPTION:    Compute the register mask for this GPE. One bit is set in the
72 *                correct position for the input GPE.
73 *****/
74
75 UINT32
76 AcpiHwGetGpeRegisterBit (
77     ACPI_GPE_EVENT_INFO    *GpeEventInfo,
78     ACPI_GPE_REGISTER_INFO *GpeRegisterInfo)
79 {
80     return ((UINT32) 1 <<
81         (GpeEventInfo->GpeNumber - GpeRegisterInfo->BaseGpeNumber));
82 }
83
84
85 /*****
86 *
87 * FUNCTION:      AcpiHwLowSetGpe
88 *
89 * PARAMETERS:    GpeEventInfo      - Info block for the GPE to be disabled
90 *                Action            - Enable or disable
91 *
92 * RETURN:        Status
93 *
94 * DESCRIPTION:    Enable or disable a single GPE in the parent enable register.
95 *
96 *****/
97
98 ACPI_STATUS
99 AcpiHwLowSetGpe (
100     ACPI_GPE_EVENT_INFO    *GpeEventInfo,
101     UINT32                  Action)
102 {
103     ACPI_GPE_REGISTER_INFO *GpeRegisterInfo;
104     ACPI_STATUS             Status;
105     UINT32                  EnableMask;
106     UINT32                  RegisterBit;
107
108     ACPI_FUNCTION_ENTRY ();
109
110     /* Get the info block for the entire GPE register */
111
112     GpeRegisterInfo = GpeEventInfo->RegisterInfo;
113     if (!GpeRegisterInfo)
114     {
115         return (AE_NOT_EXIST);
116     }
117 }
```

```

120  /* 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);
132  switch (Action)
133  {
134      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));
157              ACPI_ERROR ((AE_INFO, "Invalid GPE Action, %u\n", Action));
158              return (AE_BAD_PARAMETER);
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);
203  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 */
```

new/usr/src/common/acpica/components/hardware/hwpci.c

1

```
*****
14483 Fri Jan 17 07:30:19 2014
new/usr/src/common/acpica/components/hardware/hwpci.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: hwpci - Obtain PCI bus, device, and function numbers
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __HWPCI_C__

46 #include "acpi.h"
47 #include "accommon.h"

50 #define _COMPONENT          ACPI_NAMESPACE
51 #define ACPI_MODULE_NAME    ("hwpci")

54 /* PCI configuration space values */

56 #define PCI_CFG_HEADER_TYPE_REG      0x0E
57 #define PCI_CFG_PRIMARY_BUS_NUMBER_REG 0x18
58 #define PCI_CFG_SECONDARY_BUS_NUMBER_REG 0x19
```

new/usr/src/common/acpica/components/hardware/hwpci.c

2

```
60 /* PCI header values */

62 #define PCI_HEADER_TYPE_MASK          0x7F
63 #define PCI_TYPE_BRIDGE               0x01
64 #define PCI_TYPE_CARDBUS_BRIDGE      0x02

66 typedef struct acpi_pci_device
67 {
68     ACPI_HANDLE      Device;
69     struct acpi_pci_device *Next;

71 } ACPI_PCI_DEVICE;
    unchanged_portion_omitted

244 /*****
245 *
246 * FUNCTION:      AcpiHwProcessPciList
247 *
248 * PARAMETERS:    PciId          - Initial values for the PCI ID. May be
249 *                                modified by this function.
250 *                ListHead       - Device list created by
251 *                                AcpiHwBuildPciList
252 *
253 * RETURN:        Status
254 *
255 * DESCRIPTION:   Walk downward through the PCI device list, getting the device
256 *                info for each, via the PCI configuration space and updating
257 *                the PCI ID as necessary. Deletes the list during traversal.
258 *
259 *****/

261 static ACPI_STATUS
262 AcpiHwProcessPciList (
263     ACPI_PCI_ID      *PciId,
264     ACPI_PCI_DEVICE  *ListHead)
265 {
266     ACPI_STATUS      Status = AE_OK;
267     *Info;
268     UINT16           BusNumber;
269     BOOLEAN          IsBridge = TRUE;

272     ACPI_FUNCTION_NAME (HwProcessPciList);

275     ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION,
276         "Input PciId:  Seg %4.4X Bus %4.4X Dev %4.4X Func %4.4X\n",
277         PciId->Segment, PciId->Bus, PciId->Device, PciId->Function));

279     BusNumber = PciId->Bus;

281     /*
282     * Descend down the namespace tree, collecting PCI device, function,
283     * and bus numbers. BusNumber is only important for PCI bridges.
284     * Algorithm: As we descend the tree, use the last valid PCI device,
285     * function, and bus numbers that are discovered, and assign them
286     * to the PCI ID for the target device.
287     */
288     Info = ListHead;
289     while (Info)
290     {
291         Status = AcpiHwGetPciDeviceInfo (PciId, Info->Device,
292             &BusNumber, &IsBridge);
293         if (ACPI_FAILURE (Status))
```



```
294     {
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
```

new/usr/src/common/acpica/components/hardware/hwregs.c

1

\*\*\*\*\*

21657 Fri Jan 17 07:30:20 2014

new/usr/src/common/acpica/components/hardware/hwregs.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: hwregs - Read/write access functions for the various ACPI
4 *                   control and status registers.
5 *
6 *****/
7
8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
45 #define __HWREGS_C__
46
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acevents.h"
50
51 #define _COMPONENT          ACPI_HARDWARE
52 #define ACPI_MODULE_NAME    ("hwregs")
53
54
55 #if (!ACPI_REDUCED_HARDWARE)
56
57 /* Local Prototypes */
```

new/usr/src/common/acpica/components/hardware/hwregs.c

2

```
59 static ACPI_STATUS
60 AcpiHwReadMultiple (
61     UINT32                *Value,
62     ACPI_GENERIC_ADDRESS  *RegisterA,
63     ACPI_GENERIC_ADDRESS  *RegisterB);
64
65 static ACPI_STATUS
66 AcpiHwWriteMultiple (
67     UINT32                Value,
68     ACPI_GENERIC_ADDRESS  *RegisterA,
69     ACPI_GENERIC_ADDRESS  *RegisterB);
70
71 #endif /* !ACPI_REDUCED_HARDWARE */
72
73 /*****
74 *
75 * FUNCTION:      AcpiHwValidateRegister
76 *
77 * PARAMETERS:    Reg          - GAS register structure
78 *                MaxBitWidth  - Max BitWidth supported (32 or 64)
79 *                Address       - Pointer to where the gas->address
80 *                               is returned
81 *
82 * RETURN:        Status
83 *
84 * DESCRIPTION:   Validate the contents of a GAS register. Checks the GAS
85 *                pointer, Address, SpaceId, BitWidth, and BitOffset.
86 *
87 *****/
88
89 ACPI_STATUS
90 AcpiHwValidateRegister (
91     ACPI_GENERIC_ADDRESS  *Reg,
92     UINT8                 MaxBitWidth,
93     UINT64                 *Address)
94 {
95     /* Must have a valid pointer to a GAS structure */
96
97     if (!Reg)
98     {
99         return (AE_BAD_PARAMETER);
100     }
101
102     /*
103     * Copy the target address. This handles possible alignment issues.
104     * Address must not be null. A null address also indicates an optional
105     * ACPI register that is not supported, so no error message.
106     */
107     ACPI_MOVE_64_TO_64 (Address, &Reg->Address);
108     if (!(*Address))
109     {
110         return (AE_BAD_ADDRESS);
111     }
112
113     /* Validate the SpaceID */
114
115     if ((Reg->SpaceId != ACPI_ADR_SPACE_SYSTEM_MEMORY) &&
116         (Reg->SpaceId != ACPI_ADR_SPACE_SYSTEM_IO))
117     {
118         ACPI_ERROR ((AE_INFO,
119             "Unsupported address space: 0x%X", Reg->SpaceId));
120         return (AE_SUPPORT);
121     }
122 }
```

```

124  /* 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  * FUNCTION:      AcpiHwRead
150  *
151  * PARAMETERS:    Value          - Where the value is returned
152  *                Reg            - GAS register structure
153  *
154  * RETURN:        Status
155  *
156  * DESCRIPTION:    Read from either memory or IO space. This is a 32-bit max
157  *                version of AcpiRead, used internally since the overhead of
158  *                64-bit values is not needed.
159  *
160  * LIMITATIONS:    <These limitations also apply to AcpiHwWrite>
161  *                BitWidth must be exactly 8, 16, or 32.
162  *                SpaceID must be SystemMemory or SystemIO.
163  *                BitOffset and AccessWidth are currently ignored, as there has
164  *                not been a need to implement these.
165  *
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;);
202
203      *Value = (UINT32) Value64;
204      Address, Value, Reg->BitWidth;);
205  }
206  else /* ACPI_ADR_SPACE_SYSTEM_IO, validated earlier */
207  {
208      Status = AcpiHwReadPort ((ACPI_IO_ADDRESS)
209                               Address, Value, Reg->BitWidth);
210  }

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  * FUNCTION:      AcpiHwWrite
222  *
223  * PARAMETERS:    Value          - Value to be written
224  *                Reg            - GAS register structure
225  *
226  * RETURN:        Status
227  *
228  * DESCRIPTION:    Write to either memory or IO space. This is a 32-bit max
229  *                version of AcpiWrite, used internally since the overhead of
230  *                64-bit values is not needed.
231  *
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  ACPI_DEBUG_PRINT ((ACPI_DB_IO,
271  "Wrote: %8.8X width %2d to %8.8X%8.8X (%s)\n",
272  Value, Reg->BitWidth, ACPI_FORMAT_UINT64 (Address),
273  AcpiUtGetRegionName (Reg->SpaceId)));

275  return (Status);
276  }

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  *****/

292  ACPI_STATUS
293  AcpiHwClearAcpiStatus (
294      void)
295  {
296      ACPI_STATUS      Status;
297      ACPI_CPU_FLAGS    LockFlags = 0;

300      ACPI_FUNCTION_TRACE (HwClearAcpiStatus);

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_XPmlaStatus.Address)));

307      LockFlags = AcpiOsAcquireLock (AcpiGbl_HardwareLock);

309      /* Clear the fixed events in PM1 A/B */

311      Status = AcpiHwRegisterWrite (ACPI_REGISTER_PM1_STATUS,
312      ACPI_BITMASK_ALL_FIXED_STATUS);
313      if (ACPI_FAILURE (Status))
314      {
315          goto UnlockAndExit;
316      }

318      /* Clear the GPE Bits in all GPE registers in all GPE blocks */

```

```

320      Status = AcpiEvWalkGpeList (AcpiHwClearGpeBlock, NULL);

322  UnlockAndExit:
323      AcpiOsReleaseLock (AcpiGbl_HardwareLock, LockFlags);
324      return ACPI_STATUS (Status);
325  }

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  *****/

340  ACPI_BIT_REGISTER_INFO *
341  AcpiHwGetBitRegisterInfo (
342      UINT32          RegisterId)
343  {
344      ACPI_FUNCTION_ENTRY ();

347      if (RegisterId > ACPI_BITREG_MAX)
348      {
349          ACPI_ERROR ((AE_INFO, "Invalid BitRegister ID: 0x%X", RegisterId));
350          return (NULL);
351      }

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  *****/

412  ACPI_STATUS
413  AcpiHwRegisterRead (
414      UINT32          RegisterId,
415      UINT32          *ReturnValue)
416  {
417      UINT32          Value = 0;
418      ACPI_STATUS      Status;

421      ACPI_FUNCTION_TRACE (HwRegisterRead);

424      switch (RegisterId)
425      {

```

```

426     case ACPI_REGISTER_PM1_STATUS:          /* PM1 A/B: 16-bit access each */
427
428         Status = AcpiHwReadMultiple (&Value,
429             &AcpiGbl_XPmlaStatus,
430             &AcpiGbl_XPmlbStatus);
431         break;
432
433     case ACPI_REGISTER_PM1_ENABLE:          /* PM1 A/B: 16-bit access each */
434
435         Status = AcpiHwReadMultiple (&Value,
436             &AcpiGbl_XPmlaEnable,
437             &AcpiGbl_XPmlbEnable);
438         break;
439
440     case ACPI_REGISTER_PM1_CONTROL:        /* PM1 A/B: 16-bit access each */
441
442         Status = AcpiHwReadMultiple (&Value,
443             &AcpiGbl_FADT.XPmlaControlBlock,
444             &AcpiGbl_FADT.XPmlbControlBlock);
445
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;
453
454     case ACPI_REGISTER_PM2_CONTROL:        /* 8-bit access */
455
456         Status = AcpiHwRead (&Value, &AcpiGbl_FADT.XPm2ControlBlock);
457         break;
458
459     case ACPI_REGISTER_PM_TIMER:          /* 32-bit access */
460
461         Status = AcpiHwRead (&Value, &AcpiGbl_FADT.XPmTimerBlock);
462         break;
463
464     case ACPI_REGISTER_SMI_COMMAND_BLOCK:  /* 8-bit access */
465
466         Status = AcpiHwReadPort (AcpiGbl_FADT.SmiCommand, &Value, 8);
467         break;
468
469     default:
470
471     default:
472         ACPI_ERROR ((AE_INFO, "Unknown Register ID: 0x%X",
473             RegisterId));
474         Status = AE_BAD_PARAMETER;
475         break;
476     }
477
478     if (ACPI_SUCCESS (Status))
479     {
480         *ReturnValue = Value;
481     }
482
483     return ACPI_STATUS (Status);
484 }

```

```

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  *****/
511
512 ACPI_STATUS
513 AcpiHwRegisterWrite (
514     UINT32              RegisterId,
515     UINT32              Value)
516 {
517     ACPI_STATUS          Status;
518     UINT32              ReadValue;
519
520     ACPI_FUNCTION_TRACE (HwRegisterWrite);
521
522     switch (RegisterId)
523     {
524     case ACPI_REGISTER_PM1_STATUS:          /* PM1 A/B: 16-bit access each */
525         /*
526          * Handle the "ignored" bit in PM1 Status. According to the ACPI
527          * specification, ignored bits are to be preserved when writing.
528          * Normally, this would mean a read/modify/write sequence. However,
529          * preserving a bit in the status register is different. Writing a
530          * one clears the status, and writing a zero preserves the status.
531          * Therefore, we must always write zero to the ignored bit.
532          *
533          * This behavior is clarified in the ACPI 4.0 specification.
534          */
535         Value &= ~ACPI_PM1_STATUS_PRESERVED_BITS;
536
537         Status = AcpiHwWriteMultiple (Value,
538             &AcpiGbl_XPmlaStatus,
539             &AcpiGbl_XPmlbStatus);
540         break;
541
542     case ACPI_REGISTER_PM1_ENABLE:          /* PM1 A/B: 16-bit access each */
543
544         Status = AcpiHwWriteMultiple (Value,
545             &AcpiGbl_XPmlaEnable,
546             &AcpiGbl_XPmlbEnable);
547         break;
548
549     }
550 }

```

```

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.XPmlaControlBlock,
558                                  &AcpiGbl_FADT.XPmlbControlBlock);
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.XPmlaControlBlock,
572                                   &AcpiGbl_FADT.XPmlbControlBlock);
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 Fri Jan 17 07:30:20 2014
new/usr/src/common/acpica/components/hardware/hwsleep.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: hwsleep.c - ACPI Hardware Sleep/Wake Support functions for the
4 *       original/legacy sleep/PM registers.
5 *
6 *****/

8 /*
9  * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"

48 #define _COMPONENT          ACPI_HARDWARE
49 #define ACPI_MODULE_NAME    ("hwsleep")

52 #if (!ACPI_REDUCED_HARDWARE) /* Entire module */
53 /*****
54 *
55 * FUNCTION:      AcpiHwLegacySleep
56 *
57 * PARAMETERS:    SleepState      - Which sleep state to enter
58 *
59 * RETURN:        Status

```

```

60 *
61 * DESCRIPTION: Enter a system sleep state via the legacy FADT PM registers
62 *              THIS FUNCTION MUST BE CALLED WITH INTERRUPTS DISABLED
63 *
64 *****/

66 ACPI_STATUS
67 AcpiHwLegacySleep (
68     UINT8                SleepState)
69 {
70     ACPI_BIT_REGISTER_INFO *SleepTypeRegInfo;
71     ACPI_BIT_REGISTER_INFO *SleepEnableRegInfo;
72     UINT32                  PmlaControl;
73     UINT32                  PmlbControl;
74     UINT32                  InValue;
75     ACPI_STATUS              Status;

78     ACPI_FUNCTION_TRACE (HwLegacySleep);

81     SleepTypeRegInfo = AcpiHwGetBitRegisterInfo (ACPI_BITREG_SLEEP_TYPE);
82     SleepEnableRegInfo = AcpiHwGetBitRegisterInfo (ACPI_BITREG_SLEEP_ENABLE);

84     /* Clear wake status */

86     Status = AcpiWriteBitRegister (ACPI_BITREG_WAKE_STATUS, ACPI_CLEAR_STATUS);
87     if (ACPI_FAILURE (Status))
88     {
89         return_ACPI_STATUS (Status);
90     }

92     /* Clear all fixed and general purpose status bits */

94     Status = AcpiHwClearAcpiStatus ();
95     if (ACPI_FAILURE (Status))
96     {
97         return_ACPI_STATUS (Status);
98     }

100     /*
101     * 1) Disable/Clear all GPEs
102     * 2) Enable all wakeup GPEs
103     */
104     Status = AcpiHwDisableAllGpes ();
105     if (ACPI_FAILURE (Status))
106     {
107         return_ACPI_STATUS (Status);
108     }
109     AcpiGbl_SystemAwakeAndRunning = FALSE;

111     Status = AcpiHwEnableAllWakeupGpes ();
112     if (ACPI_FAILURE (Status))
113     {
114         return_ACPI_STATUS (Status);
115     }

117     /* Get current value of PM1A control */

119     Status = AcpiHwRegisterRead (ACPI_REGISTER_PM1_CONTROL,
120                                 &PmlaControl);
121     if (ACPI_FAILURE (Status))
122     {
123         return_ACPI_STATUS (Status);
124     }
125     ACPI_DEBUG_PRINT ((ACPI_DB_INIT,

```

```

126     "Entering sleep state [%u]\n", SleepState));
128     /* Clear the SLP_EN and SLP_TYP fields */
130     PmlaControl &= ~(SleepTypeRegInfo->AccessBitMask |
131                     SleepEnableRegInfo->AccessBitMask);
132     PmlbControl = PmlaControl;
134     /* Insert the SLP_TYP bits */
136     PmlaControl |= (AcpiGbl_SleepTypeA << SleepTypeRegInfo->BitPosition);
137     PmlbControl |= (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 = AcpiHwWritePmlControl (PmlaControl, PmlbControl);
147     if (ACPI_FAILURE (Status))
148     {
149         return_ACPI_STATUS (Status);
150     }
152     /* Insert the sleep enable (SLP_EN) bit */
154     PmlaControl |= SleepEnableRegInfo->AccessBitMask;
155     PmlbControl |= 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 = AcpiHwWritePmlControl (PmlaControl, PmlbControl);
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  * FUNCTION:      AcpiHwLegacyWakePrep
210  *
211  * PARAMETERS:    SleepState          - Which sleep state we just exited
212  *
213  * RETURN:        Status
214  *
215  * DESCRIPTION:   Perform the first state of OS-independent ACPI cleanup after a
216  *               sleep.
217  *               Called with interrupts ENABLED.
218  *
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               PmlaControl;
230     UINT32               PmlbControl;
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                                     &PmlaControl);
253         if (ACPI_SUCCESS (Status))
254         {
255             /* Clear the SLP_EN and SLP_TYP fields */
257             PmlaControl &= ~(SleepTypeRegInfo->AccessBitMask |

```



```

258     SleepEnableRegInfo->AccessBitMask);
259     PmlbControl = PmlaControl;

261     /* Insert the SLP_TYP bits */

263     PmlaControl |= (AcpiGbl_SleepTypeA <<
264         SleepTypeRegInfo->BitPosition);
265     PmlbControl |= (AcpiGbl_SleepTypeB <<
266         SleepTypeRegInfo->BitPosition);

268     /* Write the control registers and ignore any errors */

270     (void) AcpiHwWritePmlControl (PmlaControl, PmlbControl);
271 }
272 }

274     return_ACPI_STATUS (Status);
275 }

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  *****/

291 ACPI_STATUS
292 AcpiHwLegacyWake (
293     UINT8             SleepState)
294 {
295     ACPI_STATUS        Status;

298     ACPI_FUNCTION_TRACE (HwLegacyWake);

301     /* Ensure EnterSleepStatePrep -> EnterSleepState ordering */

303     AcpiGbl_SleepTypeA = ACPI_SLEEP_TYPE_INVALID;
304     AcpiHwExecuteSleepMethod (METHOD_PATHNAME__SST, ACPI_SST_WAKING);

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     }

320     Status = AcpiHwEnableAllRuntimeGpes ();
321     if (ACPI_FAILURE (Status))
322     {
323         return_ACPI_STATUS (Status);

```

```

324     }

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);

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;

340     /* Enable power button */

342     (void) AcpiWriteBitRegister(
343         AcpiGbl_FixedEventInfo[ACPI_EVENT_POWER_BUTTON].EnableRegisterId,
344         ACPI_ENABLE_EVENT);

346     (void) AcpiWriteBitRegister(
347         AcpiGbl_FixedEventInfo[ACPI_EVENT_POWER_BUTTON].StatusRegisterId,
348         ACPI_CLEAR_STATUS);

350     AcpiHwExecuteSleepMethod (METHOD_PATHNAME__SST, ACPI_SST_WORKING);
351     return_ACPI_STATUS (Status);
352 }

354 #endif /* !ACPI_REduced_HARDWARE */

```

new/usr/src/common/acpica/components/hardware/hwtimer.c

1

\*\*\*\*\*

7124 Fri Jan 17 07:30:20 2014  
new/usr/src/common/acpica/components/hardware/hwtimer.c  
update to acpica-unix2-20140114  
update to acpica-unix2-20130927  
acpica-unix2-20130823  
PANKOV's restructure  
\*\*\*\*\*

```
1 /*****
2 *
3 * Name: hwtimer.c - ACPI Power Management Timer Interface
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 EXPORT\_ACPI\_INTERFACES

```
46 #include "acpi.h"
47 #include "accommon.h"
```

```
49 #define _COMPONENT          ACPI_HARDWARE
50       ACPI_MODULE_NAME      ("hwtimer")
```

53 #if (!ACPI\_REDUCED\_HARDWARE) /\* Entire module \*/

54 /\*\*\*\*\*

```
55 *
56 * FUNCTION:      AcpiGetTimerResolution
```

new/usr/src/common/acpica/components/hardware/hwtimer.c

2

```
57 *
58 * PARAMETERS:    Resolution          - Where the resolution is returned
59 *
60 * RETURN:        Status and timer resolution
61 *
62 * DESCRIPTION:    Obtains resolution of the ACPI PM Timer (24 or 32 bits).
63 *
64 *****/
```

```
66 ACPI_STATUS
67 AcpiGetTimerResolution (
68     UINT32                *Resolution)
69 {
70     ACPI_FUNCTION_TRACE (AcpiGetTimerResolution);
```

```
73     if (!Resolution)
74     {
75         return_ACPI_STATUS (AE_BAD_PARAMETER);
76     }
77
78     if ((AcpiGbl_FADT.Flags & ACPI_FADT_32BIT_TIMER) == 0)
79     {
80         *Resolution = 24;
81     }
82     else
83     {
84         *Resolution = 32;
85     }
```

```
87     return_ACPI_STATUS (AE_OK);
88 }
```

90 ACPI\_EXPORT\_SYMBOL (AcpiGetTimerResolution)

93 /\*\*\*\*\*

```
94 *
95 * FUNCTION:      AcpiGetTimer
96 *
97 * PARAMETERS:    Ticks                - Where the timer value is returned
98 *
99 * RETURN:        Status and current timer value (ticks)
100 *
101 * DESCRIPTION:    Obtains current value of ACPI PM Timer (in ticks).
102 *
103 *****/
```

```
105 ACPI_STATUS
106 AcpiGetTimer (
107     UINT32                *Ticks)
108 {
109     ACPI_STATUS            Status;
```

112 ACPI\_FUNCTION\_TRACE (AcpiGetTimer);

```
115     if (!Ticks)
116     {
117         return_ACPI_STATUS (AE_BAD_PARAMETER);
118     }
```

120 /\* ACPI 5.0A: PM Timer is optional \*/

122 if (!AcpiGbl\_FADT.XPmTimerBlock.Address)

```

123 {
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;
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 */

```

new/usr/src/common/acpica/components/hardware/hwvalid.c

1

\*\*\*\*\*

11591 Fri Jan 17 07:30:21 2014

new/usr/src/common/acpica/components/hardware/hwvalid.c

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: hwvalid - I/O request validation
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __HWVALID_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48
49 #define _COMPONENT ACPI_HARDWARE
50 #define ACPI_MODULE_NAME ("hwvalid")
51
52 /* Local prototypes */
53
54 static ACPI_STATUS
55 AcpiHwValidateIoRequest (
56     ACPI_IO_ADDRESS Address,
57     UINT32 BitWidth);
```

new/usr/src/common/acpica/components/hardware/hwvalid.c

2

```
60 /*
61 * Protected I/O ports. Some ports are always illegal, and some are
62 * conditionally illegal. This table must remain ordered by port address.
63 *
64 * The table is used to implement the Microsoft port access rules that
65 * first appeared in Windows XP. Some ports are always illegal, and some
66 * ports are only illegal if the BIOS calls _OSI with a WinXP string or
67 * later (meaning that the BIOS itself is post-XP.)
68 *
69 * This provides ACPICA with the desired port protections and
70 * Microsoft compatibility.
71 *
72 * Description of port entries:
73 * DMA: DMA controller
74 * PIC0: Programmable Interrupt Controller (8259A)
75 * PIT1: System Timer 1
76 * PIT2: System Timer 2 failsafe
77 * RTC: Real-time clock
78 * CMOS: Extended CMOS
79 * DMA1: DMA 1 page registers
80 * DMA1L: DMA 1 Ch 0 low page
81 * DMA2: DMA 2 page registers
82 * DMA2L: DMA 2 low page refresh
83 * ARBC: Arbitration control
84 * SETUP: Reserved system board setup
85 * POS: POS channel select
86 * PIC1: Cascaded PIC
87 * IDMA: ISA DMA
88 * ELCR: PIC edge/level registers
89 * PCI: PCI configuration space
90 */
91 static const ACPI_PORT_INFO AcpiProtectedPorts[] =
92 {
93     {"DMA", 0x0000, 0x000F, ACPI_OSI_WIN_XP},
94     {"PIC0", 0x0020, 0x0021, ACPI_ALWAYS_ILLEGAL},
95     {"PIT1", 0x0040, 0x0043, ACPI_OSI_WIN_XP},
96     {"PIT2", 0x0048, 0x004B, ACPI_OSI_WIN_XP},
97     {"RTC", 0x0070, 0x0071, ACPI_OSI_WIN_XP},
98     {"CMOS", 0x0074, 0x0076, ACPI_OSI_WIN_XP},
99     {"DMA1", 0x0081, 0x0083, ACPI_OSI_WIN_XP},
100     {"DMA1L", 0x0087, 0x0087, ACPI_OSI_WIN_XP},
101     {"DMA2", 0x0089, 0x008B, ACPI_OSI_WIN_XP},
102     {"DMA2L", 0x008F, 0x008F, ACPI_OSI_WIN_XP},
103     {"ARBC", 0x0090, 0x0091, ACPI_OSI_WIN_XP},
104     {"SETUP", 0x0093, 0x0094, ACPI_OSI_WIN_XP},
105     {"POS", 0x0096, 0x0097, ACPI_OSI_WIN_XP},
106     {"PIC1", 0x00A0, 0x00A1, ACPI_ALWAYS_ILLEGAL},
107     {"IDMA", 0x00C0, 0x00DF, ACPI_OSI_WIN_XP},
108     {"ELCR", 0x04D0, 0x04D1, ACPI_ALWAYS_ILLEGAL},
109     {"PCI", 0x0CF8, 0x0CFF, ACPI_OSI_WIN_XP}
110 };
111
112 #define ACPI_PORT_INFO_ENTRIES ACPI_ARRAY_LENGTH (AcpiProtectedPorts)
113
114
115 /*****
116 *
117 * FUNCTION: AcpiHwValidateIoRequest
118 *
119 * PARAMETERS: Address Address of I/O port/register
120 * BitWidth Number of bits (8,16,32)
121 *
122 * RETURN: Status
123 */
```

```

124 * 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_

```

new/usr/src/common/acpica/components/hardware/hwxface.c

1

\*\*\*\*\*

19617 Fri Jan 17 07:30:21 2014  
new/usr/src/common/acpica/components/hardware/hwxface.c  
update to acpica-unix2-20140114  
update to acpica-unix2-20130927  
acpica-unix2-20130823  
PANKOV's restructure  
\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: hwxface - Public ACPICA hardware interfaces
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 EXPORT_ACPI_INTERFACES
45
46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acnamesp.h"
49
50 #define _COMPONENT ACPI_HARDWARE
51 #define ACPI_MODULE_NAME ("hwxface")
52
53 /*****
54 *
55 * FUNCTION: AcpiReset
56 */
```

new/usr/src/common/acpica/components/hardware/hwxface.c

2

```
57 *
58 * PARAMETERS: None
59 *
60 * RETURN: Status
61 *
62 * DESCRIPTION: Set reset register in memory or IO space. Note: Does not
63 * support reset register in PCI config space, this must be
64 * handled separately.
65 *
66 *****/
67
68 ACPI_STATUS
69 AcpiReset (
70     void)
71 {
72     ACPI_GENERIC_ADDRESS *ResetReg;
73     ACPI_STATUS Status;
74
75     ACPI_FUNCTION_TRACE (AcpiReset);
76
77     ResetReg = &AcpiGbl_FADT.ResetRegister;
78
79     /* Check if the reset register is supported */
80
81     if (!(AcpiGbl_FADT.Flags & ACPI_FADT_RESET_REGISTER) ||
82         !ResetReg->Address)
83     {
84         return ACPI_STATUS (AE_NOT_EXIST);
85     }
86
87     if (ResetReg->SpaceId == ACPI_ADR_SPACE_SYSTEM_IO)
88     {
89         /*
90          * For I/O space, write directly to the OS. This bypasses the port
91          * validation mechanism, which may block a valid write to the reset
92          * register.
93          *
94          * NOTE:
95          * The ACPI spec requires the reset register width to be 8, so we
96          * hardcode it here and ignore the FADT value. This maintains
97          * compatibility with other ACPI implementations that have allowed
98          * BIOS code with bad register width values to go unnoticed.
99          */
100         Status = AcpiOsWritePort ((ACPI_IO_ADDRESS) ResetReg->Address,
101                                 AcpiGbl_FADT.ResetValue, ACPI_RESET_REGISTER_WIDTH);
102     }
103     else
104     {
105         /* Write the reset value to the reset register */
106         Status = AcpiHwWrite (AcpiGbl_FADT.ResetValue, ResetReg);
107     }
108
109     return ACPI_STATUS (Status);
110 }
111
112 ACPI_EXPORT_SYMBOL (AcpiReset)
113
114 /*****
115 *
116 * FUNCTION: AcpiRead
117 */
```

```

122 * PARAMETERS:  Value          - Where the value is returned
123 *              Reg            - GAS register structure
124 *
125 * RETURN:      Status
126 *
127 * DESCRIPTION: Read from either memory or IO space.
128 *
129 * LIMITATIONS: <These limitations also apply to AcpiWrite>
130 *             BitWidth must be exactly 8, 16, 32, or 64.
131 *             SpaceId must be SystemMemory or SystemIO.
132 *             BitOffset and AccessWidth are currently ignored, as there has
133 *             not been a need to implement these.
134 *
135 *****/
137 ACPI_STATUS
138 AcpiRead (
139     UINT64          *ReturnValue,
140     ACPI_GENERIC_ADDRESS *Reg)
141 {
142     UINT32          ValueLo;
143     UINT32          ValueHi;
144     UINT32          Value;
145     UINT32          Width;
146     UINT64          Address;
147     ACPI_STATUS      Status;
148
149     ACPI_FUNCTION_NAME (AcpiRead);
150
151     if (!ReturnValue)
152     {
153         return (AE_BAD_PARAMETER);
154     }
155
156     /* Validate contents of the GAS register. Allow 64-bit transfers */
157
158     Status = AcpiHwValidateRegister (Reg, 64, &Address);
159     if (ACPI_FAILURE (Status))
160     {
161         return (Status);
162     }
163
164     Width = Reg->BitWidth;
165     if (Width == 64)
166     {
167         Width = 32; /* Break into two 32-bit transfers */
168     }
169
170     /* Initialize entire 64-bit return value to zero */
171
172     *ReturnValue = 0;
173     Value = 0;
174
175     /*
176      * Two address spaces supported: Memory or I/O. PCI_Config is
177      * Two address spaces supported: Memory or IO. PCI_Config is
178      * not supported here because the GAS structure is insufficient
179      */
180     if (Reg->SpaceId == ACPI_ADR_SPACE_SYSTEM_MEMORY)
181     {
182         Status = AcpiOsReadMemory ((ACPI_PHYSICAL_ADDRESS)
183             Address, ReturnValue, Reg->BitWidth);
184         Address, &Value, Width);
185         if (ACPI_FAILURE (Status))

```

```

174     {
175         return (Status);
176     }
177 }
178 else /* ACPI_ADR_SPACE_SYSTEM_IO, validated earlier */
179     *ReturnValue = Value;
180
181     if (Reg->BitWidth == 64)
182     {
183         ValueLo = 0;
184         ValueHi = 0;
185         /* Read the top 32 bits */
186
187         Width = Reg->BitWidth;
188         if (Width == 64)
189         {
190             Status = AcpiOsReadMemory ((ACPI_PHYSICAL_ADDRESS)
191                 (Address + 4), &Value, 32);
192             if (ACPI_FAILURE (Status))
193             {
194                 Width = 32; /* Break into two 32-bit transfers */
195                 return (Status);
196             }
197
198             *ReturnValue |= ((UINT64) Value << 32);
199         }
200     }
201     else /* ACPI_ADR_SPACE_SYSTEM_IO, validated earlier */
202     {
203         Status = AcpiHwReadPort ((ACPI_IO_ADDRESS)
204             Address, &ValueLo, Width);
205         Address, &Value, Width);
206         if (ACPI_FAILURE (Status))
207         {
208             return (Status);
209         }
210
211         *ReturnValue = Value;
212
213         if (Reg->BitWidth == 64)
214         {
215             /* Read the top 32 bits */
216
217             Status = AcpiHwReadPort ((ACPI_IO_ADDRESS)
218                 (Address + 4), &ValueHi, 32);
219             (Address + 4), &Value, 32);
220             if (ACPI_FAILURE (Status))
221             {
222                 return (Status);
223             }
224
225             *ReturnValue |= ((UINT64) Value << 32);
226         }
227
228         /* Set the return value only if status is AE_OK */
229
230         *ReturnValue = (ValueLo | ((UINT64) ValueHi << 32));
231     }
232
233     ACPI_DEBUG_PRINT ((ACPI_DB_IO,
234         "Read:  %8.8X%8.8X width %2d from %8.8X%8.8X (%s)\n",
235         ACPI_FORMAT_UINT64 (*ReturnValue), Reg->BitWidth,
236         ACPI_FORMAT_UINT64 (Address),
237         AcpiUtGetRegionName (Reg->SpaceId)));
238
239     return (AE_OK);
240     return (Status);
241 }

```

```

222 ACPI_EXPORT_SYMBOL (AcpiRead)

225 /*****
226 *
227 * FUNCTION:      AcpiWrite
228 *
229 * PARAMETERS:    Value          - Value to be written
230 *                Reg            - GAS register structure
231 *
232 * RETURN:        Status
233 *
234 * DESCRIPTION:   Write to either memory or IO space.
235 *
236 *****/

238 ACPI_STATUS
239 AcpiWrite (
240     UINT64          Value,
241     ACPI_GENERIC_ADDRESS *Reg)
242 {
243     UINT32          Width;
244     UINT64          Address;
245     ACPI_STATUS      Status;

248     ACPI_FUNCTION_NAME (AcpiWrite);

251     /* Validate contents of the GAS register. Allow 64-bit transfers */

253     Status = AcpiHwValidateRegister (Reg, 64, &Address);
254     if (ACPI_FAILURE (Status))
255     {
256         return (Status);
257     }

265     Width = Reg->BitWidth;
266     if (Width == 64)
267     {
268         Width = 32; /* Break into two 32-bit transfers */
269     }

259     /*
260     * Two address spaces supported: Memory or IO. PCI_Config is
261     * not supported here because the GAS structure is insufficient
262     */
263     if (Reg->SpaceId == ACPI_ADR_SPACE_SYSTEM_MEMORY)
264     {
265         Status = AcpiOsWriteMemory ((ACPI_PHYSICAL_ADDRESS)
266             Address, Value, Reg->BitWidth);
267         Address, ACPI_LODWORD (Value), Width);
268         if (ACPI_FAILURE (Status))
269         {
270             return (Status);
271         }
272     }
273     else /* ACPI_ADR_SPACE_SYSTEM_IO, validated earlier */

284         if (Reg->BitWidth == 64)
273     {
274         Width = Reg->BitWidth;
275         if (Width == 64)
276             Status = AcpiOsWriteMemory ((ACPI_PHYSICAL_ADDRESS)
277                 (Address + 4), ACPI_HIWORD (Value), 32);
278         if (ACPI_FAILURE (Status))

```

```

276     {
277         Width = 32; /* Break into two 32-bit transfers */
278         return (Status);
279     }

292     }
293     }
294     else /* ACPI_ADR_SPACE_SYSTEM_IO, validated earlier */
295     {
280         Status = AcpiHwWritePort ((ACPI_IO_ADDRESS)
281             Address, ACPI_LODWORD (Value), Width);
282         if (ACPI_FAILURE (Status))
283         {
284             return (Status);
285         }

287         if (Reg->BitWidth == 64)
288         {
289             Status = AcpiHwWritePort ((ACPI_IO_ADDRESS)
290                 (Address + 4), ACPI_HIWORD (Value), 32);
291             if (ACPI_FAILURE (Status))
292             {
293                 return (Status);
294             }
295         }
296     }

298     ACPI_DEBUG_PRINT ((ACPI_DB_IO,
299         "Wrote: %8.8X%8.8X width %2d to %8.8X%8.8X (%s)\n",
300         ACPI_FORMAT_UINT64 (Value), Reg->BitWidth,
301         ACPI_FORMAT_UINT64 (Address),
302         AcpiUtGetRegionName (Reg->SpaceId)));

304     return (Status);
305 }

307 ACPI_EXPORT_SYMBOL (AcpiWrite)

310 #if (!ACPI_REDUCED_HARDWARE)
311 /*****
312 *
313 * FUNCTION:      AcpiReadBitRegister
314 *
315 * PARAMETERS:    RegisterId      - ID of ACPI Bit Register to access
316 *                ReturnValue      - Value that was read from the register,
317 *                                normalized to bit position zero.
318 *
319 * RETURN:        Status and the value read from the specified Register. Value
320 *                returned is normalized to bit0 (is shifted all the way right)
321 *
322 * DESCRIPTION:   ACPI BitRegister read function. Does not acquire the HW lock.
323 *
324 * SUPPORTS:      Bit fields in PM1 Status, PM1 Enable, PM1 Control, and
325 *                PM2 Control.
326 *
327 * Note: The hardware lock is not required when reading the ACPI bit registers
328 *        since almost all of them are single bit and it does not matter that
329 *        the parent hardware register can be split across two physical
330 *        registers. The only multi-bit field is SLP_TYP in the PM1 control
331 *        register, but this field does not cross an 8-bit boundary (nor does
332 *        it make much sense to actually read this field.)
333 *
334 *****/

336 ACPI_STATUS

```



```

337 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  * FUNCTION:      AcpiWriteBitRegister
385  *
386  * PARAMETERS:    RegisterId      - ID of ACPI Bit Register to access
387  *                Value          - Value to write to the register, in bit
388  *                               position zero. The bit is automatically
389  *                               shifted to the correct position.
390  *
391  * RETURN:        Status
392  *
393  * DESCRIPTION:    ACPI Bit Register write function. Acquires the hardware lock
394  *                 since most operations require a read/modify/write sequence.
395  *
396  * SUPPORTS:       Bit fields in PM1 Status, PM1 Enable, PM1 Control, and
397  *                 PM2 Control.
398  *
399  * Note that at this level, the fact that there may be actually two
400  * hardware registers (A and B - and B may not exist) is abstracted.

```

```

402  *
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
505  * RETURN:        Status - ACPI status
506  *
507  * DESCRIPTION:   Obtain the SLP_TYPa and SLP_TYPb values for the requested
508  *                sleep state via the appropriate \_Sx object.
509  * DESCRIPTION:   Obtain the SLP_TYPa and SLP_TYPb values for the requested sleep
510  *                state.
511  *
512  * The sleep state package returned from the corresponding \_Sx object
513  * must contain at least one integer.
514  *
515  * March 2005:
516  * Added support for a package that contains two integers. This
517  * goes against the ACPI specification which defines this object as a
518  * package with one encoded DWORD integer. However, existing practice
519  * by many BIOS vendors is to return a package with 2 or more integer
520  * elements, at least one per sleep type (A/B).
521  *
522  * January 2013:
523  * Therefore, we must be prepared to accept a package with either a
524  * single integer or multiple integers.
525  *
526  * The single integer DWORD format is as follows:
527  *   BYTE 0 - Value for the PM1A SLP_TYP register
528  *   BYTE 1 - Value for the PM1B SLP_TYP register
529  *   BYTE 2-3 - Reserved
530  *
531  * The dual integer format is as follows:
532  *   Integer 0 - Value for the PM1A SLP_TYP register
533  *   Integer 1 - Value for the PM1B SLP_TYP register

```

```

531  *
532  *****/

534 ACPI_STATUS
535 AcpiGetSleepTypeData (
536     UINT8          SleepState,
537     UINT8          *SleepTypeA,
538     UINT8          *SleepTypeB)
539 {
540     ACPI_STATUS      Status;
541     ACPI_STATUS      Status = AE_OK;
542     ACPI_EVALUATE_INFO *Info;
543     ACPI_OPERAND_OBJECT **Elements;

545     ACPI_FUNCTION_TRACE (AcpiGetSleepTypeData);

547     /* Validate parameters */

549     if ((SleepState > ACPI_S_STATES_MAX) ||
550         !SleepTypeA || !SleepTypeB)
551     {
552         return ACPI_STATUS (AE_BAD_PARAMETER);
553     }

555     /* Allocate the evaluation information block */

557     Info = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_EVALUATE_INFO));
558     if (!Info)
559     {
560         return ACPI_STATUS (AE_NO_MEMORY);
561     }

563     /*
564      * Evaluate the \_Sx namespace object containing the register values
565      * for this state
566      */
567     Info->RelativePathname = ACPI_CAST_PTR (
568         char, AcpiGbl_SleepStateNames[SleepState]);
569     Info->Pathname = ACPI_CAST_PTR (char, AcpiGbl_SleepStateNames[SleepState]);

571     /* Evaluate the namespace object containing the values for this state */

573     Status = AcpiNsEvaluate (Info);
574     if (ACPI_FAILURE (Status))
575     {
576         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
577             "%s while evaluating SleepState [%s]\n",
578             AcpiFormatException (Status), Info->Pathname));

579         goto Cleanup;
580     }

582     /* Must have a return object */

584     if (!Info->ReturnObject)
585     {
586         ACPI_ERROR ((AE_INFO, "No Sleep State object returned from [%s]",
587             Info->RelativePathname));
588         Status = AE_AML_NO_RETURN_VALUE;
589         goto Cleanup;
590     }
591     Info->Pathname);
592     Status = AE_NOT_EXIST;

```

```

584     }

586     /* Return object must be of type Package */
587     /* It must be of type Package */

588     if (Info->ReturnObject->Common.Type != ACPI_TYPE_PACKAGE)
589     else if (Info->ReturnObject->Common.Type != ACPI_TYPE_PACKAGE)
590     {
591         ACPI_ERROR ((AE_INFO, "Sleep State return object is not a Package"));
592         Status = AE_AML_OPERAND_TYPE;
593         goto Cleanup1;
594     }

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.
599     * The package must have at least two elements. NOTE (March 2005): This
600     * goes against the current ACPI spec which defines this object as a
601     * package with one encoded DWORD element. However, existing practice
602     * by BIOS vendors seems to be to have 2 or more elements, at least
603     * one per sleep type (A/B).
604     */
605     Elements = Info->ReturnObject->Package.Elements;
606     switch (Info->ReturnObject->Package.Count)
607     else if (Info->ReturnObject->Package.Count < 2)
608     {
609     case 0:

610         Status = AE_AML_PACKAGE_LIMIT;
611         break;

612     case 1:

613         if (Elements[0]->Common.Type != ACPI_TYPE_INTEGER)
614         {
615             Status = AE_AML_OPERAND_TYPE;
616             break;
617             ACPI_ERROR ((AE_INFO,
618             "Sleep State return package does not have at least two elements"));
619             Status = AE_AML_NO_OPERAND;
620         }

621         /* A valid _Sx_ package with one integer */
622         /* The first two elements must both be of type Integer */

623         *SleepTypeA = (UINT8) Elements[0]->Integer.Value;
624         *SleepTypeB = (UINT8) (Elements[0]->Integer.Value >> 8);
625         break;

626     case 2:
627     default:

628         if ((Elements[0]->Common.Type != ACPI_TYPE_INTEGER) ||
629             (Elements[1]->Common.Type != ACPI_TYPE_INTEGER))
630         else if (((Info->ReturnObject->Package.Elements[0])->Common.Type
631             != ACPI_TYPE_INTEGER) ||
632             ((Info->ReturnObject->Package.Elements[1])->Common.Type
633             != ACPI_TYPE_INTEGER))
634         {
635             ACPI_ERROR ((AE_INFO,
636             "Sleep State return package elements are not both Integers "
637             "{%s, %s}",
638             AcpiUtGetObjectTypeName (Info->ReturnObject->Package.Elements[0]),
639             AcpiUtGetObjectTypeName (Info->ReturnObject->Package.Elements[1]));
640             Status = AE_AML_OPERAND_TYPE;

```

```

629         break;
630     }
631     else
632     {
633         /* Valid _Sx_ package size, type, and value */

634         /* A valid _Sx_ package with two integers */

635         *SleepTypeA = (UINT8) Elements[0]->Integer.Value;
636         *SleepTypeB = (UINT8) Elements[1]->Integer.Value;
637         break;
638         *SleepTypeA = (UINT8)
639             (Info->ReturnObject->Package.Elements[0])->Integer.Value;
640         *SleepTypeB = (UINT8)
641             (Info->ReturnObject->Package.Elements[1])->Integer.Value;
642     }

643 Cleanup1:
644     AcpiUtRemoveReference (Info->ReturnObject);

645 Cleanup:
646     if (ACPI_FAILURE (Status))
647     {
648         ACPI_EXCEPTION ((AE_INFO, Status,
649         "While evaluating Sleep State [%s]", Info->RelativePathname));
650         "While evaluating SleepState [%s], bad Sleep object %p type %s",
651         Info->Pathname, Info->ReturnObject,
652         AcpiUtGetObjectTypeName (Info->ReturnObject));
653     }

654     AcpiUtRemoveReference (Info->ReturnObject);

655 Cleanup:
656     ACPI_FREE (Info);
657     return ACPI_STATUS (Status);
658 }

unchanged_portion_omitted

```

new/usr/src/common/acpica/components/hardware/hwxfsleep.c

1

```
*****
13990 Fri Jan 17 07:30:22 2014
new/usr/src/common/acpica/components/hardware/hwxfsleep.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Name: hwxfsleep.c - ACPI Hardware Sleep/Wake External Interfaces
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 EXPORT_ACPI_INTERFACES

46 #include "acpi.h"
47 #include "accommon.h"

49 #define _COMPONENT          ACPI_HARDWARE
50       ACPI_MODULE_NAME    ("hwxfsleep")

52 /* Local prototypes */

54 static ACPI_STATUS
55 AcpiHwSleepDispatch (
56     UINT8                SleepState,
57     UINT32                FunctionId);

59 /*
```

new/usr/src/common/acpica/components/hardware/hwxfsleep.c

2

```
60 * Dispatch table used to efficiently branch to the various sleep
61 * functions.
62 */
63 #define ACPI_SLEEP_FUNCTION_ID          0
64 #define ACPI_WAKE_PREP_FUNCTION_ID      1
65 #define ACPI_WAKE_FUNCTION_ID          2

67 /* Legacy functions are optional, based upon ACPI_REduced_HARDWARE */

69 static ACPI_SLEEP_FUNCTIONS      AcpiSleepDispatch[] =
70 {
71     {ACPI_HW_OPTIONAL_FUNCTION (AcpiHwLegacySleep),      AcpiHwExtendedSleep},
72     {ACPI_HW_OPTIONAL_FUNCTION (AcpiHwLegacyWakePrep),   AcpiHwExtendedWakePrep},
73     {ACPI_HW_OPTIONAL_FUNCTION (AcpiHwLegacyWake),       AcpiHwExtendedWake}
74 };

77 /*
78 * These functions are removed for the ACPI_REduced_HARDWARE case:
79 *     AcpiSetFirmwareWakingVector
80 *     AcpiSetFirmwareWakingVector64
81 *     AcpiEnterSleepStateS4bios
82 */

84 #if (!ACPI_REduced_HARDWARE)
85 /*****
86 *
87 * FUNCTION:      AcpiSetFirmwareWakingVector
88 *
89 * PARAMETERS:    PhysicalAddress      - 32-bit physical address of ACPI real mode
90 *                                           entry point.
91 *
92 * RETURN:        Status
93 *
94 * DESCRIPTION:   Sets the 32-bit FirmwareWakingVector field of the FACS
95 *
96 *****/

98 ACPI_STATUS
99 AcpiSetFirmwareWakingVector (
100     UINT32                PhysicalAddress)
101 {
102     ACPI_FUNCTION_TRACE (AcpiSetFirmwareWakingVector);

105     /*
106     * According to the ACPI specification 2.0c and later, the 64-bit
107     * waking vector should be cleared and the 32-bit waking vector should
108     * be used, unless we want the wake-up code to be called by the BIOS in
109     * Protected Mode. Some systems (for example HP dv5-i004nr) are known
110     * to fail to resume if the 64-bit vector is used.
111     */

113     /* Set the 32-bit vector */

115     AcpiGbl_FACS->FirmwareWakingVector = PhysicalAddress;

117     /* Clear the 64-bit vector if it exists */

119     if ((AcpiGbl_FACS->Length > 32) && (AcpiGbl_FACS->Version >= 1))
120     {
121         AcpiGbl_FACS->XFirmwareWakingVector = 0;
122     }

124     return ACPI_STATUS (AE_OK);
125 }
```

```

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.
259 *
260 *****/

262 static ACPI_STATUS
263 AcpiHwSleepDispatch (
264     UINT8          SleepState,
265     UINT32          FunctionId)
266 {
267     ACPI_STATUS      Status;
268     ACPI_SLEEP_FUNCTIONS *SleepFunctions = &AcpiSleepDispatch[FunctionId];

271 #if (!ACPI_REduced_HARDWARE)
272 /*
273  * If the Hardware Reduced flag is set (from the FADT), we must
274  * use the extended sleep registers (FADT). Note: As per the ACPI
275  * specification, these extended registers are to be used for HW-reduced
276  * platforms only. They are not general-purpose replacements for the
277  * legacy PM register sleep support.
278  */
279 if (AcpiGbl_ReducedHardware)
280 {
281     Status = SleepFunctions->ExtendedFunction (SleepState);
282 }
283 else
284 {
285     /* Legacy sleep */

287     Status = SleepFunctions->LegacyFunction (SleepState);
288 }

290 return (Status);

292 #else
293 /*
294  * For the case where reduced-hardware-only code is being generated,
295  * we know that only the extended sleep registers are available
296  */
297 Status = SleepFunctions->ExtendedFunction (SleepState);
298 return (Status);

300 #endif /* !ACPI_REduced_HARDWARE */
301 }

304 /*****
305  * FUNCTION:  AcpiEnterSleepStatePrep
306  *
307  * PARAMETERS: SleepState - Which sleep state to enter
308  *
309  * RETURN:     Status
310  *
311  * DESCRIPTION: Prepare to enter a system sleep state.
312  * This function must execute with interrupts enabled.
313  * We break sleeping into 2 stages so that OSPM can handle
314  * various OS-specific tasks between the two steps.
315  *
316  *****/

319 ACPI_STATUS
320 AcpiEnterSleepStatePrep (
321     UINT8          SleepState)
322 {
323     ACPI_STATUS      Status;

```

```

324     ACPI_OBJECT_LIST ArgList;
325     ACPI_OBJECT      Arg;
326     UINT32           SstValue;

329     ACPI_FUNCTION_TRACE (AcpiEnterSleepStatePrep);

332     Status = AcpiGetSleepTypeData (SleepState,
333                                     &AcpiGbl_SleepTypeA, &AcpiGbl_SleepTypeB);
334     if (ACPI_FAILURE (Status))
335     {
336         return ACPI_STATUS (Status);
337     }

339     /* Execute the _PTS method (Prepare To Sleep) */

341     ArgList.Count = 1;
342     ArgList.Pointer = &Arg;
343     Arg.Type = ACPI_TYPE_INTEGER;
344     Arg.Integer.Value = SleepState;

346     Status = AcpiEvaluateObject (NULL, METHOD_PATHNAME__PTS, &ArgList, NULL);
347     if (ACPI_FAILURE (Status) && Status != AE_NOT_FOUND)
348     {
349         return ACPI_STATUS (Status);
350     }

352     /* Setup the argument to the _SST method (System Status) */

354     switch (SleepState)
355     {
356     case ACPI_STATE_S0:

358         SstValue = ACPI_SST_WORKING;
359         break;

361     case ACPI_STATE_S1:
362     case ACPI_STATE_S2:
363     case ACPI_STATE_S3:

365         SstValue = ACPI_SST_SLEEPING;
366         break;

368     case ACPI_STATE_S4:

370         SstValue = ACPI_SST_SLEEP_CONTEXT;
371         break;

373     default:

375         SstValue = ACPI_SST_INDICATOR_OFF; /* Default is off */
376         break;
377     }

379     /*
380     * Set the system indicators to show the desired sleep state.
381     * _SST is an optional method (return no error if not found)
382     */
383     AcpiHwExecuteSleepMethod (METHOD_PATHNAME__SST, SstValue);
384     return ACPI_STATUS (AE_OK);
385 }

387 ACPI_EXPORT_SYMBOL (AcpiEnterSleepStatePrep)

```

```

390 /*****
391 *
392 * FUNCTION:      AcpiEnterSleepState
393 *
394 * PARAMETERS:    SleepState          - Which sleep state to enter
395 *
396 * RETURN:        Status
397 *
398 * DESCRIPTION:   Enter a system sleep state
399 *                THIS FUNCTION MUST BE CALLED WITH INTERRUPTS DISABLED
400 *
401 *****/
403 ACPI_STATUS
404 AcpiEnterSleepState (
405     UINT8                SleepState)
406 {
407     ACPI_STATUS           Status;

410     ACPI_FUNCTION_TRACE (AcpiEnterSleepState);

413     if ((AcpiGbl_SleepTypeA > ACPI_SLEEP_TYPE_MAX) ||
414         (AcpiGbl_SleepTypeB > ACPI_SLEEP_TYPE_MAX))
415     {
416         ACPI_ERROR ((AE_INFO, "Sleep values out of range: A=0x%X B=0x%X",
417             AcpiGbl_SleepTypeA, AcpiGbl_SleepTypeB));
418         return_ACPI_STATUS (AE_AML_OPERAND_VALUE);
419     }

421     Status = AcpiHwSleepDispatch (SleepState, ACPI_SLEEP_FUNCTION_ID);
422     return_ACPI_STATUS (Status);
423 }

425 ACPI_EXPORT_SYMBOL (AcpiEnterSleepState)

428 /*****
429 *
430 * FUNCTION:      AcpiLeaveSleepStatePrep
431 *
432 * PARAMETERS:    SleepState          - Which sleep state we are exiting
433 *
434 * RETURN:        Status
435 *
436 * DESCRIPTION:   Perform the first state of OS-independent ACPI cleanup after a
437 *                sleep. Called with interrupts DISABLED.
438 *                We break wake/resume into 2 stages so that OSPM can handle
439 *                various OS-specific tasks between the two steps.
440 *
441 *****/
443 ACPI_STATUS
444 AcpiLeaveSleepStatePrep (
445     UINT8                SleepState)
446 {
447     ACPI_STATUS           Status;

450     ACPI_FUNCTION_TRACE (AcpiLeaveSleepStatePrep);

453     Status = AcpiHwSleepDispatch (SleepState, ACPI_WAKE_PREP_FUNCTION_ID);
454     return_ACPI_STATUS (Status);
455 }

```

```

457 ACPI_EXPORT_SYMBOL (AcpiLeaveSleepStatePrep)

460 /*****
461 *
462 * FUNCTION:      AcpiLeaveSleepState
463 *
464 * PARAMETERS:    SleepState          - Which sleep state we are exiting
465 *
466 * RETURN:        Status
467 *
468 * DESCRIPTION:   Perform OS-independent ACPI cleanup after a sleep
469 *                Called with interrupts ENABLED.
470 *
471 *****/
473 ACPI_STATUS
474 AcpiLeaveSleepState (
475     UINT8                SleepState)
476 {
477     ACPI_STATUS           Status;

480     ACPI_FUNCTION_TRACE (AcpiLeaveSleepState);

483     Status = AcpiHwSleepDispatch (SleepState, ACPI_WAKE_FUNCTION_ID);
484     return_ACPI_STATUS (Status);
485 }

487 ACPI_EXPORT_SYMBOL (AcpiLeaveSleepState)

```

new/usr/src/common/acpica/components/namespace/nsaccess.c

1

```
*****
22575 Fri Jan 17 07:30:22 2014
new/usr/src/common/acpica/components/namespace/nsaccess.c
update to acpica-unix2-20140114
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 - 2014, 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 MERCHANTABILITY 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 __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 *
```

new/usr/src/common/acpica/components/namespace/nsaccess.c

2

```
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
126     {
127         ACPI_EXCEPTION ((AE_INFO, Status,
128             "Could not create predefined name %s",
129             InitVal->Name));
130     }
131     continue;
132 }
133
134 /*
135  * Name entered successfully. If entry in PreDefinedNames[] specifies
136  * an initial value, create the initial value.
137  */
138 if (InitVal->Val)
139 {
140     Status = AcpiOsPredefinedOverride (InitVal, &Val);
141     if (ACPI_FAILURE (Status))
142     {
143         ACPI_ERROR ((AE_INFO,
144             "Could not override predefined %s",
145             InitVal->Name));
146     }
147
148     if (!Val)
149     {
150         Val = InitVal->Val;
151     }
152
153     /*
154     * Entry requests an initial value, allocate a
155     * descriptor for it.
156     */
157     ObjDesc = AcpiUtCreateInternalObject (InitVal->Type);
158     if (!ObjDesc)
159     {
160         Status = AE_NO_MEMORY;
161         goto UnlockAndExit;
162     }
163
164     /*
165     * Convert value string from table entry to
166     * internal representation. Only types actually
167     * used for initial values are implemented here.
168     */
169     switch (InitVal->Type)
170     {
171     case ACPI_TYPE_METHOD:
172         ObjDesc->Method.ParamCount = (UINT8) ACPI_TO_INTEGER (Val);
173         ObjDesc->Common.Flags |= AOPOBJ_DATA_VALID;
174
175 #if defined (ACPI_ASL_COMPILER)
176         /* Save the parameter count for the iASL compiler */
177         NewNode->Value = ObjDesc->Method.ParamCount;
178 #else
179         /* Mark this as a very SPECIAL method */
180
181         ObjDesc->Method.InfoFlags = ACPI_METHOD_INTERNAL_ONLY;
182         ObjDesc->Method.Dispatch.Implementation = AcpiUtOsiImplementatio
183 #endif
184         break;
185
186     case ACPI_TYPE_INTEGER:

```

```

189         ObjDesc->Integer.Value = ACPI_TO_INTEGER (Val);
190         break;
191
192     case ACPI_TYPE_STRING:
193
194         /* Build an object around the static string */
195
196         ObjDesc->String.Length = (UINT32) ACPI_STRLLEN (Val);
197         ObjDesc->String.Pointer = Val;
198         ObjDesc->Common.Flags |= AOPOBJ_STATIC_POINTER;
199         break;
200
201     case ACPI_TYPE_MUTEX:
202
203         ObjDesc->Mutex.Node = NewNode;
204         ObjDesc->Mutex.SyncLevel = (UINT8) (ACPI_TO_INTEGER (Val) - 1);
205
206         /* Create a mutex */
207
208         Status = AcpiOsCreateMutex (&ObjDesc->Mutex.OsMutex);
209         if (ACPI_FAILURE (Status))
210         {
211             AcpiUtRemoveReference (ObjDesc);
212             goto UnlockAndExit;
213         }
214
215         /* Special case for ACPI Global Lock */
216
217         if (ACPI_STRCMP (InitVal->Name, "_GL_") == 0)
218         {
219             AcpiGbl_GlobalLockMutex = ObjDesc;
220
221             /* Create additional counting semaphore for global lock */
222
223             Status = AcpiOsCreateSemaphore (
224                 1, 0, &AcpiGbl_GlobalLockSemaphore);
225             if (ACPI_FAILURE (Status))
226             {
227                 AcpiUtRemoveReference (ObjDesc);
228                 goto UnlockAndExit;
229             }
230         }
231         break;
232
233     default:
234
235         ACPI_ERROR ((AE_INFO, "Unsupported initial type value 0x%X",
236             InitVal->Type));
237         AcpiUtRemoveReference (ObjDesc);
238         ObjDesc = NULL;
239         continue;
240     }
241
242     /* Store pointer to value descriptor in the Node */
243
244     Status = AcpiNsAttachObject (NewNode, ObjDesc,
245         ObjDesc->Common.Type);
246
247     /* Remove local reference to the object */
248
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 */
405
406         ThisNode = AcpiGbl_RootNode;
407         SearchParentFlag = ACPI_NS_NO_UPSEARCH;
408
409         /* Point to name segment part */
410
411         Path++;
412
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 */
419
420         ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
421             "Searching relative to prefix scope [%4.4s] (%p)\n",
422             AcpiUtGetNodeName (PrefixNode), PrefixNode));
423
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 */
433
434             SearchParentFlag = ACPI_NS_NO_UPSEARCH;
435
436             /*
437             * Point past this prefix to the name segment
438             * part or the next Parent Prefix
439             */
440             Path++;
441
442             /* Backup to the parent node */
443
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");
455     return ACPI_STATUS (AE_NOT_FOUND);
456 }
457
458 if (SearchParentFlag == ACPI_NS_NO_UPSEARCH)
459 {
460     ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
461         "Search scope is [%4.4s], path has %u carat(s)\n",
462         AcpiUtGetNodeName (ThisNode), NumCarats));
463 }
464
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;
487
488     ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
489         "Prefix-only Pathname (Zero name segments), Flags=%X\n",
490         Flags));
491     break;
492
493 case AML_DUAL_NAME_PREFIX:
494
495     /* More than one NameSeg, search rules do not apply */
496
497     SearchParentFlag = ACPI_NS_NO_UPSEARCH;
498
499     /* Two segments, point to first name segment */
500
501     NumSegments = 2;
502     Path++;
503
504     ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
505         "Dual Pathname (2 segments, Flags=%X)\n", Flags));
506     break;
507
508 case AML_MULTI_NAME_PREFIX_OP:
509
510     /* More than one NameSeg, search rules do not apply */
511
512     SearchParentFlag = ACPI_NS_NO_UPSEARCH;

```

```

514      /* Extract segment count, point to first name segment */
515
516      Path++;
517      NumSegments = (UINT32) (UINT8) *Path;
518      Path++;
519
520      ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
521        "Multi Pathname (%u Segments, Flags=%X)\n",
522        NumSegments, Flags));
523      break;
524
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;
531
532      ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
533        "Simple Pathname (1 segment, Flags=%X)\n", Flags));
534      break;
535      }
536
537      ACPI_DEBUG_EXEC (AcpiNsPrintPathname (NumSegments, Path));
538  }
539
540  /*
541   * Search namespace for each segment of the name. Loop through and
542   * verify (or add to the namespace) each name segment.
543   *
544   * The object type is significant only at the last name
545   * segment. (We don't care about the types along the path, only
546   * the type of the final target object.)
547   */
548  ThisSearchType = ACPI_TYPE_ANY;
549  CurrentNode = ThisNode;
550  while (NumSegments && CurrentNode)
551  {
552      NumSegments--;
553      if (!NumSegments)
554      {
555          /* This is the last segment, enable typechecking */
556
557          ThisSearchType = Type;
558
559          /*
560           * Only allow automatic parent search (search rules) if the caller
561           * requested it AND we have a single, non-fully-qualified NameSeg
562           */
563          if ((SearchParentFlag != ACPI_NS_NO_UPSEARCH) &&
564            (Flags & ACPI_NS_SEARCH_PARENT))
565          {
566              LocalFlags |= ACPI_NS_SEARCH_PARENT;
567          }
568
569          /* Set error flag according to caller */
570
571          if (Flags & ACPI_NS_ERROR_IF_FOUND)
572          {
573              LocalFlags |= ACPI_NS_ERROR_IF_FOUND;
574          }
575      }
576
577      /* Extract one ACPI name from the front of the pathname */

```

```

580      ACPI_MOVE_32_TO_32 (&SimpleName, Path);
581
582      /* Try to find the single (4 character) ACPI name */
583
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 */
591
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          }
597
598          *ReturnNode = ThisNode;
599          return ACPI_STATUS (Status);
600      }
601
602      /* More segments to follow? */
603
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              }
618
619              if (AcpiNsOpensScope ((ACPI_NAMESPACE_NODE *)
620                ThisNode->Object)->Type))
621              {
622                  ThisNode = (ACPI_NAMESPACE_NODE *) ThisNode->Object;
623              }
624          }
625      }
626
627      /* Special handling for the last segment (NumSegments == 0) */
628
629      else
630      {
631          /*
632           * Sanity typecheck of the target object:
633           *
634           * 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 */
652
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     }
659
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
670     /* Point to next name segment and make this node current */
671
672     Path += ACPI_NAME_SIZE;
673     CurrentNode = ThisNode;
674 }
675
676 /* Always check if we need to open a new scope */
677
678 if (!(Flags & ACPI_NS_DONT_OPEN_SCOPE) && (WalkState))
679 {
680     /*
681     * If entry is a type which opens a scope, push the new scope on the
682     * scope stack.
683     */
684     if (AcpiNsOpensScope (Type))
685     {
686         Status = AcpiDsScopeStackPush (ThisNode, Type, WalkState);
687         if (ACPI_FAILURE (Status))
688         {
689             return_ACPI_STATUS (Status);
690         }
691     }
692 }
693
694 *ReturnNode = ThisNode;
695 return_ACPI_STATUS (AE_OK);
696 }
697 }
```

new/usr/src/common/acpica/components/namespace/nsalloc.c

1

```
*****
16136 Fri Jan 17 07:30:22 2014
new/usr/src/common/acpica/components/namespace/nsalloc.c
update to acpica-unix2-20140114
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: nsalloc - Namespace allocation and deletion utilities
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #define __NSALLOC_C__
46
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acnamespace.h"
50
51
52 #define _COMPONENT ACPI_NAMESPACE
53 #define ACPI_MODULE_NAME("nsalloc")
54
55 /*****
56 *
57 *****/
```

new/usr/src/common/acpica/components/namespace/nsalloc.c

2

```
58 * FUNCTION: AcpiNsCreateNode
59 *
60 * PARAMETERS: Name - Name of the new node (4 char ACPI name)
61 *
62 * RETURN: New namespace node (Null on failure)
63 *
64 * DESCRIPTION: Create a namespace node
65 *
66 *****/
67
68 ACPI_NAMESPACE_NODE *
69 AcpiNsCreateNode (
70     UINT32 Name)
71 {
72     ACPI_NAMESPACE_NODE *Node;
73     #ifdef ACPI_DBG_TRACK_ALLOCATIONS
74     UINT32 Temp;
75     #endif
76
77     ACPI_FUNCTION_TRACE (NsCreateNode);
78
79     Node = AcpiOsAcquireObject (AcpiGbl_NamespaceCache);
80     if (!Node)
81     {
82         return_PTR (NULL);
83     }
84
85     ACPI_MEM_TRACKING (AcpiGbl_NsNodeList->TotalAllocated++);
86
87     #ifdef ACPI_DBG_TRACK_ALLOCATIONS
88     Temp = AcpiGbl_NsNodeList->TotalAllocated -
89           AcpiGbl_NsNodeList->TotalFreed;
90     if (Temp > AcpiGbl_NsNodeList->MaxOccupied)
91     {
92         AcpiGbl_NsNodeList->MaxOccupied = Temp;
93     }
94     #endif
95
96     Node->Name.Integer = Name;
97     ACPI_SET_DESCRIPTOR_TYPE (Node, ACPI_DESC_TYPE_NAMED);
98     return_PTR (Node);
99 }
100
101
102 /*****
103 *
104 * FUNCTION: AcpiNsDeleteNode
105 *
106 * PARAMETERS: Node - Node to be deleted
107 *
108 * RETURN: None
109 *
110 * DESCRIPTION: Delete a namespace node. All node deletions must come through
111 * here. Detaches any attached objects, including any attached
112 * data. If a handler is associated with attached data, it is
113 * invoked before the node is deleted.
114 *
115 *****/
116
117 void
118 AcpiNsDeleteNode (
119     ACPI_NAMESPACE_NODE *Node)
120 {
121     ACPI_OPERAND_OBJECT *ObjDesc;
122
123     #ifdef ACPI_DBG_TRACK_ALLOCATIONS
```

```
124     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.
139     * Delete an attached data object if present (an object that was created
140     * and attached via AcpiAttachData). Note: After any normal object is
141     * detached above, the only possible remaining object is a data object.
142     */
143     ObjDesc = Node->Object;
144     while (ObjDesc &&
145           if (ObjDesc &&
146               (ObjDesc->Common.Type == ACPI_TYPE_LOCAL_DATA))
147       {
148         /* Invoke the attached data deletion handler if present */

149         if (ObjDesc->Data.Handler)
150         {
151           ObjDesc->Data.Handler (Node, ObjDesc->Data.Pointer);
152         }

153         NextDesc = ObjDesc->Common.NextObject;
154         AcpiUtRemoveReference (ObjDesc);
155         ObjDesc = NextDesc;
156       }

157     /* Special case for the statically allocated root node */

158     if (Node == AcpiGbl_RootNode)
159     {
160       return;
161     }

162     /* Now we can delete the node */

163     (void) AcpiOsReleaseObject (AcpiGbl_NamespaceCache, Node);

164     ACPI_MEM_TRACKING (AcpiGbl_NsNodeList->TotalFreed++);
165     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "Node %p, Remaining %X\n",
166                       Node, AcpiGbl_CurrentNodeCount));
167 }

168 _unchanged_portion_omitted_
```

new/usr/src/common/acpica/components/namespace/nsarguments.c

1

```
*****
10923 Fri Jan 17 07:30:23 2014
new/usr/src/common/acpica/components/namespace/nsarguments.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: nsarguments - Validation of args for ACPI predefined methods
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acnamesp.h"
47 #include "acpredef.h"

50 #define _COMPONENT          ACPI_NAMESPACE
51 #define ACPI_MODULE_NAME    ("nsarguments")

54 /*****
55 *
56 * FUNCTION:      AcpiNsCheckArgumentTypes
57 *
58 * PARAMETERS:    Info - Method execution information block
59 *
60 * RETURN:        None

```

new/usr/src/common/acpica/components/namespace/nsarguments.c

2

```
61 *
62 * DESCRIPTION: Check the incoming argument count and all argument types
63 *               against the argument type list for a predefined name.
64 *
65 *****/

67 void
68 AcpiNsCheckArgumentTypes (
69     ACPI_EVALUATE_INFO    *Info)
70 {
71     UINT16                ArgTypeList;
72     UINT8                  ArgCount;
73     UINT8                  ArgType;
74     UINT8                  UserArgType;
75     UINT32                 i;

78     /* If not a predefined name, cannot typecheck args */

80     if (!Info->Predefined)
81     {
82         return;
83     }

85     ArgTypeList = Info->Predefined->Info.ArgumentList;
86     ArgCount = METHOD_GET_ARG_COUNT (ArgTypeList);

88     /* Typecheck all arguments */

90     for (i = 0; ((i < ArgCount) && (i < Info->ParamCount)); i++)
91     {
92         ArgType = METHOD_GET_NEXT_TYPE (ArgTypeList);
93         UserArgType = Info->Parameters[i]->Common.Type;

95         if (UserArgType != ArgType)
96         {
97             ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, ACPI_WARN_ALWAYS
98                 "Argument #%u type mismatch - "
99                 "Found [%s], ACPI requires [%s]", (i + 1),
100                 AcpiUtGetTypeName (UserArgType),
101                 AcpiUtGetTypeName (ArgType)));
102         }
103     }
104 }

107 /*****
108 *
109 * FUNCTION:      AcpiNsCheckAcpiCompliance
110 *
111 * PARAMETERS:    Pathname - Full pathname to the node (for error msgs)
112 *                Node      - Namespace node for the method/object
113 *                Predefined - Pointer to entry in predefined name table
114 *
115 * RETURN:        None
116 *
117 * DESCRIPTION: Check that the declared parameter count (in ASL/AML) for a
118 *               predefined name is what is expected (matches what is defined in
119 *               the ACPI specification for this predefined name.)
120 *
121 *****/

123 void
124 AcpiNsCheckAcpiCompliance (
125     char                *Pathname,
126     ACPI_NAMESPACE_NODE *Node,

```



```

127     const ACPI_PREDEFINED_INFO *Predefined)
128 {
129     UINT32          AmlParamCount;
130     UINT32          RequiredParamCount;

133     if (!Predefined)
134     {
135         return;
136     }

138     /* Get the ACPI-required arg count from the predefined info table */

140     RequiredParamCount = METHOD_GET_ARG_COUNT (Predefined->Info.ArgumentList);

142     /*
143      * If this object is not a control method, we can check if the ACPI
144      * spec requires that it be a method.
145      */
146     if (Node->Type != ACPI_TYPE_METHOD)
147     {
148         if (RequiredParamCount > 0)
149         {
150             /* Object requires args, must be implemented as a method */

152             ACPI_BIOS_ERROR_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
153             "Object (%s) must be a control method with %u arguments",
154             AcpiUtGetTypeName (Node->Type), RequiredParamCount));
155         }
156         else if (!RequiredParamCount && !Predefined->Info.ExpectedBtypes)
157         {
158             /* Object requires no args and no return value, must be a method */

160             ACPI_BIOS_ERROR_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
161             "Object (%s) must be a control method "
162             "with no arguments and no return value",
163             AcpiUtGetTypeName (Node->Type)));
164         }

166         return;
167     }

169     /*
170      * This is a control method.
171      * Check that the ASL/AML-defined parameter count for this method
172      * matches the ACPI-required parameter count
173      *
174      * Some methods are allowed to have a "minimum" number of args (_SCP)
175      * because their definition in ACPI has changed over time.
176      *
177      * Note: These are BIOS errors in the declaration of the object
178      */
179     AmlParamCount = Node->Object->Method.ParamCount;

181     if (AmlParamCount < RequiredParamCount)
182     {
183         ACPI_BIOS_ERROR_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
184         "Insufficient arguments - "
185         "ASL declared %u, ACPI requires %u",
186         AmlParamCount, RequiredParamCount));
187     }
188     else if ((AmlParamCount > RequiredParamCount) &&
189             !(Predefined->Info.ArgumentList & ARG_COUNT_IS_MINIMUM))
190     {
191         ACPI_BIOS_ERROR_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
192         "Excess arguments - "

```

```

193         "ASL declared %u, ACPI requires %u",
194         AmlParamCount, RequiredParamCount));
195     }
196 }

199 /*****
200  *
201  * FUNCTION:      AcpiNsCheckArgumentCount
202  *
203  * PARAMETERS:    Pathname      - Full pathname to the node (for error msgs)
204  *                Node          - Namespace node for the method/object
205  *                UserParamCount - Number of args passed in by the caller
206  *                Predefined     - Pointer to entry in predefined name table
207  *
208  * RETURN:        None
209  *
210  * DESCRIPTION:   Check that incoming argument count matches the declared
211  *                parameter count (in the ASL/AML) for an object.
212  *
213  *****/

215 void
216 AcpiNsCheckArgumentCount (
217     char            *Pathname,
218     ACPI_NAMESPACE_NODE *Node,
219     UINT32          UserParamCount,
220     const ACPI_PREDEFINED_INFO *Predefined)
221 {
222     UINT32          AmlParamCount;
223     UINT32          RequiredParamCount;

226     if (!Predefined)
227     {
228         /*
229          * Not a predefined name. Check the incoming user argument count
230          * against the count that is specified in the method/object.
231          */
232         if (Node->Type != ACPI_TYPE_METHOD)
233         {
234             if (UserParamCount)
235             {
236                 ACPI_INFO_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
237                 "%u arguments were passed to a non-method ACPI object (%s)",
238                 UserParamCount, AcpiUtGetTypeName (Node->Type)));
239             }

241             return;
242         }

244         /*
245          * This is a control method. Check the parameter count.
246          * We can only check the incoming argument count against the
247          * argument count declared for the method in the ASL/AML.
248          *
249          * Emit a message if too few or too many arguments have been passed
250          * by the caller.
251          *
252          * Note: Too many arguments will not cause the method to
253          * fail. However, the method will fail if there are too few
254          * arguments and the method attempts to use one of the missing ones.
255          */
256         AmlParamCount = Node->Object->Method.ParamCount;

258         if (UserParamCount < AmlParamCount)

```

```
259     {
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     }
272
273     return;
274 }
275
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);
287
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 }
```

new/usr/src/common/acpica/components/namespace/nsconvert.c

1

```
*****
14083 Fri Jan 17 07:30:23 2014
new/usr/src/common/acpica/components/namespace/nsconvert.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: nsconvert - Object conversions for objects returned by
4 * predefined methods
5 *
6 *****/

8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __NSCONVERT_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acnamesp.h"
50 #include "acinterp.h"
51 #include "acpredef.h"
52 #include "amlresrc.h"

54 #define _COMPONENT          ACPI_NAMESPACE
55       ACPI_MODULE_NAME    ("nsconvert")

58 /*****
59 *
60 * FUNCTION:      AcpiNsConvertToInteger
```

new/usr/src/common/acpica/components/namespace/nsconvert.c

2

```
61 *
62 * PARAMETERS:   OriginalObject      - Object to be converted
63 *               ReturnObject        - Where the new converted object is returned
64 *
65 * RETURN:       Status. AE_OK if conversion was successful.
66 *
67 * DESCRIPTION:  Attempt to convert a String/Buffer object to an Integer.
68 *
69 *****/

71 ACPI_STATUS
72 AcpiNsConvertToInteger (
73     ACPI_OPERAND_OBJECT *OriginalObject,
74     ACPI_OPERAND_OBJECT **ReturnObject)
75 {
76     ACPI_OPERAND_OBJECT *NewObject;
77     ACPI_STATUS Status;
78     UINT64 Value = 0;
79     UINT32 i;

82     switch (OriginalObject->Common.Type)
83     {
84     case ACPI_TYPE_STRING:

86         /* String-to-Integer conversion */

88         Status = AcpiUtStrtoul64 (OriginalObject->String.Pointer,
89                                 ACPI_ANY_BASE, &Value);
90         if (ACPI_FAILURE (Status))
91         {
92             return (Status);
93         }
94         break;

96     case ACPI_TYPE_BUFFER:

98         /* Buffer-to-Integer conversion. Max buffer size is 64 bits. */

100         if (OriginalObject->Buffer.Length > 8)
101         {
102             return (AE_AML_OPERAND_TYPE);
103         }

105         /* Extract each buffer byte to create the integer */

107         for (i = 0; i < OriginalObject->Buffer.Length; i++)
108         {
109             Value |= ((UINT64) OriginalObject->Buffer.Pointer[i] << (i * 8));
110         }
111         break;

113     default:

115         return (AE_AML_OPERAND_TYPE);
116     }

118     NewObject = AcpiUtCreateIntegerObject (Value);
119     if (!NewObject)
120     {
121         return (AE_NO_MEMORY);
122     }

124     *ReturnObject = NewObject;
125     return (AE_OK);
126 }
```

```

129 /*****
130 *
131 * FUNCTION:      AcpiNsConvertToString
132 *
133 * PARAMETERS:    OriginalObject      - Object to be converted
134 *                ReturnObject        - Where the new converted object is returned
135 *
136 * RETURN:        Status. AE_OK if conversion was successful.
137 *
138 * DESCRIPTION:    Attempt to convert a Integer/Buffer object to a String.
139 *
140 *****/
142 ACPI_STATUS
143 AcpiNsConvertToString (
144     ACPI_OPERAND_OBJECT *OriginalObject,
145     ACPI_OPERAND_OBJECT **ReturnObject)
146 {
147     ACPI_OPERAND_OBJECT *NewObject;
148     ACPI_SIZE Length;
149     ACPI_STATUS Status;

152     switch (OriginalObject->Common.Type)
153     {
154     case ACPI_TYPE_INTEGER:
155         /*
156          * Integer-to-String conversion. Commonly, convert
157          * an integer of value 0 to a NULL string. The last element of
158          * _BIF and _BIX packages occasionally need this fix.
159          */
160         if (OriginalObject->Integer.Value == 0)
161         {
162             /* Allocate a new NULL string object */

164             NewObject = AcpiUtCreateStringObject (0);
165             if (!NewObject)
166             {
167                 return (AE_NO_MEMORY);
168             }
169         }
170     else
171     {
172         Status = AcpiExConvertToString (OriginalObject, &NewObject,
173                                         ACPI_IMPLICIT_CONVERT_HEX);
174         if (ACPI_FAILURE (Status))
175         {
176             return (Status);
177         }
178     }
179     break;

181     case ACPI_TYPE_BUFFER:
182         /*
183          * Buffer-to-String conversion. Use a ToString
184          * conversion, no transform performed on the buffer data. The best
185          * example of this is the _BIF method, where the string data from
186          * the battery is often (incorrectly) returned as buffer object(s).
187          */
188         Length = 0;
189         while ((Length < OriginalObject->Buffer.Length) &&
190              (OriginalObject->Buffer.Pointer[Length]))
191         {
192             Length++;

```

```

193     }
195     /* Allocate a new string object */
197     NewObject = AcpiUtCreateStringObject (Length);
198     if (!NewObject)
199     {
200         return (AE_NO_MEMORY);
201     }
203     /*
204     * Copy the raw buffer data with no transform. String is already NULL
205     * terminated at Length+1.
206     */
207     ACPI_MEMCPY (NewObject->String.Pointer,
208                 OriginalObject->Buffer.Pointer, Length);
209     break;

211     default:

213         return (AE_AML_OPERAND_TYPE);
214     }

216     *ReturnObject = NewObject;
217     return (AE_OK);
218 }

221 /*****
222 *
223 * FUNCTION:      AcpiNsConvertToBuffer
224 *
225 * PARAMETERS:    OriginalObject      - Object to be converted
226 *                ReturnObject        - Where the new converted object is returned
227 *
228 * RETURN:        Status. AE_OK if conversion was successful.
229 *
230 * DESCRIPTION:    Attempt to convert a Integer/String/Package object to a Buffer.
231 *
232 *****/
234 ACPI_STATUS
235 AcpiNsConvertToBuffer (
236     ACPI_OPERAND_OBJECT *OriginalObject,
237     ACPI_OPERAND_OBJECT **ReturnObject)
238 {
239     ACPI_OPERAND_OBJECT *NewObject;
240     ACPI_STATUS Status;
241     ACPI_OPERAND_OBJECT **Elements;
242     UINT32 *DwordBuffer;
243     UINT32 Count;
244     UINT32 i;

247     switch (OriginalObject->Common.Type)
248     {
249     case ACPI_TYPE_INTEGER:
250         /*
251          * Integer-to-Buffer conversion.
252          * Convert the Integer to a packed-byte buffer. _MAT and other
253          * objects need this sometimes, if a read has been performed on a
254          * field object that is less than or equal to the global integer
255          * size (32 or 64 bits).
256          */
257         Status = AcpiExConvertToBuffer (OriginalObject, &NewObject);
258         if (ACPI_FAILURE (Status))

```

```

259     {
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
424     *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 }

```

new/usr/src/common/acpica/components/namespace/nsdump.c

1

```
*****
26291 Fri Jan 17 07:30:23 2014
new/usr/src/common/acpica/components/namespace/nsdump.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: nsdump - table dumping routines for debug
4 *
5 *****/
6 /
7 *
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __NSDUMP_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acnamesp.h"
49 #include "acoutput.h"
50
51 #define _COMPONENT ACPI_NAMESPACE
52 #define ACPI_MODULE_NAME ("nsdump")
53
54 /* Local prototypes */
55
56 #ifndef ACPI_OBSOLETE_FUNCTIONS
```

new/usr/src/common/acpica/components/namespace/nsdump.c

2

```
58 void
59 AcpiNsDumpRootDevices (
60     void);
61
62 static ACPI_STATUS
63 AcpiNsDumpOneDevice (
64     ACPI_HANDLE ObjHandle,
65     UINT32 Level,
66     void *Context,
67     void **ReturnValue);
68 #endif
69
70 #if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)
71
72 static ACPI_STATUS
73 AcpiNsDumpOneObjectPath (
74     ACPI_HANDLE ObjHandle,
75     UINT32 Level,
76     void *Context,
77     void **ReturnValue);
78
79 static ACPI_STATUS
80 AcpiNsGetMaxDepth (
81     ACPI_HANDLE ObjHandle,
82     UINT32 Level,
83     void *Context,
84     void **ReturnValue);
85
86 /*****
87 *
88 * FUNCTION: AcpiNsPrintPathname
89 *
90 * PARAMETERS: NumSegments - Number of ACPI name segments
91 * Pathname - The compressed (internal) path
92 *
93 * RETURN: None
94 *
95 * DESCRIPTION: Print an object's full namespace pathname
96 *
97 *****/
98
99 void
100 AcpiNsPrintPathname (
101     UINT32 NumSegments,
102     char *Pathname)
103 {
104     UINT32 i;
105
106     ACPI_FUNCTION_NAME (NsPrintPathname);
107
108     /* Check if debug output enabled */
109
110     if (!ACPI_IS_DEBUG_ENABLED (ACPI_LV_NAMES, ACPI_NAMESPACE))
111     {
112         if (!(AcpiDbgLevel & ACPI_LV_NAMES) || !(AcpiDbgLayer & ACPI_NAMESPACE))
113         {
114             return;
115         }
116     }
117
118     /* Print the entire name */
119
120     ACPI_DEBUG_PRINT ((ACPI_DB_NAMES, "[%s]");
```

```

123 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     }
131
132     Pathname += ACPI_NAME_SIZE;
133     NumSegments--;
134     if (NumSegments)
135     {
136         AcpiOsPrintf (".");
137     }
138 }
139
140 AcpiOsPrintf ("\n");
141 }
142
143 /*****
144 * FUNCTION:      AcpiNsDumpPathname
145 *
146 * PARAMETERS:    Handle          - Object
147 *                Msg             - Prefix message
148 *                Level           - Desired debug level
149 *                Component       - Caller's component ID
150 *
151 * RETURN:        None
152 *
153 * DESCRIPTION:   Print an object's full namespace pathname
154 *                Manages allocation/freeing of a pathname buffer
155 *
156 * *****/
157
158 void
159 AcpiNsDumpPathname (
160     ACPI_HANDLE      Handle,
161     char             *Msg,
162     UINT32           Level,
163     UINT32           Component)
164 {
165     ACPI_FUNCTION_TRACE (NsDumpPathname);
166
167     /* Do this only if the requested debug level and component are enabled */
168
169     if (!ACPI_IS_DEBUG_ENABLED (Level, Component))
170     {
171         if (!(AcpiDbgLevel & Level) || !(AcpiDbgLayer & Component))
172             return_VOID;
173     }
174
175     /* Convert handle to a full pathname and print it (with supplied message) */
176
177     AcpiNsPrintNodePathname (Handle, Msg);
178     AcpiOsPrintf ("\n");
179     return_VOID;
180 }
181
182 /*****
183 *
184 * *****/

```

```

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 *****/
201
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;
217
218     ACPI_FUNCTION_NAME (NsDumpOneObject);
219
220     /* Is output enabled? */
221
222     if (!(AcpiDbgLevel & Info->DebugLevel))
223     {
224         return (AE_OK);
225     }
226
227     if (!ObjHandle)
228     {
229         ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "Null object handle\n"));
230         return (AE_OK);
231     }
232
233     ThisNode = AcpiNsValidateHandle (ObjHandle);
234     if (!ThisNode)
235     {
236         ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "Invalid object handle %p\n",
237             ObjHandle));
238         return (AE_OK);
239     }
240
241     Type = ThisNode->Type;
242
243     /* Check if the owner matches */
244
245     if ((Info->OwnerId != ACPI_OWNER_ID_MAX) &&
246         (Info->OwnerId != ThisNode->OwnerId))
247     {
248         return (AE_OK);
249     }
250
251     if (!(Info->DisplayType & ACPI_DISPLAY_SHORT))

```



```

254 {
255     /* Indent the object according to the level */
256
257     AcpiOsPrintf ("%2d%s", (UINT32) Level - 1, (int) Level * 2, " ");
258
259     /* Check the node type and name */
260
261     if (Type > ACPI_TYPE_LOCAL_MAX)
262     {
263         ACPI_WARNING ((AE_INFO, "Invalid ACPI Object Type 0x%08X", Type));
264     }
265
266     AcpiOsPrintf ("%4.4s", AcpiUtGetNodeName (ThisNode));
267 }
268
269 /* Now we can print out the pertinent information */
270
271 AcpiOsPrintf (" %-12s %p %2.2X ",
272     AcpiUtGetTypeName (Type), ThisNode, ThisNode->OwnerId);
273
274 DbgLevel = AcpiDbgLevel;
275 AcpiDbgLevel = 0;
276 ObjDesc = AcpiNsGetAttachedObject (ThisNode);
277 AcpiDbgLevel = DbgLevel;
278
279 /* Temp nodes are those nodes created by a control method */
280
281 if (ThisNode->Flags & ANOBJ_TEMPORARY)
282 {
283     AcpiOsPrintf ("(T) ");
284 }
285
286 switch (Info->DisplayType & ACPI_DISPLAY_MASK)
287 {
288     case ACPI_DISPLAY_SUMMARY:
289
290         if (!ObjDesc)
291         {
292             /* No attached object. Some types should always have an object */
293             /* No attached object, we are done */
294
295             switch (Type)
296             {
297                 case ACPI_TYPE_INTEGER:
298                 case ACPI_TYPE_PACKAGE:
299                 case ACPI_TYPE_BUFFER:
300                 case ACPI_TYPE_STRING:
301                 case ACPI_TYPE_METHOD:
302
303                     AcpiOsPrintf ("<No attached object>");
304                     break;
305
306                 default:
307                     break;
308             }
309
310             AcpiOsPrintf ("\n");
311             return (AE_OK);
312         }
313
314         switch (Type)
315         {
316             case ACPI_TYPE_PROCESSOR:
317
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;
322
323     case ACPI_TYPE_DEVICE:
324
325         AcpiOsPrintf ("Notify Object: %p\n", ObjDesc);
326         break;
327
328     case ACPI_TYPE_METHOD:
329
330         AcpiOsPrintf ("Args %X Len %.4X Aml %p\n",
331             (UINT32) ObjDesc->Method.ParamCount,
332             ObjDesc->Method.AmlLength, ObjDesc->Method.AmlStart);
333         break;
334
335     case ACPI_TYPE_INTEGER:
336
337         AcpiOsPrintf ("= %8.8X%8.8X\n",
338             ACPI_FORMAT_UINT64 (ObjDesc->Integer.Value));
339         break;
340
341     case ACPI_TYPE_PACKAGE:
342
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;
353
354     case ACPI_TYPE_BUFFER:
355
356         if (ObjDesc->Common.Flags & AOPOBJ_DATA_VALID)
357         {
358             AcpiOsPrintf ("Len %2X",
359                 ObjDesc->Buffer.Length);
360
361             /* Dump some of the buffer */
362
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                 AcpiOsPrintf ("\n");
371             }
372             else
373             {
374                 AcpiOsPrintf ("[Length not yet evaluated]\n");
375             }
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. AcpiNsGetMaxDepth
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

```

new/usr/src/common/acpica/components/namespace/nsdumpdv.c

1

```
*****
5063 Fri Jan 17 07:30:24 2014
new/usr/src/common/acpica/components/namespace/nsdumpdv.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: nsdump - table dumping routines for debug
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __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"
```

new/usr/src/common/acpica/components/namespace/nsdumpdv.c

2

```
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
```

new/usr/src/common/acpica/components/namespace/nseval.c

1

```
*****
16392 Fri Jan 17 07:30:24 2014
new/usr/src/common/acpica/components/namespace/nseval.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: nseval - Object evaluation, includes control method execution
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __NSEVAL_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acparser.h"
49 #include "acinterp.h"
50 #include "acnamesp.h"

53 #define _COMPONENT          ACPI_NAMESPACE
54     ACPI_MODULE_NAME        ("nseval")

56 /* Local prototypes */

58 static void
59 AcpiNsExecModuleCode (
```

new/usr/src/common/acpica/components/namespace/nseval.c

2

```
60     ACPI_OPERAND_OBJECT    *MethodObj,
61     ACPI_EVALUATE_INFO     *Info);

64 /*****
65 *
66 * FUNCTION:      AcpiNsEvaluate
67 *
68 * PARAMETERS:    Info          - Evaluation info block, contains:
69 *                  PrefixNode    - Prefix or Method/Object Node to execute
70 *                  RelativePath   - Name of method to execute, If NULL, the
71 *                                Node is the object to execute
72 *                  Parameters     - List of parameters to pass to the method,
73 *                                terminated by NULL. Params itself may be
74 *                                NULL if no parameters are being passed.
75 *                  ReturnObject   - Where to put method's return value (if
76 *                                any). If NULL, no value is returned.
77 *                  ParameterType  - Type of Parameter list
78 *                  ReturnObject   - Where to put method's return value (if
79 *                                any). If NULL, no value is returned.
80 *                  Flags          - ACPI_IGNORE_RETURN_VALUE to delete return
81 *
82 * RETURN:        Status
83 *
84 * DESCRIPTION:   Execute a control method or return the current value of an
85 *                ACPI namespace object.
86 *
87 * MUTEX:         Locks interpreter
88 *
89 *****/

91 ACPI_STATUS
92 AcpiNsEvaluate (
93     ACPI_EVALUATE_INFO     *Info)
94 {
95     ACPI_STATUS             Status;

98     ACPI_FUNCTION_TRACE (NsEvaluate);

101     if (!Info)
102     {
103         return ACPI_STATUS (AE_BAD_PARAMETER);
104     }

106     if (!Info->Node)
107     {
108         /*
109          * Get the actual namespace node for the target object if we
110          * need to. Handles these cases:
111          *
112          * 1) Null node, valid pathname from root (absolute path)
113          * 2) Node and valid pathname (path relative to Node)
114          * 3) Node, Null pathname
115          */
116         Status = AcpiNsGetNode (Info->PrefixNode, Info->RelativePathname,
117                                 ACPI_NS_NO_UPSEARCH, &Info->Node);
118         if (ACPI_FAILURE (Status))
119         {
120             return ACPI_STATUS (Status);
121         }
122     }

124     /*
125     * For a method alias, we must grab the actual method node so that
```



```

126  * 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  */
261
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 ();
279
280 /* TBD: ResolveNodeToValue has a strange interface, fix */
281
282 Info->ReturnObject = ACPI_CAST_PTR (ACPI_OPERAND_OBJECT, Info->Node);
283
284 Status = AcpiExResolveNodeToValue (ACPI_CAST_INDIRECT_PTR (
285     ACPI_NAMESPACE_NODE, &Info->ReturnObject), NULL);
286 AcpiExExitInterpreter ();
287
288 if (ACPI_FAILURE (Status))
289 {
290     goto Cleanup;
291 }
292
293 ACPI_DEBUG_PRINT ((ACPI_DB_NAMES, "Returned object %p [%s]\n",
294     Info->ReturnObject,
295     AcpiUtGetObjectTypeName (Info->ReturnObject)));
296
297 Status = AE_CTRL_RETURN_VALUE; /* Always has a "return value" */
298 break;
299 }
300
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);
307
308 /* Check if there is a return value that must be dealt with */
309
310 if (Status == AE_CTRL_RETURN_VALUE)
311 {
312     /* If caller does not want the return value, delete it */
313
314     if (Info->Flags & ACPI_IGNORE_RETURN_VALUE)
315     {
316         AcpiUtRemoveReference (Info->ReturnObject);
317         Info->ReturnObject = NULL;
318     }
319
320     /* Map AE_CTRL_RETURN_VALUE to AE_OK, we are done with it */
321
322     Status = AE_OK;
323 }

```

```

325     ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
326         "**** Completed evaluation of object %s ***\n",
327         Info->RelativePathname));
328
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 }
338
339 /*****
340  * FUNCTION:      AcpiNsExecModuleCodeList
341  *
342  * PARAMETERS:    None
343  *
344  * RETURN:        None. Exceptions during method execution are ignored, since
345  *                we cannot abort a table load.
346  *
347  * DESCRIPTION:   Execute all elements of the global module-level code list.
348  *                Each element is executed as a single control method.
349  *****/
350 void
351 AcpiNsExecModuleCodeList (
352     void)
353 {
354     ACPI_OPERAND_OBJECT *Prev;
355     ACPI_OPERAND_OBJECT *Next;
356     ACPI_EVALUATE_INFO *Info;
357     UINT32 MethodCount = 0;
358
359     ACPI_FUNCTION_TRACE (NsExecModuleCodeList);
360
361     /* Exit now if the list is empty */
362
363     Next = AcpiGbl_ModuleCodeList;
364     if (!Next)
365     {
366         return_VOID;
367     }
368
369     /* Allocate the evaluation information block */
370
371     Info = ACPI_ALLOCATE (sizeof (ACPI_EVALUATE_INFO));
372     if (!Info)
373     {
374         return_VOID;
375     }
376
377     /* Walk the list, executing each "method" */
378
379     while (Next)
380     {
381         Prev = Next;
382         Next = Next->Method.Mutexp;
383     }

```

```

390      /* Clear the link field and execute the method */
391
392      Prev->Method.Mutex = NULL;
393      AcpiNsExecModuleCode (Prev, Info);
394      MethodCount++;
395
396      /* Delete the (temporary) method object */
397
398      AcpiUtRemoveReference (Prev);
399  }
400
401  ACPI_INFO ((AE_INFO,
402    "Executed %u blocks of module-level executable AML code",
403    MethodCount));
404
405  ACPI_FREE (Info);
406  AcpiGbl_ModuleCodeList = NULL;
407  return_VOID;
408 }
409
410 /*****
411  *
412  * FUNCTION:      AcpiNsExecModuleCode
413  *
414  * PARAMETERS:    MethodObj      - Object container for the module-level code
415  *                Info           - Info block for method evaluation
416  *
417  * RETURN:        None. Exceptions during method execution are ignored, since
418  *                we cannot abort a table load.
419  *
420  * DESCRIPTION:   Execute a control method containing a block of module-level
421  *                executable AML code. The control method is temporarily
422  *                installed to the root node, then evaluated.
423  *
424  *****/
425
426 static void
427 AcpiNsExecModuleCode (
428     ACPI_OPERAND_OBJECT *MethodObj,
429     ACPI_EVALUATE_INFO *Info)
430 {
431     ACPI_OPERAND_OBJECT *ParentObj;
432     ACPI_NAMESPACE_NODE *ParentNode;
433     ACPI_OBJECT_TYPE Type;
434     ACPI_STATUS Status;
435
436     ACPI_FUNCTION_TRACE (NsExecModuleCode);
437
438     /*
439     * Get the parent node. We cheat by using the NextObject field
440     * of the method object descriptor.
441     */
442     ParentNode = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE,
443       MethodObj->Method.NextObject);
444     Type = AcpiNsGetType (ParentNode);
445
446     /*
447     * Get the region handler and save it in the method object. We may need
448     * this if an operation region declaration causes a _REG method to be run.
449     *
450     * We can't do this in AcpiPsLinkModuleCode because
451     * AcpiGbl_RootNode->Object is NULL at PASS1.
452     */

```

```

456     if ((Type == ACPI_TYPE_DEVICE) && ParentNode->Object)
457     {
458         MethodObj->Method.Dispatch.Handler =
459             ParentNode->Object->Device.Handler;
460     }
461
462     /* Must clear NextObject (AcpiNsAttachObject needs the field) */
463
464     MethodObj->Method.NextObject = NULL;
465
466     /* Initialize the evaluation information block */
467
468     ACPI_MEMSET (Info, 0, sizeof (ACPI_EVALUATE_INFO));
469     Info->PrefixNode = ParentNode;
470
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     }
481
482     /* Install the method (module-level code) in the parent node */
483
484     Status = AcpiNsAttachObject (ParentNode, MethodObj,
485       ACPI_TYPE_METHOD);
486     if (ACPI_FAILURE (Status))
487     {
488         goto Exit;
489     }
490
491     /* Execute the parent node as a control method */
492
493     Status = AcpiNsEvaluate (Info);
494
495     ACPI_DEBUG_PRINT ((ACPI_DB_INIT, "Executed module-level code at %p\n",
496       MethodObj->Method.AmlStart));
497
498     /* Delete a possible implicit return value (in slack mode) */
499
500     if (Info->ReturnObject)
501     {
502         AcpiUtRemoveReference (Info->ReturnObject);
503     }
504
505     /* Detach the temporary method object */
506
507     AcpiNsDetachObject (ParentNode);
508
509     /* Restore the original parent object */
510
511     if (ParentObj)
512     {
513         Status = AcpiNsAttachObject (ParentNode, ParentObj, Type);
514     }
515     else
516     {
517         ParentNode->Type = (UINT8) Type;
518     }
519
520 Exit:
521     if (ParentObj)

```

```
522 {  
523     AcpiUtRemoveReference (ParentObj);  
524 }  
525 return_VOID;  
526 }
```

new/usr/src/common/acpica/components/namespace/nsinit.c

1

```
*****
19552 Fri Jan 17 07:30:24 2014
new/usr/src/common/acpica/components/namespace/nsinit.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: nsinit - namespace initialization
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __NSXFINIT_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acnamesp.h"
50 #include "acdispat.h"
51 #include "acinterp.h"

53 #define _COMPONENT ACPI_NAMESPACE
54 ACPI_MODULE_NAME("nsinit")

56 /* Local prototypes */

58 static ACPI_STATUS
```

new/usr/src/common/acpica/components/namespace/nsinit.c

2

```
59 AcpiNsInitOneObject (
60     ACPI_HANDLE ObjHandle,
61     UINT32 Level,
62     void *Context,
63     void **ReturnValue);

65 static ACPI_STATUS
66 AcpiNsInitOneDevice (
67     ACPI_HANDLE ObjHandle,
68     UINT32 NestingLevel,
69     void *Context,
70     void **ReturnValue);

72 static ACPI_STATUS
73 AcpiNsFindIniMethods (
74     ACPI_HANDLE ObjHandle,
75     UINT32 NestingLevel,
76     void *Context,
77     void **ReturnValue);

80 /*****
81 *
82 * FUNCTION: AcpiNsInitializeObjects
83 *
84 * PARAMETERS: None
85 *
86 * RETURN: Status
87 *
88 * DESCRIPTION: Walk the entire namespace and perform any necessary
89 * initialization on the objects found therein
90 *
91 *****/

93 ACPI_STATUS
94 AcpiNsInitializeObjects (
95     void)
96 {
97     ACPI_STATUS Status;
98     ACPI_INIT_WALK_INFO Info;

101     ACPI_FUNCTION_TRACE (NsInitializeObjects);

104     ACPI_DEBUG_PRINT ((ACPI_DB_DISPATCH,
105         "**** Starting initialization of namespace objects ****\n"));
106     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT,
107         "Completing Region/Field/Buffer/Package initialization:\n"));
108     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT,
109         "Completing Region/Field/Buffer/Package initialization:"));

109     /* Set all init info to zero */

111     ACPI_MEMSET (&Info, 0, sizeof (ACPI_INIT_WALK_INFO));

113     /* Walk entire namespace from the supplied root */

115     Status = AcpiWalkNamespace (ACPI_TYPE_ANY, ACPI_ROOT_OBJECT,
116         ACPI_UINT32_MAX, AcpiNsInitOneObject, NULL,
117         &Info, NULL);
118     if (ACPI_FAILURE (Status))
119     {
120         ACPI_EXCEPTION ((AE_INFO, Status, "During WalkNamespace"));
121     }

123     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT,
```

```

124      "    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 OS 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
253 *           ReturnValue   - Not used
254 *
255 * RETURN:      Status
256 *
257 * DESCRIPTION: Callback from AcpiWalkNamespace. Invoked for every object
258 *              within the namespace.
259 *
260 *              Currently, the only objects that require initialization are:
261 *              1) Methods
262 *              2) Op Regions
263 *
264 *****/

266 static ACPI_STATUS
267 AcpiNsInitOneObject (
268     ACPI_HANDLE      ObjHandle,
269     UINT32           Level,
270     void             *Context,
271     void             **ReturnValue)
272 {
273     ACPI_OBJECT_TYPE  Type;
274     ACPI_STATUS       Status = AE_OK;
275     ACPI_INIT_WALK_INFO *Info = (ACPI_INIT_WALK_INFO *) Context;
276     ACPI_NAMESPACE_NODE *Node = (ACPI_NAMESPACE_NODE *) ObjHandle;
277     ACPI_OPERAND_OBJECT *ObjDesc;

280     ACPI_FUNCTION_NAME (NsInitOneObject);

283     Info->ObjectCount++;

285     /* And even then, we are only interested in a few object types */

287     Type = AcpiNsGetType (ObjHandle);
288     ObjDesc = AcpiNsGetAttachedObject (Node);
289     if (!ObjDesc)
290     {
291         return (AE_OK);
292     }

294     /* Increment counters for object types we are looking for */

296     switch (Type)
297     {
298     case ACPI_TYPE_REGION:

300         Info->OpRegionCount++;
301         break;

303     case ACPI_TYPE_BUFFER_FIELD:

305         Info->FieldCount++;
306         break;

308     case ACPI_TYPE_LOCAL_BANK_FIELD:

310         Info->FieldCount++;
311         break;

313     case ACPI_TYPE_BUFFER:

315         Info->BufferCount++;
316         break;

```

```

318     case ACPI_TYPE_PACKAGE:

320         Info->PackageCount++;
321         break;

323     default:

325         /* No init required, just exit now */

327         return (AE_OK);
328     }

330     /* If the object is already initialized, nothing else to do */

332     if (ObjDesc->Common.Flags & AOPOBJ_DATA_VALID)
333     {
334         return (AE_OK);
335     }

337     /* Must lock the interpreter before executing AML code */

339     AcpiExEnterInterpreter ();

341     /*
342     * Each of these types can contain executable AML code within the
343     * declaration.
344     */
345     switch (Type)
346     {
347     case ACPI_TYPE_REGION:

349         Info->OpRegionInit++;
350         Status = AcpiDsGetRegionArguments (ObjDesc);
351         break;

353     case ACPI_TYPE_BUFFER_FIELD:

355         Info->FieldInit++;
356         Status = AcpiDsGetBufferFieldArguments (ObjDesc);
357         break;

359     case ACPI_TYPE_LOCAL_BANK_FIELD:

361         Info->FieldInit++;
362         Status = AcpiDsGetBankFieldArguments (ObjDesc);
363         break;

365     case ACPI_TYPE_BUFFER:

367         Info->BufferInit++;
368         Status = AcpiDsGetBufferArguments (ObjDesc);
369         break;

371     case ACPI_TYPE_PACKAGE:

373         Info->PackageInit++;
374         Status = AcpiDsGetPackageArguments (ObjDesc);
375         break;

377     default:

379         /* No other types can get here */

381         break;
382     }

```

```

384     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     }
390
391     /*
392     * Print a dot for each object unless we are going to print the entire
393     * pathname
394     */
395     if (!(AcpiDbgLevel & ACPI_LV_INIT_NAMES))
396     {
397         ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT, ".");
398     }
399
400     /*
401     * We ignore errors from above, and always return OK, since we don't want
402     * to abort the walk on any single error.
403     */
404     AcpiExExitInterpreter ();
405     return (AE_OK);
406 }
407
408 /*****
409 * FUNCTION:      AcpiNsFindIniMethods
410 * PARAMETERS:    ACPI_WALK_CALLBACK
411 * RETURN:        ACPI_STATUS
412 * DESCRIPTION:   Called during namespace walk. Finds objects named _INI under
413 *                device/processor/thermal objects, and marks the entire subtree
414 *                with a SUBTREE_HAS_INI flag. This flag is used during the
415 *                subsequent device initialization walk to avoid entire subtrees
416 *                that do not contain an _INI.
417 *****/
418
419 static ACPI_STATUS
420 AcpiNsFindIniMethods (
421     ACPI_HANDLE          ObjHandle,
422     UINT32               NestingLevel,
423     void                 *Context,
424     void                 **ReturnValue)
425 {
426     ACPI_DEVICE_WALK_INFO *Info = ACPI_CAST_PTR (ACPI_DEVICE_WALK_INFO, Context);
427     ACPI_NAMESPACE_NODE *Node;
428     ACPI_NAMESPACE_NODE *ParentNode;
429
430     /* Keep count of device/processor/thermal objects */
431
432     Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, ObjHandle);
433     if ((Node->Type == ACPI_TYPE_DEVICE) ||
434         (Node->Type == ACPI_TYPE_PROCESSOR) ||
435         (Node->Type == ACPI_TYPE_THERMAL))
436     {
437         Info->DeviceCount++;
438         return (AE_OK);
439     }
440
441     /* 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     }
445
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:
456
457         /* Mark parent and bubble up the INI present flag to the root */
458
459         while (ParentNode)
460         {
461             ParentNode->Flags |= ANOBJ_SUBTREE_HAS_INI;
462             ParentNode = ParentNode->Parent;
463         }
464         break;
465
466     default:
467         break;
468     }
469
470     return (AE_OK);
471 }
472
473 /*****
474 * FUNCTION:      AcpiNsInitOneDevice
475 * PARAMETERS:    ACPI_WALK_CALLBACK
476 * RETURN:        ACPI_STATUS
477 * DESCRIPTION:   This is called once per device soon after ACPI is enabled
478 *                to initialize each device. It determines if the device is
479 *                present, and if so, calls _INI.
480 *****/
481
482 static ACPI_STATUS
483 AcpiNsInitOneDevice (
484     ACPI_HANDLE          ObjHandle,
485     UINT32               NestingLevel,
486     void                 *Context,
487     void                 **ReturnValue)
488 {
489     ACPI_DEVICE_WALK_INFO *WalkInfo = ACPI_CAST_PTR (ACPI_DEVICE_WALK_INFO, Context);
490     ACPI_EVALUATE_INFO *Info = WalkInfo->EvaluateInfo;
491     UINT32               Flags;
492     ACPI_STATUS           Status;
493     ACPI_NAMESPACE_NODE *DeviceNode;
494
495     ACPI_FUNCTION_TRACE (NsInitOneDevice);
496
497     /* 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 }

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 }

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));

542 Status = AcpiUtExecute_STA (DeviceNode, &Flags);
543 if (ACPI_FAILURE (Status))
544 {
545     /* Ignore error and move on to next device */

547     return_ACPI_STATUS (AE_OK);
548 }

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 }

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 }

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));

621 ACPI_MEMSET (Info, 0, sizeof (ACPI_EVALUATE_INFO));
622 Info->PrefixNode = DeviceNode;
623 Info->RelativePathname = METHOD_NAME__INI;
624 Info->Pathname = METHOD_NAME__INI;
625 Info->Parameters = NULL;
626 Info->Flags = ACPI_IGNORE_RETURN_VALUE;

627 Status = AcpiNsEvaluate (Info);
628 if (ACPI_SUCCESS (Status))
629 {
630     WalkInfo->Num_INI++;

631     if ((AcpiDbgLevel <= ACPI_LV_ALL_EXCEPTIONS) &&
632         (!(AcpiDbgLevel & ACPI_LV_INFO)))
633     {
634         ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INIT, ".");
635     }
636 }

```

```
633 #ifdef ACPI_DEBUG_OUTPUT
634     else if (Status != AE_NOT_FOUND)
635     {
636         /* Ignore error and move on to next device */
637
638         char *ScopeName = AcpiNsGetExternalPathname (Info->Node);
639         char *ScopeName = AcpiNsGetExternalPathname (Info->ResolvedNode);
640
641         ACPI_EXCEPTION ((AE_INFO, Status, "during %s._INI execution",
642             ScopeName));
643         ACPI_FREE (ScopeName);
644     }
645 #endif
646
647     /* Ignore errors from above */
648
649     Status = AE_OK;
650
651     /*
652     * The _INI method has been run if present; call the Global Initialization
653     * Handler for this device.
654     */
655     if (AcpiGbl_InitHandler)
656     {
657         Status = AcpiGbl_InitHandler (DeviceNode, ACPI_INIT_DEVICE_INI);
658     }
659
660     return ACPI_STATUS (Status);
661 }
662
663 unchanged_portion_omitted
```

new/usr/src/common/acpica/components/namespace/nsload.c

1

```
*****
10010 Fri Jan 17 07:30:25 2014
new/usr/src/common/acpica/components/namespace/nsload.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: nsload - namespace loading/expanding/contracting procedures
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __NSLOAD_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acnamesp.h"
49 #include "acdispat.h"
50 #include "actables.h"

53 #define _COMPONENT          ACPI_NAMESPACE
54         ACPI_MODULE_NAME    ("nsload")

56 /* Local prototypes */

58 #ifndef ACPI_FUTURE_IMPLEMENTATION
```

new/usr/src/common/acpica/components/namespace/nsload.c

2

```
59 ACPI_STATUS
60 AcpiNsUnloadNamespace (
61     ACPI_HANDLE             Handle);

63 static ACPI_STATUS
64 AcpiNsDeleteSubtree (
65     ACPI_HANDLE             StartHandle);
66 #endif

69 #ifndef ACPI_NO_METHOD_EXECUTION
70 /*****
71 *
72 * FUNCTION:      AcpiNsLoadTable
73 *
74 * PARAMETERS:    TableIndex      - Index for table to be loaded
75 *                Node            - Owning NS node
76 *
77 * RETURN:        Status
78 *
79 * DESCRIPTION:   Load one ACPI table into the namespace
80 *
81 *****/

83 ACPI_STATUS
84 AcpiNsLoadTable (
85     UINT32              TableIndex,
86     ACPI_NAMESPACE_NODE *Node)
87 {
88     ACPI_STATUS          Status;

91     ACPI_FUNCTION_TRACE (NsLoadTable);

94     /*
95     * Parse the table and load the namespace with all named
96     * objects found within. Control methods are NOT parsed
97     * at this time. In fact, the control methods cannot be
98     * parsed until the entire namespace is loaded, because
99     * if a control method makes a forward reference (call)
100     * to another control method, we can't continue parsing
101     * because we don't know how many arguments to parse next!
102     */
103     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
104     if (ACPI_FAILURE (Status))
105     {
106         return_ACPI_STATUS (Status);
107     }

109     /* If table already loaded into namespace, just return */

111     if (AcpiTbIsTableLoaded (TableIndex))
112     {
113         Status = AE_ALREADY_EXISTS;
114         goto Unlock;
115     }

117     ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
118         "**** Loading table into namespace ****\n"));

120     Status = AcpiTbAllocateOwnerId (TableIndex);
121     if (ACPI_FAILURE (Status))
122     {
123         goto Unlock;
124     }
125 }
```

```
126     Status = AcpiNsParseTable (TableIndex, Node);
127     if (ACPI_SUCCESS (Status))
128     {
129         AcpiTbSetTableLoadedFlag (TableIndex, TRUE);
130     }
131     else
132     {
133         (void) AcpiTbReleaseOwnerId (TableIndex);
134     }
135
136 Unlock:
137     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
138
139     if (ACPI_FAILURE (Status))
140     {
141         return_ACPI_STATUS (Status);
142     }
143
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"));
152
153     Status = AcpiDsInitializeObjects (TableIndex, Node);
154
155     ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
156         "**** Completed Table Method Parsing and Object Initialization\n"));
157
158     return_ACPI_STATUS (Status);
159 }
160
161 unchanged_portion_omitted
162 #endif
163 #endif
```

new/usr/src/common/acpica/components/namespace/nsnames.c

1

```
*****
8693 Fri Jan 17 07:30:25 2014
new/usr/src/common/acpica/components/namespace/nsnames.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: nsnames - Name manipulation and search
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __NSNAMES_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "amlcode.h"
49 #include "acnamesp.h"

52 #define _COMPONENT          ACPI_NAMESPACE
53       ACPI_MODULE_NAME    ("nsnames")

56 /*****
57 *
58 * FUNCTION:      AcpiNsBuildExternalPath
```

new/usr/src/common/acpica/components/namespace/nsnames.c

2

```
59 *
60 * PARAMETERS:    Node          - NS node whose pathname is needed
61 *               Size          - Size of the pathname
62 *               *NameBuffer    - Where to return the pathname
63 *
64 * RETURN:        Status
65 *               Places the pathname into the NameBuffer, in external format
66 *               (name segments separated by path separators)
67 *
68 * DESCRIPTION:   Generate a full pathaname
69 *
70 *****/

72 ACPI_STATUS
73 AcpiNsBuildExternalPath (
74     ACPI_NAMESPACE_NODE *Node,
75     ACPI_SIZE Size,
76     char *NameBuffer)
77 {
78     ACPI_SIZE Index;
79     ACPI_NAMESPACE_NODE *ParentNode;

82     ACPI_FUNCTION_ENTRY ();

85     /* Special case for root */

87     Index = Size - 1;
88     if (Index < ACPI_NAME_SIZE)
89     {
90         NameBuffer[0] = AML_ROOT_PREFIX;
91         NameBuffer[1] = 0;
92         return (AE_OK);
93     }

95     /* Store terminator byte, then build name backwards */

97     ParentNode = Node;
98     NameBuffer[Index] = 0;

100     while ((Index > ACPI_NAME_SIZE) && (ParentNode != AcpiGbl_RootNode))
101     {
102         Index -= ACPI_NAME_SIZE;

104         /* Put the name into the buffer */

106         ACPI_MOVE_32_TO_32 ((NameBuffer + Index), &ParentNode->Name);
107         ParentNode = ParentNode->Parent;

109         /* Prefix name with the path separator */

111         Index--;
112         NameBuffer[Index] = ACPI_PATH_SEPARATOR;
113     }

115     /* Overwrite final separator with the root prefix character */

117     NameBuffer[Index] = AML_ROOT_PREFIX;

119     if (Index != 0)
120     {
121         ACPI_ERROR ((AE_INFO,
122             "Could not construct external pathname; index=%u, size=%u, Path=%s",
123             (UINT32) Index, (UINT32) Size, &NameBuffer[Size]));
```

```
125     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_
```

new/usr/src/common/acpica/components/namespace/nsobject.c

1

```
*****
13934 Fri Jan 17 07:30:26 2014
new/usr/src/common/acpica/components/namespace/nsobject.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: nsobject - Utilities for objects attached to namespace
4 *                               table entries
5 *
6 *****/
8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
44
46 #define __NSOBJECT_C__
47
48 #include "acpi.h"
49 #include "accommon.h"
50 #include "acnamesp.h"
51
53 #define _COMPONENT          ACPI_NAMESPACE
54         ACPI_MODULE_NAME    ("nsobject")
55
57 /*****
58 *
```

new/usr/src/common/acpica/components/namespace/nsobject.c

2

```
59 * FUNCTION:      AcpiNsAttachObject
60 *
61 * PARAMETERS:    Node           - Parent Node
62 *                Object         - Object to be attached
63 *                Type           - Type of object, or ACPI_TYPE_ANY if not
64 *                               known
65 *
66 * RETURN:        Status
67 *
68 * DESCRIPTION:    Record the given object as the value associated with the
69 *                 name whose ACPI_HANDLE is passed. If Object is NULL
70 *                 and Type is ACPI_TYPE_ANY, set the name as having no value.
71 *                 Note: Future may require that the Node->Flags field be passed
72 *                 as a parameter.
73 *
74 * MUTEX:          Assumes namespace is locked
75 *
76 *****/
77
78 ACPI_STATUS
79 AcpiNsAttachObject (
80     ACPI_NAMESPACE_NODE *Node,
81     ACPI_OPERAND_OBJECT *Object,
82     ACPI_OBJECT_TYPE Type)
83 {
84     ACPI_OPERAND_OBJECT *ObjDesc;
85     ACPI_OPERAND_OBJECT *LastObjDesc;
86     ACPI_OBJECT_TYPE ObjectType = ACPI_TYPE_ANY;
87
88     ACPI_FUNCTION_TRACE (NsAttachObject);
89
90     /*
91      * Parameter validation
92      */
93     if (!Node)
94     {
95         /* Invalid handle */
96
97         ACPI_ERROR ((AE_INFO, "Null NamedObj handle"));
98         return_ACPI_STATUS (AE_BAD_PARAMETER);
99     }
100
101     if (!Object && (ACPI_TYPE_ANY != Type))
102     {
103         /* Null object */
104
105         ACPI_ERROR ((AE_INFO, "Null object, but type not ACPI_TYPE_ANY"));
106         return_ACPI_STATUS (AE_BAD_PARAMETER);
107     }
108
109     if (ACPI_GET_DESCRIPTOR_TYPE (Node) != ACPI_DESC_TYPE_NAMED)
110     {
111         /* Not a name handle */
112
113         ACPI_ERROR ((AE_INFO, "Invalid handle %p [%s]",
114             Node, AcpiUtGetDescriptorName (Node)));
115         return_ACPI_STATUS (AE_BAD_PARAMETER);
116     }
117
118     /* Check if this object is already attached */
119
120     if (Node->Object == Object)
121     {
122     }
123
124 }
```

new/usr/src/common/acpica/components/namespace/nsobject.c

3

```
125     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;
```

new/usr/src/common/acpica/components/namespace/nsobject.c

4

```
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



new/usr/src/common/acpica/components/namespace/nsparse.c

1

```
*****
7053 Fri Jan 17 07:30:26 2014
new/usr/src/common/acpica/components/namespace/nsparse.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: nsparse - namespace interface to AML parser
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __NSPARSE_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acnamesp.h"
49 #include "acparser.h"
50 #include "acdispat.h"
51 #include "actables.h"

54 #define _COMPONENT          ACPI_NAMESPACE
55       ACPI_MODULE_NAME    ("nsparse")

58 /*****
```

new/usr/src/common/acpica/components/namespace/nsparse.c

2

```
59 *
60 * FUNCTION:      NsOneCompleteParse
61 *
62 * PARAMETERS:    PassNumber          - 1 or 2
63 *               TableDesc            - The table to be parsed.
64 *
65 * RETURN:        Status
66 *
67 * DESCRIPTION:    Perform one complete parse of an ACPI/AML table.
68 *
69 *****/

71 ACPI_STATUS
72 AcpiNsOneCompleteParse (
73     UINT32          PassNumber,
74     UINT32          TableIndex,
75     ACPI_NAMESPACE_NODE *StartNode)
76 {
77     ACPI_PARSE_OBJECT *ParseRoot;
78     ACPI_STATUS        Status;
79     UINT32             AmlLength;
80     UINT8              AmlStart;
81     ACPI_WALK_STATE     WalkState;
82     ACPI_TABLE_HEADER   *Table;
83     ACPI_OWNER_ID       OwnerId;

86     ACPI_FUNCTION_TRACE (NsOneCompleteParse);

89     Status = AcpiTbGetOwnerId (TableIndex, &OwnerId);
90     if (ACPI_FAILURE (Status))
91     {
92         return_ACPI_STATUS (Status);
93     }

95     /* Create and init a Root Node */

97     ParseRoot = AcpiPsCreateScopeOp ();
98     if (!ParseRoot)
99     {
100         return_ACPI_STATUS (AE_NO_MEMORY);
101     }

103     /* Create and initialize a new walk state */

105     WalkState = AcpiDsCreateWalkState (OwnerId, NULL, NULL, NULL);
106     if (!WalkState)
107     {
108         AcpiPsFreeOp (ParseRoot);
109         return_ACPI_STATUS (AE_NO_MEMORY);
110     }

112     Status = AcpiGetTableByIndex (TableIndex, &Table);
113     if (ACPI_FAILURE (Status))
114     {
115         AcpiDsDeleteWalkState (WalkState);
116         AcpiPsFreeOp (ParseRoot);
117         return_ACPI_STATUS (Status);
118     }

120     /* Table must consist of at least a complete header */

122     if (Table->Length < sizeof (ACPI_TABLE_HEADER))
123     {
124         Status = AE_BAD_HEADER;
```

```
125     }
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     }
133
134     if (ACPI_FAILURE (Status))
135     {
136         AcpiDsDeleteWalkState (WalkState);
137         goto Cleanup;
138     }
139
140     /* StartNode is the default location to load the table */
141
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     }
151
152     /* Parse the AML */
153
154     ACPI_DEBUG_PRINT ((ACPI_DB_PARSE, "**PARSE* pass %u parse\n", PassNumber));
155     Status = AcpiPsParseAml (WalkState);
156
157 Cleanup:
158     AcpiPsDeleteParseTree (ParseRoot);
159     return ACPI_STATUS (Status);
160 }
```

unchanged portion omitted

new/usr/src/common/acpica/components/namespace/nspredef.c

1

```
*****
14310 Fri Jan 17 07:30:26 2014
new/usr/src/common/acpica/components/namespace/nspredef.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: nspredef - Validation of ACPI predefined methods and objects
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 ACPI_CREATE_PREDEFINED_TABLE

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acnamesp.h"
49 #include "acpredef.h"

52 #define _COMPONENT          ACPI_NAMESPACE
53       ACPI_MODULE_NAME    ("nspredef")

56 /*****
57 *
58 * This module validates predefined ACPI objects that appear in the namespace,
59 * at the time they are evaluated (via AcpiEvaluateObject). The purpose of this
```

new/usr/src/common/acpica/components/namespace/nspredef.c

2

```
60 * validation is to detect problems with BIOS-exposed predefined ACPI objects
61 * before the results are returned to the ACPI-related drivers.
62 *
63 * There are several areas that are validated:
64 *
65 * 1) The number of input arguments as defined by the method/object in the
66 * ASL is validated against the ACPI specification.
67 * 2) The type of the return object (if any) is validated against the ACPI
68 * specification.
69 * 3) For returned package objects, the count of package elements is
70 * validated, as well as the type of each package element. Nested
71 * packages are supported.
72 *
73 * For any problems found, a warning message is issued.
74 *
75 *****/

78 /* Local prototypes */

80 static ACPI_STATUS
81 AcpiNsCheckReference (
82     ACPI_EVALUATE_INFO      *Info,
83     ACPI_OPERAND_OBJECT     *ReturnObject);

85 static UINT32
86 AcpiNsGetBitmappedType (
87     ACPI_OPERAND_OBJECT     *ReturnObject);

90 /*****
91 *
92 * FUNCTION:      AcpiNsCheckReturnValue
93 *
94 * PARAMETERS:    Node           - Namespace node for the method/object
95 *                Info           - Method execution information block
96 *                UserParamCount - Number of parameters actually passed
97 *                ReturnStatus    - Status from the object evaluation
98 *                ReturnObjectPtr - Pointer to the object returned from the
99 *                               evaluation of a method or object
100 *
101 * RETURN:        Status
102 *
103 * DESCRIPTION:   Check the value returned from a predefined name.
104 *
105 *****/

107 ACPI_STATUS
108 AcpiNsCheckReturnValue (
109     ACPI_NAMESPACE_NODE     *Node,
110     ACPI_EVALUATE_INFO      *Info,
111     UINT32                  UserParamCount,
112     ACPI_STATUS              ReturnStatus,
113     ACPI_OPERAND_OBJECT     **ReturnObjectPtr)
114 {
115     ACPI_STATUS              Status;
116     const ACPI_PREDEFINED_INFO *Predefined;

119     /* If not a predefined name, we cannot validate the return object */

121     Predefined = Info->Predefined;
122     if (!Predefined)
123     {
124         return (AE_OK);
125     }
```

```

127  /*
128  * If the method failed or did not actually return an object, we cannot
129  * validate the return object
130  */
131  if ((ReturnStatus != AE_OK) &&
132      (ReturnStatus != AE_CTRL_RETURN_VALUE))
133  {
134      return (AE_OK);
135  }

137  /*
138  * Return value validation and possible repair.
139  *
140  * 1) Don't perform return value validation/repair if this feature
141  * has been disabled via a global option.
142  *
143  * 2) We have a return value, but if one wasn't expected, just exit,
144  * this is not a problem. For example, if the "Implicit Return"
145  * feature is enabled, methods will always return a value.
146  *
147  * 3) If the return value can be of any type, then we cannot perform
148  * any validation, just exit.
149  */
150  if (AcpiGbl_DisableAutoRepair ||
151      (!Predefined->Info.ExpectedBtypes) ||
152      (Predefined->Info.ExpectedBtypes == ACPI_RTYPE_ALL))
153  {
154      return (AE_OK);
155  }

157  /*
158  * Check that the type of the main return object is what is expected
159  * for this predefined name
160  */
161  Status = AcpiNsCheckObjectType (Info, ReturnObjectPtr,
162      Predefined->Info.ExpectedBtypes, ACPI_NOT_PACKAGE_ELEMENT);
163  if (ACPI_FAILURE (Status))
164  {
165      goto Exit;
166  }

168  /*
169  *
170  * 4) If there is no return value and it is optional, just return
171  * AE_OK (_WAK).
172  */
173  if (!(*ReturnObjectPtr))
174  {
175      goto Exit;
176  }

178  /*
179  * For returned Package objects, check the type of all sub-objects.
180  * Note: Package may have been newly created by call above.
181  */
182  if ((*ReturnObjectPtr)->Common.Type == ACPI_TYPE_PACKAGE)
183  {
184      Info->ParentPackage = *ReturnObjectPtr;
185      Status = AcpiNsCheckPackage (Info, ReturnObjectPtr);
186      if (ACPI_FAILURE (Status))
187      {
188          /* We might be able to fix some errors */

190          if ((Status != AE_AML_OPERAND_TYPE) &&
191              (Status != AE_AML_OPERAND_VALUE))

```

```

192      {
193          goto Exit;
194      }
195  }
196  }

198  /*
199  * The return object was OK, or it was successfully repaired above.
200  * Now make some additional checks such as verifying that package
201  * objects are sorted correctly (if required) or buffer objects have
202  * the correct data width (bytes vs. dwords). These repairs are
203  * performed on a per-name basis, i.e., the code is specific to
204  * particular predefined names.
205  */
206  Status = AcpiNsComplexRepairs (Info, Node, Status, ReturnObjectPtr);

208 Exit:
209  /*
210  * If the object validation failed or if we successfully repaired one
211  * or more objects, mark the parent node to suppress further warning
212  * messages during the next evaluation of the same method/object.
213  */
214  if (ACPI_FAILURE (Status) ||
215      (Info->ReturnFlags & ACPI_OBJECT_REPAIRED))
216  {
217      Node->Flags |= ANOBJ_EVALUATED;
218  }

220  return (Status);
221  }

224  /*****
225  *
226  * FUNCTION:      AcpiNsCheckObjectType
227  *
228  * PARAMETERS:    Info          - Method execution information block
229  *                 ReturnObjectPtr - Pointer to the object returned from the
230  *                               evaluation of a method or object
231  *                 ExpectedBtypes - Bitmap of expected return type(s)
232  *                 PackageIndex  - Index of object within parent package (if
233  *                               applicable - ACPI_NOT_PACKAGE_ELEMENT
234  *                               otherwise)
235  *
236  * RETURN:        Status
237  *
238  * DESCRIPTION:   Check the type of the return object against the expected object
239  *                 type(s). Use of Btype allows multiple expected object types.
240  *
241  *****/

243  ACPI_STATUS
244  AcpiNsCheckObjectType (
245      ACPI_EVALUATE_INFO *Info,
246      ACPI_OPERAND_OBJECT **ReturnObjectPtr,
247      UINT32 ExpectedBtypes,
248      UINT32 PackageIndex)
249  {
250      ACPI_OPERAND_OBJECT *ReturnObject = *ReturnObjectPtr;
251      ACPI_STATUS Status = AE_OK;
252      char TypeBuffer[48]; /* Room for 5 types */

255  /* A Namespace node should not get here, but make sure */

257  if (ReturnObject &&

```

```

258     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 }
266
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 }
281
282 /* For reference objects, check that the reference type is correct */
283
284 if ((Info->ReturnBtype & ExpectedBtypes) == ACPI_RTYPE_REFERENCE)
285 {
286     Status = AcpiNsCheckReference (Info, ReturnObject);
287     return (Status);
288 }
289
290 /* Attempt simple repair of the returned object if necessary */
291
292 Status = AcpiNsSimpleRepair (Info, ExpectedBtypes,
293     PackageIndex, ReturnObjectPtr);
294 if (ACPI_SUCCESS (Status))
295 {
296     return (AE_OK); /* Successful repair */
297 }
298
300 TypeErrorExit:
301
302 /* Create a string with all expected types for this predefined object */
303
304 AcpiUtGetExpectedReturnTypes (TypeBuffer, ExpectedBtypes);
305
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 }
325
326 return (AE_AML_OPERAND_TYPE);
327 }
328
329
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  *****/
345
346 static ACPI_STATUS
347 AcpiNsCheckReference (
348     ACPI_EVALUATE_INFO      *Info,
349     ACPI_OPERAND_OBJECT     *ReturnObject)
350 {
351     /*
352     * Check the reference object for the correct reference type (opcode).
353     * The only type of reference that can be converted to an ACPI_OBJECT is
354     * a reference to a named object (reference class: NAME)
355     */
356     if (ReturnObject->Reference.Class == ACPI_REFCLASS_NAME)
357     {
358         return (AE_OK);
359     }
360 }
361
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));
366
367 return (AE_AML_OPERAND_TYPE);
368 }
369
370
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  *****/
384
385 static UINT32
386 AcpiNsGetBitmappedType (
387     ACPI_OPERAND_OBJECT     *ReturnObject)
388 {
389     UINT32
390         ReturnBtype;

```

```
392     if (!ReturnObject)
393     {
394         return (ACPI_RTYPE_NONE);
395     }
396
397     /* Map ACPI_OBJECT_TYPE to internal bitmapped type */
398
399     switch (ReturnObject->Common.Type)
400     {
401     case ACPI_TYPE_INTEGER:
402
403         ReturnBtype = ACPI_RTYPE_INTEGER;
404         break;
405
406     case ACPI_TYPE_BUFFER:
407
408         ReturnBtype = ACPI_RTYPE_BUFFER;
409         break;
410
411     case ACPI_TYPE_STRING:
412
413         ReturnBtype = ACPI_RTYPE_STRING;
414         break;
415
416     case ACPI_TYPE_PACKAGE:
417
418         ReturnBtype = ACPI_RTYPE_PACKAGE;
419         break;
420
421     case ACPI_TYPE_LOCAL_REFERENCE:
422
423         ReturnBtype = ACPI_RTYPE_REFERENCE;
424         break;
425
426     default:
427
428         /* Not one of the supported objects, must be incorrect */
429
430         ReturnBtype = ACPI_RTYPE_ANY;
431         break;
432     }
433
434     return (ReturnBtype);
435 }
```

new/usr/src/common/acpica/components/namespace/nsprepkg.c

1

```
*****
20409 Fri Jan 17 07:30:27 2014
new/usr/src/common/acpica/components/namespace/nsprepkg.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: nsprepkg - Validation of package objects for predefined names
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acnamesp.h"
47 #include "acpredef.h"

50 #define _COMPONENT          ACPI_NAMESPACE
51       ACPI_MODULE_NAME    ("nsprepkg")

54 /* Local prototypes */

56 static ACPI_STATUS
57 AcpiNsCheckPackageList (
58     ACPI_EVALUATE_INFO    *Info,
59     const ACPI_PREDEFINED_INFO *Package,
60     ACPI_OPERAND_OBJECT    **Elements,
```

new/usr/src/common/acpica/components/namespace/nsprepkg.c

2

```
61     UINT32                Count);

63 static ACPI_STATUS
64 AcpiNsCheckPackageElements (
65     ACPI_EVALUATE_INFO    *Info,
66     ACPI_OPERAND_OBJECT    **Elements,
67     UINT8                  Type1,
68     UINT32                  Count1,
69     UINT8                  Type2,
70     UINT32                  Count2,
71     UINT32                  StartIndex);

74 /*****
75 *
76 * FUNCTION:      AcpiNsCheckPackage
77 *
78 * PARAMETERS:    Info          - Method execution information block
79 *                ReturnObjectPtr - Pointer to the object returned from the
80 *                                evaluation of a method or object
81 *
82 * RETURN:        Status
83 *
84 * DESCRIPTION:   Check a returned package object for the correct count and
85 *                correct type of all sub-objects.
86 *
87 *****/

89 ACPI_STATUS
90 AcpiNsCheckPackage (
91     ACPI_EVALUATE_INFO    *Info,
92     ACPI_OPERAND_OBJECT    **ReturnObjectPtr)
93 {
94     ACPI_OPERAND_OBJECT    *ReturnObject = *ReturnObjectPtr;
95     const ACPI_PREDEFINED_INFO *Package;
96     ACPI_OPERAND_OBJECT    **Elements;
97     ACPI_STATUS              Status = AE_OK;
98     UINT32                   ExpectedCount;
99     UINT32                   Count;
100     UINT32                   i;

103     ACPI_FUNCTION_NAME (NsCheckPackage);

106     /* The package info for this name is in the next table entry */

108     Package = Info->Predefined + 1;

110     ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
111         "%s Validating return Package of Type %X, Count %X\n",
112         Info->FullPathname, Package->RetInfo.Type,
113         ReturnObject->Package.Count));

115     /*
116      * For variable-length Packages, we can safely remove all embedded
117      * and trailing NULL package elements
118      */
119     AcpiNsRemoveNullElements (Info, Package->RetInfo.Type, ReturnObject);

121     /* Extract package count and elements array */

123     Elements = ReturnObject->Package.Elements;
124     Count = ReturnObject->Package.Count;

126     /*
```

```

127  * Most packages must have at least one element. The only exception
128  * is the variable-length package (ACPI_PTTYPE1_VAR).
129  */
130  if (!Count)
131  {
132      if (Package->RetInfo.Type == ACPI_PTTYPE1_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  * PTTYPE1 packages contain no subpackages
147  * PTTYPE2 packages contain sub-packages
148  */
149  switch (Package->RetInfo.Type)
150  {
151  case ACPI_PTTYPE1_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_PTTYPE1_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;
194
195  case ACPI_PTTYPE1_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_PTTYPE2_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_PTTYPE2_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 }

```

```

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 *****/

```

```

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      * Up to two groups of package elements are supported by the data
604      * structure. All elements in each group must be of the same type.
605      * The second group can have a count of zero.
606      */
607     for (i = 0; i < Count1; i++)
608     {
609         Status = AcpiNsCheckObjectType (Info, ThisElement,
610         Type1, i + StartIndex);
611         if (ACPI_FAILURE (Status))
612         {
613             return (Status);
614         }
615         ThisElement++;
616     }
617
618     for (i = 0; i < Count2; i++)
619     {
620         Status = AcpiNsCheckObjectType (Info, ThisElement,
621         Type2, (i + Count1 + StartIndex));
622         if (ACPI_FAILURE (Status))
623         {
624             return (Status);
625         }
626         ThisElement++;
627     }
628
629     return (AE_OK);
630 }
631

```

new/usr/src/common/acpica/components/namespace/nsrepair.c

1

```
*****
2012 Fri Jan 17 07:30:27 2014
new/usr/src/common/acpica/components/namespace/nsrepair.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: nsrepair - Repair for objects returned by predefined methods
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __NSREPAIR_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acnamesp.h"
49 #include "acinterp.h"
50 #include "acpredef.h"
51 #include "amlresrc.h"

53 #define _COMPONENT          ACPI_NAMESPACE
54       ACPI_MODULE_NAME    ("nsrepair")

57 /*****
58 *
59 * This module attempts to repair or convert objects returned by the
```

new/usr/src/common/acpica/components/namespace/nsrepair.c

2

```
60 * predefined methods to an object type that is expected, as per the ACPI
61 * specification. The need for this code is dictated by the many machines that
62 * return incorrect types for the standard predefined methods. Performing these
63 * conversions here, in one place, eliminates the need for individual ACPI
64 * device drivers to do the same. Note: Most of these conversions are different
65 * than the internal object conversion routines used for implicit object
66 * conversion.
67 *
68 * The following conversions can be performed as necessary:
69 *
70 * Integer -> String
71 * Integer -> Buffer
72 * String -> Integer
73 * String -> Buffer
74 * Buffer -> Integer
75 * Buffer -> String
76 * Buffer -> Package of Integers
77 * Package -> Package of one Package
78 *
79 * Additional conversions that are available:
80 * Convert a null return or zero return value to an EndTag descriptor
81 * Convert an ASCII string to a Unicode buffer
82 *
83 * An incorrect standalone object is wrapped with required outer package
84 *
85 * Additional possible repairs:
86 * Required package elements that are NULL replaced by Integer/String/Buffer
87 *
88 *****/

91 /* Local prototypes */

93 static const ACPI_SIMPLE_REPAIR_INFO *
94 AcpiNsMatchSimpleRepair (
95     ACPI_NAMESPACE_NODE *Node,
96     UINT32 ReturnBtype,
97     UINT32 PackageIndex);

100 /*
101 * Special but simple repairs for some names.
102 *
103 * 2nd argument: Unexpected types that can be repaired
104 */
105 static const ACPI_SIMPLE_REPAIR_INFO AcpiObjectRepairInfo[] =
106 {
107     /* Resource descriptor conversions */

109     { "_CRS", ACPI_RTYPE_INTEGER | ACPI_RTYPE_STRING | ACPI_RTYPE_BUFFER | ACPI_
110               ACPI_NOT_PACKAGE_ELEMENT,
111       AcpiNsConvertToResource },
112     { "_DMA", ACPI_RTYPE_INTEGER | ACPI_RTYPE_STRING | ACPI_RTYPE_BUFFER | ACPI_
113               ACPI_NOT_PACKAGE_ELEMENT,
114       AcpiNsConvertToResource },
115     { "_PRS", ACPI_RTYPE_INTEGER | ACPI_RTYPE_STRING | ACPI_RTYPE_BUFFER | ACPI_
116               ACPI_NOT_PACKAGE_ELEMENT,
117       AcpiNsConvertToResource },

119     /* Unicode conversions */

121     { "_MLS", ACPI_RTYPE_STRING, 1,
122       AcpiNsConvertToUnicode },
123     { "_STR", ACPI_RTYPE_STRING | ACPI_RTYPE_BUFFER,
124       ACPI_NOT_PACKAGE_ELEMENT,
125       AcpiNsConvertToUnicode },
```

```

126     { {0,0,0,0}, 0, 0, NULL } /* Table terminator */
127 };

130 /*****
131  *
132  * FUNCTION:      AcpiNsSimpleRepair
133  *
134  * PARAMETERS:    Info           - Method execution information block
135  *                ExpectedBtypes - Object types expected
136  *                PackageIndex   - Index of object within parent package (if
137  *                                applicable - ACPI_NOT_PACKAGE_ELEMENT
138  *                                otherwise)
139  *                ReturnObjectPtr - Pointer to the object returned from the
140  *                                evaluation of a method or object
141  *
142  * RETURN:        Status. AE_OK if repair was successful.
143  *
144  * DESCRIPTION:   Attempt to repair/convert a return object of a type that was
145  *                not expected.
146  *
147  *****/

149 ACPI_STATUS
150 AcpiNsSimpleRepair (
151     ACPI_EVALUATE_INFO *Info,
152     UINT32 ExpectedBtypes,
153     UINT32 PackageIndex,
154     ACPI_OPERAND_OBJECT **ReturnObjectPtr)
155 {
156     ACPI_OPERAND_OBJECT *ReturnObject = *ReturnObjectPtr;
157     ACPI_OPERAND_OBJECT *NewObject = NULL;
158     ACPI_STATUS Status;
159     const ACPI_SIMPLE_REPAIR_INFO *Predefined;

162     ACPI_FUNCTION_NAME (NsSimpleRepair);

165     /*
166     * Special repairs for certain names that are in the repair table.
167     * Check if this name is in the list of repairable names.
168     */
169     Predefined = AcpiNsMatchSimpleRepair (Info->Node,
170     Info->ReturnBtype, PackageIndex);
171     if (Predefined)
172     {
173         if (!ReturnObject)
174         {
175             ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname,
176             ACPI_WARN_ALWAYS, "Missing expected return value"));
177         }

179         Status = Predefined->ObjectConverter (ReturnObject, &NewObject);
180         if (ACPI_FAILURE (Status))
181         {
182             /* A fatal error occurred during a conversion */

184             ACPI_EXCEPTION ((AE_INFO, Status,
185             "During return object analysis"));
186             return (Status);
187         }
188         if (NewObject)
189         {
190             goto ObjectRepaired;
191         }

```

```

192     }

194     /*
195     * Do not perform simple object repair unless the return type is not
196     * expected.
197     */
198     if (Info->ReturnBtype & ExpectedBtypes)
199     {
200         return (AE_OK);
201     }

203     /*
204     * At this point, we know that the type of the returned object was not
205     * one of the expected types for this predefined name. Attempt to
206     * repair the object by converting it to one of the expected object
207     * types for this predefined name.
208     */

210     /*
211     * If there is no return value, check if we require a return value for
212     * this predefined name. Either one return value is expected, or none,
213     * for both methods and other objects.
214     *
215     * Exit now if there is no return object. Warning if one was expected.
216     */
217     if (!ReturnObject)
218     {
219         if (ExpectedBtypes && (!(ExpectedBtypes & ACPI_RTYPE_NONE)))
220         {
221             ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname,
222             ACPI_WARN_ALWAYS, "Missing expected return value"));
223         }
224         return (AE_AML_NO_RETURN_VALUE);
225     }

228     if (ExpectedBtypes & ACPI_RTYPE_INTEGER)
229     {
230         Status = AcpiNsConvertToInteger (ReturnObject, &NewObject);
231         if (ACPI_SUCCESS (Status))
232         {
233             goto ObjectRepaired;
234         }
235     }
236     if (ExpectedBtypes & ACPI_RTYPE_STRING)
237     {
238         Status = AcpiNsConvertToString (ReturnObject, &NewObject);
239         if (ACPI_SUCCESS (Status))
240         {
241             goto ObjectRepaired;
242         }
243     }
244     if (ExpectedBtypes & ACPI_RTYPE_BUFFER)
245     {
246         Status = AcpiNsConvertToBuffer (ReturnObject, &NewObject);
247         if (ACPI_SUCCESS (Status))
248         {
249             goto ObjectRepaired;
250         }
251     }
252     if (ExpectedBtypes & ACPI_RTYPE_PACKAGE)
253     {
254         /*
255         * A package is expected. We will wrap the existing object with a
256         * new package object. It is often the case that if a variable-length
257         * package is required, but there is only a single object needed, the

```

```

258     * 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 }
274
275 /* We cannot repair this object */
276
277 return (AE_AML_OPERAND_TYPE);
278
279 ObjectRepaired:
280
281 /* Object was successfully repaired */
282
283 if (PackageIndex != ACPI_NOT_PACKAGE_ELEMENT)
284 {
285     /*
286      * The original object is a package element. We need to
287      * decrement the reference count of the original object,
288      * for removing it from the package.
289      *
290      * However, if the original object was just wrapped with a
291      * package object as part of the repair, we don't need to
292      * change the reference count.
293      */
294     if (!(Info->ReturnFlags & ACPI_OBJECT_WRAPPED))
295     {
296         NewObject->Common.ReferenceCount =
297             ReturnObject->Common.ReferenceCount;
298
299         if (ReturnObject->Common.ReferenceCount > 1)
300         {
301             ReturnObject->Common.ReferenceCount--;
302         }
303     }
304
305     ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
306         "%s: Converted %s to expected %s at Package index %u\n",
307         Info->FullPathname, AcpiUtGetObjectTypeName (ReturnObject),
308         AcpiUtGetObjectTypeName (NewObject), PackageIndex));
309 }
310 else
311 {
312     ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
313         "%s: Converted %s to expected %s\n",
314         Info->FullPathname, AcpiUtGetObjectTypeName (ReturnObject),
315         AcpiUtGetObjectTypeName (NewObject)));
316 }
317
318 /* Delete old object, install the new return object */
319
320 AcpiUtRemoveReference (ReturnObject);
321 *ReturnObjectPtr = NewObject;
322 Info->ReturnFlags |= ACPI_OBJECT_REPAIRED;

```

```

324     return (AE_OK);
325 }
326
327 /*****
328  *
329  * FUNCTION:      AcpiNsMatchSimpleRepair
330  *
331  * PARAMETERS:    Node           - Namespace node for the method/object
332  *                ReturnBtype    - Object type that was returned
333  *                PackageIndex    - Index of object within parent package (if
334  *                                applicable - ACPI_NOT_PACKAGE_ELEMENT
335  *                                otherwise)
336  *
337  * RETURN:        Pointer to entry in repair table. NULL indicates not found.
338  *
339  * DESCRIPTION:   Check an object name against the repairable object list.
340  *
341  *****/
342
343 static const ACPI_SIMPLE_REPAIR_INFO *
344 AcpiNsMatchSimpleRepair (
345     ACPI_NAMESPACE_NODE *Node,
346     UINT32 ReturnBtype,
347     UINT32 PackageIndex)
348 {
349     const ACPI_SIMPLE_REPAIR_INFO *ThisName;
350
351     /* Search info table for a repairable predefined method/object name */
352
353     ThisName = AcpiObjectRepairInfo;
354     while (ThisName->ObjectConverter)
355     {
356         if (ACPI_COMPARE_NAME (Node->Name.Ascii, ThisName->Name))
357         {
358             /* Check if we can actually repair this name/type combination */
359
360             if ((ReturnBtype & ThisName->UnexpectedBtypes) &&
361                 (PackageIndex == ThisName->PackageIndex))
362             {
363                 return (ThisName);
364             }
365
366             return (NULL);
367         }
368         ThisName++;
369     }
370
371     return (NULL); /* Name was not found in the repair table */
372 }
373
374 /*****
375  *
376  * FUNCTION:      AcpiNsRepairNullElement
377  *
378  * PARAMETERS:    Info           - Method execution information block
379  *                ExpectedBtypes - Object types expected
380  *                PackageIndex    - Index of object within parent package (if
381  *                                applicable - ACPI_NOT_PACKAGE_ELEMENT
382  *                                otherwise)
383  *                ReturnObjectPtr - Pointer to the object returned from the
384  *                                evaluation of a method or object
385  *
386  * RETURN:        Status. AE_OK if repair was successful.
387  *****/

```

```

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;
404
405     ACPI_FUNCTION_NAME (NsRepairNullElement);
406
407     /* No repair needed if return object is non-NULL */
408
409     if (ReturnObject)
410     {
411         return (AE_OK);
412     }
413
414     /*
415     * Attempt to repair a NULL element of a Package object. This applies to
416     * predefined names that return a fixed-length package and each element
417     * is required. It does not apply to variable-length packages where NULL
418     * elements are allowed, especially at the end of the package.
419     */
420     if (ExpectedBtypes & ACPI_RTYPE_INTEGER)
421     {
422         /* Need an Integer - create a zero-value integer */
423         NewObject = AcpiUtCreateIntegerObject ((UINT64) 0);
424     }
425     else if (ExpectedBtypes & ACPI_RTYPE_STRING)
426     {
427         /* Need a String - create a NULL string */
428         NewObject = AcpiUtCreateStringObject (0);
429     }
430     else if (ExpectedBtypes & ACPI_RTYPE_BUFFER)
431     {
432         /* Need a Buffer - create a zero-length buffer */
433         NewObject = AcpiUtCreateBufferObject (0);
434     }
435     else
436     {
437         /* Error for all other expected types */
438         return (AE_AML_OPERAND_TYPE);
439     }
440
441     if (!NewObject)
442     {
443         return (AE_NO_MEMORY);
444     }
445
446     /* Set the reference count according to the parent Package object */
447
448     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))
459
460     *ReturnObjectPtr = NewObject;
461     Info->ReturnFlags |= ACPI_OBJECT_REPAIRED;
462     return (AE_OK);
463 }
464
465 *****/
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 *****/
481
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;
493
494     ACPI_FUNCTION_NAME (NsRemoveNullElements);
495
496     /*
497     * We can safely remove all NULL elements from these package types:
498     * PTYPE1_VAR packages contain a variable number of simple data types.
499     * PTYPE2 packages contain a variable number of sub-packages.
500     */
501     switch (PackageType)
502     {
503     case ACPI_PTYPE1_VAR:
504     case ACPI_PTYPE2:
505     case ACPI_PTYPE2_COUNT:
506     case ACPI_PTYPE2_PKG_COUNT:
507     case ACPI_PTYPE2_FIXED:
508     case ACPI_PTYPE2_MIN:
509     case ACPI_PTYPE2_REV_FIXED:
510     case ACPI_PTYPE2_FIX_VAR:
511
512         break;
513
514     default:
515     case ACPI_PTYPE1_FIXED:
516     case ACPI_PTYPE1_OPTION:
517         return;
518     }
519 }

```

```

522     Count = ObjDesc->Package.Count;
523     NewCount = Count;

525     Source = ObjDesc->Package.Elements;
526     Dest = Source;

528     /* Examine all elements of the package object, remove nulls */

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     }

544     /* Update parent package if any null elements were removed */

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)));

552         /* NULL terminate list and update the package count */

554         *Dest = NULL;
555         ObjDesc->Package.Count = NewCount;
556     }
557 }

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  *****/

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;

593     ACPI_FUNCTION_NAME (NsWrapWithPackage);

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     }

606     PkgObjDesc->Package.Elements[0] = OriginalObject;

608     ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
609         "%s: Wrapped %s with expected Package object\n",
610         Info->FullPathname, AcpiUtGetObjectTypeName (OriginalObject)));

612     /* Return the new object in the object pointer */

614     *ObjDescPtr = PkgObjDesc;
615     Info->ReturnFlags |= ACPI_OBJECT_REPAIRED | ACPI_OBJECT_WRAPPED;
616     return (AE_OK);
617 }

```



new/usr/src/common/acpica/components/namespace/nsrepair2.c

1

```
*****
33442 Fri Jan 17 07:30:27 2014
new/usr/src/common/acpica/components/namespace/nsrepair2.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: nsrepair2 - Repair for objects returned by specific
4 * predefined methods
5 *
6 *****/

8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __NSREPAIR2_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acnamesp.h"

51 #define _COMPONENT          ACPI_NAMESPACE
52       ACPI_MODULE_NAME    ("nsrepair2")

55 /*
56 * Information structure and handler for ACPI predefined names that can
57 * be repaired on a per-name basis.
58 */
59 typedef
```

new/usr/src/common/acpica/components/namespace/nsrepair2.c

2

```
60 ACPI_STATUS (*ACPI_REPAIR_FUNCTION) (
61     ACPI_EVALUATE_INFO    *Info,
62     ACPI_OPERAND_OBJECT    **ReturnObjectPtr);

64 typedef struct acpi_repair_info
65 {
66     char                    Name[ACPI_NAME_SIZE];
67     ACPI_REPAIR_FUNCTION    RepairFunction;
68 } ACPI_REPAIR_INFO;

72 /* Local prototypes */

74 static const ACPI_REPAIR_INFO *
75 AcpiNsMatchComplexRepair (
76     ACPI_NAMESPACE_NODE    *Node);

78 static ACPI_STATUS
79 AcpiNsRepair_ALR (
80     ACPI_EVALUATE_INFO    *Info,
81     ACPI_OPERAND_OBJECT    **ReturnObjectPtr);

83 static ACPI_STATUS
84 AcpiNsRepair_CID (
85     ACPI_EVALUATE_INFO    *Info,
86     ACPI_OPERAND_OBJECT    **ReturnObjectPtr);

88 static ACPI_STATUS
89 AcpiNsRepair_CST (
90     ACPI_EVALUATE_INFO    *Info,
91     ACPI_OPERAND_OBJECT    **ReturnObjectPtr);

93 static ACPI_STATUS
94 AcpiNsRepair_FDE (
95     ACPI_EVALUATE_INFO    *Info,
96     ACPI_OPERAND_OBJECT    **ReturnObjectPtr);

98 static ACPI_STATUS
99 AcpiNsRepair_HID (
100     ACPI_EVALUATE_INFO    *Info,
101     ACPI_OPERAND_OBJECT    **ReturnObjectPtr);

103 static ACPI_STATUS
104 AcpiNsRepair_PRT (
105     ACPI_EVALUATE_INFO    *Info,
106     ACPI_OPERAND_OBJECT    **ReturnObjectPtr);

108 static ACPI_STATUS
109 AcpiNsRepair_PSS (
110     ACPI_EVALUATE_INFO    *Info,
111     ACPI_OPERAND_OBJECT    **ReturnObjectPtr);

113 static ACPI_STATUS
114 AcpiNsRepair_TSS (
115     ACPI_EVALUATE_INFO    *Info,
116     ACPI_OPERAND_OBJECT    **ReturnObjectPtr);

118 static ACPI_STATUS
119 AcpiNsCheckSortedList (
120     ACPI_EVALUATE_INFO    *Info,
121     ACPI_OPERAND_OBJECT    **ReturnObject,
122     UINT32                StartIndex,
123     UINT32                ExpectedCount,
124     UINT32                SortIndex,
125     UINT8                 SortDirection,
```

```

126     char                *SortKeyName);

128 /* Values for SortDirection above */

130 #define ACPI_SORT_ASCENDING    0
131 #define ACPI_SORT_DESCENDING  1

133 static void
134 AcpiNsRemoveElement (
135     ACPI_OPERAND_OBJECT *ObjDesc,
136     UINT32                Index);

138 static void
139 AcpiNsSortList (
140     ACPI_OPERAND_OBJECT **Elements,
141     UINT32                Count,
142     UINT32                Index,
143     UINT8                SortDirection);

146 /*
147  * This table contains the names of the predefined methods for which we can
148  * perform more complex repairs.
149  *
150  * As necessary:
151  *
152  * _ALR: Sort the list ascending by AmbientIlluminance
153  * _CID: Strings: uppercase all, remove any leading asterisk
154  * _CST: Sort the list ascending by C state type
155  * _FDE: Convert Buffer of BYTES to a Buffer of DWORDs
156  * _GTM: Convert Buffer of BYTES to a Buffer of DWORDs
157  * _HID: Strings: uppercase all, remove any leading asterisk
158  * _PRT: Fix reversed SourceName and SourceIndex
159  * _PSS: Sort the list descending by Power
160  * _TSS: Sort the list descending by Power
161  *
162  * Names that must be packages, but cannot be sorted:
163  *
164  * _BCL: Values are tied to the Package index where they appear, and cannot
165  * be moved or sorted. These index values are used for _BQC and _BCM.
166  * However, we can fix the case where a buffer is returned, by converting
167  * it to a Package of integers.
168  */
169 static const ACPI_REPAIR_INFO    AcpiNsRepairableNames[] =
170 {
171     {"_ALR", AcpiNsRepair_ALR},
172     {"_CID", AcpiNsRepair_CID},
173     {"_CST", AcpiNsRepair_CST},
174     {"_FDE", AcpiNsRepair_FDE},
175     {"_GTM", AcpiNsRepair_FDE}, /* _GTM has same repair as _FDE */
176     {"_HID", AcpiNsRepair_HID},
177     {"_PRT", AcpiNsRepair_PRT},
178     {"_PSS", AcpiNsRepair_PSS},
179     {"_TSS", AcpiNsRepair_TSS},
180     {{0,0,0,0}, NULL} /* Table terminator */
181 };

184 #define ACPI_FDE_FIELD_COUNT    5
185 #define ACPI_FDE_BYTE_BUFFER_SIZE 5
186 #define ACPI_FDE_DWORD_BUFFER_SIZE (ACPI_FDE_FIELD_COUNT * sizeof (UINT32))

189 /*****
190  *
191  * FUNCTION:    AcpiNsComplexRepairs

```

```

192  *
193  * PARAMETERS:  Info                - Method execution information block
194  *              Node                - Namespace node for the method/object
195  *              ValidateStatus      - Original status of earlier validation
196  *              ReturnObjectPtr     - Pointer to the object returned from the
197  *                                  evaluation of a method or object
198  *
199  * RETURN:      Status. AE_OK if repair was successful. If name is not
200  *              matched, ValidateStatus is returned.
201  *
202  * DESCRIPTION: Attempt to repair/convert a return object of a type that was
203  *              not expected.
204  *
205  *****/

207 ACPI_STATUS
208 AcpiNsComplexRepairs (
209     ACPI_EVALUATE_INFO *Info,
210     ACPI_NAMESPACE_NODE *Node,
211     ACPI_STATUS          ValidateStatus,
212     ACPI_OPERAND_OBJECT **ReturnObjectPtr)
213 {
214     const ACPI_REPAIR_INFO *Predefined;
215     ACPI_STATUS              Status;

218     /* Check if this name is in the list of repairable names */

220     Predefined = AcpiNsMatchComplexRepair (Node);
221     if (!Predefined)
222     {
223         return (ValidateStatus);
224     }

226     Status = Predefined->RepairFunction (Info, ReturnObjectPtr);
227     return (Status);
228 }

231 /*****
232  *
233  * FUNCTION:    AcpiNsMatchComplexRepair
234  *
235  * PARAMETERS:  Node                - Namespace node for the method/object
236  *
237  * RETURN:      Pointer to entry in repair table. NULL indicates not found.
238  *
239  * DESCRIPTION: Check an object name against the repairable object list.
240  *
241  *****/

243 static const ACPI_REPAIR_INFO *
244 AcpiNsMatchComplexRepair (
245     ACPI_NAMESPACE_NODE *Node)
246 {
247     const ACPI_REPAIR_INFO *ThisName;

250     /* Search info table for a repairable predefined method/object name */

252     ThisName = AcpiNsRepairableNames;
253     while (ThisName->RepairFunction)
254     {
255         if (ACPI_COMPARE_NAME (Node->Name.Ascii, ThisName->Name))
256         {
257             return (ThisName);

```

```

258     }
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 }

```

```

391 /*****
392 *
393 * FUNCTION:    AcpiNsRepair_CID
394 *
395 * PARAMETERS:  Info          - Method execution information block
396 *              ReturnObjectPtr - Pointer to the object returned from the
397 *                              evaluation of a method or object
398 *
399 * RETURN:      Status. AE_OK if object is OK or was repaired successfully
400 *
401 * DESCRIPTION: Repair for the _CID object. If a string, ensure that all
402 *              letters are uppercase and that there is no leading asterisk.
403 *              If a Package, ensure same for all string elements.
404 *
405 *****/
406
407 static ACPI_STATUS
408 AcpiNsRepair_CID (
409     ACPI_EVALUATE_INFO *Info,
410     ACPI_OPERAND_OBJECT **ReturnObjectPtr)
411 {
412     ACPI_STATUS Status;
413     ACPI_OPERAND_OBJECT *ReturnObject = *ReturnObjectPtr;
414     ACPI_OPERAND_OBJECT **ElementPtr;
415     ACPI_OPERAND_OBJECT *OriginalElement;
416     UINT16 OriginalRefCount;
417     UINT32 i;
418
419     /* Check for _CID as a simple string */
420
421     if (ReturnObject->Common.Type == ACPI_TYPE_STRING)
422     {
423         Status = AcpiNsRepair_HID (Info, ReturnObjectPtr);
424         return (Status);
425     }
426
427     /* Exit if not a Package */
428
429     if (ReturnObject->Common.Type != ACPI_TYPE_PACKAGE)
430     {
431         return (AE_OK);
432     }
433
434     /* Examine each element of the _CID package */
435
436     ElementPtr = ReturnObject->Package.Elements;
437     for (i = 0; i < ReturnObject->Package.Count; i++)
438     {
439         OriginalElement = *ElementPtr;
440         OriginalRefCount = OriginalElement->Common.ReferenceCount;
441
442         Status = AcpiNsRepair_HID (Info, ElementPtr);
443         if (ACPI_FAILURE (Status))
444         {
445             return (Status);
446         }
447     }
448
449     /* Take care with reference counts */
450
451     if (OriginalElement != *ElementPtr)
452     {
453         /* Element was replaced */
454
455         (*ElementPtr)->Common.ReferenceCount =

```

```

456     OriginalRefCount;
457
458     AcpiUtRemoveReference (OriginalElement);
459 }
460
461     ElementPtr++;
462 }
463
464     return (AE_OK);
465 }
466
467 /*****
468 *
469 * FUNCTION:    AcpiNsRepair_CST
470 *
471 * PARAMETERS:  Info          - Method execution information block
472 *              ReturnObjectPtr - Pointer to the object returned from the
473 *                              evaluation of a method or object
474 *
475 * RETURN:      Status. AE_OK if object is OK or was repaired successfully
476 *
477 * DESCRIPTION: Repair for the _CST object:
478 *              1. Sort the list ascending by C state type
479 *              2. Ensure type cannot be zero
480 *              3. A sub-package count of zero means _CST is meaningless
481 *              4. Count must match the number of C state sub-packages
482 *
483 *
484 *****/
485
486 static ACPI_STATUS
487 AcpiNsRepair_CST (
488     ACPI_EVALUATE_INFO *Info,
489     ACPI_OPERAND_OBJECT **ReturnObjectPtr)
490 {
491     ACPI_OPERAND_OBJECT *ReturnObject = *ReturnObjectPtr;
492     ACPI_OPERAND_OBJECT **OuterElements;
493     UINT32 OuterElementCount;
494     ACPI_OPERAND_OBJECT *ObjDesc;
495     ACPI_STATUS Status;
496     BOOLEAN Removing;
497     UINT32 i;
498
499     ACPI_FUNCTION_NAME (NsRepair_CST);
500
501     /*
502     * Check if the C-state type values are proportional.
503     */
504     OuterElementCount = ReturnObject->Package.Count - 1;
505     i = 0;
506     while (i < OuterElementCount)
507     {
508         OuterElements = &ReturnObject->Package.Elements[i + 1];
509         Removing = FALSE;
510
511         if ((*OuterElements)->Package.Count == 0)
512         {
513             ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
514                 "SubPackage[%u] - removing entry due to zero count", i));
515             Removing = TRUE;
516             goto RemoveElement;
517         }
518
519         ObjDesc = (*OuterElements)->Package.Elements[1]; /* Index1 = Type */

```

```

522     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     }

529 RemoveElement:
530     if (Removing)
531     {
532         AcpiNsRemoveElement (ReturnObject, i + 1);
533         OuterElementCount--;
534     }
535     else
536     {
537         i++;
538     }
539 }

541 /* Update top-level package count, Type "Integer" checked elsewhere */

543 ObjDesc = ReturnObject->Package.Elements[0];
544 ObjDesc->Integer.Value = OuterElementCount;

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 }

557 return (AE_OK);
558 }

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  *****/

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;

587     ACPI_FUNCTION_NAME (NsRepair_HID);

```

```

590     /* We only care about string _HID objects (not integers) */

592     if (ReturnObject->Common.Type != ACPI_TYPE_STRING)
593     {
594         return (AE_OK);
595     }

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"));

602         /* Return AE_OK anyway, let driver handle it */

604         Info->ReturnFlags |= ACPI_OBJECT_REPAIRED;
605         return (AE_OK);
606     }

608     /* It is simplest to always create a new string object */

610     NewString = AcpiUtCreateStringObject (ReturnObject->String.Length);
611     if (!NewString)
612     {
613         return (AE_NO_MEMORY);
614     }

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--;

628         ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
629             "%s: Removed invalid leading asterisk\n", Info->FullPathname));
630     }

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     }

645     AcpiUtRemoveReference (ReturnObject);
646     *ReturnObjectPtr = NewString;
647     return (AE_OK);
648 }

651 /*****
652  *
653  * FUNCTION:      AcpiNsRepair_PRT

```

```

654 *
655 * PARAMETERS:  Info          - Method execution information block
656 *              ReturnObjectPtr - Pointer to the object returned from the
657 *                              evaluation of a method or object
658 *
659 * RETURN:      Status. AE_OK if object is OK or was repaired successfully
660 *
661 * DESCRIPTION: Repair for the _PRT object. If necessary, fix reversed
662 *              SourceName and SourceIndex field, a common BIOS bug.
663 *
664 *****/
665
666 static ACPI_STATUS
667 AcpiNsRepair_PRT (
668     ACPI_EVALUATE_INFO *Info,
669     ACPI_OPERAND_OBJECT **ReturnObjectPtr)
670 {
671     ACPI_OPERAND_OBJECT *PackageObject = *ReturnObjectPtr;
672     ACPI_OPERAND_OBJECT **TopObjectList;
673     ACPI_OPERAND_OBJECT **SubObjectList;
674     ACPI_OPERAND_OBJECT *ObjDesc;
675     UINT32 ElementCount;
676     UINT32 Index;
677
678     /* Each element in the _PRT package is a subpackage */
679
680     TopObjectList = PackageObject->Package.Elements;
681     ElementCount = PackageObject->Package.Count;
682
683     for (Index = 0; Index < ElementCount; Index++)
684     {
685         SubObjectList = (*TopObjectList)->Package.Elements;
686
687         /*
688          * If the BIOS has erroneously reversed the _PRT SourceName (index 2)
689          * and the SourceIndex (index 3), fix it. _PRT is important enough to
690          * workaroud this BIOS error. This also provides compatibility with
691          * other ACPI implementations.
692          */
693         ObjDesc = SubObjectList[3];
694         if (!ObjDesc || (ObjDesc->Common.Type != ACPI_TYPE_INTEGER))
695         {
696             SubObjectList[3] = SubObjectList[2];
697             SubObjectList[2] = ObjDesc;
698             Info->ReturnFlags |= ACPI_OBJECT_REPAIRED;
699
700             ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
701                 "PRT[%X]: Fixed reversed SourceName and SourceIndex",
702                 Index));
703         }
704
705         /* Point to the next ACPI_OPERAND_OBJECT in the top level package */
706         TopObjectList++;
707     }
708
709     return (AE_OK);
710 }
711
712
713
714
715 *****/
716 *
717 * FUNCTION:  AcpiNsRepair_PSS
718 *
719 * PARAMETERS: Info          - Method execution information block

```

```

720 *              ReturnObjectPtr - Pointer to the object returned from the
721 *                              evaluation of a method or object
722 *
723 * RETURN:      Status. AE_OK if object is OK or was repaired successfully
724 *
725 * DESCRIPTION: Repair for the _PSS object. If necessary, sort the object list
726 *              by the CPU frequencies. Check that the power dissipation values
727 *              are all proportional to CPU frequency (i.e., sorting by
728 *              frequency should be the same as sorting by power.)
729 *
730 *****/
731
732 static ACPI_STATUS
733 AcpiNsRepair_PSS (
734     ACPI_EVALUATE_INFO *Info,
735     ACPI_OPERAND_OBJECT **ReturnObjectPtr)
736 {
737     ACPI_OPERAND_OBJECT *ReturnObject = *ReturnObjectPtr;
738     ACPI_OPERAND_OBJECT **OuterElements;
739     ACPI_OPERAND_OBJECT *ObjDesc;
740     ACPI_OPERAND_OBJECT *PreviousValue;
741     ACPI_STATUS Status;
742     ACPI_STATUS i;
743
744     /*
745      * Entries (sub-packages) in the _PSS Package must be sorted by power
746      * dissipation, in descending order. If it appears that the list is
747      * incorrectly sorted, sort it. We sort by CpuFrequency, since this
748      * should be proportional to the power.
749      */
750     Status = AcpiNsCheckSortedList (Info, ReturnObject, 0, 6, 0,
751         ACPI_SORT_DESCENDING, "CpuFrequency");
752     if (ACPI_FAILURE (Status))
753     {
754         return (Status);
755     }
756
757     /*
758      * We now know the list is correctly sorted by CPU frequency. Check if
759      * the power dissipation values are proportional.
760      */
761     PreviousValue = ACPI_UINT32_MAX;
762     OuterElements = ReturnObject->Package.Elements;
763     OuterElementCount = ReturnObject->Package.Count;
764
765     for (i = 0; i < OuterElementCount; i++)
766     {
767         Elements = (*OuterElements)->Package.Elements;
768         ObjDesc = Elements[1]; /* Index1 = PowerDissipation */
769
770         if ((UINT32) ObjDesc->Integer.Value > PreviousValue)
771         {
772             ACPI_WARN_PREDEFINED ((AE_INFO, Info->FullPathname, Info->NodeFlags,
773                 "SubPackage[%u,%u] - suspicious power dissipation values",
774                 i-1, i));
775         }
776
777         PreviousValue = (UINT32) ObjDesc->Integer.Value;
778         OuterElements++;
779     }
780
781     return (AE_OK);
782 }

```

```

788 /*****
789 *
790 * FUNCTION:    AcpiNsRepair_TSS
791 *
792 * PARAMETERS:  Info           - Method execution information block
793 *              ReturnObjectPtr - Pointer to the object returned from the
794 *                              evaluation of a method or object
795 *
796 * RETURN:      Status. AE_OK if object is OK or was repaired successfully
797 *
798 * DESCRIPTION: Repair for the _TSS object. If necessary, sort the object list
799 *              descending by the power dissipation values.
800 *
801 *****/
802
803 static ACPI_STATUS
804 AcpiNsRepair_TSS (
805     ACPI_EVALUATE_INFO *Info,
806     ACPI_OPERAND_OBJECT **ReturnObjectPtr)
807 {
808     ACPI_OPERAND_OBJECT *ReturnObject = *ReturnObjectPtr;
809     ACPI_STATUS Status;
810     ACPI_NAMESPACE_NODE *Node;
811
812     /*
813     * We can only sort the _TSS return package if there is no _PSS in the
814     * same scope. This is because if _PSS is present, the ACPI specification
815     * dictates that the _TSS Power Dissipation field is to be ignored, and
816     * therefore some BIOSs leave garbage values in the _TSS Power field(s).
817     * In this case, it is best to just return the _TSS package as-is.
818     * (May, 2011)
819     */
820     Status = AcpiNsGetNode (Info->Node, "^_PSS",
821                             ACPI_NS_NO_UPSEARCH, &Node);
822     if (ACPI_SUCCESS (Status))
823     {
824         return (AE_OK);
825     }
826
827     Status = AcpiNsCheckSortedList (Info, ReturnObject, 0, 5, 1,
828                                     ACPI_SORT_DESCENDING, "PowerDissipation");
829
830     return (Status);
831 }
832
833 /*****
834 *
835 * FUNCTION:    AcpiNsCheckSortedList
836 *
837 * PARAMETERS:  Info           - Method execution information block
838 *              ReturnObject   - Pointer to the top-level returned object
839 *              StartIndex      - Index of the first sub-package
840 *              ExpectedCount    - Minimum length of each sub-package
841 *              SortIndex       - Sub-package entry to sort on
842 *              SortDirection    - Ascending or descending
843 *              SortKeyName     - Name of the SortIndex field
844 *
845 * RETURN:      Status. AE_OK if the list is valid and is sorted correctly or
846 *              has been repaired by sorting the list.
847 *
848 * DESCRIPTION: Check if the package list is valid and sorted correctly by the
849 *              SortIndex. If not, then sort the list.
850 *
851 *****/

```

```

852 *
853 *****/
854
855 static ACPI_STATUS
856 AcpiNsCheckSortedList (
857     ACPI_EVALUATE_INFO *Info,
858     ACPI_OPERAND_OBJECT *ReturnObject,
859     UINT32 StartIndex,
860     UINT32 ExpectedCount,
861     UINT32 SortIndex,
862     UINT8 SortDirection,
863     char *SortKeyName)
864 {
865     ACPI_OPERAND_OBJECT *OuterElementCount;
866     ACPI_OPERAND_OBJECT **OuterElements;
867     ACPI_OPERAND_OBJECT *ObjDesc;
868     UINT32 i;
869     UINT32 PreviousValue;
870
871     ACPI_FUNCTION_NAME (NsCheckSortedList);
872
873     /* The top-level object must be a package */
874
875     if (ReturnObject->Common.Type != ACPI_TYPE_PACKAGE)
876     {
877         return (AE_AML_OPERAND_TYPE);
878     }
879
880     /*
881     * NOTE: assumes list of sub-packages contains no NULL elements.
882     * Any NULL elements should have been removed by earlier call
883     * to AcpiNsRemoveNullElements.
884     */
885     OuterElementCount = ReturnObject->Package.Count;
886     if (!OuterElementCount || StartIndex >= OuterElementCount)
887     {
888         return (AE_AML_PACKAGE_LIMIT);
889     }
890
891     OuterElements = &ReturnObject->Package.Elements[StartIndex];
892     OuterElementCount -= StartIndex;
893
894     PreviousValue = 0;
895     if (SortDirection == ACPI_SORT_DESCENDING)
896     {
897         PreviousValue = ACPI_UINT32_MAX;
898     }
899
900     /* Examine each subpackage */
901
902     for (i = 0; i < OuterElementCount; i++)
903     {
904         /* Each element of the top-level package must also be a package */
905
906         if ((*OuterElements->Common.Type != ACPI_TYPE_PACKAGE)
907             {
908             return (AE_AML_OPERAND_TYPE);
909         }
910
911         /* Each sub-package must have the minimum length */
912
913         if ((*OuterElements->Package.Count < ExpectedCount)
914             {

```

```

918     return (AE_AML_PACKAGE_LIMIT);
919 }

921 Elements = (*OuterElements)->Package.Elements;
922 ObjDesc = Elements[SortIndex];

924 if (ObjDesc->Common.Type != ACPI_TYPE_INTEGER)
925 {
926     return (AE_AML_OPERAND_TYPE);
927 }

929 /*
930  * The list must be sorted in the specified order. If we detect a
931  * discrepancy, sort the entire list.
932  */
933 if (((SortDirection == ACPI_SORT_ASCENDING) &&
934     (ObjDesc->Integer.Value < PreviousValue)) ||
935     ((SortDirection == ACPI_SORT_DESCENDING) &&
936     (ObjDesc->Integer.Value > PreviousValue)))
937 {
938     AcpiNsSortList (&ReturnObject->Package.Elements[StartIndex],
939         OuterElementCount, SortIndex, SortDirection);

941     Info->ReturnFlags |= ACPI_OBJECT_REPAIRED;

943     ACPI_DEBUG_PRINT ((ACPI_DB_REPAIR,
944         "%s: Repaired unsorted list - now sorted by %s\n",
945         Info->FullPathname, SortKeyName));
946     return (AE_OK);
947 }

949 PreviousValue = (UINT32) ObjDesc->Integer.Value;
950 OuterElements++;
951 }

953 return (AE_OK);
954 }

957 /*****
958  *
959  * FUNCTION:      AcpiNsSortList
960  *
961  * PARAMETERS:    Elements          - Package object element list
962  *                Count             - Element count for above
963  *                Index             - Sort by which package element
964  *                SortDirection     - Ascending or Descending sort
965  *
966  * RETURN:        None
967  *
968  * DESCRIPTION:   Sort the objects that are in a package element list.
969  *
970  * NOTE: Assumes that all NULL elements have been removed from the package,
971  *        and that all elements have been verified to be of type Integer.
972  *
973  *****/

975 static void
976 AcpiNsSortList (
977     ACPI_OPERAND_OBJECT **Elements,
978     UINT32 Count,
979     UINT32 Index,
980     UINT8 SortDirection)
981 {
982     ACPI_OPERAND_OBJECT *ObjDesc1;
983     ACPI_OPERAND_OBJECT *ObjDesc2;

```

```

984     ACPI_OPERAND_OBJECT *TempObj;
985     UINT32 i;
986     UINT32 j;

989     /* Simple bubble sort */

991     for (i = 1; i < Count; i++)
992     {
993         for (j = (Count - 1); j >= i; j--)
994         {
995             ObjDesc1 = Elements[j-1]->Package.Elements[Index];
996             ObjDesc2 = Elements[j]->Package.Elements[Index];

998             if (((SortDirection == ACPI_SORT_ASCENDING) &&
999                 (ObjDesc1->Integer.Value > ObjDesc2->Integer.Value)) ||

1001                 ((SortDirection == ACPI_SORT_DESCENDING) &&
1002                 (ObjDesc1->Integer.Value < ObjDesc2->Integer.Value)))
1003             {
1004                 TempObj = Elements[j-1];
1005                 Elements[j-1] = Elements[j];
1006                 Elements[j] = TempObj;
1007             }
1008         }
1009     }
1010 }

1013 /*****
1014  *
1015  * FUNCTION:      AcpiNsRemoveElement
1016  *
1017  * PARAMETERS:    ObjDesc          - Package object element list
1018  *                Index            - Index of element to remove
1019  *
1020  * RETURN:        None
1021  *
1022  * DESCRIPTION:   Remove the requested element of a package and delete it.
1023  *
1024  *****/

1026 static void
1027 AcpiNsRemoveElement (
1028     ACPI_OPERAND_OBJECT *ObjDesc,
1029     UINT32 Index)
1030 {
1031     ACPI_OPERAND_OBJECT **Source;
1032     ACPI_OPERAND_OBJECT **Dest;
1033     UINT32 Count;
1034     UINT32 NewCount;
1035     i;

1038     ACPI_FUNCTION_NAME (NsRemoveElement);

1041     Count = ObjDesc->Package.Count;
1042     NewCount = Count - 1;

1044     Source = ObjDesc->Package.Elements;
1045     Dest = Source;

1047     /* Examine all elements of the package object, remove matched index */
1049     for (i = 0; i < Count; i++)

```



```
1050 {
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 }
```

new/usr/src/common/acpica/components/namespace/nssearch.c

1

```
*****
14383 Fri Jan 17 07:30:27 2014
new/usr/src/common/acpica/components/namespace/nssearch.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: nssearch - Namespace search
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __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 #define ACPI_MODULE_NAME ("nssearch")

57 /* Local prototypes */
```

new/usr/src/common/acpica/components/namespace/nssearch.c

2

```
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)));
124         }
125     }
126 #endif
```

```
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

new/usr/src/common/acpica/components/namespace/nsutils.c

1

```
*****
22821 Fri Jan 17 07:30:28 2014
new/usr/src/common/acpica/components/namespace/nsutils.c
update to acpica-unix2-20140114
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOV's 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 - 2014, 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 MERCHANTABILITY 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 #define ACPI_MODULE_NAME ("nsutils")
55 /* Local prototypes */
57 static BOOLEAN
```

new/usr/src/common/acpica/components/namespace/nsutils.c

2

```
58 AcpiNsValidPathSeparator (
59     char Sep);
57 #ifndef 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 *
111 * FUNCTION: AcpiNsValidRootPrefix
112 *
113 * PARAMETERS: Prefix - Character to be checked
114 *
115 * RETURN: TRUE if a valid prefix
116 *
117 * DESCRIPTION: Check if a character is a valid ACPI Root prefix
118 *
119 *****/
```

```

125 BOOLEAN
126 AcpiNsValidRootPrefix (
127     char          Prefix)
128 {
129
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 {
150
151     return ((BOOLEAN) (Sep == '.'));
152 }

155 /*****
156  *
157  * FUNCTION:      AcpiNsGetType
158  *
159  * PARAMETERS:    Node          - Parent Node to be examined
160  *
161  * RETURN:        Type field from Node whose handle is passed
162  *
163  * DESCRIPTION:    Return the type of a Namespace node
164  *
165  *****/

168 ACPI_OBJECT_TYPE
169 AcpiNsGetType (
170     ACPI_NAMESPACE_NODE *Node)
171 {
172     ACPI_FUNCTION_TRACE (NsGetType);

173     if (!Node)
174     {
175         ACPI_WARNING ((AE_INFO, "Null Node parameter"));
176         return_UINT8 (ACPI_TYPE_ANY);
177         return_UINT32 (ACPI_TYPE_ANY);
178     }

179     return_UINT8 (Node->Type);
180     return_UINT32 ((ACPI_OBJECT_TYPE) Node->Type);
181 }

184 /*****
185  *
186  * FUNCTION:      AcpiNsLocal
187  *
188  *****/

```

```

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);

156
157     if (!AcpiUtValidObjectType (Type))
158     {
159         /* Type code out of range */

160
161         ACPI_WARNING ((AE_INFO, "Invalid Object Type 0x%X", Type));
162         return_UINT32 (ACPI_NS_NORMAL);
163     }

164
165     return_UINT32 (AcpiGbl_NsProperties[Type] & ACPI_NS_LOCAL);
166     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;

190
191     ACPI_FUNCTION_ENTRY ();

192
193     NextExternalChar = Info->ExternalName;
194     Info->NumCarats = 0;
195     Info->NumSegments = 0;
196     Info->FullyQualified = FALSE;

197
198     /*
199     * For the internal name, the required length is 4 bytes per segment, plus
200     * 1 each for RootPrefix, MultiNamePrefixOp, segment count, trailing null
201     * (which is not really needed, but no there's harm in putting it there)
202     *
203     * strlen() + 1 covers the first NameSeg, which has no path separator
204     */

```

```

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) ||
342                 if (AcpiNsValidPathSeparator (*ExternalName) ||
343                     (*ExternalName == 0))
344             {
345                 /* Pad the segment with underscore(s) if segment is short */
346
347                 Result[i] = '_';
348             }
349             else
350             {
351                 /* Convert the character to uppercase and save it */
352
353                 Result[i] = (char) ACPI_TOUPPER ((int) *ExternalName);
354                 ExternalName++;
355             }
356         }
357
358         /* Now we must have a path separator, or the pathname is bad */
359
360         if (!ACPI_IS_PATH_SEPARATOR (*ExternalName) &&
361             if (!AcpiNsValidPathSeparator (*ExternalName) &&
362                 (*ExternalName != 0))
363         {
364             return ACPI_STATUS (AE_BAD_PATHNAME);
365             return ACPI_STATUS (AE_BAD_PARAMETER);
366         }
367
368         /* Move on the next segment */
369
370         ExternalName++;
371         Result += ACPI_NAME_SIZE;
372     }
373
374     /* Terminate the string */
375
376     *Result = 0;
377
378     if (Info->FullyQualified)
379     {
380         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Returning [%p] (abs) \"\\%s\\\"\\n",
381             InternalName, InternalName));
382     }
383     else
384     {
385         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "Returning [%p] (rel) \"%s\\\"\\n",
386             InternalName, InternalName));
387     }
388
389     return ACPI_STATUS (AE_OK);
390 }
391
392 _____unchanged portion omitted_____
393
453 /*****
454 *
455 * FUNCTION:    AcpiNsExternalizeName
456 *
457 * PARAMETERS:  InternalNameLength - Length 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 *****/
469
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;
483
484     ACPI_FUNCTION_TRACE (NsExternalizeName);
485
486     if (!InternalNameLength ||
487         !InternalName ||
488         !ConvertedName)
489     {
490         return ACPI_STATUS (AE_BAD_PARAMETER);
491     }
492
493     /* Check for a prefix (one '\\' | one or more '^') */
494
495     switch (InternalName[0])
496     {
497     case AML_ROOT_PREFIX:
498
499         case '\\':
500             PrefixLength = 1;
501             break;
502
503     case AML_PARENT_PREFIX:
504
505         case '^':
506             for (i = 0; i < InternalNameLength; i++)
507             {
508                 if (ACPI_IS_PARENT_PREFIX (InternalName[i]))
509                 {
510                     if (InternalName[i] == '^')
511                     {
512                         PrefixLength = i + 1;
513                     }
514                     else
515                     {
516                         break;
517                     }
518                 }
519             }
520
521     if (i == InternalNameLength)
522     {
523         PrefixLength = i;
524     }

```

```

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;
622         (*ConvertedName)[j++] = InternalName[NamesIndex++];
623         (*ConvertedName)[j++] = InternalName[NamesIndex++];
624         (*ConvertedName)[j++] = InternalName[NamesIndex++];
625         (*ConvertedName)[j++] = InternalName[NamesIndex++];
626     }
627 }
628 if (ConvertedNameLength)
629 {
630     *ConvertedNameLength = (UINT32) RequiredLength;
631 }
632 return_ACPI_STATUS (AE_OK);
633 }
634 unchanged_portion_omitted
635
636 void
637 AcpiNsTerminate (
638     /* *****
639     * FUNCTION:      AcpiNsTerminate
640     * PARAMETERS:   none
641     * RETURN:       none
642     * DESCRIPTION:  free memory allocated for namespace and ACPI table storage.
643     * *****
644     */
645 )

```



```

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
705     * 1) Free the entire namespace -- all nodes and objects
706     * Delete all object descriptors attached to namespace nodes
707     */
708     AcpiNsDeleteNamespaceSubtree (AcpiGbl_RootNode);

709     /* Delete any objects attached to the root node */
710     /* Detach any objects attached to the root */

711     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
712     if (ACPI_FAILURE (Status))
713     {
714         ObjDesc = AcpiNsGetAttachedObject (AcpiGbl_RootNode);
715         if (ObjDesc)
716         {
717             return_VOID;
718             AcpiNsDetachObject (AcpiGbl_RootNode);
719         }
720     }

721     AcpiNsDeleteNode (AcpiGbl_RootNode);
722     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);

723     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "Namespace freed\n"));
724     return_VOID;

725     /*
726     * FUNCTION:      AcpiNsOpensScope
727     * PARAMETERS:    Type - A valid namespace type
728     * RETURN:        NEWSCOPE if the passed type "opens a name scope" according
729     *                to the ACPI specification, else 0
730     */
731     /*
732     * *****
733     */

734     UINT32
735     AcpiNsOpensScope (
736         ACPI_OBJECT_TYPE      Type)
737 {
738     ACPI_FUNCTION_ENTRY ();
739     ACPI_FUNCTION_TRACE_STR (NsOpensScope, AcpiUtGetTypeName (Type));

740     if (Type > ACPI_TYPE_LOCAL_MAX)
741     {
742         if (!AcpiUtValidObjectType (Type))
743         {
744             /* type code out of range */

745             ACPI_WARNING ((AE_INFO, "Invalid Object Type 0x%X", Type));
746             return (ACPI_NS_NORMAL);
747         }
748         return_UINT32 (ACPI_NS_NORMAL);
749     }

```

```

750     return (((UINT32) AcpiGbl_NsProperties[Type]) & ACPI_NS_NEWSCOPE);
751     return_UINT32 (((UINT32) AcpiGbl_NsProperties[Type]) & ACPI_NS_NEWSCOPE);
752 }

753 /*
754 * FUNCTION:      AcpiNsGetNode
755 * PARAMETERS:    *Pathname - Name to be found, in external (ASL) format. The
756 *                \ (backslash) and ^ (carat) prefixes, and the
757 *                . (period) to separate segments are supported.
758 *                PrefixNode - Root of subtree to be searched, or NS_ALL for the
759 *                root of the name space. If Name is fully
760 *                qualified (first INT8 is '\'), the passed value
761 *                of Scope will not be accessed.
762 *                Flags - Used to indicate whether to perform upsearch or
763 *                not.
764 *                ReturnNode - Where the Node is returned
765 * DESCRIPTION:    Look up a name relative to a given scope and return the
766 *                corresponding Node. NOTE: Scope can be null.
767 * MUTEX:          Locks namespace
768 * *****
769 */

770 ACPI_STATUS
771 AcpiNsGetNode (
772     ACPI_NAMESPACE_NODE *PrefixNode,
773     const char *Pathname,
774     UINT32 Flags,
775     ACPI_NAMESPACE_NODE **ReturnNode)
776 {
777     ACPI_GENERIC_STATE ScopeInfo;
778     ACPI_STATUS Status;
779     char *InternalPath;

780     ACPI_FUNCTION_TRACE_PTR (NsGetNode, ACPI_CAST_PTR (char, Pathname));

781     /* Simplest case is a null pathname */

782     if (!Pathname)
783     {
784         *ReturnNode = PrefixNode;
785         if (!PrefixNode)
786         {
787             *ReturnNode = AcpiGbl_RootNode;
788         }
789         return_ACPI_STATUS (AE_OK);
790     }

791     /* Quick check for a reference to the root */

792     if (ACPI_IS_ROOT_PREFIX (Pathname[0]) && (!Pathname[1]))
793     {
794         *ReturnNode = AcpiGbl_RootNode;
795         return_ACPI_STATUS (AE_OK);
796     }

797     /* Convert path to internal representation */

798     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
```

new/usr/src/common/acpica/components/namespace/nswalk.c

1

```
*****
12574 Fri Jan 17 07:30:28 2014
new/usr/src/common/acpica/components/namespace/nswalk.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: nswalk - Functions for walking the ACPI namespace
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __NSWALK_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acnamesp.h"

52 #define _COMPONENT          ACPI_NAMESPACE
53       ACPI_MODULE_NAME      ("nswalk")

56 /*****
57 *
58 * FUNCTION:      AcpiNsGetNextNode
```

new/usr/src/common/acpica/components/namespace/nswalk.c

2

```
59 *
60 * PARAMETERS:    ParentNode      - Parent node whose children we are
61 *                                     getting
62 *                                     ChildNode    - Previous child that was found.
63 *                                     The NEXT child will be returned
64 *
65 * RETURN:        ACPI_NAMESPACE_NODE - Pointer to the NEXT child or NULL if
66 *                                     none is found.
67 *
68 * DESCRIPTION:    Return the next peer node within the namespace. If Handle
69 *                 is valid, Scope is ignored. Otherwise, the first node
70 *                 within Scope is returned.
71 *
72 *****/

74 ACPI_NAMESPACE_NODE *
75 AcpiNsGetNextNode (
76     ACPI_NAMESPACE_NODE *ParentNode,
77     ACPI_NAMESPACE_NODE *ChildNode)
78 {
79     ACPI_FUNCTION_ENTRY ();

82     if (!ChildNode)
83     {
84         /* It's really the parent's _scope_ that we want */

86         return (ParentNode->Child);
87     }

89     /* Otherwise just return the next peer */

91     return (ChildNode->Peer);
92 }

    unchanged portion omitted

159 /*****
160 *
161 * FUNCTION:      AcpiNsWalkNamespace
162 *
163 * PARAMETERS:    Type              - ACPI_OBJECT_TYPE to search for
164 *                 StartNode        - Handle in namespace where search begins
165 *                 MaxDepth         - Depth to which search is to reach
166 *                 Flags            - Whether to unlock the NS before invoking
167 *                                     the callback routine
168 *                 DescendingCallback - Called during tree descent
169 *                 PreOrderVisit    - Called during tree pre-order visit
170 *                                     when an object of "Type" is found
171 *                 AscendingCallback - Called during tree ascent
172 *                 PostOrderVisit   - Called during tree post-order visit
173 *                                     when an object of "Type" is found
174 *                 Context          - Passed to user function(s) above
175 *                 ReturnValue       - from the UserFunction if terminated
176 *                                     early. Otherwise, returns NULL.
177 *
178 * RETURNS:       Status
179 *
180 * DESCRIPTION:    Performs a modified depth-first walk of the namespace tree,
181 *                 starting (and ending) at the node specified by StartHandle.
182 *                 The callback function is called whenever a node that matches
183 *                 the type parameter is found. If the callback function returns
184 *                 a non-zero value, the search is terminated immediately and
185 *                 this value is returned to the caller.

186 *
187 * The point of this procedure is to provide a generic namespace
188 * walk routine that can be called from multiple places to
```

```

186 *           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,
200     ACPI_WALK_CALLBACK     PreOrderVisit,
201     ACPI_WALK_CALLBACK     PostOrderVisit,
202     void                   *Context,
203     void                   **ReturnValue)
204 {
205     ACPI_STATUS            Status;
206     ACPI_STATUS            MutexStatus;
207     ACPI_NAMESPACE_NODE    *ChildNode;
208     ACPI_NAMESPACE_NODE    *ParentNode;
209     ACPI_OBJECT_TYPE        ChildType;
210     UINT32                 Level;
211     BOOLEAN                NodePreviouslyVisited = FALSE;

212     ACPI_FUNCTION_TRACE (NsWalkNamespace);

213
214     /* Special case for the namespace Root Node */
215
216     if (StartNode == ACPI_ROOT_OBJECT)
217     {
218         StartNode = AcpiGbl_RootNode;
219     }

220
221     /* Null child means "get first node" */
222
223     ParentNode = StartNode;
224     ChildNode = AcpiNsGetNextNode (ParentNode, NULL);
225     ChildType = ACPI_TYPE_ANY;
226     Level = 1;

227
228     /*
229     * Traverse the tree of nodes until we bubble back up to where we
230     * started. When Level is zero, the loop is done because we have
231     * bubbled up to (and passed) the original parent handle (StartEntry)
232     */
233     while (Level > 0 && ChildNode)
234     {
235         Status = AE_OK;

236
237         /* Found next child, get the type if we are not searching for ANY */
238
239         if (Type != ACPI_TYPE_ANY)
240         {
241             ChildType = ChildNode->Type;
242         }

243
244         /*
245         * Ignore all temporary namespace nodes (created during control
246         * method execution) unless told otherwise. These temporary nodes
247         * can cause a race condition because they can be deleted during
248         * 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     }

258
259     /* Type must match requested type */
260
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         }

275
276         /*
277         * Invoke the user function, either descending, ascending,
278         * Invoke the user function, either pre-order or post-order
279         * or both.
280         */
281         if (!NodePreviouslyVisited)
282         {
283             if (DescendingCallback)
284             {
285                 if (PreOrderVisit)
286                 {
287                     Status = DescendingCallback (ChildNode, Level,
288                                                 Status = PreOrderVisit (ChildNode, Level,
289                                                                     Context, ReturnValue);
290                 }
291             }
292             else
293             {
294                 if (AscendingCallback)
295                 {
296                     if (PostOrderVisit)
297                     {
298                         Status = AscendingCallback (ChildNode, Level,
299                                                     Status = PostOrderVisit (ChildNode, Level,
300                                                                     Context, ReturnValue);
301                     }
302                 }
303             }
304         }

305
306         if (Flags & ACPI_NS_WALK_UNLOCK)
307         {
308             MutexStatus = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
309             if (ACPI_FAILURE (MutexStatus))
310             {
311                 return ACPI_STATUS (MutexStatus);
312             }
313         }

314
315         switch (Status)
316         {
317             case AE_OK:
318             case AE_CTRL_DEPTH:

```

new/usr/src/common/acpica/components/namespace/nswalk.c

5

```
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;
```

new/usr/src/common/acpica/components/namespace/nswalk.c

6

```
377         NodePreviouslyVisited = TRUE;
378     }
379 }

381 /* Complete walk, not terminated by user function */

383 return_ACPI_STATUS (AE_OK);
384 }
```

new/usr/src/common/acpica/components/namespace/nsxfeval.c

1

```
*****
32410 Fri Jan 17 07:30:29 2014
new/usr/src/common/acpica/components/namespace/nsxfeval.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: nsxfeval - Public interfaces to the ACPI subsystem
4 *                               ACPI Object evaluation interfaces
5 *
6 *****/

8 /*
9 * Copyright (C) 2000 - 2014, 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 * ("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 MERCHANTABILITY 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 #define __NSXFEVAL_C__
47 #define EXPORT_ACPI_INTERFACES

49 #include "acpi.h"
50 #include "accommon.h"
51 #include "acnamesp.h"
52 #include "acinterp.h"

55 #define _COMPONENT          ACPI_NAMESPACE
56         ACPI_MODULE_NAME    ("nsxfeval")
```

new/usr/src/common/acpica/components/namespace/nsxfeval.c

2

```
58 /* Local prototypes */

60 static void
61 AcpiNsResolveReferences (
62     ACPI_EVALUATE_INFO    *Info);

65 /*****
66 *
67 * FUNCTION:      AcpiEvaluateObjectTyped
68 *
69 * PARAMETERS:    Handle           - Object handle (optional)
70 *                Pathname         - Object pathname (optional)
71 *                ExternalParams    - List of parameters to pass to method,
72 *                                   terminated by NULL. May be NULL
73 *                                   if no parameters are being passed.
74 *                ReturnBuffer      - Where to put method's return value (if
75 *                                   any). If NULL, no value is returned.
76 *                ReturnTypeInfo    - Expected type of return object
77 *
78 * RETURN:        Status
79 *
80 * DESCRIPTION:    Find and evaluate the given object, passing the given
81 *                 parameters if necessary. One of "Handle" or "Pathname" must
82 *                 be valid (non-null)
83 *
84 *****/

86 ACPI_STATUS
87 AcpiEvaluateObjectTyped (
88     ACPI_HANDLE            Handle,
89     ACPI_STRING            Pathname,
90     ACPI_OBJECT_LIST       *ExternalParams,
91     ACPI_BUFFER            *ReturnBuffer,
92     ACPI_OBJECT_TYPE        ReturnTypeInfo)
93 {
94     ACPI_STATUS            Status;
95     BOOLEAN                FreeBufferOnError = FALSE;
96     BOOLEAN                MustFree = FALSE;

98     ACPI_FUNCTION_TRACE (AcpiEvaluateObjectTyped);

101     /* Return buffer must be valid */

103     if (!ReturnBuffer)
104     {
105         return ACPI_STATUS (AE_BAD_PARAMETER);
106     }

108     if (ReturnBuffer->Length == ACPI_ALLOCATE_BUFFER)
109     {
110         FreeBufferOnError = TRUE;
111         MustFree = TRUE;
112     }

113     /* Evaluate the object */

115     Status = AcpiEvaluateObject (Handle, Pathname,
116                                 ExternalParams, ReturnBuffer);
117     Status = AcpiEvaluateObject (Handle, Pathname, ExternalParams, ReturnBuffer)
118     if (ACPI_FAILURE (Status))
119     {
120         return ACPI_STATUS (Status);
121     }
122 }
```

```

120     }

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)
152     {
153         if (MustFree)
154         {
155             /*
156              * Free a buffer created via ACPI_ALLOCATE_BUFFER.
157              * Note: We use AcpiOsFree here because AcpiOsAllocate was used
158              * to allocate the buffer. This purposefully bypasses the
159              * (optionally enabled) allocation tracking mechanism since we
160              * only want to track internal allocations.
161              */
162             /* 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
231          * If there are parameters to be passed to a control method, the external
232          * objects must all be converted to internal objects
233          */
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     */
246     if (!Pathname)
247     {
248         ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
249             "Both Handle and Pathname are NULL"));
250     }
251     else
252     {
253         ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
254             "Null Handle with relative pathname [%s]", Pathname));
255     }
256
257     Status = AE_BAD_PARAMETER;
258     goto Cleanup;
259 }
260
261 Info->RelativePathname = Pathname;
262
263 /*
264  * Convert all external objects passed as arguments to the
265  * internal version(s).
266  */
267 if (ExternalParams && ExternalParams->Count)
268 {
269     Info->ParamCount = (UINT16) ExternalParams->Count;
270
271     /* Warn on impossible argument count */
272
273     if (Info->ParamCount > ACPI_METHOD_NUM_ARGS)
274     {
275         ACPI_WARN_PREDEFINED ((AE_INFO, Pathname, ACPI_WARN_ALWAYS,
276             "Excess arguments (%u) - using only %u",
277             Info->ParamCount, ACPI_METHOD_NUM_ARGS));
278
279         Info->ParamCount = ACPI_METHOD_NUM_ARGS;
280     }
281
282     /*
283      * Allocate a new parameter block for the internal objects
284      * Add 1 to count to allow for null terminated internal list
285      */
286     Info->Parameters = ACPI_ALLOCATE_ZEROED (
287         ((ACPI_SIZE) Info->ParamCount + 1) * sizeof (void *));
288     ((ACPI_SIZE) ExternalParams->Count + 1) * sizeof (void *));
289     if (!Info->Parameters)
290     {
291         Status = AE_NO_MEMORY;
292         goto Cleanup;
293     }
294
295     /* Convert each external object in the list to an internal object */
296
297     for (i = 0; i < Info->ParamCount; i++)
298     for (i = 0; i < ExternalParams->Count; i++)
299     {
300         Status = AcpiUtCopyEobjectToIobject (
301             &ExternalParams->Pointer[i], &Info->Parameters[i]);
302         if (ACPI_FAILURE (Status))
303         {
304             goto Cleanup;
305         }
306     }
307
308     Info->Parameters[Info->ParamCount] = NULL;
309     Info->Parameters[ExternalParams->Count] = NULL;
310 }

```

```

310 #if 0
311
312     /*
313      * Begin incoming argument count analysis. Check for too few args
314      * and too many args.
315      * Three major cases:
316      * 1) Fully qualified pathname
317      * 2) No handle, not fully qualified pathname (error)
318      * 3) Valid handle
319      */
320
321     switch (AcpiNsGetType (Info->Node))
322     if ((Pathname) &&
323         (AcpiNsValidRootPrefix (Pathname[0])))
324     {
325         case ACPI_TYPE_METHOD:
326             /* The path is fully qualified, just evaluate by name */
327
328             /* Check incoming argument count against the method definition */
329
330             if (Info->ObjDesc->Method.ParamCount > Info->ParamCount)
331             {
332                 ACPI_ERROR ((AE_INFO,
333                     "Insufficient arguments (%u) - %u are required",
334                     Info->ParamCount,
335                     Info->ObjDesc->Method.ParamCount));
336
337                 Status = AE_MISSING_ARGUMENTS;
338                 goto Cleanup;
339                 Info->PrefixNode = NULL;
340                 Status = AcpiNsEvaluate (Info);
341             }
342
343             else if (Info->ObjDesc->Method.ParamCount < Info->ParamCount)
344             else if (!Handle)
345             {
346                 ACPI_WARNING ((AE_INFO,
347                     "Excess arguments (%u) - only %u are required",
348                     Info->ParamCount,
349                     Info->ObjDesc->Method.ParamCount));
350
351                 /* Just pass the required number of arguments */
352
353                 Info->ParamCount = Info->ObjDesc->Method.ParamCount;
354             }
355
356             /*
357              * Any incoming external objects to be passed as arguments to the
358              * method must be converted to internal objects
359              * A handle is optional iff a fully qualified pathname is specified.
360              * Since we've already handled fully qualified names above, this is
361              * an error
362              */
363             if (Info->ParamCount)
364             if (!Pathname)
365             {
366                 /*
367                  * Allocate a new parameter block for the internal objects
368                  * Add 1 to count to allow for null terminated internal list
369                  */
370                 Info->Parameters = ACPI_ALLOCATE_ZEROED (
371                     ((ACPI_SIZE) Info->ParamCount + 1) * sizeof (void *));
372                 if (!Info->Parameters)
373                 {
374

```



```

360     Status = AE_NO_MEMORY;
361     goto Cleanup;
362     ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
363         "Both Handle and Pathname are NULL"));
364     }
365
366     /* Convert each external object in the list to an internal object */
367     for (i = 0; i < Info->ParamCount; i++)
368     else
369     {
370         Status = AcpiUtCopyEobjectToIobject (
371             &ExternalParams->Pointer[i], &Info->Parameters[i]);
372         if (ACPI_FAILURE (Status))
373         {
374             goto Cleanup;
375             ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
376                 "Null Handle with relative pathname [%s]", Pathname));
377         }
378     }
379
380     Info->Parameters[Info->ParamCount] = NULL;
381     Status = AE_BAD_PARAMETER;
382     }
383     break;
384
385 default:
386     /* Warn if arguments passed to an object that is not a method */
387     if (Info->ParamCount)
388     else
389     {
390         ACPI_WARNING ((AE_INFO,
391             "%u arguments were passed to a non-method ACPI object",
392             Info->ParamCount));
393     }
394     break;
395 }
396 /* We have a namespace a node and a possible relative path */
397
398 #endif
399
400 /* Now we can evaluate the object */
401
402 Status = AcpiNsEvaluate (Info);
403 }
404
405 /*
406 * If we are expecting a return value, and all went well above,
407 * copy the return value to an external object.
408 */
409 if (ReturnBuffer)
410 {
411     if (!Info->ReturnObject)
412     {
413         ReturnBuffer->Length = 0;
414     }
415     else
416     {
417         if (ACPI_GET_DESCRIPTOR_TYPE (Info->ReturnObject) ==
418             ACPI_DESC_TYPE_NAMED)
419         {
420             /*
421              * 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 }
427
428 if (ACPI_SUCCESS (Status))
429 {
430     /* Dereference Index and RefOf references */
431
432     AcpiNsResolveReferences (Info);
433
434     /* Get the size of the returned object */
435
436     Status = AcpiUtGetObjectSize (Info->ReturnObject,
437         &BufferSpaceNeeded);
438     if (ACPI_SUCCESS (Status))
439     {
440         /* Validate/Allocate/Clear caller buffer */
441
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 */
458
459             Status = AcpiUtCopyIobjectToEobject (Info->ReturnObject,
460                 ReturnBuffer);
461         }
462     }
463 }
464 }
465 }
466
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 ();
474
475     /* Remove one reference on the return object (should delete it) */
476
477     AcpiUtRemoveReference (Info->ReturnObject);
478     AcpiExExitInterpreter ();
479 }
480
481 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
585  *                PreOrderVisit      - Called during tree pre-order visit
586  *                AscendingCallback  - Called during tree ascent
587  *                PostOrderVisit     - Called during tree post-order visit
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,
617     ACPI_WALK_CALLBACK PreOrderVisit,
618     ACPI_WALK_CALLBACK PostOrderVisit,
619     void              *Context,
620     void              **ReturnValue)
621 {
622     ACPI_STATUS      Status;

623     ACPI_FUNCTION_TRACE (AcpiWalkNamespace);

624
625     /* Parameter validation */
626
627     if ((Type > ACPI_TYPE_LOCAL_MAX) ||
628         (!MaxDepth) ||
629         (!DescendingCallback && !AscendingCallback) ||
630         (!PreOrderVisit && !PostOrderVisit))
631     {
632         return_ACPI_STATUS (AE_BAD_PARAMETER);
633     }
634
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);
650     }
651
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     }
663
664     /* Now we can validate the starting node */
665
666     if (!AcpiNsValidateHandle (StartObject))
667     {
668         Status = AE_BAD_PARAMETER;
669         goto UnlockAndExit2;
670     }
671
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);

558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
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      }
762
763      NoMatch = ACPI_STRCMP (Hid->String, Info->Hid);
764      ACPI_FREE (Hid);
765
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          }
781
782          /* Walk the CID list */
783
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 */
790
791                  Found = TRUE;
792                  break;
793              }
794          }
795
796          ACPI_FREE (Cid);
797          if (!Found)
798          {
799              return (AE_OK);
800          }
801      }
802  }

```

```

804  /* Run _STA to determine if device is present */
805
806  Status = AcpiUtExecute_STA (Node, &Flags);
807  if (ACPI_FAILURE (Status))
808  {
809      return (AE_CTRL_DEPTH);
810  }
811
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  }
822
823  /* We have a valid device, invoke the user function */
824
825  Status = Info->UserFunction (ObjHandle, NestingLevel, Info->Context,
826                          ReturnValue);
827  return (Status);
828 }

```

unchanged portion omitted

```

1076 ACPI_EXPORT_SYMBOL (AcpiGetData)

```

new/usr/src/common/acpica/components/namespace/nsxfname.c

1

```
*****
20599 Fri Jan 17 07:30:29 2014
new/usr/src/common/acpica/components/namespace/nsxfname.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: nsxfname - Public interfaces to the ACPI subsystem
4 *                               ACPI Namespace oriented interfaces
5 *
6 *****/
7
8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #define __NSXFNAME_C__
46 #define EXPORT_ACPI_INTERFACES
47
48 #include "acpi.h"
49 #include "accommon.h"
50 #include "acnamesp.h"
51 #include "acparser.h"
52 #include "amlcode.h"
53
54
55 #define _COMPONENT          ACPI_NAMESPACE
56 #define ACPI_MODULE_NAME    ("nsxfname")
```

new/usr/src/common/acpica/components/namespace/nsxfname.c

2

```
58 /* Local prototypes */
59
60 static char *
61 AcpiNsCopyDeviceId (
62     ACPI_PNP_DEVICE_ID *Dest,
63     ACPI_PNP_DEVICE_ID *Source,
64     ACPI_DEVICE_ID *Dest,
65     ACPI_DEVICE_ID *Source,
66     char *StringArea);
67
68 /*****
69 * FUNCTION:      AcpiGetHandle
70 *
71 * PARAMETERS:    Parent          - Object to search under (search scope).
72 *                Pathname        - Pointer to an ascii 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 *****/
84
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;
94
95     ACPI_FUNCTION_ENTRY ();
96
97     /* Parameter Validation */
98
99     if (!RetHandle || !Pathname)
100     {
101         return (AE_BAD_PARAMETER);
102     }
103
104     /* Convert a parent handle to a prefix node */
105
106     if (Parent)
107     {
108         PrefixNode = AcpiNsValidateHandle (Parent);
109         if (!PrefixNode)
110         {
111             return (AE_BAD_PARAMETER);
112         }
113     }
114
115     /*
116     * Valid cases are:
117     * 1) Fully qualified pathname
118     * 2) Parent + Relative pathname
119     */
120
121 }
```

```

122     * Error for <null Parent + relative path>
123     */
124     if (ACPI_IS_ROOT_PREFIX (Pathname[0]))
125     if (AcpiNsValidRootPrefix (Pathname[0]))
126     {
127         /* Pathname is fully qualified (starts with '\') */
128
129         /* Special case for root-only, since we can't search for it */
130
131         if (!ACPI_STRCMP (Pathname, ACPI_NS_ROOT_PATH))
132         {
133             *RetHandle = ACPI_CAST_PTR (ACPI_HANDLE, AcpiGbl_RootNode);
134             return (AE_OK);
135         }
136     }
137     else if (!PrefixNode)
138     {
139         /* Relative path with null prefix is disallowed */
140
141         return (AE_BAD_PARAMETER);
142     }
143
144     /* Find the Node and convert to a handle */
145
146     Status = AcpiNsGetNode (PrefixNode, Pathname, ACPI_NS_NO_UPSEARCH, &Node);
147     if (ACPI_SUCCESS (Status))
148     {
149         *RetHandle = ACPI_CAST_PTR (ACPI_HANDLE, Node);
150     }
151
152     return (Status);
153 }
154
155 ACPI_EXPORT_SYMBOL (AcpiGetHandle)
156
157 /*****
158  * FUNCTION:      AcpiGetName
159  *
160  * PARAMETERS:    Handle          - Handle to be converted to a pathname
161  *                NameType        - Full pathname or single segment
162  *                Buffer            - Buffer for returned path
163  *
164  * RETURN:        Pointer to a string containing the fully qualified Name.
165  *
166  * DESCRIPTION:    This routine returns the fully qualified name associated with
167  *                the Handle parameter. This and the AcpiPathnameToHandle are
168  *                complementary functions.
169  *
170  * *****/
171
172
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;
182
183     /* Parameter validation */
184
185     if (NameType > ACPI_NAME_TYPE_MAX)

```

```

187     {
188         return (AE_BAD_PARAMETER);
189     }
190
191     Status = AcpiUtValidateBuffer (Buffer);
192     if (ACPI_FAILURE (Status))
193     {
194         return (Status);
195     }
196
197     if (NameType == ACPI_FULL_PATHNAME)
198     {
199         /* Get the full pathname (From the namespace root) */
200
201         Status = AcpiNsHandleToPathname (Handle, Buffer);
202         return (Status);
203     }
204
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     }
214
215     Node = AcpiNsValidateHandle (Handle);
216     if (!Node)
217     {
218         Status = AE_BAD_PARAMETER;
219         goto UnlockAndExit;
220     }
221
222     /* Validate/Allocate/Clear caller buffer */
223
224     Status = AcpiUtInitializeBuffer (Buffer, ACPI_PATH_SEGMENT_LENGTH);
225     if (ACPI_FAILURE (Status))
226     {
227         goto UnlockAndExit;
228     }
229
230     /* Just copy the ACPI name from the Node and zero terminate it */
231
232     NodeName = AcpiUtGetNodeName (Node);
233     ACPI_MOVE_NAME (Buffer->Pointer, NodeName);
234     ACPI_STRNCPY (Buffer->Pointer, AcpiUtGetNodeName (Node),
235                 ACPI_NAME_SIZE);
236     ((char *) Buffer->Pointer) [ACPI_NAME_SIZE] = 0;
237     Status = AE_OK;
238
239     UnlockAndExit:
240
241     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
242     return (Status);
243 }
244
245 ACPI_EXPORT_SYMBOL (AcpiGetName)
246
247 /*****
248  *
249  * FUNCTION:      AcpiNsCopyDeviceId
250  *

```

```

251 * PARAMETERS: Dest      - Pointer to the destination PNP_DEVICE_ID
252 *              Source    - Pointer to the source PNP_DEVICE_ID
253 *              Dest      - Pointer to the destination DEVICE_ID
254 *              Source    - Pointer to the source DEVICE_ID
255 *              StringArea - Pointer to where to copy the dest string
256 * RETURN:      Pointer to the next string area
257 * DESCRIPTION: Copy a single PNP_DEVICE_ID, including the string data.
258 * DESCRIPTION: Copy a single DEVICE_ID, including the string data.
259 *****/

261 static char *
262 AcpiNsCopyDeviceId (
263     ACPI_PNP_DEVICE_ID *Dest,
264     ACPI_PNP_DEVICE_ID *Source,
265     ACPI_DEVICE_ID *Dest,
266     ACPI_DEVICE_ID *Source,
267     char *StringArea)
268 {
269     /* Create the destination DEVICE_ID */

270     /* Create the destination PNP_DEVICE_ID */

271     Dest->String = StringArea;
272     Dest->Length = Source->Length;

273     /* Copy actual string and return a pointer to the next string area */

274     ACPI_MEMCPY (StringArea, Source->String, Source->Length);
275     return (StringArea + Source->Length);
276 }

280 /*****
281 *
282 * FUNCTION:      AcpiGetObjectInfo
283 *
284 * PARAMETERS:    Handle      - Object Handle
285 *                ReturnBuffer - Where the info is returned
286 *
287 * RETURN:        Status
288 *
289 * DESCRIPTION:    Returns information about an object as gleaned from the
290 *                namespace node and possibly by running several standard
291 *                control methods (Such as in the case of a device.)
292 *
293 * For Device and Processor objects, run the Device _HID, _UID, _CID, _SUB,
294 * _STA, _ADR, _SxW, and _SxD methods.
295 * For Device and Processor objects, run the Device _HID, _UID, _CID, _STA,
296 * _ADR, _SxW, and _SxD methods.
297 *
298 * Note: Allocates the return buffer, must be freed by the caller.
299 *****/

300 ACPI_STATUS
301 AcpiGetObjectInfo (
302     ACPI_HANDLE Handle,
303     ACPI_DEVICE_INFO **ReturnBuffer)
304 {
305     ACPI_NAMESPACE_NODE *Node;
306     ACPI_DEVICE_INFO *Info;
307     ACPI_PNP_DEVICE_ID_LIST *CidList = NULL;
308     ACPI_PNP_DEVICE_ID *Hid = NULL;

```

```

309     ACPI_PNP_DEVICE_ID *Uid = NULL;
310     ACPI_PNP_DEVICE_ID *Sub = NULL;
311     ACPI_DEVICE_ID_LIST *CidList = NULL;
312     ACPI_DEVICE_ID *Hid = NULL;
313     ACPI_DEVICE_ID *Uid = NULL;
314     char *NextIdString;
315     ACPI_OBJECT_TYPE Type;
316     ACPI_NAME Name;
317     UINT8 ParamCount = 0;
318     UINT8 Valid = 0;
319     UINT32 InfoSize;
320     i;
321     ACPI_STATUS Status;

322     /* Parameter validation */

323     if (!Handle || !ReturnBuffer)
324     {
325         return (AE_BAD_PARAMETER);
326     }

327     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
328     if (ACPI_FAILURE (Status))
329     {
330         return (Status);
331     }
332     goto Cleanup;

333     Node = AcpiNsValidateHandle (Handle);
334     if (!Node)
335     {
336         (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
337         return (AE_BAD_PARAMETER);
338     }

339     /* Get the namespace node data while the namespace is locked */

340     InfoSize = sizeof (ACPI_DEVICE_INFO);
341     Type = Node->Type;
342     Name = Node->Name.Integer;

343     if (Node->Type == ACPI_TYPE_METHOD)
344     {
345         ParamCount = Node->Object->Method.ParamCount;
346     }

347     Status = AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
348     if (ACPI_FAILURE (Status))
349     {
350         return (Status);
351     }

352     if ((Type == ACPI_TYPE_DEVICE) ||
353         (Type == ACPI_TYPE_PROCESSOR))
354     {
355         /*
356          * Get extra info for ACPI Device/Processor objects only:
357          * Run the Device _HID, _UID, _SUB, and _CID methods.
358          * Run the Device _HID, _UID, and _CID methods.
359          *
360          * Note: none of these methods are required, so they may or may
361          * not be present for this device. The Info->Valid bitfield is used
362          * to indicate which methods were found and run successfully.
363          */

```

```

370      /* Execute the Device._HID method */
371
372      Status = AcpiUtExecute_HID (Node, &Hid);
373      if (ACPI_SUCCESS (Status))
374      {
375          InfoSize += Hid->Length;
376          Valid |= ACPI_VALID_HID;
377      }
378
379      /* Execute the Device._UID method */
380
381      Status = AcpiUtExecute_UID (Node, &Uid);
382      if (ACPI_SUCCESS (Status))
383      {
384          InfoSize += Uid->Length;
385          Valid |= ACPI_VALID_UID;
386      }
387
388      /* Execute the Device._SUB method */
389
390      Status = AcpiUtExecute_SUB (Node, &Sub);
391      if (ACPI_SUCCESS (Status))
392      {
393          InfoSize += Sub->Length;
394          Valid |= ACPI_VALID_SUB;
395      }
396
397      /* Execute the Device._CID method */
398
399      Status = AcpiUtExecute_CID (Node, &CidList);
400      if (ACPI_SUCCESS (Status))
401      {
402          /* Add size of CID strings and CID pointer array */
403
404          InfoSize += (CidList->ListSize - sizeof (ACPI_PNP_DEVICE_ID_LIST));
405          InfoSize += (CidList->ListSize - sizeof (ACPI_DEVICE_ID_LIST));
406          Valid |= ACPI_VALID_CID;
407      }
408
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      }
419
420      /* Get the fixed-length data */
421
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
430           * Note: none of these methods are required, so they may or may
431           * not be present for this device. The Info->Valid bitfield is used
432           * to indicate which methods were found and run successfully.
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      */
438
439      /* Execute the Device._STA method */
440
441      Status = AcpiUtExecute_STA (Node, &Info->CurrentStatus);
442      if (ACPI_SUCCESS (Status))
443      {
444          Valid |= ACPI_VALID_STA;
445      }
446
447      /* Execute the Device._ADR method */
448
449      Status = AcpiUtEvaluateNumericObject (METHOD_NAME__ADR, Node,
450          &Info->Address);
451      if (ACPI_SUCCESS (Status))
452      {
453          Valid |= ACPI_VALID_ADR;
454      }
455
456      /* Execute the Device._SxW methods */
457
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      }
465
466      /* Execute the Device._SxD methods */
467
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
476      /*
477       * Create a pointer to the string area of the return buffer.
478       * Point to the end of the base ACPI_DEVICE_INFO structure.
479       */
480      NextIdString = ACPI_CAST_PTR (char, Info->CompatibleIdList.Ids);
481      if (CidList)
482      {
483          /* Point past the CID PNP_DEVICE_ID array */
484          /* Point past the CID DEVICE_ID array */
485
486          NextIdString += ((ACPI_SIZE) CidList->Count * sizeof (ACPI_PNP_DEVICE_ID
487          NextIdString += ((ACPI_SIZE) CidList->Count * sizeof (ACPI_DEVICE_ID));
488      }
489
490      /*
491       * Copy the HID, UID, SUB, and CIDs to the return buffer.
492       * The variable-length strings are copied to the reserved area
493       * at the end of the buffer.
494       * Copy the HID, UID, and CIDs to the return buffer. The variable-length
495       * strings are copied to the reserved area at the end of the buffer.
496       *
497       * For HID and CID, check if the ID is a PCI Root Bridge.
498       */

```



```

496     if (Hid)
497     {
498         NextIdString = AcpiNsCopyDeviceId (&Info->HardwareId,
499             Hid, NextIdString);
501
502         if (AcpiUtIsPciRootBridge (Hid->String))
503         {
504             Info->Flags |= ACPI_PCI_ROOT_BRIDGE;
505         }
507
508         if (Uid)
509         {
510             NextIdString = AcpiNsCopyDeviceId (&Info->UniqueId,
511                 Uid, NextIdString);
512         }
513
514         if (Sub)
515         {
516             NextIdString = AcpiNsCopyDeviceId (&Info->SubsystemId,
517                 Sub, NextIdString);
518         }
519
520         if (CidList)
521         {
522             Info->CompatibleIdList.Count = CidList->Count;
523             Info->CompatibleIdList.ListSize = CidList->ListSize;
524
525             /* Copy each CID */
526
527             for (i = 0; i < CidList->Count; i++)
528             {
529                 NextIdString = AcpiNsCopyDeviceId (&Info->CompatibleIdList.Ids[i],
530                     &CidList->Ids[i], NextIdString);
531
532                 if (AcpiUtIsPciRootBridge (CidList->Ids[i].String))
533                 {
534                     Info->Flags |= ACPI_PCI_ROOT_BRIDGE;
535                 }
536             }
537
538             /* Copy the fixed-length data */
539
540             Info->InfoSize = InfoSize;
541             Info->Type = Type;
542             Info->Name = Name;
543             Info->ParamCount = ParamCount;
544             Info->Valid = Valid;
545
546             *ReturnBuffer = Info;
547             Status = AE_OK;
548
549 Cleanup:
550         if (Hid)
551         {
552             ACPI_FREE (Hid);
553         }
554         if (Uid)
555         {
556             ACPI_FREE (Uid);
557         }
558         if (Sub)
559         {
560             ACPI_FREE (Sub);
561         }

```

```

562     }
563     if (CidList)
564     {
565         ACPI_FREE (CidList);
566     }
567     return (Status);
568 }

```

unchanged\_portion\_omitted

new/usr/src/common/acpica/components/namespace/nsxfobj.c

1

```
*****
7714 Fri Jan 17 07:30:30 2014
new/usr/src/common/acpica/components/namespace/nsxfobj.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: nsxfobj - Public interfaces to the ACPI subsystem
4 *                   ACPI Object oriented interfaces
5 *
6 *****/
7
8 /*
9 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
45
46 #define __NSXFOBJ_C__
47 #define EXPORT_ACPI_INTERFACES
48
49 #include "acpi.h"
50 #include "accommon.h"
51 #include "acnamesp.h"
52
53 #define _COMPONENT          ACPI_NAMESPACE
54 #define ACPI_MODULE_NAME    ("nsxfobj")
55
56 /*****
```

new/usr/src/common/acpica/components/namespace/nsxfobj.c

2

```
58 *
59 * FUNCTION:      AcpiGetType
60 *
61 * PARAMETERS:    Handle          - Handle of object whose type is desired
62 *               RetType         - Where the type will be placed
63 *
64 * RETURN:        Status
65 *
66 * DESCRIPTION:   This routine returns the type associatd with a particular handle
67 *
68 *****/
69
70 ACPI_STATUS
71 AcpiGetType (
72     ACPI_HANDLE          Handle,
73     ACPI_OBJECT_TYPE     *RetType)
74 {
75     ACPI_NAMESPACE_NODE *Node;
76     ACPI_STATUS          Status;
77
78     /* Parameter Validation */
79
80     if (!RetType)
81     {
82         return (AE_BAD_PARAMETER);
83     }
84
85     /*
86      * Special case for the predefined Root Node
87      * (return type ANY)
88      */
89     if (Handle == ACPI_ROOT_OBJECT)
90     {
91         *RetType = ACPI_TYPE_ANY;
92         return (AE_OK);
93     }
94
95     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
96     if (ACPI_FAILURE (Status))
97     {
98         return (Status);
99     }
100
101     /* Convert and validate the handle */
102
103     Node = AcpiNsValidateHandle (Handle);
104     if (!Node)
105     {
106         (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
107         return (AE_BAD_PARAMETER);
108     }
109
110     *RetType = Node->Type;
111
112     Status = AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
113     return (Status);
114 }
115
116 unchanged_portion_omitted
117
118 285 ACPI_EXPORT_SYMBOL (AcpiGetNextObject)
```

new/usr/src/common/acpica/components/parser/psargs.c

1

```
*****
25895 Fri Jan 17 07:30:30 2014
new/usr/src/common/acpica/components/parser/psargs.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: psargs - Parse AML opcode arguments
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __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
```

new/usr/src/common/acpica/components/parser/psargs.c

2

```
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     PackageLength = 0;
87     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 *****/
171
172 char *
173 AcpiPsGetNextNamestring (
174     ACPI_PARSE_STATE      *ParserState)
175 {
176     UINT8                *Start = ParserState->Aml;
177     UINT8                *End = ParserState->Aml;
178
179     ACPI_FUNCTION_TRACE (PsGetNextNamestring);
180
181     /* Point past any namestring prefix characters (backslash or carat) */
182
183     while (ACPI_IS_ROOT_PREFIX (*End) ||
184            ACPI_IS_PARENT_PREFIX (*End))
185     while (AcpiPsIsPrefixChar (*End))
186     {
187         End++;
188     }
189
190     /* Decode the path prefix character */
191
192     switch (*End)
193     {
194     case 0:
195
196         /* NullName */
197
198         if (End == Start)
199         {
200             Start = NULL;
201         }
202         End++;
203         break;
204
205     case AML_DUAL_NAME_PREFIX:
206
207         /* Two name segments */
208
209         End += 1 + (2 * ACPI_NAME_SIZE);
210         break;
211
212     case AML_MULTI_NAME_PREFIX_OP:
213
214         /* Multiple name segments, 4 chars each, count in next byte */
215
216         End += 2 + (*(End + 1) * ACPI_NAME_SIZE);
217         break;
218
219     default:

```

```

222     /* Single name segment */
223
224     End += ACPI_NAME_SIZE;
225     break;
226 }
227
228 ParserState->Aml = End;
229 return_PTR ((char *) Start);
230 }
231
232 unchanged_portion_omitted
233
234 /*****
235 *
236 * FUNCTION:      AcpiPsGetNextSimpleArg
237 *
238 * PARAMETERS:    ParserState          - Current parser state object
239 *                ArgType              - The argument type (AML_*_ARG)
240 *                Arg                  - Where the argument is returned
241 *
242 * RETURN:        None
243 *
244 * DESCRIPTION:   Get the next simple argument (constant, string, or namestring)
245 *
246 *****/
247
248 void
249 AcpiPsGetNextSimpleArg (
250     ACPI_PARSE_STATE      *ParserState,
251     UINT32                ArgType,
252     ACPI_PARSE_OBJECT     *Arg)
253 {
254     {
255         UINT32            Length;
256         UINT16            Opcode;
257         UINT8             *Aml = ParserState->Aml;
258
259         ACPI_FUNCTION_TRACE_U32 (PsGetNextSimpleArg, ArgType);
260
261         switch (ArgType)
262         {
263         case ARGP_BYTEDATA:
264
265             /* Get 1 byte from the AML stream */
266
267             Opcode = AML_BYTE_OP;
268             Arg->Common.Value.Integer = (UINT64) *Aml;
269             Length = 1;
270             break;
271
272         case ARGP_WORDDATA:
273
274             /* Get 2 bytes from the AML stream */
275
276             Opcode = AML_WORD_OP;
277             ACPI_MOVE_16_TO_64 (&Arg->Common.Value.Integer, Aml);
278             Length = 2;
279             break;
280
281         case ARGP_DWORDDATA:
282
283             /* 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;
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:
553     default:

554         Opcode = AML_INT_RESERVEDFIELD_OP;
555         ParserState->Aml++;
556         Opcode = AML_INT_NAMEDFIELD_OP;
557         break;

558     case AML_FIELD_ACCESS_OP:
559     case 0x00:

560         Opcode = AML_INT_ACCESSFIELD_OP;
561         Opcode = AML_INT_RESERVEDFIELD_OP;
562         ParserState->Aml++;
563         break;

564     case AML_FIELD_CONNECTION_OP:
565     case 0x01:

566         Opcode = AML_INT_CONNECTION_OP;
567         Opcode = AML_INT_ACCESSFIELD_OP;
568         ParserState->Aml++;
569         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 */
583
584  Field = AcpiPsAllocOp (Opcode);
585  if (!Field)
586  {
587      return_PTR (NULL);
588  }
589
590  Field->Common.AmlOffset = AmlOffset;
591
592  /* Decode the field type */
593
594  switch (Opcode)
595  {
596  case AML_INT_NAMEDFIELD_OP:
597
598      /* Get the 4-character name */
599
600      ACPI_MOVE_32_TO_32 (&Name, ParserState->Aml);
601      AcpiPsSetName (Field, Name);
602      ParserState->Aml += ACPI_NAME_SIZE;
603
604      /* Get the length which is encoded as a package length */
605
606      Field->Common.Value.Size = AcpiPsGetNextPackageLength (ParserState);
607      break;
608
609  case AML_INT_RESERVEDFIELD_OP:
610
611      /* Get the length which is encoded as a package length */
612
613      Field->Common.Value.Size = AcpiPsGetNextPackageLength (ParserState);
614      break;
615
616  case AML_INT_ACCESSFIELD_OP:
617  case AML_INT_EXTACCESSFIELD_OP:
618
619      /*
620       * Get AccessType and AccessAttrib and merge into the field Op
621       * AccessType is first operand, AccessAttribute is second. stuff
622       * these bytes into the node integer value for convenience.
623       * AccessType is first operand, AccessAttribute is second
624       */
625
626      /* Get the two bytes (Type/Attribute) */
627
628      AccessType = ACPI_GET8 (ParserState->Aml);
629      Field->Common.Value.Integer = (((UINT32) ACPI_GET8 (ParserState->Aml) <<
630      ParserState->Aml++;
631      AccessAttribute = ACPI_GET8 (ParserState->Aml);
632      Field->Common.Value.Integer |= ACPI_GET8 (ParserState->Aml);
633      ParserState->Aml++;
634
635      Field->Common.Value.Integer = (UINT8) AccessType;
636      Field->Common.Value.Integer |= (UINT16) (AccessAttribute << 8);
637
638      /* This opcode has a third byte, AccessLength */
639
640      if (Opcode == AML_INT_EXTACCESSFIELD_OP)
641      {
642          AccessLength = ACPI_GET8 (ParserState->Aml);
643          ParserState->Aml++;

```

```

644      Field->Common.Value.Integer |= (UINT32) (AccessLength << 16);
645  }
646  break;
647
648
649  case AML_INT_CONNECTION_OP:
650
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++;
658
659          PkgEnd = ParserState->Aml;
660          PkgLength = AcpiPsGetNextPackageLength (ParserState);
661          PkgEnd += PkgLength;
662
663          if (ParserState->Aml < PkgEnd)
664          {
665              /* Non-empty list */
666
667              Arg = AcpiPsAllocOp (AML_INT_BYTELIST_OP);
668              if (!Arg)
669              {
670                  AcpiPsFreeOp (Field);
671                  return_PTR (NULL);
672              }
673
674              /* Get the actual buffer length argument */
675
676              Opcode = ACPI_GET8 (ParserState->Aml);
677              ParserState->Aml++;
678
679              switch (Opcode)
680              {
681              case AML_BYTE_OP:          /* AML_BYTEDATA_ARG */
682
683                  BufferLength = ACPI_GET8 (ParserState->Aml);
684                  ParserState->Aml += 1;
685                  break;
686
687              case AML_WORD_OP:         /* AML_WORDDATA_ARG */
688
689                  BufferLength = ACPI_GET16 (ParserState->Aml);
690                  ParserState->Aml += 2;
691                  break;
692
693              case AML_DWORD_OP:        /* AML_DWORDDATA_ARG */
694
695                  BufferLength = ACPI_GET32 (ParserState->Aml);
696                  ParserState->Aml += 4;
697                  break;
698
699              default:
700
701                  BufferLength = 0;
702                  break;
703              }
704
705              /* Fill in bytelist data */
706
707              Arg->Named.Value.Size = BufferLength;
708              Arg->Named.Data = ParserState->Aml;
709          }

```

```

711      /* Skip to End of byte data */
712
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      }
723
724      /* Get the Namestring argument */
725
726      Arg->Common.Value.Name = AcpiPsGetNextNamestring (ParserState);
727  }
728
729  /* Link the buffer/namestring to parent (CONNECTION_OP) */
730
731  AcpiPsAppendArg (Field, Arg);
732  break;
733
734 default:
735
736      /* Opcode was set in previous switch */
737      break;
738  }
739
740  return_PTR (Field);
741 }
742
743
744 /*****
745  *
746  * FUNCTION:      AcpiPsGetNextArg
747  *
748  * PARAMETERS:    WalkState          - Current state
749  *                ParserState        - Current parser state object
750  *                ArgType            - The argument type (AML*_ARG)
751  *                ReturnArg          - Where the next arg is returned
752  *
753  * RETURN:        Status, and an op object containing the next argument.
754  *
755  * DESCRIPTION:   Get next argument (including complex list arguments that require
756  *                pushing the parser stack)
757  *
758  * *****/
759
760 ACPI_STATUS
761 AcpiPsGetNextArg (
762     ACPI_WALK_STATE      *WalkState,
763     ACPI_PARSE_STATE     *ParserState,
764     UINT32                ArgType,
765     ACPI_PARSE_OBJECT    **ReturnArg)
766 {
767     ACPI_PARSE_OBJECT     *Arg = NULL;
768     ACPI_PARSE_OBJECT     *Prev = NULL;
769     ACPI_PARSE_OBJECT     *Field;
770     UINT32                Subop;
771     ACPI_STATUS            Status = AE_OK;
772
773     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:
786
787         /* Constants, strings, and namestrings are all the same size */
788
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;
796
797     case ARGP_PKGLLENGTH:
798
799         /* Package length, nothing returned */
800
801         ParserState->PkgEnd = AcpiPsGetNextPackageEnd (ParserState);
802         break;
803
804     case ARGP_FIELDLIST:
805
806         if (ParserState->Aml < ParserState->PkgEnd)
807         {
808             /* Non-empty list */
809
810             while (ParserState->Aml < ParserState->PkgEnd)
811             {
812                 Field = AcpiPsGetNextField (ParserState);
813                 if (!Field)
814                 {
815                     return ACPI_STATUS (AE_NO_MEMORY);
816                 }
817
818                 if (Prev)
819                 {
820                     Prev->Common.Next = Field;
821                 }
822                 else
823                 {
824                     Arg = Field;
825                 }
826                 Prev = Field;
827             }
828
829             /* Skip to End of byte data */
830
831             ParserState->Aml = ParserState->PkgEnd;
832         }
833         break;
834
835     case ARGP_BYTELIST:
836
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))
868          AcpiPsIsPrefixChar (Subop)
869      {
870          /* NullName or NameString */
872          Arg = AcpiPsAllocOp (AML_INT_NAMEPATH_OP);
873          if (!Arg)
874          {
875              return_ACPI_STATUS (AE_NO_MEMORY);
876          }
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



new/usr/src/common/acpica/components/parser/psloop.c

1

```
*****
20478 Fri Jan 17 07:30:31 2014
new/usr/src/common/acpica/components/parser/psloop.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: psloop - Main AML parse loop
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 /*
46 * Parse the AML and build an operation tree as most interpreters, (such as
47 * Perl) do. Parsing is done by hand rather than with a YACC generated parser
48 * to tightly constrain stack and dynamic memory usage. Parsing is kept
49 * flexible and the code fairly compact by parsing based on a list of AML
50 * opcode templates in AmlOpInfo[].
51 */

53 #include "acpi.h"
54 #include "accommon.h"
55 #include "acparser.h"
56 #include "acdispat.h"
57 #include "amlcode.h"

59 #define _COMPONENT                ACPI_PARSER
```

new/usr/src/common/acpica/components/parser/psloop.c

2

```
60         ACPI_MODULE_NAME        ("psloop")

63 /* Local prototypes */

65 static ACPI_STATUS
66 AcpiPsGetArguments (
67     ACPI_WALK_STATE      *WalkState,
68     UINT8                *AmlOpStart,
69     ACPI_PARSE_OBJECT     *Op);

71 static void
72 AcpiPsLinkModuleCode (
73     ACPI_PARSE_OBJECT     *ParentOp,
74     UINT8                *AmlStart,
75     UINT32               AmlLength,
76     ACPI_OWNER_ID        OwnerId);

79 /*****
80 *
81 * FUNCTION:      AcpiPsGetArguments
82 *
83 * PARAMETERS:    WalkState      - Current state
84 *                AmlOpStart     - Op start in AML
85 *                Op             - Current Op
86 *
87 * RETURN:        Status
88 *
89 * DESCRIPTION:   Get arguments for passed Op.
90 *
91 *****/

93 static ACPI_STATUS
94 AcpiPsGetArguments (
95     ACPI_WALK_STATE      *WalkState,
96     UINT8                *AmlOpStart,
97     ACPI_PARSE_OBJECT     *Op)
98 {
99     ACPI_STATUS           Status = AE_OK;
100     ACPI_PARSE_OBJECT     *Arg = NULL;
101     const ACPI_OPCODE_INFO *OpInfo;

104     ACPI_FUNCTION_TRACE_PTR (PsGetArguments, WalkState);

107     switch (Op->Common.AmlOpcode)
108     {
109     case AML_BYTE_OP:      /* AML_BYTEDATA_ARG */
110     case AML_WORD_OP:      /* AML_WORDDATA_ARG */
111     case AML_DWORD_OP:     /* AML_DWORDDATA_ARG */
112     case AML_QWORD_OP:     /* AML_QWORDDATA_ARG */
113     case AML_STRING_OP:    /* AML_ASCIICHARLIST_ARG */

115         /* Fill in constant or string argument directly */

117         AcpiPsGetNextSimpleArg (&(WalkState->ParserState),
118             GET_CURRENT_ARG_TYPE (WalkState->ArgTypes), Op);
119         break;

121     case AML_INT_NAMEPATH_OP: /* AML_NAMESTRING_ARG */

123         Status = AcpiPsGetNextNamepath (WalkState, &(WalkState->ParserState), Op
124         if (ACPI_FAILURE (Status))
125             {
```

```

126     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->ArgCount)
137 {
138     WalkState->AmlOffset = (UINT32) ACPI_PTR_DIFF (WalkState->ParserState,
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 *
391 * DESCRIPTION: Parse AML (pointed to by the current parser state) and return
392 *             a tree of ops.
393 *
394 *****/
396 ACPI_STATUS
397 AcpiPsParseLoop (
398     ACPI_WALK_STATE      *WalkState)
399 {
400     ACPI_STATUS      Status = AE_OK;
401     ACPI_PARSE_OBJECT *Op = NULL; /* current op */
402     ACPI_PARSE_STATE *ParserState;
403     UINT8
404     AmlOpStart = NULL;

406     ACPI_FUNCTION_TRACE_PTR (PsParseLoop, WalkState);

409     if (WalkState->DescendingCallback == NULL)
410     {
411         return ACPI_STATUS (AE_BAD_PARAMETER);
412     }

414     ParserState = &WalkState->ParserState;
415     WalkState->ArgTypes = 0;

417 #if (!defined (ACPI_NO_METHOD_EXECUTION) && !defined (ACPI_CONSTANT_EVAL_ONLY))

419     if (WalkState->WalkType & ACPI_WALK_METHOD_RESTART)
420     {
421         /* We are restarting a preempted control method */

423         if (AcpiPsHasCompletedScope (ParserState))
424         {
425             /*
426              * We must check if a predicate to an IF or WHILE statement
427              * was just completed
428              */
429             if ((ParserState->Scope->ParseScope.Op) &&
430                 ((ParserState->Scope->ParseScope.Op->Common.AmlOpcode == AML_IF_O
431                  (ParserState->Scope->ParseScope.Op->Common.AmlOpcode == AML_WHIL
432                  (WalkState->ControlState) &&
433                  (WalkState->ControlState->Common.State ==
434                   ACPI_CONTROL_PREDICATE_EXECUTING)))
435             {
436                 /*
437                  * A predicate was just completed, get the value of the
438                  * predicate and branch based on that value
439                  */
440                 WalkState->Op = NULL;
441                 Status = AcpiPsGetPredicateValue (WalkState, ACPI_TO_POINTER (TR
442                 if (ACPI_FAILURE (Status) &&
443                     ((Status & AE_CODE_MASK) != AE_CODE_CONTROL))
444                 {
445                     if (Status == AE_AML_NO_RETURN_VALUE)
446                     {
447                         ACPI_EXCEPTION ((AE_INFO, Status,
448                          "Invoked method did not return a value"));
449                     }

451                     ACPI_EXCEPTION ((AE_INFO, Status, "GetPredicate Failed"));
452                     return ACPI_STATUS (Status);
453                 }

455     Status = AcpiPsNextParseState (WalkState, Op, Status);

```

```

456     }

458     AcpiPsPopScope (ParserState, &Op,
459                     &WalkState->ArgTypes, &WalkState->ArgCount);
460     ACPI_DEBUG_PRINT ((ACPI_DB_PARSE, "Popped scope, Op=%p\n", Op));
461 }
462 else if (WalkState->PrevOp)
463 {
464     /* We were in the middle of an op */

466     Op = WalkState->PrevOp;
467     WalkState->ArgTypes = WalkState->PrevArgTypes;
468 }
469 }
470 #endif

472 /* Iterative parsing loop, while there is more AML to process: */

474 while ((ParserState->Aml < ParserState->AmlEnd) || (Op))
475 {
476     AmlOpStart = ParserState->Aml;
477     if (!Op)
478     {
479         Status = AcpiPsCreateOp (WalkState, AmlOpStart, &Op);
480         if (ACPI_FAILURE (Status))
481         {
482             if (Status == AE_CTRL_PARSE_CONTINUE)
483             {
484                 continue;
485             }

487             if (Status == AE_CTRL_PARSE_PENDING)
488             {
489                 Status = AE_OK;
490             }

492             Status = AcpiPsCompleteOp (WalkState, &Op, Status);
493             if (ACPI_FAILURE (Status))
494             {
495                 return ACPI_STATUS (Status);
496             }

498             continue;
499         }

501         Op->Common.AmlOffset = WalkState->AmlOffset;

503         if (WalkState->OpInfo)
504         {
505             ACPI_DEBUG_PRINT ((ACPI_DB_PARSE,
506              "Opcode %4.4X [%s] Op %p Aml %p AmlOffset %5.5X\n",
507              (UINT32) Op->Common.AmlOpcode, WalkState->OpInfo->Name,
508              Op, ParserState->Aml, Op->Common.AmlOffset));
509         }
510     }

513     /*
514     * Start ArgCount at zero because we don't know if there are
515     * any args yet
516     */
517     WalkState->ArgCount = 0;

519     /* Are there any arguments that must be processed? */

521     if (WalkState->ArgTypes)

```

```

522     {
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 }

```

new/usr/src/common/acpica/components/parser/psobject.c

1

```
*****
19963 Fri Jan 17 07:30:31 2014
new/usr/src/common/acpica/components/parser/psobject.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: psobject - Support for parse objects
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"
48 #include "amlcode.h"

50 #define _COMPONENT          ACPI_PARSER
51 #define ACPI_MODULE_NAME    ("psobject")

54 /* Local prototypes */

56 static ACPI_STATUS
57 AcpiPsGetAmlOpcode (
58     ACPI_WALK_STATE          *WalkState);
```

new/usr/src/common/acpica/components/parser/psobject.c

2

```
61 /*****
62 *
63 * FUNCTION:      AcpiPsGetAmlOpcode
64 *
65 * PARAMETERS:    WalkState          - Current state
66 *
67 * RETURN:        Status
68 *
69 * DESCRIPTION:    Extract the next AML opcode from the input stream.
70 *
71 *****/

73 static ACPI_STATUS
74 AcpiPsGetAmlOpcode (
75     ACPI_WALK_STATE          *WalkState)
76 {
78     ACPI_FUNCTION_TRACE_PTR (PsGetAmlOpcode, WalkState);

81     WalkState->AmlOffset = (UINT32) ACPI_PTR_DIFF (WalkState->ParserState.Aml,
82                                                     WalkState->ParserState.AmlStart);
83     WalkState->Opcode = AcpiPsPeekOpcode (&(WalkState->ParserState));

85     /*
86      * First cut to determine what we have found:
87      * 1) A valid AML opcode
88      * 2) A name string
89      * 3) An unknown/invalid opcode
90      */
91     WalkState->OpInfo = AcpiPsGetOpcodeInfo (WalkState->Opcode);

93     switch (WalkState->OpInfo->Class)
94     {
95     case AML_CLASS_ASCII:
96     case AML_CLASS_PREFIX:
97         /*
98          * Starts with a valid prefix or ASCII char, this is a name
99          * string. Convert the bare name string to a namepath.
100          */
101         WalkState->Opcode = AML_INT_NAMEPATH_OP;
102         WalkState->ArgTypes = ARGV_NAMESTRING;
103         break;

105     case AML_CLASS_UNKNOWN:

107         /* The opcode is unrecognized. Complain and skip unknown opcodes */

109         if (WalkState->PassNumber == 2)
110         {
111             ACPI_ERROR ((AE_INFO,
112                         "Unknown opcode 0x%.2X at table offset 0x%.4X, ignoring",
113                         WalkState->Opcode,
114                         (UINT32) (WalkState->AmlOffset + sizeof (ACPI_TABLE_HEADER))));

116             ACPI_DUMP_BUFFER ((WalkState->ParserState.Aml - 16), 48);

118 #ifdef ACPI_ASL_COMPILER
119             /*
120              * This is executed for the disassembler only. Output goes
121              * to the disassembled ASL output file.
122              */
123             AcpiOsPrintf (
124                 "/*\nError: Unknown opcode 0x%.2X at table offset 0x%.4X, contex
125                 WalkState->Opcode,
126                 (UINT32) (WalkState->AmlOffset + sizeof (ACPI_TABLE_HEADER)));
```

```

128      /* Dump the context surrounding the invalid opcode */
129
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   }
136
137   /* Increment past one-byte or two-byte opcode */
138
139   WalkState->ParserState.Aml++;
140   if (WalkState->Opcode > 0xFF) /* Can only happen if first byte is 0x5B */
141   {
142       WalkState->ParserState.Aml++;
143   }
144
145   return_ACPI_STATUS (AE_CTRL_PARSE_CONTINUE);
146
147   default:
148
149       /* Found opcode info, this is a normal opcode */
150
151       WalkState->ParserState.Aml += AcpiPsGetOpcodeSize (WalkState->Opcode);
152       WalkState->ArgTypes = WalkState->OpInfo->ParseArgs;
153       break;
154   }
155
156   return_ACPI_STATUS (AE_OK);
157 }
158
159
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  *****/
174
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;
184
185     ACPI_FUNCTION_TRACE_PTR (PsBuildNamedOp, WalkState);
186
187     UnnamedOp->Common.Value.Arg = NULL;
188     UnnamedOp->Common.ArgListLength = 0;
189     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         }
206
207         AcpiPsAppendArg (UnnamedOp, Arg);
208         INCREMENT_ARG_LIST (WalkState->ArgTypes);
209     }
210
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     }
218
219     /* We know that this arg is a name, move to next arg */
220
221     INCREMENT_ARG_LIST (WalkState->ArgTypes);
222
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;
228
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     }
235
236     if (!*Op)
237     {
238         return_ACPI_STATUS (AE_CTRL_PARSE_CONTINUE);
239     }
240
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     }
250
251     AcpiPsAppendArg (*Op, UnnamedOp->Common.Value.Arg);
252
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 oregion 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->OpInfo);
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                 }
515             }

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 }
```

new/usr/src/common/acpica/components/parser/psopcode.c

1

```
*****
33177 Fri Jan 17 07:30:31 2014
new/usr/src/common/acpica/components/parser/psopcode.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: psopcode - Parser/Interpreter opcode information table
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "acparser.h"
48 #include "acopcode.h"
49 #include "amlcode.h"

51 #define _COMPONENT          ACPI_PARSER
52 #define ACPI_MODULE_NAME    ("psopcode")

56 static const UINT8        AcpiGbl_ArgumentCount[] = {0,1,1,1,1,2,2,2,3,3,6};
```

new/usr/src/common/acpica/components/parser/psopcode.c

2

```
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
122     AML_NAME_OP
123     AML_ALIAS_OP
124     AML_MUTEX_OP
125     AML_EVENT_OP
126     AML_REGION_OP
127     AML_CREATE_FIELD_OP
128     AML_CREATE_BIT_FIELD_OP
129     AML_CREATE_BYTE_FIELD_OP
130     AML_CREATE_WORD_FIELD_OP
131     AML_CREATE_DWORD_FIELD_OP
132     AML_CREATE_QWORD_FIELD_OP
133     AML_INT_NAMEDFIELD_OP
134     AML_INT_METHODCALL_OP
135     AML_INT_NAMEPATH_OP

137     Opcodes that define named ACPI objects (AML_NAMED flag)

139     AML_SCOPE_OP
140     AML_DEVICE_OP
141     AML_THERMAL_ZONE_OP
142     AML_METHOD_OP
143     AML_POWER_RES_OP
144     AML_PROCESSOR_OP
145     AML_NAME_OP
146     AML_ALIAS_OP
147     AML_MUTEX_OP
148     AML_EVENT_OP
149     AML_REGION_OP
150     AML_INT_NAMEDFIELD_OP

152     Opcodes that contain executable AML as part of the definition that
153     must be deferred until needed

155     AML_METHOD_OP
156     AML_VAR_PACKAGE_OP
157     AML_CREATE_FIELD_OP
158     AML_CREATE_BIT_FIELD_OP
159     AML_CREATE_BYTE_FIELD_OP
160     AML_CREATE_WORD_FIELD_OP
161     AML_CREATE_DWORD_FIELD_OP
162     AML_CREATE_QWORD_FIELD_OP
163     AML_REGION_OP
164     AML_BUFFER_OP

166     Field opcodes

168     AML_CREATE_FIELD_OP
169     AML_FIELD_OP
170     AML_INDEX_FIELD_OP
171     AML_BANK_FIELD_OP

173     Field "Create" opcodes

175     AML_CREATE_FIELD_OP
176     AML_CREATE_BIT_FIELD_OP
177     AML_CREATE_BYTE_FIELD_OP
178     AML_CREATE_WORD_FIELD_OP
179     AML_CREATE_DWORD_FIELD_OP
180     AML_CREATE_QWORD_FIELD_OP

182     *****/

185 /*
186  * Master Opcode information table. A summary of everything we know about each

```

```

187  * opcode, all in one place.
188  */
189  const ACPI_OPCODE_INFO    AcpiGbl_AmlOpInfo[AML_NUM_OPCODES] =
190  {
191  /*! [Begin] no source code translation */
192  /* Index                Name                Parser Args                Interpreter Ar
193
194  /* 00 */ ACPI_OP ("Zero",                ARGP_ZERO_OP,                ARG1_ZERO_OP,
195  /* 01 */ ACPI_OP ("One",                  ARGP_ONE_OP,                  ARG1_ONE_OP,
196  /* 02 */ ACPI_OP ("Alias",                ARGP_ALIAS_OP,                ARG1_ALIAS_OP,
197  /* 03 */ ACPI_OP ("Name",                ARGP_NAME_OP,                  ARG1_NAME_OP,
198  /* 04 */ ACPI_OP ("ByteConst",            ARGP_BYTE_OP,                  ARG1_BYTE_OP,
199  /* 05 */ ACPI_OP ("WordConst",            ARGP_WORD_OP,                  ARG1_WORD_OP,
200  /* 06 */ ACPI_OP ("DwordConst",            ARGP_DWORD_OP,                ARG1_DWORD_OP,
201  /* 07 */ ACPI_OP ("String",                ARGP_STRING_OP,                ARG1_STRING_O
202  /* 08 */ ACPI_OP ("Scope",                ARGP_SCOPE_OP,                ARG1_SCOPE_OP,
203  /* 09 */ ACPI_OP ("Buffer",                ARGP_BUFFER_OP,                ARG1_BUFFER_O
204  /* 0A */ ACPI_OP ("Package",                ARGP_PACKAGE_OP,                ARG1_PACKAGE_
205  /* 0B */ ACPI_OP ("Method",                ARGP_METHOD_OP,                ARG1_METHOD_O
206  /* 0C */ ACPI_OP ("Local0",                ARGP_LOCAL0,                  ARG1_LOCAL0,
207  /* 0D */ ACPI_OP ("Local1",                ARGP_LOCAL1,                  ARG1_LOCAL1,
208  /* 0E */ ACPI_OP ("Local2",                ARGP_LOCAL2,                  ARG1_LOCAL2,
209  /* 0F */ ACPI_OP ("Local3",                ARGP_LOCAL3,                  ARG1_LOCAL3,
210  /* 10 */ ACPI_OP ("Local4",                ARGP_LOCAL4,                  ARG1_LOCAL4,
211  /* 11 */ ACPI_OP ("Local5",                ARGP_LOCAL5,                  ARG1_LOCAL5,
212  /* 12 */ ACPI_OP ("Local6",                ARGP_LOCAL6,                  ARG1_LOCAL6,
213  /* 13 */ ACPI_OP ("Local7",                ARGP_LOCAL7,                  ARG1_LOCAL7,
214  /* 14 */ ACPI_OP ("Arg0",                  ARGP_ARG0,                    ARG1_ARG0,
215  /* 15 */ ACPI_OP ("Arg1",                  ARGP_ARG1,                    ARG1_ARG1,
216  /* 16 */ ACPI_OP ("Arg2",                  ARGP_ARG2,                    ARG1_ARG2,
217  /* 17 */ ACPI_OP ("Arg3",                  ARGP_ARG3,                    ARG1_ARG3,
218  /* 18 */ ACPI_OP ("Arg4",                  ARGP_ARG4,                    ARG1_ARG4,
219  /* 19 */ ACPI_OP ("Arg5",                  ARGP_ARG5,                    ARG1_ARG5,
220  /* 1A */ ACPI_OP ("Arg6",                  ARGP_ARG6,                    ARG1_ARG6,
221  /* 1B */ ACPI_OP ("Store",                ARGP_STORE_OP,                ARG1_STORE_OP,
222  /* 1C */ ACPI_OP ("RefOf",                ARGP_REF_OF_OP,                ARG1_REF_OF_O
223  /* 1D */ ACPI_OP ("Add",                  ARGP_ADD_OP,                  ARG1_ADD_OP,
224  /* 1E */ ACPI_OP ("Concatenate",            ARGP_CONCAT_OP,                ARG1_CONCAT_O
225  /* 1F */ ACPI_OP ("Subtract",                ARGP_SUBTRACT_OP,                ARG1_SUBTRACT
226  /* 20 */ ACPI_OP ("Increment",                ARGP_INCREMENT_OP,                ARG1_INCREMEN
227  /* 21 */ ACPI_OP ("Decrement",                ARGP_DECREMENT_OP,                ARG1_DECREMEN
228  /* 22 */ ACPI_OP ("Multiply",                ARGP_MULTIPLY_OP,                ARG1_MULTIPLY
229  /* 23 */ ACPI_OP ("Divide",                ARGP_DIVIDE_OP,                ARG1_DIVIDE_O
230  /* 24 */ ACPI_OP ("ShiftLeft",                ARGP_SHIFT_LEFT_OP,                ARG1_SHIFT_LE
231  /* 25 */ ACPI_OP ("ShiftRight",                ARGP_SHIFT_RIGHT_OP,                ARG1_SHIFT_RI
232  /* 26 */ ACPI_OP ("And",                  ARGP_BIT_AND_OP,                ARG1_BIT_AND_
233  /* 27 */ ACPI_OP ("NAnd",                  ARGP_BIT_NAND_OP,                ARG1_BIT_NAND
234  /* 28 */ ACPI_OP ("Or",                  ARGP_BIT_OR_OP,                ARG1_BIT_OR_O
235  /* 29 */ ACPI_OP ("NOr",                  ARGP_BIT_NOR_OP,                ARG1_BIT_NOR_
236  /* 2A */ ACPI_OP ("XOr",                  ARGP_BIT_XOR_OP,                ARG1_BIT_XOR_
237  /* 2B */ ACPI_OP ("Not",                  ARGP_BIT_NOT_OP,                ARG1_BIT_NOT_
238  /* 2C */ ACPI_OP ("FindSetLeftBit",            ARGP_FIND_SET_LEFT_BIT_OP,            ARG1_FIND_SET
239  /* 2D */ ACPI_OP ("FindSetRightBit",            ARGP_FIND_SET_RIGHT_BIT_OP,            ARG1_FIND_SET
240  /* 2E */ ACPI_OP ("DerefOf",                ARGP_DEREF_OF_OP,                ARG1_DEREF_OF
241  /* 2F */ ACPI_OP ("Notify",                ARGP_NOTIFY_OP,                ARG1_NOTIFY_O
242  /* 30 */ ACPI_OP ("SizeOf",                ARGP_SIZE_OF_OP,                ARG1_SIZE_OF_
243  /* 31 */ ACPI_OP ("Index",                ARGP_INDEX_OP,                ARG1_INDEX_OP,
244  /* 32 */ ACPI_OP ("Match",                ARGP_MATCH_OP,                ARG1_MATCH_OP,
245  /* 33 */ ACPI_OP ("CreateDwordField",            ARGP_CREATE_DWORD_FIELD_OP,            ARG1_CREATE_D
246  /* 34 */ ACPI_OP ("CreateWordField",            ARGP_CREATE_WORD_FIELD_OP,            ARG1_CREATE_W
247  /* 35 */ ACPI_OP ("CreateByteField",            ARGP_CREATE_BYTE_FIELD_OP,            ARG1_CREATE_B
248  /* 36 */ ACPI_OP ("CreateBitField",            ARGP_CREATE_BIT_FIELD_OP,            ARG1_CREATE_B
249  /* 37 */ ACPI_OP ("ObjectType",                ARGP_TYPE_OP,                ARG1_TYPE_OP,
250  /* 38 */ ACPI_OP ("LAnd",                  ARGP_LAND_OP,                ARG1_LAND_OP,
251  /* 39 */ ACPI_OP ("LOr",                  ARGP_LOR_OP,                ARG1_LOR_OP,
252  /* 3A */ ACPI_OP ("LNot",                  ARGP_LNOT_OP,                ARG1_LNOT_OP,

```

```

253 /* 3B */ ACPI_OP ("LEqual",      ARGP_LEQUAL_OP,      ARG1_LEQUAL_O
254 /* 3C */ ACPI_OP ("LGreater",    ARGP_LGREATER_OP,    ARG1_LGREATER
255 /* 3D */ ACPI_OP ("LLess",       ARGP_LLESS_OP,       ARG1_LLESS_OP
256 /* 3E */ ACPI_OP ("If",         ARG1_IF_OP,         ARG1_IF_OP,
257 /* 3F */ ACPI_OP ("Else",       ARG1_ELSE_OP,       ARG1_ELSE_OP,
258 /* 40 */ ACPI_OP ("While",      ARG1_WHILE_OP,      ARG1_WHILE_OP
259 /* 41 */ ACPI_OP ("Noop",       ARG1_NOOP_OP,       ARG1_NOOP_OP,
260 /* 42 */ ACPI_OP ("Return",     ARG1_RETURN_O,      ARG1_RETURN_O
261 /* 43 */ ACPI_OP ("Break",     ARG1_BREAK_OP,     ARG1_BREAK_OP
262 /* 44 */ ACPI_OP ("BreakPoint", ARG1_BREAK_PO,     ARG1_BREAK_PO
263 /* 45 */ ACPI_OP ("Ones",      ARG1_ONES_OP,      ARG1_ONES_OP,

```

```
265 /* Prefixed opcodes (Two-byte opcodes with a prefix op) */
```

```

267 /* 46 */ ACPI_OP ("Mutex",      ARGP_MUTEX_OP,      ARG1_MUTEX_OP
268 /* 47 */ ACPI_OP ("Event",     ARG1_EVENT_OP,     ARG1_EVENT_OP
269 /* 48 */ ACPI_OP ("CondRefOf",  ARG1_COND_REF_OF_OP, ARG1_COND_REF
270 /* 49 */ ACPI_OP ("CreateField", ARG1_CREATE_F,     ARG1_CREATE_F
271 /* 4A */ ACPI_OP ("Load",      ARG1_LOAD_OP,      ARG1_LOAD_OP,
272 /* 4B */ ACPI_OP ("Stall",     ARG1_STALL_OP,     ARG1_STALL_OP
273 /* 4C */ ACPI_OP ("Sleep",    ARG1_SLEEP_OP,    ARG1_SLEEP_OP
274 /* 4D */ ACPI_OP ("Acquire",   ARG1_ACQUIRE_OP,  ARG1_ACQUIRE_
275 /* 4E */ ACPI_OP ("Signal",    ARG1_SIGNAL_O,     ARG1_SIGNAL_O
276 /* 4F */ ACPI_OP ("Wait",     ARG1_WAIT_OP,     ARG1_WAIT_OP,
277 /* 50 */ ACPI_OP ("Reset",    ARG1_RESET_OP,    ARG1_RESET_OP
278 /* 51 */ ACPI_OP ("Release",   ARG1_RELEASE_OP,   ARG1_RELEASE_
279 /* 52 */ ACPI_OP ("FromBCD",   ARG1_FROM_BCD_OP,  ARG1_FROM_BCD
280 /* 53 */ ACPI_OP ("ToBCD",    ARG1_TO_BCD_O,     ARG1_TO_BCD_O
281 /* 54 */ ACPI_OP ("Unload",   ARG1_UNLOAD_O,     ARG1_UNLOAD_O
282 /* 55 */ ACPI_OP ("Revision",  ARG1_REVISION,     ARG1_REVISION
283 /* 56 */ ACPI_OP ("Debug",    ARG1_DEBUG_OP,    ARG1_DEBUG_OP
284 /* 57 */ ACPI_OP ("Fatal",    ARG1_FATAL_OP,    ARG1_FATAL_OP
285 /* 58 */ ACPI_OP ("OperationRegion", ARG1_REGION_O,     ARG1_REGION_O
286 /* 59 */ ACPI_OP ("Field",    ARG1_FIELD_OP,    ARG1_FIELD_OP
287 /* 5A */ ACPI_OP ("Device",   ARG1_DEVICE_O,     ARG1_DEVICE_O
288 /* 5B */ ACPI_OP ("Processor", ARG1_PROCESSO,     ARG1_PROCESSO
289 /* 5C */ ACPI_OP ("PowerResource", ARG1_POWER_RE,     ARG1_POWER_RE
290 /* 5D */ ACPI_OP ("ThermalZone", ARG1_THERMAL,     ARG1_THERMAL
291 /* 5E */ ACPI_OP ("IndexField", ARG1_INDEX_FI,     ARG1_INDEX_FI
292 /* 5F */ ACPI_OP ("BankField", ARG1_BANK_FIE,     ARG1_BANK_FIE

```

```
294 /* Internal opcodes that map to invalid AML opcodes */
```

```

296 /* 60 */ ACPI_OP ("LNotEqual",  ARGP_LNOTEQUAL_OP,  ARG1_LNOTEQUA
297 /* 61 */ ACPI_OP ("LLessEqual", ARGP_LLESSEQUAL_OP, ARG1_LLESSEQU
298 /* 62 */ ACPI_OP ("LGreaterEqual", ARG1_LGREATEREQUA, ARG1_LGREATER
299 /* 63 */ ACPI_OP ("-NamePath-", ARG1_NAMEPATH_OP,  ARG1_NAMEPATH
300 /* 64 */ ACPI_OP ("-MethodCall-", ARG1_METHODCALL_OP, ARG1_METHODDCA
301 /* 65 */ ACPI_OP ("-ByteList-", ARG1_BYTIELIST_OP,  ARG1_BYTIELIST
302 /* 66 */ ACPI_OP ("-ReservedField-", ARG1_RESERVEDFIELD_OP, ARG1_RESERVED
303 /* 67 */ ACPI_OP ("-NamedField-", ARG1_NAMEDFIELD_OP,  ARG1_NAMEDFIE
304 /* 68 */ ACPI_OP ("-AccessField-", ARG1_ACCESSFIELD_OP, ARG1_ACCESSFI
305 /* 69 */ ACPI_OP ("-StaticString-", ARG1_STATICSTRING_OP, ARG1_STATICCST
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", ARG1_QWORD_OP, ARG1_QWORD_OP
314 /* 6F */ ACPI_OP ("Package", /* Var */ ARG1_VAR_PACKAGE_OP, ARG1_VAR_PACK
315 /* 70 */ ACPI_OP ("ConcatenateResTemplate", ARG1_CONCAT_RES_OP, ARG1_CONCAT_R
316 /* 71 */ ACPI_OP ("Mod", ARG1_MOD_OP, ARG1_MOD_OP,
317 /* 72 */ ACPI_OP ("CreateQwordField", ARG1_CREATE_Q, ARG1_CREATE_Q
318 /* 73 */ ACPI_OP ("ToBuffer", ARG1_TO_BUFFER_OP, ARG1_TO_BUFFERE

```

```

319 /* 74 */ ACPI_OP ("ToDecimalString", ARG1_TO_DEC_STR_OP, ARG1_TO_DEC_S
320 /* 75 */ ACPI_OP ("ToHexString", ARG1_TO_HEX_STR_OP, ARG1_TO_HEX_S
321 /* 76 */ ACPI_OP ("ToInteger", ARG1_TO_INTEGER_OP, ARG1_TO_INTEG
322 /* 77 */ ACPI_OP ("ToString", ARG1_TO_STRING_OP, ARG1_TO_STRIN
323 /* 78 */ ACPI_OP ("CopyObject", ARG1_COPY_OP, ARG1_COPY_OP,
324 /* 79 */ ACPI_OP ("Mid", ARG1_MID_OP, ARG1_MID_OP,
325 /* 7A */ ACPI_OP ("Continue", ARG1_CONTINUE_OP, ARG1_CONTINUE
326 /* 7B */ ACPI_OP ("LoadTable", ARG1_LOAD_TABLE_OP, ARG1_LOAD_TAB
327 /* 7C */ ACPI_OP ("DataTableRegion", ARG1_DATA_REGION_OP, ARG1_DATA_REG
328 /* 7D */ ACPI_OP ("[EvalSubTree]", ARG1_SCOPE_OP, ARG1_SCOPE_OP

```

```
330 /* ACPI 3.0 opcodes */
```

```

332 /* 7E */ ACPI_OP ("Timer", ARG1_TIMER_OP, ARG1_TIMER_OP
336 /* 7E */ ACPI_OP ("Timer", ARG1_TIMER_OP, ARG1_TIMER_OP

```

```
334 /* ACPI 5.0 opcodes */
```

```
338 /*! [End] no source code translation !*/
339 */
```

```

336 /* 7F */ ACPI_OP ("-ConnectField-", ARG1_CONNECTFIELD_OP, ARG1_CONNECTTF
337 /* 80 */ ACPI_OP ("-ExtAccessField-", ARG1_CONNECTFIELD_OP, ARG1_CONNECTTF
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 /*
348 /*
349 /* 0x00 */ 0x00, 0x01, _UNK, _UNK, _UNK, _UNK, 0x02, _UNK,
350 /* 0x08 */ 0x03, _UNK, 0x04, 0x05, 0x06, 0x07, 0x0E, _UNK,
351 /* 0x10 */ 0x08, 0x09, 0x0a, 0x0F, 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, _PF, _PF,
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, _PF, _UNK, _PF, _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
385 * pair. It returns an index into the opcode table (AcpiGbl_AmlOpInfo)
386 */
387 static const UINT8 AcpiGbl_LongOpIndex[NUM_EXTENDED_OPCODE] =
388 {
389 /*          0      1      2      3      4      5      6      7 */
390 /*          8      9      A      B      C      D      E      F */
391 /* 0x00 */ _UNK, 0x46, 0x47, _UNK, _UNK, _UNK, _UNK, _UNK,
392 /* 0x08 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
393 /* 0x10 */ _UNK, _UNK, 0x48, 0x49, _UNK, _UNK, _UNK, _UNK,
394 /* 0x18 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, 0x7B,
395 /* 0x20 */ 0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f, 0x50, 0x51,
396 /* 0x28 */ 0x52, 0x53, 0x54, _UNK, _UNK, _UNK, _UNK, _UNK,
397 /* 0x30 */ 0x55, 0x56, 0x57, 0x7e, _UNK, _UNK, _UNK, _UNK,
398 /* 0x38 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
399 /* 0x40 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
400 /* 0x48 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
401 /* 0x50 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
402 /* 0x58 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
403 /* 0x60 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
404 /* 0x68 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
405 /* 0x70 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
406 /* 0x78 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
407 /* 0x80 */ 0x58, 0x59, 0x5a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f,
408 /* 0x88 */ 0x7C,
340 };

412 /*****
413 *
414 * FUNCTION:      AcpiPsGetOpcodeInfo
415 *
416 * PARAMETERS:    Opcode                - The AML opcode
417 *
418 * RETURN:        A pointer to the info about the opcode.
419 *
420 * DESCRIPTION:   Find AML opcode description based on the opcode.
421 *               NOTE: This procedure must ALWAYS return a valid pointer!
422 *
423 *****/
425 const ACPI_OPCODE_INFO *
426 AcpiPsGetOpcodeInfo (
427     UINT16                Opcode)
428 {
429     ACPI_FUNCTION_NAME (PsGetOpcodeInfo);

432 /*
433  * Detect normal 8-bit opcode or extended 16-bit opcode
434  */
435 if (!(Opcode & 0xFF00))
436 {
437     /* Simple (8-bit) opcode: 0-255, can't index beyond table */

439     return (&AcpiGbl_AmlOpInfo [AcpiGbl_ShortOpIndex [(UINT8) Opcode]]);
440 }

442 if (((Opcode & 0xFF00) == AML_EXTENDED_OPCODE) &&
443     (((UINT8) Opcode) <= MAX_EXTENDED_OPCODE))
444 {
445     /* Valid extended (16-bit) opcode */

447     return (&AcpiGbl_AmlOpInfo [AcpiGbl_LongOpIndex [(UINT8) Opcode]]);
448 }

```

```

450 /* Unknown AML opcode */
452 ACPI_DEBUG_PRINT ((ACPI_DB_EXEC,
453     "Unknown AML opcode [%4.4X]\n", Opcode));

455 return (&AcpiGbl_AmlOpInfo [_UNK]);
456 }

459 /*****
460 *
461 * FUNCTION:      AcpiPsGetOpcodeName
462 *
463 * PARAMETERS:    Opcode                - The AML opcode
464 *
465 * RETURN:        A pointer to the name of the opcode (ASCII String)
466 *               Note: Never returns NULL.
467 *
468 * DESCRIPTION:   Translate an opcode into a human-readable string
469 *
470 *****/
472 char *
473 AcpiPsGetOpcodeName (
474     UINT16                Opcode)
475 {
476 #if defined(ACPI_DISASSEMBLER) || defined(ACPI_DEBUG_OUTPUT)

478     const ACPI_OPCODE_INFO *Op;

481     Op = AcpiPsGetOpcodeInfo (Opcode);

483     /* Always guaranteed to return a valid pointer */

485     return (Op->Name);

487 #else
488     return ("OpcodeName unavailable");
490 #endif
491 }

494 /*****
495 *
496 * FUNCTION:      AcpiPsGetArgumentCount
497 *
498 * PARAMETERS:    OpType                - Type associated with the AML opcode
499 *
500 * RETURN:        Argument count
501 *
502 * DESCRIPTION:   Obtain the number of expected arguments for an AML opcode
503 *
504 *****/
506 UINT8
507 AcpiPsGetArgumentCount (
508     UINT32                OpType)
509 {
511     if (OpType <= AML_TYPE_EXEC_6A_0T_1R)
512     {
513         return (AcpiGbl_ArgumentCount[OpType]);
514     }

```

`new/usr/src/common/acpica/components/parser/psopcode.c`

9

```
516     return (0);  
517 }
```



new/usr/src/common/acpica/components/parser/psopinfo.c

1

```
*****
9834 Fri Jan 17 07:30:32 2014
new/usr/src/common/acpica/components/parser/psopinfo.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: psopinfo - AML opcode information functions and dispatch tables
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"
48 #include "acopcode.h"
49 #include "amlcode.h"

52 #define _COMPONENT          ACPI_PARSER
53       ACPI_MODULE_NAME    ("psopinfo")

56 extern const UINT8      AcpiGbl_ShortOpIndex[];
57 extern const UINT8      AcpiGbl_LongOpIndex[];

59 static const UINT8      AcpiGbl_ArgumentCount[] = {0,1,1,1,1,2,2,2,3,3,6};
```

new/usr/src/common/acpica/components/parser/psopinfo.c

2

```
62 /*****
63 *
64 * FUNCTION:      AcpiPsGetOpcodeInfo
65 *
66 * PARAMETERS:    Opcode              - The AML opcode
67 *
68 * RETURN:        A pointer to the info about the opcode.
69 *
70 * DESCRIPTION:   Find AML opcode description based on the opcode.
71 * NOTE: This procedure must ALWAYS return a valid pointer!
72 *
73 *****/

75 const ACPI_OPCODE_INFO *
76 AcpiPsGetOpcodeInfo (
77     UINT16              Opcode)
78 {
79     #ifdef ACPI_DEBUG_OUTPUT
80     const char          *OpcodeName = "Unknown AML opcode";
81     #endif

83     ACPI_FUNCTION_NAME (PsGetOpcodeInfo);

86     /*
87      * Detect normal 8-bit opcode or extended 16-bit opcode
88      */
89     if (!(Opcode & 0xFF00))
90     {
91         /* Simple (8-bit) opcode: 0-255, can't index beyond table */

93         return (&AcpiGbl_AmlOpInfo [AcpiGbl_ShortOpIndex [(UINT8) Opcode]]);
94     }

96     if (((Opcode & 0xFF00) == AML_EXTENDED_OPCODE) &&
97         (((UINT8) Opcode) <= MAX_EXTENDED_OPCODE))
98     {
99         /* Valid extended (16-bit) opcode */

101         return (&AcpiGbl_AmlOpInfo [AcpiGbl_LongOpIndex [(UINT8) Opcode]]);
102     }

104 #if defined ACPI_ASL_COMPILER && defined ACPI_DEBUG_OUTPUT
105 #include "asldefine.h"

107     switch (Opcode)
108     {
109     case AML_RAW_DATA_BYTE:
110         OpcodeName = "-Raw Data Byte-";
111         break;

113     case AML_RAW_DATA_WORD:
114         OpcodeName = "-Raw Data Word-";
115         break;

117     case AML_RAW_DATA_DWORD:
118         OpcodeName = "-Raw Data Dword-";
119         break;

121     case AML_RAW_DATA_QWORD:
122         OpcodeName = "-Raw Data Qword-";
123         break;

125     case AML_RAW_DATA_BUFFER:
```

```

126     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)
177
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");
189 #endif
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 /*
227 /*      0      1      2      3      4      5      6      7      */
228 /*      8      9      A      B      C      D      E      F      */
229 /* 0x00 */ 0x00, 0x01, _UNK, _UNK, _UNK, _UNK, 0x02, _UNK,
230 /* 0x08 */ 0x03, _UNK, 0x04, 0x05, 0x06, 0x07, 0x0E, _UNK,
231 /* 0x10 */ 0x08, 0x09, 0x0A, 0x0F, 0x0B, _UNK, _UNK, _UNK,
232 /* 0x18 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
233 /* 0x20 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
234 /* 0x28 */ _UNK, _UNK, _UNK, _UNK, _UNK, 0x63, PFX, PFX,
235 /* 0x30 */ 0x67, 0x66, 0x68, 0x65, 0x69, 0x64, 0x6A, 0x7D,
236 /* 0x38 */ 0x7F, 0x80, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
237 /* 0x40 */ _UNK, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC,
238 /* 0x48 */ _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC,
239 /* 0x50 */ _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC, _ASC,
240 /* 0x58 */ _ASC, _ASC, _ASC, _UNK, _UNK, _UNK, _UNK, _ASC,
241 /* 0x60 */ 0x0C, 0x0D, 0x0E, 0x0F, 0x10, 0x11, 0x12, 0x13,
242 /* 0x68 */ 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1A, _UNK,
243 /* 0x70 */ 0x1B, 0x1C, 0x1D, 0x1E, 0x1F, 0x20, 0x21, 0x22,
244 /* 0x78 */ 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2A,
245 /* 0x80 */ 0x2B, 0x2C, 0x2D, 0x2E, 0x2F, 0x70, 0x71, 0x2F, 0x30,
246 /* 0x88 */ 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x72,
247 /* 0x90 */ 0x38, 0x39, 0x3A, 0x3B, 0x3C, 0x3D, 0x73, 0x74,
248 /* 0x98 */ 0x75, 0x76, _UNK, _UNK, 0x77, 0x78, 0x79, 0x7A,
249 /* 0xA0 */ 0x3E, 0x3F, 0x40, 0x41, 0x42, 0x43, 0x60, 0x61,
250 /* 0xA8 */ 0x62, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
251 /* 0xB0 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
252 /* 0xB8 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
253 /* 0xC0 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
254 /* 0xC8 */ _UNK, _UNK, _UNK, _UNK, 0x44, _UNK, _UNK, _UNK,
255 /* 0xD0 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
256 /* 0xD8 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
257 /* 0xE0 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
258 /* 0xE8 */ _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,

```

```

258 /* 0xF0 */      _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
259 /* 0xF8 */      _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, 0x45,
260 };

262 /*
263  * This table is indexed by the second opcode of the extended opcode
264  * pair. It returns an index into the opcode table (AcpiGbl_AmlOpInfo)
265  */
266 const UINT8 AcpiGbl_LongOpIndex[NUM_EXTENDED_OPCODE] =
267 {
268 /*          0      1      2      3      4      5      6      7 */
269 /*          8      9      A      B      C      D      E      F */
270 /* 0x00 */      _UNK, 0x46, 0x47, _UNK, _UNK, _UNK, _UNK, _UNK,
271 /* 0x08 */      _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
272 /* 0x10 */      _UNK, _UNK, 0x48, 0x49, _UNK, _UNK, _UNK, _UNK,
273 /* 0x18 */      _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, 0x7B,
274 /* 0x20 */      0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f, 0x50, 0x51,
275 /* 0x28 */      0x52, 0x53, 0x54, _UNK, _UNK, _UNK, _UNK, _UNK,
276 /* 0x30 */      0x55, 0x56, 0x57, 0x7e, _UNK, _UNK, _UNK, _UNK,
277 /* 0x38 */      _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
278 /* 0x40 */      _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
279 /* 0x48 */      _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
280 /* 0x50 */      _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
281 /* 0x58 */      _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
282 /* 0x60 */      _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
283 /* 0x68 */      _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
284 /* 0x70 */      _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
285 /* 0x78 */      _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK, _UNK,
286 /* 0x80 */      0x58, 0x59, 0x5a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f,
287 /* 0x88 */      0x7C,
288 };

```

new/usr/src/common/acpica/components/parser/psparse.c

1

```
*****
21770 Fri Jan 17 07:30:32 2014
new/usr/src/common/acpica/components/parser/psparse.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: psparse - Parser top level AML parse routines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 /*
46 * Parse the AML and build an operation tree as most interpreters,
47 * like Perl, do. Parsing is done by hand rather than with a YACC
48 * generated parser to tightly constrain stack and dynamic memory
49 * usage. At the same time, parsing is kept flexible and the code
50 * fairly compact by parsing based on a list of AML opcode
51 * templates in AmlOpInfo[]
52 */

54 #include "acpi.h"
55 #include "accommon.h"
56 #include "acparser.h"
57 #include "acdispat.h"
58 #include "amlcode.h"
```

new/usr/src/common/acpica/components/parser/psparse.c

2

```
59 #include "acinterp.h"

61 #define _COMPONENT ACPI_PARSER
62 #define ACPI_MODULE_NAME ("psparse")

65 /*****
66 *
67 * FUNCTION: AcpiPsGetOpcodeSize
68 *
69 * PARAMETERS: Opcode - An AML opcode
70 *
71 * RETURN: Size of the opcode, in bytes (1 or 2)
72 *
73 * DESCRIPTION: Get the size of the current opcode.
74 *
75 *****/

77 UINT32
78 AcpiPsGetOpcodeSize (
79     UINT32 Opcode)
80 {
81     /* Extended (2-byte) opcode if > 255 */
82
83     if (Opcode > 0x00FF)
84     {
85         return (2);
86     }
87
88     /* Otherwise, just a single byte opcode */
89
90     return (1);
91 }
92 }
93
94 unchanged portion omitted

130 /*****
131 *
132 * FUNCTION: AcpiPsCompleteThisOp
133 *
134 * PARAMETERS: WalkState - Current State
135 * Op - Op to complete
136 *
137 * RETURN: Status
138 *
139 * DESCRIPTION: Perform any cleanup at the completion of an Op.
140 *
141 *****/

143 ACPI_STATUS
144 AcpiPsCompleteThisOp (
145     ACPI_WALK_STATE *WalkState,
146     ACPI_PARSE_OBJECT *Op)
147 {
148     ACPI_PARSE_OBJECT *Prev;
149     ACPI_PARSE_OBJECT *Next;
150     const ACPI_OPCODE_INFO *ParentInfo;
151     ACPI_PARSE_OBJECT *ReplacementOp = NULL;
152     ACPI_STATUS Status = AE_OK;

155     ACPI_FUNCTION_TRACE_PTR (PsCompleteThisOp, Op);

158     /* 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_TREE)
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         break;
246     }

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

new/usr/src/common/acpica/components/parser/psscope.c

1

```
*****
8723 Fri Jan 17 07:30:32 2014
new/usr/src/common/acpica/components/parser/psscope.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: psscope - Parser scope stack management routines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "acparser.h"

49 #define _COMPONENT          ACPI_PARSER
50 #define ACPI_MODULE_NAME    ("psscope")

53 /*****
54 *
55 * FUNCTION:      AcpiPsGetParentScope
56 *
57 * PARAMETERS:    ParserState      - Current parser state object
58 *
```

new/usr/src/common/acpica/components/parser/psscope.c

2

```
59 * RETURN:        Pointer to an Op object
60 *
61 * DESCRIPTION:    Get parent of current op being parsed
62 *
63 *****/

65 ACPI_PARSE_OBJECT *
66 AcpiPsGetParentScope (
67     ACPI_PARSE_STATE      *ParserState)
68 {
69
70     return (ParserState->Scope->ParseScope.Op);
71 }

    unchanged_portion_omitted
```



new/usr/src/common/acpica/components/parser/pstree.c

1

```
*****
8614 Fri Jan 17 07:30:33 2014
new/usr/src/common/acpica/components/parser/pstree.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: pstree - Parser op tree manipulation/traversal/search
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __PSTREE_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acparser.h"
50 #include "amlcode.h"

52 #define _COMPONENT          ACPI_PARSER
53       ACPI_MODULE_NAME    ("pstree")

55 /* Local prototypes */

57 #ifdef ACPI_OBSOLETE_FUNCTIONS
58 ACPI_PARSE_OBJECT *
```

new/usr/src/common/acpica/components/parser/pstree.c

2

```
59 AcpiPsGetChild (
60     ACPI_PARSE_OBJECT      *op);
61 #endif

64 /*****
65 *
66 * FUNCTION:      AcpiPsGetArg
67 *
68 * PARAMETERS:    Op          - Get an argument for this op
69 *                Argn        - Nth argument to get
70 *
71 * RETURN:        The argument (as an Op object). NULL if argument does not exist
72 *
73 * DESCRIPTION:   Get the specified op's argument.
74 *
75 *****/

77 ACPI_PARSE_OBJECT *
78 AcpiPsGetArg (
79     ACPI_PARSE_OBJECT      *Op,
80     UINT32                  Argn)
81 {
82     ACPI_PARSE_OBJECT      *Arg = NULL;
83     const ACPI_OPCODE_INFO *OpInfo;

86     ACPI_FUNCTION_ENTRY ();

88 /*
89     if (Op->Common.AmlOpcode == AML_INT_CONNECTION_OP)
90     {
91         return (Op->Common.Value.Arg);
92     }
93 */

94     /* Get the info structure for this opcode */

96     OpInfo = AcpiPsGetOpcodeInfo (Op->Common.AmlOpcode);
97     if (OpInfo->Class == AML_CLASS_UNKNOWN)
98     {
99         /* Invalid opcode or ASCII character */

101         return (NULL);
102     }

104     /* Check if this opcode requires argument sub-objects */

106     if (!(OpInfo->Flags & AML_HAS_ARGS))
107     {
108         /* Has no linked argument objects */

110         return (NULL);
111     }

113     /* Get the requested argument object */

115     Arg = Op->Common.Value.Arg;
116     while (Arg && Argn)
117     {
118         Argn--;
119         Arg = Arg->Common.Next;
120     }

122     return (Arg);
123 }

unchanged_portion_omitted_
```

```

291 #ifndef ACPI_OBSOLETE_FUNCTIONS
292 /*****
293  *
294  * FUNCTION:      AcpiPsGetChild
295  *
296  * PARAMETERS:    Op          - Get the child of this Op
297  *
298  * RETURN:        Child Op, Null if none is found.
299  *
300  * DESCRIPTION:   Get op's children or NULL if none
301  *
302  *****/
304 ACPI_PARSE_OBJECT *
305 AcpiPsGetChild (
306     ACPI_PARSE_OBJECT *Op)
307 {
308     ACPI_PARSE_OBJECT *Child = NULL;

311     ACPI_FUNCTION_ENTRY ();

314     switch (Op->Common.AmlOpcode)
315     {
316     case AML_SCOPE_OP:
317     case AML_ELSE_OP:
318     case AML_DEVICE_OP:
319     case AML_THERMAL_ZONE_OP:
320     case AML_INT_METHODCALL_OP:

322         Child = AcpiPsGetArg (Op, 0);
323         break;

325     case AML_BUFFER_OP:
326     case AML_PACKAGE_OP:
327     case AML_METHOD_OP:
328     case AML_IF_OP:
329     case AML_WHILE_OP:
330     case AML_FIELD_OP:

332         Child = AcpiPsGetArg (Op, 1);
333         break;

335     case AML_POWER_RES_OP:
336     case AML_INDEX_FIELD_OP:

338         Child = AcpiPsGetArg (Op, 2);
339         break;

341     case AML_PROCESSOR_OP:
342     case AML_BANK_FIELD_OP:

344         Child = AcpiPsGetArg (Op, 3);
345         break;

347     default:

346     default:
349         /* All others have no children */

```

```

351         break;
352     }

354     return (Child);
355 }
356 #endif

```

new/usr/src/common/acpica/components/parser/psutils.c

1

```
*****
7255 Fri Jan 17 07:30:33 2014
new/usr/src/common/acpica/components/parser/psutils.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: psutils - Parser miscellaneous utilities (Parser only)
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "acparser.h"
48 #include "amlcode.h"

50 #define _COMPONENT          ACPI_PARSER
51 #define ACPI_MODULE_NAME    ("psutils")

54 /*****
55 *
56 * FUNCTION:      AcpiPsCreateScopeOp
57 *
58 * PARAMETERS:    None
```

new/usr/src/common/acpica/components/parser/psutils.c

2

```
59 *
60 * RETURN:        A new Scope object, null on failure
61 *
62 * DESCRIPTION:    Create a Scope and associated namepath op with the root name
63 *
64 *****/

66 ACPI_PARSE_OBJECT *
67 AcpiPsCreateScopeOp (
68     void)
69 {
70     ACPI_PARSE_OBJECT      *ScopeOp;

73     ScopeOp = AcpiPsAllocOp (AML_SCOPE_OP);
74     if (!ScopeOp)
75     {
76         return (NULL);
77     }

79     ScopeOp->Named.Name = ACPI_ROOT_NAME;
80     return (ScopeOp);
81 }

unchanged_portion_omitted_

240 /*
241 * Is "c" a namestring prefix character?
242 */
243 BOOLEAN
244 AcpiPsIsPrefixChar (
245     UINT32                c)
246 {
247     return ((BOOLEAN) (c == '\\' || c == '^'));
248 }

251 /*
241 * Get op's name (4-byte name segment) or 0 if unnamed
242 */
243 UINT32
244 AcpiPsGetName (
245     ACPI_PARSE_OBJECT      *Op)
246 {
248     /* The "generic" object has no name associated with it */

250     if (Op->Common.Flags & ACPI_PARSEOP_GENERIC)
251     {
252         return (0);
253     }

255     /* Only the "Extended" parse objects have a name */

257     return (Op->Named.Name);
258 }

unchanged_portion_omitted_
```

new/usr/src/common/acpica/components/parser/pswalk.c

1

```
*****
3794 Fri Jan 17 07:30:33 2014
new/usr/src/common/acpica/components/parser/pswalk.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: pswalk - Parser routines to walk parsed op tree(s)
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include "acpi.h"
46 #include "accommon.h"
47 #include "acparser.h"

49 #define _COMPONENT          ACPI_PARSER
50 #define ACPI_MODULE_NAME    ("pswalk")

53 /*****
54 *
55 * FUNCTION:      AcpiPsDeleteParseTree
56 *
57 * PARAMETERS:    SubtreeRoot      - Root of tree (or subtree) to delete
58 *
59 *****/
```

new/usr/src/common/acpica/components/parser/pswalk.c

2

```
59 * RETURN:      None
60 *
61 * DESCRIPTION: Delete a portion of or an entire parse tree.
62 *
63 *****/

65 void
66 AcpiPsDeleteParseTree (
67     ACPI_PARSE_OBJECT *SubtreeRoot)
68 {
69     ACPI_PARSE_OBJECT *Op = SubtreeRoot;
70     ACPI_PARSE_OBJECT *Next = NULL;
71     ACPI_PARSE_OBJECT *Parent = NULL;

74     ACPI_FUNCTION_TRACE_PTR (PsDeleteParseTree, SubtreeRoot);

77     /* Visit all nodes in the subtree */

79     while (Op)
80     {
81         /* Check if we are not ascending */

83         if (Op != Parent)
84         {
85             /* Look for an argument or child of the current op */

87             Next = AcpiPsGetArg (Op, 0);
88             if (Next)
89             {
90                 /* Still going downward in tree (Op is not completed yet) */

92                 Op = Next;
93                 continue;
94             }
95         }

97         /* No more children, this Op is complete. */

99         Next = Op->Common.Next;
100        Parent = Op->Common.Parent;

102        AcpiPsFreeOp (Op);

104        /* If we are back to the starting point, the walk is complete. */

106        if (Op == SubtreeRoot)
107        {
108            return_VOID;
109        }
110        if (Next)
111        {
112            Op = Next;
113        }
114        else
115        {
116            Op = Parent;
117        }
118    }

120    return_VOID;
121 }
```

unchanged\_portion\_omitted

new/usr/src/common/acpica/components/parser/psxface.c

1

```
*****
11999 Fri Jan 17 07:30:34 2014
new/usr/src/common/acpica/components/parser/psxface.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: psxface - Parser external interfaces
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __PSXFACE_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acparser.h"
49 #include "acdispat.h"
50 #include "acinterp.h"
51 #include "actables.h"

54 #define _COMPONENT          ACPI_PARSER
55         ACPI_MODULE_NAME    ("psxface")

57 /* Local Prototypes */
```

new/usr/src/common/acpica/components/parser/psxface.c

2

```
59 static void
60 AcpiPsStartTrace (
61     ACPI_EVALUATE_INFO    *Info);

63 static void
64 AcpiPsStopTrace (
65     ACPI_EVALUATE_INFO    *Info);

67 static void
68 AcpiPsUpdateParameterList (
69     ACPI_EVALUATE_INFO    *Info,
70     UINT16                 Action);

73 /*****
74 *
75 * FUNCTION:      AcpiDebugTrace
76 *
77 * PARAMETERS:    MethodName      - Valid ACPI name string
78 *                DebugLevel      - Optional level mask. 0 to use default
79 *                DebugLayer      - Optional layer mask. 0 to use default
80 *                Flags            - bit 1: one shot(1) or persistent(0)
81 *
82 * RETURN:        Status
83 *
84 * DESCRIPTION:    External interface to enable debug tracing during control
85 *                method execution
86 *
87 *****/

89 ACPI_STATUS
90 AcpiDebugTrace (
91     char                    *Name,
92     UINT32                  DebugLevel,
93     UINT32                  DebugLayer,
94     UINT32                  Flags)
95 {
96     ACPI_STATUS             Status;

99     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
100     if (ACPI_FAILURE (Status))
101     {
102         return (Status);
103     }

105     /* TBDs: Validate name, allow full path or just nameseg */

107     AcpiGbl_TraceMethodName = *ACPI_CAST_PTR (UINT32, Name);
108     AcpiGbl_TraceFlags = Flags;

110     if (DebugLevel)
111     {
112         AcpiGbl_TraceDbgLevel = DebugLevel;
113     }
114     if (DebugLayer)
115     {
116         AcpiGbl_TraceDbgLayer = DebugLayer;
117     }

119     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
120     return (AE_OK);
121 }

124 /*****
```

```

125 *
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))
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))
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     AcpiTbCheckDsdtdHeader ();

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.
353      * Start method evaluation with an implicit return of zero. This is done
354      * for Windows compatibility.
355      */
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\_



new/usr/src/common/acpica/components/resources/rsaddr.c

1

```
*****
13908 Fri Jan 17 07:30:34 2014
new/usr/src/common/acpica/components/resources/rsaddr.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: rsaddr - Address resource descriptors (16/32/64)
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __RSADDR_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acresrc.h"

50 #define _COMPONENT ACPI_RESOURCES
51 #define ACPI_MODULE_NAME ("rsaddr")

54 /*****
55 *
56 * AcpiRsConvertAddress16 - All WORD (16-bit) address resources
57 *
58 *****/
```

new/usr/src/common/acpica/components/resources/rsaddr.c

2

```
60 ACPI_RS_CONVERT_INFO AcpiRsConvertAddress16[5] =
61 {
62     {ACPI_RSC_INITGET, ACPI_RESOURCE_TYPE_ADDRESS16,
63      ACPI_RS_SIZE (ACPI_RESOURCE_ADDRESS16),
64      ACPI_RSC_TABLE_SIZE (AcpiRsConvertAddress16)},
65
66     {ACPI_RSC_INITSET, ACPI_RESOURCE_NAME_ADDRESS16,
67      sizeof (AML_RESOURCE_ADDRESS16),
68      0},
69
70     /* Resource Type, General Flags, and Type-Specific Flags */
71
72     {ACPI_RSC_ADDRESS, 0, 0, 0},
73
74     /*
75     * These fields are contiguous in both the source and destination:
76     * Address Granularity
77     * Address Range Minimum
78     * Address Range Maximum
79     * Address Translation Offset
80     * Address Length
81     */
82     {ACPI_RSC_MOVE16, ACPI_RS_OFFSET (Data.Address16.Granularity),
83      AML_OFFSET (Address16.Granularity),
84      5},
85
86     /* Optional ResourceSource (Index and String) */
87
88     {ACPI_RSC_SOURCE, ACPI_RS_OFFSET (Data.Address16.ResourceSource),
89      0,
90      sizeof (AML_RESOURCE_ADDRESS16)}
91 };

_____unchanged_portion_omitted_____
```

new/usr/src/common/acpica/components/resources/rscale.c

1

```
*****
23607 Fri Jan 17 07:30:35 2014
new/usr/src/common/acpica/components/resources/rscale.c
update to acpica-unix2-20140114
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOV's restructure
*****
1 / *****
2 *
3 * Module Name: rscale - Calculate stream and list lengths
4 *
5 *****/
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __RSALC_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acresrc.h"
49 #include "acnamesp.h"
50
51 #define _COMPONENT ACPI_RESOURCES
52 #define ACPI_MODULE_NAME ("rscale")
53
54 /* Local prototypes */
```

new/usr/src/common/acpica/components/resources/rscale.c

2

```
58 static UINT8
59 AcpiRsCountSetBits (
60     UINT16                      BitField);
61
62 static ACPI_RS_LENGTH
63 AcpiRsStructOptionLength (
64     ACPI_RESOURCE_SOURCE *ResourceSource);
65
66 static UINT32
67 AcpiRsStreamOptionLength (
68     UINT32                      ResourceLength,
69     UINT32                      MinimumTotalLength);
70
71 /*****
72 *
73 * FUNCTION:      AcpiRsCountSetBits
74 *
75 * PARAMETERS:    BitField - Field in which to count bits
76 *
77 * RETURN:        Number of bits set within the field
78 *
79 * DESCRIPTION:   Count the number of bits set in a resource field. Used for
80 *                (Short descriptor) interrupt and DMA lists.
81 *
82 * *****/
83
84 static UINT8
85 AcpiRsCountSetBits (
86     UINT16                      BitField)
87 {
88     UINT8                      BitsSet;
89
90     ACPI_FUNCTION_ENTRY ();
91
92     for (BitsSet = 0; BitField; BitsSet++)
93     {
94         /* Zero the least significant bit that is set */
95         BitField &= (UINT16) (BitField - 1);
96     }
97
98     return (BitsSet);
99 }
100
101 /*
102 * unchanged portion omitted
103 */
104
105 /*****
106 *
107 * FUNCTION:      AcpiRsGetAmlLength
108 *
109 * PARAMETERS:    Resource - Pointer to the resource linked list
110 *                ResourceListSize - Size of the resource linked list
111 *                SizeNeeded - Where the required size is returned
112 *
113 * RETURN:        Status
114 *
115 * DESCRIPTION:   Takes a linked list of internal resource descriptors and
116 *                calculates the size buffer needed to hold the corresponding
117 *                external resource byte stream.
118 *
119 * *****/
120
121 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)
229     while (Resource)
230     {
231         /* 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 UIN32 interrupt number)
347  */
348 TotalSize = (ACPI_RS_LENGTH)
349 (TotalSize +
350  ((Resource->Data.ExtendedIrq.InterruptCount - 1) * 4) +
351
352  /* Add the size of the optional ResourceSource info */
353
354  AcpiRsStructOptionLength (
355    &Resource->Data.ExtendedIrq.ResourceSource));
356  break;
357
358 case ACPI_RESOURCE_TYPE_GPIO:
359
360   TotalSize = (ACPI_RS_LENGTH) (TotalSize + (Resource->Data.Gpio.PinTa
361   Resource->Data.Gpio.ResourceSource.StringLength +
362   Resource->Data.Gpio.VendorLength);
363
364   break;
365
366 case ACPI_RESOURCE_TYPE_SERIAL_BUS:
367
368   TotalSize = AcpiGbl_AmlResourceSerialBusSizes [Resource->Data.Common
369
370   TotalSize = (ACPI_RS_LENGTH) (TotalSize +
371   Resource->Data.I2cSerialBus.ResourceSource.StringLength +
372   Resource->Data.I2cSerialBus.VendorLength);
373
374   break;
375
376 default:
377
378   break;
379 }
380
381 /* Update the total */
382
383 AmlSizeNeeded += TotalSize;
384
385 /* Point to the next object */
386
387 Resource = ACPI_ADD_PTR (ACPI_RESOURCE, Resource, Resource->Length);
388 }
389
390 /* Did not find an EndTag resource descriptor */
391
392 return ACPI_STATUS (AE_AML_NO_RESOURCE_END_TAG);
393 }
394
395
396
397
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 *
410 *      buffer needed to hold the corresponding internal resource
411 *      descriptor linked list.
412 *****/
413
414 ACPI_STATUS
415 AcpiRsGetListLength (
416     UINT8          *AmlBuffer,
417     UINT32         AmlBufferLength,
418     ACPI_SIZE      *SizeNeeded)
419 {
420     ACPI_STATUS      Status;
421     *EndAml;
422     *Buffer;
423     BufferSize;
424     Templ6;
425     ResourceLength;
426     ExtraStructBytes;
427     ResourceIndex;
428     MinimumAmlResourceLength;
429     AML_RESOURCE     *AmlResource;
430
431     ACPI_FUNCTION_TRACE (RsGetListLength);
432
433     *SizeNeeded = ACPI_RS_SIZE_MIN;      /* Minimum size is one EndTag */
434     *SizeNeeded = 0;
435     EndAml = AmlBuffer + AmlBufferLength;
436
437     /* Walk the list of AML resource descriptors */
438
439     while (AmlBuffer < EndAml)
440     {
441         /* Validate the Resource Type and Resource Length */
442
443         Status = AcpiUtValidateResource (NULL, AmlBuffer, &ResourceIndex);
444         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         }
453
454         AmlResource = (void *) AmlBuffer;
455
456         /* Get the resource length and base (minimum) AML size */
457
458         ResourceLength = AcpiUtGetResourceLength (AmlBuffer);
459         MinimumAmlResourceLength = AcpiGbl_ResourceAmlSizes[ResourceIndex];
460
461         /*
462          * Augment the size for descriptors with optional
463          * and/or variable length fields
464          */
465         ExtraStructBytes = 0;
466         Buffer = AmlBuffer + AcpiUtGetResourceHeaderLength (AmlBuffer);
467
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
512     * End Tag:
513     * This is the normal exit, add size of EndTag
514     */
515     *SizeNeeded += ACPI_RS_SIZE_MIN;
516     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++)
687     for (TableIndex = 0; TableIndex < 4 && !NameFound; TableIndex++)
688     {
689         if (*SubObjectList && /* Null object allowed */
690             ((ACPI_TYPE_STRING ==
691              (*SubObjectList->Common.Type) ||
692              ((ACPI_TYPE_LOCAL_REFERENCE ==
693               (*SubObjectList->Common.Type) &&
694                ((*SubObjectList->Reference.Class ==
695                 ACPI_REFCLASS_NAME))))
696             {
697                 NameFound = TRUE;
698             }
699             else
700             {
701                 /* Look at the next element */

703                 SubObjectList++;
704             }
705         }

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     }
731     else
732     {
733         /*

```

new/usr/src/common/acpica/components/resources/rscalc.c

11

```
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     }
738
739     /* Round up the size since each element must be aligned */
740
741     TempSizeNeeded = ACPI_ROUND_UP_TO_64BIT (TempSizeNeeded);
742
743     /* Point to the next ACPI_OPERAND_OBJECT */
744
745     TopObjectList++;
746 }
747
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

new/usr/src/common/acpica/components/resources/rscreate.c

1

```
*****
16706 Fri Jan 17 07:30:35 2014
new/usr/src/common/acpica/components/resources/rscreate.c
update to acpica-unix2-20140114
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: rscreate - Create resource lists/tables
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __RSCREATE_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acresrc.h"
49 #include "acnamesp.h"
50
51 #define _COMPONENT ACPI_RESOURCES
52 #define ACPI_MODULE_NAME ("rscreate")
53
54 /*****
55 *
56 *
57 * FUNCTION: AcpiBufferToResource
```

new/usr/src/common/acpica/components/resources/rscreate.c

2

```
58 *
59 * PARAMETERS: AmlBuffer - Pointer to the resource byte stream
60 *             AmlBufferLength - Length of the AmlBuffer
61 *             ResourcePtr - Where the converted resource is returned
62 *
63 * RETURN: Status
64 *
65 * DESCRIPTION: Convert a raw AML buffer to a resource list
66 *
67 *****/
68
69 ACPI_STATUS
70 AcpiBufferToResource (
71     UINT8 *AmlBuffer,
72     UINT16 AmlBufferLength,
73     ACPI_RESOURCE **ResourcePtr)
74 {
75     ACPI_STATUS Status;
76     ACPI_SIZE ListSizeNeeded;
77     void *Resource;
78     void *CurrentResourcePtr;
79
80 /*
81  * Note: we allow AE_AML_NO_RESOURCE_END_TAG, since an end tag
82  * is not required here.
83  */
84
85 /* Get the required length for the converted resource */
86
87 Status = AcpiRsGetListLength (AmlBuffer, AmlBufferLength,
88                               &ListSizeNeeded);
89 if (Status == AE_AML_NO_RESOURCE_END_TAG)
90 {
91     Status = AE_OK;
92 }
93 if (ACPI_FAILURE (Status))
94 {
95     return (Status);
96 }
97
98 /* Allocate a buffer for the converted resource */
99
100 Resource = ACPI_ALLOCATE_ZEROED (ListSizeNeeded);
101 CurrentResourcePtr = Resource;
102 if (!Resource)
103 {
104     return (AE_NO_MEMORY);
105 }
106
107 /* Perform the AML-to-Resource conversion */
108
109 Status = AcpiUtWalkAmlResources (NULL, AmlBuffer, AmlBufferLength,
110                                  AcpiRsConvertAmlToResources, &CurrentResourcePtr);
111 if (Status == AE_AML_NO_RESOURCE_END_TAG)
112 {
113     Status = AE_OK;
114 }
115 if (ACPI_FAILURE (Status))
116 {
117     ACPI_FREE (Resource);
118 }
119 else
120 {
121     *ResourcePtr = Resource;
122 }
```



```

124     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 many 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 {
151
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,
196     Status = AcpiUtWalkAmlResources (AmlStart, AmlBufferLength,
197     AcpiRsConvertAmlToResources, &Resource);
198     if (ACPI_FAILURE (Status))
199     {
200         return_ACPI_STATUS (Status);
201     }

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
214 *                PackageObject      - Pointer to an ACPI_OPERAND_OBJECT
215 *                package
216 *                OutputBuffer        - Pointer to the user's buffer
217 *
218 * RETURN:        Status: AE_OK if okay, else a valid ACPI_STATUS code.
219 *                If the OutputBuffer is too small, the error will be
220 *                AE_BUFFER_OVERFLOW and OutputBuffer->Length will point
221 *                to the size buffer needed.
222 *
223 * DESCRIPTION:    Takes the ACPI_OPERAND_OBJECT package and creates a
224 *                linked list of PCI interrupt descriptions
225 *
226 * NOTE: It is the caller's responsibility to ensure that the start of the
227 * output buffer is aligned properly (if necessary).
228 *****/

229 ACPI_STATUS
230 AcpiRsCreatePciRoutingTable (
231     ACPI_OPERAND_OBJECT *PackageObject,
232     ACPI_BUFFER *OutputBuffer)
233 {
234     {
235         UINT8            *Buffer;
236         ACPI_OPERAND_OBJECT **TopObjectList;
237         ACPI_OPERAND_OBJECT **SubObjectList;
238         ACPI_OPERAND_OBJECT *ObjDesc;
239         ACPI_SIZE         BufferSizeNeeded = 0;
240         UINT32            NumberOfElements;
241         UINT32            Index;
242         ACPI_PCI_ROUTING_TABLE *UserPrt;
243         ACPI_NAMESPACE_NODE *Node;
244         ACPI_STATUS        Status;
245         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     /*
342     * If the BIOS has erroneously reversed the _PRT SourceName (index 2)
343     * and the SourceIndex (index 3), fix it. _PRT is important enough to
344     * workaround this BIOS error. This also provides compatibility with
345     * other ACPI implementations.
346     */
347     ObjDesc = SubObjectList[3];
348     if (!ObjDesc || (ObjDesc->Common.Type != ACPI_TYPE_INTEGER))
349     {
350         SubObjectList[3] = SubObjectList[2];
351         SubObjectList[2] = ObjDesc;

352         ACPI_WARNING ((AE_INFO,
353                       "(PRT[%X].Source) SourceName and SourceIndex are reversed, fixed
354                       Index));
355     }

356     /*
357     * 3) Third subobject: Dereference the PRT.SourceName
358     * The name may be unresolved (slack mode), so allow a null object
359     */
360     ObjDesc = SubObjectList[2];
361     if (ObjDesc)
362     {
363         switch (ObjDesc->Common.Type)
364         {
365             case ACPI_TYPE_LOCAL_REFERENCE:

366                 if (ObjDesc->Reference.Class != ACPI_REFCLASS_NAME)
367                 {
368                     ACPI_ERROR ((AE_INFO,
369                                 "(PRT[%u].Source) Need name, found Reference Class 0x%X",
370                                 Index, ObjDesc->Reference.Class));
371                     return_ACPI_STATUS (AE_BAD_DATA);
372                 }
373             }

```

```

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
440  *
441  * PARAMETERS:    LinkedListBuffer  - Pointer to the resource linked list
442  *                OutputBuffer      - Pointer to the user's buffer
443  *
444  * RETURN:        Status AE_OK if okay, else a valid ACPI_STATUS code.
445  *                If the OutputBuffer is too small, the error will be
446  *                AE_BUFFER_OVERFLOW and OutputBuffer->Length will point
447  *                to the size buffer needed.
448  *
449  * DESCRIPTION:    Converts a list of device resources to an AML bytestream
450  *                to be used as input for the _SRS control method.
451  *
452  * DESCRIPTION:    Takes the linked list of device resources and
453  *                creates a bytestream to be used as input for the
454  *                _SRS control method.
455  *
456  *****/

451 ACPI_STATUS
452 AcpiRsCreateAmlResources (
453     ACPI_BUFFER      *ResourceList,
454     ACPI_BUFFER      *LinkedListBuffer,
455     ACPI_BUFFER      *OutputBuffer)
456 {
457     ACPI_STATUS      Status;
458     ACPI_SIZE         AmlSizeNeeded = 0;

460     ACPI_FUNCTION_TRACE (RsCreateAmlResources);

463     /* Params already validated, no need to re-validate here */
464     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "LinkedListBuffer = %p\n",
465         LinkedListBuffer));

466     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "ResourceList Buffer = %p\n",
467         ResourceList->Pointer));

468     /*
469      * Params already validated, so we don't re-validate here
470      *
471      * Pass the LinkedListBuffer into a module that calculates
472      * the buffer size needed for the byte stream.
473      */
474     Status = AcpiRsGetAmlLength (LinkedListBuffer,
475         &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);
492     Status = AcpiRsConvertResourcesToAml (LinkedListBuffer, AmlSizeNeeded,
493         OutputBuffer->Pointer);
494     if (ACPI_FAILURE (Status))
495     {
496         return_ACPI_STATUS (Status);
497     }
498     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "OutputBuffer %p Length %X\n",
499         OutputBuffer->Pointer, (UINT32) OutputBuffer->Length));
500     return_ACPI_STATUS (AE_OK);
501 }
```

new/usr/src/common/acpica/components/resources/rsdump.c

1

```
*****
17150 Fri Jan 17 07:30:36 2014
new/usr/src/common/acpica/components/resources/rsdump.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: rsdump - Functions to display the resource structures.
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __RSDUMP_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acresrc.h"

51 #define _COMPONENT          ACPI_RESOURCES
52 #define ACPI_MODULE_NAME    ("rsdump")

55 #if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)

57 /* Local prototypes */

59 static void
```

new/usr/src/common/acpica/components/resources/rsdump.c

2

```
60 AcpiRsOutString (
61     char                *Title,
62     char                *Value);

64 static void
65 AcpiRsOutInteger8 (
66     char                *Title,
67     UINT8               Value);

69 static void
70 AcpiRsOutInteger16 (
71     char                *Title,
72     UINT16              Value);

74 static void
75 AcpiRsOutInteger32 (
76     char                *Title,
77     UINT32              Value);

79 static void
80 AcpiRsOutInteger64 (
81     char                *Title,
82     UINT64              Value);

84 static void
85 AcpiRsOutTitle (
86     char                *Title);

88 static void
89 AcpiRsDumpByteList (
90     UINT16              Length,
91     UINT8               *Data);

93 static void
94 AcpiRsDumpWordList (
95     UINT16              Length,
96     UINT16              *Data);

98 static void
99 AcpiRsDumpDwordList (
100     UINT8               Length,
101     UINT32              *Data);

103 static void
104 AcpiRsDumpShortByteList (
105     UINT8               Length,
106     UINT8               *Data);

108 static void
109 AcpiRsDumpResourceSource (
110     ACPI_RESOURCE_SOURCE *ResourceSource);

112 static void
113 AcpiRsDumpAddressCommon (
114     ACPI_RESOURCE_DATA  *Resource);

116 static void
117 AcpiRsDumpDescriptor (
118     void                *Resource,
119     ACPI_RSDUMP_INFO    *Table);

122 /*****
123 *
124 * FUNCTION:    AcpiRsDumpDescriptor
125 */
```

```

126 * PARAMETERS:  Resource      - Buffer containing the resource
127 *              Table         - Table entry to decode the resource
128 *
129 * RETURN:      None
130 *
131 * DESCRIPTION: Dump a resource descriptor based on a dump table entry.
132 *
133 *****/

135 static void
136 AcpiRsDumpDescriptor (
137     void                *Resource,
138     ACPI_RSDUMP_INFO    *Table)
139 {
140     UINT8                *Target = NULL;
141     UINT8                *PreviousTarget;
142     char                 *Name;
143     UINT8                Count;

146     /* First table entry must contain the table length (# of table entries) */

148     Count = Table->Offset;

150     while (Count)
151     {
152         PreviousTarget = Target;
153         Target = ACPI_ADD_PTR (UINT8, Resource, Table->Offset);
154         Name = Table->Name;

156         switch (Table->Opcode)
157         {
158         case ACPI_RSD_TITLE:
159             /*
160              * Optional resource title
161              */
162             if (Table->Name)
163             {
164                 AcpiOsPrintf ("%s Resource\n", Name);
165             }
166             break;

168             /* Strings */

170         case ACPI_RSD_LITERAL:

172             AcpiRsOutString (Name, ACPI_CAST_PTR (char, Table->Pointer));
173             break;

175         case ACPI_RSD_STRING:

177             AcpiRsOutString (Name, ACPI_CAST_PTR (char, Target));
178             break;

180             /* Data items, 8/16/32/64 bit */

182         case ACPI_RSD_UINT8:

184             if (Table->Pointer)
185             {
186                 AcpiRsOutString (Name, ACPI_CAST_PTR (char,
187                     Table->Pointer [*Target]));
188             }
189             else
190             {
191                 AcpiRsOutInteger8 (Name, ACPI_GET8 (Target));

```

```

192     }
193     break;

195     case ACPI_RSD_UINT16:

197         AcpiRsOutInteger16 (Name, ACPI_GET16 (Target));
198         break;

200     case ACPI_RSD_UINT32:

202         AcpiRsOutInteger32 (Name, ACPI_GET32 (Target));
203         break;

205     case ACPI_RSD_UINT64:

207         AcpiRsOutInteger64 (Name, ACPI_GET64 (Target));
208         break;

210     /* Flags: 1-bit and 2-bit flags supported */

212     case ACPI_RSD_1BITFLAG:

214         AcpiRsOutString (Name, ACPI_CAST_PTR (char,
215             Table->Pointer [*Target & 0x01]));
216         break;

218     case ACPI_RSD_2BITFLAG:

220         AcpiRsOutString (Name, ACPI_CAST_PTR (char,
221             Table->Pointer [*Target & 0x03]));
222         break;

224     case ACPI_RSD_3BITFLAG:

226         AcpiRsOutString (Name, ACPI_CAST_PTR (char,
227             Table->Pointer [*Target & 0x07]));
228         break;

230     case ACPI_RSD_SHORTLIST:
231         /*
232          * Short byte list (single line output) for DMA and IRQ resources
233          * Note: The list length is obtained from the previous table entry
234          */
235         if (PreviousTarget)
236         {
237             AcpiRsOutTitle (Name);
238             AcpiRsDumpShortByteList (*PreviousTarget, Target);
239         }
240         break;

242     case ACPI_RSD_SHORTLISTX:
243         /*
244          * Short byte list (single line output) for GPIO vendor data
245          * Note: The list length is obtained from the previous table entry
246          */
247         if (PreviousTarget)
248         {
249             AcpiRsOutTitle (Name);
250             AcpiRsDumpShortByteList (*PreviousTarget,
251                 *(ACPI_CAST_INDIRECT_PTR (UINT8, Target)));
252         }
253         break;

255     case ACPI_RSD_LONGLIST:
256         /*
257          * Long byte list for Vendor resource data

```

```

258     /* 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, Target))
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:
391
392         AcpiRsOutInteger8 ("Resource Type",
393             (UINT8) Resource->Address.ResourceType);
394         break;
395     }
396
397     /* Decode the general flags */
398
399     AcpiRsDumpDescriptor (Resource, AcpiRsDumpGeneralFlags);
400 }
401
402
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  *****/
414
415 void
416 AcpiRsDumpResourceList (
417     ACPI_RESOURCE             *ResourceList)
418 {
419     UINT32                    Count = 0;
420     UINT32                    Type;
421
422     ACPI_FUNCTION_ENTRY ();
423
424     /* Check if debug output enabled */
425
426     if (!ACPI_IS_DEBUG_ENABLED (ACPI_LV_RESOURCES, _COMPONENT))
427     {
428         return;
429     }
430
431     /* Walk list and dump all resource descriptors (END_TAG terminates) */
432
433     do
434     {
435         AcpiOsPrintf ("\n[%02X] ", Count);
436         Count++;
437
438         /* Validate Type before dispatch */
439
440         Type = ResourceList->Type;
441         if (Type > ACPI_RESOURCE_TYPE_MAX)
442         {
443             AcpiOsPrintf (
444                 "Invalid descriptor type (%X) in resource list\n",
445                 ResourceList->Type);
446             return;
447         }
448
449         /* Sanity check the length. It must not be zero, or we loop forever */
450
451         if (!ResourceList->Length)
452         {
453             AcpiOsPrintf (

```

```

456         "Invalid zero length descriptor in resource list\n");
457         return;
458     }
459
460     /* Dump the resource descriptor */
461
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     }
472
473     /* Point to the next resource structure */
474
475     ResourceList = ACPI_NEXT_RESOURCE (ResourceList);
476
477     /* Exit when END_TAG descriptor is reached */
478
479     } while (Type != ACPI_RESOURCE_TYPE_END_TAG);
480 }
481
482
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  *****/
494
495 void
496 AcpiRsDumpIrqList (
497     UINT8                    *RouteTable)
498 {
499     ACPI_PCI_ROUTING_TABLE    *PrtElement;
500     UINT8                    Count;
501
502     ACPI_FUNCTION_ENTRY ();
503
504     /* Check if debug output enabled */
505
506     if (!ACPI_IS_DEBUG_ENABLED (ACPI_LV_RESOURCES, _COMPONENT))
507     {
508         return;
509     }
510
511     PrtElement = ACPI_CAST_PTR (ACPI_PCI_ROUTING_TABLE, RouteTable);
512
513     /* Dump all table elements, Exit on zero length element */
514
515     for (Count = 0; PrtElement->Length; Count++)
516     {
517         AcpiOsPrintf ("\n[%02X] PCI IRQ Routing Table Package\n", Count);
518         AcpiRsDumpDescriptor (PrtElement, AcpiRsDumpPrt);
519     }
520 }

```



```

522     PrtElement = ACPI_ADD_PTR (ACPI_PCI_ROUTING_TABLE,
523     PrtElement, PrtElement->Length);
524 }
525 }

528 /*****
529 *
530 * FUNCTION:     AcpiRsOut*
531 *
532 * PARAMETERS:  Title      - Name of the resource field
533 *              Value      - Value of the resource field
534 *
535 * RETURN:      None
536 *
537 * DESCRIPTION: Miscellaneous helper functions to consistently format the
538 *              output of the resource dump routines
539 *
540 *****/

542 static void
543 AcpiRsOutString (
544     char          *Title,
545     char          *Value)
546 {
547     AcpiOsPrintf ("%27s : %s", Title, Value);
548     if (!*Value)
549     {
550         AcpiOsPrintf ("[NULL NAMESTRING]");
551     }
552     AcpiOsPrintf ("\n");
553 }

555 static void
556 AcpiRsOutInteger8 (
557     char          *Title,
558     UINT8         Value)
559 {
560     AcpiOsPrintf ("%27s : %2.2X\n", Title, Value);
561 }

563 static void
564 AcpiRsOutInteger16 (
565     char          *Title,
566     UINT16        Value)
567 {
568     AcpiOsPrintf ("%27s : %4.4X\n", Title, Value);
569 }

571 static void
572 AcpiRsOutInteger32 (
573     char          *Title,
574     UINT32        Value)
575 {
576     AcpiOsPrintf ("%27s : %8.8X\n", Title, Value);
577 }

579 static void
580 AcpiRsOutInteger64 (
581     char          *Title,
582     UINT64        Value)
583 {
584     AcpiOsPrintf ("%27s : %8.8X%8.8X\n", Title,
585     ACPI_FORMAT_UINT64 (Value));
586 }

```

```

588 static void
589 AcpiRsOutTitle (
590     char          *Title)
591 {
592     AcpiOsPrintf ("%27s : ", Title);
593 }

596 /*****
597 *
598 * FUNCTION:     AcpiRsDump*List
599 *
600 * PARAMETERS:  Length      - Number of elements in the list
601 *              Data        - Start of the list
602 *
603 * RETURN:      None
604 *
605 * DESCRIPTION: Miscellaneous functions to dump lists of raw data
606 *
607 *****/

609 static void
610 AcpiRsDumpByteList (
611     UINT16        Length,
612     UINT8         *Data)
613 {
614     UINT8         i;

617     for (i = 0; i < Length; i++)
618     {
619         AcpiOsPrintf ("%25s%2.2X : %2.2X\n",
620             "Byte", i, Data[i]);
621     }
622 }

624 static void
625 AcpiRsDumpShortByteList (
626     UINT8         Length,
627     UINT8         *Data)
628 {
629     UINT8         i;

632     for (i = 0; i < Length; i++)
633     {
634         AcpiOsPrintf ("%X ", Data[i]);
635     }
636     AcpiOsPrintf ("\n");
637 }

639 static void
640 AcpiRsDumpDwordList (
641     UINT8         Length,
642     UINT32        *Data)
643 {
644     UINT8         i;

647     for (i = 0; i < Length; i++)
648     {
649         AcpiOsPrintf ("%25s%2.2X : %8.8X\n",
650             "Dword", i, Data[i]);
651     }
652 }

```

```
654 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 Fri Jan 17 07:30:36 2014
new/usr/src/common/acpica/components/resources/rsdumpinfo.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: rsdumpinfo - Tables used to display resource descriptors.
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __RSDUMPINFO_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acresrc.h"

51 #define _COMPONENT          ACPI_RESOURCES
52 #define ACPI_MODULE_NAME    ("rsdumpinfo")

55 #if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)

58 #define ACPI_RSD_OFFSET(f)      (UINT8) ACPI_OFFSET (ACPI_RESOURCE_DATA,f)
59 #define ACPI_PRT_OFFSET(f)     (UINT8) ACPI_OFFSET (ACPI_PCI_ROUTING_TABLE,
60 #define ACPI_RSD_TABLE_SIZE(name) (sizeof(name) / sizeof (ACPI_RSDUMP_INFO))

```

```

63 /*****
64 *
65 * Resource Descriptor info tables
66 *
67 * Note: The first table entry must be a Title or Literal and must contain
68 * the table length (number of table entries)
69 *
70 *****/

72 ACPI_RSDUMP_INFO          AcpiRsDumpIrq[7] =
73 {
74     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpIrq),          "IRQ"
75     {ACPI_RSD_UINT8 ,    ACPI_RSD_OFFSET (Irq.DescriptorLength),        "Des
76     {ACPI_RSD_1BITFLAG,  ACPI_RSD_OFFSET (Irq.Triggering),              "Tri
77     {ACPI_RSD_1BITFLAG,  ACPI_RSD_OFFSET (Irq.Polarity),                "Pol
78     {ACPI_RSD_2BITFLAG,  ACPI_RSD_OFFSET (Irq.Sharable),                "Sha
79     {ACPI_RSD_UINT8 ,    ACPI_RSD_OFFSET (Irq.InterruptCount),          "Int
80     {ACPI_RSD_SHORTLIST, ACPI_RSD_OFFSET (Irq.Interrupts[0]),          "Int
81 };

83 ACPI_RSDUMP_INFO          AcpiRsDumpDma[6] =
84 {
85     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpDma),          "DMA
86     {ACPI_RSD_2BITFLAG,  ACPI_RSD_OFFSET (Dma.Type),                    "Spe
87     {ACPI_RSD_1BITFLAG,  ACPI_RSD_OFFSET (Dma.BusMaster),                "Mas
88     {ACPI_RSD_2BITFLAG,  ACPI_RSD_OFFSET (Dma.Transfer),                "Tra
89     {ACPI_RSD_UINT8 ,    ACPI_RSD_OFFSET (Dma.ChannelCount),            "Cha
90     {ACPI_RSD_SHORTLIST, ACPI_RSD_OFFSET (Dma.Channels[0]),            "Cha
91 };

93 ACPI_RSDUMP_INFO          AcpiRsDumpStartDpf[4] =
94 {
95     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpStartDpf),      "Sta
96     {ACPI_RSD_UINT8 ,    ACPI_RSD_OFFSET (StartDpf.DescriptorLength),    "Des
97     {ACPI_RSD_2BITFLAG,  ACPI_RSD_OFFSET (StartDpf.CompatibilityPriority), "Com
98     {ACPI_RSD_2BITFLAG,  ACPI_RSD_OFFSET (StartDpf.PerformanceRobustness), "Per
99 };

101 ACPI_RSDUMP_INFO          AcpiRsDumpEndDpf[1] =
102 {
103     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpEndDpf),        "End
104 };

106 ACPI_RSDUMP_INFO          AcpiRsDumpIo[6] =
107 {
108     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpIo),            "I/O
109     {ACPI_RSD_1BITFLAG,  ACPI_RSD_OFFSET (Io.IoDecode),                  "Add
110     {ACPI_RSD_UINT16,    ACPI_RSD_OFFSET (Io.Minimum),                   "Add
111     {ACPI_RSD_UINT16,    ACPI_RSD_OFFSET (Io.Maximum),                   "Add
112     {ACPI_RSD_UINT8 ,    ACPI_RSD_OFFSET (Io.Alignment),                 "Ali
113     {ACPI_RSD_UINT8 ,    ACPI_RSD_OFFSET (Io.AddressLength),             "Add
114 };

116 ACPI_RSDUMP_INFO          AcpiRsDumpFixedIo[3] =
117 {
118     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpFixedIo),        "Fix
119     {ACPI_RSD_UINT16,    ACPI_RSD_OFFSET (FixedIo.Address),               "Add
120     {ACPI_RSD_UINT8 ,    ACPI_RSD_OFFSET (FixedIo.AddressLength),         "Add
121 };

123 ACPI_RSDUMP_INFO          AcpiRsDumpVendor[3] =
124 {
125     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpVendor),        "Ven
126     {ACPI_RSD_UINT16,    ACPI_RSD_OFFSET (Vendor.ByteLength),             "Len

```

```

127     {ACPI_RSD_LONGLIST, ACPI_RSD_OFFSET (Vendor.ByteData[0]), "Ven
128 };

130 ACPI_RSDUMP_INFO      AcpiRsDumpEndTag[1] =
131 {
132     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpEndTag), "End
133 };

135 ACPI_RSDUMP_INFO      AcpiRsDumpMemory24[6] =
136 {
137     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpMemory24), "24-
138     {ACPI_RSD_1BITFLAG,  ACPI_RSD_OFFSET (Memory24.WriteProtect), "Wri
139     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (Memory24.Minimum), "Add
140     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (Memory24.Maximum), "Add
141     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (Memory24.Alignment), "Ali
142     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (Memory24.AddressLength), "Add
143 };

145 ACPI_RSDUMP_INFO      AcpiRsDumpMemory32[6] =
146 {
147     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpMemory32), "32-
148     {ACPI_RSD_1BITFLAG,  ACPI_RSD_OFFSET (Memory32.WriteProtect), "Wri
149     {ACPI_RSD_UINT32,     ACPI_RSD_OFFSET (Memory32.Minimum), "Add
150     {ACPI_RSD_UINT32,     ACPI_RSD_OFFSET (Memory32.Maximum), "Add
151     {ACPI_RSD_UINT32,     ACPI_RSD_OFFSET (Memory32.Alignment), "Ali
152     {ACPI_RSD_UINT32,     ACPI_RSD_OFFSET (Memory32.AddressLength), "Add
153 };

155 ACPI_RSDUMP_INFO      AcpiRsDumpFixedMemory32[4] =
156 {
157     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpFixedMemory32), "32-
158     {ACPI_RSD_1BITFLAG,  ACPI_RSD_OFFSET (FixedMemory32.WriteProtect), "Wri
159     {ACPI_RSD_UINT32,     ACPI_RSD_OFFSET (FixedMemory32.Address), "Add
160     {ACPI_RSD_UINT32,     ACPI_RSD_OFFSET (FixedMemory32.AddressLength), "Add
161 };

163 ACPI_RSDUMP_INFO      AcpiRsDumpAddress16[8] =
164 {
165     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpAddress16), "16-
166     {ACPI_RSD_ADDRESS,    0, NULL
167     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (Address16.Granularity), "Gra
168     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (Address16.Minimum), "Add
169     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (Address16.Maximum), "Add
170     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (Address16.TranslationOffset), "Tra
171     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (Address16.AddressLength), "Add
172     {ACPI_RSD_SOURCE,     ACPI_RSD_OFFSET (Address16.ResourceSource), NULL
173 };

175 ACPI_RSDUMP_INFO      AcpiRsDumpAddress32[8] =
176 {
177     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpAddress32), "32-B
178     {ACPI_RSD_ADDRESS,    0, NULL
179     {ACPI_RSD_UINT32,     ACPI_RSD_OFFSET (Address32.Granularity), "Gra
180     {ACPI_RSD_UINT32,     ACPI_RSD_OFFSET (Address32.Minimum), "Add
181     {ACPI_RSD_UINT32,     ACPI_RSD_OFFSET (Address32.Maximum), "Add
182     {ACPI_RSD_UINT32,     ACPI_RSD_OFFSET (Address32.TranslationOffset), "Tra
183     {ACPI_RSD_UINT32,     ACPI_RSD_OFFSET (Address32.AddressLength), "Add
184     {ACPI_RSD_SOURCE,     ACPI_RSD_OFFSET (Address32.ResourceSource), NULL
185 };

187 ACPI_RSDUMP_INFO      AcpiRsDumpAddress64[8] =
188 {
189     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpAddress64), "64-
190     {ACPI_RSD_ADDRESS,    0, NULL
191     {ACPI_RSD_UINT64,     ACPI_RSD_OFFSET (Address64.Granularity), "Gra
192     {ACPI_RSD_UINT64,     ACPI_RSD_OFFSET (Address64.Minimum), "Add

```

```

193     {ACPI_RSD_UINT64,     ACPI_RSD_OFFSET (Address64.Maximum), "Add
194     {ACPI_RSD_UINT64,     ACPI_RSD_OFFSET (Address64.TranslationOffset), "Tra
195     {ACPI_RSD_UINT64,     ACPI_RSD_OFFSET (Address64.AddressLength), "Add
196     {ACPI_RSD_SOURCE,     ACPI_RSD_OFFSET (Address64.ResourceSource), NULL
197 };

199 ACPI_RSDUMP_INFO      AcpiRsDumpExtAddress64[8] =
200 {
201     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpExtAddress64), "64-
202     {ACPI_RSD_ADDRESS,    0, NULL
203     {ACPI_RSD_UINT64,     ACPI_RSD_OFFSET (ExtAddress64.Granularity), "Gra
204     {ACPI_RSD_UINT64,     ACPI_RSD_OFFSET (ExtAddress64.Minimum), "Add
205     {ACPI_RSD_UINT64,     ACPI_RSD_OFFSET (ExtAddress64.Maximum), "Add
206     {ACPI_RSD_UINT64,     ACPI_RSD_OFFSET (ExtAddress64.TranslationOffset), "Tra
207     {ACPI_RSD_UINT64,     ACPI_RSD_OFFSET (ExtAddress64.AddressLength), "Add
208     {ACPI_RSD_UINT64,     ACPI_RSD_OFFSET (ExtAddress64.TypeSpecific), "Typ
209 };

211 ACPI_RSDUMP_INFO      AcpiRsDumpExtIrq[8] =
212 {
213     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpExtIrq), "Ext
214     {ACPI_RSD_1BITFLAG,  ACPI_RSD_OFFSET (ExtendedIrq.ProducerConsumer), "Typ
215     {ACPI_RSD_1BITFLAG,  ACPI_RSD_OFFSET (ExtendedIrq.Triggering), "Tri
216     {ACPI_RSD_1BITFLAG,  ACPI_RSD_OFFSET (ExtendedIrq.Polarity), "Pol
217     {ACPI_RSD_2BITFLAG,  ACPI_RSD_OFFSET (ExtendedIrq.Sharable), "Sha
218     {ACPI_RSD_SOURCE,     ACPI_RSD_OFFSET (ExtendedIrq.ResourceSource), NULL
219     {ACPI_RSD_UINT8,      ACPI_RSD_OFFSET (ExtendedIrq.InterruptCount), "Int
220     {ACPI_RSD_DWORDLIST, ACPI_RSD_OFFSET (ExtendedIrq.Interrupts[0]), "Int
221 };

223 ACPI_RSDUMP_INFO      AcpiRsDumpGenericReg[6] =
224 {
225     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpGenericReg), "Gen
226     {ACPI_RSD_UINT8,      ACPI_RSD_OFFSET (GenericReg.SpaceId), "Spa
227     {ACPI_RSD_UINT8,      ACPI_RSD_OFFSET (GenericReg.BitWidth), "Bit
228     {ACPI_RSD_UINT8,      ACPI_RSD_OFFSET (GenericReg.BitOffset), "Bit
229     {ACPI_RSD_UINT8,      ACPI_RSD_OFFSET (GenericReg.AccessSize), "Acc
230     {ACPI_RSD_UINT64,     ACPI_RSD_OFFSET (GenericReg.Address), "Add
231 };

233 ACPI_RSDUMP_INFO      AcpiRsDumpGpio[16] =
234 {
235     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpGpio), "GPI
236     {ACPI_RSD_UINT8,      ACPI_RSD_OFFSET (Gpio.RevisionId), "Rev
237     {ACPI_RSD_UINT8,      ACPI_RSD_OFFSET (Gpio.ConnectionType), "Con
238     {ACPI_RSD_1BITFLAG,  ACPI_RSD_OFFSET (Gpio.ProducerConsumer), "Pro
239     {ACPI_RSD_UINT8,      ACPI_RSD_OFFSET (Gpio.PinConfig), "Pin
240     {ACPI_RSD_2BITFLAG,  ACPI_RSD_OFFSET (Gpio.Sharable), "Sha
241     {ACPI_RSD_2BITFLAG,  ACPI_RSD_OFFSET (Gpio.IoRestriction), "IoR
242     {ACPI_RSD_1BITFLAG,  ACPI_RSD_OFFSET (Gpio.Triggering), "Tri
243     {ACPI_RSD_2BITFLAG,  ACPI_RSD_OFFSET (Gpio.Polarity), "Pol
244     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (Gpio.DriveStrength), "Dri
245     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (Gpio.DebounceTimeout), "Deb
246     {ACPI_RSD_SOURCE,     ACPI_RSD_OFFSET (Gpio.ResourceSource), "Res
247     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (Gpio.PinTableLength), "Pin
248     {ACPI_RSD_WORDLIST,  ACPI_RSD_OFFSET (Gpio.PinTable), "Pin
249     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (Gpio.VendorLength), "Ven
250     {ACPI_RSD_SHORTLISTX, ACPI_RSD_OFFSET (Gpio.VendorData), "Ven
251 };

253 ACPI_RSDUMP_INFO      AcpiRsDumpFixedDma[4] =
254 {
255     {ACPI_RSD_TITLE,      ACPI_RSD_TABLE_SIZE (AcpiRsDumpFixedDma), "Fix
256     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (FixedDma.RequestLines), "Req
257     {ACPI_RSD_UINT16,     ACPI_RSD_OFFSET (FixedDma.Channels), "Cha
258     {ACPI_RSD_UINT8,      ACPI_RSD_OFFSET (FixedDma.Width), "Tra

```

```

259 };

261 #define ACPI_RS_DUMP_COMMON_SERIAL_BUS \
262 {ACPI_RSD_UINT8, ACPI_RSD_OFFSET (CommonSerialBus.RevisionId), "Rev"
263 {ACPI_RSD_UINT8, ACPI_RSD_OFFSET (CommonSerialBus.Type), "Typ"
264 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (CommonSerialBus.ProducerConsumer), "Pro"
265 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (CommonSerialBus.SlaveMode), "Sla"
266 {ACPI_RSD_UINT8, ACPI_RSD_OFFSET (CommonSerialBus.TypeRevisionId), "Typ"
267 {ACPI_RSD_UINT16, ACPI_RSD_OFFSET (CommonSerialBus.TypeDataLength), "Typ"
268 {ACPI_RSD_SOURCE, ACPI_RSD_OFFSET (CommonSerialBus.ResourceSource), "Res"
269 {ACPI_RSD_UINT16, ACPI_RSD_OFFSET (CommonSerialBus.VendorLength), "Ven"
270 {ACPI_RSD_SHORTLISTX, ACPI_RSD_OFFSET (CommonSerialBus.VendorData), "Ven"

272 ACPI_RS_DUMP_INFO AcpiRsDumpCommonSerialBus[10] =
273 {
274 {ACPI_RSD_TITLE, ACPI_RSD_TABLE_SIZE (AcpiRsDumpCommonSerialBus), "Com
275 ACPI_RS_DUMP_COMMON_SERIAL_BUS
276 };

278 ACPI_RS_DUMP_INFO AcpiRsDumpI2cSerialBus[13] =
279 {
280 {ACPI_RSD_TITLE, ACPI_RSD_TABLE_SIZE (AcpiRsDumpI2cSerialBus), "I2C
281 ACPI_RS_DUMP_COMMON_SERIAL_BUS
282 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (I2cSerialBus.AccessMode), "Acc
283 {ACPI_RSD_UINT32, ACPI_RSD_OFFSET (I2cSerialBus.ConnectionSpeed), "Con
284 {ACPI_RSD_UINT16, ACPI_RSD_OFFSET (I2cSerialBus.SlaveAddress), "Sla
285 };

287 ACPI_RS_DUMP_INFO AcpiRsDumpSpiSerialBus[17] =
288 {
289 {ACPI_RSD_TITLE, ACPI_RSD_TABLE_SIZE (AcpiRsDumpSpiSerialBus), "Spi
290 ACPI_RS_DUMP_COMMON_SERIAL_BUS
291 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (SpiSerialBus.WireMode), "Wir
292 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (SpiSerialBus.DevicePolarity), "Dev
293 {ACPI_RSD_UINT8, ACPI_RSD_OFFSET (SpiSerialBus.DataBitLength), "Dat
294 {ACPI_RSD_UINT8, ACPI_RSD_OFFSET (SpiSerialBus.ClockPhase), "Clo
295 {ACPI_RSD_UINT8, ACPI_RSD_OFFSET (SpiSerialBus.ClockPolarity), "Clo
296 {ACPI_RSD_UINT16, ACPI_RSD_OFFSET (SpiSerialBus.DevicesSelection), "Dev
297 {ACPI_RSD_UINT32, ACPI_RSD_OFFSET (SpiSerialBus.ConnectionSpeed), "Con
298 };

300 ACPI_RS_DUMP_INFO AcpiRsDumpUartSerialBus[19] =
301 {
302 {ACPI_RSD_TITLE, ACPI_RSD_TABLE_SIZE (AcpiRsDumpUartSerialBus), "Ua
303 ACPI_RS_DUMP_COMMON_SERIAL_BUS
304 {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (UartSerialBus.FlowControl), "Fl
305 {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (UartSerialBus.StopBits), "St
306 {ACPI_RSD_3BITFLAG, ACPI_RSD_OFFSET (UartSerialBus.DataBits), "Da
307 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (UartSerialBus.Endian), "En
308 {ACPI_RSD_UINT8, ACPI_RSD_OFFSET (UartSerialBus.Parity), "Pa
309 {ACPI_RSD_UINT8, ACPI_RSD_OFFSET (UartSerialBus.LinesEnabled), "Li
310 {ACPI_RSD_UINT16, ACPI_RSD_OFFSET (UartSerialBus.RxFifoSize), "Rx
311 {ACPI_RSD_UINT16, ACPI_RSD_OFFSET (UartSerialBus.TxFifoSize), "Tx
312 {ACPI_RSD_UINT32, ACPI_RSD_OFFSET (UartSerialBus.DefaultBaudRate), "Co
313 };

315 /*
316 * Tables used for common address descriptor flag fields
317 */
318 ACPI_RS_DUMP_INFO AcpiRsDumpGeneralFlags[5] =
319 {
320 {ACPI_RSD_TITLE, ACPI_RSD_TABLE_SIZE (AcpiRsDumpGeneralFlags), NULL
321 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.ProducerConsumer), "Con
322 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.Decode), "Add
323 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.MinAddressFixed), "Min
324 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.MaxAddressFixed), "Max

```

```

325 };

327 ACPI_RS_DUMP_INFO AcpiRsDumpMemoryFlags[5] =
328 {
329 {ACPI_RSD_LITERAL, ACPI_RSD_TABLE_SIZE (AcpiRsDumpMemoryFlags), "Res
330 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.Info.Mem.WriteProtect), "Wri
331 {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (Address.Info.Mem.Caching), "Cac
332 {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (Address.Info.Mem.RangeType), "Ran
333 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.Info.Mem.Translation), "Tra
334 };

336 ACPI_RS_DUMP_INFO AcpiRsDumpIoFlags[4] =
337 {
338 {ACPI_RSD_LITERAL, ACPI_RSD_TABLE_SIZE (AcpiRsDumpIoFlags), "Res
339 {ACPI_RSD_2BITFLAG, ACPI_RSD_OFFSET (Address.Info.Io.RangeType), "Ran
340 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.Info.Io.Translation), "Tra
341 {ACPI_RSD_1BITFLAG, ACPI_RSD_OFFSET (Address.Info.Io.TranslationType), "Tra
342 };

345 /*
346 * Table used to dump _PRT contents
347 */
348 ACPI_RS_DUMP_INFO AcpiRsDumpPrt[5] =
349 {
350 {ACPI_RSD_TITLE, ACPI_RSD_TABLE_SIZE (AcpiRsDumpPrt), NULL
351 {ACPI_RSD_UINT64, ACPI_RSD_OFFSET (Address), "Add
352 {ACPI_RSD_UINT32, ACPI_RSD_OFFSET (Pin), "Pin
353 {ACPI_RSD_STRING, ACPI_RSD_OFFSET (Source[0]), "Sou
354 {ACPI_RSD_UINT32, ACPI_RSD_OFFSET (SourceIndex), "Sou
355 };

357 #endif

```

new/usr/src/common/acpica/components/resources/rsinfo.c

1

```
*****
12406 Fri Jan 17 07:30:36 2014
new/usr/src/common/acpica/components/resources/rsinfo.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: rsinfo - Dispatch and Info tables
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __RSINFO_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acresrc.h"

50 #define _COMPONENT          ACPI_RESOURCES
51 #define ACPI_MODULE_NAME    ("rsinfo")

53 /*
54 * Resource dispatch and information tables. Any new resource types (either
55 * Large or Small) must be reflected in each of these tables, so they are here
56 * in one place.
57 *
58 * The tables for Large descriptors are indexed by bits 6:0 of the AML
```

new/usr/src/common/acpica/components/resources/rsinfo.c

2

```
59 * descriptor type byte. The tables for Small descriptors are indexed by
60 * bits 6:3 of the descriptor byte. The tables for internal resource
61 * descriptors are indexed by the ACPI_RESOURCE_TYPE field.
62 */

65 /* Dispatch table for resource-to-AML (Set Resource) conversion functions */

67 ACPI_RS_CONVERT_INFO          *AcpiGbl_SetResourceDispatch[] =
68 {
69     AcpiRsSetIrq,              /* 0x00, ACPI_RESOURCE_TYPE_IRQ */
70     AcpiRsConvertDma,          /* 0x01, ACPI_RESOURCE_TYPE_DMA */
71     AcpiRsSetStartDpf,        /* 0x02, ACPI_RESOURCE_TYPE_START_DEPENDENT */
72     AcpiRsConvertEndDpf,      /* 0x03, ACPI_RESOURCE_TYPE_END_DEPENDENT */
73     AcpiRsConvertIo,          /* 0x04, ACPI_RESOURCE_TYPE_IO */
74     AcpiRsConvertFixedIo,     /* 0x05, ACPI_RESOURCE_TYPE_FIXED_IO */
75     AcpiRsSetVendor,         /* 0x06, ACPI_RESOURCE_TYPE_VENDOR */
76     AcpiRsConvertEndTag,      /* 0x07, ACPI_RESOURCE_TYPE_END_TAG */
77     AcpiRsConvertMemory24,    /* 0x08, ACPI_RESOURCE_TYPE_MEMORY24 */
78     AcpiRsConvertMemory32,    /* 0x09, ACPI_RESOURCE_TYPE_MEMORY32 */
79     AcpiRsConvertFixedMemory32, /* 0x0A, ACPI_RESOURCE_TYPE_FIXED_MEMORY32 */
80     AcpiRsConvertAddress16,   /* 0x0B, ACPI_RESOURCE_TYPE_ADDRESS16 */
81     AcpiRsConvertAddress32,   /* 0x0C, ACPI_RESOURCE_TYPE_ADDRESS32 */
82     AcpiRsConvertAddress64,   /* 0x0D, ACPI_RESOURCE_TYPE_ADDRESS64 */
83     AcpiRsConvertExtAddress64, /* 0x0E, ACPI_RESOURCE_TYPE_EXTENDED_ADDRESS */
84     AcpiRsConvertExtIrq,     /* 0x0F, ACPI_RESOURCE_TYPE_EXTENDED_IRQ */
85     AcpiRsConvertGenericReg, /* 0x10, ACPI_RESOURCE_TYPE_GENERIC_REGISTER */
86     AcpiRsConvertGpio,       /* 0x11, ACPI_RESOURCE_TYPE_GPIO */
87     AcpiRsConvertFixedDma,    /* 0x12, ACPI_RESOURCE_TYPE_FIXED_DMA */
88     NULL,                    /* 0x13, ACPI_RESOURCE_TYPE_SERIAL_BUS - Use
89     AcpiRsConvertGenericReg   /* 0x10, ACPI_RESOURCE_TYPE_GENERIC_REGISTER
90 };

91 /* Dispatch tables for AML-to-resource (Get Resource) conversion functions */

93 ACPI_RS_CONVERT_INFO          *AcpiGbl_GetResourceDispatch[] =
94 {
95     /* Small descriptors */
96
97     NULL,                    /* 0x00, Reserved */
98     NULL,                    /* 0x01, Reserved */
99     NULL,                    /* 0x02, Reserved */
100    NULL,                    /* 0x03, Reserved */
101    AcpiRsGetIrq,             /* 0x04, ACPI_RESOURCE_NAME_IRQ */
102    AcpiRsConvertDma,         /* 0x05, ACPI_RESOURCE_NAME_DMA */
103    AcpiRsGetStartDpf,        /* 0x06, ACPI_RESOURCE_NAME_START_DEPENDENT */
104    AcpiRsConvertEndDpf,      /* 0x07, ACPI_RESOURCE_NAME_END_DEPENDENT */
105    AcpiRsConvertIo,          /* 0x08, ACPI_RESOURCE_NAME_IO */
106    AcpiRsConvertFixedIo,     /* 0x09, ACPI_RESOURCE_NAME_FIXED_IO */
107    AcpiRsConvertFixedDma,    /* 0x0A, ACPI_RESOURCE_NAME_FIXED_DMA */
108    NULL,                    /* 0x0A, Reserved */
109    NULL,                    /* 0x0B, Reserved */
110    NULL,                    /* 0x0C, Reserved */
111    NULL,                    /* 0x0D, Reserved */
112    AcpiRsGetVendorSmall,     /* 0x0E, ACPI_RESOURCE_NAME_VENDOR_SMALL */
113    AcpiRsConvertEndTag,      /* 0x0F, ACPI_RESOURCE_NAME_END_TAG */

114    /* Large descriptors */

116    NULL,                    /* 0x00, Reserved */
117    AcpiRsConvertMemory24,    /* 0x01, ACPI_RESOURCE_NAME_MEMORY24 */
118    AcpiRsConvertGenericReg, /* 0x02, ACPI_RESOURCE_NAME_GENERIC_REGISTER */
119    NULL,                    /* 0x03, Reserved */
120    AcpiRsGetVendorLarge,     /* 0x04, ACPI_RESOURCE_NAME_VENDOR_LARGE */
121    AcpiRsConvertMemory32,    /* 0x05, ACPI_RESOURCE_NAME_MEMORY32 */
122    AcpiRsConvertFixedMemory32, /* 0x06, ACPI_RESOURCE_NAME_FIXED_MEMORY32 */
123 }
```

```

123 AcpiRsConvertAddress32, /* 0x07, ACPI_RESOURCE_NAME_ADDRESS32 */
124 AcpiRsConvertAddress16, /* 0x08, ACPI_RESOURCE_NAME_ADDRESS16 */
125 AcpiRsConvertExtIrq, /* 0x09, ACPI_RESOURCE_NAME_EXTENDED_IRQ */
126 AcpiRsConvertAddress64, /* 0x0A, ACPI_RESOURCE_NAME_ADDRESS64 */
127 AcpiRsConvertExtAddress64, /* 0x0B, ACPI_RESOURCE_NAME_EXTENDED_ADDRESS
128 AcpiRsConvertGpio, /* 0x0C, ACPI_RESOURCE_NAME_GPIO */
129 NULL, /* 0x0D, Reserved */
130 NULL, /* 0x0E, ACPI_RESOURCE_NAME_SERIAL_BUS - Use
124 AcpiRsConvertExtAddress64 /* 0x0B, ACPI_RESOURCE_NAME_EXTENDED_ADDRESS
131 };

```

```

133 /* Subtype table for SerialBus -- I2C, SPI, and UART */

```

```

135 ACPI_RS_CONVERT_INFO *AcpiGbl_ConvertResourceSerialBusDispatch[] =
136 {
137     NULL,
138     AcpiRsConvertI2cSerialBus,
139     AcpiRsConvertSpiSerialBus,
140     AcpiRsConvertUartSerialBus,
141 };

```

```

144 #if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)

```

```

146 /* Dispatch table for resource dump functions */

```

```

148 ACPI_RS_DUMP_INFO *AcpiGbl_DumpResourceDispatch[] =
149 {
150     AcpiRsDumpIrq, /* ACPI_RESOURCE_TYPE_IRQ */
151     AcpiRsDumpDma, /* ACPI_RESOURCE_TYPE_DMA */
152     AcpiRsDumpStartDpf, /* ACPI_RESOURCE_TYPE_START_DEPENDENT */
153     AcpiRsDumpEndDpf, /* ACPI_RESOURCE_TYPE_END_DEPENDENT */
154     AcpiRsDumpIo, /* ACPI_RESOURCE_TYPE_IO */
155     AcpiRsDumpFixedIo, /* ACPI_RESOURCE_TYPE_FIXED_IO */
156     AcpiRsDumpVendor, /* ACPI_RESOURCE_TYPE_VENDOR */
157     AcpiRsDumpEndTag, /* ACPI_RESOURCE_TYPE_END_TAG */
158     AcpiRsDumpMemory24, /* ACPI_RESOURCE_TYPE_MEMORY24 */
159     AcpiRsDumpMemory32, /* ACPI_RESOURCE_TYPE_MEMORY32 */
160     AcpiRsDumpFixedMemory32, /* ACPI_RESOURCE_TYPE_FIXED_MEMORY32 */
161     AcpiRsDumpAddress16, /* ACPI_RESOURCE_TYPE_ADDRESS16 */
162     AcpiRsDumpAddress32, /* ACPI_RESOURCE_TYPE_ADDRESS32 */
163     AcpiRsDumpAddress64, /* ACPI_RESOURCE_TYPE_ADDRESS64 */
164     AcpiRsDumpExtAddress64, /* ACPI_RESOURCE_TYPE_EXTENDED_ADDRESS64 */
165     AcpiRsDumpExtIrq, /* ACPI_RESOURCE_TYPE_EXTENDED_IRQ */
166     AcpiRsDumpGenericReg, /* ACPI_RESOURCE_TYPE_GENERIC_REGISTER */
167     AcpiRsDumpGpio, /* ACPI_RESOURCE_TYPE_GPIO */
168     AcpiRsDumpFixedDma, /* ACPI_RESOURCE_TYPE_FIXED_DMA */
169     NULL, /* ACPI_RESOURCE_TYPE_SERIAL_BUS */
170 };

```

```

172 ACPI_RS_DUMP_INFO *AcpiGbl_DumpSerialBusDispatch[] =
173 {
174     NULL,
175     AcpiRsDumpI2cSerialBus, /* AML_RESOURCE_I2C_BUS_TYPE */
176     AcpiRsDumpSpiSerialBus, /* AML_RESOURCE_SPI_BUS_TYPE */
177     AcpiRsDumpUartSerialBus, /* AML_RESOURCE_UART_BUS_TYPE */
178 };
179 #endif

```

```

182 /*
183  * Base sizes for external AML resource descriptors, indexed by internal type.
184  * Includes size of the descriptor header (1 byte for small descriptors,
185  * 3 bytes for large descriptors)
186  */
187 const UINT8 AcpiGbl_AmlResourceSizes[] =

```

```

188 {
189     sizeof (AML_RESOURCE_IRQ), /* ACPI_RESOURCE_TYPE_IRQ (optional
190     sizeof (AML_RESOURCE_DMA), /* ACPI_RESOURCE_TYPE_DMA */
191     sizeof (AML_RESOURCE_START_DEPENDENT), /* ACPI_RESOURCE_TYPE_START_DEPENDEN
192     sizeof (AML_RESOURCE_END_DEPENDENT), /* ACPI_RESOURCE_TYPE_END_DEPENDENT
193     sizeof (AML_RESOURCE_IO), /* ACPI_RESOURCE_TYPE_IO */
194     sizeof (AML_RESOURCE_FIXED_IO), /* ACPI_RESOURCE_TYPE_FIXED_IO */
195     sizeof (AML_RESOURCE_VENDOR_SMALL), /* ACPI_RESOURCE_TYPE_VENDOR */
196     sizeof (AML_RESOURCE_END_TAG), /* ACPI_RESOURCE_TYPE_END_TAG */
197     sizeof (AML_RESOURCE_MEMORY24), /* ACPI_RESOURCE_TYPE_MEMORY24 */
198     sizeof (AML_RESOURCE_MEMORY32), /* ACPI_RESOURCE_TYPE_MEMORY32 */
199     sizeof (AML_RESOURCE_FIXED_MEMORY32), /* ACPI_RESOURCE_TYPE_FIXED_MEMORY32
200     sizeof (AML_RESOURCE_ADDRESS16), /* ACPI_RESOURCE_TYPE_ADDRESS16 */
201     sizeof (AML_RESOURCE_ADDRESS32), /* ACPI_RESOURCE_TYPE_ADDRESS32 */
202     sizeof (AML_RESOURCE_ADDRESS64), /* ACPI_RESOURCE_TYPE_ADDRESS64 */
203     sizeof (AML_RESOURCE_EXTENDED_ADDRESS64), /* ACPI_RESOURCE_TYPE_EXTENDED_ADDRE
204     sizeof (AML_RESOURCE_EXTENDED_IRQ), /* ACPI_RESOURCE_TYPE_EXTENDED_IRQ */
205     sizeof (AML_RESOURCE_GENERIC_REGISTER), /* ACPI_RESOURCE_TYPE_GENERIC_REGIST
206     sizeof (AML_RESOURCE_GPIO), /* ACPI_RESOURCE_TYPE_GPIO */
207     sizeof (AML_RESOURCE_FIXED_DMA), /* ACPI_RESOURCE_TYPE_FIXED_DMA */
208     sizeof (AML_RESOURCE_COMMON_SERIALBUS), /* ACPI_RESOURCE_TYPE_SERIAL_BUS */
178     sizeof (AML_RESOURCE_GENERIC_REGISTER) /* ACPI_RESOURCE_TYPE_GENERIC_REGIST
209 };

```

```

212 const UINT8 AcpiGbl_ResourceStructSizes[] =
213 {
214     /* Small descriptors */

```

```

216 0,
217 0,
218 0,
219 0,
220 ACPI_RS_SIZE (ACPI_RESOURCE_IRQ),
221 ACPI_RS_SIZE (ACPI_RESOURCE_DMA),
222 ACPI_RS_SIZE (ACPI_RESOURCE_START_DEPENDENT),
223 ACPI_RS_SIZE_MIN,
224 ACPI_RS_SIZE (ACPI_RESOURCE_IO),
225 ACPI_RS_SIZE (ACPI_RESOURCE_FIXED_IO),
226 ACPI_RS_SIZE (ACPI_RESOURCE_FIXED_DMA),
227 0,
228 0,
229 0,
230 0,
230 ACPI_RS_SIZE (ACPI_RESOURCE_VENDOR),
231 ACPI_RS_SIZE_MIN,

```

```

233 /* Large descriptors */

```

```

235 0,
236 ACPI_RS_SIZE (ACPI_RESOURCE_MEMORY24),
237 ACPI_RS_SIZE (ACPI_RESOURCE_GENERIC_REGISTER),
238 0,
239 ACPI_RS_SIZE (ACPI_RESOURCE_VENDOR),
240 ACPI_RS_SIZE (ACPI_RESOURCE_MEMORY32),
241 ACPI_RS_SIZE (ACPI_RESOURCE_FIXED_MEMORY32),
242 ACPI_RS_SIZE (ACPI_RESOURCE_ADDRESS32),
243 ACPI_RS_SIZE (ACPI_RESOURCE_ADDRESS16),
244 ACPI_RS_SIZE (ACPI_RESOURCE_EXTENDED_IRQ),
245 ACPI_RS_SIZE (ACPI_RESOURCE_ADDRESS64),
246 ACPI_RS_SIZE (ACPI_RESOURCE_EXTENDED_ADDRESS64),
247 ACPI_RS_SIZE (ACPI_RESOURCE_GPIO),
248 ACPI_RS_SIZE (ACPI_RESOURCE_COMMON_SERIALBUS),
216 ACPI_RS_SIZE (ACPI_RESOURCE_EXTENDED_ADDRESS64)
249 };

```

```
251 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 };
```



new/usr/src/common/acpica/components/resources/rsio.c

1

```
*****
10605 Fri Jan 17 07:30:36 2014
new/usr/src/common/acpica/components/resources/rsio.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: rsio - IO and DMA resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __RSIO_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acresrc.h"

50 #define _COMPONENT          ACPI_RESOURCES
51 #define ACPI_MODULE_NAME    ("rsio")

54 /*****
55 *
56 * AcpiRsConvertIo
57 *
58 *****/
```

new/usr/src/common/acpica/components/resources/rsio.c

2

```
60 ACPI_RS_CONVERT_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 Fri Jan 17 07:30:37 2014
new/usr/src/common/acpica/components/resources/rsirq.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: rsirq - IRQ resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __RSIRQ_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acresrc.h"

50 #define _COMPONENT          ACPI_RESOURCES
51 #define ACPI_MODULE_NAME    ("rsirq")

54 /*****
55 *
56 * AcpiRsGetIrq
57 *
58 *****/

```

```

60 ACPI_RS_CONVERT_INFO      AcpiRsGetIrq[9] =
60 ACPI_RS_CONVERT_INFO      AcpiRsGetIrq[8] =
61 {
62     {ACPI_RSC_INITGET,    ACPI_RESOURCE_TYPE_IRQ,
63      ACPI_RS_SIZE (ACPI_RESOURCE_IRQ),
64      ACPI_RSC_TABLE_SIZE (AcpiRsGetIrq)},

66     /* Get the IRQ mask (bytes 1:2) */

68     {ACPI_RSC_BITMASK16, ACPI_RS_OFFSET (Data.Irq.Interrupts[0]),
69      AML_OFFSET (Irq.IrqMask),
70      ACPI_RS_OFFSET (Data.Irq.InterruptCount)},

72     /* Set default flags (others are zero) */

74     {ACPI_RSC_SET8,        ACPI_RS_OFFSET (Data.Irq.Triggering),
75      ACPI_EDGE_SENSITIVE,
76      1},

78     /* Get the descriptor length (2 or 3 for IRQ descriptor) */

80     {ACPI_RSC_2BITFLAG,    ACPI_RS_OFFSET (Data.Irq.DescriptorLength),
81      AML_OFFSET (Irq.DescriptorType),
82      0},

84     /* All done if no flag byte present in descriptor */

86     {ACPI_RSC_EXIT_NE,     ACPI_RSC_COMPARE_AML_LENGTH, 0, 3},

88     /* Get flags: Triggering[0], Polarity[3], Sharing[4], Wake[5] */
88     /* Get flags: Triggering[0], Polarity[3], Sharing[4] */

90     {ACPI_RSC_1BITFLAG,    ACPI_RS_OFFSET (Data.Irq.Triggering),
91      AML_OFFSET (Irq.Flags),
92      0},

94     {ACPI_RSC_1BITFLAG,    ACPI_RS_OFFSET (Data.Irq.Polarity),
95      AML_OFFSET (Irq.Flags),
96      3},

98     {ACPI_RSC_1BITFLAG,    ACPI_RS_OFFSET (Data.Irq.Sharable),
99      AML_OFFSET (Irq.Flags),
100     4},

102     {ACPI_RSC_1BITFLAG,    ACPI_RS_OFFSET (Data.Irq.WakeCapable),
103      AML_OFFSET (Irq.Flags),
104      5},
105     4},
105 };

108 /*****
109 *
110 * AcpiRsSetIrq
111 *
112 *****/

114 ACPI_RS_CONVERT_INFO      AcpiRsSetIrq[14] =
114 ACPI_RS_CONVERT_INFO      AcpiRsSetIrq[13] =
115 {
116     /* Start with a default descriptor of length 3 */

118     {ACPI_RSC_INITSET,     ACPI_RESOURCE_NAME_IRQ,
119      sizeof (AML_RESOURCE_IRQ),
120      ACPI_RSC_TABLE_SIZE (AcpiRsSetIrq)},

```

```

122  /* Convert interrupt list to 16-bit IRQ bitmask */
124  {ACPI_RSC_BITMASK16, ACPI_RS_OFFSET (Data.Irq.Interrupts[0]),
125   AML_OFFSET (Irq.IrqMask),
126   ACPI_RS_OFFSET (Data.Irq.InterruptCount)},

128  /* Set flags: Triggering[0], Polarity[3], Sharing[4], Wake[5] */
129  /* Set the flags byte */

130  {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Irq.Triggering),
131   AML_OFFSET (Irq.Flags),
132   0},

134  {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Irq.Polarity),
135   AML_OFFSET (Irq.Flags),
136   3},

138  {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Irq.Sharable),
139   AML_OFFSET (Irq.Flags),
140   4},

142  {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Irq.WakeCapable),
143   AML_OFFSET (Irq.Flags),
144   5},

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},

154  /* Set length to 2 bytes (no flags byte) */

156  {ACPI_RSC_LENGTH, 0, 0, sizeof (AML_RESOURCE_IRQ_NOFLAGS)},

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},

168  /* Reset length to 3 bytes (descriptor with flags byte) */

170  {ACPI_RSC_LENGTH, 0, 0, sizeof (AML_RESOURCE_IRQ)},

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},

180  {ACPI_RSC_EXIT_NE, ACPI_RSC_COMPARE_VALUE,
181   ACPI_RS_OFFSET (Data.Irq.Polarity),
182   ACPI_ACTIVE_HIGH},

184  {ACPI_RSC_EXIT_NE, ACPI_RSC_COMPARE_VALUE,
185   ACPI_RS_OFFSET (Data.Irq.Sharable),

```

```

186   ACPI_EXCLUSIVE},

188   /* We can optimize to a 2-byte IrqNoFlags() descriptor */

190   {ACPI_RSC_LENGTH, 0, 0, sizeof (AML_RESOURCE_IRQ_NOFLAGS)}
191 };

194 /*****
195  *
196  * AcpiRsConvertExtIrq
197  *
198  *****/

200 ACPI_RS_CONVERT_INFO AcpiRsConvertExtIrq[10] =
201 {
202   {ACPI_RSC_INITGET, ACPI_RESOURCE_TYPE_EXTENDED_IRQ,
203    ACPI_RS_SIZE (ACPI_RESOURCE_EXTENDED_IRQ),
204    ACPI_RSC_TABLE_SIZE (AcpiRsConvertExtIrq)},

206   {ACPI_RSC_INITSET, ACPI_RESOURCE_NAME_EXTENDED_IRQ,
207    sizeof (AML_RESOURCE_EXTENDED_IRQ),
208    0},

210   /*
211    * Flags: Producer/Consumer[0], Triggering[1], Polarity[2],
212    *        Sharing[3], Wake[4]
213    */
214   /* Flag bits */

214   {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.ExtendedIrq.ProducerConsumer),
215    AML_OFFSET (ExtendedIrq.Flags),
216    0},

218   {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.ExtendedIrq.Triggering),
219    AML_OFFSET (ExtendedIrq.Flags),
220    1},

222   {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.ExtendedIrq.Polarity),
223    AML_OFFSET (ExtendedIrq.Flags),
224    2},

226   {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.ExtendedIrq.Sharable),
227    AML_OFFSET (ExtendedIrq.Flags),
228    3},

230   {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.ExtendedIrq.WakeCapable),
231    AML_OFFSET (ExtendedIrq.Flags),
232    4},

234   /* IRQ Table length (Byte4) */

236   {ACPI_RSC_COUNT, ACPI_RS_OFFSET (Data.ExtendedIrq.InterruptCount),
237    AML_OFFSET (ExtendedIrq.InterruptCount),
238    sizeof (UINT32)},

240   /* Copy every IRQ in the table, each is 32 bits */

242   {ACPI_RSC_MOVE32, ACPI_RS_OFFSET (Data.ExtendedIrq.Interrupts[0]),
243    AML_OFFSET (ExtendedIrq.Interrupts[0]),
244    0},

246   /* Optional ResourceSource (Index and String) */

248   {ACPI_RSC_SOURCEX, ACPI_RS_OFFSET (Data.ExtendedIrq.ResourceSource),

```

```
249         ACPI_RS_OFFSET (Data.ExtendedIrq.Interrupts[0]),
250         sizeof (AML_RESOURCE_EXTENDE_IRQ)}
251 };
252 _____unchanged_portion_omitted_____
253
254 292 /*****
255 293  *
256 294  * AcpiRsConvertFixedDma
257 295  *
258 296  *****/
259
260 298 ACPI_RS_CONVERT_INFO    AcpiRsConvertFixedDma[4] =
261 299 {
262 300     {ACPI_RSC_INITGET,    ACPI_RESOURCE_TYPE_FIXED_DMA,
263 301     ACPI_RS_SIZE (ACPI_RESOURCE_FIXED_DMA),
264 302     ACPI_RSC_TABLE_SIZE (AcpiRsConvertFixedDma)},
265
266 304     {ACPI_RSC_INITSET,    ACPI_RESOURCE_NAME_FIXED_DMA,
267 305     sizeof (AML_RESOURCE_FIXED_DMA),
268 306     0},
269
270 308     /*
271 309     * These fields are contiguous in both the source and destination:
272 310     * RequestLines
273 311     * Channels
274 312     */
275 313     {ACPI_RSC_MOVE16,    ACPI_RS_OFFSET (Data.FixedDma.RequestLines),
276 314     AML_OFFSET (FixedDma.RequestLines),
277 315     2},
278
279 317     {ACPI_RSC_MOVE8,    ACPI_RS_OFFSET (Data.FixedDma.Width),
280 318     AML_OFFSET (FixedDma.Width),
281 319     1},
282 320 };
283
```

new/usr/src/common/acpica/components/resources/rslist.c

1

```
*****
9588 Fri Jan 17 07:30:37 2014
new/usr/src/common/acpica/components/resources/rslist.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: rslist - Linked list utilities
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __RSLIST_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acresrc.h"

50 #define _COMPONENT          ACPI_RESOURCES
51 #define ACPI_MODULE_NAME    ("rslist")

54 /*****
55 *
56 * FUNCTION:      AcpiRsConvertAmlToResources
57 *
58 * PARAMETERS:    ACPI_WALK_AML_CALLBACK
59 *                 ResourcePtr          - Pointer to the buffer that will
```

new/usr/src/common/acpica/components/resources/rslist.c

2

```
60 *
61 *
62 * RETURN:        Status
63 *
64 * DESCRIPTION:    Convert an AML resource to an internal representation of the
65 *                 resource that is aligned and easier to access.
66 *
67 *****/

69 ACPI_STATUS
70 AcpiRsConvertAmlToResources (
71     UINT8             *Aml,
72     UINT32             Length,
73     UINT32             Offset,
74     UINT8             ResourceIndex,
75     **Context)
76 {
77     ACPI_RESOURCE      **ResourcePtr = ACPI_CAST_INDIRECT_PTR (
78         ACPI_RESOURCE, Context);
79     ACPI_RESOURCE      *Resource;
80     AML_RESOURCE        *AmlResource;
81     ACPI_RS_CONVERT_INFO *ConversionTable;
82     ACPI_STATUS          Status;

85     ACPI_FUNCTION_TRACE (RsConvertAmlToResources);

88     /*
89     * Check that the input buffer and all subsequent pointers into it
90     * are aligned on a native word boundary. Most important on IA64
91     */
92     Resource = *ResourcePtr;
93     if (ACPI_IS_MISALIGNED (Resource))
94     {
95         ACPI_WARNING ((AE_INFO,
96             "Misaligned resource pointer %p", Resource));
97     }

99     /* Get the appropriate conversion info table */

101     AmlResource = ACPI_CAST_PTR (AML_RESOURCE, Aml);
102     if (AcpiUtGetResourceType (Aml) == ACPI_RESOURCE_NAME_SERIAL_BUS)
103     {
104         if (AmlResource->CommonSerialBus.Type > AML_RESOURCE_MAX_SERIALBUSTYPE)
105         {
106             ConversionTable = NULL;
107         }
108     }
109     else
110     {
111         /* This is an I2C, SPI, or UART SerialBus descriptor */

112         ConversionTable =
113             AcpiGbl_ConvertResourceSerialBusDispatch[
114                 AmlResource->CommonSerialBus.Type];
115     }
116 }
117 else
118 {
119     ConversionTable =
120         AcpiGbl_GetResourceDispatch[ResourceIndex];
121 }

123 if (!ConversionTable)
124 {
125     ACPI_ERROR ((AE_INFO,
```

```

126     "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  * FUNCTION:    AcpiRsConvertResourcesToAml
156  *
157  * PARAMETERS:  Resource          - Pointer to the resource linked list
158  *              AmlSizeNeeded     - Calculated size of the byte stream
159  *                               - needed from calling AcpiRsGetAmlLength()
160  *                               - The size of the OutputBuffer is
161  *                               - guaranteed to be >= AmlSizeNeeded
162  *              OutputBuffer      - Pointer to the buffer that will
163  *                               - contain the byte stream
164  *
165  * RETURN:      Status
166  *
167  * DESCRIPTION: Takes the resource linked list and parses it, creating a
168  *              byte stream of resources in the caller's output buffer
169  *
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_RS_CONVERT_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 }
```

new/usr/src/common/acpica/components/resources/rsmemory.c

1

```
*****
8546 Fri Jan 17 07:30:38 2014
new/usr/src/common/acpica/components/resources/rsmemory.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: rsmem24 - Memory resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __RSMEMORY_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acresrc.h"

50 #define _COMPONENT          ACPI_RESOURCES
51 #define ACPI_MODULE_NAME    ("rsmemory")

54 /*****
55 *
56 * AcpiRsConvertMemory24
57 *
58 *****/
```

new/usr/src/common/acpica/components/resources/rsmemory.c

2

```
60 ACPI_RS_CONVERT_INFO      AcpiRsConvertMemory24[4] =
61 {
62     {ACPI_RSC_INITGET,     ACPI_RESOURCE_TYPE_MEMORY24,
63      ACPI_RS_SIZE (ACPI_RESOURCE_MEMORY24),
64      ACPI_RSC_TABLE_SIZE (AcpiRsConvertMemory24)},

66     {ACPI_RSC_INITSET,     ACPI_RESOURCE_NAME_MEMORY24,
67      sizeof (AML_RESOURCE_MEMORY24),
68      0},

70     /* Read/Write bit */

72     {ACPI_RSC_1BITFLAG,    ACPI_RS_OFFSET (Data.Memory24.WriteProtect),
73      AML_OFFSET (Memory24.Flags),
74      0},

75     /*
76     * These fields are contiguous in both the source and destination:
77     * Minimum Base Address
78     * Maximum Base Address
79     * Address Base Alignment
80     * Range Length
81     */
82     {ACPI_RSC_MOVE16,      ACPI_RS_OFFSET (Data.Memory24.Minimum),
83      AML_OFFSET (Memory24.Minimum),
84      4},
85 };

_____unchanged_portion_omitted_____
```



new/usr/src/common/acpica/components/resources/rsmisc.c

1

```
*****
25173 Fri Jan 17 07:30:38 2014
new/usr/src/common/acpica/components/resources/rsmisc.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: rsmisc - Miscellaneous resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __RSMISC_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acresrc.h"

50 #define _COMPONENT          ACPI_RESOURCES
51 #define ACPI_MODULE_NAME    ("rsmisc")

54 #define INIT_RESOURCE_TYPE(i)      i->ResourceOffset
55 #define INIT_RESOURCE_LENGTH(i)    i->AmlOffset
56 #define INIT_TABLE_LENGTH(i)      i->Value

58 #define COMPARE_OPCODE(i)          i->ResourceOffset
59 #define COMPARE_TARGET(i)         i->AmlOffset
```

new/usr/src/common/acpica/components/resources/rsmisc.c

2

```
60 #define COMPARE_VALUE(i)          i->Value

63 /*****
64 *
65 * FUNCTION:      AcpiRsConvertAmlToResource
66 *
67 * PARAMETERS:    Resource      - Pointer to the resource descriptor
68 *                Aml           - Where the AML descriptor is returned
69 *                Info          - Pointer to appropriate conversion table
70 *
71 * RETURN:        Status
72 *
73 * DESCRIPTION:    Convert an external AML resource descriptor to the corresponding
74 *                internal resource descriptor
75 *
76 *****/

78 ACPI_STATUS
79 AcpiRsConvertAmlToResource (
80     ACPI_RESOURCE      *Resource,
81     AML_RESOURCE        *Aml,
82     ACPI_RS_CONVERT_INFO *Info)
83 {
84     ACPI_RS_LENGTH      AmlResourceLength;
85     void                *Source;
86     void                *Destination;
87     char                *Target;
88     UINT8               Count;
89     UINT8               FlagsMode = FALSE;
90     UINT16              ItemCount = 0;
91     UINT16              Temp16 = 0;

94     ACPI_FUNCTION_TRACE (RsConvertAmlToResource);

97     if (!Info)
98     {
99         return ACPI_STATUS (AE_BAD_PARAMETER);
100     }

102     if (((ACPI_SIZE) Resource) & 0x3)
103     {
104         /* Each internal resource struct is expected to be 32-bit aligned */

106         ACPI_WARNING ((AE_INFO,
107             "Misaligned resource pointer (get): %p Type 0x%2.2X Length %u",
108             Resource, Resource->Type, Resource->Length));
109     }

111     /* Extract the resource Length field (does not include header length) */

113     AmlResourceLength = AcpiUtGetResourceLength (Aml);

115     /*
116     * First table entry must be ACPI_RSC_INITxxx and must contain the
117     * table length (# of table entries)
118     */
119     Count = INIT_TABLE_LENGTH (Info);
120     while (Count)
121     {
122         /*
123         * Source is the external AML byte stream buffer,
124         * destination is the internal resource descriptor
125         */
```

```

126     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;
272
273     case ACPI_RSC_MOVE_GPIO_PIN:
274
275         /* Generate and set the PIN data pointer */
276
277         Target = (char *) ACPI_ADD_PTR (void, Resource,
278             (Resource->Length - ItemCount * 2));
279         *(UINT16 **) Destination = ACPI_CAST_PTR (UINT16, Target);
280
281         /* Copy the PIN data */
282
283         Source = ACPI_ADD_PTR (void, Aml, ACPI_GET16 (Source));
284         AcpiRsMoveData (Target, Source, ItemCount, Info->Opcode);
285         break;
286
287     case ACPI_RSC_MOVE_GPIO_RES:
288
289         /* Generate and set the ResourceSource string pointer */
290
291         Target = (char *) ACPI_ADD_PTR (void, Resource,
292             (Resource->Length - ItemCount));
293         *(UINT8 **) Destination = ACPI_CAST_PTR (UINT8, Target);
294
295         /* Copy the ResourceSource string */
296
297         Source = ACPI_ADD_PTR (void, Aml, ACPI_GET16 (Source));
298         AcpiRsMoveData (Target, Source, ItemCount, Info->Opcode);
299         break;
300
301     case ACPI_RSC_MOVE_SERIAL_VEN:
302
303         /* Generate and set the Vendor Data pointer */
304
305         Target = (char *) ACPI_ADD_PTR (void, Resource,
306             (Resource->Length - ItemCount));
307         *(UINT8 **) Destination = ACPI_CAST_PTR (UINT8, Target);
308
309         /* Copy the Vendor Data */
310
311         Source = ACPI_ADD_PTR (void, Aml, Info->Value);
312         AcpiRsMoveData (Target, Source, ItemCount, Info->Opcode);
313         break;
314
315     case ACPI_RSC_MOVE_SERIAL_RES:
316
317         /* Generate and set the ResourceSource string pointer */
318
319         Target = (char *) ACPI_ADD_PTR (void, Resource,
320             (Resource->Length - ItemCount));
321         *(UINT8 **) Destination = ACPI_CAST_PTR (UINT8, Target);
322
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;
328
329     case ACPI_RSC_SET8:
330
331         ACPI_MEMSET (Destination, Info->AmlOffset, Info->Value);
332         break;
333
334     case ACPI_RSC_DATA8:
335
336         Target = ACPI_ADD_PTR (char, Resource, Info->Value);
337         ACPI_MEMCPY (Destination, Source, ACPI_GET16 (Target));
338         break;
339
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;
349
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;
358
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));
366
367         Resource->Length +=
368             AcpiRsGetResourceSource (AmlResourceLength, (ACPI_RS_LENGTH)
369                 (((ItemCount - 1) * sizeof (UINT32)) + Info->Value),
370                 Destination, Aml, Target);
371         break;
372
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         }
382
383         Target = ACPI_ADD_PTR (char, Resource, Info->Value);
384         ACPI_SET8 (Target, ItemCount);
385         break;
386
387     case ACPI_RSC_BITMASK16:
388         /*
389          * 16-bit encoded bitmask (IRQ macro)

```

```

390      */
391      ACPI_MOVE_16_TO_16 (&Templ6, Source);

393      ItemCount = AcpiRsDecodeBitmask (Templ6, 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_RS_CONVERT_INFO *Info)
473  {
474      void                *Source = NULL;
475      void                *Destination;
476      char                *Target;
477      ACPI_RSDESC_SIZE    AmlLength = 0;
478      UINT8               Count;
479      UINT16              Templ6 = 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;

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;

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;

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;

549     case ACPI_RSC_COUNT:

551     ItemCount = ACPI_GET8 (Source);
552     ACPI_SET8 (Destination, ItemCount);

554     AmlLength = (UINT16) (AmlLength + (Info->Value * (ItemCount - 1)));
555     break;

557     case ACPI_RSC_COUNT16:

559     ItemCount = ACPI_GET16 (Source);
560     AmlLength = (UINT16) (AmlLength + ItemCount);
561     AcpiRsSetResourceLength (AmlLength, Aml);
562     break;

564     case ACPI_RSC_COUNT_GPIO_PIN:

566     ItemCount = ACPI_GET16 (Source);
567     ACPI_SET16 (Destination, AmlLength);

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;

575     case ACPI_RSC_COUNT_GPIO_VEN:

577     ItemCount = ACPI_GET16 (Source);
578     ACPI_SET16 (Destination, ItemCount);

580     AmlLength = (UINT16) (AmlLength + (Info->Value * ItemCount));
581     AcpiRsSetResourceLength (AmlLength, Aml);
582     break;

584     case ACPI_RSC_COUNT_GPIO_RES:

586     /* Set resource source string length */

```

```

588     ItemCount = ACPI_GET16 (Source);
589     ACPI_SET16 (Destination, AmlLength);

591     /* Compute offset for the Vendor Data */

593     AmlLength = (UINT16) (AmlLength + ItemCount);
594     Target = ACPI_ADD_PTR (void, Aml, Info->Value);

596     /* Set vendor offset only if there is vendor data */

598     if (Resource->Data.Gpio.VendorLength)
599     {
600         ACPI_SET16 (Target, AmlLength);
601     }

603     AcpiRsSetResourceLength (AmlLength, Aml);
604     break;

606     case ACPI_RSC_COUNT_SERIAL_VEN:

608     ItemCount = ACPI_GET16 (Source);
609     ACPI_SET16 (Destination, ItemCount + Info->Value);
610     AmlLength = (UINT16) (AmlLength + ItemCount);
611     AcpiRsSetResourceLength (AmlLength, Aml);
612     break;

614     case ACPI_RSC_COUNT_SERIAL_RES:

616     ItemCount = ACPI_GET16 (Source);
617     AmlLength = (UINT16) (AmlLength + ItemCount);
618     AcpiRsSetResourceLength (AmlLength, Aml);
619     break;

621     case ACPI_RSC_LENGTH:

623     AcpiRsSetResourceLength (Info->Value, Aml);
624     break;

626     case ACPI_RSC_MOVE8:
627     case ACPI_RSC_MOVE16:
628     case ACPI_RSC_MOVE32:
629     case ACPI_RSC_MOVE64:

631     if (Info->Value)
632     {
633         ItemCount = Info->Value;
634     }
635     AcpiRsMoveData (Destination, Source, ItemCount, Info->Opcode);
636     break;

638     case ACPI_RSC_MOVE_GPIO_PIN:

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;

646     case ACPI_RSC_MOVE_GPIO_RES:

648     /* Used for both ResourceSource string and VendorData */

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         Templ6 = AcpiRsEncodeBitmask (Source,
711             *ACPI_ADD_PTR (UINT8, Resource, Info->Value));
712         ACPI_MOVE_16_TO_16 (Destination, &Templ6);
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     }
799
800     Resource->Data.ExtendedIrq.InterruptCount = Temp8;
801     if (Temp8 < 1)
802     {
803         /* Must have at least one IRQ */
804
805         return_ACPI_STATUS (AE_AML_BAD_RESOURCE_LENGTH);
806     }
807
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
```

new/usr/src/common/acpica/components/resources/rsserial.c

1

```
*****
17026 Fri Jan 17 07:30:38 2014
new/usr/src/common/acpica/components/resources/rsserial.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: rserial - GPIO/SerialBus resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __RSIRQ_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acresrc.h"

50 #define _COMPONENT ACPI_RESOURCES
51 #define ACPI_MODULE_NAME ("rserial")

54 /*****
55 *
56 * AcpiRsConvertGpio
57 *
58 *****/

60 ACPI_RS_CONVERT_INFO AcpiRsConvertGpio[18] =
```

new/usr/src/common/acpica/components/resources/rsserial.c

2

```
61 {
62     {ACPI_RSC_INITGET, ACPI_RESOURCE_TYPE_GPIO,
63      ACPI_RS_SIZE (ACPI_RESOURCE_GPIO),
64      ACPI_RSC_TABLE_SIZE (AcpiRsConvertGpio)},

66     {ACPI_RSC_INITSET, ACPI_RESOURCE_NAME_GPIO,
67      sizeof (AML_RESOURCE_GPIO),
68      0},

70 /*
71 * These fields are contiguous in both the source and destination:
72 * RevisionId
73 * ConnectionType
74 */
75 {ACPI_RSC_MOVE8, ACPI_RS_OFFSET (Data.Gpio.RevisionId),
76  AML_OFFSET (Gpio.RevisionId),
77  2},

79 {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Gpio.ProducerConsumer),
80  AML_OFFSET (Gpio.Flags),
81  0},

83 {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Gpio.Sharable),
84  AML_OFFSET (Gpio.IntFlags),
85  3},

87 {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Gpio.WakeCapable),
88  AML_OFFSET (Gpio.IntFlags),
89  4},

91 {ACPI_RSC_2BITFLAG, ACPI_RS_OFFSET (Data.Gpio.IoRestriction),
92  AML_OFFSET (Gpio.IntFlags),
93  0},

95 {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.Gpio.Triggering),
96  AML_OFFSET (Gpio.IntFlags),
97  0},

99 {ACPI_RSC_2BITFLAG, ACPI_RS_OFFSET (Data.Gpio.Polarity),
100  AML_OFFSET (Gpio.IntFlags),
101  1},

103 {ACPI_RSC_MOVE8, ACPI_RS_OFFSET (Data.Gpio.PinConfig),
104  AML_OFFSET (Gpio.PinConfig),
105  1},

107 /*
108 * These fields are contiguous in both the source and destination:
109 * DriveStrength
110 * DebounceTimeout
111 */
112 {ACPI_RSC_MOVE16, ACPI_RS_OFFSET (Data.Gpio.DriveStrength),
113  AML_OFFSET (Gpio.DriveStrength),
114  2},

116 /* Pin Table */

118 {ACPI_RSC_COUNT_GPIO_PIN, ACPI_RS_OFFSET (Data.Gpio.PinTableLength),
119  AML_OFFSET (Gpio.PinTableOffset),
120  AML_OFFSET (Gpio.ResourceOffset)},

122 {ACPI_RSC_MOVE_GPIO_PIN, ACPI_RS_OFFSET (Data.Gpio.PinTable),
123  AML_OFFSET (Gpio.PinTableOffset),
124  0},

126 /* 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_RS_CONVERT_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_RS_CONVERT_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_RS_CONVERT_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 */
394  {ACPI_RSC_2BITFLAG, ACPI_RS_OFFSET (Data.UartSerialBus.FlowControl),
395   AML_OFFSET (UartSerialBus.TypeSpecificFlags),
396   0},
398  {ACPI_RSC_2BITFLAG, ACPI_RS_OFFSET (Data.UartSerialBus.StopBits),
399   AML_OFFSET (UartSerialBus.TypeSpecificFlags),
400   2},
402  {ACPI_RSC_3BITFLAG, ACPI_RS_OFFSET (Data.UartSerialBus.DataBits),
403   AML_OFFSET (UartSerialBus.TypeSpecificFlags),
404   4},
406  {ACPI_RSC_1BITFLAG, ACPI_RS_OFFSET (Data.UartSerialBus.Endian),
407   AML_OFFSET (UartSerialBus.TypeSpecificFlags),
408   7},
410  {ACPI_RSC_MOVE8,     ACPI_RS_OFFSET (Data.UartSerialBus.Parity),
411   AML_OFFSET (UartSerialBus.Parity),
412   1},
414  {ACPI_RSC_MOVE8,     ACPI_RS_OFFSET (Data.UartSerialBus.LinesEnabled),
415   AML_OFFSET (UartSerialBus.LinesEnabled),
416   1},
418  {ACPI_RSC_MOVE16,    ACPI_RS_OFFSET (Data.UartSerialBus.RxFifoSize),
419   AML_OFFSET (UartSerialBus.RxFifoSize),
420   1},
422  {ACPI_RSC_MOVE16,    ACPI_RS_OFFSET (Data.UartSerialBus.TxFifoSize),
423   AML_OFFSET (UartSerialBus.TxFifoSize),
424   1},
426  {ACPI_RSC_MOVE32,    ACPI_RS_OFFSET (Data.UartSerialBus.DefaultBaudRate),
427   AML_OFFSET (UartSerialBus.DefaultBaudRate),
428   1},
429  };
```

new/usr/src/common/acpica/components/resources/rsutils.c

1

```
*****
25635 Fri Jan 17 07:30:38 2014
new/usr/src/common/acpica/components/resources/rsutils.c
update to acpica-unix2-20140114
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: rsutils - Utilities for the resource manager
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #define __RSUTILS_C__
46
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acnamesp.h"
50 #include "acresrc.h"
51
52 #define _COMPONENT ACPI_RESOURCES
53 #define ACPI_MODULE_NAME ("rsutils")
54
55 /*****
```

new/usr/src/common/acpica/components/resources/rsutils.c

2

```
58 *
59 * FUNCTION: AcpiRsDecodeBitmask
60 *
61 * PARAMETERS: Mask - Bitmask to decode
62 * List - Where the converted list is returned
63 *
64 * RETURN: Count of bits set (length of list)
65 *
66 * DESCRIPTION: Convert a bit mask into a list of values
67 *
68 *****/
69
70 UINT8
71 AcpiRsDecodeBitmask (
72     UINT16 Mask,
73     *List)
74 {
75     UINT8 i;
76     UINT8 BitCount;
77
78     ACPI_FUNCTION_ENTRY ();
79
80     /* Decode the mask bits */
81
82     for (i = 0, BitCount = 0; Mask; i++)
83     {
84         if (Mask & 0x0001)
85         {
86             List[BitCount] = i;
87             BitCount++;
88         }
89         Mask >>= 1;
90     }
91
92     return (BitCount);
93 }
94
95 /* unchanged portion omitted */
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115 /*****
116 *
117 * FUNCTION: AcpiRsMoveData
118 *
119 * PARAMETERS: Destination - Pointer to the destination descriptor
120 * Source - Pointer to the source descriptor
121 * ItemCount - How many items to move
122 * MoveType - Byte width
123 *
124 * RETURN: None
125 *
126 * DESCRIPTION: Move multiple data items from one descriptor to another. Handles
127 * alignment issues and endian issues if necessary, as configured
128 * via the ACPI_MOVE_* macros. (This is why a memcpy is not used)
129 *
130 *****/
131
132 void
133 AcpiRsMoveData (
134     void *Destination,
135     void *Source,
136     UINT16 ItemCount,
137     UINT8 MoveType)
138 {
```

```

159     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
186              * endian conversion and/or accomodate hardware that cannot perform
187              * misaligned memory transfers
188              */
189             case ACPI_RSC_MOVE16:
190             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_BTTYPE_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);
756     Status = AcpiUtEvaluateObject (Handle, Path, ACPI_BTYPE_BUFFER, &ObjDesc);
757     if (ACPI_FAILURE (Status))
758     {
759         return_ACPI_STATUS (Status);
760     }

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;
819     Info->Pathname = METHOD_NAME__SRS;
820     Info->Parameters = Args;
821     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);
831     Status = AcpiRsCreateAmlResources (InBuffer->Pointer, &Buffer);
832     if (ACPI_FAILURE (Status))
833     {
834         goto Cleanup;
835     }

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 = APOBJ_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 }

new/usr/src/common/acpica/components/resources/rsxface.c

1

```
*****
22711 Fri Jan 17 07:30:39 2014
new/usr/src/common/acpica/components/resources/rsxface.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: rsxface - Public interfaces to the resource manager
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #define __RSXFACE_C__
46 #define EXPORT_ACPI_INTERFACES
47
48 #include "acpi.h"
49 #include "accommon.h"
50 #include "acresrc.h"
51 #include "acnamesp.h"
52
53 #define _COMPONENT ACPI_RESOURCES
54 ACPI_MODULE_NAME ("rsxface")
55
56 /* Local macros for 16,32-bit to 64-bit conversion */
```

new/usr/src/common/acpica/components/resources/rsxface.c

2

```
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);
72
73 /* Local prototypes */
74
75 static ACPI_STATUS
76 AcpiRsMatchVendorResource (
77     ACPI_RESOURCE *Resource,
78     void *Context);
79
80 static ACPI_STATUS
81 AcpiRsValidateParameters (
82     ACPI_HANDLE DeviceHandle,
83     ACPI_BUFFER *Buffer,
84     ACPI_NAMESPACE_NODE **ReturnNode);
85
86 /*****
87 *
88 * FUNCTION: AcpiRsValidateParameters
89 *
90 * PARAMETERS: DeviceHandle - Handle to a device
91 * Buffer - Pointer to a data buffer
92 * ReturnNode - Pointer to where the device node is returned
93 *
94 * RETURN: Status
95 *
96 * DESCRIPTION: Common parameter validation for resource interfaces
97 *
98 *****/
99
100 static ACPI_STATUS
101 AcpiRsValidateParameters (
102     ACPI_HANDLE DeviceHandle,
103     ACPI_BUFFER *Buffer,
104     ACPI_NAMESPACE_NODE **ReturnNode)
105 {
106     ACPI_STATUS Status;
107     ACPI_NAMESPACE_NODE *Node;
108
109     ACPI_FUNCTION_TRACE (RsValidateParameters);
110
111     /* Must have a valid handle to an ACPI device */
112     if (!DeviceHandle)
113     {
114         return ACPI_STATUS (AE_BAD_PARAMETER);
115     }
116
117     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
481 * DESCRIPTION: Walk a resource template for the specified device 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
592 * FUNCTION:    AcpiWalkResources
593 *
594 * PARAMETERS: Buffer - Formatted buffer returned by one of the
595 *                  various Get*Resource functions
596 *                  DeviceHandle - Handle to the device object for the
597 *                  device we are querying
598 *                  Name         - Method name of the resources we want
599 *                          (METHOD_NAME__CRS or METHOD_NAME__PRS)
600 *                  UserFunction - Called for each resource
601 *                  Context      - Passed to UserFunction
602 *
603 * RETURN:     Status
604 *
605 * DESCRIPTION: Walks the input resource template. The UserFunction is called
606 *              once for each resource in the list.
607 * DESCRIPTION: Retrieves the current or possible resource list for the
608 *              specified device. The UserFunction is called once for
609 *              each resource in the list.
610 *
611 *****/

613 ACPI_STATUS
614 AcpiWalkResourceBuffer (
615     ACPI_BUFFER *Buffer,
616     AcpiWalkResources (
617         ACPI_HANDLE DeviceHandle,
618         char *Name,
619         ACPI_WALK_RESOURCE_CALLBACK UserFunction,
620         void *Context)
621     {
622         ACPI_STATUS Status = AE_OK;
623         ACPI_STATUS Status;
624         ACPI_BUFFER Buffer;
625         ACPI_RESOURCE *Resource;
626         ACPI_RESOURCE *ResourceEnd;

628         /* Parameter validation */

629         if (!Buffer || !Buffer->Pointer || !UserFunction)
630         if (!DeviceHandle || !UserFunction || !Name ||
631             (!ACPI_COMPARE_NAME (Name, METHOD_NAME__CRS) &&
632              !ACPI_COMPARE_NAME (Name, METHOD_NAME__PRS)))
633         {
634             return ACPI_STATUS (AE_BAD_PARAMETER);
635         }

636         /* Buffer contains the resource list and length */
637         /* Get the _CRS or _PRS resource list */

638         Resource = ACPI_CAST_PTR (ACPI_RESOURCE, Buffer->Pointer);
639         ResourceEnd = ACPI_ADD_PTR (ACPI_RESOURCE, Buffer->Pointer, Buffer->Length);
640         Buffer.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
641         Status = AcpiRsGetMethodData (DeviceHandle, Name, &Buffer);
642         if (ACPI_FAILURE (Status))
643         {
644             return ACPI_STATUS (Status);
645         }
646     }

```

```

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 */
636      /* 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);
674      Resource = ACPI_ADD_PTR (ACPI_RESOURCE, Resource, Resource->Length);
675  }

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  *                               - 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

```

new/usr/src/common/acpica/components/tables/tbfadt.c

1

```
*****
25213 Fri Jan 17 07:30:39 2014
new/usr/src/common/acpica/components/tables/tbfadt.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: tbfadt    - FADT table utilities
4 *
5 *****/
6
7 /*
8  * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define __TBFADT_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48 #include "actables.h"
49
50 #define _COMPONENT          ACPI_TABLES
51 #define ACPI_MODULE_NAME    ("tbfadt")
52
53 /* Local prototypes */
54
55 static void
56 AcpiTbInitGenericAddress (
57     ACPI_GENERIC_ADDRESS *GenericAddress,
58     UINT8 SpaceId,
```

new/usr/src/common/acpica/components/tables/tbfadt.c

2

```
59     UINT8 ByteWidth,
60     UINT64 Address,
61     char *RegisterName);
62
63 static void
64 AcpiTbConvertFadt (
65     void);
66
67 static void
68 AcpiTbSetupFadtRegisters (
69     void);
70
71 static UINT64
72 AcpiTbSelectAddress (
73     char *RegisterName,
74     UINT32 Address32,
75     UINT64 Address64);
76
77
78 /* Table for conversion of FADT to common internal format and FADT validation */
79
80 typedef struct acpi_fadt_info
81 {
82     char *Name;
83     UINT16 Address64;
84     UINT16 Address32;
85     UINT16 Length;
86     UINT8 DefaultLength;
87     UINT8 Type;
88
89 } ACPI_FADT_INFO;
90
91 #define ACPI_FADT_OPTIONAL 0
92 #define ACPI_FADT_REQUIRED 1
93 #define ACPI_FADT_SEPARATE_LENGTH 2
94
95 static ACPI_FADT_INFO FadtInfoTable[] =
96 {
97     {"Pm1aEventBlock",
98      ACPI_FADT_OFFSET (XPm1aEventBlock),
99      ACPI_FADT_OFFSET (Pm1aEventBlock),
100      ACPI_FADT_OFFSET (Pm1aEventLength),
101      ACPI_PM1_REGISTER_WIDTH * 2, /* Enable + Status register */
102      ACPI_FADT_REQUIRED},
103
104     {"Pm1bEventBlock",
105      ACPI_FADT_OFFSET (XPm1bEventBlock),
106      ACPI_FADT_OFFSET (Pm1bEventBlock),
107      ACPI_FADT_OFFSET (Pm1bEventLength),
108      ACPI_PM1_REGISTER_WIDTH * 2, /* Enable + Status register */
109      ACPI_FADT_OPTIONAL},
110
111     {"Pm1aControlBlock",
112      ACPI_FADT_OFFSET (XPm1aControlBlock),
113      ACPI_FADT_OFFSET (Pm1aControlBlock),
114      ACPI_FADT_OFFSET (Pm1aControlLength),
115      ACPI_PM1_REGISTER_WIDTH,
116      ACPI_FADT_REQUIRED},
117
118     {"Pm1bControlBlock",
119      ACPI_FADT_OFFSET (XPm1bControlBlock),
120      ACPI_FADT_OFFSET (Pm1bControlBlock),
121      ACPI_FADT_OFFSET (Pm1bControlLength),
122      ACPI_PM1_REGISTER_WIDTH,
123      ACPI_FADT_OPTIONAL},
```

```

125     {"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;
165 } ACPI_FADT_PM_INFO;

168 static ACPI_FADT_PM_INFO    FadtPmInfoTable[] =
169 {
170     {&AcpiGbl_XPmlaStatus,
171      ACPI_FADT_OFFSET (XPmlaEventBlock),
172      0},

174     {&AcpiGbl_XPmlaEnable,
175      ACPI_FADT_OFFSET (XPmlaEventBlock),
176      1},

178     {&AcpiGbl_XPmlbStatus,
179      ACPI_FADT_OFFSET (XPmlbEventBlock),
180      0},

182     {&AcpiGbl_XPmlbEnable,
183      ACPI_FADT_OFFSET (XPmlbEventBlock),
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
258 *
259 * DESCRIPTION: Select between 32-bit and 64-bit versions of addresses within
260 *              the FADT. Used for the FACS and DSDT addresses.
261 *
262 * NOTES:
263 *
264 * Check for FACS and DSDT address mismatches. An address mismatch between
265 * the 32-bit and 64-bit address fields (FIRMWARE_CTRL/X_FIRMWARE_CTRL and
266 * DSDT/X_DSDT) could be a corrupted address field or it might indicate
267 * the presence of two FACS or two DSDT tables.
268 *
269 * November 2013:
270 * By default, as per the ACPICA specification, a valid 64-bit address is
271 * used regardless of the value of the 32-bit address. However, this
272 * behavior can be overridden via the AcpiGbl_Use32BitFadtAddresses flag.
273 *
274 *****/
276 static UINT64
277 AcpiTbSelectAddress (
278     char                *RegisterName,
279     UINT32              Address32,
280     UINT64              Address64)
281 {
282     if (!Address64)
283     {
284         /* 64-bit address is zero, use 32-bit address */
285
286         return ((UINT64) Address32);
287     }
288
289     if (Address32 &&
290         (Address64 != (UINT64) Address32))
291     {
292         /* Address mismatch between 32-bit and 64-bit versions */
293
294         ACPI_BIOS_WARNING ((AE_INFO,
295             "32/64X %s address mismatch in FADT: "
296             "0x%8.8X/0x%8.8X%8.8X, using %u-bit address",
297             RegisterName, Address32, ACPI_FORMAT_UINT64 (Address64),
298             AcpiGbl_Use32BitFadtAddresses ? 32 : 64));
299
300         /* 32-bit address override */
301
302         if (AcpiGbl_Use32BitFadtAddresses)
303         {
304             return ((UINT64) Address32);
305         }
306     }
307
308     /* Default is to use the 64-bit address */
309
310     return (Address64);
311 }
312
313 *****/
315 /*****
316 *
317 * FUNCTION:      AcpiTbParseFadt
318 *
319 * PARAMETERS:    TableIndex          - Index for the FADT
320 *
321 * RETURN:        None
322 */

```

```

323 * DESCRIPTION: Initialize the FADT, DSDT and FACS tables
324 *              (FADT contains the addresses of the DSDT and FACS)
325 *
326 *****/
328 void
329 AcpiTbParseFadt (
330     UINT32              TableIndex)
331 {
332     UINT32              Length;
333     ACPI_TABLE_HEADER   *Table;
334
335     /*
336      * The FADT has multiple versions with different lengths,
337      * and it contains pointers to both the DSDT and FACS tables.
338      *
339      * Get a local copy of the FADT and convert it to a common format
340      * Map entire FADT, assumed to be smaller than one page.
341      */
342     Length = AcpiGbl_RootTableList.Tables[TableIndex].Length;
343
344     Table = AcpiOsMapMemory (
345         AcpiGbl_RootTableList.Tables[TableIndex].Address, Length);
346     if (!Table)
347     {
348         return;
349     }
350
351     /*
352      * Validate the FADT checksum before we copy the table. Ignore
353      * checksum error as we want to try to get the DSDT and FACS.
354      */
355     (void) AcpiTbVerifyChecksum (Table, Length);
356
357     /* Create a local copy of the FADT in common ACPI 2.0+ format */
358
359     AcpiTbCreateLocalFadt (Table, Length);
360
361     /* All done with the real FADT, unmap it */
362
363     AcpiOsUnmapMemory (Table, Length);
364
365     /* Obtain the DSDT and FACS tables via their addresses within the FADT */
366
367     AcpiTbInstallTable ((ACPI_PHYSICAL_ADDRESS) AcpiGbl_FADT.XDsdtd,
368         ACPI_SIG_DSDT, ACPI_TABLE_INDEX_DSDT);
369
370     /* If Hardware Reduced flag is set, there is no FACS */
371
372     if (!AcpiGbl_ReducedHardware)
373     {
374         AcpiTbInstallTable ((ACPI_PHYSICAL_ADDRESS) AcpiGbl_FADT.XFacs,
375             ACPI_SIG_FACS, ACPI_TABLE_INDEX_FACS);
376     }
377 }
378
379 *****/
381 /*****
382 *
383 * FUNCTION:      AcpiTbCreateLocalFadt
384 *
385 * PARAMETERS:    Table              - Pointer to BIOS FADT
386 *                Length             - Length of the table
387 *
388 * RETURN:        None

```

```

389 *
390 * DESCRIPTION: Get a local copy of the FADT and convert it to a common format.
391 *              Performs validation on some important FADT fields.
392 *
393 * NOTE:        We create a local copy of the FADT regardless of the version.
394 *
395 *****/

397 void
398 AcpiTbCreateLocalFadt (
399     ACPI_TABLE_HEADER *Table,
400     UINT32 Length)
401 {
402     /*
403     * Check if the FADT is larger than the largest table that we expect
404     * (the ACPI 5.0 version). If so, truncate the table, and issue
405     * a warning.
406     */
407     if (Length > sizeof (ACPI_TABLE_FADT))
408     {
409         ACPI_BIOS_WARNING ((AE_INFO,
410             "FADT (revision %u) is longer than ACPI 5.0 version, "
411             "truncating length %u to %u",
412             Table->Revision, Length, (UINT32) sizeof (ACPI_TABLE_FADT)));
413     }
414
415     /* Clear the entire local FADT */
416
417     ACPI_MEMSET (&AcpiGbl_FADT, 0, sizeof (ACPI_TABLE_FADT));
418
419     /* Copy the original FADT, up to sizeof (ACPI_TABLE_FADT) */
420
421     ACPI_MEMCPY (&AcpiGbl_FADT, Table,
422         ACPI_MIN (Length, sizeof (ACPI_TABLE_FADT)));
423
424     /* Take a copy of the Hardware Reduced flag */
425
426     AcpiGbl_ReducedHardware = FALSE;
427     if (AcpiGbl_FADT.Flags & ACPI_FADT_HW_REDUCED)
428     {
429         AcpiGbl_ReducedHardware = TRUE;
430     }
431
432     /* Convert the local copy of the FADT to the common internal format */
433
434     AcpiTbConvertFadt ();
435
436     /* Initialize the global ACPI register structures */
437
438     AcpiTbSetupFadtRegisters ();
439 }
440
441 *****/
442
443 /******
444 *
445 * FUNCTION:      AcpiTbConvertFadt
446 *
447 * PARAMETERS:    None - AcpiGbl_FADT is used.
448 *
449 * RETURN:        None
450 *
451 * DESCRIPTION:    Converts all versions of the FADT to a common internal format.
452 *                 Expand 32-bit addresses to 64-bit as necessary. Also validate
453 *                 important fields within the FADT.
454 */

```

```

455 * NOTE:          AcpiGbl_FADT must be of size (ACPI_TABLE_FADT), and must
456 *                contain a copy of the actual BIOS-provided FADT.
457 *
458 * Notes on 64-bit register addresses:
459 *
460 * After this FADT conversion, later ACPICA code will only use the 64-bit "X"
461 * fields of the FADT for all ACPI register addresses.
462 *
463 * The 64-bit X fields are optional extensions to the original 32-bit FADT
464 * V1.0 fields. Even if they are present in the FADT, they are optional and
465 * are unused if the BIOS sets them to zero. Therefore, we must copy/expand
466 * 32-bit V1.0 fields to the 64-bit X fields if the the 64-bit X field is
467 * originally zero.
468 *
469 * For ACPI 1.0 FADTs (that contain no 64-bit addresses), all 32-bit address
470 * fields are expanded to the corresponding 64-bit X fields in the internal
471 * common FADT.
472 *
473 * For ACPI 2.0+ FADTs, all valid (non-zero) 32-bit address fields are expanded
474 * to the corresponding 64-bit X fields, if the 64-bit field is originally
475 * zero. Adhering to the ACPI specification, we completely ignore the 32-bit
476 * field if the 64-bit field is valid, regardless of whether the host OS is
477 * 32-bit or 64-bit.
478 *
479 * Possible additional checks:
480 * (AcpiGbl_FADT.PmlEventLength >= 4)
481 * (AcpiGbl_FADT.PmlControlLength >= 2)
482 * (AcpiGbl_FADT.PmTimerLength >= 4)
483 * Gpe block lengths must be multiple of 2
484 *
485 *****/

487 static void
488 AcpiTbConvertFadt (
489     void)
490 {
491     char *Name;
492     ACPI_GENERIC_ADDRESS *Address64;
493     UINT32 Address32;
494     UINT8 Length;
495     UINT32 i;
496
497     /*
498     * For ACPI 1.0 FADTs (revision 1 or 2), ensure that reserved fields which
499     * should be zero are indeed zero. This will workaround BIOSs that
500     * inadvertently place values in these fields.
501     */
502     /*
503     * The ACPI 1.0 reserved fields that will be zeroed are the bytes located
504     * at offset 45, 55, 95, and the word located at offset 109, 110.
505     */
506     /* Note: The FADT revision value is unreliable. Only the length can be
507     * trusted.
508     */
509     if (AcpiGbl_FADT.Header.Length <= ACPI_FADT_V2_SIZE)
510     {
511         AcpiGbl_FADT.PreferredProfile = 0;
512         AcpiGbl_FADT.PstateControl = 0;
513         AcpiGbl_FADT.CstControl = 0;
514         AcpiGbl_FADT.BootFlags = 0;
515     }
516
517     /*
518     * Now we can update the local FADT length to the length of the
519     * current FADT version as defined by the ACPI specification.
520     * Thus, we will have a common FADT internally.

```

```

521  */
522  AcpiGbl_FADT.Header.Length = sizeof (ACPI_TABLE_FADT);

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);

531  AcpiGbl_FADT.XDsdT = AcpiTbSelectAddress ("DSDT",
532      AcpiGbl_FADT.DsdT, AcpiGbl_FADT.XDsdT);

534  /* If Hardware Reduced flag is set, we are all done */

536  if (AcpiGbl_ReducedHardware)
537  {
538      return;
539  }

541  /* Examine all of the 64-bit extended address fields (X fields) */

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);

552      Address64 = ACPI_ADD_PTR (ACPI_GENERIC_ADDRESS,
553          &AcpiGbl_FADT, FadtInfoTable[i].Address64);

555      Length = *ACPI_ADD_PTR (UINT8,
556          &AcpiGbl_FADT, FadtInfoTable[i].Length);

558      Name = FadtInfoTable[i].Name;

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 */

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 */

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));

608          if (AcpiGbl_Use32BitFadtAddresses)
609          {
610              /* 32-bit address override */

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  }

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 accommodate, 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  }

637  if (FadtInfoTable[i].Type & ACPI_FADT_REQUIRED)
638  {
639      /*
640       * Field is required (PMLaEvent, PMLaControl).
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  * FUNCTION:      AcpiTbSetupFadtRegisters
673  *
674  * PARAMETERS:    None, uses AcpiGbl_FADT.
675  *
676  * RETURN:        None
677  *
678  * DESCRIPTION:   Initialize global ACPI PM1 register definitions. Optionally,
679  *                force FADT register definitions to their default lengths.
680  *
681  * *****/
682

684 static void
685 AcpiTbSetupFadtRegisters (
686     void)
687 {
688     ACPI_GENERIC_ADDRESS *Target64;
689     ACPI_GENERIC_ADDRESS *Source64;
690     UINT8 PmlRegisterByteWidth;
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     PmlRegisterByteWidth = (UINT8)
731         ACPI_DIV_16 (AcpiGbl_FADT.XPmlaEventBlock.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 (PmlEventLength/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, PmlRegisterByteWidth,
756                                       Source64->Address +
757                                           (FadtPmInfoTable[i].RegisterNum * PmlRegisterByteWidth),
758                                       "PmRegisters");
759         }
760     }
761 }

```

new/usr/src/common/acpica/components/tables/tbfind.c

1

```
*****
5211 Fri Jan 17 07:30:40 2014
new/usr/src/common/acpica/components/tables/tbfind.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: tbfind - find table
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __TBFIND_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "actables.h"

50 #define _COMPONENT ACPI_TABLES
51 #define ACPI_MODULE_NAME ("tbfind")

54 /*****
55 *
56 * FUNCTION: AcpiTbFindTable
57 *
58 * PARAMETERS: Signature - String with ACPI table signature
```

new/usr/src/common/acpica/components/tables/tbfind.c

2

```
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;
135
136         ACPI_DEBUG_PRINT ((ACPI_DB_TABLES, "Found table [%4.4s]\n",
137                           Header.Signature));
138         return_ACPI_STATUS (AE_OK);
139     }
140 }
141
142 return_ACPI_STATUS (AE_NOT_FOUND);
143 }
```

unchanged\_portion\_omitted

new/usr/src/common/acpica/components/tables/tbinstal.c

1

```
*****
23466 Fri Jan 17 07:30:40 2014
new/usr/src/common/acpica/components/tables/tbinstal.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: tbinstal - ACPI table installation and removal
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __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 *
```

new/usr/src/common/acpica/components/tables/tbinstal.c

2

```
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     {
100         /* Always calculate checksum, ignore bad checksum if requested */
101         Status = AcpiTbVerifyChecksum (TableDesc->Pointer, TableDesc->Length);
102     }

103     return_ACPI_STATUS (Status);
104 }

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;
126     ACPI_TABLE_HEADER      *OverrideTable = NULL;
127
128     ACPI_FUNCTION_TRACE (TbAddTable);
129
130     if (!TableDesc->Pointer)
131     {
132         Status = AcpiTbVerifyTable (TableDesc);
133         if (ACPI_FAILURE (Status) || !TableDesc->Pointer)
134         {
135             return ACPI_STATUS (Status);
136         }
137     }
138
139     /*
140     * Validate the incoming table signature.
141     *
142     * 1) Originally, we checked the table signature for "SSDT" or "PSDT".
143     * 2) We added support for OEMx tables, signature "OEM".
144     * 3) Valid tables were encountered with a null signature, so we just
145     *    gave up on validating the signature, (05/2008).
146     * 4) We encountered non-AML tables such as the MADT, which caused
147     *    interpreter errors and kernel faults. So now, we once again allow
148     *    only "SSDT", "OEMx", and now, also a null signature. (05/2011).
149     */
150     if ((TableDesc->Pointer->Signature[0] != 0x00) &&
151         (!ACPI_COMPARE_NAME (TableDesc->Pointer->Signature, ACPI_SIG_SSDT)) &&
152         (ACPI_STRNCMP (TableDesc->Pointer->Signature, "OEM", 3)))
153     {
154         ACPI_BIOS_ERROR ((AE_INFO,
155             "Table has invalid signature [%4.4s] (0x%8.8X), "
156             "must be SSDT or OEMx",
157             AcpiUtValidAcpiName (TableDesc->Pointer->Signature) ?
158             ACPI_ERROR ((AE_INFO,
159                 "Table has invalid signature [%4.4s] (0x%8.8X), must be SSDT or OEMx"
160                 AcpiUtValidAcpiName (*(UINT32 *) TableDesc->Pointer->Signature) ?
161                 TableDesc->Pointer->Signature : "????",
162                 *(UINT32 *) TableDesc->Pointer->Signature));
163         return ACPI_STATUS (AE_BAD_SIGNATURE);
164     }
165
166     (void) AcpiUtAcquireMutex (ACPI_MTX_TABLES);
167
168     /* 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         }
180     }
181
182     /*

```

```

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     }
189
190     if (ACPI_MEMCMP (TableDesc->Pointer,
191         AcpiGbl_RootTableList.Tables[i].Pointer,
192         AcpiGbl_RootTableList.Tables[i].Length))
193     {
194         continue;
195     }
196
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     */
211
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;
218
219     if (AcpiGbl_RootTableList.Tables[i].Flags & ACPI_TABLE_IS_LOADED)
220     {
221         /* Table is still loaded, this is an error */
222
223         Status = AE_ALREADY_EXISTS;
224         goto Release;
225     }
226     else
227     {
228         /* Table was unloaded, allow it to be reloaded */
229
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 }
236
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);
244 Status = AcpiOsTableOverride (TableDesc->Pointer, &OverrideTable);
245 if (ACPI_SUCCESS (Status) && OverrideTable)
246 {
247     ACPI_INFO ((AE_INFO,

```

```

250     "%4.4s @ 0x%p Table override, replaced with:",
251     TableDesc->Pointer->Signature,
252     ACPI_CAST_PTR (void, TableDesc->Address));

254     /* We can delete the table that was passed as a parameter */

256     AcpiTbDeleteTable (TableDesc);

258     /* Setup descriptor for the new table */

260     TableDesc->Address = ACPI_PTR_TO_PHYSADDR (OverrideTable);
261     TableDesc->Pointer = OverrideTable;
262     TableDesc->Length = OverrideTable->Length;
263     TableDesc->Flags = ACPI_TABLE_ORIGIN_OVERRIDE;
264 }

245     /* Add the table to the global root table list */

247     Status = AcpiTbStoreTable (TableDesc->Address, TableDesc->Pointer,
248                               TableDesc->Length, TableDesc->Flags, TableIndex);
249     if (ACPI_FAILURE (Status))
250     {
251         goto Release;
252     }

254 PrintHeader:
255     AcpiTbPrintTableHeader (TableDesc->Address, TableDesc->Pointer);

257 Release:
258     (void) AcpiUtReleaseMutex (ACPI_MTX_TABLES);
259     return ACPI_STATUS (Status);
260 }

263 /*****
264 *
265 * FUNCTION:      AcpiTbTableOverride
266 *
267 * PARAMETERS:    TableHeader      - Header for the original table
268 *                TableDesc        - Table descriptor initialized for the
269 *                                original table. May or may not be mapped.
270 *
271 * RETURN:        Pointer to the entire new table. NULL if table not overridden.
272 *                If overridden, installs the new table within the input table
273 *                descriptor.
274 *
275 * DESCRIPTION:    Attempt table override by calling the OSL override functions.
276 *                Note: If the table is overridden, then the entire new table
277 *                is mapped and returned by this function.
278 *
279 *****/

281 ACPI_TABLE_HEADER *
282 AcpiTbTableOverride (
283     ACPI_TABLE_HEADER *TableHeader,
284     ACPI_TABLE_DESC *TableDesc)
285 {
286     ACPI_STATUS Status;
287     ACPI_TABLE_HEADER *NewTable = NULL;
288     ACPI_PHYSICAL_ADDRESS NewAddress = 0;
289     UINT32 NewTableLength = 0;
290     UINT8 NewFlags;
291     char *OverrideType;

294     /* (1) Attempt logical override (returns a logical address) */

```

```

296     Status = AcpiOsTableOverride (TableHeader, &NewTable);
297     if (ACPI_SUCCESS (Status) && NewTable)
298     {
299         NewAddress = ACPI_PTR_TO_PHYSADDR (NewTable);
300         NewTableLength = NewTable->Length;
301         NewFlags = ACPI_TABLE_ORIGIN_OVERRIDE;
302         OverrideType = "Logical";
303         goto FinishOverride;
304     }

306     /* (2) Attempt physical override (returns a physical address) */

308     Status = AcpiOsPhysicalTableOverride (TableHeader,
309                                           &NewAddress, &NewTableLength);
310     if (ACPI_SUCCESS (Status) && NewAddress && NewTableLength)
311     {
312         /* Map the entire new table */

314         NewTable = AcpiOsMapMemory (NewAddress, NewTableLength);
315         if (!NewTable)
316         {
317             ACPI_EXCEPTION ((AE_INFO, AE_NO_MEMORY,
318                             "%4.4s " ACPI_PRINTF_UINT
319                             " Attempted physical table override failed",
320                             TableHeader->Signature,
321                             ACPI_FORMAT_TO_UINT (TableDesc->Address)));
322             return (NULL);
323         }

325         OverrideType = "Physical";
326         NewFlags = ACPI_TABLE_ORIGIN_MAPPED;
327         goto FinishOverride;
328     }

330     return (NULL); /* There was no override */

333 FinishOverride:

335     ACPI_INFO ((AE_INFO, "%4.4s " ACPI_PRINTF_UINT
336                " %s table override, new table: " ACPI_PRINTF_UINT,
337                TableHeader->Signature,
338                ACPI_FORMAT_TO_UINT (TableDesc->Address),
339                OverrideType, ACPI_FORMAT_TO_UINT (NewTable)));

341     /* We can now unmap/delete the original table (if fully mapped) */

343     AcpiTbDeleteTable (TableDesc);

345     /* Setup descriptor for the new table */

347     TableDesc->Address = NewAddress;
348     TableDesc->Pointer = NewTable;
349     TableDesc->Length = NewTableLength;
350     TableDesc->Flags = NewFlags;

352     return (NewTable);
353 }

356 /*****
357 *
358 * FUNCTION:      AcpiTbResizeRootTableList
359 *
360 * PARAMETERS:    None

```

```

361 *
362 * RETURN:      Status
363 *
364 * DESCRIPTION: Expand the size of global table array
365 *
366 *****/
368 ACPI_STATUS
369 AcpiTbResizeRootTableList (
370     void)
371 {
372     ACPI_TABLE_DESC      *Tables;
373     UINT32                TableCount;

376     ACPI_FUNCTION_TRACE (TbResizeRootTableList);

379     /* AllowResize flag is a parameter to AcpiInitializeTables */

381     if (!(AcpiGbl_RootTableList.Flags & ACPI_ROOT_ALLOW_RESIZE))
382     {
383         ACPI_ERROR ((AE_INFO, "Resize of Root Table Array is not allowed"));
384         return_ACPI_STATUS (AE_SUPPORT);
385     }

387     /* Increase the Table Array size */

389     if (AcpiGbl_RootTableList.Flags & ACPI_ROOT_ORIGIN_ALLOCATED)
390     {
391         TableCount = AcpiGbl_RootTableList.MaxTableCount;
392     }
393     else
394     {
395         TableCount = AcpiGbl_RootTableList.CurrentTableCount;
396     }

398     Tables = ACPI_ALLOCATE_ZEROED (
399         ((ACPI_SIZE) TableCount + ACPI_ROOT_TABLE_SIZE_INCREMENT) *
400         ((ACPI_SIZE) AcpiGbl_RootTableList.MaxTableCount +
401             ACPI_ROOT_TABLE_SIZE_INCREMENT) *
402         sizeof (ACPI_TABLE_DESC));
403     if (!Tables)
404     {
405         ACPI_ERROR ((AE_INFO, "Could not allocate new root table array"));
406         return_ACPI_STATUS (AE_NO_MEMORY);
407     }

409     /* Copy and free the previous table array */

411     if (AcpiGbl_RootTableList.Tables)
412     {
413         ACPI_MEMCPY (Tables, AcpiGbl_RootTableList.Tables,
414             (ACPI_SIZE) TableCount * sizeof (ACPI_TABLE_DESC));
415         (ACPI_SIZE) AcpiGbl_RootTableList.MaxTableCount * sizeof (ACPI_TABLE
416             if (AcpiGbl_RootTableList.Flags & ACPI_ROOT_ORIGIN_ALLOCATED)
417             {
418                 ACPI_FREE (AcpiGbl_RootTableList.Tables);
419             }

421     AcpiGbl_RootTableList.Tables = Tables;
422     AcpiGbl_RootTableList.MaxTableCount =
423         TableCount + ACPI_ROOT_TABLE_SIZE_INCREMENT;
424     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;

425     return_ACPI_STATUS (AE_OK);
426 }
427
428     unchanged_portion_omitted

486 /*****
487 *
488 * FUNCTION:      AcpiTbDeleteTable
489 *
490 * PARAMETERS:    TableIndex          - Table index
491 *
492 * RETURN:        None
493 *
494 * DESCRIPTION:    Delete one internal ACPI table
495 *
496 *****/

498 void
499 AcpiTbDeleteTable (
500     ACPI_TABLE_DESC      *TableDesc)
501 {
503     /* Table must be mapped or allocated */

505     if (!TableDesc->Pointer)
506     {
507         return;
508     }

510     switch (TableDesc->Flags & ACPI_TABLE_ORIGIN_MASK)
511     {
512     case ACPI_TABLE_ORIGIN_MAPPED:

514         AcpiOsUnmapMemory (TableDesc->Pointer, TableDesc->Length);
515         break;

517     case ACPI_TABLE_ORIGIN_ALLOCATED:

519         ACPI_FREE (TableDesc->Pointer);
520         break;

522     /* Not mapped or allocated, there is nothing we can do */

524     default:

526         return;
527     }

529     TableDesc->Pointer = NULL;
530 }

533 /*****
534 *
535 * FUNCTION:      AcpiTbTerminate
536 *
537 * PARAMETERS:    None
538 *
539 * RETURN:        None
540 *
541 * DESCRIPTION:    Delete all internal ACPI tables
542 *

```

```
543  *****/
545 void
546 AcpiTbTerminate (
547     void)
548 {
549     UINT32          i;

552     ACPI_FUNCTION_TRACE (TbTerminate);

555     (void) AcpiUtAcquireMutex (ACPI_MTX_TABLES);

557     /* Delete the individual tables */

559     for (i = 0; i < AcpiGbl_RootTableList.CurrentTableCount; i++)
560     {
561         AcpiTbDeleteTable (&AcpiGbl_RootTableList.Tables[i]);
562     }

564     /*
565      * Delete the root table array if allocated locally. Array cannot be
566      * mapped, so we don't need to check for that flag.
567      */
568     if (AcpiGbl_RootTableList.Flags & ACPI_ROOT_ORIGIN_ALLOCATED)
569     {
570         ACPI_FREE (AcpiGbl_RootTableList.Tables);
571     }

573     AcpiGbl_RootTableList.Tables = NULL;
574     AcpiGbl_RootTableList.Flags = 0;
575     AcpiGbl_RootTableList.CurrentTableCount = 0;

577     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "ACPI Tables freed\n"));
578     (void) AcpiUtReleaseMutex (ACPI_MTX_TABLES);

580     return_VOID;
581 }
__unchanged_portion_omitted__
```



new/usr/src/common/acpica/components/tables/tbprint.c

1

```
*****
8724 Fri Jan 17 07:30:41 2014
new/usr/src/common/acpica/components/tables/tbprint.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: tbprint - Table output utilities
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __TBPRINT_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "actables.h"

50 #define _COMPONENT          ACPI_TABLES
51 #define ACPI_MODULE_NAME    ("tbprint")

54 /* Local prototypes */

56 static void
57 AcpiTbFixString (
58     char                *String,
59     ACPI_SIZE            Length);
```

new/usr/src/common/acpica/components/tables/tbprint.c

2

```
61 static void
62 AcpiTbCleanupTableHeader (
63     ACPI_TABLE_HEADER    *OutHeader,
64     ACPI_TABLE_HEADER    *Header);

67 /*****
68 *
69 * FUNCTION:      AcpiTbFixString
70 *
71 * PARAMETERS:    String          - String to be repaired
72 *                Length          - Maximum length
73 *
74 * RETURN:        None
75 *
76 * DESCRIPTION:   Replace every non-printable or non-ascii byte in the string
77 *                with a question mark '?'.
78 *
79 *****/

81 static void
82 AcpiTbFixString (
83     char                *String,
84     ACPI_SIZE            Length)
85 {
87     while (Length && *String)
88     {
89         if (!ACPI_IS_PRINT (*String))
90         {
91             *String = '?';
92         }
93         String++;
94         Length--;
95     }
96 }

99 /*****
100 *
101 * FUNCTION:      AcpiTbCleanupTableHeader
102 *
103 * PARAMETERS:    OutHeader      - Where the cleaned header is returned
104 *                Header         - Input ACPI table header
105 *
106 * RETURN:        Returns the cleaned header in OutHeader
107 *
108 * DESCRIPTION:   Copy the table header and ensure that all "string" fields in
109 *                the header consist of printable characters.
110 *
111 *****/

113 static void
114 AcpiTbCleanupTableHeader (
115     ACPI_TABLE_HEADER    *OutHeader,
116     ACPI_TABLE_HEADER    *Header)
117 {
119     ACPI_MEMCPY (OutHeader, Header, sizeof (ACPI_TABLE_HEADER));

121     AcpiTbFixString (OutHeader->Signature, ACPI_NAME_SIZE);
122     AcpiTbFixString (OutHeader->OemId, ACPI_OEM_ID_SIZE);
123     AcpiTbFixString (OutHeader->OemTableId, ACPI_OEM_TABLE_ID_SIZE);
124     AcpiTbFixString (OutHeader->AslCompilerId, ACPI_NAME_SIZE);
125 }
```

```

128 /*****
129 *
130 * FUNCTION:      AcpiTbPrintTableHeader
131 *
132 * PARAMETERS:    Address          - Table physical address
133 *                Header           - Table header
134 *
135 * RETURN:        None
136 *
137 * DESCRIPTION:   Print an ACPI table header. Special cases for FACS and RSDP.
138 *
139 *****/
141 void
142 AcpiTbPrintTableHeader (
143     ACPI_PHYSICAL_ADDRESS    Address,
144     ACPI_TABLE_HEADER        *Header)
145 {
146     ACPI_TABLE_HEADER        LocalHeader;
147
148     /*
149     * The reason that we use ACPI_PRINTF_UINT and ACPI_FORMAT_TO_UINT is to
150     * support both 32-bit and 64-bit hosts/addresses in a consistent manner.
151     * The %p specifier does not emit uniform output on all hosts. On some,
152     * leading zeros are not supported.
153     */
154     if (ACPI_COMPARE_NAME (Header->Signature, ACPI_SIG_FACS))
155     {
156         /* FACS only has signature and length fields */
157
158         ACPI_INFO ((AE_INFO, "%-4.4s " ACPI_PRINTF_UINT " %06X",
159             Header->Signature, ACPI_FORMAT_TO_UINT (Address),
160             Header->Length));
161     }
162     else if (ACPI_VALIDATE_RSDP_SIG (Header->Signature))
163     {
164         /* RSDP has no common fields */
165
166         ACPI_MEMCPY (LocalHeader.OemId,
167             ACPI_CAST_PTR (ACPI_TABLE_RSDP, Header)->OemId, ACPI_OEM_ID_SIZE);
168         AcpiTbFixString (LocalHeader.OemId, ACPI_OEM_ID_SIZE);
169
170         ACPI_INFO ((AE_INFO, "RSDP " ACPI_PRINTF_UINT " %06X (v%.2d %-.6s)",
171             ACPI_FORMAT_TO_UINT (Address),
172             (ACPI_CAST_PTR (ACPI_TABLE_RSDP, Header)->Revision > 0) ?
173             ACPI_CAST_PTR (ACPI_TABLE_RSDP, Header)->Length : 20,
174             ACPI_CAST_PTR (ACPI_TABLE_RSDP, Header)->Revision,
175             LocalHeader.OemId));
176     }
177     else
178     {
179         /* Standard ACPI table with full common header */
180
181         AcpiTbCleanupTableHeader (&LocalHeader, Header);
182
183         ACPI_INFO ((AE_INFO,
184             "%-4.4s " ACPI_PRINTF_UINT
185             " %06X (v%.2d %-.6s %-8.8s %08X %-4.4s %08X)",
186             LocalHeader.Signature, ACPI_FORMAT_TO_UINT (Address),
187             LocalHeader.Length, LocalHeader.Revision, LocalHeader.OemId,
188             LocalHeader.OemTableId, LocalHeader.OemRevision,
189             LocalHeader.AslCompilerId, LocalHeader.AslCompilerRevision));
190     }
191 }

```

```

192 }

195 /*****
196 *
197 * FUNCTION:      AcpiTbValidateChecksum
198 *
199 * PARAMETERS:    Table            - ACPI table to verify
200 *                Length           - Length of entire table
201 *
202 * RETURN:        Status
203 *
204 * DESCRIPTION:   Verifies that the table checksums to zero. Optionally returns
205 *                exception on bad checksum.
206 *
207 *****/
209 ACPI_STATUS
210 AcpiTbVerifyChecksum (
211     ACPI_TABLE_HEADER        *Table,
212     UINT32                   Length)
213 {
214     UINT8                    Checksum;
215
216     /*
217     * FACS/S3PT:
218     * They are the odd tables, have no standard ACPI header and no checksum
219     */
220
221     if (ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_S3PT) ||
222         ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_FACS))
223     {
224         return (AE_OK);
225     }
226
227     /* Compute the checksum on the table */
228
229     Checksum = AcpiTbChecksum (ACPI_CAST_PTR (UINT8, Table), Length);
230
231     /* Checksum ok? (should be zero) */
232
233     if (Checksum)
234     {
235         ACPI_BIOS_WARNING ((AE_INFO,
236             "Incorrect checksum in table [%4.4s] - 0x%.2.2X, "
237             "should be 0x%.2.2X",
238             Table->Signature, Table->Checksum,
239             (UINT8) (Table->Checksum - Checksum)));
240
241         #if (ACPI_CHECKSUM_ABORT)
242             return (AE_BAD_CHECKSUM);
243         #endif
244     }
245
246     return (AE_OK);
247 }

251 /*****
252 *
253 * FUNCTION:      AcpiTbChecksum
254 *
255 * PARAMETERS:    Buffer            - Pointer to memory region to be checked
256 *                Length           - Length of this memory region
257 *

```

```
258 * RETURN:      Checksum (UINT8)
259 *
260 * DESCRIPTION: Calculates circular checksum of memory region.
261 *
262 *****/

264 UINT8
265 AcpiTbChecksum (
266     UINT8          *Buffer,
267     UINT32          Length)
268 {
269     UINT8          Sum = 0;
270     UINT8          *End = Buffer + Length;

273     while (Buffer < End)
274     {
275         Sum = (UINT8) (Sum + *(Buffer++));
276     }

278     return (Sum);
279 }
```

new/usr/src/common/acpica/components/tables/tbutils.c

1

```
*****
20805 Fri Jan 17 07:30:41 2014
new/usr/src/common/acpica/components/tables/tbutils.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: tbutils - ACPI Table utilities
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define __TBUTILS_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48 #include "actables.h"
49
50 #define _COMPONENT          ACPI_TABLES
51 #define ACPI_MODULE_NAME    ("tbutils")
52
53
54 /* Local prototypes */
55
56 static ACPI_STATUS
57 AcpiTbValidateXsdt (
58     ACPI_PHYSICAL_ADDRESS    Address);
```

new/usr/src/common/acpica/components/tables/tbutils.c

2

```
60 static ACPI_PHYSICAL_ADDRESS
61 AcpiTbGetRootTableEntry (
62     UINT8                *TableEntry,
63     UINT32                TableEntrySize);
64
65
66 #if (!ACPI_REDUCED_HARDWARE)
67 /*****
68 *
69 * FUNCTION:      AcpiTbInitializeFacs
70 *
71 * PARAMETERS:    None
72 *
73 * RETURN:        Status
74 *
75 * DESCRIPTION:   Create a permanent mapping for the FADT and save it in a global
76 *                 for accessing the Global Lock and Firmware Waking Vector
77 *
78 *****/
79
80 ACPI_STATUS
81 AcpiTbInitializeFacs (
82     void)
83 {
84     ACPI_STATUS    Status;
85
86     /* If Hardware Reduced flag is set, there is no FACS */
87
88     if (AcpiGbl_ReducedHardware)
89     {
90         AcpiGbl_FACS = NULL;
91         return (AE_OK);
92     }
93
94     Status = AcpiGetTableByIndex (ACPI_TABLE_INDEX_FACS,
95                                   ACPI_CAST_INDIRECT_PTR (ACPI_TABLE_HEADER, &AcpiGbl_FACS));
96     return (Status);
97 }
98 #endif /* !ACPI_REDUCED_HARDWARE */
99
100 /*****
101 *
102 * FUNCTION:      AcpiTbTablesLoaded
103 *
104 * PARAMETERS:    None
105 *
106 * RETURN:        TRUE if required ACPI tables are loaded
107 *
108 * DESCRIPTION:   Determine if the minimum required ACPI tables are present
109 *                 (FADT, FACS, DSDT)
110 *
111 *****/
112
113
114
115 BOOLEAN
116 AcpiTbTablesLoaded (
117     void)
118 {
119
120     if (AcpiGbl_RootTableList.CurrentTableCount >= 3)
121     {
122         return (TRUE);
123     }
124 }
```

```

125     return (FALSE);
126 }

129 /*****
130  *
131  * FUNCTION:     AcpiTbCheckDsdtdHeader
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 AcpiTbCheckDsdtdHeader (
145     void)
146 {
147
148     /* Compare original length and checksum to current values */
149
150     if (AcpiGbl_OriginalDsdtdHeader.Length != AcpiGbl_DSDT->Length ||
151         AcpiGbl_OriginalDsdtdHeader.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_OriginalDsdtdHeader);
157         AcpiTbPrintTableHeader (0, AcpiGbl_DSDT);
158
159         /* Disable further error messages */
160
161         AcpiGbl_OriginalDsdtdHeader.Length = AcpiGbl_DSDT->Length;
162         AcpiGbl_OriginalDsdtdHeader.Checksum = AcpiGbl_DSDT->Checksum;
163     }
164 }

167 /*****
168  *
169  * FUNCTION:     AcpiTbCopyDsdtd
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 AcpiTbCopyDsdtd (
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 */
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  }

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];

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);

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  }

300  AcpiTbPrintTableHeader (TableDesc->Address, FinalTable);

302  /* Set the global integer width (based upon revision of the DSDT) */

304  if (TableIndex == ACPI_TABLE_INDEX_DSDT)
305  {
306      AcpiUtSetIntegerWidth (FinalTable->Revision);
307  }

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  }

```

```

323  UnmapAndExit:

325  /* Always unmap the table header that we mapped above */

327  AcpiOsUnmapMemory (Table, sizeof (ACPI_TABLE_HEADER));
328  }

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   *****/

348  static ACPI_PHYSICAL_ADDRESS
349  AcpiTbGetRootTableEntry (
350      UINT8          *TableEntry,
351      UINT32         TableEntrySize)
352  {
353      UINT64         Address64;

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);

377  #if ACPI_MACHINE_WIDTH == 32
378      if (Address64 > ACPI_UINT32_MAX)
379      {
380          /* Will truncate 64-bit address to 32 bits, issue warning */

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     }
390 }

393 /*****
394  *
395  * FUNCTION:    AcpiTbValidateXsdt
396  *
397  * PARAMETERS:  Address          - Physical address of the XSDT (from RSDP)
398  *
399  * RETURN:      Status. AE_OK if the table appears to be valid.
400  *
401  * DESCRIPTION: Validate an XSDT to ensure that it is of minimum size and does
402  *              not contain any NULL entries. A problem that is seen in the
403  *              field is that the XSDT exists, but is actually useless because
404  *              of one or more (or all) NULL entries.
405  *
406  *****/

408 static ACPI_STATUS
409 AcpiTbValidateXsdt (
410     ACPI_PHYSICAL_ADDRESS XsdtAddress)
411 {
412     ACPI_TABLE_HEADER *Table;
413     UINT8 *NextEntry;
414     ACPI_PHYSICAL_ADDRESS Address;
415     UINT32 Length;
416     UINT32 EntryCount;
417     ACPI_STATUS Status;
418     UINT32 i;

421     /* Get the XSDT length */

423     Table = AcpiOsMapMemory (XsdtAddress, sizeof (ACPI_TABLE_HEADER));
424     if (!Table)
425     {
426         return (AE_NO_MEMORY);
427     }

429     Length = Table->Length;
430     AcpiOsUnmapMemory (Table, sizeof (ACPI_TABLE_HEADER));

432     /*
433      * Minimum XSDT length is the size of the standard ACPI header
434      * plus one physical address entry
435      */
436     if (Length < (sizeof (ACPI_TABLE_HEADER) + ACPI_XSDT_ENTRY_SIZE))
437     {
438         return (AE_INVALID_TABLE_LENGTH);
439     }

441     /* Map the entire XSDT */

443     Table = AcpiOsMapMemory (XsdtAddress, Length);
444     if (!Table)
445     {
446         return (AE_NO_MEMORY);
447     }

449     /* Get the number of entries and pointer to first entry */

451     Status = AE_OK;
452     NextEntry = ACPI_ADD_PTR (UINT8, Table, sizeof (ACPI_TABLE_HEADER));
453     EntryCount = (UINT32) ((Table->Length - sizeof (ACPI_TABLE_HEADER)) /
454         ACPI_XSDT_ENTRY_SIZE);

```

```

456     /* Validate each entry (physical address) within the XSDT */

458     for (i = 0; i < EntryCount; i++)
459     {
460         Address = AcpiTbGetRootTableEntry (NextEntry, ACPI_XSDT_ENTRY_SIZE);
461         if (!Address)
462         {
463             /* Detected a NULL entry, XSDT is invalid */

465             Status = AE_NULL_ENTRY;
466             break;
467         }

469         NextEntry += ACPI_XSDT_ENTRY_SIZE;
470     }

472     /* Unmap table */

474     AcpiOsUnmapMemory (Table, Length);
475     return (Status);
476 }

479 /*****
480  *
481  * FUNCTION:    AcpiTbParseRootTable
482  *
483  * PARAMETERS:  Rsdp          - Pointer to the RSDP
484  *
485  * RETURN:      Status
486  *
487  * DESCRIPTION: This function is called to parse the Root System Description
488  *              Table (RSDT or XSDT)
489  *
490  * NOTE:        Tables are mapped (not copied) for efficiency. The FACS must
491  *              be mapped and cannot be copied because it contains the actual
492  *              memory location of the ACPI Global Lock.
493  *
494  *****/

496 ACPI_STATUS
497 AcpiTbParseRootTable (
498     ACPI_PHYSICAL_ADDRESS RsdpAddress)
499 {
500     ACPI_TABLE_RSDP *Rsdp;
501     UINT32 TableEntrySize;
502     UINT32 i;
503     UINT32 TableCount;
504     ACPI_TABLE_HEADER *Table;
505     ACPI_PHYSICAL_ADDRESS Address;
506     UINT32 Length;
507     UINT8 *TableEntry;
508     ACPI_STATUS Status;

511     ACPI_FUNCTION_TRACE (TbParseRootTable);

514     /* Map the entire RSDP and extract the address of the RSDT or XSDT */

516     Rsdp = AcpiOsMapMemory (RsdpAddress, sizeof (ACPI_TABLE_RSDP));
517     if (!Rsdp)
518     {
519         return ACPI_STATUS (AE_NO_MEMORY);
520     }

```

```

522 AcpiTbPrintTableHeader (RsdpAddress,
523     ACPI_CAST_PTR (ACPI_TABLE_HEADER, Rsdp));

525 /* Use XSDT if present and not overridden. Otherwise, use RSDT */

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) */

543     Address = (ACPI_PHYSICAL_ADDRESS) Rsdp->RsdtpPhysicalAddress;
544     TableEntrySize = ACPI_RSDT_ENTRY_SIZE;
545 }

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));

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 */

567     Address = (ACPI_PHYSICAL_ADDRESS) Rsdp->RsdtpPhysicalAddress;
568     TableEntrySize = ACPI_RSDT_ENTRY_SIZE;
569 }
570 }

572 /* Map the RSDT/XSDT table header to get the full table length */

574 Table = AcpiOsMapMemory (Address, sizeof (ACPI_TABLE_HEADER));
575 if (!Table)
576 {
577     return_ACPI_STATUS (AE_NO_MEMORY);
578 }

580 AcpiTbPrintTableHeader (Address, Table);

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));

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 }

596 Table = AcpiOsMapMemory (Address, Length);
597 if (!Table)
598 {
599     return_ACPI_STATUS (AE_NO_MEMORY);
600 }

602 /* Validate the root table checksum */

604 Status = AcpiTbVerifyChecksum (Table, Length);
605 if (ACPI_FAILURE (Status))
606 {
607     AcpiOsUnmapMemory (Table, Length);
608     return_ACPI_STATUS (Status);
609 }

611 /* Get the number of entries and pointer to first entry */

613 TableCount = (UINT32) ((Table->Length - sizeof (ACPI_TABLE_HEADER)) /
614     TableEntrySize);
615 TableEntry = ACPI_ADD_PTR (UINT8, Table, sizeof (ACPI_TABLE_HEADER));

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;

624 /* Initialize the root table array from the RSDT/XSDT */

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 */

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     }

643     /* Get the table physical address (32-bit for RSDT, 64-bit for XSDT) */

645     AcpiGbl_RootTableList.Tables[AcpiGbl_RootTableList.CurrentTableCount].Ad
646         AcpiTbGetRootTableEntry (TableEntry, TableEntrySize);

648     TableEntry += TableEntrySize;
649     AcpiGbl_RootTableList.CurrentTableCount++;
650 }

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 }
```

new/usr/src/common/acpica/components/tables/tbxface.c

1

```
*****
15006 Fri Jan 17 07:30:41 2014
new/usr/src/common/acpica/components/tables/tbxface.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: tbxface - ACPI table-oriented external interfaces
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define __TBXFACE_C__
45 #define EXPORT_ACPI_INTERFACES
46
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "actables.h"
50
51 #define _COMPONENT          ACPI_TABLES
52 #define ACPI_MODULE_NAME    ("tbxface")
53
54
55 /*****
56 *
57 * FUNCTION:      AcpiAllocateRootTable
58 *
59 *****/
```

new/usr/src/common/acpica/components/tables/tbxface.c

2

```
59 * PARAMETERS:   InitialTableCount - Size of InitialTableArray, in number of
60 *               ACPI_TABLE_DESC structures
61 *
62 * RETURN:       Status
63 *
64 * DESCRIPTION:  Allocate a root table array. Used by iASL compiler and
65 *               AcpiInitializeTables.
66 *
67 *****/
68
69 ACPI_STATUS
70 AcpiAllocateRootTable (
71     UINT32                InitialTableCount)
72 {
73
74     AcpiGbl_RootTableList.MaxTableCount = InitialTableCount;
75     AcpiGbl_RootTableList.Flags = ACPI_ROOT_ALLOW_RESIZE;
76
77     return (AcpiTbResizeRootTableList ());
78 }
79
80
81 /*****
82 *
83 * FUNCTION:      AcpiInitializeTables
84 *
85 * PARAMETERS:   InitialTableArray - Pointer to an array of pre-allocated
86 *               ACPI_TABLE_DESC structures. If NULL, the
87 *               array is dynamically allocated.
88 *               InitialTableCount - Size of InitialTableArray, in number of
89 *               ACPI_TABLE_DESC structures
90 *               AllowResize - Flag to tell Table Manager if resize of
91 *               pre-allocated array is allowed. Ignored
92 *               if InitialTableArray is NULL.
93 *
94 * RETURN:       Status
95 *
96 * DESCRIPTION:  Initialize the table manager, get the RSDP and RSDT/XSDT.
97 *
98 * NOTE:        Allows static allocation of the initial table array in order
99 *               to avoid the use of dynamic memory in confined environments
100 *               such as the kernel boot sequence where it may not be available.
101 *
102 *               If the host OS memory managers are initialized, use NULL for
103 *               InitialTableArray, and the table will be dynamically allocated.
104 *
105 *****/
106
107 ACPI_STATUS
108 AcpiInitializeTables (
109     ACPI_TABLE_DESC        *InitialTableArray,
110     UINT32                 InitialTableCount,
111     BOOLEAN                AllowResize)
112 {
113     ACPI_PHYSICAL_ADDRESS  RsdpAddress;
114     ACPI_STATUS             Status;
115
116
117     ACPI_FUNCTION_TRACE (AcpiInitializeTables);
118
119
120     /*
121     * Setup the Root Table Array and allocate the table array
122     * if requested
123     */
124     if (!InitialTableArray)
```

```

125 {
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 */
135
136     ACPI_MEMSET (InitialTableArray, 0,
137                 (ACPI_SIZE) InitialTableCount * sizeof (ACPI_TABLE_DESC));
138
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 }
147
148 /* Get the address of the RSDP */
149
150 RsdAddress = AcpiOsGetRootPointer ();
151 if (!RsdAddress)
152 {
153     return_ACPI_STATUS (AE_NOT_FOUND);
154 }
155
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 (RsdAddress);
162 return_ACPI_STATUS (Status);
163 }
164
165 ACPI_EXPORT_SYMBOL_INIT (AcpiInitializeTables)
166
167 /*****
168  *
169  * FUNCTION:      AcpiReallocateRootTable
170  *
171  * PARAMETERS:    None
172  *
173  * RETURN:        Status
174  *
175  * DESCRIPTION:   Reallocate Root Table List into dynamic memory. Copies the
176  *                root list from the previously provided scratch area. Should
177  *                be called once dynamic memory allocation is available in the
178  *                kernel.
179  *
180  * *****/
181
182 ACPI_STATUS
183 AcpiReallocateRootTable (
184     void)
185 {
186     ACPI_STATUS      Status;
187
188     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     }
201
202     AcpiGbl_RootTableList.Flags |= ACPI_ROOT_ALLOW_RESIZE;
203
204     Status = AcpiTbResizeRootTableList ();
205     return_ACPI_STATUS (Status);
206 }
207
208 ACPI_EXPORT_SYMBOL_INIT (AcpiReallocateRootTable)
209
210 /*****
211  *
212  * FUNCTION:      AcpiGetTableHeader
213  *
214  * PARAMETERS:    Signature          - ACPI signature of needed table
215  *                Instance          - Which instance (for SSDTs)
216  *                OutTableHeader    - The pointer to the table header to fill
217  *
218  * RETURN:        Status and pointer to mapped table header
219  *
220  * DESCRIPTION:   Finds an ACPI table header.
221  *
222  * NOTE:          Caller is responsible in unmapping the header with
223  *                AcpiOsUnmapMemory
224  *
225  * *****/
226
227 ACPI_STATUS
228 AcpiGetTableHeader (
229     char                *Signature,
230     UINT32              Instance,
231     ACPI_TABLE_HEADER   *OutTableHeader)
232 {
233     ACPI_STATUS      Status;
234     ACPI_TABLE_HEADER TableHeader;
235     ACPI_TABLE_HEADER TableHeader;
236     ACPI_TABLE_HEADER TableHeader;
237
238     /* Parameter validation */
239
240     if (!Signature || !OutTableHeader)
241     {
242         return (AE_BAD_PARAMETER);
243     }
244
245     /* Walk the root table list */
246
247     for (i = 0, j = 0; i < AcpiGbl_RootTableList.CurrentTableCount; i++)
248     {
249         if (!ACPI_COMPARE_NAME (&(AcpiGbl_RootTableList.Tables[i].Signature),
250                                Signature))
251         {
252             continue;
253         }
254
255         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)
```

new/usr/src/common/acpica/components/tables/tbxload.c

1

```
*****
12699 Fri Jan 17 07:30:41 2014
new/usr/src/common/acpica/components/tables/tbxload.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: tbxload - Table load/unload external interfaces
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __TBXFLOAD_C__
45 #define EXPORT_ACPI_INTERFACES

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acnamesp.h"
50 #include "actables.h"

52 #define _COMPONENT          ACPI_TABLES
53       ACPI_MODULE_NAME    ("tbxload")

55 /* Local prototypes */

57 static ACPI_STATUS
58 AcpiTbLoadNamespace (
59     void);
```

new/usr/src/common/acpica/components/tables/tbxload.c

2

```
62 /*****
63 *
64 * FUNCTION:      AcpiLoadTables
65 *
66 * PARAMETERS:    None
67 *
68 * RETURN:        Status
69 *
70 * DESCRIPTION:    Load the ACPI tables from the RSDT/XSDT
71 *
72 *****/

74 ACPI_STATUS
75 AcpiLoadTables (
76     void)
77 {
78     ACPI_STATUS          Status;

81     ACPI_FUNCTION_TRACE (AcpiLoadTables);

84     /* Load the namespace from the tables */

86     Status = AcpiTbLoadNamespace ();
87     if (ACPI_FAILURE (Status))
88     {
89         ACPI_EXCEPTION ((AE_INFO, Status,
90             "While loading namespace from ACPI tables"));
91     }

93     return ACPI_STATUS (Status);
94 }

96 ACPI_EXPORT_SYMBOL_INIT (AcpiLoadTables)

99 /*****
100 *
101 * FUNCTION:      AcpiTbLoadNamespace
102 *
103 * PARAMETERS:    None
104 *
105 * RETURN:        Status
106 *
107 * DESCRIPTION:    Load the namespace from the DSDT and all SSDTs/PSDTs found in
108 *                 the RSDT/XSDT.
109 *
110 *****/

112 static ACPI_STATUS
113 AcpiTbLoadNamespace (
114     void)
115 {
116     ACPI_STATUS          Status;
117     UINT32                i;
118     ACPI_TABLE_HEADER     *NewDsdtd;

121     ACPI_FUNCTION_TRACE (TbLoadNamespace);

124     (void) AcpiUtAcquireMutex (ACPI_MTX_TABLES);
```

```

126  /*
127  * Load the namespace. The DSDT is required, but any SSDT and
128  * PSST 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  }

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;

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_CopyDsdtdLocally)
156  {
157      NewDsdtd = AcpiTbCopyDsdtd (ACPI_TABLE_INDEX_DSDT);
158      if (NewDsdtd)
159      {
160          AcpiGbl_DSDT = NewDsdtd;
161      }
162  }

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_OriginalDsdtdHeader, AcpiGbl_DSDT,
169              sizeof (ACPI_TABLE_HEADER));

171  (void) AcpiUtReleaseMutex (ACPI_MTX_TABLES);

173  /* Load and parse tables */

175  Status = AcpiNsLoadTable (ACPI_TABLE_INDEX_DSDT, AcpiGbl_RootNode);
176  if (ACPI_FAILURE (Status))
177  {
178      return ACPI_STATUS (Status);
179  }

181  /* Load any SSDT or PSST tables. Note: Loop leaves tables locked */

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_PSST) ||
190          ACPI_FAILURE (AcpiTbVerifyTable (
191              &AcpiGbl_RootTableList.Tables[i])))

```

```

192  {
193      continue;
194  }

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_DisableSsdtdTableLoad)
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  }

208  /* Ignore errors while loading tables, get as many as possible */

210  (void) AcpiUtReleaseMutex (ACPI_MTX_TABLES);
211  (void) AcpiNsLoadTable (i, AcpiGbl_RootNode);
212  (void) AcpiUtAcquireMutex (ACPI_MTX_TABLES);
213  }

215  ACPI_INFO ((AE_INFO, "All ACPI Tables successfully acquired"));

217  UnlockAndExit:
218  (void) AcpiUtReleaseMutex (ACPI_MTX_TABLES);
219  return ACPI_STATUS (Status);
220  }

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  *****/

240  ACPI_STATUS
241  AcpiLoadTable (
242      ACPI_TABLE_HEADER *Table)
243  {
244      ACPI_STATUS Status;
245      ACPI_TABLE_DESC TableDesc;
246      UINT32 TableIndex;

249  ACPI_FUNCTION_TRACE (AcpiLoadTable);

252  /* Parameter validation */

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     }
397
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     }
409
410     Status = AcpiTbReleaseOwnerId (i);
411     AcpiTbSetTableLoadedFlag (i, FALSE);
412     break;
413 }
414
415 (void) AcpiUtReleaseMutex (ACPI_MTX_INTERPRETER);
416 return ACPI_STATUS (Status);
417 }
418
419 ACPI_EXPORT_SYMBOL (AcpiUnloadParentTable)
```

new/usr/src/common/acpica/components/tables/tbxfoot.c

1

```
*****
8842 Fri Jan 17 07:30:41 2014
new/usr/src/common/acpica/components/tables/tbxfoot.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: tbxfoot - Find the root ACPI table (RSDT)
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __TBXFOOT_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "actables.h"

51 #define _COMPONENT          ACPI_TABLES
52 #define ACPI_MODULE_NAME    ("tbxfoot")

54 /* Local prototypes */

56 static UINT8 *
57 AcpiTbScanMemoryForRsdp (
58     UINT8 *StartAddress,
```

new/usr/src/common/acpica/components/tables/tbxfoot.c

2

```
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
68 static ACPI_STATUS
69 AcpiTbValidateRsdp (
70     ACPI_TABLE_RSDP *Rsdp)
71 {
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))
79     {
80         /* Nope, BAD Signature */
81
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"));
224 ACPI_ERROR ((AE_INFO, "A valid RSDP was not found"));
225 return_ACPI_STATUS (AE_NOT_FOUND);
226 }

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 *
244 static UINT8 *
245 AcpiTbScanMemoryForRsdp (
246     UINT8          *StartAddress,
247     UINT32         Length)
248 {
249     ACPI_STATUS      Status;
250     UINT8            *MemRover;
251     UINT8            *EndAddress;
252
253     ACPI_FUNCTION_TRACE (TbScanMemoryForRsdp);
254
255     EndAddress = StartAddress + Length;
256
257     /* Search from given start address for the requested length */
258
259     for (MemRover = StartAddress; MemRover < EndAddress;
260          MemRover += ACPI_RSDP_SCAN_STEP)
261     {
262         /* The RSDP signature and checksum must both be correct */
263
264         Status = AcpiTbValidateRsdp (ACPI_CAST_PTR (ACPI_TABLE_RSDP, MemRover));
265         if (ACPI_SUCCESS (Status))
266         {
267             /* Sig and checksum valid, we have found a real RSDP */
268
269             ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
270                               "RSDP located at physical address %p\n", MemRover));
271             return_PTR (MemRover);
272         }
273
274         /* No sig match or bad checksum, keep searching */
275     }
276
277     /* Searched entire block, no RSDP was found */
278
279     ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
280                       "Searched entire block from %p, valid RSDP was not found\n",
281                       StartAddress));
282     return_PTR (NULL);
283 }
284 }
```

new/usr/src/common/acpica/components/utilities/utaddress.c

1

```
*****
10352 Fri Jan 17 07:30:42 2014
new/usr/src/common/acpica/components/utilities/utaddress.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: utaddress - OpRegion address range check
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTADDRESS_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acnamesp.h"

51 #define _COMPONENT          ACPI_UTILITIES
52 #define ACPI_MODULE_NAME    ("utaddress")

55 /*****
56 *
57 * FUNCTION:      AcpiUtAddAddressRange
58 *
59 * PARAMETERS:    SpaceId          - Address space ID
```

new/usr/src/common/acpica/components/utilities/utaddress.c

2

```
60 *          Address          - OpRegion start address
61 *          Length          - OpRegion length
62 *          RegionNode      - OpRegion namespace node
63 *
64 * RETURN:       Status
65 *
66 * DESCRIPTION:  Add the Operation Region address range to the global list.
67 *               The only supported Space IDs are Memory and I/O. Called when
68 *               the OpRegion address/length operands are fully evaluated.
69 *
70 * MUTEX:        Locks the namespace
71 *
72 * NOTE: Because this interface is only called when an OpRegion argument
73 * list is evaluated, there cannot be any duplicate RegionNodes.
74 * Duplicate Address/Length values are allowed, however, so that multiple
75 * address conflicts can be detected.
76 *
77 *****/

79 ACPI_STATUS
80 AcpiUtAddAddressRange (
81     ACPI_ADR_SPACE_TYPE    SpaceId,
82     ACPI_PHYSICAL_ADDRESS  Address,
83     UINT32                  Length,
84     ACPI_NAMESPACE_NODE    *RegionNode)
85 {
86     ACPI_ADDRESS_RANGE      *RangeInfo;
87     ACPI_STATUS              Status;

90     ACPI_FUNCTION_TRACE (UtAddAddressRange);

93     if ((SpaceId != ACPI_ADR_SPACE_SYSTEM_MEMORY) &&
94         (SpaceId != ACPI_ADR_SPACE_SYSTEM_IO))
95     {
96         return_ACPI_STATUS (AE_OK);
97     }

99     /* Allocate/init a new info block, add it to the appropriate list */

101     RangeInfo = ACPI_ALLOCATE (sizeof (ACPI_ADDRESS_RANGE));
102     if (!RangeInfo)
103     {
104         return_ACPI_STATUS (AE_NO_MEMORY);
105     }

107     RangeInfo->StartAddress = Address;
108     RangeInfo->EndAddress = (Address + Length - 1);
109     RangeInfo->RegionNode = RegionNode;

111     Status = AcpiUtAcquireMutex (ACPI_MTX_NAMESPACE);
112     if (ACPI_FAILURE (Status))
113     {
114         ACPI_FREE (RangeInfo);
115         return_ACPI_STATUS (Status);
116     }

118     RangeInfo->Next = AcpiGbl_AddressRangeList[SpaceId];
119     AcpiGbl_AddressRangeList[SpaceId] = RangeInfo;

121     ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
122         "\nAdded [%4.4s] address range: 0x%p-0x%p\n",
123         AcpiUtGetNodeName (RangeInfo->RegionNode),
124         ACPI_CAST_PTR (void, Address),
125         ACPI_CAST_PTR (void, RangeInfo->EndAddress)));
```

```

127     (void) AcpiUtReleaseMutex (ACPI_MTX_NAMESPACE);
128     return ACPI_STATUS (AE_OK);
129 }

132 /*****
133  *
134  * FUNCTION:      AcpiUtRemoveAddressRange
135  *
136  * PARAMETERS:    SpaceId          - Address space ID
137  *                RegionNode       - OpRegion namespace node
138  *
139  * RETURN:        None
140  *
141  * DESCRIPTION:    Remove the Operation Region from the global list. The only
142  *                supported Space IDs are Memory and I/O. Called when an
143  *                OpRegion is deleted.
144  *
145  * MUTEX:         Assumes the namespace is locked
146  *
147  *****/

149 void
150 AcpiUtRemoveAddressRange (
151     ACPI_ADR_SPACE_TYPE      SpaceId,
152     ACPI_NAMESPACE_NODE      *RegionNode)
153 {
154     ACPI_ADDRESS_RANGE        *RangeInfo;
155     ACPI_ADDRESS_RANGE        *Prev;

158     ACPI_FUNCTION_TRACE (UtRemoveAddressRange);

161     if ((SpaceId != ACPI_ADR_SPACE_SYSTEM_MEMORY) &&
162         (SpaceId != ACPI_ADR_SPACE_SYSTEM_IO))
163     {
164         return_VOID;
165     }

167     /* Get the appropriate list head and check the list */

169     RangeInfo = Prev = AcpiGbl_AddressRangeList[SpaceId];
170     while (RangeInfo)
171     {
172         if (RangeInfo->RegionNode == RegionNode)
173         {
174             if (RangeInfo == Prev) /* Found at list head */
175             {
176                 AcpiGbl_AddressRangeList[SpaceId] = RangeInfo->Next;
177             }
178             else
179             {
180                 Prev->Next = RangeInfo->Next;
181             }

183             ACPI_DEBUG_PRINT ((ACPI_DB_NAMES,
184                 "\nRemoved [%4.4s] address range: 0x%p-0x%p\n",
185                 AcpiUtGetNodeName (RangeInfo->RegionNode),
186                 ACPI_CAST_PTR (void, RangeInfo->StartAddress),
187                 ACPI_CAST_PTR (void, RangeInfo->EndAddress)));

189             ACPI_FREE (RangeInfo);
190             return_VOID;
191         }

```

```

193     Prev = RangeInfo;
194     RangeInfo = RangeInfo->Next;
195 }

197     return_VOID;
198 }

201 /*****
202  *
203  * FUNCTION:      AcpiUtCheckAddressRange
204  *
205  * PARAMETERS:    SpaceId          - Address space ID
206  *                Address          - Start address
207  *                Length           - Length of address range
208  *                Warn             - TRUE if warning on overlap desired
209  *
210  * RETURN:        Count of the number of conflicts detected. Zero is always
211  *                returned for Space IDs other than Memory or I/O.
212  *
213  * DESCRIPTION:    Check if the input address range overlaps any of the
214  *                ASL operation region address ranges. The only supported
215  *                Space IDs are Memory and I/O.
216  *
217  * MUTEX:         Assumes the namespace is locked.
218  *
219  *****/

221 UINT32
222 AcpiUtCheckAddressRange (
223     ACPI_ADR_SPACE_TYPE      SpaceId,
224     ACPI_PHYSICAL_ADDRESS    Address,
225     UINT32                   Length,
226     BOOLEAN                  Warn)
227 {
228     ACPI_ADDRESS_RANGE        *RangeInfo;
229     ACPI_PHYSICAL_ADDRESS    EndAddress;
230     char                       *Pathname;
231     UINT32                   OverlapCount = 0;

234     ACPI_FUNCTION_TRACE (UtCheckAddressRange);

237     if ((SpaceId != ACPI_ADR_SPACE_SYSTEM_MEMORY) &&
238         (SpaceId != ACPI_ADR_SPACE_SYSTEM_IO))
239     {
240         return_UINT32 (0);
241     }

243     RangeInfo = AcpiGbl_AddressRangeList[SpaceId];
244     EndAddress = Address + Length - 1;

246     /* Check entire list for all possible conflicts */

248     while (RangeInfo)
249     {
250         /*
251          * Check if the requested address/length overlaps this
252          * address range. There are four cases to consider:
253          *
254          * 1) Input address/length is contained completely in the
255          *    address range
256          * 2) Input address/length overlaps range at the range start
257          * 3) Input address/length overlaps range at the range end

```

```

258     * 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 */
264
265         OverlapCount++;
266         if (Warn) /* Optional warning message */
267         {
268             Pathname = AcpiNsGetExternalPathname (RangeInfo->RegionNode);
269
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     }
281
282     RangeInfo = RangeInfo->Next;
283 }
284
285 return_UINT32 (OverlapCount);
286 }
287
288 /*****
289 *
290 * FUNCTION:      AcpiUtDeleteAddressLists
291 *
292 * PARAMETERS:    None
293 *
294 * RETURN:        None
295 *
296 * DESCRIPTION:   Delete all global address range lists (called during
297 *               subsystem shutdown).
298 *
299 * *****/
300
301 void
302 AcpiUtDeleteAddressLists (
303     void)
304 {
305     ACPI_ADDRESS_RANGE *Next;
306     ACPI_ADDRESS_RANGE *RangeInfo;
307     int i;
308
309     /* Delete all elements in all address range lists */
310
311     for (i = 0; i < ACPI_ADDRESS_RANGE_MAX; i++)
312     {
313         Next = AcpiGbl_AddressRangeList[i];
314
315         while (Next)
316         {
317             RangeInfo = Next;
318             Next = RangeInfo->Next;
319             ACPI_FREE (RangeInfo);
320         }
321     }

```

```

324     AcpiGbl_AddressRangeList[i] = NULL;
325 }
326 }

```



new/usr/src/common/acpica/components/utilities/utalloc.c

1

```
*****
10014 Fri Jan 17 07:30:42 2014
new/usr/src/common/acpica/components/utilities/utalloc.c
update to acpica-unix2-20140114
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 - 2014, 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 MERCHANTABILITY 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 __UTALLOC_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acdebug.h"
49
50 #define _COMPONENT          ACPI_UTILITIES
51         ACPI_MODULE_NAME    ("utalloc")
52
53 #if !defined (USE_NATIVE_ALLOCATE_ZEROED)
54 /*****
55 *
56 */
57
```

new/usr/src/common/acpica/components/utilities/utalloc.c

2

```
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 *****/
68
69 void *
70 AcpiOsAllocateZeroed (
71     ACPI_SIZE      Size)
72 {
73     void            *Allocation;
74
75     ACPI_FUNCTION_ENTRY ();
76
77     Allocation = AcpiOsAllocate (Size);
78     if (Allocation)
79     {
80         /* Clear the memory block */
81
82         ACPI_MEMSET (Allocation, 0, Size);
83     }
84
85     return (Allocation);
86 }
87
88 #endif /* !USE_NATIVE_ALLOCATE_ZEROED */
89
90
91
92
93 /*****
94 *
95 * FUNCTION:      AcpiUtCreateCaches
96 *
97 * PARAMETERS:    None
98 *
99 * RETURN:        Status
100 *
101 * DESCRIPTION:    Create all local caches
102 *
103 *****/
104
105 ACPI_STATUS
106 AcpiUtCreateCaches (
107     void)
108 {
109     ACPI_STATUS      Status;
110
111     /* Object Caches, for frequently used objects */
112
113     Status = AcpiOsCreateCache ("Acpi-Namespace", sizeof (ACPI_NAMESPACE_NODE),
114                                ACPI_MAX_NAMESPACE_CACHE_DEPTH, &AcpiGbl_NamespaceCache);
115     if (ACPI_FAILURE (Status))
116     {
117         return (Status);
118     }
119
120     Status = AcpiOsCreateCache ("Acpi-State", sizeof (ACPI_GENERIC_STATE),
121                                ACPI_MAX_STATE_CACHE_DEPTH, &AcpiGbl_StateCache);
122
```

```

123     if (ACPI_FAILURE (Status))
124     {
125         return (Status);
126     }
127
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     }
134
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     }
141
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     }
148
149 #ifdef ACPI_DBG_TRACK_ALLOCATIONS
150     /* Memory allocation lists */
151
152     Status = AcpiUtCreateList ("Acpi-Global", 0,
153                               &AcpiGbl_GlobalList);
154     if (ACPI_FAILURE (Status))
155     {
156         return (Status);
157     }
158
159     Status = AcpiUtCreateList ("Acpi-Namespace", sizeof (ACPI_NAMESPACE_NODE),
160                               &AcpiGbl_NsNodeList);
161     if (ACPI_FAILURE (Status))
162     {
163         return (Status);
164     }
165 #endif
166
167 return (AE_OK);
168 }

```

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;
298
299     /* Parameter validation */
300
301     if (!Buffer || !RequiredLength)
302     {
303         return (AE_BAD_PARAMETER);
304     }
305
306     /*
307      * Buffer->Length is used as both an input and output parameter. Get the
308      * input actual length and set the output required buffer length.
309      */
310     InputBufferLength = Buffer->Length;
311     Buffer->Length = RequiredLength;
312
313     /*
314      * The input buffer length contains the actual buffer length, or the type
315      * of buffer to be allocated by this routine.
316      */
317     switch (InputBufferLength)
318     {
319     case ACPI_NO_BUFFER:
320
321         /* Return the exception (and the required buffer length) */
322
323         return (AE_BUFFER_OVERFLOW);
324
325     case ACPI_ALLOCATE_BUFFER:
326         /*
327          * Allocate a new buffer. We directly call AcpiOsAllocate here to
328          * purposefully bypass the (optionally enabled) internal allocation
329          * tracking mechanism since we only want to track internal
330          * allocations. Note: The caller should use AcpiOsFree to free this
331          * buffer created via ACPI_ALLOCATE_BUFFER.
332          */
333
334         /* Allocate a new buffer */
335
336         Buffer->Pointer = AcpiOsAllocate (RequiredLength);
337         break;
338
339     case ACPI_ALLOCATE_LOCAL_BUFFER:
340
341         /* Allocate a new buffer with local interface to allow tracking */
342
343         Buffer->Pointer = ACPI_ALLOCATE (RequiredLength);
344         break;
345
346     default:
347
348         /* Existing buffer: Validate the size of the buffer */
349
350         if (InputBufferLength < RequiredLength)
351         {
352             return (AE_BUFFER_OVERFLOW);
353         }
354         break;
355     }
356
357     /* 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 }

```

new/usr/src/common/acpica/components/utilities/utbuffer.c

1

```
*****
6838 Fri Jan 17 07:30:42 2014
new/usr/src/common/acpica/components/utilities/utbuffer.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: utbuffer - Buffer dump routines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTBUFFER_C__

46 #include "acpi.h"
47 #include "accommon.h"

49 #define _COMPONENT          ACPI_UTILITIES
50 #define ACPI_MODULE_NAME    ("utbuffer")

53 /*****
54 *
55 * FUNCTION:      AcpiUtDumpBuffer
56 *
57 * PARAMETERS:    Buffer          - Buffer to dump
58 *                  Count         - Amount to dump, in bytes
59 *                  Display        - BYTE, WORD, DWORD, or QWORD display:
60 *                                  DB_BYTE_DISPLAY
```

new/usr/src/common/acpica/components/utilities/utbuffer.c

2

```
61 *                                  DB_WORD_DISPLAY
62 *                                  DB_DWORD_DISPLAY
63 *                                  DB_QWORD_DISPLAY
64 *                                  BaseOffset      - Beginning buffer offset (display only)
65 *
66 * RETURN:        None
67 *
68 * DESCRIPTION:    Generic dump buffer in both hex and ascii.
69 *
70 *****/

72 void
73 AcpiUtDumpBuffer (
74     UINT8          *Buffer,
75     UINT32         Count,
76     UINT32         Display,
77     UINT32         BaseOffset)
78 {
79     UINT32         i = 0;
80     UINT32         j;
81     UINT32         Temp32;
82     UINT8          BufChar;

85     if (!Buffer)
86     {
87         AcpiOsPrintf ("Null Buffer Pointer in DumpBuffer!\n");
88         return;
89     }

91     if ((Count < 4) || (Count & 0x01))
92     {
93         Display = DB_BYTE_DISPLAY;
94     }

96     /* Nasty little dump buffer routine! */

98     while (i < Count)
99     {
100         /* Print current offset */

102         AcpiOsPrintf ("%6.4X: ", (BaseOffset + i));

104         /* Print 16 hex chars */

106         for (j = 0; j < 16; j)
107         {
108             if (i + j >= Count)
109             {
110                 /* Dump fill spaces */

112                 AcpiOsPrintf ("%*s", ((Display * 2) + 1), " ");
113                 j += Display;
114                 continue;
115             }

117             switch (Display)
118             {
119             case DB_BYTE_DISPLAY:
120                 /* Default is BYTE display */

122                 AcpiOsPrintf ("%02X ", Buffer[(ACPI_SIZE) i + j]);
123                 break;

125                 case DB_WORD_DISPLAY:
```

```

127     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 }

```

new/usr/src/common/acpica/components/utilities/utcache.c

1

```
*****
9247 Fri Jan 17 07:30:43 2014
new/usr/src/common/acpica/components/utilities/utcache.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: utcache - local cache allocation routines
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTCACHE_C__
45
46 #include "acpi.h"
47 #include "accommon.h"
48
49 #define _COMPONENT          ACPI_UTILITIES
50 #define ACPI_MODULE_NAME    ("utcache")
51
52
53 #ifdef ACPI_USE_LOCAL_CACHE
54 /*****
55 *
56 * FUNCTION:      AcpiOsCreateCache
57 */
```

new/usr/src/common/acpica/components/utilities/utcache.c

2

```
58 * PARAMETERS:  CacheName      - Ascii name for the cache
59 *              ObjectSize     - Size of each cached object
60 *              MaxDepth       - Maximum depth of the cache (in objects)
61 *              ReturnCache    - Where the new cache object is returned
62 *
63 * RETURN:      Status
64 *
65 * DESCRIPTION: Create a cache object
66 *
67 *****/
68
69 ACPI_STATUS
70 AcpiOsCreateCache (
71     char                *CacheName,
72     UINT16              ObjectSize,
73     UINT16              MaxDepth,
74     ACPI_MEMORY_LIST    **ReturnCache)
75 {
76     ACPI_MEMORY_LIST    *Cache;
77
78     ACPI_FUNCTION_ENTRY ();
79
80     if (!CacheName || !ReturnCache || (ObjectSize < 16))
81     {
82         return (AE_BAD_PARAMETER);
83     }
84
85     /* Create the cache object */
86
87     Cache = AcpiOsAllocate (sizeof (ACPI_MEMORY_LIST));
88     if (!Cache)
89     {
90         return (AE_NO_MEMORY);
91     }
92
93     /* Populate the cache object and return it */
94
95     ACPI_MEMSET (Cache, 0, sizeof (ACPI_MEMORY_LIST));
96     Cache->LinkOffset = 8;
97     Cache->ListName = CacheName;
98     Cache->ObjectSize = ObjectSize;
99     Cache->MaxDepth = MaxDepth;
100
101     *ReturnCache = Cache;
102     return (AE_OK);
103 }
104
105
106 /*****
107 *
108 * FUNCTION:      AcpiOsPurgeCache
109 *
110 * PARAMETERS:  Cache      - Handle to cache object
111 *
112 * RETURN:      Status
113 *
114 * DESCRIPTION: Free all objects within the requested cache.
115 *
116 *****/
117
118
119 ACPI_STATUS
120 AcpiOsPurgeCache (
121     ACPI_MEMORY_LIST    *Cache)
122 {
```

```

123 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 */

```



new/usr/src/common/acpica/components/utilities/utclib.c

1

```
*****
24537 Fri Jan 17 07:30:43 2014
new/usr/src/common/acpica/components/utilities/utclib.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: cmclib - Local implementation of C library functions
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __CMCLIB_C__
47 #include "acpi.h"
48 #include "accommon.h"

50 /*
51 * These implementations of standard C Library routines can optionally be
52 * used if a C library is not available. In general, they are less efficient
53 * than an inline or assembly implementation
54 */

56 #define _COMPONENT          ACPI_UTILITIES
57 #define ACPI_MODULE_NAME    ("cmclib")
```

new/usr/src/common/acpica/components/utilities/utclib.c

2

```
60 #ifndef ACPI_USE_SYSTEM_CLIBRARY

62 #define NEGATIVE    1
63 #define POSITIVE    0

66 /*****
67 *
68 * FUNCTION:      AcpiUtMemcmp (memcmp)
69 *
70 * PARAMETERS:    Buffer1      - First Buffer
71 *                Buffer2      - Second Buffer
72 *                Count       - Maximum # of bytes to compare
73 *
74 * RETURN:        Index where Buffers mismatched, or 0 if Buffers matched
75 *
76 * DESCRIPTION:   Compare two Buffers, with a maximum length
77 *
78 *****/

80 int
81 AcpiUtMemcmp (
82     const char      *Buffer1,
83     const char      *Buffer2,
84     ACPI_SIZE        Count)
85 {
87     for ( ; Count-- && (*Buffer1 == *Buffer2); Buffer1++, Buffer2++)
88     {
89     }

91     return ((Count == ACPI_SIZE_MAX) ? 0 : ((unsigned char) *Buffer1 -
92         (unsigned char) *Buffer2));
93 }

96 /*****
97 *
98 * FUNCTION:      AcpiUtMemcpy (memcpy)
99 *
100 * PARAMETERS:    Dest      - Target of the copy
101 *                Src       - Source buffer to copy
102 *                Count     - Number of bytes to copy
103 *
104 * RETURN:        Dest
105 *
106 * DESCRIPTION:   Copy arbitrary bytes of memory
107 *
108 *****/

110 void *
111 AcpiUtMemcpy (
112     void          *Dest,
113     const void    *Src,
114     ACPI_SIZE      Count)
115 {
116     char          *New = (char *) Dest;
117     char          *Old = (char *) Src;

120     while (Count)
121     {
122         *New = *Old;
123         New++;
124         Old++;
125         Count--;
```

```

126     }

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;

```

```

259  /* Copy the string */
261  for (String = DstString;
262       Count && (Count--, (*String++ = *SrcString++)); )
263  {;}
265  /* Pad with nulls if necessary */
267  while (Count--)
268  {
269      *String = 0;
270      String++;
271  }
273  /* Return original pointer */
275  return (DstString);
276 }

279 /*****
280  *
281  * FUNCTION:      AcpiUtStrcmp (strcmp)
282  *
283  * PARAMETERS:    String1          - First string
284  *                String2          - Second string
285  *
286  * RETURN:        Index where strings mismatched, or 0 if strings matched
287  *
288  * DESCRIPTION:   Compare two null terminated strings
289  *****/
290
292 int
293 AcpiUtStrcmp (
294     const char      *String1,
295     const char      *String2)
296 {
299     for ( ; (*String1 == *String2); String2++)
300     {
301         if (!*String1++)
302         {
303             return (0);
304         }
305     }
307     return ((unsigned char) *String1 - (unsigned char) *String2);
308 }

311 #ifdef ACPI_FUTURE_IMPLEMENTATION
312 /* Not used at this time */
313 /*****
314  *
315  * FUNCTION:      AcpiUtStrchr (strchr)
316  *
317  * PARAMETERS:    String          - Search string
318  *                ch              - character to search for
319  *
320  * RETURN:        Ptr to char or NULL if not found
321  *
322  * DESCRIPTION:   Search a string for a character
323  *****/

```

```

324  *****/
326 char *
327 AcpiUtStrchr (
328     const char      *String,
329     int              ch)
330 {
333     for ( ; (*String); String++)
334     {
335         if ((*String) == (char) ch)
336         {
337             return ((char *) String);
338         }
339     }
341     return (NULL);
342 }
343 #endif

345 /*****
346  *
347  * FUNCTION:      AcpiUtStrncmp (strncmp)
348  *
349  * PARAMETERS:    String1          - First string
350  *                String2          - Second string
351  *                Count            - Maximum # of bytes to compare
352  *
353  * RETURN:        Index where strings mismatched, or 0 if strings matched
354  *
355  * DESCRIPTION:   Compare two null terminated strings, with a maximum length
356  *****/
357

359 int
360 AcpiUtStrncmp (
361     const char      *String1,
362     const char      *String2,
363     ACPI_SIZE       Count)
364 {
367     for ( ; Count-- && (*String1 == *String2); String2++)
368     {
369         if (!*String1++)
370         {
371             return (0);
372         }
373     }
375     return ((Count == ACPI_SIZE_MAX) ? 0 : ((unsigned char) *String1 -
376                                             (unsigned char) *String2));
377 }

380 /*****
381  *
382  * FUNCTION:      AcpiUtStrcat (Strcat)
383  *
384  * PARAMETERS:    DstString        - Target of the copy
385  *                SrcString        - The source string to copy
386  *
387  * RETURN:        DstString
388  *
389  * DESCRIPTION:   Append a null terminated string to a null terminated string

```

```

390 *
391 *****/
393 char *
394 AcpiUtStrcat (
395     char          *DstString,
396     const char    *SrcString)
397 {
398     char          *String;
399
401     /* Find end of the destination string */
402
403     for (String = DstString; *String++; )
404     { ; }
405
406     /* Concatenate the string */
407
408     for (--String; (*String++ = *SrcString++); )
409     { ; }
410
411     return (DstString);
412 }
413
414 /*****
415 *
416 * FUNCTION:      AcpiUtStrncat (strncat)
417 *
418 * PARAMETERS:    DstString      - Target of the copy
419 *                SrcString      - The source string to copy
420 *                Count          - Maximum # of bytes to copy
421 *
422 * RETURN:        DstString
423 *
424 * DESCRIPTION:   Append a null terminated string to a null terminated string,
425 *                with a maximum count.
426 *
427 * *****/
428
429 char *
430 AcpiUtStrncat (
431     char          *DstString,
432     const char    *SrcString,
433     ACPI_SIZE     Count)
434 {
435     char          *String;
436
437     if (Count)
438     {
439         /* Find end of the destination string */
440
441         for (String = DstString; *String++; )
442         { ; }
443
444         /* Concatenate the string */
445
446         for (--String; (*String++ = *SrcString++) && --Count; )
447         { ; }
448
449         /* Null terminate if necessary */
450
451         if (!Count)
452         {
453             *String = 0;
454         }
455     }

```

```

456     }
457 }
458
459 return (DstString);
460 }
461
462 /*****
463 *
464 * FUNCTION:      AcpiUtStrstr (strstr)
465 *
466 * PARAMETERS:    String1      - Target string
467 *                String2      - Substring to search for
468 *
469 * RETURN:        Where substring match starts, Null if no match found
470 *
471 * DESCRIPTION:   Checks if String2 occurs in String1. This is not really a
472 *                full implementation of strstr, only sufficient for command
473 *                matching
474 *
475 * *****/
476
477 char *
478 AcpiUtStrstr (
479     char          *String1,
480     char          *String2)
481 {
482     char          *String;
483
484     if (AcpiUtStrlen (String2) > AcpiUtStrlen (String1))
485     {
486         return (NULL);
487     }
488
489     /* Walk entire string, comparing the letters */
490
491     for (String = String1; *String; )
492     {
493         if (*String2 != *String)
494         {
495             return (NULL);
496         }
497         String2++;
498         String++;
499     }
500
501     return (String1);
502 }
503
504 /*****
505 *
506 * FUNCTION:      AcpiUtStrtoul (strtoul)
507 *
508 * PARAMETERS:    String      - Null terminated string
509 *                Terminator   - Where a pointer to the terminating byte is
510 *                               returned
511 *                Base         - Radix of the string
512 *
513 * RETURN:        Converted value
514 *
515 * DESCRIPTION:   Convert a string into a 32-bit unsigned value.
516 *                Note: use AcpiUtStrtoul64 for 64-bit integers.
517 * *****/

```

```

522 *****/
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;
536
537     /*
538      * Save the value of the pointer to the buffer's first
539      * character, save the current errno value, and then
540      * skip over any white space in the buffer:
541      */
542     StringStart = String;
543     while (ACPI_IS_SPACE (*String) || *String == '\t')
544     {
545         ++String;
546     }
547
548     /*
549      * The buffer may contain an optional plus or minus sign.
550      * If it does, then skip over it but remember what it was:
551      */
552     if (*String == '-')
553     {
554         sign = NEGATIVE;
555         ++String;
556     }
557     else if (*String == '+')
558     {
559         ++String;
560         sign = POSITIVE;
561     }
562     else
563     {
564         sign = POSITIVE;
565     }
566
567     /*
568      * If the input parameter Base is zero, then we need to
569      * determine if it is octal, decimal, or hexadecimal:
570      */
571     if (Base == 0)
572     {
573         if (*String == '0')
574         {
575             if (AcpiUtToLower (*(++String)) == 'x')
576             {
577                 Base = 16;
578                 ++String;
579             }
580             else
581             {
582                 Base = 8;
583             }
584         }
585         else
586         {
587             Base = 10;
588         }
589     }
590     else if (Base < 2 || Base > 36)
591     {
592         /*
593          * The specified Base parameter is not in the domain of
594          * this function:
595          */
596         goto done;
597     }
598
599     /*
600      * For octal and hexadecimal bases, skip over the leading
601      * 0 or 0x, if they are present.
602      */
603     if (Base == 8 && *String == '0')
604     {
605         String++;
606     }
607
608     if (Base == 16 &&
609         *String == '0' &&
610         AcpiUtToLower (*(++String)) == 'x')
611     {
612         String++;
613     }
614
615     /*
616      * Main loop: convert the string to an unsigned long:
617      */
618     while (*String)
619     {
620         if (ACPI_IS_DIGIT (*String))
621         {
622             index = (UINT32) ((UINT8) *String - '0');
623         }
624         else
625         {
626             index = (UINT32) AcpiUtToUpper (*String);
627             if (ACPI_IS_UPPER (index))
628             {
629                 index = index - 'A' + 10;
630             }
631             else
632             {
633                 goto done;
634             }
635         }
636
637         if (index >= Base)
638         {
639             goto done;
640         }
641
642         /*
643          * Check to see if value is out of range:
644          */
645         if (ReturnValue > ((ACPI_UINT32_MAX - (UINT32) index) /
646             (UINT32) Base))
647         {
648             Status = AE_ERROR;
649             ReturnValue = 0;
650             /* reset */
651         }
652         else
653         {
654             ReturnValue = ReturnValue * Base + index;
655         }
656         ++String;
657     }
658
659     *Terminator = String;
660     return ReturnValue;
661 }

```

```

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 }
599
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 }
608
609 if (Base == 16 &&
610     *String == '0' &&
611     AcpiUtToLower (*(++String)) == 'x')
612 {
613     String++;
614 }
615
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     }
637
638     if (index >= Base)
639     {
640         goto done;
641     }
642
643     /*
644      * Check to see if value is out of range:
645      */
646     if (ReturnValue > ((ACPI_UINT32_MAX - (UINT32) index) /
647         (UINT32) Base))
648     {
649         Status = AE_ERROR;
650         ReturnValue = 0;
651         /* reset */
652     }
653     else
654     {
655         ReturnValue = ReturnValue * Base + index;
656     }
657     ++String;
658 }
659
660 *Terminator = String;
661 return ReturnValue;

```

```

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  * PARAMETERS:    c          - Character to convert
701  * RETURN:        Converted character as an int
702  * DESCRIPTION:   Convert character to uppercase
703  *
704  * *****/
705
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  *****/
729
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  *****/
746
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 '&' */
787     _ACPI_PU,          /* 0x27 39 ''' */
788     _ACPI_PU,          /* 0x28 40 '(' */
789     _ACPI_PU,          /* 0x29 41 ')' */
790     _ACPI_PU,          /* 0x2A 42 '*' */
791     _ACPI_PU,          /* 0x2B 43 '+' */
792     _ACPI_PU,          /* 0x2C 44 ',' */
793     _ACPI_PU,          /* 0x2D 45 '-' */
794     _ACPI_PU,          /* 0x2E 46 '.' */
795     _ACPI_PU,          /* 0x2F 47 '/' */
796     _ACPI_XD|_ACPI_DI, /* 0x30 48 '0' */
797     _ACPI_XD|_ACPI_DI, /* 0x31 49 '1' */
798     _ACPI_XD|_ACPI_DI, /* 0x32 50 '2' */
799     _ACPI_XD|_ACPI_DI, /* 0x33 51 '3' */
800     _ACPI_XD|_ACPI_DI, /* 0x34 52 '4' */
801     _ACPI_XD|_ACPI_DI, /* 0x35 53 '5' */
802     _ACPI_XD|_ACPI_DI, /* 0x36 54 '6' */
803     _ACPI_XD|_ACPI_DI, /* 0x37 55 '7' */
804     _ACPI_XD|_ACPI_DI, /* 0x38 56 '8' */
805     _ACPI_XD|_ACPI_DI, /* 0x39 57 '9' */
806     _ACPI_PU,          /* 0x3A 58 ':' */
807     _ACPI_PU,          /* 0x3B 59 ';' */
808     _ACPI_PU,          /* 0x3C 60 '<' */
809     _ACPI_PU,          /* 0x3D 61 '=' */
810     _ACPI_PU,          /* 0x3E 62 '>' */
811     _ACPI_PU,          /* 0x3F 63 '?' */
812     _ACPI_PU,          /* 0x40 64 '@' */
813     _ACPI_XD|_ACPI_UP, /* 0x41 65 'A' */
814     _ACPI_XD|_ACPI_UP, /* 0x42 66 'B' */
815     _ACPI_XD|_ACPI_UP, /* 0x43 67 'C' */
816     _ACPI_XD|_ACPI_UP, /* 0x44 68 'D' */
817     _ACPI_XD|_ACPI_UP, /* 0x45 69 'E' */
818     _ACPI_XD|_ACPI_UP, /* 0x46 70 'F' */
819     _ACPI_UP,          /* 0x47 71 'G' */
820     _ACPI_UP,          /* 0x48 72 'H' */
821     _ACPI_UP,          /* 0x49 73 'I' */
822     _ACPI_UP,          /* 0x4A 74 'J' */
823     _ACPI_UP,          /* 0x4B 75 'K' */
824     _ACPI_UP,          /* 0x4C 76 'L' */
825     _ACPI_UP,          /* 0x4D 77 'M' */
826     _ACPI_UP,          /* 0x4E 78 'N' */
827     _ACPI_UP,          /* 0x4F 79 'O' */
828     _ACPI_UP,          /* 0x50 80 'P' */
829     _ACPI_UP,          /* 0x51 81 'Q' */
830     _ACPI_UP,          /* 0x52 82 'R' */
831     _ACPI_UP,          /* 0x53 83 'S' */
832     _ACPI_UP,          /* 0x54 84 'T' */
833     _ACPI_UP,          /* 0x55 85 'U' */
834     _ACPI_UP,          /* 0x56 86 'V' */
835     _ACPI_UP,          /* 0x57 87 'W' */
836     _ACPI_UP,          /* 0x58 88 'X' */
837     _ACPI_UP,          /* 0x59 89 'Y' */
838     _ACPI_UP,          /* 0x5A 90 'Z' */
839     _ACPI_PU,          /* 0x5B 91 '[' */
840     _ACPI_PU,          /* 0x5C 92 '\' */
841     _ACPI_PU,          /* 0x5D 93 ']' */
842     _ACPI_PU,          /* 0x5E 94 '^' */
843     _ACPI_PU,          /* 0x5F 95 '_' */
844     _ACPI_PU,          /* 0x60 96 '`' */
845     _ACPI_XD|_ACPI_LO, /* 0x61 97 'a' */
846     _ACPI_XD|_ACPI_LO, /* 0x62 98 'b' */
847     _ACPI_XD|_ACPI_LO, /* 0x63 99 'c' */
848     _ACPI_XD|_ACPI_LO, /* 0x64 100 'd' */
849     _ACPI_XD|_ACPI_LO, /* 0x65 101 'e' */
850     _ACPI_XD|_ACPI_LO, /* 0x66 102 'f' */
851     _ACPI_LO,          /* 0x67 103 'g' */

```

```

852     _ACPI_LO,          /* 0x68 104 'h' */
853     _ACPI_LO,          /* 0x69 105 'i' */
854     _ACPI_LO,          /* 0x6A 106 'j' */
855     _ACPI_LO,          /* 0x6B 107 'k' */
856     _ACPI_LO,          /* 0x6C 108 'l' */
857     _ACPI_LO,          /* 0x6D 109 'm' */
858     _ACPI_LO,          /* 0x6E 110 'n' */
859     _ACPI_LO,          /* 0x6F 111 'o' */
860     _ACPI_LO,          /* 0x70 112 'p' */
861     _ACPI_LO,          /* 0x71 113 'q' */
862     _ACPI_LO,          /* 0x72 114 'r' */
863     _ACPI_LO,          /* 0x73 115 's' */
864     _ACPI_LO,          /* 0x74 116 't' */
865     _ACPI_LO,          /* 0x75 117 'u' */
866     _ACPI_LO,          /* 0x76 118 'v' */
867     _ACPI_LO,          /* 0x77 119 'w' */
868     _ACPI_LO,          /* 0x78 120 'x' */
869     _ACPI_LO,          /* 0x79 121 'y' */
870     _ACPI_LO,          /* 0x7A 122 'z' */
871     _ACPI_PU,          /* 0x7B 123 '{' */
872     _ACPI_PU,          /* 0x7C 124 '|' */
873     _ACPI_PU,          /* 0x7D 125 '}' */
874     _ACPI_PU,          /* 0x7E 126 '~' */
875     _ACPI_CN,          /* 0x7F 127 DEL */

877     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, /* 0x80 to 0x8F */
878     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, /* 0x90 to 0x9F */
879     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, /* 0xA0 to 0xAF */
880     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, /* 0xB0 to 0xBF */
881     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, /* 0xC0 to 0xCF */
882     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, /* 0xD0 to 0xDF */
883     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, /* 0xE0 to 0xEF */
884     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, /* 0xF0 to 0xFF */
885     0, /* 0x100 */
886 };

889 #endif /* ACPI_USE_SYSTEM_CLIBRARY */

```

new/usr/src/common/acpica/components/utilities/utcopy.c

1

```
*****
31406 Fri Jan 17 07:30:43 2014
new/usr/src/common/acpica/components/utilities/utcopy.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utcopy - Internal to external object translation utilities
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTCOPY_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acnamesp.h"

51 #define _COMPONENT          ACPI_UTILITIES
52 #define ACPI_MODULE_NAME    ("utcopy")

54 /* Local prototypes */

56 static ACPI_STATUS
57 AcpiUtCopyIsimpleToEsimple (
58     ACPI_OPERAND_OBJECT    *InternalObject,
```

new/usr/src/common/acpica/components/utilities/utcopy.c

2

```
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 }

```

```

251 /*****
252 *
253 * FUNCTION:      AcpiUtCopyIelementToEelement
254 *
255 * PARAMETERS:    ACPI_PKG_CALLBACK
256 *
257 * RETURN:        Status
258 *
259 * DESCRIPTION:   Copy one package element to another package element
260 *
261 *****/

263 static ACPI_STATUS
264 AcpiUtCopyIelementToEelement (
265     UINT8          ObjectType,
266     ACPI_OPERAND_OBJECT *SourceObject,
267     ACPI_GENERIC_STATE *State,
268     void           *Context)
269 {
270     ACPI_STATUS      Status = AE_OK;
271     ACPI_PKG_INFO    *Info = (ACPI_PKG_INFO *) Context;
272     ACPI_SIZE        ObjectSpace;
273     UINT32           ThisIndex;
274     ACPI_OBJECT      *TargetObject;

277     ACPI_FUNCTION_ENTRY ();

280     ThisIndex = State->Pkg.Index;
281     TargetObject = (ACPI_OBJECT *)
282         &((ACPI_OBJECT *) (State->Pkg.DestObject))->Package.Elements[ThisIndex];

284     switch (ObjectType)
285     {
286     case ACPI_COPY_TYPE_SIMPLE:

287         /*
288          * This is a simple or null object
289          */
290         Status = AcpiUtCopyIsimpleToEsimple (SourceObject,
291             TargetObject, Info->FreeSpace, &ObjectSpace);
292         if (ACPI_FAILURE (Status))
293         {
294             return (Status);
295         }
296         break;

298     case ACPI_COPY_TYPE_PACKAGE:

299         /*
300          * Build the package object
301          */
302         TargetObject->Type = ACPI_TYPE_PACKAGE;
303         TargetObject->Package.Count = SourceObject->Package.Count;
304         TargetObject->Package.Elements =
305             ACPI_CAST_PTR (ACPI_OBJECT, Info->FreeSpace);

307         /*
308          * Pass the new package object back to the package walk routine
309          */
310         State->Pkg.ThisTargetObj = TargetObject;

312         /*

```

```

313         * Save space for the array of objects (Package elements)
314         * update the buffer length counter
315         */
316         ObjectSpace = ACPI_ROUND_UP_TO_NATIVE_WORD (
317             (ACPI_SIZE) TargetObject->Package.Count *
318             sizeof (ACPI_OBJECT));
319         break;

321     default:

322     default:
323         return (AE_BAD_PARAMETER);
324     }

326     Info->FreeSpace += ObjectSpace;
327     Info->Length += ObjectSpace;
328     return (Status);
329 }

    unchanged_portion_omitted

456 /*****
457 *
458 * FUNCTION:      AcpiUtCopyEsimpleToIsimple
459 *
460 * PARAMETERS:    ExternalObject      - The external object to be converted
461 *               RetInternalObject    - Where the internal object is returned
462 *
463 * RETURN:        Status
464 *
465 * DESCRIPTION:   This function copies an external object to an internal one.
466 *               NOTE: Pointers can be copied, we don't need to copy data.
467 *               (The pointers have to be valid in our address space no matter
468 *               what we do with them!)
469 *
470 *****/

472 static ACPI_STATUS
473 AcpiUtCopyEsimpleToIsimple (
474     ACPI_OBJECT      *ExternalObject,
475     ACPI_OPERAND_OBJECT **RetInternalObject)
476 {
477     ACPI_OPERAND_OBJECT *InternalObject;

480     ACPI_FUNCTION_TRACE (UtCopyEsimpleToIsimple);

483     /*
484     * Simple types supported are: String, Buffer, Integer
485     */
486     switch (ExternalObject->Type)
487     {
488     case ACPI_TYPE_STRING:
489     case ACPI_TYPE_BUFFER:
490     case ACPI_TYPE_INTEGER:
491     case ACPI_TYPE_LOCAL_REFERENCE:

493         InternalObject = AcpiUtCreateInternalObject (
494             (UINT8) ExternalObject->Type);
495         if (!InternalObject)
496         {
497             return ACPI_STATUS (AE_NO_MEMORY);
498         }
499         break;

```

```

501     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         /* An incoming reference is defined to be a namespace node */
578         /* TBD: should validate incoming handle */

569         InternalObject->Reference.Class = ACPI_REFCLASS_REFOF;
570         InternalObject->Reference.Object = ExternalObject->Reference.Handle;
580         InternalObject->Reference.Class = ACPI_REFCLASS_NAME;
581         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_

```

new/usr/src/common/acpica/components/utilities/utdebug.c

1

```
*****
19080 Fri Jan 17 07:30:44 2014
new/usr/src/common/acpica/components/utilities/utdebug.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utdebug - Debug print/trace routines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTDEBUG_C__
45 #define EXPORT_ACPI_INTERFACES

47 #include "acpi.h"
48 #include "accommon.h"

50 #define _COMPONENT          ACPI_UTILITIES
51 #define ACPI_MODULE_NAME    ("utdebug")

54 #ifndef ACPI_DEBUG_OUTPUT

56 static ACPI_THREAD_ID      AcpiGbl_PrevThreadId = (ACPI_THREAD_ID) 0xFFFFFFFF;
```

new/usr/src/common/acpica/components/utilities/utdebug.c

2

```
57 static char      *AcpiGbl_FnEntryStr = "----Entry";
58 static char      *AcpiGbl_FnExitStr  = "----Exit-";

60 /* Local prototypes */

62 static const char *
63 AcpiUtTrimFunctionName (
64     const char      *FunctionName);

67 /*****
68 *
69 * FUNCTION:      AcpiUtInitStackPtrTrace
70 *
71 * PARAMETERS:    None
72 *
73 * RETURN:        None
74 *
75 * DESCRIPTION:   Save the current CPU stack pointer at subsystem startup
76 *
77 *****/

79 void
80 AcpiUtInitStackPtrTrace (
81     void)
82 {
83     ACPI_SIZE      CurrentSp;

86     AcpiGbl_EntryStackPointer = &CurrentSp;
87 }

90 /*****
91 *
92 * FUNCTION:      AcpiUtTrackStackPtr
93 *
94 * PARAMETERS:    None
95 *
96 * RETURN:        None
97 *
98 * DESCRIPTION:   Save the current CPU stack pointer
99 *
100 *****/

102 void
103 AcpiUtTrackStackPtr (
104     void)
105 {
106     ACPI_SIZE      CurrentSp;

109     if (&CurrentSp < AcpiGbl_LowestStackPointer)
110     {
111         AcpiGbl_LowestStackPointer = &CurrentSp;
112     }

114     if (AcpiGbl_NestingLevel > AcpiGbl_Deepnesting)
115     {
116         AcpiGbl_Deepnesting = AcpiGbl_NestingLevel;
117     }
118 }

121 /*****
122 *
```

```

123 * FUNCTION:      AcpiUtTrimFunctionName
124 *
125 * PARAMETERS:    FunctionName      - Ascii string containing a procedure name
126 *
127 * RETURN:        Updated pointer to the function name
128 *
129 * DESCRIPTION:    Remove the "Acpi" prefix from the function name, if present.
130 *                This allows compiler macros such as __FUNCTION__ to be used
131 *                with no change to the debug output.
132 *
133 *****/
134
135 static const char *
136 AcpiUtTrimFunctionName (
137     const char      *FunctionName)
138 {
139     /* All Function names are longer than 4 chars, check is safe */
140
141     if (*(ACPI_CAST_PTR (UINT32, FunctionName)) == ACPI_PREFIX_MIXED)
142     {
143         /* This is the case where the original source has not been modified */
144
145         return (FunctionName + 4);
146     }
147
148     if (*(ACPI_CAST_PTR (UINT32, FunctionName)) == ACPI_PREFIX_LOWER)
149     {
150         /* This is the case where the source has been 'linuxized' */
151
152         return (FunctionName + 5);
153     }
154
155     return (FunctionName);
156 }
157
158
159 /*****
160 *
161 * FUNCTION:      AcpiDebugPrint
162 *
163 * PARAMETERS:    RequestedDebugLevel - Requested debug print level
164 *                LineNumber          - Caller's line number (for error output)
165 *                FunctionName         - Caller's procedure name
166 *                ModuleName           - Caller's module name
167 *                ComponentId          - Caller's component ID
168 *                Format                - Printf format field
169 *                ...                  - Optional printf arguments
170 *
171 * RETURN:        None
172 *
173 * DESCRIPTION:    Print error message with prefix consisting of the module name,
174 *                line number, and component ID.
175 *
176 *
177 *****/
178
179 void ACPI_INTERNAL_VAR_XFACE
180 AcpiDebugPrint (
181     UINT32          RequestedDebugLevel,
182     UINT32          LineNumber,
183     const char      *FunctionName,
184     const char      *ModuleName,
185     UINT32          ComponentId,
186     const char      *Format,
187     ...)
188 {

```

```

189     ACPI_THREAD_ID  ThreadId;
190     va_list         args;
191
192
193     /* Check if debug output enabled */
194
195     if (!ACPI_IS_DEBUG_ENABLED (RequestedDebugLevel, ComponentId))
196     {
197         return;
198     }
199
200     /*
201     * Thread tracking and context switch notification
202     */
203     ThreadId = AcpiOsGetThreadId ();
204     if (ThreadId != AcpiGbl_PrevThreadId)
205     {
206         if (ACPI_LV_THREADS & AcpiDbgLevel)
207         {
208             AcpiOsPrintf (
209                 "\n**** Context Switch from TID %u to TID %u ****\n\n",
210                 (UINT32) AcpiGbl_PrevThreadId, (UINT32) ThreadId);
211         }
212
213         AcpiGbl_PrevThreadId = ThreadId;
214         AcpiGbl_NestingLevel = 0;
215     }
216
217     /*
218     * Display the module name, current line number, thread ID (if requested),
219     * current procedure nesting level, and the current procedure name
220     */
221     AcpiOsPrintf ("%9s-%04ld ", ModuleName, LineNumber);
222
223 #ifdef ACPI_APPLICATION
224     /*
225     * For AcpiExec/iASL only, emit the thread ID and nesting level.
226     * Note: nesting level is really only useful during a single-thread
227     * execution. Otherwise, multiple threads will keep resetting the
228     * level.
229     */
230     if (ACPI_LV_THREADS & AcpiDbgLevel)
231     {
232         AcpiOsPrintf ("[%u] ", (UINT32) ThreadId);
233     }
234
235     AcpiOsPrintf ("[%02ld] ", AcpiGbl_NestingLevel);
236 #endif
237
238     AcpiOsPrintf ("%s-%22.22s: ", AcpiUtTrimFunctionName (FunctionName));
239
240     va_start (args, Format);
241     AcpiOsVprintf (Format, args);
242     va_end (args);
243 }
244
245 ACPI_EXPORT_SYMBOL (AcpiDebugPrint)
246
247
248 /*****
249 *
250 * FUNCTION:      AcpiDebugPrintRaw
251 *
252 * PARAMETERS:    RequestedDebugLevel - Requested debug print level
253 *                LineNumber          - Caller's line number
254 *                FunctionName         - Caller's procedure name

```

```

255 *      ModuleName      - Caller's module name
256 *      ComponentId     - Caller's component ID
257 *      Format          - Printf format field
258 *      ...             - Optional printf arguments
259 *
260 * RETURN:      None
261 *
262 * DESCRIPTION: Print message with no headers. Has same interface as
263 *              DebugPrint so that the same macros can be used.
264 *
265 *****/
267 void ACPI_INTERNAL_VAR_XFACE
268 AcpiDebugPrintRaw (
269     UINT32      RequestedDebugLevel,
270     UINT32      LineNumber,
271     const char  *FunctionName,
272     const char  *ModuleName,
273     UINT32      ComponentId,
274     const char  *Format,
275     ...)
276 {
277     va_list      args;
278
279     /* Check if debug output enabled */
280
281     if (!ACPI_IS_DEBUG_ENABLED (RequestedDebugLevel, ComponentId))
282     {
283         return;
284     }
285
286     va_start (args, Format);
287     AcpiOsVprintf (Format, args);
288     va_end (args);
289 }
290
292 ACPI_EXPORT_SYMBOL (AcpiDebugPrintRaw)
293
295 /*****
296 *
297 * FUNCTION:      AcpiUtTrace
298 *
299 * PARAMETERS:   LineNumber      - Caller's line number
300 *               FunctionName    - Caller's procedure name
301 *               ModuleName      - Caller's module name
302 *               ComponentId     - Caller's component ID
303 *
304 * RETURN:      None
305 *
306 * DESCRIPTION: Function entry trace. Prints only if TRACE_FUNCTIONS bit is
307 *              set in DebugLevel
308 *
309 *****/
311 void
312 AcpiUtTrace (
313     UINT32      LineNumber,
314     const char  *FunctionName,
315     const char  *ModuleName,
316     UINT32      ComponentId)
317 {
318
319     AcpiGbl_NestingLevel++;
320     AcpiUtTrackStackPtr ();

```

```

322     /* Check if enabled up-front for performance */
323
324     if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
325     {
326         AcpiDebugPrint (ACPI_LV_FUNCTIONS,
327             LineNumber, FunctionName, ModuleName, ComponentId,
328             "%s\n", AcpiGbl_FnEntryStr);
329     }
330 }
331
332 ACPI_EXPORT_SYMBOL (AcpiUtTrace)
333
335 /*****
336 *
337 * FUNCTION:      AcpiUtTracePtr
338 *
339 * PARAMETERS:   LineNumber      - Caller's line number
340 *               FunctionName    - Caller's procedure name
341 *               ModuleName      - Caller's module name
342 *               ComponentId     - Caller's component ID
343 *               Pointer         - Pointer to display
344 *
345 * RETURN:      None
346 *
347 * DESCRIPTION: Function entry trace. Prints only if TRACE_FUNCTIONS bit is
348 *              set in DebugLevel
349 *
350 *****/
352 void
353 AcpiUtTracePtr (
354     UINT32      LineNumber,
355     const char  *FunctionName,
356     const char  *ModuleName,
357     UINT32      ComponentId,
358     void        *Pointer)
359 {
360
361     AcpiGbl_NestingLevel++;
362     AcpiUtTrackStackPtr ();
363
364     /* Check if enabled up-front for performance */
365
366     if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
367     {
368         AcpiDebugPrint (ACPI_LV_FUNCTIONS,
369             LineNumber, FunctionName, ModuleName, ComponentId,
370             "%s %p\n", AcpiGbl_FnEntryStr, Pointer);
371     }
372 }
373
375 /*****
376 *
377 * FUNCTION:      AcpiUtTraceStr
378 *
379 * PARAMETERS:   LineNumber      - Caller's line number
380 *               FunctionName    - Caller's procedure name
381 *               ModuleName      - Caller's module name
382 *               ComponentId     - Caller's component ID
383 *               String          - Additional string to display
384 *
385 * RETURN:      None
386 *

```



```

387 * DESCRIPTION: Function entry trace. Prints only if TRACE_FUNCTIONS bit is
388 *             set in DebugLevel
389 *
390 *****/

392 void
393 AcpiUtTraceStr (
394     UINT32          LineNumber,
395     const char      *FunctionName,
396     const char      *ModuleName,
397     UINT32          ComponentId,
398     char            *String)
399 {
401     AcpiGbl_NestingLevel++;
402     AcpiUtTrackStackPtr ();

404     /* Check if enabled up-front for performance */

406     if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
407     {
408         AcpiDebugPrint (ACPI_LV_FUNCTIONS,
409             LineNumber, FunctionName, ModuleName, ComponentId,
410             "%s %s\n", AcpiGbl_FnEntryStr, String);
411     }
412 }

415 *****/
416 *
417 * FUNCTION:     AcpiUtTraceU32
418 *
419 * PARAMETERS:  LineNumber      - Caller's line number
420 *              FunctionName    - Caller's procedure name
421 *              ModuleName      - Caller's module name
422 *              ComponentId     - Caller's component ID
423 *              Integer         - Integer to display
424 *
425 * RETURN:      None
426 *
427 * DESCRIPTION: Function entry trace. Prints only if TRACE_FUNCTIONS bit is
428 *             set in DebugLevel
429 *
430 *****/

432 void
433 AcpiUtTraceU32 (
434     UINT32          LineNumber,
435     const char      *FunctionName,
436     const char      *ModuleName,
437     UINT32          ComponentId,
438     UINT32          Integer)
439 {
441     AcpiGbl_NestingLevel++;
442     AcpiUtTrackStackPtr ();

444     /* Check if enabled up-front for performance */

446     if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
447     {
448         AcpiDebugPrint (ACPI_LV_FUNCTIONS,
449             LineNumber, FunctionName, ModuleName, ComponentId,
450             "%s %08X\n", AcpiGbl_FnEntryStr, Integer);
451     }
452 }

```

```

455 *****/
456 *
457 * FUNCTION:     AcpiUtExit
458 *
459 * PARAMETERS:  LineNumber      - Caller's line number
460 *              FunctionName    - Caller's procedure name
461 *              ModuleName      - Caller's module name
462 *              ComponentId     - Caller's component ID
463 *
464 * RETURN:      None
465 *
466 * DESCRIPTION: Function exit trace. Prints only if TRACE_FUNCTIONS bit is
467 *             set in DebugLevel
468 *
469 *****/

471 void
472 AcpiUtExit (
473     UINT32          LineNumber,
474     const char      *FunctionName,
475     const char      *ModuleName,
476     UINT32          ComponentId)
477 {
479     /* Check if enabled up-front for performance */

481     if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
482     {
483         AcpiDebugPrint (ACPI_LV_FUNCTIONS,
484             LineNumber, FunctionName, ModuleName, ComponentId,
485             "%s\n", AcpiGbl_FnExitStr);
486     }

488     if (AcpiGbl_NestingLevel)
489     {
490         AcpiGbl_NestingLevel--;
491     }
492 }

494 ACPI_EXPORT_SYMBOL (AcpiUtExit)

497 *****/
498 *
499 * FUNCTION:     AcpiUtStatusExit
500 *
501 * PARAMETERS:  LineNumber      - Caller's line number
502 *              FunctionName    - Caller's procedure name
503 *              ModuleName      - Caller's module name
504 *              ComponentId     - Caller's component ID
505 *              Status          - Exit status code
506 *
507 * RETURN:      None
508 *
509 * DESCRIPTION: Function exit trace. Prints only if TRACE_FUNCTIONS bit is
510 *             set in DebugLevel. Prints exit status also.
511 *
512 *****/

514 void
515 AcpiUtStatusExit (
516     UINT32          LineNumber,
517     const char      *FunctionName,
518     const char

```

```

519     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 * FUNCTION:      AcpiUtValueExit
554 *
555 * PARAMETERS:   LineNumber      - Caller's line number
556 *               FunctionName    - Caller's procedure name
557 *               ModuleName      - Caller's module name
558 *               ComponentId     - Caller's component ID
559 *               Value           - Value to be printed with exit msg
560 *
561 * RETURN:       None
562 *
563 * DESCRIPTION:  Function exit trace. Prints only if TRACE_FUNCTIONS bit is
564 *               set in DebugLevel. Prints exit value also.
565 *
566 * *****/
567
568 void
569 AcpiUtValueExit (
570     UINT32             LineNumber,
571     const char         *FunctionName,
572     const char         *ModuleName,
573     UINT32             ComponentId,
574     UINT64             Value)
575 {
576
577     /* Check if enabled up-front for performance */
578
579     if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
580     {
581         AcpiDebugPrint (ACPI_LV_FUNCTIONS,
582                         LineNumber, FunctionName, ModuleName, ComponentId,
583                         "%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 * FUNCTION:      AcpiUtPtrExit
599 *
600 * PARAMETERS:   LineNumber      - Caller's line number
601 *               FunctionName    - Caller's procedure name
602 *               ModuleName      - Caller's module name
603 *               ComponentId     - Caller's component ID
604 *               Ptr             - Pointer to display
605 *
606 * RETURN:       None
607 *
608 * DESCRIPTION:  Function exit trace. Prints only if TRACE_FUNCTIONS bit is
609 *               set in DebugLevel. Prints exit value also.
610 *
611 * *****/
612
613 void
614 AcpiUtPtrExit (
615     UINT32             LineNumber,
616     const char         *FunctionName,
617     const char         *ModuleName,
618     UINT32             ComponentId,
619     void*              Ptr)
620 {
621
622     /* Check if enabled up-front for performance */
623
624     if (ACPI_IS_DEBUG_ENABLED (ACPI_LV_FUNCTIONS, ComponentId))
625     {
626         AcpiDebugPrint (ACPI_LV_FUNCTIONS,
627                         LineNumber, FunctionName, ModuleName, ComponentId,
628                         "%s %p\n", AcpiGbl_FnExitStr, Ptr);
629     }
630
631     if (AcpiGbl_NestingLevel)
632     {
633         AcpiGbl_NestingLevel--;
634     }
635 }
636
637 #endif

```

new/usr/src/common/acpica/components/utilities/utdecode.c

1

```
*****
16487 Fri Jan 17 07:30:44 2014
new/usr/src/common/acpica/components/utilities/utdecode.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utdecode - Utility decoding routines (value-to-string)
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTDECODE_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acnamesp.h"

50 #define _COMPONENT          ACPI_UTILITIES
51 #define ACPI_MODULE_NAME    ("utdecode")

54 /*****
55 *
56 * FUNCTION:      AcpiFormatException
57 *
58 * PARAMETERS:    Status          - The ACPI_STATUS code to be formatted
```

new/usr/src/common/acpica/components/utilities/utdecode.c

2

```
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_NEWSCOPE,        /* 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 */
85     ACPI_NS_NEWSCOPE | ACPI_NS_LOCAL, /* 25 Resource Desc */
86     ACPI_NS_NEWSCOPE | ACPI_NS_LOCAL, /* 26 Resource Field */
87     ACPI_NS_NEWSCOPE,        /* 27 Scope */
88     ACPI_NS_NORMAL,          /* 28 Extra */
89     ACPI_NS_NORMAL,          /* 29 Data */
90     ACPI_NS_NORMAL,          /* 30 Invalid */
91 };

```

unchanged\_portion\_omitted

```

126 /*****
127  *
128  * FUNCTION:      AcpiUtGetRegionName
129  *
130  * PARAMETERS:    Space ID          - ID for the region
131  *
132  * RETURN:        Decoded region SpaceId name
133  *
134  * DESCRIPTION:    Translate a Space ID into a name string (Debug only)
135  *
136  *****/

```

138 /\* Region type decoding \*/

```

140 const char      *AcpiGbl_RegionTypes[ACPI_NUM_PREDEFINED_REGIONS] =
141 {

```

```

142     "SystemMemory",    /* 0x00 */
143     "SystemIO",        /* 0x01 */
144     "PCI_Config",      /* 0x02 */
145     "EmbeddedControl", /* 0x03 */
146     "SMBus",           /* 0x04 */
147     "SystemCMOS",      /* 0x05 */
148     "PCIBARTarget",    /* 0x06 */
149     "IPMI",            /* 0x07 */
150     "GeneralPurposeIo", /* 0x08 */
151     "GenericSerialBus", /* 0x09 */
152     "PCC",             /* 0x0A */
153 };

```

unchanged\_portion\_omitted

```

516 /*****
517  *
518  * FUNCTION:      AcpiUtGetNotifyName
519  *
520  * PARAMETERS:    NotifyValue      - Value from the Notify() request
521  *
522  * RETURN:        Decoded name for the notify value
523  *
524  * DESCRIPTION:    Translate a Notify Value to a notify namestring.
525  *
526  *****/

```

528 /\* Names for Notify() values, used for debug output \*/

```

530 static const char      *AcpiGbl_NotifyValueNames[ACPI_NOTIFY_MAX + 1] =
568 static const char

```

```

531 {
532     /* 00 */ "Bus Check",
533     /* 01 */ "Device Check",
534     /* 02 */ "Device Wake",
535     /* 03 */ "Eject Request",
536     /* 04 */ "Device Check Light",
537     /* 05 */ "Frequency Mismatch",
538     /* 06 */ "Bus Mode Mismatch",
539     /* 07 */ "Power Fault",
540     /* 08 */ "Capabilities Check",
541     /* 09 */ "Device PLD Check",
542     /* 10 */ "Reserved",
543     /* 11 */ "System Locality Update",
544     /* 12 */ "Shutdown Request"
545 };

```

```

547 const char *
548 AcpiUtGetNotifyName (
549     UINT32          NotifyValue)
550 {

```

```

552     if (NotifyValue <= ACPI_NOTIFY_MAX)
553     {
554         return (AcpiGbl_NotifyValueNames[NotifyValue]);
555     }
556     else if (NotifyValue <= ACPI_MAX_SYS_NOTIFY)
557     {
558         return ("Reserved");
559     }
560     else if (NotifyValue <= ACPI_MAX_DEVICE_SPECIFIC_NOTIFY)
561     {
562         return ("Device Specific");
563     }
564     else
565     {
566         return ("Hardware Specific");
567     }
568 }

```

unchanged\_portion\_omitted

new/usr/src/common/acpica/components/utilities/utdelete.c

1

```
*****
23081 Fri Jan 17 07:30:44 2014
new/usr/src/common/acpica/components/utilities/utdelete.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utdelete - object deletion and reference count utilities
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTDELETE_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acinterp.h"
49 #include "acnamesp.h"
50 #include "acevents.h"

53 #define _COMPONENT          ACPI_UTILITIES
54         ACPI_MODULE_NAME    ("utdelete")

56 /* Local prototypes */

58 static void
```

new/usr/src/common/acpica/components/utilities/utdelete.c

2

```
59 AcpiUtDeleteInternalObj (
60     ACPI_OPERAND_OBJECT *Object);

62 static void
63 AcpiUtUpdateRefCount (
64     ACPI_OPERAND_OBJECT *Object,
65     UINT32 Action);

68 /*****
69 *
70 * FUNCTION:      AcpiUtDeleteInternalObj
71 *
72 * PARAMETERS:    Object          - Object to be deleted
73 *
74 * RETURN:        None
75 *
76 * DESCRIPTION:   Low level object deletion, after reference counts have been
77 *                updated (All reference counts, including sub-objects!)
78 *
79 *****/

81 static void
82 AcpiUtDeleteInternalObj (
83     ACPI_OPERAND_OBJECT *Object)
84 {
85     void *ObjPointer = NULL;
86     ACPI_OPERAND_OBJECT *HandlerDesc;
87     ACPI_OPERAND_OBJECT *SecondDesc;
88     ACPI_OPERAND_OBJECT *NextDesc;
89     ACPI_OPERAND_OBJECT **LastObjPtr;

92     ACPI_FUNCTION_TRACE_PTR (UtDeleteInternalObj, Object);

95     if (!Object)
96     {
97         return_VOID;
98     }

100     /*
101     * Must delete or free any pointers within the object that are not
102     * actual ACPI objects (for example, a raw buffer pointer).
103     */
104     switch (Object->Common.Type)
105     {
106     case ACPI_TYPE_STRING:

108         ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "**** String %p, ptr %p\n",
109             Object, Object->String.Pointer));

111         /* Free the actual string buffer */

113         if (!(Object->Common.Flags & APOBJ_STATIC_POINTER))
114         {
115             /* But only if it is NOT a pointer into an ACPI table */

117             ObjPointer = Object->String.Pointer;
118         }
119         break;

121     case ACPI_TYPE_BUFFER:

123         ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "**** Buffer %p, ptr %p\n",
```

```

124     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;
254
255         /* Remove the region object from the handler's list */
256
257         while (NextDesc)
258         {
259             if (NextDesc == Object)
260             {
261                 *LastObjPtr = NextDesc->Region.Next;
262                 break;
263             }
264
265             /* Walk the linked list of handler */
266
267             LastObjPtr = &NextDesc->Region.Next;
268             NextDesc = NextDesc->Region.Next;
269         }
270
271         if (HandlerDesc->AddressSpace.HandlerFlags &
272             ACPI_ADDR_HANDLER_DEFAULT_INSTALLED)
273         {
274             /* Deactivate region and free region context */
275
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
284             AcpiUtRemoveReference (HandlerDesc);
285         }
286
287         /* Now we can free the Extra object */
288
289         AcpiUtDeleteObjectDesc (SecondDesc);
290     }
291     break;
292
293 case ACPI_TYPE_BUFFER_FIELD:
294
295     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
296         "***** Buffer Field %p\n", Object));
297
298     SecondDesc = AcpiNsGetSecondaryObject (Object);
299     if (SecondDesc)
300     {
301         AcpiUtDeleteObjectDesc (SecondDesc);
302     }
303     break;
304
305 case ACPI_TYPE_LOCAL_BANK_FIELD:
306
307     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
308         "***** Bank Field %p\n", Object));
309
310     SecondDesc = AcpiNsGetSecondaryObject (Object);
311     if (SecondDesc)

```

```

313     {
314         AcpiUtDeleteObjectDesc (SecondDesc);
315     }
316     break;
317
318 default:
319
320     default:
321     break;
322
323 /* Free any allocated memory (pointer within the object) found above */
324
325 if (ObjPointer)
326 {
327     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "Deleting Object Subptr %p\n",
328         ObjPointer));
329     ACPI_FREE (ObjPointer);
330 }
331
332 /* Now the object can be safely deleted */
333
334 ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "Deleting Object %p [%s]\n",
335     Object, AcpiUtGetObjectTypeName (Object)));
336
337 AcpiUtDeleteObjectDesc (Object);
338 return_VOID;
339 }
340
341 /*****
342 *
343 * FUNCTION:      AcpiUtDeleteInternalObjectList
344 *
345 * PARAMETERS:    ObjList          - Pointer to the list to be deleted
346 *
347 * RETURN:        None
348 *
349 * DESCRIPTION:   This function deletes an internal object list, including both
350 *                simple objects and package objects
351 *
352 * *****/
353
354 void
355 AcpiUtDeleteInternalObjectList (
356     ACPI_OPERAND_OBJECT **ObjList)
357 {
358     ACPI_OPERAND_OBJECT **InternalObj;
359
360     ACPI_FUNCTION_ENTRY ();
361     ACPI_FUNCTION_TRACE (UtDeleteInternalObjectList);
362
363     /* Walk the null-terminated internal list */
364
365     for (InternalObj = ObjList; *InternalObj; InternalObj++)
366     {
367         AcpiUtRemoveReference (*InternalObj);
368     }
369
370     /* Free the combined parameter pointer list and object array */
371
372     ACPI_FREE (ObjList);
373     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)
385 *
386 * RETURN:       None. Sets new reference count within the object
387 * RETURN:       New ref count
388 * DESCRIPTION:  Modify the reference count for an internal acpi object
389 * DESCRIPTION:  Modify the ref count and return it.
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;
400     UINT16 Count;
401     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.
413     * Perform the reference count action (increment, decrement, force delete)
414     */
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;
424         NewCount++;
425         Object->Common.ReferenceCount = NewCount;
426         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, [Decrement]\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, [Decrement]\n", Object, NewCount));
458     }

460     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
461         "Obj %p Type %.2X Refs %.2X [Decrement]\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;

```



```
new/usr/src/common/acpica/components/utilities/utdelete.c
```

```

530 while (Object)
531 {
532     /* Make sure that this isn't a namespace handle */
533
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);
539     }
540
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->Common.Notify.NotifyList[i];
558             while (PrevObject)
559             {
560                 NextObject = PrevObject->Notify.Next[i];
561                 AcpiUtUpdateRefCount (PrevObject, Action);
562                 PrevObject = NextObject;
563             }
564         }
565
566         /* Update the notify objects for these types (if present) */
567
568         AcpiUtUpdateRefCount (Object->Common.Notify.SystemNotify, Action);
569         AcpiUtUpdateRefCount (Object->Common.Notify.DeviceNotify, Action);
570         break;
571
572     case ACPI_TYPE_PACKAGE:
573         /*
574          * We must update all the sub-objects of the package,
575          * each of whom may have their own sub-objects.
576          */
577         for (i = 0; i < Object->Package.Count; i++)
578         {
579             /*
580              * Null package elements are legal and can be simply
581              * ignored.
582              * Push each element onto the stack for later processing.
583              * Note: There can be null elements within the package,
584              * these are simply ignored
585              */
586             NextObject = Object->Package.Elements[i];
587             if (!NextObject)
588             {
589                 continue;
590             }
591
592             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);
604     Object->Package.Elements[i], Action, &StateList);
605     if (ACPI_FAILURE (Status))
606     {
607         goto ErrorExit;
608     }
609     break;
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);
727     ACPI_FUNCTION_TRACE_PTR (UtAddReference, Object);
729     /* Ensure that we have a valid object */
731     if (!AcpiUtValidInternalObject (Object))
732     {
733         return;
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;
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);
765     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.
770     */
771     if (!Object ||
772         (ACPI_GET_DESCRIPTOR_TYPE (Object) == ACPI_DESC_TYPE_NAMED))
774     {

```

```

775     return;
776     return_VOID;
777 }
778 /* Ensure that we have a valid object */
780 if (!AcpiUtValidInternalObject (Object))
781 {
782     return;
783     return_VOID;
784 }
785 ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS,
786     "Obj %p Current Refs=%X [To Be Decrementd]\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;
796     return_VOID;
797 }

```

new/usr/src/common/acpica/components/utilities/uterror.c

1

```
*****
10595 Fri Jan 17 07:30:45 2014
new/usr/src/common/acpica/components/utilities/uterror.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: uterror - Various internal error/warning output functions
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTERROR_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acnamesp.h"

51 #define _COMPONENT          ACPI_UTILITIES
52 #define ACPI_MODULE_NAME    ("uterror")

55 /*
56 * This module contains internal error functions that may
57 * be configured out.
58 */
59 #if !defined (ACPI_NO_ERROR_MESSAGES)
```

new/usr/src/common/acpica/components/utilities/uterror.c

2

```
61 /*****
62 *
63 * FUNCTION:      AcpiUtPredefinedWarning
64 *
65 * PARAMETERS:    ModuleName      - Caller's module name (for error output)
66 *                LineNumber      - Caller's line number (for error output)
67 *                Pathname        - Full pathname to the node
68 *                NodeFlags       - From Namespace node for the method/object
69 *                Format          - Printf format string + additional args
70 *
71 * RETURN:        None
72 *
73 * DESCRIPTION:   Warnings for the predefined validation module. Messages are
74 *                only emitted the first time a problem with a particular
75 *                method/object is detected. This prevents a flood of error
76 *                messages for methods that are repeatedly evaluated.
77 *
78 *****/

80 void ACPI_INTERNAL_VAR_XFACE
81 AcpiUtPredefinedWarning (
82     const char      *ModuleName,
83     UINT32          LineNumber,
84     char             *Pathname,
85     UINT8            NodeFlags,
86     const char      *Format,
87     ...)
88 {
89     va_list          ArgList;

92     /*
93     * Warning messages for this method/object will be disabled after the
94     * first time a validation fails or an object is successfully repaired.
95     */
96     if (NodeFlags & ANOBJ_EVALUATED)
97     {
98         return;
99     }

101     AcpiOsPrintf (ACPI_MSG_WARNING "%s: ", Pathname);

103     va_start (ArgList, Format);
104     AcpiOsVprintf (Format, ArgList);
105     ACPI_MSG_SUFFIX;
106     va_end (ArgList);
107 }

110 /*****
111 *
112 * FUNCTION:      AcpiUtPredefinedInfo
113 *
114 * PARAMETERS:    ModuleName      - Caller's module name (for error output)
115 *                LineNumber      - Caller's line number (for error output)
116 *                Pathname        - Full pathname to the node
117 *                NodeFlags       - From Namespace node for the method/object
118 *                Format          - Printf format string + additional args
119 *
120 * RETURN:        None
121 *
122 * DESCRIPTION:   Info messages for the predefined validation module. Messages
123 *                are only emitted the first time a problem with a particular
124 *                method/object is detected. This prevents a flood of
125 *                messages for methods that are repeatedly evaluated.
126 */
```

```

127 *****/
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 */

```

new/usr/src/common/acpica/components/utilities/uteval.c

1

```
*****
11359 Fri Jan 17 07:30:45 2014
new/usr/src/common/acpica/components/utilities/uteval.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: uteval - Object evaluation
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTEVAL_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acnamesp.h"

51 #define _COMPONENT          ACPI_UTILITIES
52 #define ACPI_MODULE_NAME    ("uteval")

55 /*****
56 *
57 * FUNCTION:      AcpiUtEvaluateObject
58 * *****/
```

new/usr/src/common/acpica/components/utilities/uteval.c

2

```
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 ExpectedReturnTypes,
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;
99     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);

128         Status = AE_NOT_EXIST;
129     }

131     goto Cleanup;
132 }

134 /* Map the return object type to the bitmapped type */

136 switch ((Info->ReturnObject)->Common.Type)
137 {
138     case ACPI_TYPE_INTEGER:

140         ReturnBtype = ACPI_BTYPE_INTEGER;
141         break;

143     case ACPI_TYPE_BUFFER:

145         ReturnBtype = ACPI_BTYPE_BUFFER;
146         break;

148     case ACPI_TYPE_STRING:

150         ReturnBtype = ACPI_BTYPE_STRING;
151         break;

153     case ACPI_TYPE_PACKAGE:

155         ReturnBtype = ACPI_BTYPE_PACKAGE;
156         break;

158     default:

160         ReturnBtype = 0;
161         break;
162 }

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 }

176 /* Is the return object one of the expected types? */

178 if (!(ExpectedReturnBtypes & ReturnBtype))
179 {
180     ACPI_ERROR_METHOD ("Return object type is incorrect",
181         PrefixNode, Path, AE_TYPE);

183     ACPI_ERROR ((AE_INFO,
184         "Type returned from %s was incorrect: %s, expected Btypes: 0x%X",
185         Path, AcpiUtGetObjectTypeName (Info->ReturnObject),
186         ExpectedReturnBtypes));

188     /* On error exit, we must delete the return object */

```

```

190     AcpiUtRemoveReference (Info->ReturnObject);
191     Status = AE_TYPE;
192     goto Cleanup;
193 }

195 /* Object type is OK, return it */

197 *ReturnDesc = Info->ReturnObject;

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).
265  *                *Flags.
266  *
267  *                NOTE: Internal function, no parameter validation
268  *****/

270 ACPI_STATUS
271 AcpiUtExecute_STA (
272     ACPI_NAMESPACE_NODE *DeviceNode,
273     UINT32 *Flags)
274 {
275     ACPI_OPERAND_OBJECT *ObjDesc;
276     ACPI_STATUS Status;

279     ACPI_FUNCTION_TRACE (UtExecute_STA);

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)));

297             *Flags = ACPI_UINT32_MAX;
298             Status = AE_OK;
299         }

301     return ACPI_STATUS (Status);

```



**new/usr/src/common/acpica/components/utilities/uteval.c**

**5**

```
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
```

new/usr/src/common/acpica/components/utilities/utexcep.c

1

```
*****
5238 Fri Jan 17 07:30:46 2014
new/usr/src/common/acpica/components/utilities/utexcep.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: utexcep - Exception code support
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTEXCEP_C__
46 #define EXPORT_ACPI_INTERFACES

48 #define ACPI_DEFINE_EXCEPTION_TABLE
49 #include "acpi.h"
50 #include "accommon.h"

53 #define _COMPONENT          ACPI_UTILITIES
54       ACPI_MODULE_NAME    ("utexcep")

57 /*****
58 *
59 * FUNCTION:      AcpiFormatException
```

new/usr/src/common/acpica/components/utilities/utexcep.c

2

```
60 *
61 * PARAMETERS:      Status          - The ACPI_STATUS code to be formatted
62 *
63 * RETURN:          A string containing the exception text. A valid pointer is
64 *                  always returned.
65 *
66 * DESCRIPTION:      This function translates an ACPI exception into an ASCII
67 *                  string. Returns "unknown status" string for invalid codes.
68 *
69 *****/

71 const char *
72 AcpiFormatException (
73     ACPI_STATUS          Status)
74 {
75     const ACPI_EXCEPTION_INFO *Exception;

78     ACPI_FUNCTION_ENTRY ();

81     Exception = AcpiUtValidateException (Status);
82     if (!Exception)
83     {
84         /* Exception code was not recognized */

86         ACPI_ERROR ((AE_INFO,
87                     "Unknown exception code: 0x%8.8X", Status));

89         return ("UNKNOWN_STATUS_CODE");
90     }

92     return (Exception->Name);
93 }

95 ACPI_EXPORT_SYMBOL (AcpiFormatException)

98 /*****
99 *
100 * FUNCTION:      AcpiUtValidateException
101 *
102 * PARAMETERS:      Status          - The ACPI_STATUS code to be formatted
103 *
104 * RETURN:          A string containing the exception text. NULL if exception is
105 *                  not valid.
106 *
107 * DESCRIPTION:      This function validates and translates an ACPI exception into
108 *                  an ASCII string.
109 *
110 *****/

112 const ACPI_EXCEPTION_INFO *
113 AcpiUtValidateException (
114     ACPI_STATUS          Status)
115 {
116     UINT32                SubStatus;
117     const ACPI_EXCEPTION_INFO *Exception = NULL;

120     ACPI_FUNCTION_ENTRY ();

123     /*
124     * Status is composed of two parts, a "type" and an actual code
125     */
```

```
126     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 }
```

new/usr/src/common/acpica/components/utilities/utgglobal.c

1

```
*****
14312 Fri Jan 17 07:30:46 2014
new/usr/src/common/acpica/components/utilities/utgglobal.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: utgglobal - Global variables for the ACPI subsystem
4 *
5 *****/
6 /
7 *
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTGGLOBAL_C__
45 #define EXPORT_ACPI_INTERFACES
46 #define DEFINE_ACPI_GLOBALS
47
48 #include "acpi.h"
49 #include "accommon.h"
50
51 #define _COMPONENT ACPI_UTILITIES
52 #define ACPI_MODULE_NAME ("utgglobal")
53
54
55 /*****
56 *
57 * Static global variable initialization.
58 */
```

new/usr/src/common/acpica/components/utilities/utgglobal.c

2

```
58 *
59 *****/
60
61 /*
62 * We want the debug switches statically initialized so they
63 * are already set when the debugger is entered.
64 */
65
66 /* Debug switch - level and trace mask */
67
68 #ifdef ACPI_DEBUG_OUTPUT
69 UINT32 AcpiDbgLevel = ACPI_DEBUG_DEFAULT;
70 #else
71 UINT32 AcpiDbgLevel = ACPI_NORMAL_DEFAULT;
72 #endif
73
74 /* Debug switch - layer (component) mask */
75
76 UINT32 AcpiDbgLayer = ACPI_COMPONENT_DEFAULT;
77 UINT32 AcpiGbl_NestingLevel = 0;
78
79 /* Debugger globals */
80
81 BOOLEAN AcpiGbl_DbTerminateThreads = FALSE;
82 BOOLEAN AcpiGbl_AbortMethod = FALSE;
83 BOOLEAN AcpiGbl_MethodExecuting = FALSE;
84
85 /* System flags */
86
87 UINT32 AcpiGbl_StartupFlags = 0;
88
89 /* System starts uninitialized */
90
91 BOOLEAN AcpiGbl_Shutdown = TRUE;
92
93 const char *AcpiGbl_SleepStateNames[ACPI_S_STATE_COUNT] =
94 {
95     "\\S0_",
96     "\\S1_",
97     "\\S2_",
98     "\\S3_",
99     "\\S4_",
100     "\\S5_"
101 };
102
103 #ifndef __UNCHANGED_PORTION_OMITTED__
104
105 #if (!ACPI_REduced_HARDWARE)
106 /*****
107 *
108 * Event and Hardware globals
109 *
110 *****/
111
112 ACPI_BIT_REGISTER_INFO AcpiGbl_BitRegisterInfo[ACPI_NUM_BITREG] =
113 {
114     /* Name Parent Register Regi
115
116     /* ACPI_BITREG_TIMER_STATUS */ {ACPI_REGISTER_PM1_STATUS, ACPI_B
117     /* ACPI_BITREG_BUS_MASTER_STATUS */ {ACPI_REGISTER_PM1_STATUS, ACPI_B
118     /* ACPI_BITREG_GLOBAL_LOCK_STATUS */ {ACPI_REGISTER_PM1_STATUS, ACPI_B
119     /* ACPI_BITREG_POWER_BUTTON_STATUS */ {ACPI_REGISTER_PM1_STATUS, ACPI_B
120     /* ACPI_BITREG_SLEEP_BUTTON_STATUS */ {ACPI_REGISTER_PM1_STATUS, ACPI_B
121     /* ACPI_BITREG_RT_CLOCK_STATUS */ {ACPI_REGISTER_PM1_STATUS, ACPI_B
122     /* ACPI_BITREG_WAKE_STATUS */ {ACPI_REGISTER_PM1_STATUS, ACPI_B
123     /* ACPI_BITREG_PCIEXP_WAKE_STATUS */ {ACPI_REGISTER_PM1_STATUS, ACPI_B
124
125     }
126
127 */
```

```

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.
218  *
219  * DESCRIPTION:   Init ACPICA globals. All globals that require specific
220  *               initialization should be initialized here!
221  *
222  *****/
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;
295 AcpiGbl_SystemNotify.Handler    = NULL;
296 AcpiGbl_DeviceNotify.Handler    = NULL;
297 AcpiGbl_ExceptionHandler        = NULL;
298 AcpiGbl_InitHandler             = NULL;
299 AcpiGbl_TableHandler            = NULL;
300 AcpiGbl_InterfaceHandler        = NULL;
301 AcpiGbl_GlobalEventHandler      = NULL;

303 /* 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_DSMT                    = 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

new/usr/src/common/acpica/components/utilities/utids.c

1

```
*****
13188 Fri Jan 17 07:30:46 2014
new/usr/src/common/acpica/components/utilities/utids.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utids - support for device IDs - HID, UID, CID
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTIDS_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acinterp.h"

51 #define _COMPONENT          ACPI_UTILITIES
52 #define ACPI_MODULE_NAME    ("utids")

55 /*****
56 *
57 * FUNCTION:      AcpiUtExecute_HID
58 *

```

new/usr/src/common/acpica/components/utilities/utids.c

2

```
59 * PARAMETERS:  DeviceNode      - Node for the device
60 *              ReturnId        - Where the string HID is returned
61 *
62 * RETURN:      Status
63 *
64 * DESCRIPTION: Executes the _HID control method that returns the hardware
65 * ID of the device. The HID is either an 32-bit encoded EISAID
66 * Integer or a String. A string is always returned. An EISAID
67 * is converted to a string.
68 *
69 * NOTE: Internal function, no parameter validation
70 *
71 *****/

73 ACPI_STATUS
74 AcpiUtExecute_HID (
75     ACPI_NAMESPACE_NODE *DeviceNode,
76     ACPI_PNP_DEVICE_ID **ReturnId,
77     ACPI_DEVICE_ID **ReturnId)
78 {
79     ACPI_OPERAND_OBJECT *ObjDesc;
80     ACPI_PNP_DEVICE_ID *Hid;
81     ACPI_DEVICE_ID *Hid;
82     UINT32 Length;
83     ACPI_STATUS Status;

84     ACPI_FUNCTION_TRACE (UtExecute_HID);

87     Status = AcpiUtEvaluateObject (DeviceNode, METHOD_NAME__HID,
88                                     ACPI_BTTYPE_INTEGER | ACPI_BTTYPE_STRING, &ObjDesc);
89     if (ACPI_FAILURE (Status))
90     {
91         return ACPI_STATUS (Status);
92     }

94     /* Get the size of the String to be returned, includes null terminator */

96     if (ObjDesc->Common.Type == ACPI_TYPE_INTEGER)
97     {
98         Length = ACPI_EISAID_STRING_SIZE;
99     }
100     else
101     {
102         Length = ObjDesc->String.Length + 1;
103     }

105     /* Allocate a buffer for the HID */

107     Hid = ACPI_ALLOCATE_ZEROED (sizeof (ACPI_PNP_DEVICE_ID) + (ACPI_SIZE) Length);
108     if (!Hid)
109     {
110         Status = AE_NO_MEMORY;
111         goto Cleanup;
112     }

114     /* Area for the string starts after PNP_DEVICE_ID struct */
114     /* Area for the string starts after DEVICE_ID struct */

116     Hid->String = ACPI_ADD_PTR (char, Hid, sizeof (ACPI_PNP_DEVICE_ID));
116     Hid->String = ACPI_ADD_PTR (char, Hid, sizeof (ACPI_DEVICE_ID));

118     /* Convert EISAID to a string or simply copy existing string */

```

new/usr/src/common/acpica/components/utilities/utids.c

3

```
120 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_BTTYPE_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 */
```

new/usr/src/common/acpica/components/utilities/utids.c

4

```
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)
235 {
236     ACPI_OPERAND_OBJECT *ObjDesc;
237     ACPI_PNP_DEVICE_ID *UId;
238     ACPI_DEVICE_ID *UId;
239     UINT32 Length;
240     ACPI_STATUS Status;

242     ACPI_FUNCTION_TRACE (UtExecute_UID);

245     Status = AcpiUtEvaluateObject (DeviceNode, METHOD_NAME__UID,
246     ACPI_BTTYPE_INTEGER | ACPI_BTTYPE_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)
327 {
328     ACPI_OPERAND_OBJECT **CidObjects;
329     ACPI_OPERAND_OBJECT *ObjDesc;
330     ACPI_PNP_DEVICE_ID_LIST *CidList;
331     ACPI_DEVICE_ID_LIST *CidList;
332     char *NextIdString;
333     UINT32 StringAreaSize;
334     UINT32 Length;
335     UINT32 CidListSize;
336     ACPI_STATUS Status;
337     UINT32 Count;
338     i;

340     ACPI_FUNCTION_TRACE (UtExecute_CID);

343     /* Evaluate the _CID method for this device */

345     Status = AcpiUtEvaluateObject (DeviceNode, METHOD_NAME__CID,
346                                   ACPI_BTTYPE_INTEGER | ACPI_BTTYPE_STRING | ACPI_BTTYPE_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:
378         StringAreaSize += ACPI_EISAID_STRING_SIZE;
379         break;
380
381     case ACPI_TYPE_STRING:
382         StringAreaSize += CidObjects[i]->String.Length + 1;
383         break;
384
385     default:
386         Status = AE_TYPE;
387         goto Cleanup;
388     }
389 }
390
391 /*
392  * Now that we know the length of the CIDs, allocate return buffer:
393  * 1) Size of the base structure +
394  * 2) Size of the CID PNP_DEVICE_ID array +
395  * 2) Size of the CID DEVICE_ID array +
396  * 3) Size of the actual CID strings
397  */
398 CidListSize = sizeof (ACPI_PNP_DEVICE_ID_LIST) +
399 ((Count - 1) * sizeof (ACPI_PNP_DEVICE_ID)) +
400 CidListSize = sizeof (ACPI_DEVICE_ID_LIST) +
401 ((Count - 1) * sizeof (ACPI_DEVICE_ID)) +
402 StringAreaSize;
403
404 CidList = ACPI_ALLOCATE_ZEROED (CidListSize);
405 if (!CidList)
406 {
407     Status = AE_NO_MEMORY;
408     goto Cleanup;
409 }
410
411 /* Area for CID strings starts after the CID PNP_DEVICE_ID array */
412 /* Area for CID strings starts after the CID DEVICE_ID array */
413
414 NextIdString = ACPI_CAST_PTR (char, CidList->Ids) +
415 ((ACPI_SIZE) Count * sizeof (ACPI_PNP_DEVICE_ID));
416 ((ACPI_SIZE) Count * sizeof (ACPI_DEVICE_ID));
417
418 /* Copy/convert the CIDs to the return buffer */
419 for (i = 0; i < Count; i++)
420 {
421     if (CidObjects[i]->Common.Type == ACPI_TYPE_INTEGER)
422     {
423         /* Convert the Integer (EISAID) CID to a string */
424
425         AcpiExEisaIdToString (NextIdString, CidObjects[i]->Integer.Value);
426         Length = ACPI_EISAID_STRING_SIZE;
427     }
428     else /* ACPI_TYPE_STRING */
429     {
430         /* Copy the String CID from the returned object */
431
432         ACPI_STRCPY (NextIdString, CidObjects[i]->String.Pointer);
433         Length = CidObjects[i]->String.Length + 1;
434     }
435
436     CidList->Ids[i].String = NextIdString;
437     CidList->Ids[i].Length = Length;
438     NextIdString += Length;

```

```

438     }
439
440     /* Finish the CID list */
441
442     CidList->Count = Count;
443     CidList->ListSize = CidListSize;
444     *ReturnCidList = CidList;
445
446 Cleanup:
447
448     /* On exit, we must delete the _CID return object */
449
450     AcpiUtRemoveReference (ObjDesc);
451     return ACPI_STATUS (Status);
452 }

```

new/usr/src/common/acpica/components/utilities/utinit.c

1

```
*****
5073 Fri Jan 17 07:30:47 2014
new/usr/src/common/acpica/components/utilities/utinit.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utinit - Common ACPI subsystem initialization
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTINIT_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acnamesp.h"
50 #include "acevents.h"
51 #include "actables.h"

53 #define _COMPONENT ACPI_UTILITIES
54 ACPI_MODULE_NAME ("utinit")

56 /* Local prototypes */

58 static void AcpiUtTerminate (
```

new/usr/src/common/acpica/components/utilities/utinit.c

2

```
59 void);

61 #if (!ACPI_REDUCED_HARDWARE)

63 static void
64 AcpiUtFreeGpeLists (
65 void);

67 #else

69 #define AcpiUtFreeGpeLists()
70 #endif /* !ACPI_REDUCED_HARDWARE */

73 #if (!ACPI_REDUCED_HARDWARE)
74 /*****
75 *
76 * FUNCTION: AcpiUtFreeGpeLists
77 * FUNCTION: AcpiUtTerminate
78 * PARAMETERS: none
79 *
80 * RETURN: none
81 *
82 * DESCRIPTION: Free global GPE lists
83 * DESCRIPTION: Free global memory
84 *****/

86 static void
87 AcpiUtFreeGpeLists (
88 void)
89 {
90 ACPI_GPE_BLOCK_INFO *GpeBlock;
91 ACPI_GPE_BLOCK_INFO *NextGpeBlock;
92 ACPI_GPE_XRUPT_INFO *GpeXruptInfo;
93 ACPI_GPE_XRUPT_INFO *NextGpeXruptInfo;

96 /* Free global GPE blocks and related info structures */

98 GpeXruptInfo = AcpiGbl_GpeXruptListHead;
99 while (GpeXruptInfo)
100 {
101 GpeBlock = GpeXruptInfo->GpeBlockListHead;
102 while (GpeBlock)
103 {
104 NextGpeBlock = GpeBlock->Next;
105 ACPI_FREE (GpeBlock->EventInfo);
106 ACPI_FREE (GpeBlock->RegisterInfo);
107 ACPI_FREE (GpeBlock);

109 GpeBlock = NextGpeBlock;
110 }
111 NextGpeXruptInfo = GpeXruptInfo->Next;
112 ACPI_FREE (GpeXruptInfo);
113 GpeXruptInfo = NextGpeXruptInfo;
114 }
115 }

116 #endif /* !ACPI_REDUCED_HARDWARE */
```

```
119 /*****
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 }
141
142 _unchanged_portion_omitted_
```

new/usr/src/common/acpica/components/utilities/utlock.c

1

```
*****
5989 Fri Jan 17 07:30:47 2014
new/usr/src/common/acpica/components/utilities/utlock.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utlock - Reader/Writer lock interfaces
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __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 *****/
```

new/usr/src/common/acpica/components/utilities/utlock.c

2

```
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_
```

new/usr/src/common/acpica/components/utilities/utmath.c

1

```
*****
11267 Fri Jan 17 07:30:48 2014
new/usr/src/common/acpica/components/utilities/utmath.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utmath - Integer math support routines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTMATH_C__

47 #include "acpi.h"
48 #include "accommon.h"

51 #define _COMPONENT          ACPI_UTILITIES
52 #define 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 */
```

new/usr/src/common/acpica/components/utilities/utmath.c

2

```
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
```

new/usr/src/common/acpica/components/utilities/utmisc.c

1

```
*****
13127 Fri Jan 17 07:30:48 2014
new/usr/src/common/acpica/components/utilities/utmisc.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utmisc - common utility procedures
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTMISC_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acnamesp.h"

52 #define _COMPONENT          ACPI_UTILITIES
53 #define ACPI_MODULE_NAME    ("utmisc")

56 /*****
57 *
58 * FUNCTION:      AcpiUtIsPciRootBridge
59 *

```

new/usr/src/common/acpica/components/utilities/utmisc.c

2

```
60 * PARAMETERS:  Id          - The HID/CID in string format
61 *
62 * RETURN:      TRUE if the Id is a match for a PCI/PCI-Express Root Bridge
63 *
64 * DESCRIPTION: Determine if the input ID is a PCI Root Bridge ID.
65 *
66 *****/

68 BOOLEAN
69 AcpiUtIsPciRootBridge (
70     char                *Id)
71 {
72
73     /*
74      * Check if this is a PCI root bridge.
75      * ACPI 3.0+: check for a PCI Express root also.
76      */
77     if (!(ACPI_STRCMP (Id,
78         PCI_ROOT_HID_STRING)) ||
79
80         !(ACPI_STRCMP (Id,
81             PCI_EXPRESS_ROOT_HID_STRING)))
82     {
83         return (TRUE);
84     }
85
86     return (FALSE);
87 }

90 /*****
91 *
92 * FUNCTION:      AcpiUtIsAmlTable
93 *
94 * PARAMETERS:    Table      - An ACPI table
95 *
96 * RETURN:      TRUE if table contains executable AML; FALSE otherwise
97 *
98 * DESCRIPTION: Check ACPI Signature for a table that contains AML code.
99 *              Currently, these are DSDT,SSDT,PSDT. All other table types are
100 *              data tables that do not contain AML code.
101 *
102 *****/

104 BOOLEAN
105 AcpiUtIsAmlTable (
106     ACPI_TABLE_HEADER    *Table)
107 {
108
109     /* These are the only tables that contain executable AML */
110
111     if (ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_DSDT) ||
112         ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_PSDT) ||
113         ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_SSDT))
114     {
115         return (TRUE);
116     }
117
118     return (FALSE);
119 }

122 /*****
123 *
124 * FUNCTION:      AcpiUtDwordByteSwap
125 *

```

```

126 * PARAMETERS: Value          - Value to be converted
127 *
128 * RETURN:      UINT32 integer with bytes swapped
129 *
130 * DESCRIPTION: Convert a 32-bit value to big-endian (swap the bytes)
131 *
132 *****/

134 UINT32
135 AcpiUtDwordByteSwap (
136     UINT32          Value)
137 {
138     union
139     {
140         UINT32      Value;
141         UINT8       Bytes[4];
142     } Out;
143     union
144     {
145         UINT32      Value;
146         UINT8       Bytes[4];
147     } In;

150     ACPI_FUNCTION_ENTRY ();

153     In.Value = Value;

155     Out.Bytes[0] = In.Bytes[3];
156     Out.Bytes[1] = In.Bytes[2];
157     Out.Bytes[2] = In.Bytes[1];
158     Out.Bytes[3] = In.Bytes[0];

160     return (Out.Value);
161 }

164 /*****
165 *
166 * FUNCTION:    AcpiUtSetIntegerWidth
167 *
168 * PARAMETERS:  Revision          From DSDT header
169 *
170 * RETURN:      None
171 *
172 * DESCRIPTION: Set the global integer bit width based upon the revision
173 *               of the DSDT. For Revision 1 and 0, Integers are 32 bits.
174 *               For Revision 2 and above, Integers are 64 bits. Yes, this
175 *               makes a difference.
176 *
177 *****/

179 void
180 AcpiUtSetIntegerWidth (
181     UINT8          Revision)
182 {
183     if (Revision < 2)
184     {
185         /* 32-bit case */

188         AcpiGbl_IntegerBitWidth    = 32;
189         AcpiGbl_IntegerNybbleWidth = 8;
190         AcpiGbl_IntegerByteWidth   = 4;
191     }

```

```

192     else
193     {
194         /* 64-bit case (ACPI 2.0+) */

196         AcpiGbl_IntegerBitWidth    = 64;
197         AcpiGbl_IntegerNybbleWidth = 16;
198         AcpiGbl_IntegerByteWidth   = 8;
199     }
200 }

203 /*****
204 *
205 * FUNCTION:    AcpiUtCreateUpdateStateAndPush
206 *
207 * PARAMETERS:  Object          - Object to be added to the new state
208 *               Action          - Increment/Decrement
209 *               StateList       - List the state will be added to
210 *
211 * RETURN:      Status
212 *
213 * DESCRIPTION: Create a new state and push it
214 *
215 *****/

217 ACPI_STATUS
218 AcpiUtCreateUpdateStateAndPush (
219     ACPI_OPERAND_OBJECT *Object,
220     UINT16              Action,
221     ACPI_GENERIC_STATE **StateList)
222 {
223     ACPI_GENERIC_STATE *State;

226     ACPI_FUNCTION_ENTRY ();

229     /* Ignore null objects; these are expected */

231     if (!Object)
232     {
233         return (AE_OK);
234     }

236     State = AcpiUtCreateUpdateState (Object, Action);
237     if (!State)
238     {
239         return (AE_NO_MEMORY);
240     }

242     AcpiUtPushGenericState (StateList, State);
243     return (AE_OK);
244 }

247 /*****
248 *
249 * FUNCTION:    AcpiUtWalkPackageTree
250 *
251 * PARAMETERS:  SourceObject     - The package to walk
252 *               TargetObject     - Target object (if package is being copied)
253 *               WalkCallback     - Called once for each package element
254 *               Context          - Passed to the callback function
255 *
256 * RETURN:      Status
257 *

```



```

258 * DESCRIPTION: Walk through a package
259 *
260 *****/

262 ACPI_STATUS
263 AcpiUtWalkPackageTree (
264     ACPI_OPERAND_OBJECT *SourceObject,
265     void *TargetObject,
266     ACPI_PKG_CALLBACK WalkCallback,
267     void *Context)
268 {
269     ACPI_STATUS Status = AE_OK;
270     ACPI_GENERIC_STATE *StateList = NULL;
271     ACPI_GENERIC_STATE *State;
272     UINT32 ThisIndex;
273     ACPI_OPERAND_OBJECT *ThisSourceObj;

276     ACPI_FUNCTION_TRACE (UtWalkPackageTree);

279     State = AcpiUtCreatePkgState (SourceObject, TargetObject, 0);
280     if (!State)
281     {
282         return_ACPI_STATUS (AE_NO_MEMORY);
283     }

285     while (State)
286     {
287         /* Get one element of the package */

289         ThisIndex = State->Pkg.Index;
290         ThisSourceObj = (ACPI_OPERAND_OBJECT *)
291             State->Pkg.SourceObject->Package.Elements[ThisIndex];

293         /*
294          * Check for:
295          * 1) An uninitialized package element. It is completely
296          *    legal to declare a package and leave it uninitialized
297          * 2) Not an internal object - can be a namespace node instead
298          * 3) Any type other than a package. Packages are handled in else
299          *    case below.
300          */
301         if ((!ThisSourceObj) ||
302             (ACPI_GET_DESCRIPTOR_TYPE (ThisSourceObj) != ACPI_DESC_TYPE_OPERAND)
303             (ThisSourceObj->Common.Type != ACPI_TYPE_PACKAGE))
304         {
305             Status = WalkCallback (ACPI_COPY_TYPE_SIMPLE, ThisSourceObj,
306                                   State, Context);
307             if (ACPI_FAILURE (Status))
308             {
309                 return_ACPI_STATUS (Status);
310             }

312             State->Pkg.Index++;
313             while (State->Pkg.Index >= State->Pkg.SourceObject->Package.Count)
314             {
315                 /*
316                  * We've handled all of the objects at this level, This means
317                  * that we have just completed a package. That package may
318                  * have contained one or more packages itself.
319                  *
320                  * Delete this state and pop the previous state (package).
321                  */
322                 AcpiUtDeleteGenericState (State);
323                 State = AcpiUtPopGenericState (&StateList);

```

```

325         /* Finished when there are no more states */

327         if (!State)
328         {
329             /*
330              * We have handled all of the objects in the top level
331              * package just add the length of the package objects
332              * and exit
333              */
334             return_ACPI_STATUS (AE_OK);
335         }

337         /*
338          * Go back up a level and move the index past the just
339          * completed package object.
340          */
341         State->Pkg.Index++;
342     }
343 }
344 else
345 {
346     /* This is a subobject of type package */

348     Status = WalkCallback (ACPI_COPY_TYPE_PACKAGE, ThisSourceObj,
349                           State, Context);
350     if (ACPI_FAILURE (Status))
351     {
352         return_ACPI_STATUS (Status);
353     }

355     /*
356      * Push the current state and create a new one
357      * The callback above returned a new target package object.
358      */
359     AcpiUtPushGenericState (&StateList, State);
360     State = AcpiUtCreatePkgState (ThisSourceObj,
361                                   State->Pkg.ThisTargetObj, 0);

362     if (!State)
363     {
364         /* Free any stacked Update State objects */

366         while (StateList)
367         {
368             State = AcpiUtPopGenericState (&StateList);
369             AcpiUtDeleteGenericState (State);
370         }
371         return_ACPI_STATUS (AE_NO_MEMORY);
372     }
373 }
374 }

376 /* We should never get here */

378 return_ACPI_STATUS (AE_AML_INTERNAL);
379 }

382 #ifdef ACPI_DEBUG_OUTPUT
383 /*****
384  *
385  * FUNCTION:      AcpiUtDisplayInitPathname
386  *
387  * PARAMETERS:    Type                - Object type of the node
388  *                ObjHandle            - Handle whose pathname will be displayed
389  *                Path                 - Additional path string to be appended.

```

```

390 *                                     (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;
406
408     ACPI_FUNCTION_ENTRY ();
409
411     /* Only print the path if the appropriate debug level is enabled */
412
413     if (!(AcpiDbgLevel & ACPI_LV_INIT_NAMES))
414     {
415         return;
416     }
417
418     /* Get the full pathname to the node */
419
420     Buffer.Length = ACPI_ALLOCATE_LOCAL_BUFFER;
421     Status = AcpiNsHandleToPathname (ObjHandle, &Buffer);
422     if (ACPI_FAILURE (Status))
423     {
424         return;
425     }
426
427     /* Print what we're doing */
428
429     switch (Type)
430     {
431     case ACPI_TYPE_METHOD:
432
433         AcpiOsPrintf ("Executing      ");
434         break;
435
436     default:
437
438         AcpiOsPrintf ("Initializing ");
439         break;
440     }
441
442     /* Print the object type and pathname */
443
444     AcpiOsPrintf ("%12s %s",
445         AcpiUtGetTypeName (Type), (char *) Buffer.Pointer);
446
447     /* Extra path is used to append names like _STA, _INI, etc. */
448
449     if (Path)
450     {
451         AcpiOsPrintf (".%s", Path);
452     }
453     AcpiOsPrintf ("\n");
454
455     ACPI_FREE (Buffer.Pointer);

```

```

456 }
457 #endif

```

new/usr/src/common/acpica/components/utilities/utmutex.c

1

```
*****
11172 Fri Jan 17 07:30:48 2014
new/usr/src/common/acpica/components/utilities/utmutex.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utmutex - local mutex support
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTMUTEX_C__

47 #include "acpi.h"
48 #include "accommon.h"

50 #define _COMPONENT          ACPI_UTILITIES
51 #define ACPI_MODULE_NAME    ("utmutex")

53 /* Local prototypes */

55 static ACPI_STATUS
56 AcpiUtCreateMutex (
57     ACPI_MUTEX_HANDLE       MutexId);
```

new/usr/src/common/acpica/components/utilities/utmutex.c

2

```
59 static void
60 AcpiUtDeleteMutex (
61     ACPI_MUTEX_HANDLE       MutexId);

64 /*****
65 *
66 * FUNCTION:      AcpiUtMutexInitialize
67 *
68 * PARAMETERS:    None.
69 *
70 * RETURN:        Status
71 *
72 * DESCRIPTION:    Create the system mutex objects. This includes mutexes,
73 *                 spin locks, and reader/writer locks.
74 *
75 *****/

77 ACPI_STATUS
78 AcpiUtMutexInitialize (
79     void)
80 {
81     UINT32          i;
82     ACPI_STATUS      Status;

85     ACPI_FUNCTION_TRACE (UtMutexInitialize);

88     /* Create each of the predefined mutex objects */

90     for (i = 0; i < ACPI_NUM_MUTEX; i++)
91     {
92         Status = AcpiUtCreateMutex (i);
93         if (ACPI_FAILURE (Status))
94         {
95             return_ACPI_STATUS (Status);
96         }
97     }

99     /* Create the spinlocks for use at interrupt level or for speed */
100     /* Create the spinlocks for use at interrupt level */

101     Status = AcpiOsCreateLock (&AcpiGbl_GpeLock);
102     if (ACPI_FAILURE (Status))
103     {
104         return_ACPI_STATUS (Status);
105     }

107     Status = AcpiOsCreateLock (&AcpiGbl_HardwareLock);
108     if (ACPI_FAILURE (Status))
109     {
110         return_ACPI_STATUS (Status);
111     }

113     Status = AcpiOsCreateLock (&AcpiGbl_ReferenceCountLock);
114     if (ACPI_FAILURE (Status))
115     {
116         return_ACPI_STATUS (Status);
117     }

119     /* Mutex for _OSI support */

121     Status = AcpiOsCreateMutex (&AcpiGbl_OsiMutex);
122     if (ACPI_FAILURE (Status))
123     {
```

```

124     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 {
348     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 }
```

new/usr/src/common/acpica/components/utilities/utobject.c

1

```
*****
21968 Fri Jan 17 07:30:49 2014
new/usr/src/common/acpica/components/utilities/utobject.c
update to acpica-unix2-20140114
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 - 2014, 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 * 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 MERCHANTABILITY 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 #define ACPI_MODULE_NAME    ("utobject")

54 /* Local prototypes */

56 static ACPI_STATUS
57 AcpiUtGetSimpleObjectSize (
58     ACPI_OPERAND_OBJECT *Obj,
```

new/usr/src/common/acpica/components/utilities/utobject.c

2

```
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-satisfies-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     }

134     SecondObject->Common.Type = ACPI_TYPE_LOCAL_EXTRA;
135     SecondObject->Common.ReferenceCount = 1;

137     /* Link the second object to the first */

139     Object->Common.NextObject = SecondObject;
140     break;

142     default:

144         /* All others have no secondary object */
145         break;
146     }

148     /* Save the object type in the object descriptor */

150     Object->Common.Type = (UINT8) Type;

152     /* Init the reference count */

154     Object->Common.ReferenceCount = 1;

156     /* Any per-type initialization should go here */

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  * DESCRIPTION: Validate a pointer to be an ACPI_OPERAND_OBJECT
372  *****/

374 BOOLEAN
375 AcpiUtValidInternalObject (
376     void                *Object)
377 {
379     ACPI_FUNCTION_NAME (UtValidInternalObject);

382     /* Check for a null pointer */

384     if (!Object)
385     {
386         ACPI_DEBUG_PRINT ((ACPI_DB_EXEC, "**** Null Object Ptr\n"));
387         return (FALSE);
388     }

```

```

390     /* Check the descriptor type field */

392     switch (ACPI_GET_DESCRIPTOR_TYPE (Object))
393     {
394     case ACPI_DESC_TYPE_OPERAND:

396         /* The object appears to be a valid ACPI_OPERAND_OBJECT */

398         return (TRUE);

400     default:

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     }

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  *****/

471 void
472 AcpiUtDeleteObjectDesc (
473     ACPI_OPERAND_OBJECT *Object)
474 {
475     ACPI_FUNCTION_TRACE_PTR (UtDeleteObjectDesc, Object);

478     /* Object must be of type ACPI_OPERAND_OBJECT */
479     /* Object must be an ACPI_OPERAND_OBJECT */

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     }

488     (void) AcpiOsReleaseObject (AcpiGbl_OperandCache, Object);
489     return_VOID;
490 }

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 *****/
509
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;
518
519     ACPI_FUNCTION_TRACE_PTR (UtGetSimpleObjectSize, InternalObject);
520
521     /* Start with the length of the (external) Acpi object */
522     Length = sizeof (ACPI_OBJECT);
523
524     /* A NULL object is allowed, can be a legal uninitialized package element */
525     if (!InternalObject)
526     {
527         /*
528          * Object is NULL, just return the length of ACPI_OBJECT
529          * (A NULL ACPI_OBJECT is an object of all zeroes.)
530          */
531         *ObjLength = ACPI_ROUND_UP_TO_NATIVE_WORD (Length);
532         return_ACPI_STATUS (AE_OK);
533     }
534
535     /* A Namespace Node should never appear here */
536     if (ACPI_GET_DESCRIPTOR_TYPE (InternalObject) == ACPI_DESC_TYPE_NAMED)
537     {
538         /* A namespace node should never get here */
539         return_ACPI_STATUS (AE_AML_INTERNAL);
540     }
541
542     /*
543      * The final length depends on the object type
544      * Strings and Buffers are packed right up against the parent object and
545      * must be accessed byte-wise or there may be alignment problems on
546      * certain processors
547      */
548     switch (InternalObject->Common.Type)
549     {
550     case ACPI_TYPE_STRING:
551
552         Length += (ACPI_SIZE) InternalObject->String.Length + 1;
553         break;
554
555     case ACPI_TYPE_BUFFER:
556
557         Length += (ACPI_SIZE) InternalObject->Buffer.Length;
558         break;

```

```

559     case ACPI_TYPE_INTEGER:
560     case ACPI_TYPE_PROCESSOR:
561     case ACPI_TYPE_POWER:
562
563         /* No extra data for these types */
564         break;
565
566     case ACPI_TYPE_LOCAL_REFERENCE:
567
568         switch (InternalObject->Reference.Class)
569         {
570         case ACPI_REFCLASS_NAME:
571
572             /*
573              * Get the actual length of the full pathname to this object.
574              * The reference will be converted to the pathname to the object
575              */
576             Size = AcpiNsGetPathnameLength (InternalObject->Reference.Node);
577             if (!Size)
578             {
579                 return_ACPI_STATUS (AE_BAD_PARAMETER);
580             }
581
582             Length += ACPI_ROUND_UP_TO_NATIVE_WORD (Size);
583             break;
584
585         default:
586
587             /*
588              * No other reference opcodes are supported.
589              * Notably, Locals and Args are not supported, but this may be
590              * required eventually.
591              */
592             ACPI_ERROR ((AE_INFO, "Cannot convert to external object - "
593                 "unsupported Reference Class [%s] 0x%X in object %p",
594                 AcpiUtGetReferenceName (InternalObject),
595                 InternalObject->Reference.Class, InternalObject));
596             Status = AE_TYPE;
597             break;
598         }
599
600     default:
601
602         ACPI_ERROR ((AE_INFO, "Cannot convert to external object - "
603             "unsupported type [%s] 0x%X in object %p",
604             AcpiUtGetObjectTypeName (InternalObject),
605             InternalObject->Common.Type, InternalObject));
606         Status = AE_TYPE;
607         break;
608     }
609
610     *ObjLength = ACPI_ROUND_UP_TO_NATIVE_WORD (Length);
611     return_ACPI_STATUS (Status);
612 }

```



```
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;
650
651     switch (ObjectType)
652     {
653     case ACPI_COPY_TYPE_SIMPLE:
654
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         }
664
665         Info->Length += ObjectSpace;
666         break;
667
668     case ACPI_COPY_TYPE_PACKAGE:
669
670         /* Package object - nothing much to do here, let the walk handle it */
671
672         Info->NumPackages++;
673         State->Pkg.ThisTargetObj = NULL;
674         break;
675
676     default:
677
678         /* No other types allowed */
679
680         return (AE_BAD_PARAMETER);
681     }
682
683     return (Status);
684 }
```

unchanged portion omitted

new/usr/src/common/acpica/components/utilities/utosi.c

1

```
*****
15789 Fri Jan 17 07:30:49 2014
new/usr/src/common/acpica/components/utilities/utosi.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utosi - Support for the _OSI predefined control method
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTOSI_C__

46 #include "acpi.h"
47 #include "accommon.h"

50 #define _COMPONENT          ACPI_UTILITIES
51 #define ACPI_MODULE_NAME    ("utosi")

53 /*
54 * Strings supported by the _OSI predefined control method (which is
55 * implemented internally within this module.)
56 *
57 * March 2009: Removed "Linux" as this host no longer wants to respond true
58 * for this string. Basically, the only safe OS strings are windows-related
```

new/usr/src/common/acpica/components/utilities/utosi.c

2

```
59 * and in many or most cases represent the only test path within the
60 * BIOS-provided ASL code.
61 *
62 * The last element of each entry is used to track the newest version of
63 * Windows that the BIOS has requested.
64 */
65 static ACPI_INTERFACE_INFO    AcpiDefaultSupportedInterfaces[] =
66 {
67     /* Operating System Vendor Strings */

69     {"Windows 2000",          NULL, 0, ACPI_OSI_WIN_2000},          /* Windows 2000
70     {"Windows 2001",          NULL, 0, ACPI_OSI_WIN_XP},            /* Windows XP
71     {"Windows 2001 SP1",      NULL, 0, ACPI_OSI_WIN_XP_SP1},        /* Windows XP S
72     {"Windows 2001.1",        NULL, 0, ACPI_OSI_WINSRV_2003},        /* Windows Serv
73     {"Windows 2001 SP2",      NULL, 0, ACPI_OSI_WIN_XP_SP2},        /* Windows XP S
74     {"Windows 2001.1 SP1",     NULL, 0, ACPI_OSI_WINSRV_2003_SP1},    /* Windows Serv
75     {"Windows 2006",          NULL, 0, ACPI_OSI_WIN_VISTA},          /* Windows Vist
76     {"Windows 2006.1",        NULL, 0, ACPI_OSI_WINSRV_2008},        /* Windows Serv
77     {"Windows 2006 SP1",      NULL, 0, ACPI_OSI_WIN_VISTA_SP1},      /* Windows Vist
78     {"Windows 2006 SP2",      NULL, 0, ACPI_OSI_WIN_VISTA_SP2},      /* Windows Vist
79     {"Windows 2009",          NULL, 0, ACPI_OSI_WIN_7},              /* Windows 7 an
80     {"Windows 2012",          NULL, 0, ACPI_OSI_WIN_8},              /* Windows 8 an

82     /* Feature Group Strings */

84     {"Extended Address Space Descriptor", NULL, ACPI_OSI_FEATURE, 0},
83     {"Extended Address Space Descriptor", NULL, 0, 0}

86     /*
87     * All "optional" feature group strings (features that are implemented
88     * by the host) should be dynamically modified to VALID by the host via
89     * AcpiInstallInterface or AcpiUpdateInterfaces. Such optional feature
90     * group strings are set as INVALID by default here.
87     * by the host) should be dynamically added by the host via
88     * AcpiInstallInterface and should not be manually added here.
89     *
90     * Examples of optional feature group strings:
91     *
92     * "Module Device"
93     * "Processor Device"
94     * "3.0 Thermal Model"
95     * "3.0 _SCP Extensions"
96     * "Processor Aggregator Device"
97     */

93     {"Module Device",          NULL, ACPI_OSI_OPTIONAL_FEATURE, 0},
94     {"Processor Device",       NULL, ACPI_OSI_OPTIONAL_FEATURE, 0},
95     {"3.0 Thermal Model",      NULL, ACPI_OSI_OPTIONAL_FEATURE, 0},
96     {"3.0 _SCP Extensions",    NULL, ACPI_OSI_OPTIONAL_FEATURE, 0},
97     {"Processor Aggregator Device", NULL, ACPI_OSI_OPTIONAL_FEATURE, 0}
98 };

101 /*****
102 *
103 * FUNCTION:      AcpiUtInitializeInterfaces
104 *
105 * PARAMETERS:    None
106 *
107 * RETURN:        Status
108 *
109 * DESCRIPTION:   Initialize the global _OSI supported interfaces list
110 *
111 *****/

113 ACPI_STATUS
```

```

114 AcpiUtInitializeInterfaces (
115     void)
116 {
117     ACPI_STATUS      Status;
118     UINT32           i;

121     Status = AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
122     if (ACPI_FAILURE (Status))
123     {
124         return (Status);
125     }

120     (void) AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
127     AcpiGbl_SupportedInterfaces = AcpiDefaultSupportedInterfaces;

129     /* Link the static list of supported interfaces */

131     for (i = 0; i < (ACPI_ARRAY_LENGTH (AcpiDefaultSupportedInterfaces) - 1); i++)
132     {
133         AcpiDefaultSupportedInterfaces[i].Next =
134             &AcpiDefaultSupportedInterfaces[(ACPI_SIZE) i + 1];
135     }

137     AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
138     return (AE_OK);
139 }

142 /*****
143  *
144  * FUNCTION:      AcpiUtInterfaceTerminate
145  *
146  * PARAMETERS:    None
147  *
148  * RETURN:        Status
149  * RETURN:        None
150  * DESCRIPTION:   Delete all interfaces in the global list. Sets
151  *                AcpiGbl_SupportedInterfaces to NULL.
152  *
153  *****/

155 ACPI_STATUS
156 void
157 AcpiUtInterfaceTerminate (
158     void)
159 {
160     ACPI_STATUS      Status;
161     ACPI_INTERFACE_INFO *NextInterface;

163     Status = AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
164     if (ACPI_FAILURE (Status))
165     {
166         return (Status);
167     }

156     (void) AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
169     NextInterface = AcpiGbl_SupportedInterfaces;

170     while (NextInterface)
171     {
172         AcpiGbl_SupportedInterfaces = NextInterface->Next;

174         if (NextInterface->Flags & ACPI_OSI_DYNAMIC)

```

```

175     {
176         /* Only interfaces added at runtime can be freed */

165     if (NextInterface->Flags & ACPI_OSI_DYNAMIC)
166     {
178         ACPI_FREE (NextInterface->Name);
179         ACPI_FREE (NextInterface);
180     }
181     else
182     {
183         /* Interface is in static list. Reset it to invalid or valid. */

185         if (NextInterface->Flags & ACPI_OSI_DEFAULT_INVALID)
186         {
187             NextInterface->Flags |= ACPI_OSI_INVALID;
188         }
189         else
190         {
191             NextInterface->Flags &= ~ACPI_OSI_INVALID;
192         }
193     }

195     NextInterface = AcpiGbl_SupportedInterfaces;
196 }

198     AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
199     return (AE_OK);
200 }

    unchanged_portion_omitted_

320 /*****
321  *
322  * FUNCTION:      AcpiUtUpdateInterfaces
323  *
324  * PARAMETERS:    Action                - Actions to be performed during the
325  *                                           update
326  *
327  * RETURN:        Status
328  *
329  * DESCRIPTION:   Update _OSI interface strings, disabling or enabling OS vendor
330  *                strings or/and feature group strings.
331  *                Caller MUST hold AcpiGbl_OsiMutex
332  *
333  *****/

335 ACPI_STATUS
336 AcpiUtUpdateInterfaces (
337     UINT8           Action)
338 {
339     ACPI_INTERFACE_INFO *NextInterface;

342     NextInterface = AcpiGbl_SupportedInterfaces;
343     while (NextInterface)
344     {
345         if (((NextInterface->Flags & ACPI_OSI_FEATURE) &&
346             (Action & ACPI_FEATURE_STRINGS)) ||
347             (!(NextInterface->Flags & ACPI_OSI_FEATURE) &&
348             (Action & ACPI_VENDOR_STRINGS)))
349         {
350             if (Action & ACPI_DISABLE_INTERFACES)
351             {
352                 /* Mark the interfaces as invalid */

354                 NextInterface->Flags |= ACPI_OSI_INVALID;

```

```

355     }
356     else
357     {
358         /* Mark the interfaces as valid */
359
360         NextInterface->Flags &= ~ACPI_OSI_INVALID;
361     }
362 }
363
364     NextInterface = NextInterface->Next;
365 }
366
367     return (AE_OK);
368 }
369
370 /*****
371 * FUNCTION:      AcpiUtGetInterface
372 *
373 * PARAMETERS:    InterfaceName      - The interface to find
374 *
375 * RETURN:        ACPI_INTERFACE_INFO if found. NULL if not found.
376 *
377 * DESCRIPTION:   Search for the specified interface name in the global list.
378 *                Caller MUST hold AcpiGbl_OsiMutex
379 * *****/
380
381 ACPI_INTERFACE_INFO *
382 AcpiUtGetInterface (
383     ACPI_STRING      InterfaceName)
384 {
385     ACPI_INTERFACE_INFO *NextInterface;
386
387     NextInterface = AcpiGbl_SupportedInterfaces;
388     while (NextInterface)
389     {
390         if (!ACPI_STRCMP (InterfaceName, NextInterface->Name))
391         {
392             return (NextInterface);
393         }
394         NextInterface = NextInterface->Next;
395     }
396
397     return (NULL);
398 }
399
400 /*****
401 * FUNCTION:      AcpiUtOsiImplementation
402 *
403 * PARAMETERS:    WalkState          - Current walk state
404 *
405 * RETURN:        Status
406 *
407 * DESCRIPTION:   Implementation of the _OSI predefined control method. When
408 *                an invocation of _OSI is encountered in the system AML,
409 *                control is transferred to this function.
410 * *****/
411
412 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;
430
431     ACPI_FUNCTION_TRACE (UtOsiImplementation);
432
433     /* Validate the string input argument (from the AML caller) */
434
435     StringDesc = WalkState->Arguments[0].Object;
436     if (!StringDesc ||
437         (StringDesc->Common.Type != ACPI_TYPE_STRING))
438     {
439         return_ACPI_STATUS (AE_TYPE);
440     }
441
442     /* Create a return object */
443
444     ReturnDesc = AcpiUtCreateInternalObject (ACPI_TYPE_INTEGER);
445     if (!ReturnDesc)
446     {
447         return_ACPI_STATUS (AE_NO_MEMORY);
448     }
449
450     /* Default return value is 0, NOT SUPPORTED */
451
452     ReturnValue = 0;
453     Status = AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
454     if (ACPI_FAILURE (Status))
455     {
456         AcpiUtRemoveReference (ReturnDesc);
457         return_ACPI_STATUS (Status);
458     }
459     (void) AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
460
461     /* Lookup the interface in the global _OSI list */
462
463     InterfaceInfo = AcpiUtGetInterface (StringDesc->String.Pointer);
464     if (InterfaceInfo &&
465         !(InterfaceInfo->Flags & ACPI_OSI_INVALID))
466     {
467         /*
468          * The interface is supported.
469          * Update the OsiData if necessary. We keep track of the latest
470          * version of Windows that has been requested by the BIOS.
471          */
472         if (InterfaceInfo->Value > AcpiGbl_OsiData)
473         {
474             AcpiGbl_OsiData = InterfaceInfo->Value;
475         }
476
477         ReturnValue = ACPI_UINT32_MAX;
478     }
479
480     AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
481
482     /*
483     * Invoke an optional _OSI interface handler. The host OS may wish
484     * 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     }
494
495     ACPI_DEBUG_PRINT_RAW ((ACPI_DB_INFO,
496         "ACPI: BIOS _OSI(\\\"%s\\\") is %ssupported\n",
497         StringDesc->String.Pointer, ReturnValue == 0 ? "not " : ""));
498
499     /* Complete the return object */
500
501     ReturnDesc->Integer.Value = ReturnValue;
502     WalkState->ReturnDesc = ReturnDesc;
503     return_ACPI_STATUS (AE_OK);
504 }
505
506 unchanged_portion_omitted
```

new/usr/src/common/acpica/components/utilities/utownerid.c

1

```
*****
7652 Fri Jan 17 07:30:50 2014
new/usr/src/common/acpica/components/utilities/utownerid.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: utownerid - Support for Table/Method Owner IDs
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTOWNERID_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acnamesp.h"

52 #define _COMPONENT          ACPI_UTILITIES
53 #define ACPI_MODULE_NAME    ("utownerid")

56 /*****
57 *
58 * FUNCTION:      AcpiUtAllocateOwnerId
59 *
60 * PARAMETERS:    OwnerId          - Where the new owner ID is returned
```

new/usr/src/common/acpica/components/utilities/utownerid.c

2

```
61 *
62 * RETURN:        Status
63 *
64 * DESCRIPTION:    Allocate a table or method owner ID. The owner ID is used to
65 *                 track objects created by the table or method, to be deleted
66 *                 when the method exits or the table is unloaded.
67 *
68 *****/

70 ACPI_STATUS
71 AcpiUtAllocateOwnerId (
72     ACPI_OWNER_ID      *OwnerId)
73 {
74     UINT32              i;
75     UINT32              j;
76     UINT32              k;
77     ACPI_STATUS          Status;

80     ACPI_FUNCTION_TRACE (UtAllocateOwnerId);

83     /* Guard against multiple allocations of ID to the same location */

85     if (*OwnerId)
86     {
87         ACPI_ERROR ((AE_INFO, "Owner ID [0x%2.2X] already exists", *OwnerId));
88         return_ACPI_STATUS (AE_ALREADY_EXISTS);
89     }

91     /* Mutex for the global ID mask */

93     Status = AcpiUtAcquireMutex (ACPI_MTX_CACHES);
94     if (ACPI_FAILURE (Status))
95     {
96         return_ACPI_STATUS (Status);
97     }

99     /*
100      * Find a free owner ID, cycle through all possible IDs on repeated
101      * allocations. (ACPI_NUM_OWNERID_MASKS + 1) because first index may have
102      * to be scanned twice.
103      */
104     for (i = 0, j = AcpiGbl_LastOwnerIdIndex;
105          i < (ACPI_NUM_OWNERID_MASKS + 1);
106          i++, j++)
107     {
108         if (j >= ACPI_NUM_OWNERID_MASKS)
109         {
110             j = 0; /* Wraparound to start of mask array */
111         }

113         for (k = AcpiGbl_NextOwnerIdOffset; k < 32; k++)
114         {
115             if (AcpiGbl_OwnerIdMask[j] == ACPI_UINT32_MAX)
116             {
117                 /* There are no free IDs in this mask */

119                 break;
120             }

122             if (!(AcpiGbl_OwnerIdMask[j] & (1 << k)))
123             {
124                 /*
125                  * Found a free ID. The actual ID is the bit index plus one,
126                  * making zero an invalid Owner ID. Save this as the last ID
```

```

127     * allocated and update the global ID mask.
128     */
129     AcpiGbl_OwnerIdMask[j] |= (1 << k);

131     AcpiGbl_LastOwnerIdIndex = (UINT8) j;
132     AcpiGbl_NextOwnerIdOffset = (UINT8) (k + 1);

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));

142     ACPI_DEBUG_PRINT ((ACPI_DB_VALUES,
143     "Allocated OwnerId: %2.2X\n", (unsigned int) *OwnerId));
144     goto Exit;
145 }
146 }

148     AcpiGbl_NextOwnerIdOffset = 0;
149 }

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));

165 Exit:
166     (void) AcpiUtReleaseMutex (ACPI_MTX_CACHES);
167     return ACPI_STATUS (Status);
168 }

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 *****/

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;

```

```

195     ACPI_FUNCTION_TRACE_U32 (UtReleaseOwnerId, OwnerId);

198     /* Always clear the input OwnerId (zero is an invalid ID) */

200     *OwnerIdPtr = 0;

202     /* Zero is not a valid OwnerID */

204     if (OwnerId == 0)
205     {
206         ACPI_ERROR ((AE_INFO, "Invalid OwnerId: 0x%2.2X", OwnerId));
207         return_VOID;
208     }

210     /* Mutex for the global ID mask */

212     Status = AcpiUtAcquireMutex (ACPI_MTX_CACHES);
213     if (ACPI_FAILURE (Status))
214     {
215         return_VOID;
216     }

218     /* Normalize the ID to zero */

220     OwnerId--;

222     /* Decode ID to index/offset pair */

224     Index = ACPI_DIV_32 (OwnerId);
225     Bit = 1 << ACPI_MOD_32 (OwnerId);

227     /* Free the owner ID only if it is valid */

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     }

239     (void) AcpiUtReleaseMutex (ACPI_MTX_CACHES);
240     return_VOID;
241 }

```

new/usr/src/common/acpica/components/utilities/utpredef.c

1

```
*****
12498 Fri Jan 17 07:30:50 2014
new/usr/src/common/acpica/components/utilities/utpredef.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: utpredef - support functions for predefined names
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTPREDEF_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acpredef.h"

51 #define _COMPONENT          ACPI_UTILITIES
52 #define ACPI_MODULE_NAME    ("utpredef")

55 /*
56 * Names for the types that can be returned by the predefined objects.
57 * Used for warning messages. Must be in the same order as the ACPI_RTYPEs
58 */
59 static const char    *UtRtypeNames[] =
60 {
```

new/usr/src/common/acpica/components/utilities/utpredef.c

2

```
61     "/Integer",
62     "/String",
63     "/Buffer",
64     "/Package",
65     "/Reference",
66 };

69 /*****
70 *
71 * FUNCTION:      AcpiUtGetNextPredefinedMethod
72 *
73 * PARAMETERS:    ThisName          - Entry in the predefined method/name table
74 *
75 * RETURN:        Pointer to next entry in predefined table.
76 *
77 * DESCRIPTION:   Get the next entry in the predefined method table. Handles the
78 * cases where a package info entry follows a method name that
79 * returns a package.
80 */
81 *****/

83 const ACPI_PREDEFINED_INFO *
84 AcpiUtGetNextPredefinedMethod (
85     const ACPI_PREDEFINED_INFO *ThisName)
86 {
88     /*
89     * Skip next entry in the table if this name returns a Package
90     * (next entry contains the package info)
91     */
92     if ((ThisName->Info.ExpectedBtypes & ACPI_RTYPE_PACKAGE) &&
93         (ThisName->Info.ExpectedBtypes != ACPI_RTYPE_ALL))
94     {
95         ThisName++;
96     }

98     ThisName++;
99     return (ThisName);
100 }

103 /*****
104 *
105 * FUNCTION:      AcpiUtMatchPredefinedMethod
106 *
107 * PARAMETERS:    Name              - Name to find
108 *
109 * RETURN:        Pointer to entry in predefined table. NULL indicates not found.
110 *
111 * DESCRIPTION:   Check an object name against the predefined object list.
112 */
113 *****/

115 const ACPI_PREDEFINED_INFO *
116 AcpiUtMatchPredefinedMethod (
117     char          *Name)
118 {
119     const ACPI_PREDEFINED_INFO *ThisName;

122     /* Quick check for a predefined name, first character must be underscore */

124     if (Name[0] != '_')
125     {
126         return (NULL);
```



```

127 }
129 /* Search info table for a predefined method/object name */
131 ThisName = AcpiGbl_PredefinedMethods;
132 while (ThisName->Info.Name[0])
133 {
134     if (ACPI_COMPARE_NAME (Name, ThisName->Info.Name))
135     {
136         return (ThisName);
137     }
139     ThisName = AcpiUtGetNextPredefinedMethod (ThisName);
140 }
142 return (NULL); /* Not found */
143 }

146 /*****
147 * FUNCTION:      AcpiUtGetExpectedReturnTypes
148 *
149 * PARAMETERS:    Buffer          - Where the formatted string is returned
150 *                ExpectedBTypes - Bitfield of expected data types
151 *
152 * RETURN:        Formatted string in Buffer.
153 *
154 * DESCRIPTION:   Format the expected object types into a printable string.
155 *
156 * *****/
157

159 void
160 AcpiUtGetExpectedReturnTypes (
161     char          *Buffer,
162     UINT32        ExpectedBTypes)
163 {
164     UINT32        ThisRtype;
165     UINT32        i;
166     UINT32        j;

169     if (!ExpectedBTypes)
170     {
171         ACPI_STRCPY (Buffer, "NONE");
172         return;
173     }

175     j = 1;
176     Buffer[0] = 0;
177     ThisRtype = ACPI_RTYPE_INTEGER;

179     for (i = 0; i < ACPI_NUM_RTYPES; i++)
180     {
181         /* If one of the expected types, concatenate the name of this type */

183         if (ExpectedBTypes & ThisRtype)
184         {
185             ACPI_STRCAT (Buffer, &UtRtypeName[i][j]);
186             j = 0; /* Use name separator from now on */
187         }

189         ThisRtype <<= 1; /* Next Rtype */
190     }
191 }

```

```

194 /*****
195 *
196 * The remaining functions are used by iASL and AcpiHelp only
197 *
198 *****/

200 #if (defined ACPI_AS_L_COMPILER || defined ACPI_HELP_APP)
201 #include <stdio.h>
202 #include <string.h>

204 /* Local prototypes */

206 static UINT32
207 AcpiUtGetArgumentTypes (
208     char          *Buffer,
209     UINT16        ArgumentTypes);

212 /* Types that can be returned externally by a predefined name */

214 static const char *UtExternalTypeNames[] = /* Indexed by ACPI_TYPE_ */
215 {
216     "UNSUPPORTED-TYPE",
217     "Integer",
218     "String",
219     "Buffer",
220     "Package"
221 };

223 /* Bit widths for resource descriptor predefined names */

225 static const char *UtResourceTypeNames[] =
226 {
227     "/1",
228     "/2",
229     "/3",
230     "/8",
231     "/16",
232     "/32",
233     "/64",
234     "/variable",
235 };

238 /*****
239 *
240 * FUNCTION:      AcpiUtMatchResourceName
241 *
242 * PARAMETERS:    Name          - Name to find
243 *
244 * RETURN:        Pointer to entry in the resource table. NULL indicates not
245 *                found.
246 *
247 * DESCRIPTION:   Check an object name against the predefined resource
248 *                descriptor object list.
249 *
250 *****/

252 const ACPI_PREDEFINED_INFO *
253 AcpiUtMatchResourceName (
254     char          *Name)
255 {
256     const ACPI_PREDEFINED_INFO *ThisName;

```

```

259  /* 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  * FUNCTION:      AcpiUtDisplayPredefinedMethod
285  *
286  * PARAMETERS:    Buffer          - Scratch buffer for this function
287  *                ThisName       - Entry in the predefined method/name table
288  *                MultiLine      - TRUE if output should be on >1 line
289  *
290  * RETURN:        None
291  *
292  * DESCRIPTION:   Display information about a predefined method. Number and
293  *                type of the input arguments, and expected type(s) for the
294  *                return value, if any.
295  *
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);
395
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         }
402
403         strcat (Buffer, UtExternalTypeNames[ThisArgumentType] + SubIndex);
404         SubIndex = 0;
405     }
406
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;
432
433     *Buffer = 0;
434     SubIndex = 1;
435     Found = 0;
436
437     for (i = 0; i < NUM_RESOURCE_WIDTHS; i++)
438     {
439         if (Types & 1)
440         {
441             strcat (Buffer, &(UtResourceTypeNames[i][SubIndex]));
442             SubIndex = 0;
443             Found++;
444         }
445         Types >>= 1;
446     }
447
448     return (Found);
449 }
450
451 #endif

```

new/usr/src/common/acpica/components/utilities/utresrc.c

1

```
*****
24458 Fri Jan 17 07:30:50 2014
new/usr/src/common/acpica/components/utilities/utresrc.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utresrc - Resource management utilities
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTRESRC_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acresrc.h"

52 #define _COMPONENT          ACPI_UTILITIES
53       ACPI_MODULE_NAME    ("utresrc")

56 #if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DISASSEMBLER) || defined(ACPI_D

58 /*
59 * Strings used to decode resource descriptors.
```

new/usr/src/common/acpica/components/utilities/utresrc.c

2

```
60 * Used by both the disassembler and the debugger resource dump routines
61 */
62 const char          *AcpiGbl_BmDecode[] =
63 {
64     "NotBusMaster",
65     "BusMaster"
66 };

68 const char          *AcpiGbl_ConfigDecode[] =
69 {
70     "0 - Good Configuration",
71     "1 - Acceptable Configuration",
72     "2 - Suboptimal Configuration",
73     "3 - ***Invalid Configuration***",
74 };

76 const char          *AcpiGbl_ConsumeDecode[] =
77 {
78     "ResourceProducer",
79     "ResourceConsumer"
80 };

82 const char          *AcpiGbl_DecDecode[] =
83 {
84     "PosDecode",
85     "SubDecode"
86 };

88 const char          *AcpiGbl_HeDecode[] =
89 {
90     "Level",
91     "Edge"
92 };

94 const char          *AcpiGbl_IoDecode[] =
95 {
96     "Decode10",
97     "Decode16"
98 };

100 const char          *AcpiGbl_LlDecode[] =
101 {
102     "ActiveHigh",
103     "ActiveLow"
104 };

106 const char          *AcpiGbl_MaxDecode[] =
107 {
108     "MaxNotFixed",
109     "MaxFixed"
110 };

112 const char          *AcpiGbl_MemDecode[] =
113 {
114     "NonCacheable",
115     "Cacheable",
116     "WriteCombining",
117     "Prefetchable"
118 };

120 const char          *AcpiGbl_MinDecode[] =
121 {
122     "MinNotFixed",
123     "MinFixed"
124 };
```

```

126 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_TrnsDecode[] =
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
391 * lengths must meet the minimum required length, etc.
392 * Zero indicates a reserved (and therefore invalid) resource type.
393 */
394 static const UINT8      AcpiGbl_ResourceTypes[] =
395 {
396     /* Small descriptors */
397
398     0,
399     0,
400     0,
401     0,
402     ACPI_SMALL_VARIABLE_LENGTH, /* 04 IRQ */
403     ACPI_FIXED_LENGTH,          /* 05 DMA */
404     ACPI_SMALL_VARIABLE_LENGTH, /* 06 StartDependentFunctions */
405     ACPI_FIXED_LENGTH,          /* 07 EndDependentFunctions */
406     ACPI_FIXED_LENGTH,          /* 08 IO */
407     ACPI_FIXED_LENGTH,          /* 09 FixedIO */
408     ACPI_FIXED_LENGTH,          /* 0A FixedDMA */
409     0,
410     0,
411     0,
412     ACPI_VARIABLE_LENGTH,       /* 0E VendorShort */
413     ACPI_FIXED_LENGTH,          /* 0F EndTag */
414
415     /* Large descriptors */
416
417     0,
418     ACPI_FIXED_LENGTH,          /* 01 Memory24 */
419     ACPI_FIXED_LENGTH,          /* 02 GenericRegister */
420     0,
421     ACPI_VARIABLE_LENGTH,       /* 04 VendorLong */
422     ACPI_FIXED_LENGTH,          /* 05 Memory32 */
423     ACPI_FIXED_LENGTH,          /* 06 Memory32Fixed */
424     ACPI_VARIABLE_LENGTH,       /* 07 Dword* address */
425     ACPI_VARIABLE_LENGTH,       /* 08 Word* address */
426     ACPI_VARIABLE_LENGTH,       /* 09 ExtendedIRQ */
427     ACPI_VARIABLE_LENGTH,       /* 0A Qword* address */
428     ACPI_FIXED_LENGTH,          /* 0B Extended* address */
429     ACPI_VARIABLE_LENGTH,       /* 0C Gpio* */
430     0,
431     ACPI_VARIABLE_LENGTH        /* 0E *SerialBus */
432 };
433
434 /*****
435 *
436 * FUNCTION:      AcpiUtWalkAmlResources
437 *
438 * PARAMETERS:    WalkState      - Current walk info
439 *                Aml            - Pointer to the raw AML resource template
440 *                AmlLength      - Length of the entire template
441 *                UserFunction    - Called once for each descriptor found. If
442 *                                NULL, a pointer to the EndTag is returned
443 *                Context        - Passed to UserFunction
444 *
445 * RETURN:        Status
446 *
447 * DESCRIPTION:   Walk a raw AML resource list(buffer). User function called
448 *                once for each resource found.
449 *
450 * *****/
451
452 ACPI_STATUS
453 AcpiUtWalkAmlResources (
454     ACPI_WALK_STATE *WalkState,

```

```

456     UINT8          *Aml,
457     ACPI_SIZE      AmlLength,
458     ACPI_WALK_AML_CALLBACK UserFunction,
459     void            **Context)
460 {
461     ACPI_STATUS      Status;
462     UINT8            *EndAml;
463     UINT8            ResourceIndex;
464     UINT32           Length;
465     UINT32           Offset = 0;
466     UINT8            EndTag[2] = {0x79, 0x00};
467
468     ACPI_FUNCTION_TRACE (UtWalkAmlResources);
469
470     /* The absolute minimum resource template is one EndTag descriptor */
471
472     if (AmlLength < sizeof (AML_RESOURCE_END_TAG))
473     {
474         return ACPI_STATUS (AE_AML_NO_RESOURCE_END_TAG);
475     }
476
477     /* Point to the end of the resource template buffer */
478
479     EndAml = Aml + AmlLength;
480
481     /* Walk the byte list, abort on any invalid descriptor type or length */
482
483     while (Aml < EndAml)
484     {
485         /* Validate the Resource Type and Resource Length */
486
487         Status = AcpiUtValidateResource (WalkState, Aml, &ResourceIndex);
488         if (ACPI_FAILURE (Status))
489         {
490             /* Exit on failure. Cannot continue because the descriptor length
491              * may be bogus also.
492              */
493             return ACPI_STATUS (Status);
494         }
495
496         /* Get the length of this descriptor */
497
498         Length = AcpiUtGetDescriptorLength (Aml);
499
500         /* Invoke the user function */
501
502         if (UserFunction)
503         {
504             Status = UserFunction (Aml, Length, Offset, ResourceIndex, Context);
505             if (ACPI_FAILURE (Status))
506             {
507                 return ACPI_STATUS (Status);
508             }
509         }
510
511         /* An EndTag descriptor terminates this resource template */
512
513         if (AcpiUtGetResourceType (Aml) == ACPI_RESOURCE_NAME_END_TAG)
514         {
515             /* There must be at least one more byte in the buffer for
516              * the 2nd byte of the EndTag
517              */

```

```

522     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
787 *          header or the length field itself.
788 *
789 *****/
791 UINT16
792 AcpiUtGetResourceLength (
793     void                *Aml)
794 {
795     ACPI_RS_LENGTH      ResourceLength;

798     ACPI_FUNCTION_ENTRY ();

801     /*
802     * Byte 0 contains the descriptor name (Resource Type)
803     * Examine the large/small bit in the resource header
804     */
805     if (ACPI_GET8 (Aml) & ACPI_RESOURCE_NAME_LARGE)
806     {
807         /* Large Resource type -- bytes 1-2 contain the 16-bit length */
809         ACPI_MOVE_16_TO_16 (&ResourceLength, ACPI_ADD_PTR (UINT8, Aml, 1));

811     }
812     else
813     {
814         /* Small Resource type -- bits 2:0 of byte 0 contain the length */

816         ResourceLength = (UINT16) (ACPI_GET8 (Aml) &
817                                     ACPI_RESOURCE_NAME_SMALL_LENGTH_MASK);
818     }

820     return (ResourceLength);
821 }

824 /*****/
825 *
826 * FUNCTION:      AcpiUtGetResourceHeaderLength
827 *
828 * PARAMETERS:    Aml                - Pointer to the raw AML resource descriptor
829 *
830 * RETURN:        Length of the AML header (depends on large/small descriptor)
831 *
832 * DESCRIPTION:   Get the length of the header for this resource.
833 *
834 *****/
836 UINT8
837 AcpiUtGetResourceHeaderLength (
838     void                *Aml)
839 {
840     ACPI_FUNCTION_ENTRY ();

843     /* Examine the large/small bit in the resource header */

845     if (ACPI_GET8 (Aml) & ACPI_RESOURCE_NAME_LARGE)
846     {
847         return (sizeof (AML_RESOURCE_LARGE_HEADER));
848     }
849     else
850     {
851         return (sizeof (AML_RESOURCE_SMALL_HEADER));

```

```

852     }
853 }

856 /*****/
857 *
858 * FUNCTION:      AcpiUtGetDescriptorLength
859 *
860 * PARAMETERS:    Aml                - Pointer to the raw AML resource descriptor
861 *
862 * RETURN:        Byte length
863 *
864 * DESCRIPTION:   Get the total byte length of a raw AML descriptor, including the
865 *               length of the descriptor header and the length field itself.
866 *               Used to walk descriptor lists.
867 *
868 *****/
870 UINT32
871 AcpiUtGetDescriptorLength (
872     void                *Aml)
873 {
874     ACPI_FUNCTION_ENTRY ();

877     /*
878     * Get the Resource Length (does not include header length) and add
879     * the header length (depends on if this is a small or large resource)
880     */
881     return (AcpiUtGetResourceLength (Aml) +
882             AcpiUtGetResourceHeaderLength (Aml));
883 }

886 /*****/
887 *
888 * FUNCTION:      AcpiUtGetResourceEndTag
889 *
890 * PARAMETERS:    ObjDesc            - The resource template buffer object
891 *               EndTag              - Where the pointer to the EndTag is returned
892 *
893 * RETURN:        Status, pointer to the end tag
894 *
895 * DESCRIPTION:   Find the EndTag resource descriptor in an AML resource template
896 *               Note: allows a buffer length of zero.
897 *
898 *****/
900 ACPI_STATUS
901 AcpiUtGetResourceEndTag (
902     ACPI_OPERAND_OBJECT *ObjDesc,
903     UINT8                **EndTag)
904 {
905     ACPI_STATUS          Status;

908     ACPI_FUNCTION_TRACE (UtGetResourceEndTag);

911     /* Allow a buffer length of zero */

913     if (!ObjDesc->Buffer.Length)
914     {
915         *EndTag = ObjDesc->Buffer.Pointer;
916         return ACPI_STATUS (AE_OK);
917     }

```

```
919     /* Validate the template and get a pointer to the EndTag */
921     Status = AcpiUtWalkAmlResources (NULL, ObjDesc->Buffer.Pointer,
922                                     ObjDesc->Buffer.Length, NULL, (void **) EndTag);
924     return_ACPI_STATUS (Status);
925 }
```

new/usr/src/common/acpica/components/utilities/utstate.c

1

```
*****
10412 Fri Jan 17 07:30:50 2014
new/usr/src/common/acpica/components/utilities/utstate.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utstate - state object support procedures
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTSTATE_C__

47 #include "acpi.h"
48 #include "accommon.h"

50 #define _COMPONENT          ACPI_UTILITIES
51 #define ACPI_MODULE_NAME    ("utstate")

54 /*****
55 *
56 * FUNCTION:      AcpiUtCreatePkgStateAndPush
57 *
58 * PARAMETERS:    Object          - Object to be added to the new state
```

new/usr/src/common/acpica/components/utilities/utstate.c

2

```
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 ();
111     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);
313         return_PTR (NULL);
314     }

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);
355         return_PTR (NULL);
356     }

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 }
```

new/usr/src/common/acpica/components/utilities/utstring.c

1

```
*****
19423 Fri Jan 17 07:30:51 2014
new/usr/src/common/acpica/components/utilities/utstring.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: utstring - Common functions for strings and characters
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTSTRING_C__

47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acnamesp.h"

52 #define _COMPONENT          ACPI_UTILITIES
53       ACPI_MODULE_NAME    ("utstring")

56 /*
57 * Non-ANSI C library functions - strlwr,strupr, stricmp, and a 64-bit
58 * version of strtoul.
59 */
```

new/usr/src/common/acpica/components/utilities/utstring.c

2

```
61 #ifdef ACPI_ASL_COMPILER
62 /*****
63 *
64 * FUNCTION:      AcpiUtStrlwr (strlwr)
65 *
66 * PARAMETERS:    SrcString      - The source string to convert
67 *
68 * RETURN:        None
69 *
70 * DESCRIPTION:    Convert string to lowercase
71 *
72 * NOTE: This is not a POSIX function, so it appears here, not in utclib.c
73 *
74 *****/

76 void
77 AcpiUtStrlwr (
78     char                *SrcString)
79 {
80     char                *String;

83     ACPI_FUNCTION_ENTRY ();

86     if (!SrcString)
87     {
88         return;
89     }

91     /* Walk entire string, lowercasing the letters */

93     for (String = SrcString; *String; String++)
94     {
95         *String = (char) ACPI_TOLOWER (*String);
96     }

98     return;
99 }

102 /*****
103 *
104 * FUNCTION:      AcpiUtStricmp (stricmp)
105 *
106 * PARAMETERS:    String1        - first string to compare
107 *                String2        - second string to compare
108 *
109 * RETURN:        int that signifies string relationship. Zero means strings
110 *                are equal.
111 *
112 * DESCRIPTION:    Implementation of the non-ANSI stricmp function (compare
113 *                strings with no case sensitivity)
114 *
115 *****/

117 int
118 AcpiUtStricmp (
119     char                *String1,
120     char                *String2)
121 {
122     int                c1;
123     int                c2;
```

```

126     do
127     {
128         c1 = tolower ((int) *String1);
129         c2 = tolower ((int) *String2);

131         String1++;
132         String2++;
133     }
134     while ((c1 == c2) && (c1));

136     return (c1 - c2);
137 }
138 #endif

141 /*****
142  *
143  * FUNCTION:      AcpiUtStrupr (strupr)
144  *
145  * PARAMETERS:    SrcString      - The source string to convert
146  *
147  * RETURN:        None
148  *
149  * DESCRIPTION:   Convert string to uppercase
150  *
151  * NOTE: This is not a POSIX function, so it appears here, not in utclib.c
152  *
153  *****/

155 void
156 AcpiUtStrupr (
157     char          *SrcString)
158 {
159     char          *String;

162     ACPI_FUNCTION_ENTRY ();

165     if (!SrcString)
166     {
167         return;
168     }

170     /* Walk entire string, uppercasing the letters */

172     for (String = SrcString; *String; String++)
173     {
174         *String = (char) ACPI_TOUPPER (*String);
175     }

177     return;
178 }

181 /*****
182  *
183  * FUNCTION:      AcpiUtStrtoul64
184  *
185  * PARAMETERS:    String          - Null terminated string
186  *                Base            - Radix of the string: 16 or ACPI_ANY_BASE;
187  *                ACPI_ANY_BASE means 'in behalf of ToInteger'
188  *                RetInteger       - Where the converted integer is returned
189  *
190  * RETURN:        Status and Converted value
191  *

```

```

192  * DESCRIPTION:   Convert a string into an unsigned value. Performs either a
193  *               32-bit or 64-bit conversion, depending on the current mode
194  *               of the interpreter.
195  *               NOTE: Does not support Octal strings, not needed.
196  *
197  *****/

199 ACPI_STATUS
200 AcpiUtStrtoul64 (
201     char          *String,
202     UINT32        Base,
203     UINT64        *RetInteger)
204 {
205     UINT32        ThisDigit = 0;
206     UINT64        ReturnValue = 0;
207     UINT64        Quotient;
208     UINT64        Dividend;
209     UINT32        ToIntegerOp = (Base == ACPI_ANY_BASE);
210     UINT32        Mode32 = (AcpiGbl_IntegerByteWidth == 4);
211     UINT8          ValidDigits = 0;
212     UINT8          SignOf0x = 0;
213     UINT8          Term = 0;

216     ACPI_FUNCTION_TRACE_STR (UtStrtoul64, String);

219     switch (Base)
220     {
221     case ACPI_ANY_BASE:
222     case 16:

224         break;

226     default:

228         /* Invalid Base */

230         return ACPI_STATUS (AE_BAD_PARAMETER);
231     }

233     if (!String)
234     {
235         goto ErrorExit;
236     }

238     /* Skip over any white space in the buffer */

240     while ((*String) && (ACPI_IS_SPACE (*String) || *String == '\t'))
241     {
242         String++;
243     }

245     if (ToIntegerOp)
246     {
247         /*
248          * Base equal to ACPI_ANY_BASE means 'ToInteger operation case'.
249          * We need to determine if it is decimal or hexadecimal.
250          */
251         if ((*String == '0') && (ACPI_TOLOWER (*(String + 1)) == 'x'))
252         {
253             SignOf0x = 1;
254             Base = 16;

256             /* Skip over the leading '0x' */
257             String += 2;

```



```

258     }
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 '\\':
465     case '\"':
466     case '\\':
467         /* Single Quote */
468         /* Double Quote */
469         /* 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 }

```

```

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 *****/

```

```

557 BOOLEAN
558 AcpiUtValidAcpiName (
559     char                *Name)
560 {
561     UINT32               i;

```

```

564     ACPI_FUNCTION_ENTRY ();

```

```

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 }

```

```

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     *****/

```

```

601 void
602 AcpiUtRepairName (
603     char                *Name)
604 {
605     UINT32               i;
606     BOOLEAN              FoundBadChar = FALSE;
607     UINT32               OriginalName;

```

```

610     ACPI_FUNCTION_NAME (UtRepairName);

```

```

613     ACPI_MOVE_NAME (&OriginalName, Name);

```

```

615     /* Check each character in the name */

```

```

617     for (i = 0; i < ACPI_NAME_SIZE; i++)
618     {
619         if (AcpiUtValidAcpiChar (Name[i], i))
620         {
621             continue;
622         }

```

```

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     }

```

```

633     if (FoundBadChar)
634     {
635         /* Report warning only if in strict mode or debug mode */

```

```

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 }

```

```

653 #if defined ACPI_AS_L_COMPILER || defined ACPI_EXEC_APP

```

```

654 /*****
655 *
656 * FUNCTION:      UtConvertBackslashes
657 *
658 * PARAMETERS:    Pathname      - File pathname string to be converted
659 *
660 * RETURN:        Modifies the input Pathname
661 *
662 * DESCRIPTION:    Convert all backslashes (0x5C) to forward slashes (0x2F) within
663 *                  the entire input file pathname string.
664 *
665 *****/
667 void
668 UtConvertBackslashes (
669     char          *Pathname)
670 {
672     if (!Pathname)
673     {
674         return;
675     }
677     while (*Pathname)
678     {
679         if (*Pathname == '\\')
680         {
681             *Pathname = '/';
682         }
684         Pathname++;
685     }
686 }
687 #endif
689 #if defined (ACPI_DEBUGGER) || defined (ACPI_APPLICATION)
690 /*****
691 *
692 * FUNCTION:      AcpiUtSafeStrcpy, AcpiUtSafeStrcat, AcpiUtSafeStrncat
693 *
694 * PARAMETERS:    Adds a "DestSize" parameter to each of the standard string
695 *                  functions. This is the size of the Destination buffer.
696 *
697 * RETURN:        TRUE if the operation would overflow the destination buffer.
698 *
699 * DESCRIPTION:    Safe versions of standard Clib string functions. Ensure that
700 *                  the result of the operation will not overflow the output string
701 *                  buffer.
702 *
703 * NOTE:          These functions are typically only helpful for processing
704 *                  user input and command lines. For most ACPICA code, the
705 *                  required buffer length is precisely calculated before buffer
706 *                  allocation, so the use of these functions is unnecessary.
707 *
708 *****/
710 BOOLEAN
711 AcpiUtSafeStrcpy (
712     char          *Dest,
713     ACPI_SIZE     DestSize,
714     char          *Source)
715 {
717     if (ACPI_STRLEN (Source) >= DestSize)
718     {
719         return (TRUE);

```

```

720     }
722     ACPI_STRCPY (Dest, Source);
723     return (FALSE);
724 }
726 BOOLEAN
727 AcpiUtSafeStrcat (
728     char          *Dest,
729     ACPI_SIZE     DestSize,
730     char          *Source)
731 {
733     if ((ACPI_STRLEN (Dest) + ACPI_STRLEN (Source)) >= DestSize)
734     {
735         return (TRUE);
736     }
738     ACPI_STRCAT (Dest, Source);
739     return (FALSE);
740 }
742 BOOLEAN
743 AcpiUtSafeStrncat (
744     char          *Dest,
745     ACPI_SIZE     DestSize,
746     char          *Source,
747     ACPI_SIZE     MaxTransferLength)
748 {
749     ACPI_SIZE     ActualTransferLength;
752     ActualTransferLength = ACPI_MIN (MaxTransferLength, ACPI_STRLEN (Source));
754     if ((ACPI_STRLEN (Dest) + ActualTransferLength) >= DestSize)
755     {
756         return (TRUE);
757     }
759     ACPI_STRNCAT (Dest, Source, MaxTransferLength);
760     return (FALSE);
761 }
762 #endif

```

new/usr/src/common/acpica/components/utilities/uttrack.c

1

```
*****
23096 Fri Jan 17 07:30:51 2014
new/usr/src/common/acpica/components/utilities/uttrack.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: uttrack - Memory allocation tracking routines (debug only)
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 * 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 MERCHANTABILITY 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 */
43
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 */
54
55 #define __UTTRACK_C__
56
57 #include "acpi.h"
```

new/usr/src/common/acpica/components/utilities/uttrack.c

2

```
58 #include "accommon.h"
59
60 #ifdef ACPI_DBG_TRACK_ALLOCATIONS
61
62 #define _COMPONENT ACPI_UTILITIES
63 #define ACPI_MODULE_NAME ("uttrack")
64
65 /* Local prototypes */
66
67 static ACPI_DEBUG_MEM_BLOCK *
68 AcpiUtFindAllocation (
69     ACPI_DEBUG_MEM_BLOCK *Allocation);
70
71 void
72 AcpiUtRemoveAllocation (
73     ACPI_DEBUG_MEM_BLOCK *Allocation,
74     ACPI_SIZE Size,
75     ACPI_ALLOC_TYPE AllocType,
76     ACPI_COMPONENT Component,
77     const char *Module,
78     ACPI_LINE Line);
79
80 static ACPI_STATUS
81 AcpiUtTrackAllocation (
82     ACPI_DEBUG_MEM_BLOCK *Allocation,
83     ACPI_SIZE Size,
84     ACPI_ALLOC_TYPE AllocType,
85     ACPI_COMPONENT Component,
86     const char *Module,
87     ACPI_LINE Line);
88
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 purposes
100 *
101 *****/
102
103 ACPI_STATUS
104 AcpiUtCreateList (
105     char *ListName,
106     ACPI_UINT16 ObjectSize,
107     ACPI_MEMORY_LIST **ReturnCache)
108 {
109     ACPI_MEMORY_LIST *Cache;
110
111     Cache = AcpiOsAllocate (sizeof (ACPI_MEMORY_LIST));
112     if (!Cache)
113     {
114         return (AE_NO_MEMORY);
115     }
116
117     ACPI_MEMSET (Cache, 0, sizeof (ACPI_MEMORY_LIST));
118
119     Cache->ListName = ListName;
120     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));
164     Allocation = AcpiUtAllocate (Size + sizeof (ACPI_DEBUG_MEM_HEADER),
165         Component, Module, Line);
166     if (!Allocation)
167     {
168         /* Report allocation error */
169         ACPI_WARNING ((Module, Line,
170             "Could not allocate size %u", (UINT32) Size));
171     }
172     return (NULL);

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));
230     Allocation = AcpiUtAllocateZeroed (Size + sizeof (ACPI_DEBUG_MEM_HEADER),
231         Component, Module, Line);
232     if (!Allocation)
233     {
234         /* Report allocation error */
235         ACPI_ERROR ((Module, Line,
236             "Could not allocate size %u", (UINT32) Size));
237     }
238     return (NULL);

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 }

259 /*****
260 *
261 * FUNCTION:    AcpiUtFreeAndTrack
262 *
263 * PARAMETERS:  Allocation      - Address of the memory to deallocate
264 *              Component      - Component type of caller
265 *              Module         - Source file name of caller
266 *              Line           - Line number of caller
267 *
268 * RETURN:      None
269 *
270 * DESCRIPTION: Frees the memory at Allocation
271 *
272 *****/

274 void
275 AcpiUtFreeAndTrack (
276     void                *Allocation,
277     UINT32              Component,
278     const char          *Module,
279     UINT32              Line)
280 {
281     ACPI_DEBUG_MEM_BLOCK *DebugBlock;
282     ACPI_STATUS           Status;

285     ACPI_FUNCTION_TRACE_PTR (UtFree, Allocation);

288     if (NULL == Allocation)
289     {
290         ACPI_ERROR ((Module, Line,
291             "Attempt to delete a NULL address"));
293         return_VOID;
294     }

296     DebugBlock = ACPI_CAST_PTR (ACPI_DEBUG_MEM_BLOCK,
297         (((char *) Allocation) - sizeof (ACPI_DEBUG_MEM_HEADER)));

299     AcpiGbl_GlobalList->TotalFreed++;
300     AcpiGbl_GlobalList->CurrentTotalSize -= DebugBlock->Size;

302     Status = AcpiUtRemoveAllocation (DebugBlock,
303         Component, Module, Line);
304     if (ACPI_FAILURE (Status))
305     {
306         ACPI_EXCEPTION ((AE_INFO, Status, "Could not free memory"));
307     }

309     AcpiOsFree (DebugBlock);
310     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "%p freed (block %p)\n",
311         Allocation, DebugBlock));
311     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "%p freed\n", Allocation));
312     return_VOID;
313 }

```

```

316 /*****
317 *
318 * FUNCTION:    AcpiUtFindAllocation
319 *
320 * PARAMETERS:  Allocation      - Address of allocated memory
321 *
322 * RETURN:      Three cases:
323 *              1) List is empty, NULL is returned.
324 *              2) Element was found. Returns Allocation parameter.
325 *              3) Element was not found. Returns position where it should be
326 *                  inserted into the list.
329 * RETURN:      A list element if found; NULL otherwise.
327 *
328 * DESCRIPTION: Searches for an element in the global allocation tracking list.
329 *              If the element is not found, returns the location within the
330 *              list where the element should be inserted.
331 *
332 *              Note: The list is ordered by larger-to-smaller addresses.
333 *
334 *              This global list is used to detect memory leaks in ACPICA as
335 *              well as other issues such as an attempt to release the same
336 *              internal object more than once. Although expensive as far
337 *              as cpu time, this list is much more helpful for finding these
338 *              types of issues than using memory leak detectors outside of
339 *              the ACPICA code.
340 *
341 *****/

343 static ACPI_DEBUG_MEM_BLOCK *
344 AcpiUtFindAllocation (
345     ACPI_DEBUG_MEM_BLOCK *Allocation)
346 {
347     void *Allocation;
348     {
349         ACPI_DEBUG_MEM_BLOCK *Element;

312         ACPI_FUNCTION_ENTRY ();

350         Element = AcpiGbl_GlobalList->ListHead;
351         if (!Element)
352         {
353             return (NULL);
354         }

356         /*
357          * Search for the address.
358          *
359          * Note: List is ordered by larger-to-smaller addresses, on the
360          * assumption that a new allocation usually has a larger address
361          * than previous allocations.
362          */
363         while (Element > Allocation)
364         {
365             /* Check for end-of-list */
366             /* Search for the address. */

367             if (!Element->Next)
368                 while (Element)
369                 {
370                     if (Element == Allocation)
371                     {
372                         return (Element);
373                     }
374                     Element = Element->Next;
375                 }
376             Element = Element->Next;
377         }
378     }

```

```

373     }

375     if (Element == Allocation)
376     {
377         return (Element);
378     }

380     return (Element->Previous);
329     return (NULL);
381 }

384 /*****
385  *
386  * FUNCTION:      AcpiUtTrackAllocation
387  *
388  * PARAMETERS:    Allocation          - Address of allocated memory
389  *                Size                - Size of the allocation
390  *                AllocType           - MEM_MALLOC or MEM_CALLOC
391  *                Component           - Component type of caller
392  *                Module              - Source file name of caller
393  *                Line                - Line number of caller
394  *
395  * RETURN:        Status
344  * RETURN:        None.
396  *
397  * DESCRIPTION:   Inserts an element into the global allocation tracking list.
398  *
399  *****/

401 static ACPI_STATUS
402 AcpiUtTrackAllocation (
403     ACPI_DEBUG_MEM_BLOCK *Allocation,
404     ACPI_SIZE             Size,
405     UINT8                 AllocType,
406     UINT32                Component,
407     const char            *Module,
408     UINT32                Line)
409 {
410     ACPI_MEMORY_LIST      *MemList;
411     ACPI_DEBUG_MEM_BLOCK *Element;
412     ACPI_STATUS           Status = AE_OK;

415     ACPI_FUNCTION_TRACE_PTR (UtTrackAllocation, Allocation);

418     if (AcpiGbl_DisableMemTracking)
419     {
420         return_ACPI_STATUS (AE_OK);
421     }

423     MemList = AcpiGbl_GlobalList;
424     Status = AcpiUtAcquireMutex (ACPI_MTX_MEMORY);
425     if (ACPI_FAILURE (Status))
426     {
427         return_ACPI_STATUS (Status);
428     }

430     /*
431     * Search the global list for this address to make sure it is not
432     * already present. This will catch several kinds of problems.
433     * Search list for this address to make sure it is not already on the list.
434     * This will catch several kinds of problems.
435     */
436     Element = AcpiUtFindAllocation (Allocation);

```

```

435     if (Element == Allocation)
384     if (Element)
436     {
437         ACPI_ERROR ((AE_INFO,
438             "UtTrackAllocation: Allocation (%p) already present in global list!"
439             "UtTrackAllocation: Allocation already present in list! (%p)",
440             Allocation));

390         ACPI_ERROR ((AE_INFO, "Element %p Address %p",
391             Element, Allocation));

440         goto UnlockAndExit;
441     }

443     /* Fill in the instance data */
396     /* Fill in the instance data. */

445     Allocation->Size      = (UINT32) Size;
446     Allocation->AllocType = AllocType;
447     Allocation->Component = Component;
448     Allocation->Line      = Line;

450     ACPI_STRNCPY (Allocation->Module, Module, ACPI_MAX_MODULE_NAME);
451     Allocation->Module[ACPI_MAX_MODULE_NAME-1] = 0;

453     if (!Element)
454     {
455         /* Insert at list head */

457         if (MemList->ListHead)
458         {
459             ((ACPI_DEBUG_MEM_BLOCK *) (MemList->ListHead))->Previous = Allocation
460         }

462         Allocation->Next = MemList->ListHead;
463         Allocation->Previous = NULL;

465         MemList->ListHead = Allocation;
466     }
467     else
468     {
469         /* Insert after element */

471         Allocation->Next = Element->Next;
472         Allocation->Previous = Element;

474         if (Element->Next)
475         {
476             (Element->Next)->Previous = Allocation;
477         }

479         Element->Next = Allocation;
480     }

483 UnlockAndExit:
484     Status = AcpiUtReleaseMutex (ACPI_MTX_MEMORY);
485     return_ACPI_STATUS (Status);
486 }

489 /*****
490  *
491  * FUNCTION:      AcpiUtRemoveAllocation
492  *
493  * PARAMETERS:    Allocation          - Address of allocated memory

```



```

494 *          Component      - Component type of caller
495 *          Module         - Source file name of caller
496 *          Line           - Line number of caller
497 *
498 * RETURN:      Status
499 * RETURN:
500 * DESCRIPTION: Deletes an element from the global allocation tracking list.
501 *
502 *****/

504 static ACPI_STATUS
505 AcpiUtRemoveAllocation (
506     ACPI_DEBUG_MEM_BLOCK *Allocation,
507     UINT32 Component,
508     const char *Module,
509     UINT32 Line)
510 {
511     ACPI_MEMORY_LIST *MemList;
512     ACPI_STATUS Status;

515     ACPI_FUNCTION_NAME (UtRemoveAllocation);
516     ACPI_FUNCTION_TRACE (UtRemoveAllocation);

518     if (AcpiGbl_DisableMemTracking)
519     {
520         return (AE_OK);
521         return ACPI_STATUS (AE_OK);
522     }

523     MemList = AcpiGbl_GlobalList;
524     if (NULL == MemList->ListHead)
525     {
526         /* No allocations! */

528         ACPI_ERROR ((Module, Line,
529             "Empty allocation list, nothing to free!"));

531         return (AE_OK);
532         return ACPI_STATUS (AE_OK);
533     }

534     Status = AcpiUtAcquireMutex (ACPI_MTX_MEMORY);
535     if (ACPI_FAILURE (Status))
536     {
537         return (Status);
538         return ACPI_STATUS (Status);
539     }

540     /* Unlink */

542     if (Allocation->Previous)
543     {
544         (Allocation->Previous)->Next = Allocation->Next;
545     }
546     else
547     {
548         MemList->ListHead = Allocation->Next;
549     }

551     if (Allocation->Next)
552     {
553         (Allocation->Next)->Previous = Allocation->Previous;
554     }

```

```

556     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "Freeing %p, size 0%X\n",
557         &Allocation->UserSpace, Allocation->Size));

559     /* Mark the segment as deleted */

561     ACPI_MEMSET (&Allocation->UserSpace, 0xEA, Allocation->Size);

562     ACPI_DEBUG_PRINT ((ACPI_DB_ALLOCATIONS, "Freeing size 0%X\n",
563         Allocation->Size));

564     Status = AcpiUtReleaseMutex (ACPI_MTX_MEMORY);
565     return (Status);
566     return ACPI_STATUS (Status);
567 }

568 /*****
569 *
570 * FUNCTION:      AcpiUtDumpAllocationInfo
571 *
572 * PARAMETERS:    None
573 *
574 * RETURN:        None
575 *
576 * DESCRIPTION:   Print some info about the outstanding allocations.
577 *
578 *****/

580 void
581 AcpiUtDumpAllocationInfo (
582     void)
583 {
584     /*
585     ACPI_MEMORY_LIST *MemList;
586     */

588     ACPI_FUNCTION_TRACE (UtDumpAllocationInfo);

590     /*
591     ACPI_DEBUG_PRINT (TRACE_ALLOCATIONS | TRACE_TABLES,
592         ("%30s: %4d (%3d Kb)\n", "Current allocations",
593         MemList->CurrentCount,
594         ROUND_UP_TO_1K (MemList->CurrentSize)));

596     ACPI_DEBUG_PRINT (TRACE_ALLOCATIONS | TRACE_TABLES,
597         ("%30s: %4d (%3d Kb)\n", "Max concurrent allocations",
598         MemList->MaxConcurrentCount,
599         ROUND_UP_TO_1K (MemList->MaxConcurrentSize)));

602     ACPI_DEBUG_PRINT (TRACE_ALLOCATIONS | TRACE_TABLES,
603         ("%30s: %4d (%3d Kb)\n", "Total (all) internal objects",
604         RunningObjectCount,
605         ROUND_UP_TO_1K (RunningObjectSize)));

607     ACPI_DEBUG_PRINT (TRACE_ALLOCATIONS | TRACE_TABLES,
608         ("%30s: %4d (%3d Kb)\n", "Total (all) allocations",
609         RunningAllocCount,
610         ROUND_UP_TO_1K (RunningAllocSize)));

613     ACPI_DEBUG_PRINT (TRACE_ALLOCATIONS | TRACE_TABLES,
614         ("%30s: %4d (%3d Kb)\n", "Current Nodes",
615         AcpiGbl_CurrentNodeCount,

```

```

628 /*****
629 *
630 * FUNCTION:      AcpiUtDumpAllocations
631 *
632 * PARAMETERS:    Component          - Component(s) to dump info for.
633 *                Module             - Module to dump info for. NULL means all.
634 *
635 * RETURN:        None
636 *
637 * DESCRIPTION:    Print a list of all outstanding allocations.
638 *
639 *****/

```

```

701         if (Element->Size == sizeof (ACPI_OPERAND_OBJECT))
702             if (Element->Size == sizeof (ACPI_DESC_TYPE_OPERAND))
703             {
704                 DescriptorType = ACPI_DESC_TYPE_OPERAND;
705             }
706             break;
707
708     case ACPI_DESC_TYPE_PARSER:
709
710         if (Element->Size == sizeof (ACPI_PARSE_OBJECT))
711             if (Element->Size == sizeof (ACPI_DESC_TYPE_PARSER))
712             {
713                 DescriptorType = ACPI_DESC_TYPE_PARSER;
714             }
715             break;
716
717     case ACPI_DESC_TYPE_NAMED:
718
719         if (Element->Size == sizeof (ACPI_NAMESPACE_NODE))
720             if (Element->Size == sizeof (ACPI_DESC_TYPE_NAMED))
721             {
722                 DescriptorType = ACPI_DESC_TYPE_NAMED;
723             }
724             break;
725
726     default:
727
728         break;
729
730     /* Display additional info for the major descriptor types */
731
732     switch (DescriptorType)
733     {
734     case ACPI_DESC_TYPE_OPERAND:
735
736         AcpiOsPrintf ("%12.12s  RefCount 0x%04X\n",
737             AcpiUtGetTypeName (Descriptor->Object.Common.Type),
738             Descriptor->Object.Common.ReferenceCount);
739         break;

```

```
743             break;
745             case ACPI_DESC_TYPE_NAMED:
747                 AcpiOsPrintf ("%4.4s\n",
748                 AcpiUtGetNodeName (&Descriptor->Node));
749                 break;
751             default:
753                 AcpiOsPrintf ( "\n");
754                 break;
755             }
756         }
757     }
759     NumOutstanding++;
760 }
762     Element = Element->Next;
763 }
765 (void) AcpiUtReleaseMutex (ACPI_MTX_MEMORY);
767 /* Print summary */
769 if (!NumOutstanding)
770 {
771     ACPI_INFO ((AE_INFO, "No outstanding allocations"));
772 }
773 else
774 {
775     ACPI_ERROR ((AE_INFO, "%u(0x%X) Outstanding allocations",
776     NumOutstanding, NumOutstanding));
777 }
779     return_VOID;
780 }
782 #endif /* ACPI_DBG_TRACK_ALLOCATIONS */
```

new/usr/src/common/acpica/components/utilities/utxface.c

1

```
*****
18241 Fri Jan 17 07:30:51 2014
new/usr/src/common/acpica/components/utilities/utxface.c
update to acpica-unix2-20140114
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: utxface - External interfaces, miscellaneous utility functions
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define __UTXFACE_C__
45 #define EXPORT ACPI_INTERFACES
46
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acdebug.h"
50
51 #define _COMPONENT ACPI_UTILITIES
52 #define ACPI_MODULE_NAME ("utxface")
53
54
55 /*****
56 *
57 *****/
```

new/usr/src/common/acpica/components/utilities/utxface.c

2

```
58 * FUNCTION: AcpiTerminate
59 *
60 * PARAMETERS: None
61 *
62 * RETURN: Status
63 *
64 * DESCRIPTION: Shutdown the ACPICA subsystem and release all resources.
65 *
66 *****/
67
68 ACPI_STATUS
69 AcpiTerminate (
70     void)
71 {
72     ACPI_STATUS Status;
73
74
75     ACPI_FUNCTION_TRACE (AcpiTerminate);
76
77
78     /* Just exit if subsystem is already shutdown */
79
80     if (AcpiGbl_Shutdown)
81     {
82         ACPI_ERROR ((AE_INFO, "ACPI Subsystem is already terminated"));
83         return ACPI_STATUS (AE_OK);
84     }
85
86     /* Subsystem appears active, go ahead and shut it down */
87
88     AcpiGbl_Shutdown = TRUE;
89     AcpiGbl_StartupFlags = 0;
90     ACPI_DEBUG_PRINT ((ACPI_DB_INFO, "Shutting down ACPI Subsystem\n"));
91
92     /* Terminate the AML Debugger if present */
93
94     ACPI_DEBUGGER_EXEC (AcpiGbl_DbTerminateThreads = TRUE);
95
96     /* Shutdown and free all resources */
97
98     AcpiUtSubsystemShutdown ();
99
100    /* Free the mutex objects */
101
102    AcpiUtMutexTerminate ();
103
104
105    #ifdef ACPI_DEBUGGER
106
107        /* Shut down the debugger */
108
109        AcpiDbTerminate ();
110    #endif
111
112    /* Now we can shutdown the OS-dependent layer */
113
114    Status = AcpiOsTerminate ();
115    return ACPI_STATUS (Status);
116 }
117
118 ACPI_EXPORT_SYMBOL_INIT (AcpiTerminate)
119
120
121 #ifndef ACPI_ASL_COMPILER
122 /*****
123 *
124 *****/
```

```

124 * FUNCTION:      AcpiSubsystemStatus
125 *
126 * PARAMETERS:    None
127 *
128 * RETURN:        Status of the ACPI subsystem
129 *
130 * DESCRIPTION:    Other drivers that use the ACPI subsystem should call this
131 *                 before making any other calls, to ensure the subsystem
132 *                 initialized successfully.
133 *
134 *****/
136 ACPI_STATUS
137 AcpiSubsystemStatus (
138     void)
139 {
141     if (AcpiGbl_StartupFlags & ACPI_INITIALIZED_OK)
142     {
143         return (AE_OK);
144     }
145     else
146     {
147         return (AE_ERROR);
148     }
149 }
151 ACPI_EXPORT_SYMBOL (AcpiSubsystemStatus)
154 *****/
155 *
156 * FUNCTION:      AcpiGetSystemInfo
157 *
158 * PARAMETERS:    OutBuffer      - A buffer to receive the resources for the
159 *                                     device
160 *
161 * RETURN:        Status          - the status of the call
162 *
163 * DESCRIPTION:    This function is called to get information about the current
164 *                 state of the ACPI subsystem. It will return system information
165 *                 in the OutBuffer.
166 *
167 *                 If the function fails an appropriate status will be returned
168 *                 and the value of OutBuffer is undefined.
169 *
170 *****/
172 ACPI_STATUS
173 AcpiGetSystemInfo (
174     ACPI_BUFFER      *OutBuffer)
175 {
176     ACPI_SYSTEM_INFO *InfoPtr;
177     ACPI_STATUS      Status;
180     ACPI_FUNCTION_TRACE (AcpiGetSystemInfo);
183     /* Parameter validation */
185     Status = AcpiUtValidateBuffer (OutBuffer);
186     if (ACPI_FAILURE (Status))
187     {
188         return_ACPI_STATUS (Status);
189     }

```

```

191     /* Validate/Allocate/Clear caller buffer */
193     Status = AcpiUtInitializeBuffer (OutBuffer, sizeof (ACPI_SYSTEM_INFO));
194     if (ACPI_FAILURE (Status))
195     {
196         return_ACPI_STATUS (Status);
197     }
199     /*
200      * Populate the return buffer
201      */
202     InfoPtr = (ACPI_SYSTEM_INFO *) OutBuffer->Pointer;
204     InfoPtr->AcpiCaVersion = ACPI_CA_VERSION;
206     /* System flags (ACPI capabilities) */
208     InfoPtr->Flags = ACPI_SYS_MODE ACPI;
210     /* Timer resolution - 24 or 32 bits */
212     if (AcpiGbl_FADT.Flags & ACPI_FADT_32BIT_TIMER)
213     {
214         InfoPtr->TimerResolution = 24;
215     }
216     else
217     {
218         InfoPtr->TimerResolution = 32;
219     }
221     /* Clear the reserved fields */
223     InfoPtr->Reserved1 = 0;
224     InfoPtr->Reserved2 = 0;
226     /* Current debug levels */
228     InfoPtr->DebugLayer = AcpiDbgLayer;
229     InfoPtr->DebugLevel = AcpiDbgLevel;
231     return_ACPI_STATUS (AE_OK);
232 }
234 ACPI_EXPORT_SYMBOL (AcpiGetSystemInfo)
237 *****/
238 *
239 * FUNCTION:      AcpiGetStatistics
240 *
241 * PARAMETERS:    Stats          - Where the statistics are returned
242 *
243 * RETURN:        Status          - the status of the call
244 *
245 * DESCRIPTION:    Get the contents of the various system counters
246 *
247 *****/
249 ACPI_STATUS
250 AcpiGetStatistics (
251     ACPI_STATISTICS *Stats)
252 {
253     ACPI_FUNCTION_TRACE (AcpiGetStatistics);

```

```

256  /* Parameter validation */
257
258  if (!Stats)
259  {
260      return ACPI_STATUS (AE_BAD_PARAMETER);
261  }
262
263  /* Various interrupt-based event counters */
264
265  Stats->SciCount = AcpiSciCount;
266  Stats->GpeCount = AcpiGpeCount;
267
268  ACPI_MEMCPY (Stats->FixedEventCount, AcpiFixedEventCount,
269              sizeof (AcpiFixedEventCount));
270
271
272  /* Other counters */
273
274  Stats->MethodCount = AcpiMethodCount;
275
276  return ACPI_STATUS (AE_OK);
277 }
278
279 ACPI_EXPORT_SYMBOL (AcpiGetStatistics)
280
281
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 *****/
296
297 ACPI_STATUS
298 AcpiInstallInitializationHandler (
299     ACPI_INIT_HANDLER    Handler,
300     UINT32               Function)
301 {
302
303     if (!Handler)
304     {
305         return (AE_BAD_PARAMETER);
306     }
307
308     if (AcpiGbl_InitHandler)
309     {
310         return (AE_ALREADY_EXISTS);
311     }
312
313     AcpiGbl_InitHandler = Handler;
314     return (AE_OK);
315 }
316
317 ACPI_EXPORT_SYMBOL (AcpiInstallInitializationHandler)
318
319
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 *****/
331
332 ACPI_STATUS
333 AcpiPurgeCachedObjects (
334     void)
335 {
336     ACPI_FUNCTION_TRACE (AcpiPurgeCachedObjects);
337
338
339     (void) AcpiOsPurgeCache (AcpiGbl_StateCache);
340     (void) AcpiOsPurgeCache (AcpiGbl_OperandCache);
341     (void) AcpiOsPurgeCache (AcpiGbl_PsNodeCache);
342     (void) AcpiOsPurgeCache (AcpiGbl_PsNodeExtCache);
343
344     return ACPI_STATUS (AE_OK);
345 }
346
347 ACPI_EXPORT_SYMBOL (AcpiPurgeCachedObjects)
348
349
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 *****/
361
362 ACPI_STATUS
363 AcpiInstallInterface (
364     ACPI_STRING      InterfaceName)
365 {
366     ACPI_STATUS      Status;
367     ACPI_INTERFACE_INFO *InterfaceInfo;
368
369
370     /* Parameter validation */
371
372     if (!InterfaceName || (ACPI_STRLEN (InterfaceName) == 0))
373     {
374         return (AE_BAD_PARAMETER);
375     }
376
377     Status = AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
378     if (ACPI_FAILURE (Status))
379     {
380         return (Status);
381     }
382
383     /* Check if the interface name is already in the global list */
384
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 */
405
406      Status = AcpiUtInstallInterface (InterfaceName);
407  }
408
409  AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
410  return (Status);
411 }
412
413 ACPI_EXPORT_SYMBOL (AcpiInstallInterface)
414
415 /*****
416 *
417 * FUNCTION:      AcpiRemoveInterface
418 *
419 * PARAMETERS:    InterfaceName      - The interface to remove
420 *
421 * RETURN:        Status
422 *
423 * DESCRIPTION:   Remove an _OSI interface from the global list
424 *
425 *
426 *****/
427
428 ACPI_STATUS
429 AcpiRemoveInterface (
430     ACPI_STRING      InterfaceName)
431 {
432     ACPI_STATUS      Status;
433
434     /* Parameter validation */
435
436     if (!InterfaceName || (ACPI_STRLEN (InterfaceName) == 0))
437     {
438         return (AE_BAD_PARAMETER);
439     }
440
441     Status = AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
442     if (ACPI_FAILURE (Status))
443     {
444         return (Status);
445     }
446
447     Status = AcpiUtRemoveInterface (InterfaceName);
448
449     AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
450     return (Status);
451 }
452

```

```

454 ACPI_EXPORT_SYMBOL (AcpiRemoveInterface)
455
456 /*****
457 *
458 * FUNCTION:      AcpiInstallInterfaceHandler
459 *
460 * PARAMETERS:    Handler              - The _OSI interface handler to install
461 *                                     NULL means "remove existing handler"
462 *
463 * RETURN:        Status
464 *
465 * DESCRIPTION:   Install a handler for the predefined _OSI ACPI method.
466 *               invoked during execution of the internal implementation of
467 *               _OSI. A NULL handler simply removes any existing handler.
468 *
469 *
470 *****/
471
472 ACPI_STATUS
473 AcpiInstallInterfaceHandler (
474     ACPI_INTERFACE_HANDLER Handler)
475 {
476     ACPI_STATUS      Status;
477
478     Status = AcpiOsAcquireMutex (AcpiGbl_OsiMutex, ACPI_WAIT_FOREVER);
479     if (ACPI_FAILURE (Status))
480     {
481         return (Status);
482     }
483
484     if (Handler && AcpiGbl_InterfaceHandler)
485     {
486         Status = AE_ALREADY_EXISTS;
487     }
488     else
489     {
490         AcpiGbl_InterfaceHandler = Handler;
491     }
492
493     AcpiOsReleaseMutex (AcpiGbl_OsiMutex);
494     return (Status);
495 }
496
497 ACPI_EXPORT_SYMBOL (AcpiInstallInterfaceHandler)
498
499 /*****
500 *
501 * FUNCTION:      AcpiUpdateInterfaces
502 *
503 * PARAMETERS:    Action              - Actions to be performed during the
504 *                                     update
505 *
506 * RETURN:        Status
507 *
508 * DESCRIPTION:   Update _OSI interface strings, disabling or enabling OS vendor
509 *               string or/and feature group strings.
510 *
511 *
512 *****/
513
514 ACPI_STATUS
515 AcpiUpdateInterfaces (
516     UINT8          Action)
517 {
518     ACPI_STATUS      Status;
519

```

```

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)
```

new/usr/src/common/acpica/components/utilities/utxferror.c

1

```
*****
8737 Fri Jan 17 07:30:52 2014
new/usr/src/common/acpica/components/utilities/utxferror.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: utxferror - Various error/warning output functions
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTXFERROR_C__
45 #define EXPORT_ACPI_INTERFACES

47 #include "acpi.h"
48 #include "accommon.h"

51 #define _COMPONENT          ACPI_UTILITIES
52 #define ACPI_MODULE_NAME    ("utxferror")

54 /*
55 * This module is used for the in-kernel ACPICA as well as the ACPICA
56 * tools/applications.
57 */
```

new/usr/src/common/acpica/components/utilities/utxferror.c

2

```
59 /*****
60 *
61 * FUNCTION:      AcpiError
62 *
63 * PARAMETERS:    ModuleName      - Caller's module name (for error output)
64 *                LineNumber      - Caller's line number (for error output)
65 *                Format          - Printf format string + additional args
66 *
67 * RETURN:        None
68 *
69 * DESCRIPTION:    Print "ACPI Error" message with module/line/version info
70 *
71 *****/

73 void ACPI_INTERNAL_VAR_XFACE
74 AcpiError (
75     const char          *ModuleName,
76     UINT32              LineNumber,
77     const char          *Format,
78     ...)
79 {
80     va_list              ArgList;

83     ACPI_MSG_REDIRECT_BEGIN;
84     AcpiOsPrintf (ACPI_MSG_ERROR);

86     va_start (ArgList, Format);
87     AcpiOsVprintf (Format, ArgList);
88     ACPI_MSG_SUFFIX;
89     va_end (ArgList);

91     ACPI_MSG_REDIRECT_END;
92 }

94 ACPI_EXPORT_SYMBOL (AcpiError)

97 /*****
98 *
99 * FUNCTION:      AcpiException
100 *
101 * PARAMETERS:    ModuleName      - Caller's module name (for error output)
102 *                LineNumber      - Caller's line number (for error output)
103 *                Status          - Status to be formatted
104 *                Format          - Printf format string + additional args
105 *
106 * RETURN:        None
107 *
108 * DESCRIPTION:    Print "ACPI Exception" message with module/line/version info
109 *                  and decoded ACPI_STATUS.
110 *
111 *****/

113 void ACPI_INTERNAL_VAR_XFACE
114 AcpiException (
115     const char          *ModuleName,
116     UINT32              LineNumber,
117     ACPI_STATUS          Status,
118     const char          *Format,
119     ...)
120 {
121     va_list              ArgList;

124     ACPI_MSG_REDIRECT_BEGIN;
```

```

125     AcpiOsPrintf (ACPI_MSG_EXCEPTION "%s, ", AcpiFormatException (Status));

127     va_start (ArgList, Format);
128     AcpiOsVprintf (Format, ArgList);
129     ACPI_MSG_SUFFIX;
130     va_end (ArgList);

132     ACPI_MSG_REDIRECT_END;
133 }

135 ACPI_EXPORT_SYMBOL (AcpiException)

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 *****/

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;

162     ACPI_MSG_REDIRECT_BEGIN;
163     AcpiOsPrintf (ACPI_MSG_WARNING);

165     va_start (ArgList, Format);
166     AcpiOsVprintf (Format, ArgList);
167     ACPI_MSG_SUFFIX;
168     va_end (ArgList);

170     ACPI_MSG_REDIRECT_END;
171 }

173 ACPI_EXPORT_SYMBOL (AcpiWarning)

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 *****/

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;

203     ACPI_MSG_REDIRECT_BEGIN;
204     AcpiOsPrintf (ACPI_MSG_INFO);

206     va_start (ArgList, Format);
207     AcpiOsVprintf (Format, ArgList);
208     AcpiOsPrintf ("\n");
209     va_end (ArgList);

211     ACPI_MSG_REDIRECT_END;
212 }

214 ACPI_EXPORT_SYMBOL (AcpiInfo)

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 *****/

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;

242     ACPI_MSG_REDIRECT_BEGIN;
243     AcpiOsPrintf (ACPI_MSG_BIOS_ERROR);

245     va_start (ArgList, Format);
246     AcpiOsVprintf (Format, ArgList);
247     ACPI_MSG_SUFFIX;
248     va_end (ArgList);

250     ACPI_MSG_REDIRECT_END;
251 }

253 ACPI_EXPORT_SYMBOL (AcpiBiosError)

256 /*****

```

```
257 *
258 * FUNCTION:      AcpiBiosWarning
259 *
260 * PARAMETERS:    ModuleName      - Caller's module name (for error output)
261 *                LineNumber      - Caller's line number (for error output)
262 *                Format          - Printf format string + additional args
263 *
264 * RETURN:        None
265 *
266 * DESCRIPTION:   Print "ACPI Firmware Warning" message with module/line/version
267 *                info
268 *
269 *****/
271 void ACPI_INTERNAL_VAR_XFACE
272 AcpiBiosWarning (
273     const char      *ModuleName,
274     UINT32          LineNumber,
275     const char      *Format,
276     ...)
277 {
278     va_list          ArgList;
279
281     ACPI_MSG_REDIRECT_BEGIN;
282     AcpiOsPrintf (ACPI_MSG_BIOS_WARNING);
283
284     va_start (ArgList, Format);
285     AcpiOsVprintf (Format, ArgList);
286     ACPI_MSG_SUFFIX;
287     va_end (ArgList);
288
289     ACPI_MSG_REDIRECT_END;
290 }
292 ACPI_EXPORT_SYMBOL (AcpiBiosWarning)
```

new/usr/src/common/acpica/components/utilities/utxfinit.c

1

```
*****
11038 Fri Jan 17 07:30:52 2014
new/usr/src/common/acpica/components/utilities/utxfinit.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: utxfinit - External interfaces for ACPICA initialization
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define __UTXFINIT_C__
45 #define EXPORT_ACPICA_INTERFACES
46
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acevents.h"
50 #include "acnamesp.h"
51 #include "acdebug.h"
52 #include "actables.h"
53
54 #define _COMPONENT          ACPICA_UTILITIES
55 #define ACPI_MODULE_NAME    ("utxfinit")
```

new/usr/src/common/acpica/components/utilities/utxfinit.c

2

```
59 /*****
60 *
61 * FUNCTION:      AcpiInitializeSubsystem
62 *
63 * PARAMETERS:    None
64 *
65 * RETURN:        Status
66 *
67 * DESCRIPTION:   Initializes all global variables. This is the first function
68 *                called, so any early initialization belongs here.
69 *
70 *****/
71
72 ACPI_STATUS
73 AcpiInitializeSubsystem (
74     void)
75 {
76     ACPI_STATUS          Status;
77
78     ACPI_FUNCTION_TRACE (AcpiInitializeSubsystem);
79
80     AcpiGbl_StartupFlags = ACPI_SUBSYSTEM_INITIALIZE;
81     AcpiDebugExec (AcpiUtInitStackPtrTrace ());
82
83     /* Initialize the OS-Dependent layer */
84
85     Status = AcpiOsInitialize ();
86     if (ACPI_FAILURE (Status))
87     {
88         ACPI_EXCEPTION ((AE_INFO, Status, "During OSL initialization"));
89         return_ACPI_STATUS (Status);
90     }
91
92     /* Initialize all globals used by the subsystem */
93
94     Status = AcpiUtInitGlobals ();
95     if (ACPI_FAILURE (Status))
96     {
97         ACPI_EXCEPTION ((AE_INFO, Status, "During initialization of globals"));
98         return_ACPI_STATUS (Status);
99     }
100
101     /* Create the default mutex objects */
102
103     Status = AcpiUtMutexInitialize ();
104     if (ACPI_FAILURE (Status))
105     {
106         ACPI_EXCEPTION ((AE_INFO, Status, "During Global Mutex creation"));
107         return_ACPI_STATUS (Status);
108     }
109
110     /*
111     * Initialize the namespace manager and
112     * the root of the namespace tree
113     */
114     Status = AcpiNsRootInitialize ();
115     if (ACPI_FAILURE (Status))
116     {
117         ACPI_EXCEPTION ((AE_INFO, Status, "During Namespace initialization"));
118         return_ACPI_STATUS (Status);
119     }
120
121     /* Initialize the global OSI interfaces list with the static names */
122
123 }
```

```

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)

```

new/usr/src/common/acpica/components/utilities/utxfmutex.c

1

```
*****
6663 Fri Jan 17 07:30:52 2014
new/usr/src/common/acpica/components/utilities/utxfmutex.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: utxfmutex - external AML mutex access functions
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __UTXFMUTEX_C__

46 #include "acpi.h"
47 #include "accommon.h"
48 #include "acnamesp.h"

51 #define _COMPONENT          ACPI_UTILITIES
52 #define ACPI_MODULE_NAME    ("utxfmutex")

55 /* Local prototypes */

57 static ACPI_STATUS
58 AcpiUtGetMutexObject (
59     ACPI_HANDLE             Handle,
60     ACPI_STRING              Pathname,
```

new/usr/src/common/acpica/components/utilities/utxfmutex.c

2

```
61     ACPI_OPERAND_OBJECT     **RetObj);

64 /*****
65 *
66 * FUNCTION:      AcpiUtGetMutexObject
67 *
68 * PARAMETERS:    Handle             - Mutex or prefix handle (optional)
69 *                Pathname           - Mutex pathname (optional)
70 *                RetObj              - Where the mutex object is returned
71 *
72 * RETURN:        Status
73 *
74 * DESCRIPTION:   Get an AML mutex object. The mutex node is pointed to by
75 *                Handle:Pathname. Either Handle or Pathname can be NULL, but
76 *                not both.
77 *
78 *****/

80 static ACPI_STATUS
81 AcpiUtGetMutexObject (
82     ACPI_HANDLE             Handle,
83     ACPI_STRING              Pathname,
84     ACPI_OPERAND_OBJECT     **RetObj)
85 {
86     ACPI_NAMESPACE_NODE     *MutexNode;
87     ACPI_OPERAND_OBJECT     *MutexObj;
88     ACPI_STATUS              Status;

91     /* Parameter validation */

93     if (!RetObj || (!Handle && !Pathname))
94     {
95         return (AE_BAD_PARAMETER);
96     }

98     /* Get a the namespace node for the mutex */

100     MutexNode = Handle;
101     if (Pathname != NULL)
102     {
103         Status = AcpiGetHandle (Handle, Pathname,
104                                 ACPI_CAST_PTR (ACPI_HANDLE, &MutexNode));
105         if (ACPI_FAILURE (Status))
106         {
107             return (Status);
108         }
109     }

111     /* Ensure that we actually have a Mutex object */

113     if (!MutexNode ||
114         (MutexNode->Type != ACPI_TYPE_MUTEX))
115     {
116         return (AE_TYPE);
117     }

119     /* Get the low-level mutex object */

121     MutexObj = AcpiNsGetAttachedObject (MutexNode);
122     if (!MutexObj)
123     {
124         return (AE_NULL_OBJECT);
125     }
```



```

127     *RetObj = MutexObj;
128     return (AE_OK);
129 }

132 /*****
133  *
134  * FUNCTION:     AcpiAcquireMutex
135  *
136  * PARAMETERS:  Handle             - Mutex or prefix handle (optional)
137  *              Pathname           - Mutex pathname (optional)
138  *              Timeout            - Max time to wait for the lock (millisec)
139  *
140  * RETURN:      Status
141  *
142  * DESCRIPTION: Acquire an AML mutex. This is a device driver interface to
143  *              AML mutex objects, and allows for transaction locking between
144  *              drivers and AML code. The mutex node is pointed to by
145  *              Handle:Pathname. Either Handle or Pathname can be NULL, but
146  *              not both.
147  *
148  *****/

150 ACPI_STATUS
151 AcpiAcquireMutex (
152     ACPI_HANDLE      Handle,
153     ACPI_STRING      Pathname,
154     UINT16           Timeout)
155 {
156     ACPI_STATUS      Status;
157     ACPI_OPERAND_OBJECT *MutexObj;

160     /* Get the low-level mutex associated with Handle:Pathname */

162     Status = AcpiUtGetMutexObject (Handle, Pathname, &MutexObj);
163     if (ACPI_FAILURE (Status))
164     {
165         return (Status);
166     }

168     /* Acquire the OS mutex */

170     Status = AcpiOsAcquireMutex (MutexObj->Mutex.OsMutex, Timeout);
171     return (Status);
172 }

175 /*****
176  *
177  * FUNCTION:     AcpiReleaseMutex
178  *
179  * PARAMETERS:  Handle             - Mutex or prefix handle (optional)
180  *              Pathname           - Mutex pathname (optional)
181  *
182  * RETURN:      Status
183  *
184  * DESCRIPTION: Release an AML mutex. This is a device driver interface to
185  *              AML mutex objects, and allows for transaction locking between
186  *              drivers and AML code. The mutex node is pointed to by
187  *              Handle:Pathname. Either Handle or Pathname can be NULL, but
188  *              not both.
189  *
190  *****/

192 ACPI_STATUS

```

```

193 AcpiReleaseMutex (
194     ACPI_HANDLE      Handle,
195     ACPI_STRING      Pathname)
196 {
197     ACPI_STATUS      Status;
198     ACPI_OPERAND_OBJECT *MutexObj;

201     /* Get the low-level mutex associated with Handle:Pathname */

203     Status = AcpiUtGetMutexObject (Handle, Pathname, &MutexObj);
204     if (ACPI_FAILURE (Status))
205     {
206         return (Status);
207     }

209     /* Release the OS mutex */

211     AcpiOsReleaseMutex (MutexObj->Mutex.OsMutex);
212     return (AE_OK);
213 }

```

new/usr/src/common/acpica/include/acapps.h

1

```
*****
6054 Fri Jan 17 07:30:52 2014
new/usr/src/common/acpica/include/acapps.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: acapps - common include for ACPI applications/tools
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef _ACAPPS
45 #define _ACAPPS

48 #ifdef _MSC_VER /* disable some level-4 warnings */
49 #pragma warning(disable:4100) /* warning C4100: unreferenced formal parameter
50 #endif

52 /* Common info for tool signons */

54 #define ACPICA_NAME "Intel ACPI Component Architecture"
55 #define ACPICA_COPYRIGHT "Copyright (c) 2000 - 2014 Intel Corporation"
56 #define ACPICA_COPYRIGHT "Copyright (c) 2000 - 2011 Intel Corporation"

57 #if ACPI_MACHINE_WIDTH == 64
```

new/usr/src/common/acpica/include/acapps.h

2

```
58 #define ACPI_WIDTH "-64"

60 #elif ACPI_MACHINE_WIDTH == 32
61 #define ACPI_WIDTH "-32"

63 #else
64 #error unknown ACPI_MACHINE_WIDTH
65 #define ACPI_WIDTH "-???"

67 #endif

69 /* Macros for signons and file headers */

71 #define ACPI_COMMON_SIGNON(UtilityName) \
72 "\n%s\n%s version %8.8X%s [%s]\n%s\n\n", \
73 ACPICA_NAME, \
74 UtilityName, ((UINT32) ACPI_CA_VERSION), ACPI_WIDTH, __DATE__, \
75 ACPICA_COPYRIGHT

77 #define ACPI_COMMON_HEADER(UtilityName, Prefix) \
78 "%s\n%s version %8.8X%s [%s]\n%s\n\n", \
79 Prefix, ACPICA_NAME, \
80 Prefix, UtilityName, ((UINT32) ACPI_CA_VERSION), ACPI_WIDTH, __DATE__, \
81 Prefix, ACPICA_COPYRIGHT, \
82 Prefix

84 /* Macros for usage messages */

86 #define ACPI_USAGE_HEADER(Usage) \
87 printf ("Usage: %s\nOptions:\n", Usage);

89 #define ACPI_OPTION(Name, Description) \
90 printf (" %-18s%s\n", Name, Description);

93 #define FILE_SUFFIX_DISASSEMBLY "dsl"
94 #define ACPI_TABLE_FILE_SUFFIX ".dat"

97 /*
98 * getopt
99 */
100 int
101 AcpiGetopt(
102 int argc,
103 char **argv,
104 char *opts);

106 int
107 AcpiGetoptArgument (
108 int argc,
109 char **argv);

111 extern int AcpiGbl_Optind;
112 extern int AcpiGbl_Opterr;
113 extern int AcpiGbl_SubOptChar;
114 extern char *AcpiGbl_Optarg;

117 #ifndef ACPI_DUMP_APP
118 /*
119 * adisasm
120 */
121 ACPI_STATUS
122 AdAmlDisassemble (
123 BOOLEAN OutToFile,
```

new/usr/src/common/acpica/include/acapps.h

```
124     char          *Filename,
125     char          *Prefix,
126     char          **OutFilename,
127     BOOLEAN       GetAllTables);

129 void
130 AdPrintStatistics (
131     void);

133 ACPI_STATUS
134 AdFindDsdtdt(
135     UINT8          **DsdtdPtr,
136     UINT32         DsdtdLength);

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,
```

3

new/usr/src/common/acpica/include/acapps.h

```
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 */
```

4

new/usr/src/common/acpica/include/acbuffer.h

1

```
*****
11132 Fri Jan 17 07:30:53 2014
new/usr/src/common/acpica/include/acbuffer.h
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Name: acbuffer.h - Support for buffers returned by ACPI predefined names
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __ACBUFFER_H__
45 #define __ACBUFFER_H__

47 /*
48 * Contains buffer structures for these predefined names:
49 * _FDE, _GRT, _GTM, _PLD, _SRT
50 */

52 /*
53 * Note: C bitfields are not used for this reason:
54 *
55 * "Bitfields are great and easy to read, but unfortunately the C language
56 * does not specify the layout of bitfields in memory, which means they are
57 * essentially useless for dealing with packed data in on-disk formats or
58 * binary wire protocols." (Or ACPI tables and buffers.) "If you ask me,
59 * this decision was a design error in C. Ritchie could have picked an order
60 * and stuck with it." Norman Ramsey.
```

new/usr/src/common/acpica/include/acbuffer.h

2

```
61 * See http://stackoverflow.com/a/1053662/41661
62 */

65 /* _FDE return value */

67 typedef struct acpi_fde_info
68 {
69     UINT32 Floppy0;
70     UINT32 Floppy1;
71     UINT32 Floppy2;
72     UINT32 Floppy3;
73     UINT32 Tape;
74 } ACPI_FDE_INFO;

75 } ACPI_FDE_INFO;

77 /*
78 * _GRT return value
79 * _SRT input value
80 */
81 typedef struct acpi_grt_info
82 {
83     UINT16 Year;
84     UINT8 Month;
85     UINT8 Day;
86     UINT8 Hour;
87     UINT8 Minute;
88     UINT8 Second;
89     UINT8 Valid;
90     UINT16 Milliseconds;
91     UINT16 Timezone;
92     UINT8 Daylight;
93     UINT8 Reserved[3];
94 } ACPI_GRT_INFO;

95 } ACPI_GRT_INFO;

97 /* _GTM return value */

99 typedef struct acpi_gtm_info
100 {
101     UINT32 PioSpeed0;
102     UINT32 DmaSpeed0;
103     UINT32 PioSpeed1;
104     UINT32 DmaSpeed1;
105     UINT32 Flags;
106 } ACPI_GTM_INFO;

107 } ACPI_GTM_INFO;

109 /*
110 * Formatted _PLD return value. The minimum size is a package containing
111 * one buffer.
112 * Revision 1: Buffer is 16 bytes (128 bits)
113 * Revision 2: Buffer is 20 bytes (160 bits)
114 *
115 * Note: This structure is returned from the AcpiDecodePldBuffer
116 * interface.
117 */
118 typedef struct acpi_pld_info
119 {
120     UINT8 Revision;
121     UINT8 IgnoreColor;
122     UINT32 Color;
123     UINT16 Width;
124     UINT16 Height;
125     UINT8 UserVisible;
126     UINT8 Dock;
```

```

127     UINT8      Lid;
128     UINT8      Panel;
129     UINT8      VerticalPosition;
130     UINT8      HorizontalPosition;
131     UINT8      Shape;
132     UINT8      GroupOrientation;
133     UINT8      GroupToken;
134     UINT8      GroupPosition;
135     UINT8      Bay;
136     UINT8      Ejectable;
137     UINT8      OspmEjectRequired;
138     UINT8      CabinetNumber;
139     UINT8      CardCageNumber;
140     UINT8      Reference;
141     UINT8      Rotation;
142     UINT8      Order;
143     UINT8      Reserved;
144     UINT16     VerticalOffset;
145     UINT16     HorizontalOffset;

147 } ACPI_PLD_INFO;

150 /*
151  * Macros to:
152  *   1) Convert a _PLD buffer to internal ACPI_PLD_INFO format - ACPI_PLD_GET*
153  *   (Used by AcpiDecodePldBuffer)
154  *   2) Construct a _PLD buffer - ACPI_PLD_SET*
155  *   (Intended for BIOS use only)
156  */
157 #define ACPI_PLD_REV1_BUFFER_SIZE      16 /* For Revision 1 of the buff
158 #define ACPI_PLD_BUFFER_SIZE           20 /* For Revision 2 of the buff

160 /* First 32-bit dword, bits 0:32 */

162 #define ACPI_PLD_GET_REVISION(dword)    ACPI_GET_BITS (dword, 0, ACPI_7B
163 #define ACPI_PLD_SET_REVISION(dword,value) ACPI_SET_BITS (dword, 0, ACPI_7B

165 #define ACPI_PLD_GET_IGNORE_COLOR(dword) ACPI_GET_BITS (dword, 7, ACPI_1B
166 #define ACPI_PLD_SET_IGNORE_COLOR(dword,value) ACPI_SET_BITS (dword, 7, ACPI_1B

168 #define ACPI_PLD_GET_COLOR(dword)       ACPI_GET_BITS (dword, 8, ACPI_24
169 #define ACPI_PLD_SET_COLOR(dword,value) ACPI_SET_BITS (dword, 8, ACPI_24

171 /* Second 32-bit dword, bits 33:63 */

173 #define ACPI_PLD_GET_WIDTH(dword)       ACPI_GET_BITS (dword, 0, ACPI_16
174 #define ACPI_PLD_SET_WIDTH(dword,value) ACPI_SET_BITS (dword, 0, ACPI_16

176 #define ACPI_PLD_GET_HEIGHT(dword)      ACPI_GET_BITS (dword, 16, ACPI_1
177 #define ACPI_PLD_SET_HEIGHT(dword,value) ACPI_SET_BITS (dword, 16, ACPI_1

179 /* Third 32-bit dword, bits 64:95 */

181 #define ACPI_PLD_GET_USER_VISIBLE(dword) ACPI_GET_BITS (dword, 0, ACPI_1B
182 #define ACPI_PLD_SET_USER_VISIBLE(dword,value) ACPI_SET_BITS (dword, 0, ACPI_1B

184 #define ACPI_PLD_GET_DOCK(dword)        ACPI_GET_BITS (dword, 1, ACPI_1B
185 #define ACPI_PLD_SET_DOCK(dword,value)  ACPI_SET_BITS (dword, 1, ACPI_1B

187 #define ACPI_PLD_GET_LID(dword)         ACPI_GET_BITS (dword, 2, ACPI_1B
188 #define ACPI_PLD_SET_LID(dword,value)   ACPI_SET_BITS (dword, 2, ACPI_1B

190 #define ACPI_PLD_GET_PANEL(dword)       ACPI_GET_BITS (dword, 3, ACPI_3B
191 #define ACPI_PLD_SET_PANEL(dword,value) ACPI_SET_BITS (dword, 3, ACPI_3B

```

```

193 #define ACPI_PLD_GET_VERTICAL(dword)    ACPI_GET_BITS (dword, 6, ACPI_2B
194 #define ACPI_PLD_SET_VERTICAL(dword,value) ACPI_SET_BITS (dword, 6, ACPI_2B

196 #define ACPI_PLD_GET_HORIZONTAL(dword)  ACPI_GET_BITS (dword, 8, ACPI_2B
197 #define ACPI_PLD_SET_HORIZONTAL(dword,value) ACPI_SET_BITS (dword, 8, ACPI_2B

199 #define ACPI_PLD_GET_SHAPE(dword)       ACPI_GET_BITS (dword, 10, ACPI_4
200 #define ACPI_PLD_SET_SHAPE(dword,value) ACPI_SET_BITS (dword, 10, ACPI_4

202 #define ACPI_PLD_GET_ORIENTATION(dword) ACPI_GET_BITS (dword, 14, ACPI_1
203 #define ACPI_PLD_SET_ORIENTATION(dword,value) ACPI_SET_BITS (dword, 14, ACPI_1

205 #define ACPI_PLD_GET_TOKEN(dword)       ACPI_GET_BITS (dword, 15, ACPI_8
206 #define ACPI_PLD_SET_TOKEN(dword,value) ACPI_SET_BITS (dword, 15, ACPI_8

208 #define ACPI_PLD_GET_POSITION(dword)    ACPI_GET_BITS (dword, 23, ACPI_8
209 #define ACPI_PLD_SET_POSITION(dword,value) ACPI_SET_BITS (dword, 23, ACPI_8

211 #define ACPI_PLD_GET_BAY(dword)         ACPI_GET_BITS (dword, 31, ACPI_1
212 #define ACPI_PLD_SET_BAY(dword,value)   ACPI_SET_BITS (dword, 31, ACPI_1

214 /* Fourth 32-bit dword, bits 96:127 */

216 #define ACPI_PLD_GET_EJECTABLE(dword)    ACPI_GET_BITS (dword, 0, ACPI_1B
217 #define ACPI_PLD_SET_EJECTABLE(dword,value) ACPI_SET_BITS (dword, 0, ACPI_1B

219 #define ACPI_PLD_GET_OSPM_EJECT(dword)  ACPI_GET_BITS (dword, 1, ACPI_1B
220 #define ACPI_PLD_SET_OSPM_EJECT(dword,value) ACPI_SET_BITS (dword, 1, ACPI_1B

222 #define ACPI_PLD_GET_CABINET(dword)     ACPI_GET_BITS (dword, 2, ACPI_8B
223 #define ACPI_PLD_SET_CABINET(dword,value) ACPI_SET_BITS (dword, 2, ACPI_8B

225 #define ACPI_PLD_GET_CARD_CAGE(dword)   ACPI_GET_BITS (dword, 10, ACPI_8
226 #define ACPI_PLD_SET_CARD_CAGE(dword,value) ACPI_SET_BITS (dword, 10, ACPI_8

228 #define ACPI_PLD_GET_REFERENCE(dword)    ACPI_GET_BITS (dword, 18, ACPI_1
229 #define ACPI_PLD_SET_REFERENCE(dword,value) ACPI_SET_BITS (dword, 18, ACPI_1

231 #define ACPI_PLD_GET_ROTATION(dword)     ACPI_GET_BITS (dword, 19, ACPI_4
232 #define ACPI_PLD_SET_ROTATION(dword,value) ACPI_SET_BITS (dword, 19, ACPI_4

234 #define ACPI_PLD_GET_ORDER(dword)       ACPI_GET_BITS (dword, 23, ACPI_5
235 #define ACPI_PLD_SET_ORDER(dword,value)  ACPI_SET_BITS (dword, 23, ACPI_5

237 /* Fifth 32-bit dword, bits 128:159 (Revision 2 of _PLD only) */

239 #define ACPI_PLD_GET_VERT_OFFSET(dword)  ACPI_GET_BITS (dword, 0, ACPI_16
240 #define ACPI_PLD_SET_VERT_OFFSET(dword,value) ACPI_SET_BITS (dword, 0, ACPI_16

242 #define ACPI_PLD_GET_HORIZ_OFFSET(dword) ACPI_GET_BITS (dword, 16, ACPI_1
243 #define ACPI_PLD_SET_HORIZ_OFFSET(dword,value) ACPI_SET_BITS (dword, 16, ACPI_1

246 #endif /* ACBUFFER_H */

```

new/usr/src/common/acpica/include/accommon.h

1

```
*****
2898 Fri Jan 17 07:30:53 2014
new/usr/src/common/acpica/include/accommon.h
update to acpica-unix2-20140114
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 - 2014, 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 MERCHANTABILITY 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 #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 */
```

new/usr/src/common/acpica/include/accommon.h

2

```
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__ */
```

new/usr/src/common/acpica/include/acconfig.h

1

```
*****
8300 Fri Jan 17 07:30:53 2014
new/usr/src/common/acpica/include/acconfig.h
update to acpica-unix2-20140114
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acconfig.h - Global configuration constants
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef _ACCONFIG_H
45 #define _ACCONFIG_H

48 /*****
49 *
50 * Configuration options
51 *
52 *****/

54 /*
55 * ACPI_DEBUG_OUTPUT - This switch enables all the debug facilities of the
56 * ACPI subsystem. This includes the DEBUG_PRINT output
```

new/usr/src/common/acpica/include/acconfig.h

2

```
57 * statements. When disabled, all DEBUG_PRINT
58 * statements are compiled out.
59 *
60 * ACPI_APPLICATION - Use this switch if the subsystem is going to be run
61 * at the application level.
62 *
63 */

65 /*
66 * OS name, used for the _OS object. The _OS object is essentially obsolete,
67 * but there is a large base of ASL/AML code in existing machines that check
68 * for the string below. The use of this string usually guarantees that
69 * the ASL will execute down the most tested code path. Also, there is some
70 * code that will not execute the _OSI method unless _OS matches the string
71 * below. Therefore, change this string at your own risk.
72 */
73 #define ACPI_OS_NAME "Microsoft Windows NT"

75 /* Maximum objects in the various object caches */

77 #define ACPI_MAX_STATE_CACHE_DEPTH 96 /* State objects */
78 #define ACPI_MAX_PARSE_CACHE_DEPTH 96 /* Parse tree objects */
79 #define ACPI_MAX_EXTPARSE_CACHE_DEPTH 96 /* Parse tree objects */
80 #define ACPI_MAX_OBJECT_CACHE_DEPTH 96 /* Interpreter operand objects */
81 #define ACPI_MAX_NAMESPACE_CACHE_DEPTH 96 /* Namespace objects */

83 /*
84 * Should the subsystem abort the loading of an ACPI table if the
85 * table checksum is incorrect?
86 */
87 #ifndef ACPI_CHECKSUM_ABORT
88 #define ACPI_CHECKSUM_ABORT FALSE
89 #endif

91 /*
92 * Generate a version of ACPICA that only supports "reduced hardware"
93 * platforms (as defined in ACPI 5.0). Set to TRUE to generate a specialized
94 * version of ACPICA that ONLY supports the ACPI 5.0 "reduced hardware"
95 * model. In other words, no ACPI hardware is supported.
96 *
97 * If TRUE, this means no support for the following:
98 * PM Event and Control registers
99 * SCI interrupt (and handler)
100 * Fixed Events
101 * General Purpose Events (GPEs)
102 * Global Lock
103 * ACPI PM timer
104 * FACS table (Waking vectors and Global Lock)
105 */
106 #ifndef ACPI_REduced_HARDWARE
107 #define ACPI_REduced_HARDWARE FALSE
108 #endif

111 /*****
112 *
113 * Subsystem Constants
114 *
115 *****/

117 /* Version of ACPI supported */

119 #define ACPI_CA_SUPPORT_LEVEL 5
120 #define ACPI_CA_SUPPORT_LEVEL 3

121 /* Maximum count for a semaphore object */
```

new/usr/src/common/acpica/include/acconfig.h

3

```
123 #define ACPI_MAX_SEMAPHORE_COUNT      256
125 /* Maximum object reference count (detects object deletion issues) */
127 #define ACPI_MAX_REFERENCE_COUNT      0x800
129 /* Default page size for use in mapping memory for operation regions */
131 #define ACPI_DEFAULT_PAGE_SIZE      4096 /* Must be power of 2 */
133 /* OwnerId tracking. 8 entries allows for 255 OwnerIds */
135 #define ACPI_NUM_OWNERID_MASKS      8
137 /* Size of the root table array is increased by this increment */
139 #define ACPI_ROOT_TABLE_SIZE_INCREMENT 4
141 /* Maximum number of While() loop iterations before forced abort */
143 #define ACPI_MAX_LOOP_ITERATIONS      0xFFFF
145 /* Maximum sleep allowed via Sleep() operator */
147 #define ACPI_MAX_SLEEP      2000 /* 2000 millisec == two seconds
126 #define ACPI_MAX_SLEEP      20000 /* Two seconds */
149 /* Address Range lists are per-SpaceId (Memory and I/O only) */
151 #define ACPI_ADDRESS_RANGE_MAX      2
154 /*****
155  *
156  * ACPI Specification constants (Do not change unless the specification changes)
157  *
158  *****/
160 /* Method info (in WALK_STATE), containing local variables and argumetns */
162 #define ACPI_METHOD_NUM_LOCALS      8
163 #define ACPI_METHOD_MAX_LOCAL      7
165 #define ACPI_METHOD_NUM_ARGS      7
166 #define ACPI_METHOD_MAX_ARG      6
168 /*
169  * Operand Stack (in WALK_STATE), Must be large enough to contain METHOD_MAX_ARG
170  */
171 #define ACPI_OBJ_NUM_OPERANDS      8
172 #define ACPI_OBJ_MAX_OPERAND      7
174 /* Number of elements in the Result Stack frame, can be an arbitrary value */
176 #define ACPI_RESULTS_FRAME_OBJ_NUM      8
178 /*
179  * Maximal number of elements the Result Stack can contain,
180  * it may be an arbitray value not exceeding the types of
181  * ResultSize and ResultCount (now UINT8).
182  */
183 #define ACPI_RESULTS_OBJ_NUM_MAX      255
185 /* Constants used in searching for the RSDP in low memory */
```

new/usr/src/common/acpica/include/acconfig.h

4

```
187 #define ACPI_EBDA_PTR_LOCATION      0x0000040E /* Physical Address */
188 #define ACPI_EBDA_PTR_LENGTH      2
189 #define ACPI_EBDA_WINDOW_SIZE      1024
190 #define ACPI_HI_RSDP_WINDOW_BASE      0x000E0000 /* Physical Address */
191 #define ACPI_HI_RSDP_WINDOW_SIZE      0x00020000
192 #define ACPI_RSDP_SCAN_STEP      16
194 /* Operation regions */
196 #define ACPI_USER_REGION_BEGIN      0x80
198 /* Maximum SpaceIds for Operation Regions */
200 #define ACPI_MAX_ADDRESS_SPACE      255
201 #define ACPI_NUM_DEFAULT_SPACES      4
203 /* Array sizes. Used for range checking also */
205 #define ACPI_MAX_MATCH_OPCODE      5
207 /* RSDP checksums */
209 #define ACPI_RSDP_CHECKSUM_LENGTH      20
210 #define ACPI_RSDP_XCHECKSUM_LENGTH      36
212 /* SMBus, GSBUS and IPMI bidirectional buffer size */
186 /* SMBus and IPMI bidirectional buffer size */
214 #define ACPI_SMBUS_BUFFER_SIZE      34
215 #define ACPI_GSBUS_BUFFER_SIZE      34
216 #define ACPI_IPMI_BUFFER_SIZE      66
218 /* _SxD and _SxW control methods */
220 #define ACPI_NUM_SxD_METHODS      4
221 #define ACPI_NUM_SxW_METHODS      5
224 /*****
225  *
226  * ACPI AML Debugger
227  *
228  *****/
230 #define ACPI_DEBUGGER_MAX_ARGS      ACPI_METHOD_NUM_ARGS + 4 /* Max command
203 #define ACPI_DEBUGGER_MAX_ARGS      ACPI_METHOD_NUM_ARGS + 2 /* Max command
231 #define ACPI_DB_LINE_BUFFER_SIZE      512
233 #define ACPI_DEBUGGER_COMMAND_PROMPT      '-'
234 #define ACPI_DEBUGGER_EXECUTE_PROMPT      '%'
237 #endif /* _ACONFIG_H */
```



new/usr/src/common/acpica/include/acdebug.h

1

```
*****
10239 Fri Jan 17 07:30:54 2014
new/usr/src/common/acpica/include/acdebug.h
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Name: acdebug.h - ACPI/AML debugger
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACDEBUG_H__
45 #define __ACDEBUG_H__
46
47 #define ACPI_DEBUG_BUFFER_SIZE 0x4000 /* 16K buffer for return objects */
48
49 typedef struct acpi_db_command_info
50 typedef struct CommandInfo
51 {
52     char *Name; /* Command Name */
53     UINT8 MinArgs; /* Minimum arguments required */
54 } ACPI_DB_COMMAND_INFO;
55 } COMMAND_INFO;
```

new/usr/src/common/acpica/include/acdebug.h

2

```
57 typedef struct acpi_db_command_help
57 typedef struct ArgumentInfo
58 {
59     UINT8 LineCount; /* Number of help lines */
60     char *Invocation; /* Command Invocation */
61     char *Description; /* Command Description */
62 } ACPI_DB_COMMAND_HELP;
63
64 typedef struct acpi_db_argument_info
65 {
66     char *Name; /* Argument Name */
67 } ACPI_DB_ARGUMENT_INFO;
68 } ARGUMENT_INFO;
69
70 typedef struct acpi_db_execute_walk
71 typedef struct acpi_execute_walk
72 {
73     UINT32 Count;
74     UINT32 MaxCount;
75 } ACPI_DB_EXECUTE_WALK;
76 } ACPI_EXECUTE_WALK;
77
78 #define PARAM_LIST(pl) pl
79 #define DBTEST_OUTPUT_LEVEL(lvl) if (AcpiGbl_DbOpt_verbose)
80 #define VERBOSE_PRINT(fp) DBTEST_OUTPUT_LEVEL(lvl) {\
81     AcpiOsPrintf PARAM_LIST(fp);}
82
83 #define EX_NO_SINGLE_STEP 1
84 #define EX_SINGLE_STEP 2
85
86 /*
87 * dbxface - external debugger interfaces
88 */
89
90 ACPI_STATUS
91 AcpiDbInitialize (
92     void);
93
94 void
95 AcpiDbTerminate (
96     void);
97
98 ACPI_STATUS
99 AcpiDbSingleStep (
100     ACPI_WALK_STATE *WalkState,
101     ACPI_PARSE_OBJECT *Op,
102     UINT32 OpType);
103
104 /*
105 * dbcmds - debug commands and output routines
106 */
107
108 ACPI_NAMESPACE_NODE *
109 AcpiDbConvertToNode (
110     char *InString);
111
112 void
113 AcpiDbDisplayTableInfo (
114     char *TableArg);
115
116 void
```

```

118 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 void
168 AcpiDbExecuteTest (
169     char                *TypeArg);

172 /*
173  * dbconvert - miscellaneous conversion routines
174  */
175 ACPI_STATUS
176 AcpiDbHexCharToValue (
177     int                 HexChar,
178     UINT8               *ReturnValue);

```

```

180 ACPI_STATUS
181 AcpiDbConvertToPackage (
182     char                *String,
183     ACPI_OBJECT         *Object);

185 ACPI_STATUS
186 AcpiDbConvertToObject (
187     ACPI_OBJECT_TYPE    Type,
188     char                *String,
189     ACPI_OBJECT         *Object);

191 UINT8 *
192 AcpiDbEncodePldBuffer (
193     ACPI_PLD_INFO       *PldInfo);

195 void
196 AcpiDbDumpPldBuffer (
197     ACPI_OBJECT         *ObjDesc);

200 /*
201  * dbmethod - control method commands
202  */
203 void
204 AcpiDbSetMethodBreakpoint (
205     char                *Location,
206     ACPI_WALK_STATE     *WalkState,
207     ACPI_PARSE_OBJECT   *Op);

209 void
210 AcpiDbSetMethodCallBreakpoint (
211     ACPI_PARSE_OBJECT   *Op);

213 void
214 AcpiDbSetMethodData (
215     char                *TypeArg,
216     char                *IndexArg,
217     char                *ValueArg);

219 ACPI_STATUS
220 AcpiDbDisassembleMethod (
221     char                *Name);

223 void
224 AcpiDbDisassembleAml (
225     char                *Statements,
226     ACPI_PARSE_OBJECT   *Op);

228 void
229 AcpiDbBatchExecute (
230     char                *CountArg);

233 /*
234  * dbnames - namespace commands
235  */
236 void
237 AcpiDbSetScope (
238     char                *Name);

240 void
241 AcpiDbDumpNamespace (
242     char                *StartArg,
243     char                *DepthArg);

245 void

```

```

246 AcpiDbDumpNamespacePaths (
247     void);

249 void
250 AcpiDbDumpNamespaceByOwner (
251     char *OwnerArg,
252     char *DepthArg);

254 ACPI_STATUS
255 AcpiDbFindNameInNamespace (
256     char *NameArg);

258 void
259 AcpiDbCheckPredefinedNames (
260     void);

262 ACPI_STATUS
263 AcpiDbDisplayObjects (
264     char *ObjTypeArg,
265     char *DisplayCountArg);

267 void
268 AcpiDbCheckIntegrity (
269     void);

271 void
272 AcpiDbFindReferences (
273     char *ObjectArg);

275 void
276 AcpiDbGetBusInfo (
277     void);

280 /*
281  * dbdisply - debug display commands
282  */
283 void
284 AcpiDbDisplayMethodInfo (
285     ACPI_PARSE_OBJECT *Op);

287 void
288 AcpiDbDecodeAndDisplayObject (
289     char *Target,
290     char *OutputType);

292 void
293 AcpiDbDisplayResultObject (
294     ACPI_OPERAND_OBJECT *ObjDesc,
295     ACPI_WALK_STATE *WalkState);

297 ACPI_STATUS
298 AcpiDbDisplayAllMethods (
299     char *DisplayCountArg);

301 void
302 AcpiDbDisplayArguments (
303     void);

305 void
306 AcpiDbDisplayLocals (
307     void);

309 void
310 AcpiDbDisplayResults (
311     void);

```

```

313 void
314 AcpiDbDisplayCallingTree (
315     void);

317 void
318 AcpiDbDisplayObjectType (
319     char *ObjectArg);

321 void
322 AcpiDbDisplayArgumentObject (
323     ACPI_OPERAND_OBJECT *ObjDesc,
324     ACPI_WALK_STATE *WalkState);

327 /*
328  * dbexec - debugger control method execution
329  */
330 void
331 AcpiDbExecute (
332     char *Name,
333     char **Args,
334     ACPI_OBJECT_TYPE *Types,
335     UINT32 Flags);

337 void
338 AcpiDbCreateExecutionThreads (
339     char *NumThreadsArg,
340     char *NumLoopsArg,
341     char *MethodNameArg);

343 void
344 AcpiDbDeleteObjects (
345     UINT32 Count,
346     ACPI_OBJECT *Objects);

348 #ifdef ACPI_DBG_TRACK_ALLOCATIONS
349 UINT32
350 AcpiDbGetCacheInfo (
351     ACPI_MEMORY_LIST *Cache);
352 #endif

355 /*
356  * dbfileio - Debugger file I/O commands
357  */
358 ACPI_OBJECT_TYPE
359 AcpiDbMatchArgument (
360     char *UserArgument,
361     ACPI_DB_ARGUMENT_INFO *Arguments);
362 ARGUMENT_INFO *Arguments);

363 void
364 AcpiDbCloseDebugFile (
365     void);

367 void
368 AcpiDbOpenDebugFile (
369     char *Name);

371 ACPI_STATUS
372 AcpiDbLoadAcpiTable (
373     char *Filename);

375 ACPI_STATUS
376 AcpiDbGetTableFromFile (

```

new/usr/src/common/acpica/include/acdebug.h

```
377     char          *Filename,
378     ACPI_TABLE_HEADER **Table);

380 ACPI_STATUS
381 AcpiDbReadTableFromFile (
382     char          *Filename,
383     ACPI_TABLE_HEADER **Table);

386 /*
387  * dbhistory - debugger HISTORY command
388  */
389 void
390 AcpiDbAddToHistory (
391     char          *CommandLine);

393 void
394 AcpiDbDisplayHistory (
395     void);

397 char *
398 AcpiDbGetFromHistory (
399     char          *CommandNumArg);

401 char *
402 AcpiDbGetHistoryByIndex (
403     UINT32        CommanddNum);

406 /*
407  * dbinput - user front-end to the AML debugger
408  */
409 ACPI_STATUS
410 AcpiDbCommandDispatch (
411     char          *InputBuffer,
412     ACPI_WALK_STATE *WalkState,
413     ACPI_PARSE_OBJECT *Op);

415 void ACPI_SYSTEM_XFACE
416 AcpiDbExecuteThread (
417     void          *Context);

419 ACPI_STATUS
420 AcpiDbUserCommands (
421     char          Prompt,
422     ACPI_PARSE_OBJECT *Op);

424 char *
425 AcpiDbGetNextToken (
426     char          *String,
427     char          **Next,
428     ACPI_OBJECT_TYPE *ReturnType);

431 /*
432  * dbstats - Generation and display of ACPI table statistics
433  */
434 void
435 AcpiDbGenerateStatistics (
436     ACPI_PARSE_OBJECT *Root,
437     BOOLEAN            IsMethod);

439 ACPI_STATUS
440 AcpiDbDisplayStatistics (
441     char          *TypeArg);
```

7

new/usr/src/common/acpica/include/acdebug.h

```
444 /*
445  * dbutils - AML debugger utilities
446  */
447 void
448 AcpiDbSetOutputDestination (
449     UINT32        Where);

451 void
452 AcpiDbDumpExternalObject (
453     ACPI_OBJECT   *ObjDesc,
454     UINT32        Level);

456 void
457 AcpiDbPrepNamestring (
458     char          *Name);

460 ACPI_NAMESPACE_NODE *
461 AcpiDbLocalNsLookup (
462     char          *Name);

464 void
465 AcpiDbUInt32ToHexString (
466     AcpiDbUInt32ToHexString (
467         UINT32        Value,
468         char          *Buffer);

469 #endif /* __ACDEBUG_H__ */
```

8

new/usr/src/common/acpica/include/acdisasm.h

1

```
*****
24447 Fri Jan 17 07:30:54 2014
new/usr/src/common/acpica/include/acdisasm.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Name: acdisasm.h - AML disassembler
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #ifndef __ACDISASM_H__
45 #define __ACDISASM_H__
46
47 #include "amlresrc.h"
48
49 #define BLOCK_NONE 0
50 #define BLOCK_PAREN 1
51 #define BLOCK_BRACE 2
52 #define BLOCK_COMMA_LIST 4
53 #define ACPI_DEFAULT_RESNAME *(UINT32 *) "__RD"
54
55 /*
56 * Raw table data header. Used by disassembler and data table compiler.
57 * Do not change.
58 */
```

new/usr/src/common/acpica/include/acdisasm.h

2

```
59 */
60 #define ACPI_RAW_TABLE_DATA_HEADER "Raw Table Data"
61
62
63 typedef const struct acpi_dmtable_info
64 {
65     UINT8 Opcode;
66     UINT16 Offset;
67     char *Name;
68     UINT8 Flags;
69 } ACPI_DMTABLE_INFO;
70
71 /* Values for Flags field above */
72
73 #define DT_LENGTH 0x01 /* Field is a subtable length */
74 #define DT_FLAG 0x02 /* Field is a flag value */
75 #define DT_NON_ZERO 0x04 /* Field must be non-zero */
76 #define DT_OPTIONAL 0x08 /* Field is optional */
77 #define DT_DESCRIBES_OPTIONAL 0x10 /* Field describes an optional f
78 #define DT_COUNT 0x20 /* Currently not used */
79
80
81 /*
82 * Values for Opcode above.
83 * Note: 0-7 must not change, they are used as a flag shift value. Other
84 * than those, new values can be added wherever appropriate.
85 */
86 typedef enum
87 {
88     /* Simple Data Types */
89
90     ACPI_DMT_FLAG0 = 0,
91     ACPI_DMT_FLAG1 = 1,
92     ACPI_DMT_FLAG2 = 2,
93     ACPI_DMT_FLAG3 = 3,
94     ACPI_DMT_FLAG4 = 4,
95     ACPI_DMT_FLAG5 = 5,
96     ACPI_DMT_FLAG6 = 6,
97     ACPI_DMT_FLAG7 = 7,
98     ACPI_DMT_FLAGS0,
99     ACPI_DMT_FLAGS1,
100     ACPI_DMT_FLAGS2,
101     ACPI_DMT_FLAGS4,
102     ACPI_DMT_UINT8,
103     ACPI_DMT_UINT16,
104     ACPI_DMT_UINT24,
105     ACPI_DMT_UINT32,
106     ACPI_DMT_UINT40,
107     ACPI_DMT_UINT48,
108     ACPI_DMT_UINT56,
109     ACPI_DMT_UINT64,
110     ACPI_DMT_BUF7,
111     ACPI_DMT_BUF10,
112     ACPI_DMT_BUF16,
113     ACPI_DMT_BUF128,
114     ACPI_DMT_SIG,
115     ACPI_DMT_STRING,
116     ACPI_DMT_NAME4,
117     ACPI_DMT_NAME6,
118     ACPI_DMT_NAME8,
119
120     /* Types that are decoded to strings and miscellaneous */
121
122     ACPI_DMT_ACCWIDTH,
123     ACPI_DMT_CHKSUM,
124     ACPI_DMT_GAS,
```

```

125     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 AcpiDmTableInfoDbgpp[];
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      AcpiDmTableInfoEcddt[];
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      AcpiDmTableInfoSliit[];
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      AcpiDmTableInfoTcpc[];
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  * dmresrc1
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 AddDisassemblerHeader (
929     char *Filename);

932 #endif /* __ACDISASM_H__ */
```

new/usr/src/common/acpica/include/acdispat.h

1

```
*****
11413 Fri Jan 17 07:30:54 2014
new/usr/src/common/acpica/include/acdispat.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acdispat.h - dispatcher (parser to interpreter interface)
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef _ACDISPAT_H_
46 #define _ACDISPAT_H_

49 #define NAMEOF_LOCAL_NTE    "_L0"
50 #define NAMEOF_ARG_NTE     "_A0"

53 /*
54 * dsargs - execution of dynamic arguments for static objects
55 */
56 ACPI_STATUS
57 AcpiDsGetBufferFieldArguments (
58     ACPI_OPERAND_OBJECT *ObjDesc);
```

new/usr/src/common/acpica/include/acdispat.h

2

```
60 ACPI_STATUS
61 AcpiDsGetBankFieldArguments (
62     ACPI_OPERAND_OBJECT *ObjDesc);

64 ACPI_STATUS
65 AcpiDsGetRegionArguments (
66     ACPI_OPERAND_OBJECT *RgnDesc);

68 ACPI_STATUS
69 AcpiDsGetBufferArguments (
70     ACPI_OPERAND_OBJECT *ObjDesc);

72 ACPI_STATUS
73 AcpiDsGetPackageArguments (
74     ACPI_OPERAND_OBJECT *ObjDesc);

77 /*
78 * dscontrol - support for execution control opcodes
79 */
80 ACPI_STATUS
81 AcpiDsExecBeginControlOp (
82     ACPI_WALK_STATE *WalkState,
83     ACPI_PARSE_OBJECT *Op);

85 ACPI_STATUS
86 AcpiDsExecEndControlOp (
87     ACPI_WALK_STATE *WalkState,
88     ACPI_PARSE_OBJECT *Op);

91 /*
92 * dsopcode - support for late operand evaluation
93 */
94 ACPI_STATUS
95 AcpiDsEvalBufferFieldOperands (
96     ACPI_WALK_STATE *WalkState,
97     ACPI_PARSE_OBJECT *Op);

99 ACPI_STATUS
100 AcpiDsEvalRegionOperands (
101     ACPI_WALK_STATE *WalkState,
102     ACPI_PARSE_OBJECT *Op);

104 ACPI_STATUS
105 AcpiDsEvalTableRegionOperands (
106     ACPI_WALK_STATE *WalkState,
107     ACPI_PARSE_OBJECT *Op);

109 ACPI_STATUS
110 AcpiDsEvalDataObjectOperands (
111     ACPI_WALK_STATE *WalkState,
112     ACPI_PARSE_OBJECT *Op,
113     ACPI_OPERAND_OBJECT *ObjDesc);

115 ACPI_STATUS
116 AcpiDsEvalBankFieldOperands (
117     ACPI_WALK_STATE *WalkState,
118     ACPI_PARSE_OBJECT *Op);

120 ACPI_STATUS
121 AcpiDsInitializeRegion (
122     ACPI_HANDLE ObjHandle);
```

```

125 /*
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 /*

```

new/usr/src/common/acpica/include/acdispat.h

5

```
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);
```

new/usr/src/common/acpica/include/acdispat.h

6

```
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
```

new/usr/src/common/acpica/include/acdispat.h

```
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 (
```

7

new/usr/src/common/acpica/include/acdispat.h

```
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_ */
```

8



new/usr/src/common/acpica/include/acevents.h

1

```
*****
8717 Fri Jan 17 07:30:55 2014
new/usr/src/common/acpica/include/acevents.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Name: acevents.h - Event subcomponent prototypes and defines
4 *
5 *****/
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __ACEVENTS_H__
45 #define __ACEVENTS_H__
47 /*
48 * evevent
49 */
50 /*
51 ACPI_STATUS
52 AcpiEvInitializeEvents (
53     void);
55 ACPI_STATUS
56 AcpiEvInstallXruptHandlers (
57     void);
```

new/usr/src/common/acpica/include/acevents.h

2

```
59 UINT32
60 AcpiEvFixedEventDetect (
61     void);
63 /*
64 * evmisc
65 */
66 BOOLEAN
67 AcpiEvIsNotifyObject (
68     ACPI_NAMESPACE_NODE *Node);
70 UINT32
71 AcpiEvGetGpeNumberIndex (
72     UINT32 GpeNumber);
74 ACPI_STATUS
75 AcpiEvQueueNotifyRequest (
76     ACPI_NAMESPACE_NODE *Node,
77     UINT32 NotifyValue);
79 /*
80 * evglock - Global Lock support
81 */
82 ACPI_STATUS
83 AcpiEvInitGlobalLockHandler (
84     void);
86 ACPI_HW_DEPENDENT_RETURN_OK (
87     ACPI_STATUS
88     AcpiEvAcquireGlobalLock(
89         UINT16 Timeout));
90 ACPI_STATUS
91 AcpiEvReleaseGlobalLock(
92     void);
94 ACPI_STATUS
95 AcpiEvRemoveGlobalLockHandler (
96     void);
98 /*
99 * evgpe - Low-level GPE support
100 */
101 ACPI_STATUS
102 AcpiEvGpeDetect (
103     ACPI_GPE_XRUPT_INFO *GpeXruptList);
105 ACPI_STATUS
106 AcpiEvUpdateGpeEnableMask (
107     ACPI_GPE_EVENT_INFO *GpeEventInfo);
109 ACPI_STATUS
110 AcpiEvEnableGpe (
111     ACPI_GPE_EVENT_INFO *GpeEventInfo);
113 ACPI_STATUS
114 AcpiEvAddGpeReference (
115     ACPI_GPE_EVENT_INFO *GpeEventInfo);
```

new/usr/src/common/acpica/include/acevents.h

```
122 ACPI_STATUS
123 AcpiEvRemoveGpeReference (
124     ACPI_GPE_EVENT_INFO    *GpeEventInfo);

126 ACPI_GPE_EVENT_INFO *
127 AcpiEvGetGpeEventInfo (
128     ACPI_HANDLE             GpeDevice,
129     UINT32                   GpeNumber);

131 ACPI_GPE_EVENT_INFO *
132 AcpiEvLowGetGpeInfo (
133     UINT32                   GpeNumber,
134     ACPI_GPE_BLOCK_INFO     *GpeBlock);

136 ACPI_STATUS
137 AcpiEvFinishGpe (
138     ACPI_GPE_EVENT_INFO     *GpeEventInfo);

141 /*
142  * evgpeblk - Upper-level GPE block support
143  */
144 ACPI_STATUS
145 AcpiEvCreateGpeBlock (
146     ACPI_NAMESPACE_NODE     *GpeDevice,
147     ACPI_GENERIC_ADDRESS     *GpeBlockAddress,
148     UINT32                   RegisterCount,
149     UINT8                     GpeBlockBaseNumber,
150     UINT32                   InterruptNumber,
151     ACPI_GPE_BLOCK_INFO     **ReturnGpeBlock);

153 ACPI_STATUS
154 AcpiEvInitializeGpeBlock (
155     ACPI_GPE_XRPT_INFO       *GpeXruptInfo,
156     ACPI_GPE_BLOCK_INFO       *GpeBlock,
157     void                      *Context);

159 ACPI_HW_DEPENDENT_RETURN_OK (
160 ACPI_STATUS
161 AcpiEvDeleteGpeBlock (
162     ACPI_GPE_BLOCK_INFO     *GpeBlock))
159 ACPI_GPE_BLOCK_INFO     *GpeBlock);

164 UINT32
165 AcpiEvGpeDispatch (
166     ACPI_NAMESPACE_NODE     *GpeDevice,
167     ACPI_GPE_EVENT_INFO     *GpeEventInfo,
168     UINT32                   GpeNumber);

171 /*
172  * evgpeinit - GPE initialization and update
173  */
174 ACPI_STATUS
175 AcpiEvGpeInitialize (
176     void);

178 ACPI_HW_DEPENDENT_RETURN_VOID (
179 void
180 AcpiEvUpdateGpes (
181     ACPI_OWNER_ID           TableOwnerId))
176 ACPI_OWNER_ID           TableOwnerId);

183 ACPI_STATUS
184 AcpiEvMatchGpeMethod (
185     ACPI_HANDLE             ObjHandle,
```

3

new/usr/src/common/acpica/include/acevents.h

```
186     UINT32                   Level,
187     void                      *Context,
188     void                      **ReturnValue);

191 /*
192  * evgpeutil - GPE utilities
193  */
194 ACPI_STATUS
195 AcpiEvWalkGpeList (
196     ACPI_GPE_CALLBACK        GpeWalkCallback,
197     void                      *Context);

199 BOOLEAN
200 AcpiEvValidGpeEvent (
201     ACPI_GPE_EVENT_INFO     *GpeEventInfo);

203 ACPI_STATUS
204 AcpiEvGetGpeDevice (
205     ACPI_GPE_XRPT_INFO       *GpeXruptInfo,
206     ACPI_GPE_BLOCK_INFO       *GpeBlock,
207     void                      *Context);

209 ACPI_STATUS
203 ACPI_GPE_XRPT_INFO *
210 AcpiEvGetGpeXruptBlock (
211     UINT32                   InterruptNumber,
212     ACPI_GPE_XRPT_INFO       **GpeXruptBlock);
205     UINT32                   InterruptNumber);

214 ACPI_STATUS
215 AcpiEvDeleteGpeXrupt (
216     ACPI_GPE_XRPT_INFO       *GpeXrupt);

218 ACPI_STATUS
219 AcpiEvDeleteGpeHandlers (
220     ACPI_GPE_XRPT_INFO       *GpeXruptInfo,
221     ACPI_GPE_BLOCK_INFO       *GpeBlock,
222     void                      *Context);

225 /*
226  * evhandler - Address space handling
227  * evregion - Address Space handling
228  */
229 BOOLEAN
229 AcpiEvHasDefaultHandler (
230     ACPI_NAMESPACE_NODE     *Node,
231     ACPI_ADR_SPACE_TYPE      SpaceId);

233 ACPI_STATUS
234 AcpiEvInstallRegionHandlers (
235     void);

237 ACPI_STATUS
238 AcpiEvInstallSpaceHandler (
239     ACPI_NAMESPACE_NODE     *Node,
240     ACPI_ADR_SPACE_TYPE      SpaceId,
241     ACPI_ADR_SPACE_HANDLER    Handler,
242     ACPI_ADR_SPACE_SETUP      Setup,
243     void                      *Context);

246 /*
247  * evregion - Operation region support
248  */
```

4

```

249 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
274 AcpiEvInstallSpaceHandler (
275     ACPI_NAMESPACE_NODE *Node,
276     ACPI_ADR_SPACE_TYPE SpaceId,
277     ACPI_ADR_SPACE_HANDLER Handler,
278     ACPI_ADR_SPACE_SETUP Setup,
279     void *Context);

281 ACPI_STATUS
282 AcpiEvExecuteRegMethods (
283     ACPI_NAMESPACE_NODE *Node,
284     ACPI_ADR_SPACE_TYPE SpaceId);

286 ACPI_STATUS
287 AcpiEvExecuteRegMethod (
288     ACPI_OPERAND_OBJECT *RegionObj,
289     UINT32 Function);

291 /*
292  * evregini - Region initialization and setup
293  */
294 ACPI_STATUS
295 AcpiEvSystemMemoryRegionSetup (
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 (
352     void);

354 ACPI_HW_DEPENDENT_RETURN_VOID (
355     UINT32 ProgramSCI);

356 void
357 AcpiEvTerminate (
358     void);

359 #endif /* __ACEVENTS_H__ */

```

new/usr/src/common/acpica/include/acexcep.h

1

```
*****
18604 Fri Jan 17 07:30:55 2014
new/usr/src/common/acpica/include/acexcep.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acexcep.h - Exception codes returned by the ACPI subsystem
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #ifndef __ACEXCEP_H__
45 #define __ACEXCEP_H__
46
47
48 /* This module contains all possible exception codes for ACPI_STATUS */
49
50 /*
51 * Exception code classes
52 */
53 #define AE_CODE_ENVIRONMENTAL 0x0000 /* General ACPICA environment */
54 #define AE_CODE_PROGRAMMER 0x1000 /* External ACPICA interface call
55 #define AE_CODE ACPI_TABLES 0x2000 /* ACPI tables */
56 #define AE_CODE AML 0x3000 /* From executing AML code */
57 #define AE_CODE CONTROL 0x4000 /* Internal control codes */
58
59 #define AE_CODE_MAX 0x4000
```

new/usr/src/common/acpica/include/acexcep.h

2

```
60 #define AE_CODE_MASK 0xF000
61
62 /*
63 * Macros to insert the exception code classes
64 */
65 #define EXCEP_ENV(code) ((ACPI_STATUS) (code | AE_CODE_ENVIRONME
66 #define EXCEP_PGM(code) ((ACPI_STATUS) (code | AE_CODE_PROGRAMME
67 #define EXCEP_TBL(code) ((ACPI_STATUS) (code | AE_CODE ACPI_TABL
68 #define EXCEP_AML(code) ((ACPI_STATUS) (code | AE_CODE AML))
69 #define EXCEP_CTL(code) ((ACPI_STATUS) (code | AE_CODE_CONTROL))
70
71 /*
72 * Exception info table. The "Description" field is used only by the
73 * ACPICA help application (acpihelp).
74 */
75 typedef struct acpi_exception_info
76 {
77     char *Name;
78
79     #ifdef ACPI_HELP_APP
80     char *Description;
81     #endif
82 } ACPI_EXCEPTION_INFO;
83
84 #ifdef ACPI_HELP_APP
85 #define EXCEP_TXT(Name,Description) {Name, Description}
86 #else
87 #define EXCEP_TXT(Name,Description) {Name}
88 #endif
89
90
91 /*
92 * Success is always zero, failure is non-zero
93 */
94 #define ACPI_SUCCESS(a) (!a)
95 #define ACPI_FAILURE(a) (a)
96
97 #define AE_OK (ACPI_STATUS) 0x0000
98
99 /*
100 * Environmental exceptions
101 */
102 #define AE_ERROR EXCEP_ENV (0x0001)
103 #define AE_NO ACPI_TABLES EXCEP_ENV (0x0002)
104 #define AE_NO_NAMESPACE EXCEP_ENV (0x0003)
105 #define AE_NO_MEMORY EXCEP_ENV (0x0004)
106 #define AE_NOT_FOUND EXCEP_ENV (0x0005)
107 #define AE_NOT_EXIST EXCEP_ENV (0x0006)
108 #define AE_ALREADY_EXISTS EXCEP_ENV (0x0007)
109 #define AE_TYPE EXCEP_ENV (0x0008)
110 #define AE_NULL_OBJECT EXCEP_ENV (0x0009)
111 #define AE_NULL_ENTRY EXCEP_ENV (0x000A)
112 #define AE_BUFFER_OVERFLOW EXCEP_ENV (0x000B)
113 #define AE_STACK_OVERFLOW EXCEP_ENV (0x000C)
114 #define AE_STACK_UNDERFLOW EXCEP_ENV (0x000D)
115 #define AE_NOT_IMPLEMENTED EXCEP_ENV (0x000E)
116 #define AE_SUPPORT EXCEP_ENV (0x000F)
117 #define AE_LIMIT EXCEP_ENV (0x0010)
118 #define AE_TIME EXCEP_ENV (0x0011)
119 #define AE_ACQUIRE_DEADLOCK EXCEP_ENV (0x0012)
120 #define AE_RELEASE_DEADLOCK EXCEP_ENV (0x0013)
121 #define AE_NOT_ACQUIRED EXCEP_ENV (0x0014)
122 #define AE_ALREADY_ACQUIRED EXCEP_ENV (0x0015)
123 #define AE_NO_HARDWARE_RESPONSE EXCEP_ENV (0x0016)
124 #define AE_NO_GLOBAL_LOCK EXCEP_ENV (0x0017)
125 #define AE_ABORT_METHOD EXCEP_ENV (0x0018)
```

```

126 #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 #ifndef 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 exist"),
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 exist"),
245     EXCEP_TXT ("AE_BUFFER_OVERFLOW", "The buffer provided is too small"),
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 made"),
253     EXCEP_TXT ("AE_RELEASE_DEADLOCK", "Internal error, attempt was made"),
254     EXCEP_TXT ("AE_NOT_ACQUIRED", "An attempt to release a mutex object"),
255     EXCEP_TXT ("AE_ALREADY_ACQUIRED", "Internal error, attempt was made"),
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 requ
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 */

```

new/usr/src/common/acpica/include/acglobal.h

1

```
*****
20926 Fri Jan 17 07:30:55 2014
new/usr/src/common/acpica/include/acglobal.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Name: acglobal.h - Declarations for global variables
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACGLOBAL_H__
45 #define __ACGLOBAL_H__
46
47
48 /*
49 * Ensure that the globals are actually defined and initialized only once.
50 *
51 * The use of these macros allows a single list of globals (here) in order
52 * to simplify maintenance of the code.
53 */
54 #ifdef DEFINE_ACPI_GLOBALS
55 #define ACPI_EXTERN
56 #define ACPI_INIT_GLOBAL(a,b) a=b
57 #else
```

new/usr/src/common/acpica/include/acglobal.h

2

```
58 #define ACPI_EXTERN extern
59 #define ACPI_INIT_GLOBAL(a,b) a
60 #endif
61
62
63 #ifdef DEFINE_ACPI_GLOBALS
64
65 /* Public globals, available from outside ACPICA subsystem */
66
67 /*****
68 *
69 * Runtime configuration (static defaults that can be overridden at runtime)
70 *
71 *****/
72
73 /*
74 * Enable "slack" in the AML interpreter? Default is FALSE, and the
75 * interpreter strictly follows the ACPI specification. Setting to TRUE
76 * allows the interpreter to ignore certain errors and/or bad AML constructs.
77 *
78 * Currently, these features are enabled by this flag:
79 *
80 * 1) Allow "implicit return" of last value in a control method
81 * 2) Allow access beyond the end of an operation region
82 * 3) Allow access to uninitialized locals/args (auto-init to integer 0)
83 * 4) Allow ANY object type to be a source operand for the Store() operator
84 * 5) Allow unresolved references (invalid target name) in package objects
85 * 6) Enable warning messages for behavior that is not ACPI spec compliant
86 */
87 UINT8 ACPI_INIT_GLOBAL (AcpiGbl_EnableInterpreterSlack, FALSE);
88
89 /*
90 * Automatically serialize ALL control methods? Default is FALSE, meaning
91 * to use the Serialized/NotSerialized method flags on a per method basis.
92 * Only change this if the ASL code is poorly written and cannot handle
93 * reentrancy even though methods are marked "NotSerialized".
94 */
95 UINT8 ACPI_INIT_GLOBAL (AcpiGbl_AllMethodsSerialized, FALSE);
96
97 /*
98 * Create the predefined _OSI method in the namespace? Default is TRUE
99 * because ACPI CA is fully compatible with other ACPI implementations.
100 * Changing this will revert ACPI CA (and machine ASL) to pre-OSI behavior.
101 */
102 UINT8 ACPI_INIT_GLOBAL (AcpiGbl_CreateOsiMethod, TRUE);
103
104 /*
105 * Optionally use default values for the ACPI register widths. Set this to
106 * TRUE to use the defaults, if an FADT contains incorrect widths/lengths.
107 */
108 UINT8 ACPI_INIT_GLOBAL (AcpiGbl_UseDefaultRegisterWidths, TRUE);
109
110 /*
111 * Optionally enable output from the AML Debug Object.
112 */
113 UINT8 ACPI_INIT_GLOBAL (AcpiGbl_EnableAmlDebugObject, FALSE);
114
115 /*
116 * Optionally copy the entire DSDT to local memory (instead of simply
117 * mapping it.) There are some BIOSs that corrupt or replace the original
118 * DSDT, creating the need for this option. Default is FALSE, do not copy
119 * the DSDT.
120 */
121 UINT8 ACPI_INIT_GLOBAL (AcpiGbl_CopyDsdtdLocally, FALSE);
122
123 */
```

```

124 * Optionally ignore an XSDT if present and use the RSDT instead.
125 * Although the ACPI specification requires that an XSDT be used instead
126 * of the RSDT, the XSDT has been found to be corrupt or ill-formed on
127 * some machines. Default behavior is to use the XSDT if present.
128 */
129 UINT8          ACPI_INIT_GLOBAL (AcpiGbl_DoNotUseXsdt, FALSE);

132 /*
133 * Optionally use 32-bit FADT addresses if and when there is a conflict
134 * (address mismatch) between the 32-bit and 64-bit versions of the
135 * address. Although ACPICA adheres to the ACPI specification which
136 * requires the use of the corresponding 64-bit address if it is non-zero,
137 * some machines have been found to have a corrupted non-zero 64-bit
138 * address. Default is FALSE, do not favor the 32-bit addresses.
139 */
140 UINT8          ACPI_INIT_GLOBAL (AcpiGbl_Use32BitFadtAddresses, FALSE);

142 /*
143 * Optionally truncate I/O addresses to 16 bits. Provides compatibility
144 * with other ACPI implementations. NOTE: During ACPICA initialization,
145 * this value is set to TRUE if any Windows OSI strings have been
146 * requested by the BIOS.
147 */
148 UINT8          ACPI_INIT_GLOBAL (AcpiGbl_TruncateIoAddresses, FALSE);

150 /*
151 * Disable runtime checking and repair of values returned by control methods.
152 * Use only if the repair is causing a problem on a particular machine.
153 */
154 UINT8          ACPI_INIT_GLOBAL (AcpiGbl_DisableAutoRepair, FALSE);

156 /*
157 * Optionally do not load any SSDTs from the RSDT/XSDT during initialization.
158 * This can be useful for debugging ACPI problems on some machines.
159 */
160 UINT8          ACPI_INIT_GLOBAL (AcpiGbl_DisableSsdtTableLoad, FALSE);

162 /*
163 * We keep track of the latest version of Windows that has been requested by
164 * the BIOS.
165 */
166 UINT8          ACPI_INIT_GLOBAL (AcpiGbl_OsiData, 0);

169 /* AcpiGbl_FADT is a local copy of the FADT, converted to a common format. */

171 ACPI_TABLE_FADT      AcpiGbl_FADT;
172 UINT32               AcpiCurrentGpeCount;
173 UINT32               AcpiGbl_TraceFlags;
174 ACPI_NAME             AcpiGbl_TraceMethodName;
175 BOOLEAN              AcpiGbl_SystemAwakeAndRunning;

177 /*
178 * ACPI 5.0 introduces the concept of a "reduced hardware platform", meaning
179 * that the ACPI hardware is no longer required. A flag in the FADT indicates
180 * a reduced HW machine, and that flag is duplicated here for convenience.
181 */
182 BOOLEAN              AcpiGbl_ReducedHardware;
183 #endif

184 #endif /* DEFINE_ACPI_GLOBALS */

186 /* Do not disassemble buffers to resource descriptors */

188 ACPI_EXTERN UINT8     ACPI_INIT_GLOBAL (AcpiGbl_NoResourceDisassembly, FALSE);

```

```

190 /*****
191 *
192 * ACPI Table globals
193 *
194 *****/

196 /*
197 * AcpiGbl_RootTableList is the master list of ACPI tables that were
198 * found in the RSDT/XSDT.
199 */
200 ACPI_EXTERN ACPI_TABLE_LIST      AcpiGbl_RootTableList;

202 #if (!ACPI_REDUCED_HARDWARE)
203 ACPI_EXTERN ACPI_TABLE_FACS      *AcpiGbl_FACS;

205 #endif /* !ACPI_REDUCED_HARDWARE */

207 /* These addresses are calculated from the FADT Event Block addresses */

209 ACPI_EXTERN ACPI_GENERIC_ADDRESS AcpiGbl_XPmlaStatus;
210 ACPI_EXTERN ACPI_GENERIC_ADDRESS AcpiGbl_XPmlaEnable;

212 ACPI_EXTERN ACPI_GENERIC_ADDRESS AcpiGbl_XPmlbStatus;
213 ACPI_EXTERN ACPI_GENERIC_ADDRESS AcpiGbl_XPmlbEnable;

215 /* DSDT information. Used to check for DSDT corruption */

217 ACPI_EXTERN ACPI_TABLE_HEADER    *AcpiGbl_DSDT;
218 ACPI_EXTERN ACPI_TABLE_HEADER    AcpiGbl_OriginalDsdHeader;

220 /*
221 * Handle both ACPI 1.0 and ACPI 2.0 Integer widths. The integer width is
222 * determined by the revision of the DSDT: If the DSDT revision is less than
223 * 2, use only the lower 32 bits of the internal 64-bit Integer.
224 */
225 ACPI_EXTERN UINT8               AcpiGbl_IntegerBitWidth;
226 ACPI_EXTERN UINT8               AcpiGbl_IntegerByteWidth;
227 ACPI_EXTERN UINT8               AcpiGbl_IntegerNybbleWidth;

230 /*****
231 *
232 * Mutual exclusion within ACPICA subsystem
233 * Mutual exclusion within ACPICA subsystem
234 *****/

236 /*
237 * Predefined mutex objects. This array contains the
238 * actual OS mutex handles, indexed by the local ACPI_MUTEX_HANDLES.
239 * (The table maps local handles to the real OS handles)
240 */
241 ACPI_EXTERN ACPI_MUTEX_INFO     AcpiGbl_MutexInfo[ACPI_NUM_MUTEX];

243 /*
244 * Global lock mutex is an actual AML mutex object
245 * Global lock semaphore works in conjunction with the actual global lock
246 * Global lock spinlock is used for "pending" handshake
247 */
248 ACPI_EXTERN ACPI_OPERAND_OBJECT *AcpiGbl_GlobalLockMutex;
249 ACPI_EXTERN ACPI_SEMAPHORE      AcpiGbl_GlobalLockSemaphore;
250 ACPI_EXTERN ACPI_SPINLOCK       AcpiGbl_GlobalLockPendingLock;
251 ACPI_EXTERN UINT16              AcpiGbl_GlobalLockHandle;
252 ACPI_EXTERN BOOLEAN             AcpiGbl_GlobalLockAcquired;
253 ACPI_EXTERN BOOLEAN             AcpiGbl_GlobalLockPresent;

```



```

254 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];
290 ACPI_EXTERN ACPI_OBJECT_NOTIFY_HANDLER  AcpiGbl_DeviceNotify;
291 ACPI_EXTERN ACPI_OBJECT_NOTIFY_HANDLER  AcpiGbl_SystemNotify;
292 ACPI_EXTERN ACPI_EXCEPTION_HANDLER      AcpiGbl_ExceptionHandler;
293 ACPI_EXTERN ACPI_INIT_HANDLER           AcpiGbl_InitHandler;
294 ACPI_EXTERN ACPI_TABLE_HANDLER          AcpiGbl_TableHandler;
295 ACPI_EXTERN void                        *AcpiGbl_TableHandlerContext;
296 ACPI_EXTERN ACPI_WALK_STATE              *AcpiGbl_BreakpointWalk;
297 ACPI_EXTERN ACPI_INTERFACE_HANDLER      AcpiGbl_InterfaceHandler;
298 ACPI_EXTERN ACPI_SCI_HANDLER_INFO      *AcpiGbl_SciHandlerList;

299 /* 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_PmlEnableRegisterSave;
315 ACPI_EXTERN UINT8                        AcpiGbl_DebuggerConfiguration;
316 ACPI_EXTERN BOOLEAN                      AcpiGbl_StepToNextCall;
317 ACPI_EXTERN BOOLEAN                      AcpiGbl_AcpiHardwarePresent;

```

```

318 ACPI_EXTERN BOOLEAN                      AcpiGbl_EventsInitialized;
319 ACPI_EXTERN ACPI_INTERFACE_INFO        *AcpiGbl_SupportedInterfaces;
320 ACPI_EXTERN ACPI_ADDRESS_RANGE        *AcpiGbl_AddressRangeList[ACPI_ADDRESS_RA

322 #ifndef DEFINE_ACPI_GLOBALS

324 /* Exception codes */

326 extern char const                        *AcpiGbl_ExceptionNames_Env[];
327 extern char const                        *AcpiGbl_ExceptionNames_Pgm[];
328 extern char const                        *AcpiGbl_ExceptionNames_Tbl[];
329 extern char const                        *AcpiGbl_ExceptionNames_Aml[];
330 extern char const                        *AcpiGbl_ExceptionNames_Ctrl[];

332 /* Other miscellaneous */

334 extern BOOLEAN                          AcpiGbl_Shutdown;
335 extern UINT32                           AcpiGbl_StartupFlags;
336 extern const char                        *AcpiGbl_SleepStateNames[ACPI_S_STATE_COU
337 extern const char                        *AcpiGbl_LowestDstateNames[ACPI_NUM_SxW_M
338 extern const char                        *AcpiGbl_HighestDstateNames[ACPI_NUM_SxD_
339 extern const ACPI_OPCODE_INFO            AcpiGbl_AmlOpInfo[AML_NUM_OPCODES];
340 extern const char                        *AcpiGbl_RegionTypes[ACPI_NUM_PREDEFINED_
341 #endif

343 #ifdef ACPI_DBG_TRACK_ALLOCATIONS

345 /* Lists for tracking memory allocations */

347 ACPI_EXTERN ACPI_MEMORY_LIST             *AcpiGbl_GlobalList;
348 ACPI_EXTERN ACPI_MEMORY_LIST             *AcpiGbl_NsNodeList;
349 ACPI_EXTERN BOOLEAN                      AcpiGbl_DisplayFinalMemStats;
350 ACPI_EXTERN BOOLEAN                      AcpiGbl_DisableMemTracking;
351 #endif

353 /*****
354  *
355  * Namespace globals
356  *
357  *****/

359 #if !defined (ACPI_NO_METHOD_EXECUTION) || defined (ACPI_CONSTANT_EVAL_ONLY)
360 #define NUM_PREDEFINED_NAMES              10
361 #else
362 #define NUM_PREDEFINED_NAMES              9
363 #endif

365 ACPI_EXTERN ACPI_NAMESPACE_NODE          AcpiGbl_RootNodeStruct;
366 ACPI_EXTERN ACPI_NAMESPACE_NODE          *AcpiGbl_RootNode;
367 ACPI_EXTERN ACPI_NAMESPACE_NODE          *AcpiGbl_FadtGpeDevice;
368 ACPI_EXTERN ACPI_OPERAND_OBJECT          *AcpiGbl_ModuleCodeList;

369 extern const UINT8                       AcpiGbl_NsProperties [ACPI_NUM_NS_TYPES]
370 extern const ACPI_PREDEFINED_NAMES       AcpiGbl_PreDefinedNames [NUM_PREDEFINED_

372 #ifdef ACPI_DEBUG_OUTPUT
373 ACPI_EXTERN UINT32                       AcpiGbl_CurrentNodeCount;
374 ACPI_EXTERN UINT32                       AcpiGbl_CurrentNodeSize;
375 ACPI_EXTERN UINT32                       AcpiGbl_MaxConcurrentNodeCount;
376 ACPI_EXTERN ACPI_SIZE                    *AcpiGbl_EntryStackPointer;
377 ACPI_EXTERN ACPI_SIZE                    *AcpiGbl_LowestStackPointer;

```

```

374 ACPI_EXTERN UINT32                AcpiGbl_Deepnesting;
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_XRPT_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
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_S
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_S
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

```

new/usr/src/common/acpica/include/acglobal.h

9

```
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 #ifndef 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__ */
```

new/usr/src/common/acpica/include/achware.h

1

```
*****
5682 Fri Jan 17 07:30:56 2014
new/usr/src/common/acpica/include/achware.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: achware.h -- hardware specific interfaces
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACHWARE_H__
45 #define __ACHWARE_H__

48 /* Values for the _SST predefined method */

50 #define ACPI_SST_INDICATOR_OFF 0
51 #define ACPI_SST_WORKING 1
52 #define ACPI_SST_WAKING 2
53 #define ACPI_SST_SLEEPING 3
54 #define ACPI_SST_SLEEP_CONTEXT 4

57 /*
58 * hwacpi - high level functions
```

new/usr/src/common/acpica/include/achware.h

2

```
59 */
60 ACPI_STATUS
61 AcpiHwSetMode (
62     UINT32                      Mode);

64 UINT32
65 AcpiHwGetMode (
66     void);

69 /*
70 * hwregs - ACPI Register I/O
71 */
72 ACPI_STATUS
73 AcpiHwValidateRegister (
74     ACPI_GENERIC_ADDRESS        *Reg,
75     UINT8                       MaxBitWidth,
76     UINT64                      *Address);

78 ACPI_STATUS
79 AcpiHwRead (
80     UINT32                      *Value,
81     ACPI_GENERIC_ADDRESS        *Reg);

83 ACPI_STATUS
84 AcpiHwWrite (
85     UINT32                      Value,
86     ACPI_GENERIC_ADDRESS        *Reg);

88 ACPI_BIT_REGISTER_INFO *
89 AcpiHwGetBitRegisterInfo (
90     UINT32                      RegisterId);

92 ACPI_STATUS
93 AcpiHwWritePmlControl (
94     UINT32                      PmlaControl,
95     UINT32                      PmlbControl);

97 ACPI_STATUS
98 AcpiHwRegisterRead (
99     UINT32                      RegisterId,
100     UINT32                      *ReturnValue);

102 ACPI_STATUS
103 AcpiHwRegisterWrite (
104     UINT32                      RegisterId,
105     UINT32                      Value);

107 ACPI_STATUS
108 AcpiHwClearAcpiStatus (
109     void);

112 /*
113 * hwsleep - sleep/wake support (Legacy sleep registers)
114 */
115 ACPI_STATUS
116 AcpiHwLegacySleep (
117     UINT8                      SleepState);

119 ACPI_STATUS
120 AcpiHwLegacyWakePrep (
121     UINT8                      SleepState);

123 ACPI_STATUS
124 AcpiHwLegacyWake (
```

```

125     UINT8                SleepState);

128 /*
129  * hwesleep - sleep/wake support (Extended FADT-V5 sleep registers)
130  */
131 void
132 AcpiHwExecutesSleepMethod (
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);
171
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__ */

```

new/usr/src/common/acpica/include/acinterp.h

1

```
*****
18202 Fri Jan 17 07:30:56 2014
new/usr/src/common/acpica/include/acinterp.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acinterp.h - Interpreter subcomponent prototypes and defines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #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
```

new/usr/src/common/acpica/include/acinterp.h

2

```
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
```

new/usr/src/common/acpica/include/acinterp.h

```
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  */
```

3

new/usr/src/common/acpica/include/acinterp.h

```
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);
```

4

```

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
522 AcpiExGetNameString (
523     ACPI_OBJECT_TYPE      DataType,
524     UINT8                  *InAmlAddress,
525     char                    **OutNameString,
526     UINT32                 *OutNameLength);

529 /*
530  * exstore - Object store support
531 */
532 ACPI_STATUS
533 AcpiExStore (
534     ACPI_OPERAND_OBJECT    *ValDesc,
535     ACPI_OPERAND_OBJECT    *DestDesc,
536     ACPI_WALK_STATE        *WalkState);

538 ACPI_STATUS
539 AcpiExStoreObjectToNode (
540     ACPI_OPERAND_OBJECT    *SourceDesc,
541     ACPI_NAMESPACE_NODE    *Node,
542     ACPI_WALK_STATE        *WalkState,
543     UINT8                   ImplicitConversion);

545 #define ACPI_IMPLICIT_CONVERSION      TRUE
546 #define ACPI_NO_IMPLICIT_CONVERSION  FALSE

549 /*
550  * exstoren - resolve/store object
551 */
552 ACPI_STATUS
553 AcpiExResolveObject (
554     ACPI_OPERAND_OBJECT    **SourceDescPtr,
555     ACPI_OBJECT_TYPE        TargetType,
556     ACPI_WALK_STATE        *WalkState);

558 ACPI_STATUS
559 AcpiExStoreObjectToObject (
560     ACPI_OPERAND_OBJECT    *SourceDesc,
561     ACPI_OPERAND_OBJECT    *DestDesc,
562     ACPI_OPERAND_OBJECT    *NewDesc,
563     ACPI_WALK_STATE        *WalkState);

566 /*
567  * exstorob - store object - buffer/string
568 */
569 ACPI_STATUS
570 AcpiExStoreBufferToBuffer (
571     ACPI_OPERAND_OBJECT    *SourceDesc,
572     ACPI_OPERAND_OBJECT    *TargetDesc);

574 ACPI_STATUS
575 AcpiExStoreStringToString (
576     ACPI_OPERAND_OBJECT    *SourceDesc,
577     ACPI_OPERAND_OBJECT    *TargetDesc);

580 /*
581  * excopy - object copy
582 */
583 ACPI_STATUS
584 AcpiExCopyIntegerToIndexField (
585     ACPI_OPERAND_OBJECT    *SourceDesc,
586     ACPI_OPERAND_OBJECT    *TargetDesc);

```

```

588 ACPI_STATUS
589 AcpiExCopyIntegerToBankField (
590     ACPI_OPERAND_OBJECT    *SourceDesc,
591     ACPI_OPERAND_OBJECT    *TargetDesc);

593 ACPI_STATUS
594 AcpiExCopyDataToNamedField (
595     ACPI_OPERAND_OBJECT    *SourceDesc,
596     ACPI_NAMESPACE_NODE    *Node);

598 ACPI_STATUS
599 AcpiExCopyIntegerToBufferField (
600     ACPI_OPERAND_OBJECT    *SourceDesc,
601     ACPI_OPERAND_OBJECT    *TargetDesc);

604 /*
605  * exutils - interpreter/scanner utilities
606 */
607 void
608 AcpiExEnterInterpreter (
609     void);

611 void
612 AcpiExExitInterpreter (
613     void);

615 void
616 AcpiExReacquireInterpreter (
617     void);

619 void
620 AcpiExRelinquishInterpreter (
621     void);

623 BOOLEAN
624 void
625 AcpiExTruncateFor32bitTable (
626     ACPI_OPERAND_OBJECT    *ObjDesc);

627 void
628 AcpiExAcquireGlobalLock (
629     UINT32                  Rule);

631 void
632 AcpiExReleaseGlobalLock (
633     UINT32                  Rule);

635 void
636 AcpiExEisaIdToString (
637     char                    *Dest,
638     UINT64                  CompressedId);

640 void
641 AcpiExIntegerToString (
642     char                    *Dest,
643     UINT64                  Value);

645 BOOLEAN
646 AcpiIsValidSpaceId (
647     UINT8                   SpaceId);

650 /*
651  * exregion - default OpRegion handlers

```

```

652 */
653 ACPI_STATUS
654 AcpiExSystemMemorySpaceHandler (
655     UINT32          Function,
656     ACPI_PHYSICAL_ADDRESS Address,
657     UINT32          BitWidth,
658     UINT64          *Value,
659     void            *HandlerContext,
660     void            *RegionContext);

662 ACPI_STATUS
663 AcpiExSystemIoSpaceHandler (
664     UINT32          Function,
665     ACPI_PHYSICAL_ADDRESS Address,
666     UINT32          BitWidth,
667     UINT64          *Value,
668     void            *HandlerContext,
669     void            *RegionContext);

671 ACPI_STATUS
672 AcpiExPciConfigSpaceHandler (
673     UINT32          Function,
674     ACPI_PHYSICAL_ADDRESS Address,
675     UINT32          BitWidth,
676     UINT64          *Value,
677     void            *HandlerContext,
678     void            *RegionContext);

680 ACPI_STATUS
681 AcpiExCmosSpaceHandler (
682     UINT32          Function,
683     ACPI_PHYSICAL_ADDRESS Address,
684     UINT32          BitWidth,
685     UINT64          *Value,
686     void            *HandlerContext,
687     void            *RegionContext);

689 ACPI_STATUS
690 AcpiExPciBarSpaceHandler (
691     UINT32          Function,
692     ACPI_PHYSICAL_ADDRESS Address,
693     UINT32          BitWidth,
694     UINT64          *Value,
695     void            *HandlerContext,
696     void            *RegionContext);

698 ACPI_STATUS
699 AcpiExEmbeddedControllerSpaceHandler (
700     UINT32          Function,
701     ACPI_PHYSICAL_ADDRESS Address,
702     UINT32          BitWidth,
703     UINT64          *Value,
704     void            *HandlerContext,
705     void            *RegionContext);

707 ACPI_STATUS
708 AcpiExSmbusSpaceHandler (
709     UINT32          Function,
710     ACPI_PHYSICAL_ADDRESS Address,
711     UINT32          BitWidth,
712     UINT64          *Value,
713     void            *HandlerContext,
714     void            *RegionContext);

```

```

717 ACPI_STATUS

```

```

718 AcpiExDataTableSpaceHandler (
719     UINT32          Function,
720     ACPI_PHYSICAL_ADDRESS Address,
721     UINT32          BitWidth,
722     UINT64          *Value,
723     void            *HandlerContext,
724     void            *RegionContext);

726 #endif /* __INTERP_H__ */

```

new/usr/src/common/acpica/include/aclocal.h

1

```
*****
47041 Fri Jan 17 07:30:57 2014
new/usr/src/common/acpica/include/aclocal.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: aclocal.h - Internal data types used across the ACPI subsystem
4 *
5 *****/
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACLOCAL_H__
45 #define __ACLOCAL_H__
46
47 /* acpisrc:StructDefs -- for acpisrc conversion */
48
49 #define ACPI_SERIALIZED 0xFF
50
51 typedef UINT32 ACPI_MUTEX_HANDLE;
52 #define ACPI_GLOBAL_LOCK (ACPI_SEMAPHORE) (-1)
53
54 /* Total number of aml opcodes defined */
```

new/usr/src/common/acpica/include/aclocal.h

2

```
57 #define AML_NUM_OPCODES 0x81
57 #define AML_NUM_OPCODES 0x7F
58
59 /* Forward declarations */
60
61 struct acpi_walk_state;
62 struct acpi_obj_mutex;
63 union acpi_parse_object;
64
65 /*****
66 *
67 * Mutex typedefs and structs
68 *
69 *****/
70
71 /*****
72 *
73 * Predefined handles for the mutex objects used within the subsystem
74 * All mutex objects are automatically created by AcpiUtMutexInitialize.
75 *
76 * The acquire/release ordering protocol is implied via this list. Mutexes
77 * with a lower value must be acquired before mutexes with a higher value.
78 *
79 * NOTE: any changes here must be reflected in the AcpiGbl_MutexNames
80 * table below also!
81 */
82 #define ACPI_MTX_INTERPRETER 0 /* AML Interpreter, main lock */
83 #define ACPI_MTX_NAMESPACE 1 /* ACPI Namespace */
84 #define ACPI_MTX_TABLES 2 /* Data for ACPI tables */
85 #define ACPI_MTX_EVENTS 3 /* Data for ACPI events */
86 #define ACPI_MTX_CACHES 4 /* Internal caches, general purposes */
87 #define ACPI_MTX_MEMORY 5 /* Debug memory tracking lists */
88 #define ACPI_MTX_DEBUG_CMD_COMPLETE 6 /* AML debugger */
89 #define ACPI_MTX_DEBUG_CMD_READY 7 /* AML debugger */
90
91 #define ACPI_MAX_MUTEX 7
92 #define ACPI_NUM_MUTEX ACPI_MAX_MUTEX+1
93
94 /* Lock structure for reader/writer interfaces */
95
96 typedef struct acpi_rw_lock
97 {
98     ACPI_MUTEX WriterMutex;
99     ACPI_MUTEX ReaderMutex;
100     UINT32 NumReaders;
101 } ACPI_RW_LOCK;
102
103 /* Namespace Node flags */
104
105 #define ANOBJ_RESERVED 0x01 /* Available for use */
106 #define ANOBJ_TEMPORARY 0x02 /* Node is created by a method and
107 * is a method argument */
108 #define ANOBJ_METHOD_ARG 0x04 /* Node is a method argument */
109 #define ANOBJ_METHOD_LOCAL 0x08 /* Node is a method local */
110 #define ANOBJ_SUBTREE_HAS_INI 0x10 /* Used to optimize device initialization
111 * Set on first evaluation of no
112 * Method AML buffer is dynamic */
113 #define ANOBJ_EVALUATED 0x20
114 #define ANOBJ_ALLOCATED_BUFFER 0x40
115
116 #define ANOBJ_IS_EXTERNAL 0x08 /* iASL only: This object created
117 * Method has no return value */
118 #define ANOBJ_METHOD_NO_RETVAL 0x10
119 #define ANOBJ_METHOD_SOME_NO_RETVAL 0x20 /* iASL only: Method has at least
```

```

216 #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 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_PTTYPE1_FIXED, ACPI_PTTYPE1_VAR, ACPI_PTTYPE2,
366 * ACPI_PTTYPE2_MIN, ACPI_PTTYPE2_PKG_COUNT, ACPI_PTTYPE2_COUNT,
367 * ACPI_PTTYPE2_FIX_VAR
368 * ACPI_PTTYPE2_MIN, ACPI_PTTYPE2_PKG_COUNT, ACPI_PTTYPE2_COUNT
369 */
369 typedef struct acpi_package_info
370 {
371     UINT8 Type;
372     UINT8 ObjectTypeId;
373     UINT8 Count1;
374     UINT8 ObjectTypeId2;
375     UINT8 Count2;
376     UINT16 Reserved;
377     UINT8 Reserved;

378 } ACPI_PACKAGE_INFO;

380 /* Used for ACPI_PTTYPE2_FIXED */

382 typedef struct acpi_package_info2
383 {
384     UINT8 Type;
385     UINT8 Count;
386     UINT8 ObjectTypeId[4];
387     UINT8 Reserved;

389 } ACPI_PACKAGE_INFO2;

391 /* Used for ACPI_PTTYPE1_OPTION */

```

```

393 typedef struct acpi_package_info3
394 {
395     UINT8          Type;
396     UINT8          Count;
397     UINT8          ObjectType[2];
398     UINT8          TailObjectType;
399     UINT16         Reserved;
400     Reserved;
401 } ACPI_PACKAGE_INFO3;
402 unchanged_portion_omitted
403
404 /* Reset to default packing */
405
406 #pragma pack()
407 /* Data block used during object validation */
408
409 typedef struct acpi_predefined_data
410 {
411     char          *Pathname;
412     const ACPI_PREDEFINED_INFO *Predefined;
413     union acpi_operand_object *ParentPackage;
414     UINT32        Flags;
415     UINT8         NodeFlags;
416
417 /* Return object auto-repair info */
418 } ACPI_PREDEFINED_DATA;
419
420 typedef ACPI_STATUS (*ACPI_OBJECT_CONVERTER) (
421     union acpi_operand_object *OriginalObject,
422     union acpi_operand_object **ConvertedObject);
423 /* Defines for Flags field above */
424
425 typedef struct acpi_simple_repair_info
426 {
427     char          Name[ACPI_NAME_SIZE];
428     UINT32        UnexpectedBtypes;
429     UINT32        PackageIndex;
430     ACPI_OBJECT_CONVERTER ObjectConverter;
431 #define ACPI_OBJECT_REPAIRED 1
432
433 } ACPI_SIMPLE_REPAIR_INFO;
434
435 /*
436 * Bitmapped return value types
437 * Note: the actual data types must be contiguous, a loop in nspredef.c
438 * depends on this.
439 */
440 #define ACPI_RTYPE_ANY          0x00
441 #define ACPI_RTYPE_NONE        0x01
442 #define ACPI_RTYPE_INTEGER     0x02
443 #define ACPI_RTYPE_STRING      0x04
444 #define ACPI_RTYPE_BUFFER      0x08
445 #define ACPI_RTYPE_PACKAGE     0x10
446 #define ACPI_RTYPE_REFERENCE   0x20
447 #define ACPI_RTYPE_ALL         0x3F
448
449 #define ACPI_NUM_RTYPES        5 /* Number of actual object types */
450
451 /*
452 * Event typedefs and structs
453 */

```

```

454 *****
455
456 /* Dispatch info for each host-installed SCI handler */
457
458 typedef struct acpi_sci_handler_info
459 {
460     struct acpi_sci_handler_info *Next;
461     ACPI_SCI_HANDLER             Address; /* Address of handler */
462     void                         *Context; /* Context to be passed to h
463
464 } ACPI_SCI_HANDLER_INFO;
465
466 /* Dispatch info for each GPE -- either a method or handler, cannot be both */
467
468 typedef struct acpi_gpe_handler_info
469 {
470     ACPI_GPE_HANDLER             Address; /* Address of handler, if an
471     void                         *Context; /* Context to be passed to h
472     ACPI_NAMESPACE_NODE          *MethodNode; /* Method node for this GPE
473     UINT8                        OriginalFlags; /* Original (pre-handler) GP
474     BOOLEAN                      OriginallyEnabled; /* True if GPE was origin
475
476 } ACPI_GPE_HANDLER_INFO;
477
478 /* Notify info for implicit notify, multiple device objects */
479
480 typedef struct acpi_gpe_notify_info
481 {
482     ACPI_NAMESPACE_NODE          *DeviceNode; /* Device to be notified */
483     struct acpi_gpe_notify_info *Next;
484
485 } ACPI_GPE_NOTIFY_INFO;
486
487 /*
488 * GPE dispatch info. At any time, the GPE can have at most one type
489 * of dispatch - Method, Handler, or Implicit Notify.
490 */
491 typedef union acpi_gpe_dispatch_info
492 {
493     ACPI_NAMESPACE_NODE          *MethodNode; /* Method node for this GPE
494     ACPI_GPE_HANDLER_INFO        *Handler; /* Installed GPE handler */
495     ACPI_GPE_NOTIFY_INFO         *NotifyList; /* List of _PRW devices for
496     struct acpi_gpe_handler_info *Handler; /* Installed GPE handler */
497     ACPI_NAMESPACE_NODE          *DeviceNode; /* Parent _PRW device for im
498
499 } ACPI_GPE_DISPATCH_INFO;
500
501 /*
502 * Information about a GPE, one per each GPE in an array.
503 * NOTE: Important to keep this struct as small as possible.
504 */
505 typedef struct acpi_gpe_event_info
506 {
507     union acpi_gpe_dispatch_info Dispatch; /* Either Method, Handler, o
508     union acpi_gpe_dispatch_info *Dispatch; /* Either Method or Handler
509     struct acpi_gpe_register_info *RegisterInfo; /* Backpointer to register i
510     struct acpi_gpe_register_info *RegisterInfo; /* Backpointer to register i
511     UINT8                        Flags; /* Misc info about this GPE
512     UINT8                        GpeNumber; /* This GPE */
513     UINT8                        RuntimeCount; /* References to a run GPE *
514
515 } ACPI_GPE_EVENT_INFO;
516 unchanged_portion_omitted
517
518 typedef
519 ACPI_STATUS (*ACPI_PARSE_DOWNWARDS) (

```

new/usr/src/common/acpica/include/aclocal.h

7

```
733 struct acpi_walk_state      *WalkState,
734 union acpi_parse_object      **OutOp);

736 typedef
737 ACPI_STATUS (*ACPI_PARSE_UPWARDS) (
738 struct acpi_walk_state      *WalkState);
```

741 /\* Global handlers for AML Notifies \*/

```
743 typedef struct acpi_global_notify_handler
744 {
745     ACPI_NOTIFY_HANDLER      Handler;
746     void                      *Context;
748 } ACPI_GLOBAL_NOTIFY_HANDLER;
```

```
750 /*
751 * Notify info - used to pass info to the deferred notify
752 * handler/dispatcher.
753 */
754 typedef struct acpi_notify_info
755 {
756     ACPI_STATE_COMMON
757     UINT8                      HandlerListId;
758     ACPI_NAMESPACE_NODE        *Node;
759     union acpi_operand_object  *HandlerListHead;
760     ACPI_GLOBAL_NOTIFY_HANDLER *Global;
761     union acpi_operand_object  *HandlerObj;
```

```
762 } ACPI_NOTIFY_INFO;
763 unchanged_portion_omitted
```

```
782 /*****
783 *
784 * Interpreter typedefs and structs
785 *
786 *****/
```

```
788 typedef
789 ACPI_STATUS (*ACPI_EXECUTE_OP) (
790 struct acpi_walk_state      *WalkState);
```

792 /\* Address Range info block \*/

```
794 typedef struct acpi_address_range
795 {
796     struct acpi_address_range *Next;
797     ACPI_NAMESPACE_NODE        *RegionNode;
798     ACPI_PHYSICAL_ADDRESS      StartAddress;
799     ACPI_PHYSICAL_ADDRESS      EndAddress;
801 } ACPI_ADDRESS_RANGE;
```

```
804 /*****
805 *
806 * Parser typedefs and structs
807 *
808 *****/
```

```
810 /*
811 * AML opcode, name, and argument layout
812 */
813 typedef struct acpi_opcode_info
```

new/usr/src/common/acpica/include/aclocal.h

8

```
814 {
815 #if defined(ACPI_DISASSEMBLER) || defined(ACPI_DEBUG_OUTPUT)
816     char                      *Name;          /* Opcode name (disassembler
817 #endif
818     UINT32                     ParseArgs;      /* Grammar/Parse time argume
819     UINT32                     RuntimeArgs;    /* Interpret time arguments
820     UINT16                     Flags;          /* Misc flags */
821     UINT8                      ObjectType;     /* Corresponding internal ob
822     UINT8                      Class;          /* Opcode class */
823     UINT8                      Type;          /* Opcode type */
825 } ACPI_OPCODE_INFO;
```

827 /\* Structure for Resource Tag information \*/

```
829 typedef struct acpi_tag_info
830 {
831     UINT32                     BitOffset;
832     UINT32                     BitLength;
834 } ACPI_TAG_INFO;
```

836 /\* Value associated with the parse object \*/

```
838 typedef union acpi_parse_value
839 {
840     UINT64                     Integer;        /* Integer constant (Up to 6
841     UINT32                     Size;          /* bytelist or field size */
842     char                       *String;       /* NULL terminated string */
843     UINT8                      *Buffer;       /* buffer or string */
844     char                       *Name;         /* NULL terminated string */
845     union acpi_parse_object     *Arg;         /* arguments and contained o
846     ACPI_TAG_INFO              Tag;          /* Resource descriptor tag i
```

848 } ACPI\_PARSE\_VALUE;

```
851 #ifdef ACPI_DISASSEMBLER
852 #define ACPI_DISASM_ONLY_MEMBERS(a)      a;
853 #else
854 #define ACPI_DISASM_ONLY_MEMBERS(a)
855 #endif
```

```
857 #define ACPI_PARSE_COMMON \
858 union acpi_parse_object
859 {
860     UINT8                      *Parent;        /* Parent op */
861     UINT8                      DescriptorType; /* To differentiate various
862     UINT8                      Flags;          /* Type of Op */
863     UINT16                     AmlOpcode;      /* AML opcode */
864     UINT32                     AmlOffset;      /* Offset of declaration in
865     union acpi_parse_object     *Next;         /* Next op */
866     ACPI_NAMESPACE_NODE        *Node;         /* For use by interpreter */
867     ACPI_PARSE_VALUE            Value;         /* Value or args associated
868     UINT8                      ArgListLength; /* Number of elements in the
869     ACPI_DISASM_ONLY_MEMBERS (\
870     UINT8                      DisasmFlags;   /* Used during AML disassemb
871     UINT8                      DisasmOpcode;  /* Subtype used for disassem
872     char                       AmlOpName[16]; /* Op name (debug only) */
```

873 /\* Flags for DisasmFlags field above \*/

```
811 #define ACPI_DASM_BUFFER          0x00
812 #define ACPI_DASM_RESOURCE        0x01
813 #define ACPI_DASM_STRING          0x02
814 #define ACPI_DASM_UNICODE         0x03
815 #define ACPI_DASM_EISAID          0x04
816 #define ACPI_DASM_MATCHOP         0x05
```

```

817 #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;
893 #define 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_RESERVED_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_RESERVED_BITS     \
1030     (ACPI_PM1_CONTROL_IGNORED_BITS | ACPI_PM1_CONTROL_RESERVED_BITS)

1032 #define ACPI_PM2_CONTROL_RESERVED_BITS      0xFFFFFFF0 /* 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

```



new/usr/src/common/acpica/include/aclocal.h

11

```
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;
```

new/usr/src/common/acpica/include/aclocal.h

12

```
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
1173 #define ACPI_RESOURCE_NAME_RESERVED_S1 0x58
1174 #define ACPI_RESOURCE_NAME_RESERVED_S2 0x58
1175 #define ACPI_RESOURCE_NAME_RESERVED_S3 0x60
1176 #define ACPI_RESOURCE_NAME_RESERVED_S4 0x68
1177 #define ACPI_RESOURCE_NAME_VENDOR_SMALL 0x70
1178 #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
1197 #define ACPI_RESOURCE_NAME_LARGE_MAX 0x8B
```

```

1199 /*****
1200 *
1201 * Miscellaneous
1202 *
1203 *****/

1205 #define ACPI_ASCII_ZERO          0x30

1208 /*****
1209 *
1210 * Disassembler
1211 *
1212 *****/

1214 typedef struct acpi_external_list
1215 {
1216     char                *Path;
1217     char                *InternalPath;
1218     struct acpi_external_list *Next;
1219     UINT32              Value;
1220     UINT16              Length;
1221     UINT16              Flags;
1222     UINT8               Type;
1223     UINT8              Flags;
1224 } ACPI_EXTERNAL_LIST;

1226 /* Values for Flags field above */

1228 #define ACPI_EXT_RESOLVED_REFERENCE    0x01 /* Object was resolved durin
1229 #define ACPI_EXT_ORIGIN_FROM_FILE    0x02 /* External came from a file
1230 #define ACPI_EXT_INTERNAL_PATH_ALLOCATED 0x04 /* Deallocate internal path
1231 #define ACPI_EXT_EXTERNAL_EMITTED    0x08 /* External() statement has
1169 #define ACPI_IPATH_ALLOCATED    0x01

1234 typedef struct acpi_external_file
1235 {
1236     char                *Path;
1237     struct acpi_external_file *Next;
1238 } ACPI_EXTERNAL_FILE;

1242 /*****
1243 *
1244 * Debugger
1245 *
1246 *****/

1248 typedef struct acpi_db_method_info
1249 {
1250     ACPI_HANDLE          Method;
1251     ACPI_HANDLE          MainThreadGate;
1252     ACPI_HANDLE          ThreadCompleteGate;
1253     ACPI_HANDLE          InfoGate;
1254     ACPI_THREAD_ID       *Threads;
1255     UINT32               NumThreads;
1256     UINT32               NumCreated;
1257     UINT32               NumCompleted;

1259     char                *Name;
1260     UINT32               Flags;
1261     UINT32               NumLoops;

```

```

1262     char                Pathname[ACPI_DB_LINE_BUFFER_SIZE];
1199     char                Pathname[128];
1263     char                **Args;
1264     ACPI_OBJECT_TYPE    *Types;

1266     /*
1267     * Arguments to be passed to method for the command
1268     * Threads -
1269     *   the Number of threads, ID of current thread and
1270     *   Index of current thread inside all them created.
1271     */
1272     char                InitArgs;
1273     ACPI_OBJECT_TYPE    ArgTypes[4];
1274     char                *Arguments[4];
1275     char                NumThreadsStr[11];
1276     char                IdOfThreadStr[11];
1277     char                IndexOfThreadStr[11];

1279 } ACPI_DB_METHOD_INFO;
    unchanged portion omitted

1289 #define ACPI_DB_DISABLE_OUTPUT    0x00
1290 #define ACPI_DB_REDIRECTABLE_OUTPUT 0x01
1291 #define ACPI_DB_CONSOLE_OUTPUT    0x02
1292 #define ACPI_DB_DUPLICATE_OUTPUT  0x03

1295 /*****
1296 *
1297 * Debug
1298 *
1299 *****/

1301 /* Entry for a memory allocation (debug only) */

1303 #define ACPI_MEM_MALLOC            0
1304 #define ACPI_MEM_CALLOC            1
1305 #define ACPI_MAX_MODULE_NAME      16

1307 #define ACPI_COMMON_DEBUG_MEM_HEADER \
1308     struct acpi_debug_mem_block    *Previous; \
1309     struct acpi_debug_mem_block    *Next; \
1310     UINT32                          Size; \
1311     UINT32                          Component; \
1312     UINT32                          Line; \
1313     char                            Module[ACPI_MAX_MODULE_NAME]; \
1314     UINT8                          AllocType;

1316 typedef struct acpi_debug_mem_header
1317 {
1318     ACPI_COMMON_DEBUG_MEM_HEADER
1319 } ACPI_DEBUG_MEM_HEADER;
    unchanged portion omitted

1330 #define ACPI_MEM_LIST_GLOBAL      0
1331 #define ACPI_MEM_LIST_NSNODE     1
1332 #define ACPI_MEM_LIST_MAX        1
1333 #define ACPI_NUM_MEM_LISTS       2

1336 /*****
1337 *
1338 * Info/help support

```

```
1339 *
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__ */
```

new/usr/src/common/acpica/include/acmacros.h

1

```
*****
19969 Fri Jan 17 07:30:57 2014
new/usr/src/common/acpica/include/acmacros.h
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Name: acmacros.h - C macros for the entire subsystem.
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #ifndef __ACMACROS_H__
45 #define __ACMACROS_H__
46
47
48 /*
49 * Extract data using a pointer. Any more than a byte and we
50 * get into potential alignment issues -- see the STORE macros below.
51 * Use with care.
52 */
53 #define ACPI_CAST8(ptr)          ACPI_CAST_PTR (UINT8, (ptr))
54 #define ACPI_CAST16(ptr)         ACPI_CAST_PTR (UINT16, (ptr))
55 #define ACPI_CAST32(ptr)         ACPI_CAST_PTR (UINT32, (ptr))
56 #define ACPI_CAST64(ptr)         ACPI_CAST_PTR (UINT64, (ptr))
57 #define ACPI_GET8(ptr)           (*ACPI_CAST8 (ptr))
58 #define ACPI_GET16(ptr)          (*ACPI_CAST16 (ptr))
```

new/usr/src/common/acpica/include/acmacros.h

2

```
59 #define ACPI_GET32(ptr)          (*ACPI_CAST32 (ptr))
60 #define ACPI_GET64(ptr)          (*ACPI_CAST64 (ptr))
61 #define ACPI_SET8(ptr, val)      (*ACPI_CAST8 (ptr) = (UINT8) (val))
62 #define ACPI_SET16(ptr, val)     (*ACPI_CAST16 (ptr) = (UINT16) (val))
63 #define ACPI_SET32(ptr, val)     (*ACPI_CAST32 (ptr) = (UINT32) (val))
64 #define ACPI_SET64(ptr, val)     (*ACPI_CAST64 (ptr) = (UINT64) (val))
65
66 /*
67 * printf() format helpers. These macros are workarounds for the difficulties
68 * with emitting 64-bit integers and 64-bit pointers with the same code
69 * for both 32-bit and 64-bit hosts.
70 */
71 #define ACPI_FORMAT_UINT64(i)     ACPI_HIWORD(i), ACPI_LOWORD(i)
72
73 #if ACPI_MACHINE_WIDTH == 64
74 #define ACPI_FORMAT_NATIVE_UINT(i) ACPI_FORMAT_UINT64(i)
75 #define ACPI_FORMAT_TO_UINT(i)     ACPI_FORMAT_UINT64(i)
76 #define ACPI_PRINTF_UINT           "0x%8.8X%8.8X"
77
78 #else
79 #define ACPI_FORMAT_NATIVE_UINT(i) 0, (UINT32) (i)
80 #define ACPI_FORMAT_TO_UINT(i)     (UINT32) (i)
81 #define ACPI_PRINTF_UINT           "0x%8.8X"
82 #endif
83
84 /*
85 * Macros for moving data around to/from buffers that are possibly unaligned.
86 * If the hardware supports the transfer of unaligned data, just do the store.
87 * Otherwise, we have to move one byte at a time.
88 */
89 #ifndef ACPI_BIG_ENDIAN
90 /*
91 * Macros for big-endian machines
92 */
93
94
95 /* These macros reverse the bytes during the move, converting little-endian to b
96
97                                     /* Big Endian      <==
98                                     /* Hi...Lo
99 /* 16-bit source, 16/32/64 destination */
100
101 #define ACPI_MOVE_16_TO_16(d, s)    (((UINT8 *) (void *) (d))[0] = ((UINT8 *)
102                                     ((UINT8 *) (void *) (d))[1] = ((UINT8 *)
103
104 #define ACPI_MOVE_16_TO_32(d, s)    ((* (UINT32 *) (void *) (d)) = 0; \
105                                     ((UINT8 *) (void *) (d))[2] = ((UINT8 *)
106                                     ((UINT8 *) (void *) (d))[3] = ((UINT8 *)
107
108 #define ACPI_MOVE_16_TO_64(d, s)    ((* (UINT64 *) (void *) (d)) = 0; \
109                                     ((UINT8 *) (void *) (d))[6] = ((UINT8 *)
110                                     ((UINT8 *) (void *) (d))[7] = ((UINT8 *)
111
112 /* 32-bit source, 16/32/64 destination */
113
114 #define ACPI_MOVE_32_TO_16(d, s)     ACPI_MOVE_16_TO_16(d, s) /* Truncate
115
116 #define ACPI_MOVE_32_TO_32(d, s)    (((UINT8 *) (void *) (d))[0] = ((UINT8 *)
117                                     ((UINT8 *) (void *) (d))[1] = ((UINT8 *)
118                                     ((UINT8 *) (void *) (d))[2] = ((UINT8 *)
119                                     ((UINT8 *) (void *) (d))[3] = ((UINT8 *)
120
121 #define ACPI_MOVE_32_TO_64(d, s)    ((* (UINT64 *) (void *) (d)) = 0; \
122                                     ((UINT8 *) (void *) (d))[4] = ((UINT8 *)
123                                     ((UINT8 *) (void *) (d))[5] = ((UINT8 *)
124                                     ((UINT8 *) (void *) (d))[6] = ((UINT8 *)
```

```

125             ((UINT8 *) (void *) (d))[7] = ((UINT8 *)
127 /* 64-bit source, 16/32/64 destination */
129 #define ACPI_MOVE_64_TO_16(d, s)      ACPI_MOVE_16_TO_16(d, s)    /* Truncate
131 #define ACPI_MOVE_64_TO_32(d, s)      ACPI_MOVE_32_TO_32(d, s)    /* Truncate
133 #define ACPI_MOVE_64_TO_64(d, s)      {((UINT8 *) (void *) (d))[0] = ((UINT8 *)
134                                         ((UINT8 *) (void *) (d))[1] = ((UINT8 *)
135                                         ((UINT8 *) (void *) (d))[2] = ((UINT8 *)
136                                         ((UINT8 *) (void *) (d))[3] = ((UINT8 *)
137                                         ((UINT8 *) (void *) (d))[4] = ((UINT8 *)
138                                         ((UINT8 *) (void *) (d))[5] = ((UINT8 *)
139                                         ((UINT8 *) (void *) (d))[6] = ((UINT8 *)
140                                         ((UINT8 *) (void *) (d))[7] = ((UINT8 *)
141 #else
142 /*
143  * Macros for little-endian machines
144  */
146 #ifndef ACPI_MISALIGNMENT_NOT_SUPPORTED
148 /* The hardware supports unaligned transfers, just do the little-endian move */
150 /* 16-bit source, 16/32/64 destination */
152 #define ACPI_MOVE_16_TO_16(d, s)      *(UINT16 *) (void *) (d) = *(UINT16 *) (voi
153 #define ACPI_MOVE_16_TO_32(d, s)      *(UINT32 *) (void *) (d) = *(UINT16 *) (voi
154 #define ACPI_MOVE_16_TO_64(d, s)      *(UINT64 *) (void *) (d) = *(UINT16 *) (voi
156 /* 32-bit source, 16/32/64 destination */
158 #define ACPI_MOVE_32_TO_16(d, s)      ACPI_MOVE_16_TO_16(d, s)    /* Truncate
159 #define ACPI_MOVE_32_TO_32(d, s)      *(UINT32 *) (void *) (d) = *(UINT32 *) (voi
160 #define ACPI_MOVE_32_TO_64(d, s)      *(UINT64 *) (void *) (d) = *(UINT32 *) (voi
162 /* 64-bit source, 16/32/64 destination */
164 #define ACPI_MOVE_64_TO_16(d, s)      ACPI_MOVE_16_TO_16(d, s)    /* Truncate
165 #define ACPI_MOVE_64_TO_32(d, s)      ACPI_MOVE_32_TO_32(d, s)    /* Truncate
166 #define ACPI_MOVE_64_TO_64(d, s)      *(UINT64 *) (void *) (d) = *(UINT64 *) (voi
168 #else
169 /*
170  * The hardware does not support unaligned transfers. We must move the
171  * data one byte at a time. These macros work whether the source or
172  * the destination (or both) is/are unaligned. (Little-endian move)
173  */
175 /* 16-bit source, 16/32/64 destination */
177 #define ACPI_MOVE_16_TO_16(d, s)      {((UINT8 *) (void *) (d))[0] = ((UINT8 *)
178                                         ((UINT8 *) (void *) (d))[1] = ((UINT8 *)
180 #define ACPI_MOVE_16_TO_32(d, s)      {((UINT32 *) (void *) (d)) = 0; ACPI_MOVE
181 #define ACPI_MOVE_16_TO_64(d, s)      {((UINT64 *) (void *) (d)) = 0; ACPI_MOVE
183 /* 32-bit source, 16/32/64 destination */
185 #define ACPI_MOVE_32_TO_16(d, s)      ACPI_MOVE_16_TO_16(d, s)    /* Truncate
187 #define ACPI_MOVE_32_TO_32(d, s)      {((UINT8 *) (void *) (d))[0] = ((UINT8 *)
188                                         ((UINT8 *) (void *) (d))[1] = ((UINT8 *)
189                                         ((UINT8 *) (void *) (d))[2] = ((UINT8 *)
190                                         ((UINT8 *) (void *) (d))[3] = ((UINT8 *)

```

```

192 #define ACPI_MOVE_32_TO_64(d, s)      {((UINT64 *) (void *) (d)) = 0; ACPI_MOVE
194 /* 64-bit source, 16/32/64 destination */
196 #define ACPI_MOVE_64_TO_16(d, s)      ACPI_MOVE_16_TO_16(d, s)    /* Truncate
197 #define ACPI_MOVE_64_TO_32(d, s)      ACPI_MOVE_32_TO_32(d, s)    /* Truncate
198 #define ACPI_MOVE_64_TO_64(d, s)      {((UINT8 *) (void *) (d))[0] = ((UINT8 *)
199                                         ((UINT8 *) (void *) (d))[1] = ((UINT8 *)
200                                         ((UINT8 *) (void *) (d))[2] = ((UINT8 *)
201                                         ((UINT8 *) (void *) (d))[3] = ((UINT8 *)
202                                         ((UINT8 *) (void *) (d))[4] = ((UINT8 *)
203                                         ((UINT8 *) (void *) (d))[5] = ((UINT8 *)
204                                         ((UINT8 *) (void *) (d))[6] = ((UINT8 *)
205                                         ((UINT8 *) (void *) (d))[7] = ((UINT8 *)
206 #endif
207 #endif
210 /*
211  * Fast power-of-two math macros for non-optimized compilers
212  */
213 #define _ACPI_DIV(value, PowerOf2)      ((UINT32) ((value) >> (PowerOf2)))
214 #define _ACPI_MUL(value, PowerOf2)      ((UINT32) ((value) << (PowerOf2)))
215 #define _ACPI_MOD(value, Divisor)       ((UINT32) ((value) & ((Divisor) - 1)))
217 #define ACPI_DIV_2(a)                   _ACPI_DIV(a, 1)
218 #define ACPI_MUL_2(a)                   _ACPI_MUL(a, 1)
219 #define ACPI_MOD_2(a)                   _ACPI_MOD(a, 2)
221 #define ACPI_DIV_4(a)                   _ACPI_DIV(a, 2)
222 #define ACPI_MUL_4(a)                   _ACPI_MUL(a, 2)
223 #define ACPI_MOD_4(a)                   _ACPI_MOD(a, 4)
225 #define ACPI_DIV_8(a)                   _ACPI_DIV(a, 3)
226 #define ACPI_MUL_8(a)                   _ACPI_MUL(a, 3)
227 #define ACPI_MOD_8(a)                   _ACPI_MOD(a, 8)
229 #define ACPI_DIV_16(a)                  _ACPI_DIV(a, 4)
230 #define ACPI_MUL_16(a)                  _ACPI_MUL(a, 4)
231 #define ACPI_MOD_16(a)                  _ACPI_MOD(a, 16)
233 #define ACPI_DIV_32(a)                  _ACPI_DIV(a, 5)
234 #define ACPI_MUL_32(a)                  _ACPI_MUL(a, 5)
235 #define ACPI_MOD_32(a)                  _ACPI_MOD(a, 32)
237 /*
238  * Rounding macros (Power of two boundaries only)
239  */
240 #define ACPI_ROUND_DOWN(value, boundary) (((ACPI_SIZE)(value)) & \
241                                             (~(((ACPI_SIZE) boundary)-1)))
243 #define ACPI_ROUND_UP(value, boundary)   (((ACPI_SIZE)(value)) + \
244                                             (((ACPI_SIZE) boundary)-1)) & \
245                                             (~(((ACPI_SIZE) boundary)-1)))
247 /* Note: sizeof(ACPI_SIZE) evaluates to either 4 or 8 (32- vs 64-bit mode) */
249 #define ACPI_ROUND_DOWN_TO_32BIT(a)     ACPI_ROUND_DOWN(a, 4)
250 #define ACPI_ROUND_DOWN_TO_64BIT(a)     ACPI_ROUND_DOWN(a, 8)
251 #define ACPI_ROUND_DOWN_TO_NATIVE_WORD(a) ACPI_ROUND_DOWN(a, sizeof(ACPI_SIZE))
253 #define ACPI_ROUND_UP_TO_32BIT(a)       ACPI_ROUND_UP(a, 4)
254 #define ACPI_ROUND_UP_TO_64BIT(a)       ACPI_ROUND_UP(a, 8)
255 #define ACPI_ROUND_UP_TO_NATIVE_WORD(a) ACPI_ROUND_UP(a, sizeof(ACPI_SIZE))

```

```

257 #define ACPI_ROUND_BITS_UP_TO_BYTES(a)      ACPI_DIV_8((a) + 7)
258 #define ACPI_ROUND_BITS_DOWN_TO_BYTES(a)     ACPI_DIV_8((a))

260 #define ACPI_ROUND_UP_TO_1K(a)               (((a) + 1023) >> 10)

262 /* Generic (non-power-of-two) rounding */

264 #define ACPI_ROUND_UP_TO(value, boundary)     (((value) + ((boundary)-1)) / (bound

266 #define ACPI_IS_MISALIGNED(value)             (((ACPI_SIZE) value) & (sizeof(ACPI_

268 /*
269  * Bitmask creation
270  * Bit positions start at zero.
271  * MASK_BITS_ABOVE creates a mask starting AT the position and above
272  * MASK_BITS_BELOW creates a mask starting one bit BELOW the position
273  */
274 #define ACPI_MASK_BITS_ABOVE(position)        (~(ACPI_UINT64_MAX) << ((UINT32) (p
275 #define ACPI_MASK_BITS_BELOW(position)        ((ACPI_UINT64_MAX) << ((UINT32) (pos

277 /* Bitfields within ACPI registers */

279 #define ACPI_REGISTER_PREPARE_BITS(Val, Pos, Mask) \
280     ((Val << Pos) & Mask)

282 #define ACPI_REGISTER_INSERT_VALUE(Reg, Pos, Mask, Val) \
283     Reg = (Reg & (~(Mask))) | ACPI_REGISTER_PREPARE_BITS(Val, Pos, Mask)

285 #define ACPI_INSERT_BITS(Target, Mask, Source) \
286     Target = ((Target & (~(Mask))) | (Source & Mask))

288 /* Generic bitfield macros and masks */

290 #define ACPI_GET_BITS(SourcePtr, Position, Mask) \
291     ((*SourcePtr >> Position) & Mask)

293 #define ACPI_SET_BITS(TargetPtr, Position, Mask, Value) \
294     ((*TargetPtr |= ((Value & Mask) << Position))

296 #define ACPI_1BIT_MASK      0x00000001
297 #define ACPI_2BIT_MASK      0x00000003
298 #define ACPI_3BIT_MASK      0x00000007
299 #define ACPI_4BIT_MASK      0x0000000F
300 #define ACPI_5BIT_MASK      0x0000001F
301 #define ACPI_6BIT_MASK      0x0000003F
302 #define ACPI_7BIT_MASK      0x0000007F
303 #define ACPI_8BIT_MASK      0x000000FF
304 #define ACPI_16BIT_MASK     0x0000FFFF
305 #define ACPI_24BIT_MASK     0x00FFFFFF

307 /* Macros to extract flag bits from position zero */

309 #define ACPI_GET_1BIT_FLAG(Value)              ((Value) & ACPI_1BIT_MASK)
310 #define ACPI_GET_2BIT_FLAG(Value)              ((Value) & ACPI_2BIT_MASK)
311 #define ACPI_GET_3BIT_FLAG(Value)              ((Value) & ACPI_3BIT_MASK)
312 #define ACPI_GET_4BIT_FLAG(Value)              ((Value) & ACPI_4BIT_MASK)

314 /* Macros to extract flag bits from position one and above */

316 #define ACPI_EXTRACT_1BIT_FLAG(Field, Position) (ACPI_GET_1BIT_FLAG ((Field)
317 #define ACPI_EXTRACT_2BIT_FLAG(Field, Position) (ACPI_GET_2BIT_FLAG ((Field)
318 #define ACPI_EXTRACT_3BIT_FLAG(Field, Position) (ACPI_GET_3BIT_FLAG ((Field)
319 #define ACPI_EXTRACT_4BIT_FLAG(Field, Position) (ACPI_GET_4BIT_FLAG ((Field)

321 /* ACPI Pathname helpers */

```

```

323 #define ACPI_IS_ROOT_PREFIX(c)                ((c) == (UINT8) 0x5C) /* Backslash */
324 #define ACPI_IS_PARENT_PREFIX(c)              ((c) == (UINT8) 0x5E) /* Carat */
325 #define ACPI_IS_PATH_SEPARATOR(c)             ((c) == (UINT8) 0x2E) /* Period (dot) */

327 /*
328  * An object of type ACPI_NAMESPACE_NODE can appear in some contexts
329  * where a pointer to an object of type ACPI_OPERAND_OBJECT can also
330  * appear. This macro is used to distinguish them.
331  */
332 #define ACPI_DESCRIPTOR_PTR(d)                 ((ACPI_DESCRIPTOR *) (d))
333 /*
334 #define ACPI_GET_DESCRIPTOR_PTR(d)             (((ACPI_DESCRIPTOR *) (d)) -> Commo
335 #define ACPI_SET_DESCRIPTOR_PTR(d, p)         (((ACPI_DESCRIPTOR *) (d)) -> Commo
336 #define ACPI_GET_DESCRIPTOR_TYPE(d)           (((ACPI_DESCRIPTOR *) (d)) -> Commo
337 #define ACPI_SET_DESCRIPTOR_TYPE(d, t)        (((ACPI_DESCRIPTOR *) (d)) -> Commo

339 /*
340  * Macros for the master AML opcode table
341  */
342 #if defined(ACPI_DISASSEMBLER) || defined(ACPI_DEBUG_OUTPUT)
343 #define ACPI_OP(Name, PArgs, IArgs, ObjType, Class, Type, Flags) \
344     {Name, (UINT32)(PArgs), (UINT32)(IArgs), (UINT32)(Flags), ObjType, Class, Ty
345 #else
346 #define ACPI_OP(Name, PArgs, IArgs, ObjType, Class, Type, Flags) \
347     {(UINT32)(PArgs), (UINT32)(IArgs), (UINT32)(Flags), ObjType, Class, Type}
348 #endif

350 #define ARG_TYPE_WIDTH      5
351 #define ARG_1(x)            ((UINT32)(x))
352 #define ARG_2(x)            ((UINT32)(x) << (1 * ARG_TYPE_WIDTH))
353 #define ARG_3(x)            ((UINT32)(x) << (2 * ARG_TYPE_WIDTH))
354 #define ARG_4(x)            ((UINT32)(x) << (3 * ARG_TYPE_WIDTH))
355 #define ARG_5(x)            ((UINT32)(x) << (4 * ARG_TYPE_WIDTH))
356 #define ARG_6(x)            ((UINT32)(x) << (5 * ARG_TYPE_WIDTH))

358 #define ARG1_LIST1(a)        (ARG_1(a))
359 #define ARG1_LIST2(a, b)     (ARG_1(b) | ARG_2(a))
360 #define ARG1_LIST3(a, b, c)  (ARG_1(c) | ARG_2(b) | ARG_3(a))
361 #define ARG1_LIST4(a, b, c, d) (ARG_1(d) | ARG_2(c) | ARG_3(b) | ARG_4(a))
362 #define ARG1_LIST5(a, b, c, d, e) (ARG_1(e) | ARG_2(d) | ARG_3(c) | ARG_4(b) | ARG
363 #define ARG1_LIST6(a, b, c, d, e, f) (ARG_1(f) | ARG_2(e) | ARG_3(d) | ARG_4(c) | ARG

365 #define ARGP_LIST1(a)        (ARG_1(a))
366 #define ARGP_LIST2(a, b)     (ARG_1(a) | ARG_2(b))
367 #define ARGP_LIST3(a, b, c)  (ARG_1(a) | ARG_2(b) | ARG_3(c))
368 #define ARGP_LIST4(a, b, c, d) (ARG_1(a) | ARG_2(b) | ARG_3(c) | ARG_4(d))
369 #define ARGP_LIST5(a, b, c, d, e) (ARG_1(a) | ARG_2(b) | ARG_3(c) | ARG_4(d) | ARG
370 #define ARGP_LIST6(a, b, c, d, e, f) (ARG_1(a) | ARG_2(b) | ARG_3(c) | ARG_4(d) | ARG

372 #define GET_CURRENT_ARG_TYPE(List)              (List & ((UINT32) 0x1F))
373 #define INCREMENT_ARG_LIST(List)                (List >>= ((UINT32) ARG_TYPE_WIDTH))

375 /*
376  * Ascii error messages can be configured out
377  */
378 #ifndef ACPI_NO_ERROR_MESSAGES
379 /*
380  * Error reporting. Callers module and line number are inserted by AE_INFO,
381  * the plist contains a set of parens to allow variable-length lists.
382  * These macros are used for both the debug and non-debug versions of the code.
383  */
384 #define ACPI_ERROR_NAMESPACE(s, e)              AcpiUtNamespaceError (AE_INFO, s, e)
385 #define ACPI_ERROR_METHOD(s, n, p, e)          AcpiUtMethodError (AE_INFO, s, n, p,
386 #define ACPI_WARN_PREDEFINED(plist)            AcpiUtPredefinedWarning plist
387 #define ACPI_INFO_PREDEFINED(plist)            AcpiUtPredefinedInfo plist
388 #define ACPI_BIOS_ERROR_PREDEFINED(plist)      AcpiUtPredefinedBiosError plist

```

```
390 #else

392 /* No error messages */

394 #define ACPI_ERROR_NAMESPACE(s, e)
395 #define ACPI_ERROR_METHOD(s, n, p, e)
396 #define ACPI_WARN_PREDEFINED(plist)
397 #define ACPI_INFO_PREDEFINED(plist)
398 #define ACPI_BIOS_ERROR_PREDEFINED(plist)

400 #endif /* ACPI_NO_ERROR_MESSAGES */

402 #if (!ACPI_REDUCED_HARDWARE)
403 #define ACPI_HW_OPTIONAL_FUNCTION(addr)    addr
404 #else
405 #define ACPI_HW_OPTIONAL_FUNCTION(addr)    NULL
406 #endif

409 /*
410  * Some code only gets executed when the debugger is built in.
411  * Note that this is entirely independent of whether the
412  * DEBUG_PRINT stuff (set by ACPI_DEBUG_OUTPUT) is on, or not.
413  */
414 #ifdef ACPI_DEBUGGER
415 #define ACPI_DEBUGGER_EXEC(a)              a
416 #else
417 #define ACPI_DEBUGGER_EXEC(a)
418 #endif

421 /*
422  * Macros used for ACPICA utilities only
423  */

425 /* Generate a UUID */

427 #define ACPI_INIT_UUID(a, b, c, d0, d1, d2, d3, d4, d5, d6, d7) \
428     (a) & 0xFF, ((a) >> 8) & 0xFF, ((a) >> 16) & 0xFF, ((a) >> 24) & 0xFF, \
429     (b) & 0xFF, ((b) >> 8) & 0xFF, \
430     (c) & 0xFF, ((c) >> 8) & 0xFF, \
431     (d0), (d1), (d2), (d3), (d4), (d5), (d6), (d7)

433 #define ACPI_IS_OCTAL_DIGIT(d)              (((char)(d) >= '0') && ((char)(d) <=

436 #endif /* ACMACROS_H */
```

new/usr/src/common/acpica/include/acnames.h

1

```
*****
3582 Fri Jan 17 07:30:58 2014
new/usr/src/common/acpica/include/acnames.h
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Name: acnames.h - Global names and strings
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #ifndef __ACNAMES_H__
46 #define __ACNAMES_H__
47
48 /* Method names - these methods can appear anywhere in the namespace */
49
50 #define METHOD_NAME_HID      "_HID"
51 #define METHOD_NAME_CID      "_CID"
52 #define METHOD_NAME_UID      "_UID"
53 #define METHOD_NAME_ADR      "_ADR"
54 #define METHOD_NAME_AEI      "_AEI"
55 #define METHOD_NAME_INI      "_INI"
56 #define METHOD_NAME_STA      "_STA"
57 #define METHOD_NAME_REG      "_REG"
58 #define METHOD_NAME_SEG      "_SEG"
```

new/usr/src/common/acpica/include/acnames.h

2

```
51 #define METHOD_NAME_BBN      "_BBN"
52 #define METHOD_NAME_CBA      "_CBA"
53 #define METHOD_NAME_CID      "_CID"
54 #define METHOD_NAME_PRT      "_PRT"
55 #define METHOD_NAME_CRS      "_CRS"
56 #define METHOD_NAME_HID      "_HID"
57 #define METHOD_NAME_INI      "_INI"
58 #define METHOD_NAME_PLD      "_PLD"
59 #define METHOD_NAME_PRS      "_PRS"
60 #define METHOD_NAME_PRT      "_PRT"
61 #define METHOD_NAME_PRW      "_PRW"
62 #define METHOD_NAME_REG      "_REG"
63 #define METHOD_NAME_SB       "_SB"
64 #define METHOD_NAME_SEG      "_SEG"
65 #define METHOD_NAME_SRS      "_SRS"
66 #define METHOD_NAME_STA      "_STA"
67 #define METHOD_NAME_SUB      "_SUB"
68 #define METHOD_NAME_UID      "_UID"
69
70 /* Method names - these methods must appear at the namespace root */
71
72 #define METHOD_PATHNAME_PTS   "\\_PTS"
73 #define METHOD_PATHNAME_SST   "\\_SI._SST"
74 #define METHOD_PATHNAME_WAK   "\\_WAK"
75 #define METHOD_NAME_BFS       "\\_BFS"
76 #define METHOD_NAME_GTS       "\\_GTS"
77 #define METHOD_NAME_PTS       "\\_PTS"
78 #define METHOD_NAME_SST       "\\_SI._SST"
79 #define METHOD_NAME_WAK       "\\_WAK"
80
81 /* Definitions of the predefined namespace names */
82
83 #define ACPI_UNKNOWN_NAME     (UINT32) 0x3F3F3F3F /* Unknown name is "???"
84 #define ACPI_ROOT_NAME        (UINT32) 0x5F5F5F5C /* Root name is "_\"
85
86 #define ACPI_PREFIX_MIXED     (UINT32) 0x69706341 /* "Acpi" */
87 #define ACPI_PREFIX_LOWER     (UINT32) 0x69706361 /* "acpi" */
88
89 #define ACPI_NS_ROOT_PATH     "\\\"
90 #define ACPI_NS_SYSTEM_BUS    "_SB_"
91
92 #endif /* __ACNAMES_H__ */
```



new/usr/src/common/acpica/include/acnamesp.h

1

```
*****
13368 Fri Jan 17 07:30:58 2014
new/usr/src/common/acpica/include/acnamesp.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acnamesp.h - Namespace subcomponent prototypes and defines
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #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 */
```

new/usr/src/common/acpica/include/acnamesp.h

2

```
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,
122     ACPI_WALK_CALLBACK PreOrderVisit,
123     ACPI_WALK_CALLBACK PostOrderVisit,
124     void *Context,
```

new/usr/src/common/acpica/include/acnamesp.h

```
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
```

3

new/usr/src/common/acpica/include/acnamesp.h

```
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);
```

4

```

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     DisplayType,
271     UINT32           MaxDepth,
272     ACPI_OWNER_ID    OwnerId,
273     ACPI_HANDLE      StartHandle);

275 void
276 AcpiNsDumpObjectPaths (
277     ACPI_OBJECT_TYPE Type,
278     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
298  * nspredef - Support for predefined/reserved names
299  */
300 void
301 AcpiNsCheckArgumentCount (
302     char            *Pathname,
303     ACPI_STATUS
304     AcpiNsCheckPredefinedNames (
305         ACPI_NAMESPACE_NODE *Node,
306         UINT32             UserParamCount,
307         const ACPI_PREDEFINED_INFO *Info);
308     ACPI_STATUS      ReturnStatus,
309     ACPI_OPERAND_OBJECT **ReturnObject);

269 const ACPI_PREDEFINED_INFO *
270 AcpiNsCheckForPredefinedName (
271     ACPI_NAMESPACE_NODE *Node);

306 void
307 AcpiNsCheckAcpiCompliance (
308     AcpiNsCheckParameterCount (
309         char            *Pathname,
310         ACPI_NAMESPACE_NODE *Node,
311         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,
430     AcpiNsRepairObject (
431         ACPI_PREDEFINED_DATA *Data,
432         UINT32 ExpectedBtypes,
433         UINT32 PackageIndex,
434         ACPI_OPERAND_OBJECT **ReturnObjectPtr);

436 ACPI_STATUS
437 AcpiNsWrapWithPackage (
438     ACPI_EVALUATE_INFO *Info,
439     ACPI_OPERAND_OBJECT *OriginalObject,
440     AcpiNsRepairPackageList (
441         ACPI_PREDEFINED_DATA *Data,

```

```

438     ACPI_OPERAND_OBJECT **ObjDescPtr);

440 ACPI_STATUS
441 AcpiNsRepairNullElement (
442     ACPI_EVALUATE_INFO *Info,
443     ACPI_PREDEFINED_DATA *Data,
444     UINT32 ExpectedBtypes,
445     UINT32 PackageIndex,
446     ACPI_OPERAND_OBJECT **ReturnObjectPtr);

447 void
448 AcpiNsRemoveNullElements (
449     ACPI_EVALUATE_INFO *Info,
450     ACPI_PREDEFINED_DATA *Data,
451     UINT8 PackageType,
452     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,
461     ACPI_PREDEFINED_DATA *Data,
462     ACPI_NAMESPACE_NODE *Node,
463     ACPI_STATUS ValidateStatus,
464     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  */
497 BOOLEAN
498 AcpiNsValidRootPrefix (
499     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__ */
```

new/usr/src/common/acpica/include/acobject.h

1

\*\*\*\*\*

21194 Fri Jan 17 07:30:58 2014

new/usr/src/common/acpica/include/acobject.h

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Name: acobject.h - Definition of ACPI_OPERAND_OBJECT (Internal object only)
4 *
5 *****/
```

```
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef _ACOBJECT_H
45 #define _ACOBJECT_H
```

```
47 /* acpisrc:StructDefs -- for acpisrc conversion */
```

```
50 /*
51 * The ACPI_OPERAND_OBJECT is used to pass AML operands from the dispatcher
52 * to the interpreter, and to keep track of the various handlers such as
53 * address space handlers and notify handlers. The object is a constant
54 * size in order to allow it to be cached and reused.
55 *
56 * Note: The object is optimized to be aligned and will not work if it is
57 * byte-packed.
```

new/usr/src/common/acpica/include/acobject.h

2

```
58 */
59 #if ACPI_MACHINE_WIDTH == 64
60 #pragma pack(8)
61 #else
62 #pragma pack(4)
63 #endif
```

```
65 /*****
66 *
67 * Common Descriptors
68 *
69 *****/
```

```
71 /*
72 * Common area for all objects.
73 *
74 * DescriptorType is used to differentiate between internal descriptors, and
75 * must be in the same place across all descriptors
76 *
77 * Note: The DescriptorType and Type fields must appear in the identical
78 * position in both the ACPI_NAMESPACE_NODE and ACPI_OPERAND_OBJECT
79 * structures.
80 */
81 #define ACPI_OBJECT_COMMON_HEADER \
82     union acpi_operand_object {
83         uint8_t      *NextObject;      /* Objects linked to par
84         uint8_t      DescriptorType;   /* To differentiate vari
85         uint8_t      Type;              /* ACPI_OBJECT_TYPE */
86         uint16_t     ReferenceCount;    /* For object deletion m
87         uint8_t      Flags;
88     }
89     /* Note: There are 3 bytes available here before the
90     * next natural alignment boundary (for both 32/64 cases)
91     */
```

```
92 /* Values for Flag byte above */
```

```
94 #define AOPOBJ_AML_CONSTANT      0x01    /* Integer is an AML constant */
95 #define AOPOBJ_STATIC_POINTER    0x02    /* Data is part of an ACPI table, do
96 #define AOPOBJ_DATA_VALID        0x04    /* Object is initialized and data is
97 #define AOPOBJ_DATA_VALID        0x04    /* Object is initialized and data is
98 #define AOPOBJ_OBJECT_INITIALIZED 0x08    /* Region is initialized, _REG was r
99 #define AOPOBJ_SETUP_COMPLETE   0x10    /* Region setup is complete */
100 #define AOPOBJ_INVALID           0x20    /* Host OS won't allow a Region addr
```

```
102 /*****
103 *
104 * Basic data types
105 *
106 *****/
```

```
108 typedef struct acpi_object_common
109 {
110     ACPI_OBJECT_COMMON_HEADER
```

```
112 } ACPI_OBJECT_COMMON;
113 unchanged_portion_omitted
```

```
124 /*
125 * Note: The String and Buffer object must be identical through the
126 * pointer and length elements. There is code that depends on this.
127 * Note: The String and Buffer object must be identical through the Pointer
128 * and length elements. There is code that depends on this.
```

```
128 * Fields common to both Strings and Buffers
```

```

129 */
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
408     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
```



new/usr/src/common/acpica/include/acopcode.h

1

```
*****
22618 Fri Jan 17 07:30:59 2014
new/usr/src/common/acpica/include/acopcode.h
update to acpica-unix2-20140114
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 - 2014, 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 MERCHANTABILITY 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 #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
```

new/usr/src/common/acpica/include/acopcode.h

2

```
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 ARG_P_ACCESSFIELD_OP ARG_P_LIST1 (ARG_P_NAMESTRING)
72 #define ARG_P_ACQUIRE_OP ARG_P_LIST2 (ARG_P_SUPERNAME, ARG_P_WORDDA
73 #define ARG_P_ADD_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
74 #define ARG_P_ALIAS_OP ARG_P_LIST2 (ARG_P_NAMESTRING, ARG_P_NAME)
75 #define ARG_P_ARG0 ARG_NONE
76 #define ARG_P_ARG1 ARG_NONE
77 #define ARG_P_ARG2 ARG_NONE
78 #define ARG_P_ARG3 ARG_NONE
79 #define ARG_P_ARG4 ARG_NONE
80 #define ARG_P_ARG5 ARG_NONE
81 #define ARG_P_ARG6 ARG_NONE
82 #define ARG_P_BANK_FIELD_OP ARG_P_LIST6 (ARG_P_PKGLNGTH, ARG_P_NAMEST
83 #define ARG_P_BIT_AND_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
84 #define ARG_P_BIT_NAND_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
85 #define ARG_P_BIT_NOR_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
86 #define ARG_P_BIT_NOT_OP ARG_P_LIST2 (ARG_P_TERMARG, ARG_P_TARGET
87 #define ARG_P_BIT_OR_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
88 #define ARG_P_BIT_XOR_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
89 #define ARG_P_BREAK_OP ARG_NONE
90 #define ARG_P_BREAK_POINT_OP ARG_NONE
91 #define ARG_P_BUFFER_OP ARG_P_LIST3 (ARG_P_PKGLNGTH, ARG_P_TERMAR
92 #define ARG_P_BYTE_OP ARG_P_LIST1 (ARG_P_BYTEDATA)
93 #define ARG_P_BYTelist_OP ARG_P_LIST1 (ARG_P_NAMESTRING)
94 #define ARG_P_CONCAT_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
95 #define ARG_P_CONCAT_RES_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
96 #define ARG_P_COND_REF_OF_OP ARG_P_LIST2 (ARG_P_SUPERNAME, ARG_P_SUPERN
97 #define ARG_P_CONNECTFIELD_OP ARG_P_LIST1 (ARG_P_NAMESTRING)
98 #define ARG_P_CONTINUE_OP ARG_NONE
99 #define ARG_P_COPY_OP ARG_P_LIST2 (ARG_P_TERMARG, ARG_P_SIMPLE
100 #define ARG_P_CREATE_BIT_FIELD_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
101 #define ARG_P_CREATE_BYTE_FIELD_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
102 #define ARG_P_CREATE_DWORD_FIELD_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
103 #define ARG_P_CREATE_FIELD_OP ARG_P_LIST4 (ARG_P_TERMARG, ARG_P_TERMAR
104 #define ARG_P_CREATE_QWORD_FIELD_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
105 #define ARG_P_CREATE_WORD_FIELD_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
106 #define ARG_P_DATA_REGION_OP ARG_P_LIST4 (ARG_P_NAME, ARG_P_TERMAR
107 #define ARG_P_DEBUG_OP ARG_NONE
108 #define ARG_P_DECREMENT_OP ARG_P_LIST1 (ARG_P_SUPERNAME)
109 #define ARG_P_DEREF_OF_OP ARG_P_LIST1 (ARG_P_TERMARG)
110 #define ARG_P_DEVICE_OP ARG_P_LIST3 (ARG_P_PKGLNGTH, ARG_P_NAME,
111 #define ARG_P_DIVIDE_OP ARG_P_LIST4 (ARG_P_TERMARG, ARG_P_TERMAR
112 #define ARG_P_DWORD_OP ARG_P_LIST1 (ARG_P_DWORDDDATA)
113 #define ARG_P_ELSE_OP ARG_P_LIST2 (ARG_P_PKGLNGTH, ARG_P_TERMLI
114 #define ARG_P_EVENT_OP ARG_P_LIST1 (ARG_P_NAME)
115 #define ARG_P_FATAL_OP ARG_P_LIST3 (ARG_P_BYTEDATA, ARG_P_DWORDDD
116 #define ARG_P_FIELD_OP ARG_P_LIST4 (ARG_P_PKGLNGTH, ARG_P_NAMEST
117 #define ARG_P_FIND_SET_LEFT_BIT_OP ARG_P_LIST2 (ARG_P_TERMARG, ARG_P_TARGET
118 #define ARG_P_FIND_SET_RIGHT_BIT_OP ARG_P_LIST2 (ARG_P_TERMARG, ARG_P_TARGET
119 #define ARG_P_FROM_BCD_OP ARG_P_LIST2 (ARG_P_TERMARG, ARG_P_TARGET
120 #define ARG_P_IF_OP ARG_P_LIST3 (ARG_P_PKGLNGTH, ARG_P_TERMAR
121 #define ARG_P_INCREMENT_OP ARG_P_LIST1 (ARG_P_SUPERNAME)
122 #define ARG_P_INDEX_FIELD_OP ARG_P_LIST5 (ARG_P_PKGLNGTH, ARG_P_NAMEST
123 #define ARG_P_INDEX_OP ARG_P_LIST3 (ARG_P_TERMARG, ARG_P_TERMAR
124 #define ARG_P_LAND_OP ARG_P_LIST2 (ARG_P_TERMARG, ARG_P_TERMAR
```

```

125 #define ARGV_LEQUAL_OP          ARGV_LIST2 (ARGV_TERMARG,   ARGV_TERMARG
126 #define ARGV_LGREATER_OP        ARGV_LIST2 (ARGV_TERMARG,   ARGV_TERMARG
127 #define ARGV_LGREATEREQUAL_OP   ARGV_LIST2 (ARGV_TERMARG,   ARGV_TERMARG
128 #define ARGV_LLESS_OP           ARGV_LIST2 (ARGV_TERMARG,   ARGV_TERMARG
129 #define ARGV_LLESSEQUAL_OP      ARGV_LIST2 (ARGV_TERMARG,   ARGV_TERMARG
130 #define ARGV_LNOT_OP            ARGV_LIST1 (ARGV_TERMARG)
131 #define ARGV_LNOTEQUAL_OP       ARGV_LIST2 (ARGV_TERMARG,   ARGV_TERMARG
132 #define ARGV_LOAD_OP            ARGV_LIST2 (ARGV_NAMESTRING, ARGV_SUPERN
133 #define ARGV_LOAD_TABLE_OP      ARGV_LIST6 (ARGV_TERMARG,   ARGV_TERMARG
134 #define ARGV_LOCAL0             ARGV_NONE
135 #define ARGV_LOCAL1             ARGV_NONE
136 #define ARGV_LOCAL2             ARGV_NONE
137 #define ARGV_LOCAL3             ARGV_NONE
138 #define ARGV_LOCAL4             ARGV_NONE
139 #define ARGV_LOCAL5             ARGV_NONE
140 #define ARGV_LOCAL6             ARGV_NONE
141 #define ARGV_LOCAL7             ARGV_NONE
142 #define ARGV_LOR_OP             ARGV_LIST2 (ARGV_TERMARG,   ARGV_TERMARG
143 #define ARGV_MATCH_OP           ARGV_LIST6 (ARGV_TERMARG,   ARGV_BYTEDA
144 #define ARGV_METHOD_OP          ARGV_LIST4 (ARGV_PKGLLENGTH, ARGV_NAME,
145 #define ARGV_METHODCALL_OP      ARGV_LIST1 (ARGV_NAMESTRING)
146 #define ARGV_MID_OP             ARGV_LIST4 (ARGV_TERMARG,   ARGV_TERMARG
147 #define ARGV_MOD_OP             ARGV_LIST3 (ARGV_TERMARG,   ARGV_TERMARG
148 #define ARGV_MULTIPLY_OP        ARGV_LIST3 (ARGV_TERMARG,   ARGV_TERMARG
149 #define ARGV_MUTEX_OP           ARGV_LIST2 (ARGV_NAME,      ARGV_BYTEDA
150 #define ARGV_NAME_OP            ARGV_LIST2 (ARGV_NAME,      ARGV_DATAOB
151 #define ARGV_NAMEDFIELD_OP      ARGV_LIST1 (ARGV_NAMESTRING)
152 #define ARGV_NAMEPATH_OP        ARGV_LIST1 (ARGV_NAMESTRING)
153 #define ARGV_NOOP_OP            ARGV_NONE
154 #define ARGV_NOTIFY_OP          ARGV_LIST2 (ARGV_SUPERNAME, ARGV_TERMARG
155 #define ARGV_ONE_OP             ARGV_NONE
156 #define ARGV_ONES_OP            ARGV_NONE
157 #define ARGV_PACKAGE_OP         ARGV_LIST3 (ARGV_PKGLLENGTH, ARGV_BYTEDA
158 #define ARGV_POWER_RES_OP       ARGV_LIST5 (ARGV_PKGLLENGTH, ARGV_NAME,
159 #define ARGV_PROCESSOR_OP       ARGV_LIST6 (ARGV_PKGLLENGTH, ARGV_NAME,
160 #define ARGV_QWORD_OP           ARGV_LIST1 (ARGV_QWORDDATA)
161 #define ARGV_REF_OF_OP          ARGV_LIST1 (ARGV_SUPERNAME)
162 #define ARGV_REGION_OP          ARGV_LIST4 (ARGV_NAME,      ARGV_BYTEDA
163 #define ARGV_RELEASE_OP         ARGV_LIST1 (ARGV_SUPERNAME)
164 #define ARGV_RESERVEDFIELD_OP   ARGV_LIST1 (ARGV_NAMESTRING)
165 #define ARGV_RESET_OP           ARGV_LIST1 (ARGV_SUPERNAME)
166 #define ARGV_RETURN_OP          ARGV_LIST1 (ARGV_TERMARG)
167 #define ARGV_REVISION_OP        ARGV_NONE
168 #define ARGV_SCOPE_OP           ARGV_LIST3 (ARGV_PKGLLENGTH, ARGV_NAME,
169 #define ARGV_SERIALFIELD_OP     ARGV_LIST1 (ARGV_NAMESTRING)
170 #define ARGV_SHIFT_LEFT_OP      ARGV_LIST3 (ARGV_TERMARG,   ARGV_TERMARG
171 #define ARGV_SHIFT_RIGHT_OP     ARGV_LIST3 (ARGV_TERMARG,   ARGV_TERMARG
172 #define ARGV_SIGNAL_OP          ARGV_LIST1 (ARGV_SUPERNAME)
173 #define ARGV_SIZE_OF_OP         ARGV_LIST1 (ARGV_SUPERNAME)
174 #define ARGV_SLEEP_OP           ARGV_LIST1 (ARGV_TERMARG)
175 #define ARGV_STALL_OP           ARGV_LIST1 (ARGV_TERMARG)
176 #define ARGV_STATICSTRING_OP    ARGV_LIST1 (ARGV_NAMESTRING)
177 #define ARGV_STORE_OP           ARGV_LIST2 (ARGV_TERMARG,   ARGV_SUPERN
178 #define ARGV_STRING_OP          ARGV_LIST1 (ARGV_CHARLIST)
179 #define ARGV_SUBTRACT_OP        ARGV_LIST3 (ARGV_TERMARG,   ARGV_TERMARG
180 #define ARGV_THERMAL_ZONE_OP    ARGV_LIST3 (ARGV_PKGLLENGTH, ARGV_NAME,
181 #define ARGV_TIMER_OP           ARGV_NONE
182 #define ARGV_TO_BCD_OP          ARGV_LIST2 (ARGV_TERMARG,   ARGV_TARGET
183 #define ARGV_TO_BUFFER_OP       ARGV_LIST2 (ARGV_TERMARG,   ARGV_TARGET
184 #define ARGV_TO_DEC_STR_OP      ARGV_LIST2 (ARGV_TERMARG,   ARGV_TARGET
185 #define ARGV_TO_HEX_STR_OP      ARGV_LIST2 (ARGV_TERMARG,   ARGV_TARGET
186 #define ARGV_TO_INTEGER_OP      ARGV_LIST2 (ARGV_TERMARG,   ARGV_TARGET
187 #define ARGV_TO_STRING_OP       ARGV_LIST3 (ARGV_TERMARG,   ARGV_TERMARG
188 #define ARGV_TYPE_OP            ARGV_LIST1 (ARGV_SUPERNAME)
189 #define ARGV_UNLOAD_OP          ARGV_LIST1 (ARGV_SUPERNAME)
190 #define ARGV_VAR_PACKAGE_OP     ARGV_LIST3 (ARGV_PKGLLENGTH, ARGV_TERMARG

```

```

191 #define ARGV_WAIT_OP            ARGV_LIST2 (ARGV_SUPERNAME, ARGV_TERMARG
192 #define ARGV_WHILE_OP           ARGV_LIST3 (ARGV_PKGLLENGTH, ARGV_TERMARG
193 #define ARGV_WORD_OP            ARGV_LIST1 (ARGV_WORDDATA)
194 #define ARGV_ZERO_OP            ARGV_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 ARGV_ACCESSFIELD_OP      ARGV_INVALID_OPCODE
205 #define ARGV_ACQUIRE_OP        ARGV_LIST2 (ARGV_MUTEX,      ARGV_INTEGE
206 #define ARGV_ADD_OP             ARGV_LIST3 (ARGV_INTEGER,   ARGV_INTEGE
207 #define ARGV_ALIAS_OP           ARGV_INVALID_OPCODE
208 #define ARGV_ARG0               ARGV_NONE
209 #define ARGV_ARG1               ARGV_NONE
210 #define ARGV_ARG2               ARGV_NONE
211 #define ARGV_ARG3               ARGV_NONE
212 #define ARGV_ARG4               ARGV_NONE
213 #define ARGV_ARG5               ARGV_NONE
214 #define ARGV_ARG6               ARGV_NONE
215 #define ARGV_BANK_FIELD_OP      ARGV_INVALID_OPCODE
216 #define ARGV_BIT_AND_OP         ARGV_LIST3 (ARGV_INTEGER,   ARGV_INTEGE
217 #define ARGV_BIT_NAND_OP        ARGV_LIST3 (ARGV_INTEGER,   ARGV_INTEGE
218 #define ARGV_BIT_NOR_OP         ARGV_LIST3 (ARGV_INTEGER,   ARGV_INTEGE
219 #define ARGV_BIT_NOT_OP         ARGV_LIST2 (ARGV_INTEGER,   ARGV_TARGET
220 #define ARGV_BIT_OR_OP          ARGV_LIST3 (ARGV_INTEGER,   ARGV_INTEGE
221 #define ARGV_BIT_XOR_OP         ARGV_LIST3 (ARGV_INTEGER,   ARGV_INTEGE
222 #define ARGV_BREAK_OP           ARGV_NONE
223 #define ARGV_BREAK_POINT_OP     ARGV_NONE
224 #define ARGV_BUFFER_OP          ARGV_LIST1 (ARGV_INTEGER)
225 #define ARGV_BYTE_OP            ARGV_INVALID_OPCODE
226 #define ARGV_BYTelist_OP        ARGV_INVALID_OPCODE
227 #define ARGV_CONCAT_OP          ARGV_LIST3 (ARGV_COMPUTEDATA, ARGV_COMPUT
228 #define ARGV_CONCAT_RES_OP      ARGV_LIST3 (ARGV_BUFFER,   ARGV_BUFFER
229 #define ARGV_COND_REF_OF_OP     ARGV_LIST2 (ARGV_OBJECT_REF, ARGV_TARGET
230 #define ARGV_CONNECTFIELD_OP    ARGV_INVALID_OPCODE
231 #define ARGV_CONTINUE_OP        ARGV_INVALID_OPCODE
232 #define ARGV_COPY_OP            ARGV_LIST2 (ARGV_ANYTYPE,   ARGV_SIMPLE
233 #define ARGV_CREATE_BIT_FIELD_OP ARGV_LIST3 (ARGV_BUFFER,   ARGV_INTEGE
234 #define ARGV_CREATE_BYTE_FIELD_OP ARGV_LIST3 (ARGV_BUFFER,   ARGV_INTEGE
235 #define ARGV_CREATE_DWORD_FIELD_OP ARGV_LIST3 (ARGV_BUFFER,   ARGV_INTEGE
236 #define ARGV_CREATE_FIELD_OP    ARGV_LIST4 (ARGV_BUFFER,   ARGV_INTEGE
237 #define ARGV_CREATE_QWORD_FIELD_OP ARGV_LIST3 (ARGV_BUFFER,   ARGV_INTEGE
238 #define ARGV_CREATE_WORD_FIELD_OP ARGV_LIST3 (ARGV_BUFFER,   ARGV_INTEGE
239 #define ARGV_DATA_REGION_OP     ARGV_LIST3 (ARGV_STRING,   ARGV_STRING
240 #define ARGV_DEBUG_OP           ARGV_NONE
241 #define ARGV_DECREMENT_OP       ARGV_LIST1 (ARGV_TARGETREF)
242 #define ARGV_DEREF_OF_OP        ARGV_LIST1 (ARGV_REF_OR_STRING)
243 #define ARGV_DEVICE_OP          ARGV_INVALID_OPCODE
244 #define ARGV_DIVIDE_OP          ARGV_LIST4 (ARGV_INTEGER,   ARGV_INTEGE
245 #define ARGV_DWORD_OP           ARGV_INVALID_OPCODE
246 #define ARGV_ELSE_OP            ARGV_INVALID_OPCODE
247 #define ARGV_EVENT_OP           ARGV_INVALID_OPCODE
248 #define ARGV_FATAL_OP           ARGV_LIST3 (ARGV_INTEGER,   ARGV_INTEGE
249 #define ARGV_FIELD_OP           ARGV_INVALID_OPCODE
250 #define ARGV_FIND_SET_LEFT_BIT_OP ARGV_LIST2 (ARGV_INTEGER,   ARGV_TARGET
251 #define ARGV_FIND_SET_RIGHT_BIT_OP ARGV_LIST2 (ARGV_INTEGER,   ARGV_TARGET
252 #define ARGV_FROM_BCD_OP        ARGV_LIST2 (ARGV_INTEGER,   ARGV_FIXED_
253 #define ARGV_IF_OP              ARGV_INVALID_OPCODE
254 #define ARGV_INCREMENT_OP       ARGV_LIST1 (ARGV_TARGETREF)
255 #define ARGV_INDEX_FIELD_OP     ARGV_INVALID_OPCODE
256 #define ARGV_INDEX_OP           ARGV_LIST3 (ARGV_COMPLEXOBJ, ARGV_INTEGE

```

```

257 #define ARG1 LAND_OP          ARG1_LIST2 (ARG1_INTEGER, ARG1_INTEGE
258 #define ARG1 LEQUAL_OP       ARG1_LIST2 (ARG1_COMPUTEDATA, ARG1_COMPUT
259 #define ARG1 LGREATER_OP      ARG1_LIST2 (ARG1_COMPUTEDATA, ARG1_COMPUT
260 #define ARG1 LGREATEREQUAL_OP ARG1_INVALID_OPCODE
261 #define ARG1 LLESS_OP         ARG1_LIST2 (ARG1_COMPUTEDATA, ARG1_COMPUT
262 #define ARG1 LLESSEQUAL_OP    ARG1_INVALID_OPCODE
263 #define ARG1 LNOT_OP          ARG1_LIST1 (ARG1_INTEGER)
264 #define ARG1 LNOTEQUAL_OP     ARG1_INVALID_OPCODE
265 #define ARG1 LOAD_OP          ARG1_LIST2 (ARG1_REGION_OR_BUFFER, ARG1_T
266 #define ARG1 LOAD_TABLE_OP    ARG1_LIST6 (ARG1_STRING, ARG1_STRING
267 #define ARG1 LOCAL0           ARG_NONE
268 #define ARG1 LOCAL1           ARG_NONE
269 #define ARG1 LOCAL2           ARG_NONE
270 #define ARG1 LOCAL3           ARG_NONE
271 #define ARG1 LOCAL4           ARG_NONE
272 #define ARG1 LOCAL5           ARG_NONE
273 #define ARG1 LOCAL6           ARG_NONE
274 #define ARG1 LOCAL7           ARG_NONE
275 #define ARG1 LOR_OP           ARG1_LIST2 (ARG1_INTEGER, ARG1_INTEGE
276 #define ARG1 MATCH_OP         ARG1_LIST6 (ARG1_PACKAGE, ARG1_INTEGE
277 #define ARG1 METHOD_OP         ARG1_INVALID_OPCODE
278 #define ARG1 METHODCALL_OP     ARG1_INVALID_OPCODE
279 #define ARG1 MID_OP           ARG1_LIST4 (ARG1_BUFFER_OR_STRING, ARG1_I
280 #define ARG1 MOD_OP           ARG1_LIST3 (ARG1_INTEGER, ARG1_INTEGE
281 #define ARG1 MULTIPLY_OP      ARG1_LIST3 (ARG1_INTEGER, ARG1_INTEGE
282 #define ARG1 MUTEX_OP         ARG1_INVALID_OPCODE
283 #define ARG1 NAME_OP          ARG1_INVALID_OPCODE
284 #define ARG1 NAMEDFIELD_OP    ARG1_INVALID_OPCODE
285 #define ARG1 NAMEPATH_OP      ARG1_INVALID_OPCODE
286 #define ARG1 NOOP_OP          ARG_NONE
287 #define ARG1 NOTIFY_OP        ARG1_LIST2 (ARG1_DEVICE_REF, ARG1_INTEGE
288 #define ARG1 ONE_OP           ARG_NONE
289 #define ARG1 ONES_OP          ARG_NONE
290 #define ARG1 PACKAGE_OP       ARG1_LIST1 (ARG1_INTEGER)
291 #define ARG1 POWER_RES_OP     ARG1_INVALID_OPCODE
292 #define ARG1 PROCESSOR_OP     ARG1_INVALID_OPCODE
293 #define ARG1 QWORD_OP         ARG1_INVALID_OPCODE
294 #define ARG1 REF_OF_OP        ARG1_LIST1 (ARG1_OBJECT_REF)
295 #define ARG1 REGION_OP        ARG1_LIST2 (ARG1_INTEGER, ARG1_INTEGE
296 #define ARG1 RELEASE_OP       ARG1_LIST1 (ARG1_MUTEX)
297 #define ARG1 RESERVEDFIELD_OP ARG1_INVALID_OPCODE
298 #define ARG1 RESET_OP         ARG1_LIST1 (ARG1_EVENT)
299 #define ARG1 RETURN_OP        ARG1_INVALID_OPCODE
300 #define ARG1 REVISION_OP      ARG_NONE
301 #define ARG1 SCOPE_OP         ARG1_INVALID_OPCODE
302 #define ARG1 SERIALFIELD_OP   ARG1_INVALID_OPCODE
303 #define ARG1 SHIFT_LEFT_OP    ARG1_LIST3 (ARG1_INTEGER, ARG1_INTEGE
304 #define ARG1 SHIFT_RIGHT_OP   ARG1_LIST3 (ARG1_INTEGER, ARG1_INTEGE
305 #define ARG1 SIGNAL_OP        ARG1_LIST1 (ARG1_EVENT)
306 #define ARG1 SIZE_OF_OP       ARG1_LIST1 (ARG1_DATAOBJECT)
307 #define ARG1 SLEEP_OP         ARG1_LIST1 (ARG1_INTEGER)
308 #define ARG1 STALL_OP         ARG1_LIST1 (ARG1_INTEGER)
309 #define ARG1 STATICSTRING_OP   ARG1_INVALID_OPCODE
310 #define ARG1 STORE_OP         ARG1_LIST2 (ARG1_DATAAREFOBJ, ARG1_TARGET
311 #define ARG1 STRING_OP        ARG1_INVALID_OPCODE
312 #define ARG1 SUBTRACT_OP      ARG1_LIST3 (ARG1_INTEGER, ARG1_INTEGE
313 #define ARG1 THERMAL_ZONE_OP  ARG1_INVALID_OPCODE
314 #define ARG1 TIMER_OP         ARG_NONE
315 #define ARG1 TO_BCD_OP        ARG1_LIST2 (ARG1_INTEGER, ARG1_FIXED_
316 #define ARG1 TO_BUFFER_OP     ARG1_LIST2 (ARG1_COMPUTEDATA, ARG1_FIXED_
317 #define ARG1 TO_DEC_STR_OP    ARG1_LIST2 (ARG1_COMPUTEDATA, ARG1_FIXED_
318 #define ARG1 TO_HEX_STR_OP    ARG1_LIST2 (ARG1_COMPUTEDATA, ARG1_FIXED_
319 #define ARG1 TO_INTEGER_OP    ARG1_LIST2 (ARG1_COMPUTEDATA, ARG1_FIXED_
320 #define ARG1 TO_STRING_OP     ARG1_LIST3 (ARG1_BUFFER, ARG1_INTEGE
321 #define ARG1 TYPE_OP          ARG1_LIST1 (ARG1_ANYTYPE)
322 #define ARG1 UNLOAD_OP        ARG1_LIST1 (ARG1_DDBHANDLE)

```

```

323 #define ARG1_VAR_PACKAGE_OP   ARG1_LIST1 (ARG1_INTEGER)
324 #define ARG1_WAIT_OP          ARG1_LIST2 (ARG1_EVENT, ARG1_INTEGE
325 #define ARG1 WHILE_OP         ARG1_INVALID_OPCODE
326 #define ARG1 WORD_OP          ARG1_INVALID_OPCODE
327 #define ARG1_ZERO_OP          ARG_NONE
329 #endif /* __ACOPCODE_H__ */

```

new/usr/src/common/acpica/include/acoutput.h

1

```
*****
17260 Fri Jan 17 07:30:59 2014
new/usr/src/common/acpica/include/acoutput.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acoutput.h -- debug output
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
```

new/usr/src/common/acpica/include/acoutput.h

2

```
60 #define ACPI_PARSER 0x00000020
61 #define ACPI_DISPATCHER 0x00000040
62 #define ACPI_EXECUTOR 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
128 /* Exceptionally verbose output -- also used in the global "DebugLevel" */
130 #define ACPI_LV_AML_DISASSEMBLE      0x10000000
131 #define ACPI_LV_VERBOSE_INFO          0x20000000
132 #define ACPI_LV_FULL_TABLES           0x40000000
133 #define ACPI_LV_EVENTS                 0x80000000
134 #define ACPI_LV_VERBOSE                0xF0000000
137 /*
138  * Debug level macros that are used in the DEBUG_PRINT macros
139  */
140 #define ACPI_DEBUG_LEVEL(d1)          (UINT32) d1, ACPI_DEBUG_PARAMETERS
142 /*
143  * Exception level -- used in the global "DebugLevel"
144  */
145  * Note: For errors, use the ACPI_ERROR or ACPI_EXCEPTION interfaces.
146  * For warnings, use ACPI_WARNING.
147  */
148 #define ACPI_DB_INIT                  ACPI_DEBUG_LEVEL (ACPI_LV_INIT)
149 #define ACPI_DB_DEBUG_OBJECT           ACPI_DEBUG_LEVEL (ACPI_LV_DEBUG_OBJECT)
150 #define ACPI_DB_INFO                   ACPI_DEBUG_LEVEL (ACPI_LV_INFO)
151 #define ACPI_DB_REPAIR                  ACPI_DEBUG_LEVEL (ACPI_LV_REPAIR)
152 #define ACPI_DB_ALL_EXCEPTIONS         ACPI_DEBUG_LEVEL (ACPI_LV_ALL_EXCEPTIONS)
154 /* Trace level -- also used in the global "DebugLevel" */
156 #define ACPI_DB_INIT_NAMES             ACPI_DEBUG_LEVEL (ACPI_LV_INIT_NAMES)
157 #define ACPI_DB_THREADS                ACPI_DEBUG_LEVEL (ACPI_LV_THREADS)
158 #define ACPI_DB_PARSE                  ACPI_DEBUG_LEVEL (ACPI_LV_PARSE)
159 #define ACPI_DB_DISPATCH               ACPI_DEBUG_LEVEL (ACPI_LV_DISPATCH)
160 #define ACPI_DB_LOAD                   ACPI_DEBUG_LEVEL (ACPI_LV_LOAD)
161 #define ACPI_DB_EXEC                   ACPI_DEBUG_LEVEL (ACPI_LV_EXEC)
162 #define ACPI_DB_NAMES                  ACPI_DEBUG_LEVEL (ACPI_LV_NAMES)
163 #define ACPI_DB_OPREGION               ACPI_DEBUG_LEVEL (ACPI_LV_OPREGION)
164 #define ACPI_DB_BFIELD                 ACPI_DEBUG_LEVEL (ACPI_LV_BFIELD)
165 #define ACPI_DB_TABLES                 ACPI_DEBUG_LEVEL (ACPI_LV_TABLES)
166 #define ACPI_DB_FUNCTIONS              ACPI_DEBUG_LEVEL (ACPI_LV_FUNCTIONS)
167 #define ACPI_DB_OPTIMIZATIONS          ACPI_DEBUG_LEVEL (ACPI_LV_OPTIMIZATIONS)
168 #define ACPI_DB_VALUES                 ACPI_DEBUG_LEVEL (ACPI_LV_VALUES)
169 #define ACPI_DB_OBJECTS                ACPI_DEBUG_LEVEL (ACPI_LV_OBJECTS)
170 #define ACPI_DB_ALLOCATIONS            ACPI_DEBUG_LEVEL (ACPI_LV_ALLOCATIONS)
171 #define ACPI_DB_RESOURCES              ACPI_DEBUG_LEVEL (ACPI_LV_RESOURCES)
172 #define ACPI_DB_IO                     ACPI_DEBUG_LEVEL (ACPI_LV_IO)
173 #define ACPI_DB_INTERRUPTS             ACPI_DEBUG_LEVEL (ACPI_LV_INTERRUPTS)
174 #define ACPI_DB_USER_REQUESTS          ACPI_DEBUG_LEVEL (ACPI_LV_USER_REQUESTS)
175 #define ACPI_DB_PACKAGE                ACPI_DEBUG_LEVEL (ACPI_LV_PACKAGE)
176 #define ACPI_DB_MUTEX                  ACPI_DEBUG_LEVEL (ACPI_LV_MUTEX)
177 #define ACPI_DB_EVENTS                 ACPI_DEBUG_LEVEL (ACPI_LV_EVENTS)
179 #define ACPI_DB_ALL                    ACPI_DEBUG_LEVEL (ACPI_LV_ALL)
181 /* Defaults for DebugLevel, debug and normal */
183 #define ACPI_DEBUG_DEFAULT              (ACPI_LV_INIT | ACPI_LV_DEBUG_OBJECT | ACPI_
184 #define ACPI_NORMAL_DEFAULT            (ACPI_LV_INIT | ACPI_LV_DEBUG_OBJECT | ACPI_
185 #define ACPI_DEBUG_ALL                  (ACPI_LV_AML_DISASSEMBLE | ACPI_LV_ALL_EXCEP
188 #if defined (ACPI_DEBUG_OUTPUT) || !defined (ACPI_NO_ERROR_MESSAGES)
189 /*
190  * The module name is used primarily for error and debug messages.
191  * The __FILE__ macro is not very useful for this, because it

```

```

192  * usually includes the entire pathname to the module making the
193  * debug output difficult to read.
194  */
195 #define ACPI_MODULE_NAME(Name)          static const char ACPI_UNUSED_VAR _AcpiM
196 #else
197 /*
198  * For the no-debug and no-error-msg cases, we must at least define
199  * a null module name.
200  */
201 #define ACPI_MODULE_NAME(Name)
202 #define _AcpiModuleName ""
203 #endif
205 /*
206  * Ascii error messages can be configured out
207  */
208 #ifndef ACPI_NO_ERROR_MESSAGES
209 #define AE_INFO                          _AcpiModuleName, __LINE__
211 /*
212  * Error reporting. Callers module and line number are inserted by AE_INFO,
213  * the plist contains a set of parens to allow variable-length lists.
214  * These macros are used for both the debug and non-debug versions of the code.
215  */
216 #define ACPI_INFO(plist)                 AcpiInfo plist
217 #define ACPI_WARNING(plist)              AcpiWarning plist
218 #define ACPI_EXCEPTION(plist)            AcpiException plist
219 #define ACPI_ERROR(plist)                AcpiError plist
220 #define ACPI_BIOS_WARNING(plist)         AcpiBiosWarning plist
221 #define ACPI_BIOS_ERROR(plist)           AcpiBiosError plist
222 #define ACPI_DEBUG_OBJECT(obj,l,i)       AcpiExDoDebugObject(obj,l,i)
224 #else
226 /* No error messages */
228 #define ACPI_INFO(plist)
229 #define ACPI_WARNING(plist)
230 #define ACPI_EXCEPTION(plist)
231 #define ACPI_ERROR(plist)
232 #define ACPI_BIOS_WARNING(plist)
233 #define ACPI_BIOS_ERROR(plist)
234 #define ACPI_DEBUG_OBJECT(obj,l,i)
236 #endif /* ACPI_NO_ERROR_MESSAGES */
239 /*
240  * Debug macros that are conditionally compiled
241  */
242 #ifndef ACPI_DEBUG_OUTPUT
244 /*
245  * If ACPI_GET_FUNCTION_NAME was not defined in the compiler-dependent header,
246  * define it now. This is the case where there the compiler does not support
247  * a __FUNCTION__ macro or equivalent.
248  */
249 #ifndef ACPI_GET_FUNCTION_NAME
250 #define ACPI_GET_FUNCTION_NAME          _AcpiFunctionName
252 /*
253  * The Name parameter should be the procedure name as a quoted string.
254  * The function name is also used by the function exit macros below.
255  * Note: (const char) is used to be compatible with the debug interfaces
256  * and macros such as __FUNCTION__.
257  */

```

```

258 #define ACPI_FUNCTION_NAME(Name)      static const char _AcpiFunctionName[] =
260 #else
261 /* Compiler supports __FUNCTION__ (or equivalent) -- Ignore this macro */
263 #define ACPI_FUNCTION_NAME(Name)
264 #endif /* ACPI_GET_FUNCTION_NAME */

266 /*
267 * Common parameters used for debug output functions:
268 * line number, function name, module(file) name, component ID
269 */
270 #define ACPI_DEBUG_PARAMETERS \
271     __LINE__, ACPI_GET_FUNCTION_NAME, _AcpiModuleName, _COMPONENT

273 /* Check if debug output is currently dynamically enabled */

275 #define ACPI_IS_DEBUG_ENABLED(Level, Component) \
276     ((Level & AcpiDbgLevel) && (Component & AcpiDbgLayer))

278 /*
279 * Master debug print macros
280 * Print message if and only if:
281 *     1) Debug print for the current component is enabled
282 *     2) Debug error level or trace level for the print statement is enabled
283 *
284 * November 2012: Moved the runtime check for whether to actually emit the
285 * debug message outside of the print function itself. This improves overall
286 * performance at a relatively small code cost. Implementation involves the
287 * use of variadic macros supported by C99.
288 *
289 * Note: the ACPI_DO_WHILE0 macro is used to prevent some compilers from
290 * complaining about these constructs. On other compilers the do...while
291 * adds some extra code, so this feature is optional.
292 */
293 #ifdef ACPI_USE_DO_WHILE_0
294 #define ACPI_DO_WHILE0(a)          do a while(0)
295 #else
296 #define ACPI_DO_WHILE0(a)          a
297 #endif

299 /* DEBUG_PRINT functions */

301 #define ACPI_DEBUG_PRINT(plist)      ACPI_ACTUAL_DEBUG plist
302 #define ACPI_DEBUG_PRINT_RAW(plist) ACPI_ACTUAL_DEBUG_RAW plist

304 /* Helper macros for DEBUG_PRINT */

306 #define ACPI_DO_DEBUG_PRINT(Function, Level, Line, Filename, Modulename, Compone
307     ACPI_DO_WHILE0 ({ \
308         if (ACPI_IS_DEBUG_ENABLED (Level, Component)) \
309         { \
310             Function (Level, Line, Filename, Modulename, Component, __VA_ARGS__)
311         } \
312     })

314 #define ACPI_ACTUAL_DEBUG(Level, Line, Filename, Modulename, Component, ...) \
315     ACPI_DO_DEBUG_PRINT (AcpiDebugPrint, Level, Line, \
316         Filename, Modulename, Component, __VA_ARGS__)

318 #define ACPI_ACTUAL_DEBUG_RAW(Level, Line, Filename, Modulename, Component, ...) \
319     ACPI_DO_DEBUG_PRINT (AcpiDebugPrintRaw, Level, Line, \
320         Filename, Modulename, Component, __VA_ARGS__)

323 /*

```

```

324 * Function entry tracing
325 *
326 * The name of the function is emitted as a local variable that is
327 * intended to be used by both the entry trace and the exit trace.
328 */

330 /* Helper macro */

332 #define ACPI_TRACE_ENTRY(Name, Function, Type, Param) \
333     ACPI_FUNCTION_NAME (Name) \
334     Function (ACPI_DEBUG_PARAMETERS, (Type) (Param))

336 /* The actual entry trace macros */

338 #define ACPI_FUNCTION_TRACE(Name) \
339     ACPI_FUNCTION_NAME (Name) \
340     AcpiUtTrace (ACPI_DEBUG_PARAMETERS)

342 #define ACPI_FUNCTION_TRACE_PTR(Name, Pointer) \
343     ACPI_TRACE_ENTRY (Name, AcpiUtTracePtr, void *, Pointer)

345 #define ACPI_FUNCTION_TRACE_U32(Name, Value) \
346     ACPI_TRACE_ENTRY (Name, AcpiUtTraceU32, UINT32, Value)

348 #define ACPI_FUNCTION_TRACE_STR(Name, String) \
349     ACPI_TRACE_ENTRY (Name, AcpiUtTraceStr, char *, String)

351 #define ACPI_FUNCTION_ENTRY() \
352     AcpiUtTrackStackPtr()

355 /*
356 * Function exit tracing
357 *
358 * These macros include a return statement. This is usually considered
359 * bad form, but having a separate exit macro before the actual return
360 * is very ugly and difficult to maintain.
361 *
362 * One of the FUNCTION_TRACE macros above must be used in conjunction
363 * with these macros so that "_AcpiFunctionName" is defined.
364 *
365 * There are two versions of most of the return macros. The default version is
366 * safer, since it avoids side-effects by guaranteeing that the argument will
367 * not be evaluated twice.
368 *
369 * A less-safe version of the macros is provided for optional use if the
370 * compiler uses excessive CPU stack (for example, this may happen in the
371 * debug case if code optimization is disabled.)
372 */

374 /* Exit trace helper macro */

376 #ifndef ACPI_SIMPLE_RETURN_MACROS

378 #define ACPI_TRACE_EXIT(Function, Type, Param) \
379     ACPI_DO_WHILE0 ({ \
380         register Type _Param = (Type) (Param); \
381         Function (ACPI_DEBUG_PARAMETERS, _Param); \
382         return (_Param); \
383     })

385 #else /* Use original less-safe macros */

387 #define ACPI_TRACE_EXIT(Function, Type, Param) \
388     ACPI_DO_WHILE0 ({ \
389         Function (ACPI_DEBUG_PARAMETERS, (Type) (Param)); \

```

```

390     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__ */

```

new/usr/src/common/acpica/include/acparser.h

1

```
*****
8630 Fri Jan 17 07:30:59 2014
new/usr/src/common/acpica/include/acparser.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: acparser.h - AML Parser subcomponent prototypes and defines
4 *
5 *****/
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACPARSER_H__
46 #define __ACPARSER_H__
48
49 #define OP_HAS_RETURN_VALUE 1
51 /* Variable number of arguments. This field must be 32 bits */
53 #define ACPI_VAR_ARGS ACPI_UINT32_MAX
55
56 #define ACPI_PARSE_DELETE_TREE 0x0001
57 #define ACPI_PARSE_NO_TREE_DELETE 0x0000
58 #define ACPI_PARSE_TREE_MASK 0x0001
```

new/usr/src/common/acpica/include/acparser.h

2

```
60 #define ACPI_PARSE_LOAD_PASS1 0x0010
61 #define ACPI_PARSE_LOAD_PASS2 0x0020
62 #define ACPI_PARSE_EXECUTE 0x0030
63 #define ACPI_PARSE_MODE_MASK 0x0030
65 #define ACPI_PARSE_DEFERRED_OP 0x0100
66 #define ACPI_PARSE_DISASSEMBLE 0x0200
68 #define ACPI_PARSE_MODULE_LEVEL 0x0400
70 /*****
71 *
72 * Parser interfaces
73 *
74 *****/
77 /*
78 * psxface - Parser external interfaces
79 */
80 ACPI_STATUS
81 AcpiPsExecuteMethod (
82     ACPI_EVALUATE_INFO *Info);
85 /*
86 * psargs - Parse AML opcode arguments
87 */
88 UINT8 *
89 AcpiPsGetNextPackageEnd (
90     ACPI_PARSE_STATE *ParserState);
92 char *
93 AcpiPsGetNextNamestring (
94     ACPI_PARSE_STATE *ParserState);
96 void
97 AcpiPsGetNextSimpleArg (
98     ACPI_PARSE_STATE *ParserState,
99     UINT32 ArgType,
100     ACPI_PARSE_OBJECT *Arg);
102 ACPI_STATUS
103 AcpiPsGetNextNamepath (
104     ACPI_WALK_STATE *WalkState,
105     ACPI_PARSE_STATE *ParserState,
106     ACPI_PARSE_OBJECT *Arg,
107     BOOLEAN MethodCall);
109 ACPI_STATUS
110 AcpiPsGetNextArg (
111     ACPI_WALK_STATE *WalkState,
112     ACPI_PARSE_STATE *ParserState,
113     UINT32 ArgType,
114     ACPI_PARSE_OBJECT **ReturnArg);
117 /*
118 * psfind
119 */
120 ACPI_PARSE_OBJECT *
121 AcpiPsFindName (
122     ACPI_PARSE_OBJECT *Scope,
123     UINT32 Name,
124     UINT32 Opcode);
```



```

126 ACPI_PARSE_OBJECT*
127 AcpiPsGetParent (
128     ACPI_PARSE_OBJECT    *Op);

131 /*
132  * psoobject - 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     **Op);

141 ACPI_STATUS
142 AcpiPsCreateOp (
143     ACPI_WALK_STATE    *WalkState,
144     UINT8               *AmlOpStart,
145     ACPI_PARSE_OBJECT    **NewOp);

147 ACPI_STATUS
148 AcpiPsCompleteOp (
149     ACPI_WALK_STATE    *WalkState,
150     ACPI_PARSE_OBJECT    **Op,
151     ACPI_STATUS         Status);

153 ACPI_STATUS
154 AcpiPsCompleteFinalOp (
155     ACPI_WALK_STATE    *WalkState,
156     ACPI_PARSE_OBJECT    *Op,
157     ACPI_STATUS         Status);

160 /*
161  * psopinfo - AML Opcode information
162  */
163 const ACPI_OPCODE_INFO *
164 AcpiPsGetOpcodeInfo (
165     UINT16              Opcode);

167 char *
168 AcpiPsGetOpcodeName (
169     UINT16              Opcode);

171 UINT8
172 AcpiPsGetArgumentCount (
173     UINT32              OpType);

176 /*
177  * psparse - top level parsing routines
178  */
179 ACPI_STATUS
180 AcpiPsParseAml (
181     ACPI_WALK_STATE    *WalkState);

183 UINT32
184 AcpiPsGetOpcodeSize (
185     UINT32              Opcode);

187 UINT16
188 AcpiPsPeekOpcode (
189     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  * pssscope - 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 (

```

new/usr/src/common/acpica/include/acparser.h

```
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);
```

5

new/usr/src/common/acpica/include/acparser.h

```
322 BOOLEAN
323 AcpiPsIsLeadingChar (
324     UINT32                   c);

327 BOOLEAN
328 AcpiPsIsPrefixChar (
329     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__ */
```

6

new/usr/src/common/acpica/include/acpi.h

1

```
*****
3186 Fri Jan 17 07:31:00 2014
new/usr/src/common/acpica/include/acpi.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acpi.h - Master public include file used to interface to ACPICA
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACPI_H__
45 #define __ACPI_H__

47 /*
48 * Public include files for use by code that will interface to ACPICA.
49 *
50 * Information includes the ACPICA data types, names, exceptions, and
51 * external interface prototypes. Also included are the definitions for
52 * all ACPI tables (FADT, MADT, etc.)
53 *
54 * Note: The order of these include files is important.
55 */
56 #include "platform/acenv.h" /* Environment-specific items */
57 #include "acnames.h" /* Common ACPI names and strings */
58 #include "actypes.h" /* ACPICA data types and structures */
```

new/usr/src/common/acpica/include/acpi.h

2

```
59 #include "acexcep.h" /* ACPICA exceptions */
60 #include "actbl.h" /* ACPI table definitions */
61 #include "acoutput.h" /* Error output and Debug macros */
62 #include "acrestyp.h" /* Resource Descriptor structs */
63 #include "acpiosxf.h" /* OSL interfaces (ACPICA-to-OS) */
64 #include "acpixf.h" /* ACPI core subsystem external interfaces */
65 #ifdef ACPI_NATIVE_INTERFACE_HEADER
66 #include ACPI_NATIVE_INTERFACE_HEADER
67 #endif

69 #endif /* __ACPI_H__ */
```

new/usr/src/common/acpica/include/acpiosxf.h

1

```
*****
13374 Fri Jan 17 07:31:00 2014
new/usr/src/common/acpica/include/acpiosxf.h
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Name: acpiosxf.h - All interfaces to the OS Services Layer (OSL). These
4 * interfaces must be implemented by OSL to interface the
5 * ACPI components to the host operating system.
6 *
7 *****/
8
9 /*
10 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
45
46 #ifndef __ACPIOSXF_H__
47 #define __ACPIOSXF_H__
48
49 #include "platform/acenv.h"
50 #include "actypes.h"
51
52
53 /* Types for AcpiOsExecute */
54
55 typedef enum
56 {
57     OSL_GLOBAL_LOCK_HANDLER,
58     OSL_NOTIFY_HANDLER,
```

new/usr/src/common/acpica/include/acpiosxf.h

2

```
59     OSL_GPE_HANDLER,
60     OSL_DEBUGGER_THREAD,
61     OSL_EC_POLL_HANDLER,
62     OSL_EC_BURST_HANDLER
63
64 } ACPI_EXECUTE_TYPE;
65
66 #define ACPI_NO_UNIT_LIMIT ((UINT32) -1)
67 #define ACPI_MUTEX_SEM 1
68
69
70 /* Functions for AcpiOsSignal */
71
72 #define ACPI_SIGNAL_FATAL 0
73 #define ACPI_SIGNAL_BREAKPOINT 1
74
75 typedef struct acpi_signal_fatal_info
76 {
77     UINT32 Type;
78     UINT32 Code;
79     UINT32 Argument;
80
81 } ACPI_SIGNAL_FATAL_INFO;
82
83
84 /*
85 * OSL Initialization and shutdown primitives
86 */
87 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsInitialize
88 ACPI_STATUS
89 AcpiOsInitialize (
90     void);
91 #endif
92
93 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsTerminate
94 ACPI_STATUS
95 AcpiOsTerminate (
96     void);
97 #endif
98
99
100 /*
101 * ACPI Table interfaces
102 */
103 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsGetRootPointer
104 ACPI_PHYSICAL_ADDRESS
105 AcpiOsGetRootPointer (
106     void);
107 #endif
108
109 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsPredefinedOverride
110 ACPI_STATUS
111 AcpiOsPredefinedOverride (
112     const ACPI_PREDEFINED_NAMES *InitVal,
113     ACPI_STRING *NewVal);
114 #endif
115
116 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsTableOverride
117 ACPI_STATUS
118 AcpiOsTableOverride (
119     ACPI_TABLE_HEADER *ExistingTable,
120     ACPI_TABLE_HEADER **NewTable);
121 #endif
122
123 #ifndef ACPI_USE_ALTERNATE_PROTOTYPE_AcpiOsPhysicalTableOverride
124 ACPI_STATUS
```

```

125 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 AcpiOsPrintf (
464     const char                *Format,
465     ...);
466 #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 /* RequestedFileType 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__ */
```



new/usr/src/common/acpica/include/acpixf.h

1

```
*****
20189 Fri Jan 17 07:31:01 2014
new/usr/src/common/acpica/include/acpixf.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acpixf.h - External interfaces to the ACPI subsystem
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACXFACE_H__
46 #define __ACXFACE_H__

48 /* Current ACPICA subsystem version in YYYYMMDD format */

50 #define ACPI_CA_VERSION 0x20140114

52 #include "acconfig.h"
53 #include "actypes.h"
54 #include "actbl.h"
55 #include "acbuffer.h"
```

new/usr/src/common/acpica/include/acpixf.h

2

```
57 /*
58 * Globals that are publically available
59 */
60 extern UINT32 AcpiCurrentGpeCount;
61 extern ACPI_TABLE_FADT AcpiGbl_FADT;
62 extern BOOLEAN AcpiGbl_SystemAwakeAndRunning;
63 extern BOOLEAN AcpiGbl_ReducedHardware; /* ACPI 5.0 */
64 extern UINT8 AcpiGbl_OsiData;

66 /* Runtime configuration of debug print levels */

68 extern UINT32 AcpiDbgLevel;
69 extern UINT32 AcpiDbgLayer;

71 /* ACPICA runtime options */

73 extern UINT8 AcpiGbl_AllMethodsSerialized;
74 extern UINT8 AcpiGbl_CopyDsdtdLocally;
75 extern UINT8 AcpiGbl_CreateOsiMethod;
76 extern UINT8 AcpiGbl_DisableAutoRepair;
77 extern UINT8 AcpiGbl_DisablesSdtdTableLoad;
78 extern UINT8 AcpiGbl_DoNotUseXsdt;
79 extern UINT8 AcpiGbl_EnableAmlDebugObject;
80 extern UINT8 AcpiGbl_EnableInterpreterSlack;
81 extern UINT32 AcpiGbl_TraceFlags;
82 extern ACPI_NAME AcpiGbl_TraceMethodName;
83 extern UINT8 AcpiGbl_TruncateIoAddresses;
84 extern UINT8 AcpiGbl_Use32BitFadtAddresses;
85 extern UINT8 AcpiGbl_UseDefaultRegisterWidths;

88 /*
89 * Hardware-reduced prototypes. All interfaces that use these macros will
90 * be configured out of the ACPICA build if the ACPI_REDUCED_HARDWARE flag
91 * is set to TRUE.
92 */
93 #if (!ACPI_REDUCED_HARDWARE)
94 #define ACPI_HW_DEPENDENT_RETURN_STATUS(Prototype) \
95     Prototype;

97 #define ACPI_HW_DEPENDENT_RETURN_OK(Prototype) \
98     Prototype;

100 #define ACPI_HW_DEPENDENT_RETURN_VOID(Prototype) \
101     Prototype;

103 #else
104 #define ACPI_HW_DEPENDENT_RETURN_STATUS(Prototype) \
105     static ACPI_INLINE Prototype {return(AE_NOT_CONFIGURED);}

107 #define ACPI_HW_DEPENDENT_RETURN_OK(Prototype) \
108     static ACPI_INLINE Prototype {return(AE_OK);}

110 #define ACPI_HW_DEPENDENT_RETURN_VOID(Prototype) \
111     static ACPI_INLINE Prototype {return;}

113 #endif /* !ACPI_REDUCED_HARDWARE */

116 /*
117 * Initialization
118 */
119 ACPI_STATUS
120 AcpiInitializeTables (
121     ACPI_TABLE_DESC *InitialStorage,
122     UINT32 InitialTableCount,
```

```

123     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 /*
256  * Namespace and name interfaces
257  */
258 ACPI_STATUS
259 AcpiWalkNamespace (
260     ACPI_OBJECT_TYPE      Type,
261     ACPI_HANDLE            StartObject,
262     UINT32                 MaxDepth,
263     ACPI_WALK_CALLBACK     DescendingCallback,
264     ACPI_WALK_CALLBACK     AscendingCallback,
265     void                   *Context,
266     void                   **ReturnValue);

268 ACPI_STATUS
269 AcpiGetDevices (
270     char                   *HID,
271     ACPI_WALK_CALLBACK     UserFunction,
272     void                   *Context,
273     void                   **ReturnValue);

275 ACPI_STATUS
276 AcpiGetName (
277     ACPI_HANDLE            Object,
278     UINT32                 NameType,
279     ACPI_BUFFER            *RetPathPtr);

281 ACPI_STATUS
282 AcpiGetHandle (
283     ACPI_HANDLE            Parent,
284     ACPI_STRING            Pathname,
285     ACPI_HANDLE            *RetHandle);

287 ACPI_STATUS
288 AcpiAttachData (
289     ACPI_HANDLE            Object,
290     ACPI_OBJECT_HANDLER    Handler,
291     void                   *Data);

293 ACPI_STATUS
294 AcpiDetachData (
295     ACPI_HANDLE            Object,
296     ACPI_OBJECT_HANDLER    Handler);

298 ACPI_STATUS
299 AcpiGetData (
300     ACPI_HANDLE            Object,
301     ACPI_OBJECT_HANDLER    Handler,
302     void                   **Data);

304 ACPI_STATUS
305 AcpiDebugTrace (
306     char                   *Name,
307     UINT32                 DebugLevel,
308     UINT32                 DebugLayer,
309     UINT32                 Flags);

312 /*
313  * Object manipulation and enumeration
314  */
315 ACPI_STATUS
316 AcpiEvaluateObject (
317     ACPI_HANDLE            Object,
318     ACPI_STRING            Pathname,
319     ACPI_OBJECT_LIST       *ParameterObjects,
320     ACPI_BUFFER            *ReturnObjectBuffer);

```

```

322 ACPI_STATUS
323 AcpiEvaluateObjectTyped (
324     ACPI_HANDLE            Object,
325     ACPI_STRING            Pathname,
326     ACPI_OBJECT_LIST       *ExternalParams,
327     ACPI_BUFFER            *ReturnBuffer,
328     ACPI_OBJECT_TYPE       ReturnObject);

330 ACPI_STATUS
331 AcpiGetObjectInfo (
332     ACPI_HANDLE            Object,
333     ACPI_DEVICE_INFO       **ReturnBuffer);

335 ACPI_STATUS
336 AcpiInstallMethod (
337     UINT8                  *Buffer);

339 ACPI_STATUS
340 AcpiGetNextObject (
341     ACPI_OBJECT_TYPE       Type,
342     ACPI_HANDLE            Parent,
343     ACPI_HANDLE            Child,
344     ACPI_HANDLE            *OutHandle);

346 ACPI_STATUS
347 AcpiGetType (
348     ACPI_HANDLE            Object,
349     ACPI_OBJECT_TYPE       *OutType);

351 ACPI_STATUS
352 AcpiGetParent (
353     ACPI_HANDLE            Object,
354     ACPI_HANDLE            *OutHandle);

357 /*
358  * Handler interfaces
359  */
360 ACPI_STATUS
361 AcpiInstallInitializationHandler (
362     ACPI_INIT_HANDLER       Handler,
363     UINT32                   Function);

365 ACPI_HW_DEPENDENT_RETURN_STATUS (
366     ACPI_STATUS
367     AcpiInstallSciHandler (
368         ACPI_SCI_HANDLER     Address,
369         void                  *Context));

371 ACPI_HW_DEPENDENT_RETURN_STATUS (
372     ACPI_STATUS
373     AcpiRemoveSciHandler (
374         ACPI_SCI_HANDLER     Address));

376 ACPI_HW_DEPENDENT_RETURN_STATUS (
377     ACPI_STATUS
378     AcpiInstallGlobalEventHandler (
379         ACPI_GBL_EVENT_HANDLER Handler,
380         void                  *Context));

382 ACPI_HW_DEPENDENT_RETURN_STATUS (
383     ACPI_STATUS
384     AcpiInstallFixedEventHandler (
385         UINT32                 AcpiEvent,
386         ACPI_EVENT_HANDLER     Handler,

```

```

387 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__ */
```

new/usr/src/common/acpica/include/acpredef.h

1

```
*****
42485 Fri Jan 17 07:31:01 2014
new/usr/src/common/acpica/include/acpredef.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acpredef - Information table for ACPI predefined methods and objects
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __ACPREDEF_H__
45 #define __ACPREDEF_H__

48 /*****
49 *
50 * Return Package types
51 *
52 * 1) PTYPE1 packages do not contain sub-packages.
53 *
54 * ACPI_PTYPE1_FIXED: Fixed-length length, 1 or 2 object types:
55 *   object type
56 *   count
57 *   object type
58 *   count
59 */
```

new/usr/src/common/acpica/include/acpredef.h

2

```
60 * ACPI_PTYPE1_VAR: Variable-length length. Zero-length package is allowed:
61 *   object type (Int/Buf/Ref)
62 *
63 * ACPI_PTYPE1_OPTION: Package has some required and some optional elements
64 *   (Used for _PRW)
65 *
66 *
67 * 2) PTYPE2 packages contain a Variable-length number of sub-packages. Each
68 *   of the different types describe the contents of each of the sub-packages.
69 *
70 * ACPI_PTYPE2: Each subpackage contains 1 or 2 object types. Zero-length
71 *   parent package is allowed:
72 *   object type
73 *   count
74 *   object type
75 *   count
76 *   (Used for _ALR, _MLS, _PSS, _TRT, _TSS)
77 *
78 * ACPI_PTYPE2_COUNT: Each subpackage has a count as first element.
79 *   Zero-length parent package is allowed:
80 *   object type
81 *   (Used for _CSD, _PSD, _TSD)
82 *
83 * ACPI_PTYPE2_PKG_COUNT: Count of subpackages at start, 1 or 2 object types:
84 *   object type
85 *   count
86 *   object type
87 *   count
88 *   (Used for _CST)
89 *
90 * ACPI_PTYPE2_FIXED: Each subpackage is of Fixed-length. Zero-length
91 *   parent package is allowed.
92 *   (Used for _PRT)
93 *
94 * ACPI_PTYPE2_MIN: Each subpackage has a Variable-length but minimum length.
95 *   Zero-length parent package is allowed:
96 *   (Used for _HPX)
97 *
98 * ACPI_PTYPE2_REV_FIXED: Revision at start, each subpackage is Fixed-length
99 *   (Used for _ART, _FPS)
100 *
101 * ACPI_PTYPE2_FIX_VAR: Each subpackage consists of some fixed-length elements
102 *   followed by an optional element. Zero-length parent package is allowed.
103 *   object type
104 *   count
105 *   object type
106 *   count = 0 (optional)
107 *   (Used for _DLM)
108 *
109 *****/

111 enum AcpiReturnPackageTypes
112 {
113     ACPI_PTYPE1_FIXED      = 1,
114     ACPI_PTYPE1_VAR       = 2,
115     ACPI_PTYPE1_OPTION    = 3,
116     ACPI_PTYPE2           = 4,
117     ACPI_PTYPE2_COUNT     = 5,
118     ACPI_PTYPE2_PKG_COUNT = 6,
119     ACPI_PTYPE2_FIXED     = 7,
120     ACPI_PTYPE2_MIN       = 8,
121     ACPI_PTYPE2_REV_FIXED = 9,
122     ACPI_PTYPE2_FIX_VAR   = 10
123 };
```



```

126 /* Support macros for users of the predefined info table */
128 #define METHOD_PREDEF_ARGS_MAX      4
129 #define METHOD_ARG_BIT_WIDTH        3
130 #define METHOD_ARG_MASK              0x0007
131 #define ARG_COUNT_IS_MINIMUM        0x8000
132 #define METHOD_MAX_ARG_TYPE          ACPI_TYPE_PACKAGE

134 #define METHOD_GET_ARG_COUNT(ArgList) ((ArgList) & METHOD_ARG_MASK)
135 #define METHOD_GET_NEXT_TYPE(ArgList) (((ArgList) >= METHOD_ARG_BIT_WIDTH) &

137 /* Macros used to build the predefined info table */

139 #define METHOD_0ARGS                  0
140 #define METHOD_1ARGS(a1)              (1 | (a1 << 3))
141 #define METHOD_2ARGS(a1,a2)           (2 | (a1 << 3) | (a2 << 6))
142 #define METHOD_3ARGS(a1,a2,a3)        (3 | (a1 << 3) | (a2 << 6) | (a3 << 9))
143 #define METHOD_4ARGS(a1,a2,a3,a4)     (4 | (a1 << 3) | (a2 << 6) | (a3 << 9) |

145 #define METHOD_RETURNS(type)          (type)
146 #define METHOD_NO_RETURN_VALUE        0

148 #define PACKAGE_INFO(a,b,c,d,e,f)    {{{(a),(b),(c),(d)}, (((UINT16)(f)) <<

151 /* Support macros for the resource descriptor info table */

153 #define WIDTH_1                      0x0001
154 #define WIDTH_2                      0x0002
155 #define WIDTH_3                      0x0004
156 #define WIDTH_8                      0x0008
157 #define WIDTH_16                     0x0010
158 #define WIDTH_32                     0x0020
159 #define WIDTH_64                     0x0040
160 #define VARIABLE_DATA                0x0080
161 #define NUM_RESOURCE_WIDTHS          8

163 #define WIDTH_ADDRESS                 WIDTH_16 | WIDTH_32 | WIDTH_64

166 #ifdef ACPI_CREATE_PREDEFINED_TABLE
167 /*****
168  *
169  * Predefined method/object information table.
170  *
171  * These are the names that can actually be evaluated via AcpiEvaluateObject.
172  * Not present in this table are the following:
173  *
174  * 1) Predefined/Reserved names that are never evaluated via
175  *    AcpiEvaluateObject:
176  *      _Lxx and _Exx GPE methods
177  *      _Qxx EC methods
178  *      _T_x compiler temporary variables
179  *      _Wxx wake events
180  *
181  * 2) Predefined names that never actually exist within the AML code:
182  *      Predefined resource descriptor field names
183  *
184  * 3) Predefined names that are implemented within ACPICA:
185  *      _OSI
186  *
187  * The main entries in the table each contain the following items:
188  *
189  * Name                - The ACPI reserved name
190  * ArgumentList         - Contains (in 16 bits), the number of required
191  *                        arguments to the method (3 bits), and a 3-bit type

```

```

192 *      field for each argument (up to 4 arguments). The
193 *      METHOD_?ARGS macros generate the correct packed data.
194 * ExpectedBtypes      - Allowed type(s) for the return value.
195 *                      0 means that no return value is expected.
196 *
197 * For methods that return packages, the next entry in the table contains
198 * information about the expected structure of the package. This information
199 * is saved here (rather than in a separate table) in order to minimize the
200 * overall size of the stored data.
201 *
202 * Note: The additional braces are intended to promote portability.
203 *
204 * Note2: Table is used by the kernel-resident subsystem, the iASL compiler,
205 * and the AcpiHelp utility.
206 *
207 * TBD: _PRT - currently ignore reversed entries. Attempt to fix in nsrepair.
208 * Possibly fixing package elements like _BIF, etc.
209 *
210 *****/
212 const ACPI_PREDEFINED_INFO      AcpiGbl_PredefinedMethods[] =
213 {
214     {"_AC0",      METHOD_0ARGS,
215         METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
217     {"_AC1",      METHOD_0ARGS,
218         METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
220     {"_AC2",      METHOD_0ARGS,
221         METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
223     {"_AC3",      METHOD_0ARGS,
224         METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
226     {"_AC4",      METHOD_0ARGS,
227         METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
229     {"_AC5",      METHOD_0ARGS,
230         METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
232     {"_AC6",      METHOD_0ARGS,
233         METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
235     {"_AC7",      METHOD_0ARGS,
236         METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
238     {"_AC8",      METHOD_0ARGS,
239         METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
241     {"_AC9",      METHOD_0ARGS,
242         METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
244     {"_ADR",      METHOD_0ARGS,
245         METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},
247     {"_AEI",      METHOD_0ARGS,
248         METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},
250     {"_AL0",      METHOD_0ARGS,
251         METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
252         PACKAGE_INFO (ACPI_PTTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0
254     {"_AL1",      METHOD_0ARGS,
255         METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
256         PACKAGE_INFO (ACPI_PTTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0,0,0,0

```

```

258     {"_AL2", METHOD_0ARGS,
259      METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}, /* Variable-length (Refs)
260      PACKAGE_INFO (ACPI_PTTYPE1_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_PTTYPE1_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_PTTYPE1_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_PTTYPE1_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_PTTYPE1_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_PTTYPE1_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_PTTYPE1_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_PTTYPE1_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_PTTYPE2, 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_PTTYPE2_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_PTTYPE1_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_PTTYPE1_FIXED, ACPI_RTYPE_INTEGER, 9, ACPI

333     {"_BIX", METHOD_0ARGS,
334      METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}, /* Fixed-length (16 Int),
335      PACKAGE_INFO (ACPI_PTTYPE1_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_PTTYPE1_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_PTTYPE1_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_PTTYPE1_VAR, ACPI_RTYPE_INTEGER | ACPI_RTY

376     {"_CLS", METHOD_0ARGS,
377      METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}, /* Fixed-length (3 Int) *
378      PACKAGE_INFO (ACPI_PTTYPE1_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_PTTYPE1_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, ACPI

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),

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),

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),

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),

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),

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),

809     {{"_S1D", METHOD_0ARGS,
810        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

812     {{"_S2D", METHOD_0ARGS,
813        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

815     {{"_S3D", METHOD_0ARGS,
816        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

818     {{"_S4D", METHOD_0ARGS,
819        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

821     {{"_S0W", METHOD_0ARGS,
822        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

824     {{"_S1W", METHOD_0ARGS,
825        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

827     {{"_S2W", METHOD_0ARGS,
828        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

830     {{"_S3W", METHOD_0ARGS,
831        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

833     {{"_S4W", METHOD_0ARGS,
834        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

836     {{"_SBS", METHOD_0ARGS,
837        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

839     {{"_SCP", METHOD_1ARGS (ACPI_TYPE_INTEGER) | ARG_COUNT_IS_MINIMUM,
840        METHOD_NO_RETURN_VALUE}}; /* Acpi 1.0 allowed 1 integer arg. A

842     {{"_SDD", METHOD_1ARGS (ACPI_TYPE_BUFFER),
843        METHOD_NO_RETURN_VALUE}};

845     {{"_SEG", METHOD_0ARGS,
846        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

848     {{"_SHL", METHOD_1ARGS (ACPI_TYPE_INTEGER),
849        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

851     {{"_SLI", METHOD_0ARGS,
```

```

852         METHOD_RETURNS (ACPI_RTYPE_BUFFER)}};

854     {{"_SPD", METHOD_1ARGS (ACPI_TYPE_INTEGER),
855        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

857     {{"_SRS", METHOD_1ARGS (ACPI_TYPE_BUFFER),
858        METHOD_NO_RETURN_VALUE}};

860     {{"_SRT", METHOD_1ARGS (ACPI_TYPE_BUFFER),
861        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

863     {{"_SRV", METHOD_0ARGS,
864        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}}; /* See IPMI spec */

866     {{"_SST", METHOD_1ARGS (ACPI_TYPE_INTEGER),
867        METHOD_NO_RETURN_VALUE}};

869     {{"_STA", METHOD_0ARGS,
870        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

872     {{"_STM", METHOD_3ARGS (ACPI_TYPE_BUFFER, ACPI_TYPE_BUFFER, ACPI_TYPE_BUFF
873        METHOD_NO_RETURN_VALUE}};

875     {{"_STP", METHOD_2ARGS (ACPI_TYPE_INTEGER, ACPI_TYPE_INTEGER),
876        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

878     {{"_STR", METHOD_0ARGS,
879        METHOD_RETURNS (ACPI_RTYPE_BUFFER)}};

881     {{"_STV", METHOD_2ARGS (ACPI_TYPE_INTEGER, ACPI_TYPE_INTEGER),
882        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

884     {{"_SUB", METHOD_0ARGS,
885        METHOD_RETURNS (ACPI_RTYPE_STRING)}};

887     {{"_SUN", METHOD_0ARGS,
888        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

890     {{"_SWS", METHOD_0ARGS,
891        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

893     {{"_TC1", METHOD_0ARGS,
894        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

896     {{"_TC2", METHOD_0ARGS,
897        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

899     {{"_TDL", METHOD_0ARGS,
900        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

902     {{"_TIP", METHOD_1ARGS (ACPI_TYPE_INTEGER),
903        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

905     {{"_TIV", METHOD_1ARGS (ACPI_TYPE_INTEGER),
906        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

908     {{"_TMP", METHOD_0ARGS,
909        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

911     {{"_TPC", METHOD_0ARGS,
912        METHOD_RETURNS (ACPI_RTYPE_INTEGER)}};

914     {{"_TPT", METHOD_1ARGS (ACPI_TYPE_INTEGER),
915        METHOD_NO_RETURN_VALUE}};

917     {{"_TRT", METHOD_0ARGS,
```

```

918     METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
919     PACKAGE_INFO (ACPI_PTYPE2, ACPI_RTYPE_REFERENCE, 2, ACPI_RTY

921     {"_TSD",    METHOD_0ARGS,
922     METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
923     PACKAGE_INFO (ACPI_PTYPE2_COUNT, ACPI_RTYPE_INTEGER, 5, 0, 0, 0)

925     {"_TSP",    METHOD_0ARGS,
926     METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

928     {"_TSS",    METHOD_0ARGS,
929     METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Pkgs)
930     PACKAGE_INFO (ACPI_PTYPE2, ACPI_RTYPE_INTEGER, 5, 0, 0, 0),

932     {"_TST",    METHOD_0ARGS,
933     METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

935     {"_TTS",    METHOD_1ARGS (ACPI_TYPE_INTEGER),
936     METHOD_NO_RETURN_VALUE}},

938     {"_TZD",    METHOD_0ARGS,
939     METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Variable-length (Refs)
940     PACKAGE_INFO (ACPI_PTYPE1_VAR, ACPI_RTYPE_REFERENCE, 0, 0, 0, 0)

942     {"_TZM",    METHOD_0ARGS,
943     METHOD_RETURNS (ACPI_RTYPE_REFERENCE)}},

945     {"_TZP",    METHOD_0ARGS,
946     METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

948     {"_UID",    METHOD_0ARGS,
949     METHOD_RETURNS (ACPI_RTYPE_INTEGER | ACPI_RTYPE_STRING)}},

951     {"_UPC",    METHOD_0ARGS,
952     METHOD_RETURNS (ACPI_RTYPE_PACKAGE)}}, /* Fixed-length (4 Int) *
953     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 4, 0, 0, 0)

955     {"_UPD",    METHOD_0ARGS,
956     METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

958     {"_UPP",    METHOD_0ARGS,
959     METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

961     {"_VPO",    METHOD_0ARGS,
962     METHOD_RETURNS (ACPI_RTYPE_INTEGER)}},

964     /* Acpi 1.0 defined _WAK with no return value. Later, it was changed to retu
966     {"_WAK",    METHOD_1ARGS (ACPI_TYPE_INTEGER),
967     METHOD_RETURNS (ACPI_RTYPE_NONE | ACPI_RTYPE_INTEGER | ACPI_RTY
968     PACKAGE_INFO (ACPI_PTYPE1_FIXED, ACPI_RTYPE_INTEGER, 2, 0, 0, 0)

970     /* _WDG/_WED are MS extensions defined by "Windows Instrumentation" */

972     {"_WDG",    METHOD_0ARGS,
973     METHOD_RETURNS (ACPI_RTYPE_BUFFER)}},

975     {"_WED",    METHOD_1ARGS (ACPI_TYPE_INTEGER),
976     METHOD_RETURNS (ACPI_RTYPE_INTEGER | ACPI_RTYPE_STRING | ACPI_RT

978     PACKAGE_INFO (0, 0, 0, 0, 0, 0) /* Table terminator */
979 };
980 #else
981 extern const ACPI_PREDEFINED_INFO      AcpiGbl_PredefinedMethods[];
982 #endif

```

```

985 #if (defined ACPI_CREATE_RESOURCE_TABLE && defined ACPI_APPLICATION)
986 /*****
987  *
988  * Predefined names for use in Resource Descriptors. These names do not
989  * appear in the global Predefined Name table (since these names never
990  * appear in actual AML byte code, only in the original ASL)
991  *
992  * Note: Used by iASL compiler and AcpiHelp utility only.
993  *
994  *****/
995
996 const ACPI_PREDEFINED_INFO      AcpiGbl_ResourceNames[] =
997 {
998     {"_ADR",    WIDTH_16 | WIDTH_64, 0},
999     {"_ALN",    WIDTH_8 | WIDTH_16 | WIDTH_32, 0},
1000     {"_ASI",    WIDTH_8, 0},
1001     {"_ASZ",    WIDTH_8, 0},
1002     {"_ATT",    WIDTH_64, 0},
1003     {"_BAS",    WIDTH_16 | WIDTH_32, 0},
1004     {"_BM",     WIDTH_1, 0},
1005     {"_DBT",    WIDTH_16, 0}, /* Acpi 5.0 */
1006     {"_DEC",    WIDTH_1, 0},
1007     {"_DMA",    WIDTH_8, 0},
1008     {"_DPL",    WIDTH_1, 0}, /* Acpi 5.0 */
1009     {"_DRS",    WIDTH_16, 0}, /* Acpi 5.0 */
1010     {"_END",    WIDTH_1, 0}, /* Acpi 5.0 */
1011     {"_FLC",    WIDTH_2, 0}, /* Acpi 5.0 */
1012     {"_GRA",    WIDTH_ADDRESS, 0},
1013     {"_HE",     WIDTH_1, 0},
1014     {"_INT",    WIDTH_16 | WIDTH_32, 0},
1015     {"_IOR",    WIDTH_2, 0}, /* Acpi 5.0 */
1016     {"_LEN",    WIDTH_8 | WIDTH_ADDRESS, 0},
1017     {"_LIN",    WIDTH_8, 0}, /* Acpi 5.0 */
1018     {"_LL",     WIDTH_1, 0},
1019     {"_MAF",    WIDTH_1, 0},
1020     {"_MAX",    WIDTH_ADDRESS, 0},
1021     {"_MEM",    WIDTH_2, 0},
1022     {"_MIF",    WIDTH_1, 0},
1023     {"_MIN",    WIDTH_ADDRESS, 0},
1024     {"_MOD",    WIDTH_1, 0}, /* Acpi 5.0 */
1025     {"_MTP",    WIDTH_2, 0},
1026     {"_PAR",    WIDTH_8, 0}, /* Acpi 5.0 */
1027     {"_PHA",    WIDTH_1, 0}, /* Acpi 5.0 */
1028     {"_PIN",    WIDTH_16, 0}, /* Acpi 5.0 */
1029     {"_PPI",    WIDTH_8, 0}, /* Acpi 5.0 */
1030     {"_POL",    WIDTH_1 | WIDTH_2, 0}, /* Acpi 5.0 */
1031     {"_RBO",    WIDTH_8, 0},
1032     {"_RBW",    WIDTH_8, 0},
1033     {"_RNG",    WIDTH_1, 0},
1034     {"_RT",     WIDTH_8, 0}, /* Acpi 3.0 */
1035     {"_RW",     WIDTH_1, 0},
1036     {"_RXL",    WIDTH_16, 0}, /* Acpi 5.0 */
1037     {"_SHR",    WIDTH_2, 0},
1038     {"_SIZ",    WIDTH_2, 0},
1039     {"_SLV",    WIDTH_1, 0}, /* Acpi 5.0 */
1040     {"_SPE",    WIDTH_32, 0}, /* Acpi 5.0 */
1041     {"_STB",    WIDTH_2, 0}, /* Acpi 5.0 */
1042     {"_TRA",    WIDTH_ADDRESS, 0},
1043     {"_TRS",    WIDTH_1, 0},
1044     {"_TSF",    WIDTH_8, 0}, /* Acpi 3.0 */
1045     {"_TTP",    WIDTH_1, 0},
1046     {"_TXL",    WIDTH_16, 0}, /* Acpi 5.0 */
1047     {"_TYP",    WIDTH_2 | WIDTH_16, 0},
1048     {"_VEN",    VARIABLE_DATA, 0}, /* Acpi 5.0 */
1049     PACKAGE_INFO (0, 0, 0, 0, 0, 0) /* Table terminator */

```

```
1050 };

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
```



new/usr/src/common/acpica/include/acresrc.h

1

```
*****
12788 Fri Jan 17 07:31:01 2014
new/usr/src/common/acpica/include/acresrc.h
update to acpica-unix2-20140114
update to acpica-unix2-20131115
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Name: acresrc.h - Resource Manager function prototypes
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACRESRC_H__
45 #define __ACRESRC_H__
46
47 /* Need the AML resource descriptor structs */
48
49 #include "amlresrc.h"
50
51 /*
52 *
53 * If possible, pack the following structures to byte alignment, since we
54 * don't care about performance for debug output. Two cases where we cannot
55 * pack the structures:
56 *
57 * 1) Hardware does not support misaligned memory transfers
```

new/usr/src/common/acpica/include/acresrc.h

2

```
58 * 2) Compiler does not support pointers within packed structures
59 */
60 #if (!defined(ACPI_MISALIGNMENT_NOT_SUPPORTED) && !defined(ACPI_PACKED_POINTERS_
61 #pragma pack(1)
62 #endif
63
64 /*
65 * Individual entry for the resource conversion tables
66 */
67 typedef const struct acpi_rsconvert_info
68 {
69     UINT8 Opcode;
70     UINT8 ResourceOffset;
71     UINT8 AmlOffset;
72     UINT8 Value;
73
74 } ACPI_RS_CONVERT_INFO;
75
76 /* Resource conversion opcodes */
77
78 typedef enum
79 {
80     ACPI_RSC_INITGET = 0,
81     ACPI_RSC_INITSET,
82     ACPI_RSC_FLAGINIT,
83     ACPI_RSC_1BITFLAG,
84     ACPI_RSC_2BITFLAG,
85     ACPI_RSC_3BITFLAG,
86     ACPI_RSC_ADDRESS,
87     ACPI_RSC_BITMASK,
88     ACPI_RSC_BITMASK16,
89     ACPI_RSC_COUNT,
90     ACPI_RSC_COUNT16,
91     ACPI_RSC_COUNT_GPIO_PIN,
92     ACPI_RSC_COUNT_GPIO_RES,
93     ACPI_RSC_COUNT_GPIO_VEN,
94     ACPI_RSC_COUNT_SERIAL_RES,
95     ACPI_RSC_COUNT_SERIAL_VEN,
96     ACPI_RSC_DATA8,
97     ACPI_RSC_EXIT_EQ,
98     ACPI_RSC_EXIT_LE,
99     ACPI_RSC_EXIT_NE,
100     ACPI_RSC_LENGTH,
101     ACPI_RSC_MOVE_GPIO_PIN,
102     ACPI_RSC_MOVE_GPIO_RES,
103     ACPI_RSC_MOVE_SERIAL_RES,
104     ACPI_RSC_MOVE_SERIAL_VEN,
105     ACPI_RSC_MOVE8,
106     ACPI_RSC_MOVE16,
107     ACPI_RSC_MOVE32,
108     ACPI_RSC_MOVE64,
109     ACPI_RSC_SET8,
110     ACPI_RSC_SOURCE,
111     ACPI_RSC_SOURCEX
112
113 #define ACPI_RSC_INITGET 0
114 #define ACPI_RSC_INITSET 1
115 #define ACPI_RSC_FLAGINIT 2
116 #define ACPI_RSC_1BITFLAG 3
117 #define ACPI_RSC_2BITFLAG 4
118 #define ACPI_RSC_COUNT 5
119 #define ACPI_RSC_COUNT16 6
120 #define ACPI_RSC_LENGTH 7
121 #define ACPI_RSC_MOVE8 8
122 #define ACPI_RSC_MOVE16 9
123 #define ACPI_RSC_MOVE32 10
124 #define ACPI_RSC_MOVE64 11
```

```

90 #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_RS_CONVERT_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_RS_CONVERT_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_rs_dump_info
130 {
131     UINT8          Opcode;
132     UINT8          Offset;
133     char           *Name;
134     const char     **Pointer;

136 } ACPI_RS_DUMP_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_RS_DUMP_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_RS_CONVERT_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_RS_CONVERT_INFO *AcpiGbl_GetResourceDispatch[];

179 extern ACPI_RS_CONVERT_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,
201     ACPI_RESOURCE            *LinkedListBuffer,
202     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 (

```

new/usr/src/common/acpica/include/acresrc.h

```
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  * rscal
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,
256     ACPI_RESOURCE           *LinkedListBuffer,
257     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);
270     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 (
```

5

new/usr/src/common/acpica/include/acresrc.h

```
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_RS_CONVERT_INFO    *Info);

301 ACPI_STATUS
302 AcpiRsConvertResourceToAml (
303     ACPI_RESOURCE           *Resource,
304     AML_RESOURCE           *Aml,
305     ACPI_RS_CONVERT_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
```

6

```

349 AcpiRsSetResourceLength (
350     ACPI_RSDESC_SIZE      TotalLength,
351     AML_RESOURCE          *Aml);

354 /*
355  * rsdump
356  */
357 void
358 AcpiRsDumpResourceList (
359     ACPI_RESOURCE          *Resource);

361 void
362 AcpiRsDumpIrqList (
363     UINT8                  *RouteTable);

366 /*
367  * Resource conversion tables
368  */
369 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertDma[];
370 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertEndDpf[];
371 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertIo[];
372 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertFixedIo[];
373 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertEndTag[];
374 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertMemory24[];
375 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertGenericReg[];
376 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertMemory32[];
377 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertFixedMemory32[];
378 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertAddress32[];
379 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertAddress16[];
380 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertExtIrq[];
381 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertAddress64[];
382 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertExtAddress64[];
383 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertGpio[];
384 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertFixedDma[];
385 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertI2cSerialBus[];
386 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertSpiSerialBus[];
387 extern ACPI_RS_CONVERT_INFO    AcpiRsConvertUartSerialBus[];

389 /* These resources require separate get/set tables */

391 extern ACPI_RS_CONVERT_INFO    AcpiRsGetIrq[];
392 extern ACPI_RS_CONVERT_INFO    AcpiRsGetStartDpf[];
393 extern ACPI_RS_CONVERT_INFO    AcpiRsGetVendorSmall[];
394 extern ACPI_RS_CONVERT_INFO    AcpiRsGetVendorLarge[];

396 extern ACPI_RS_CONVERT_INFO    AcpiRsSetIrq[];
397 extern ACPI_RS_CONVERT_INFO    AcpiRsSetStartDpf[];
398 extern ACPI_RS_CONVERT_INFO    AcpiRsSetVendor[];

401 #if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DEBUGGER)
402 /*
403  * rsinfo
404  */
405 extern ACPI_RSDUMP_INFO        *AcpiGbl_DumpResourceDispatch[];
406 extern ACPI_RSDUMP_INFO        *AcpiGbl_DumpSerialBusDispatch[];

408 /*
409  * rsdumpinfo
410  */
411 extern ACPI_RSDUMP_INFO        AcpiRsDumpIrq[];
412 extern ACPI_RSDUMP_INFO        AcpiRsDumpPrt[];
413 extern ACPI_RSDUMP_INFO        AcpiRsDumpDma[];

```

```

414 extern ACPI_RSDUMP_INFO        AcpiRsDumpStartDpf[];
415 extern ACPI_RSDUMP_INFO        AcpiRsDumpEndDpf[];
416 extern ACPI_RSDUMP_INFO        AcpiRsDumpIo[];
417 extern ACPI_RSDUMP_INFO        AcpiRsDumpIoFlags[];
418 extern ACPI_RSDUMP_INFO        AcpiRsDumpFixedIo[];
419 extern ACPI_RSDUMP_INFO        AcpiRsDumpVendor[];
420 extern ACPI_RSDUMP_INFO        AcpiRsDumpEndTag[];
421 extern ACPI_RSDUMP_INFO        AcpiRsDumpMemory24[];
422 extern ACPI_RSDUMP_INFO        AcpiRsDumpMemory32[];
423 extern ACPI_RSDUMP_INFO        AcpiRsDumpMemoryFlags[];
424 extern ACPI_RSDUMP_INFO        AcpiRsDumpFixedMemory32[];
425 extern ACPI_RSDUMP_INFO        AcpiRsDumpAddress16[];
426 extern ACPI_RSDUMP_INFO        AcpiRsDumpAddress32[];
427 extern ACPI_RSDUMP_INFO        AcpiRsDumpAddress64[];
428 extern ACPI_RSDUMP_INFO        AcpiRsDumpExtAddress64[];
429 extern ACPI_RSDUMP_INFO        AcpiRsDumpExtIrq[];
430 extern ACPI_RSDUMP_INFO        AcpiRsDumpGenericReg[];
431 extern ACPI_RSDUMP_INFO        AcpiRsDumpGpio[];
432 extern ACPI_RSDUMP_INFO        AcpiRsDumpFixedDma[];
433 extern ACPI_RSDUMP_INFO        AcpiRsDumpCommonSerialBus[];
434 extern ACPI_RSDUMP_INFO        AcpiRsDumpI2cSerialBus[];
435 extern ACPI_RSDUMP_INFO        AcpiRsDumpSpiSerialBus[];
436 extern ACPI_RSDUMP_INFO        AcpiRsDumpUartSerialBus[];
437 extern ACPI_RSDUMP_INFO        AcpiRsDumpGeneralFlags[];
438 #endif

440 #endif /* __ACRESRC_H__ */

```

new/usr/src/common/acpica/include/acrestyp.h

1

```
*****
23494 Fri Jan 17 07:31:01 2014
new/usr/src/common/acpica/include/acrestyp.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acrestyp.h - Defines, types, and structures for resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __ACRESTYP_H__
45 #define __ACRESTYP_H__

48 /*
49 * Definitions for Resource Attributes
50 */
51 typedef UINT16          ACPI_RS_LENGTH;      /* Resource Length fi
52 typedef UINT32          ACPI_RSDESC_SIZE;    /* Max Resource Descr

54 /*
55 * Memory Attributes
56 */
57 #define ACPI_READ_ONLY_MEMORY          (UINT8) 0x00
58 #define ACPI_READ_WRITE_MEMORY         (UINT8) 0x01
```

new/usr/src/common/acpica/include/acrestyp.h

2

```
60 #define ACPI_NON_CACHEABLE_MEMORY      (UINT8) 0x00
61 #define ACPI_CACHABLE_MEMORY           (UINT8) 0x01
62 #define ACPI_WRITE_COMBINING_MEMORY     (UINT8) 0x02
63 #define ACPI_PREFETCHABLE_MEMORY       (UINT8) 0x03

65 /*! [Begin] no source code translation */
66 /*
67 * IO Attributes
68 * The ISA IO ranges are:      n000-n0FFh,  n400-n4FFh,  n800-n8FFh,  nC00-nCFFh.
69 * The non-ISA IO ranges are: n100-n3FFh,  n500-n7FFh,  n900-nBFFh,  nCD0-nFFFh.
70 */
71 /*! [End] no source code translation !*/

73 #define ACPI_NON_ISA_ONLY_RANGES        (UINT8) 0x01
74 #define ACPI_ISA_ONLY_RANGES            (UINT8) 0x02
75 #define ACPI_ENTIRE_RANGE                (ACPI_NON_ISA_ONLY_RANGES | ACPI_ISA_ONL

77 /* Type of translation - 1=Sparse, 0=Dense */

79 #define ACPI_SPARSE_TRANSLATION         (UINT8) 0x01

81 /*
82 * IO Port Descriptor Decode
83 */
84 #define ACPI_DECODE_10                  (UINT8) 0x00    /* 10-bit IO address dec
85 #define ACPI_DECODE_16                  (UINT8) 0x01    /* 16-bit IO address dec

87 /*
88 * Interrupt attributes - used in multiple descriptors
89 */

91 /* Triggering */

93 #define ACPI_LEVEL_SENSITIVE            (UINT8) 0x00
94 #define ACPI_EDGE_SENSITIVE            (UINT8) 0x01

96 /* Polarity */

98 #define ACPI_ACTIVE_HIGH                 (UINT8) 0x00
99 #define ACPI_ACTIVE_LOW                 (UINT8) 0x01
100 #define ACPI_ACTIVE_BOTH                (UINT8) 0x02

102 /* Sharing */

104 #define ACPI_EXCLUSIVE                   (UINT8) 0x00
105 #define ACPI_SHARED                     (UINT8) 0x01

107 /* Wake */

109 #define ACPI_NOT_WAKE_CAPABLE           (UINT8) 0x00
110 #define ACPI_WAKE_CAPABLE              (UINT8) 0x01

112 /*
113 * DMA Attributes
114 */
115 #define ACPI_COMPATIBILITY              (UINT8) 0x00
116 #define ACPI_TYPE_A                    (UINT8) 0x01
117 #define ACPI_TYPE_B                    (UINT8) 0x02
118 #define ACPI_TYPE_F                    (UINT8) 0x03

120 #define ACPI_NOT_BUS_MASTER             (UINT8) 0x00
121 #define ACPI_BUS_MASTER                 (UINT8) 0x01

123 #define ACPI_TRANSFER_8                 (UINT8) 0x00
124 #define ACPI_TRANSFER_8_16             (UINT8) 0x01
125 #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
133
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
140
141 #define ACPI_ADDRESS_NOT_FIXED        (UINT8) 0x00
142 #define ACPI_ADDRESS_FIXED           (UINT8) 0x01
143
144 #define ACPI_POS_DECODE               (UINT8) 0x00
145 #define ACPI_SUB_DECODE               (UINT8) 0x01
146
147 /* Producer/Consumer */
148
149 #define ACPI_PRODUCER                 (UINT8) 0x00
150 #define ACPI_CONSUMER                 (UINT8) 0x01
151
152
153 /*
154  * If possible, pack the following structures to byte alignment
155  */
156 #ifndef ACPI_MISALIGNMENT_NOT_SUPPORTED
157 #pragma pack(1)
158 #endif
159
160 /* UUID data structures for use in vendor-defined resource descriptors */
161
162 typedef struct acpi_uuid
163 {
164     UINT8                Data[ACPI_UUID_LENGTH];
165 } ACPI_UUID;
166
167 typedef struct acpi_vendor_uuid
168 {
169     UINT8                Subtype;
170     UINT8                Data[ACPI_UUID_LENGTH];
171
172 } ACPI_VENDOR_UUID;
173
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];
186
187 } ACPI_RESOURCE_IRQ;
188
189 typedef struct acpi_resource_dma
190 {
191     UINT8                Type;

```

```

192     UINT8                BusMaster;
193     UINT8                Transfer;
194     UINT8                ChannelCount;
195     UINT8                Channels[1];
196
197 } ACPI_RESOURCE_DMA;
198
199 typedef struct acpi_resource_start_dependent
200 {
201     UINT8                DescriptorLength;
202     UINT8                CompatibilityPriority;
203     UINT8                PerformanceRobustness;
204
205 } ACPI_RESOURCE_START_DEPENDENT;
206
207
208 /*
209  * The END_DEPENDENT_FUNCTIONS_RESOURCE struct is not
210  * needed because it has no fields
211  */
212
213
214 typedef struct acpi_resource_io
215 {
216     UINT8                IoDecode;
217     UINT8                Alignment;
218     UINT8                AddressLength;
219     UINT16               Minimum;
220     UINT16               Maximum;
221
222 } ACPI_RESOURCE_IO;
223
224 typedef struct acpi_resource_fixed_io
225 {
226     UINT16               Address;
227     UINT8                AddressLength;
228
229 } ACPI_RESOURCE_FIXED_IO;
230
231 typedef struct acpi_resource_fixed_dma
232 {
233     UINT16               RequestLines;
234     UINT16               Channels;
235     UINT8                Width;
236
237 } ACPI_RESOURCE_FIXED_DMA;
238
239 /* Values for Width field above */
240
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
247
248
249 typedef struct acpi_resource_vendor
250 {
251     UINT16               ByteLength;
252     UINT8                ByteData[1];
253
254 } ACPI_RESOURCE_VENDOR;
255
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];
264 } ACPI_RESOURCE_VENDOR_TYPED;
265
267 typedef struct acpi_resource_end_tag
268 {
269     UINT8       Checksum;
270 } ACPI_RESOURCE_END_TAG;
271
273 typedef struct acpi_resource_memory24
274 {
275     UINT8       WriteProtect;
276     UINT16      Minimum;
277     UINT16      Maximum;
278     UINT16      Alignment;
279     UINT16      AddressLength;
280 } ACPI_RESOURCE_MEMORY24;
281
283 typedef struct acpi_resource_memory32
284 {
285     UINT8       WriteProtect;
286     UINT32      Minimum;
287     UINT32      Maximum;
288     UINT32      Alignment;
289     UINT32      AddressLength;
290 } ACPI_RESOURCE_MEMORY32;
291
293 typedef struct acpi_resource_fixed_memory32
294 {
295     UINT8       WriteProtect;
296     UINT32      Address;
297     UINT32      AddressLength;
298 } ACPI_RESOURCE_FIXED_MEMORY32;
299
301 typedef struct acpi_memory_attribute
302 {
303     UINT8       WriteProtect;
304     UINT8       Caching;
305     UINT8       RangeType;
306     UINT8       Translation;
307 } ACPI_MEMORY_ATTRIBUTE;
308
310 typedef struct acpi_io_attribute
311 {
312     UINT8       RangeType;
313     UINT8       Translation;
314     UINT8       TranslationType;
315     UINT8       Reserved1;
316 } ACPI_IO_ATTRIBUTE;
317
319 typedef union acpi_resource_attribute
320 {
321     ACPI_MEMORY_ATTRIBUTE  Mem;
322     ACPI_IO_ATTRIBUTE      Io;

```

```

324     /* Used for the *WordSpace macros */
325
326     UINT8       TypeSpecific;
327 } ACPI_RESOURCE_ATTRIBUTE;
328
330 typedef struct acpi_resource_source
331 {
332     UINT8       Index;
333     UINT16      StringLength;
334     char        *StringPtr;
335 } ACPI_RESOURCE_SOURCE;
336
338 /* Fields common to all address descriptors, 16/32/64 bit */
339
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;
347
348 typedef struct acpi_resource_address
349 {
350     ACPI_RESOURCE_ADDRESS_COMMON
351 } ACPI_RESOURCE_ADDRESS;
352
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;
363 } ACPI_RESOURCE_ADDRESS16;
364
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;
375 } ACPI_RESOURCE_ADDRESS32;
376
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;
387 } 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;

524 } ACPI_RESOURCE_SPI_SERIALBUS;

526 /* Values for WireMode field above */

528 #define ACPI_SPI_4WIRE_MODE      0
529 #define ACPI_SPI_3WIRE_MODE      1

531 /* Values for DevicePolarity field above */

533 #define ACPI_SPI_ACTIVE_LOW      0
534 #define ACPI_SPI_ACTIVE_HIGH    1

536 /* Values for ClockPhase field above */

538 #define ACPI_SPI_FIRST_PHASE     0
539 #define ACPI_SPI_SECOND_PHASE    1

541 /* Values for ClockPolarity field above */

543 #define ACPI_SPI_START_LOW       0
544 #define ACPI_SPI_START_HIGH     1

547 typedef struct acpi_resource_uart_serialbus
548 {
549     ACPI_RESOURCE_SERIAL_COMMON
550     UINT8                Endian;
551     UINT8                DataBits;
552     UINT8                StopBits;
553     UINT8                FlowControl;
554     UINT8                Parity;
555     UINT8                LinesEnabled;
556     UINT16               RxFifoSize;
557     UINT16               TxFifoSize;
558     UINT32               DefaultBaudRate;

560 } ACPI_RESOURCE_UART_SERIALBUS;

562 /* Values for Endian field above */

564 #define ACPI_UART_LITTLE_ENDIAN  0
565 #define ACPI_UART_BIG_ENDIAN     1

567 /* Values for DataBits field above */

569 #define ACPI_UART_5_DATA_BITS     0
570 #define ACPI_UART_6_DATA_BITS     1
571 #define ACPI_UART_7_DATA_BITS     2
572 #define ACPI_UART_8_DATA_BITS     3
573 #define ACPI_UART_9_DATA_BITS     4

575 /* Values for StopBits field above */

577 #define ACPI_UART_NO_STOP_BITS     0
578 #define ACPI_UART_1_STOP_BIT       1
579 #define ACPI_UART_1P5_STOP_BITS    2
580 #define ACPI_UART_2_STOP_BITS      3

582 /* Values for FlowControl field above */

584 #define ACPI_UART_FLOW_CONTROL_NONE 0
585 #define ACPI_UART_FLOW_CONTROL_HW   1
586 #define ACPI_UART_FLOW_CONTROL_XON_XOFF 2

```

```

588 /* Values for Parity field above */

590 #define ACPI_UART_PARITY_NONE      0
591 #define ACPI_UART_PARITY_EVEN      1
592 #define ACPI_UART_PARITY_ODD       2
593 #define ACPI_UART_PARITY_MARK      3
594 #define ACPI_UART_PARITY_SPACE     4

596 /* Values for LinesEnabled bitfield above */

598 #define ACPI_UART_CARRIER_DETECT  (1<<2)
599 #define ACPI_UART_RING_INDICATOR  (1<<3)
600 #define ACPI_UART_DATA_SET_READY  (1<<4)
601 #define ACPI_UART_DATA_TERMINAL_READY (1<<5)
602 #define ACPI_UART_CLEAR_TO_SEND   (1<<6)
603 #define ACPI_UART_REQUEST_TO_SEND (1<<7)

606 /* ACPI_RESOURCE_TYPES */

608 #define ACPI_RESOURCE_TYPE_IRQ      0
609 #define ACPI_RESOURCE_TYPE_DMA      1
610 #define ACPI_RESOURCE_TYPE_START_DEPENDENT 2
611 #define ACPI_RESOURCE_TYPE_END_DEPENDENT 3
612 #define ACPI_RESOURCE_TYPE_IO       4
613 #define ACPI_RESOURCE_TYPE_FIXED_IO  5
614 #define ACPI_RESOURCE_TYPE_VENDOR    6
615 #define ACPI_RESOURCE_TYPE_END_TAG    7
616 #define ACPI_RESOURCE_TYPE_MEMORY24  8
617 #define ACPI_RESOURCE_TYPE_MEMORY32  9
618 #define ACPI_RESOURCE_TYPE_FIXED_MEMORY32 10
619 #define ACPI_RESOURCE_TYPE_ADDRESS16  11
620 #define ACPI_RESOURCE_TYPE_ADDRESS32  12
621 #define ACPI_RESOURCE_TYPE_ADDRESS64  13
622 #define ACPI_RESOURCE_TYPE_EXTENDED_ADDRESS64 14 /* ACPI 3.0 */
623 #define ACPI_RESOURCE_TYPE_EXTENDED_IRQ 15
624 #define ACPI_RESOURCE_TYPE_GENERIC_REGISTER 16
625 #define ACPI_RESOURCE_TYPE_GPIO       17 /* ACPI 5.0 */
626 #define ACPI_RESOURCE_TYPE_FIXED_DMA  18 /* ACPI 5.0 */
627 #define ACPI_RESOURCE_TYPE_SERIAL_BUS  19 /* ACPI 5.0 */
628 #define ACPI_RESOURCE_TYPE_MAX        19

630 /* Master union for resource descriptors */

632 typedef union acpi_resource_data
633 {
634     ACPI_RESOURCE_IRQ      Irq;
635     ACPI_RESOURCE_DMA       Dma;
636     ACPI_RESOURCE_START_DEPENDENT StartDpf;
637     ACPI_RESOURCE_IO        Io;
638     ACPI_RESOURCE_FIXED_IO  FixedIo;
639     ACPI_RESOURCE_FIXED_DMA FixedDma;
640     ACPI_RESOURCE_VENDOR    Vendor;
641     ACPI_RESOURCE_VENDOR_TYPED VendorTyped;
642     ACPI_RESOURCE_END_TAG    EndTag;
643     ACPI_RESOURCE_MEMORY24    Memory24;
644     ACPI_RESOURCE_MEMORY32    Memory32;
645     ACPI_RESOURCE_FIXED_MEMORY32 FixedMemory32;
646     ACPI_RESOURCE_ADDRESS16    Address16;
647     ACPI_RESOURCE_ADDRESS32    Address32;
648     ACPI_RESOURCE_ADDRESS64    Address64;
649     ACPI_RESOURCE_EXTENDED_ADDRESS64 ExtAddress64;
650     ACPI_RESOURCE_EXTENDED_IRQ ExtendedIrq;
651     ACPI_RESOURCE_GENERIC_REGISTER GenericReg;
652     ACPI_RESOURCE_GPIO        Gpio;
653     ACPI_RESOURCE_I2C_SERIALBUS I2cSerialBus;

```

```
654     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__ */
```

new/usr/src/common/acpica/include/acstruct.h

1

```
*****
11317 Fri Jan 17 07:31:02 2014
new/usr/src/common/acpica/include/acstruct.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acstruct.h - Internal structs
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACSTRUCT_H__
45 #define __ACSTRUCT_H__

47 /* acpisrc:StructDefs -- for acpisrc conversion */

49 /*****
50 *
51 * Tree walking typedefs and structs
52 *
53 *****/

56 /*
57 * Walk state - current state of a parse tree walk. Used for both a leisurely
58 * stroll through the tree (for whatever reason), and for control method
```

new/usr/src/common/acpica/include/acstruct.h

2

```
59 * execution.
60 */
61 #define ACPI_NEXT_OP_DOWNWARD 1
62 #define ACPI_NEXT_OP_UPWARD 2

64 /*
65 * Groups of definitions for WalkType used for different implementations of
66 * walkers (never simultaneously) - flags for interpreter:
67 */
68 #define ACPI_WALK_NON_METHOD 0
69 #define ACPI_WALK_METHOD 0x01
70 #define ACPI_WALK_METHOD_RESTART 0x02

72 /* Flags for iASL compiler only */

74 #define ACPI_WALK_CONST_REQUIRED 0x10
75 #define ACPI_WALK_CONST_OPTIONAL 0x20

78 typedef struct acpi_walk_state
79 {
80     struct acpi_walk_state *Next; /* Next WalkState in list */
81     UINT8 DescriptorType; /* To differentiate vari */
82     UINT8 WalkType; /* Info about NextOp */
83     UINT16 Opcode; /* Current AML opcode */
84     UINT8 NextOpInfo; /* Info about NextOp */
85     UINT8 NumOperands; /* Stack pointer for Ope */
86     UINT8 OperandIndex; /* Index into operand st */
87     ACPI_OWNER_ID OwnerId; /* Owner of objects crea */
88     BOOLEAN LastPredicate; /* Result of last predic */
89     UINT8 CurrentResult;
90     UINT8 ReturnUsed;
91     UINT8 ScopeDepth;
92     UINT8 PassNumber; /* Parse pass during tab */
93     UINT8 ResultSize; /* Total elements for th */
94     UINT8 ResultCount; /* Current number of occ */
95     UINT32 AmlOffset;
96     UINT32 ArgTypes;
97     UINT32 MethodBreakpoint; /* For single stepping */
98     UINT32 UserBreakpoint; /* User AML breakpoint */
99     UINT32 ParseFlags;

101     ACPI_PARSE_STATE ParserState; /* Current state of pars */
102     UINT32 PrevArgTypes;
103     UINT32 ArgCount; /* push for fixed or var

105     struct acpi_namespace_node Arguments[ACPI_METHOD_NUM_ARGS]; /* C
106     struct acpi_namespace_node LocalVariables[ACPI_METHOD_NUM_LOCALS]; /* C
107     union acpi_operand_object *Operands[ACPI_OBJ_NUM_OPERANDS + 1]; /* O
108     union acpi_operand_object **Params;

110     UINT8 *AmlLastWhile;
111     union acpi_operand_object **CallerReturnDesc;
112     ACPI_GENERIC_STATE *ControlState; /* List of control state
113     struct acpi_namespace_node *DeferredNode; /* Used when executing d
114     union acpi_operand_object *ImplicitReturnObj;
115     struct acpi_namespace_node *MethodCallNode; /* Called method Node*/
116     ACPI_PARSE_OBJECT *MethodCallOp; /* MethodCall Op if runn
117     union acpi_operand_object *MethodDesc; /* Method descriptor if
118     struct acpi_namespace_node *MethodNode; /* Method node if runnin
119     ACPI_PARSE_OBJECT *Op; /* Current parser op */
120     const ACPI_OPCODE_INFO *OpInfo; /* Info on current opcod
121     ACPI_PARSE_OBJECT *Origin; /* Start of walk [Obsole
122     union acpi_operand_object *ResultObj;
123     ACPI_GENERIC_STATE *Results; /* Stack of accumulated
124     union acpi_operand_object *ReturnDesc; /* Return object, if any
```

new/usr/src/common/acpica/include/acstruct.h

3

```
125     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.
199  * Structure used to pass object evaluation parameters.
200  * Purpose is to reduce CPU stack use.
201  */
202 typedef struct acpi_evaluate_info
203 {
204     /* The first 3 elements are passed by the caller to AcpiNsEvaluate */
205     ACPI_NAMESPACE_NODE      *PrefixNode;
206     char                      *Pathname;
207     ACPI_OPERAND_OBJECT      *ObjDesc;
208     ACPI_OPERAND_OBJECT      **Parameters;
209     ACPI_NAMESPACE_NODE      *ResolvedNode;
210     ACPI_OPERAND_OBJECT      *ReturnObject;
211     UINT8                    ParamCount;
212     UINT8                    PassNumber;
213     UINT8                    ReturnObjectType;
214     UINT8                    Flags;
215
216     ACPI_NAMESPACE_NODE      *PrefixNode; /* Input: starting node
217     char                      *RelativePathname; /* Input: path relative
218     ACPI_OPERAND_OBJECT      **Parameters; /* Input: argument list
219
220     ACPI_NAMESPACE_NODE      *Node; /* Resolved node (Prefix
221     ACPI_OPERAND_OBJECT      *ObjDesc; /* Object attached to th
222     char                      *FullPathname; /* Full pathname of the
223
224     const ACPI_PREDEFINED_INFO *Predefined; /* Used if Node is a pre
225     ACPI_OPERAND_OBJECT      *ReturnObject; /* Object returned from
226     union acpi_operand_object *ParentPackage; /* Used if return object
227
228     UINT32                    ReturnFlags; /* Used for return value
229     UINT32                    ReturnBtype; /* Bitmapped type of the
230     UINT16                    ParamCount; /* Count of the input ar
231     UINT8                    PassNumber; /* Parser pass number */
232     UINT8                    ReturnObjectType; /* Object type of the re
233     UINT8                    NodeFlags; /* Same as Node->Flags *
234     UINT8                    Flags; /* General flags */
235
236 } 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 {
```

new/usr/src/common/acpica/include/acstruct.h

4

```
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_
```

new/usr/src/common/acpica/include/actables.h

1

```
*****
4978 Fri Jan 17 07:31:02 2014
new/usr/src/common/acpica/include/actables.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: actables.h - ACPI table management
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACTABLES_H__
45 #define __ACTABLES_H__

48 ACPI_STATUS
49 AcpiAllocateRootTable (
50     UINT32                InitialTableCount);

52 /*
53 * tbxfroot - Root pointer utilities
54 */
55 ACPI_STATUS
56 AcpiTbValidateRsdp (
57     ACPI_TABLE_RSDP        *Rsdp);
```

new/usr/src/common/acpica/include/actables.h

2

```
59 UINT8 *
60 AcpiTbScanMemoryForRsdp (
61     UINT8                *StartAddress,
62     UINT32                Length);

65 /*
66 * tbfadt - FADT parse/convert/validate
67 */
68 void
69 AcpiTbParseFadt (
70     UINT32                TableIndex);

72 void
73 AcpiTbCreateLocalFadt (
74     ACPI_TABLE_HEADER        *Table,
75     UINT32                Length);

78 /*
79 * tbfind - find ACPI table
80 */
81 ACPI_STATUS
82 AcpiTbFindTable (
83     char                *Signature,
84     char                *OemId,
85     char                *OemTableId,
86     UINT32                *TableIndex);

89 /*
90 * tbinstal - Table removal and deletion
91 */
92 ACPI_STATUS
93 AcpiTbResizeRootTableList (
94     void);

96 ACPI_STATUS
97 AcpiTbVerifyTable (
98     ACPI_TABLE_DESC        *TableDesc);

100 ACPI_TABLE_HEADER *
101 AcpiTbTableOverride (
102     ACPI_TABLE_HEADER        *TableHeader,
103     ACPI_TABLE_DESC        *TableDesc);

105 ACPI_STATUS
106 AcpiTbAddTable (
107     ACPI_TABLE_DESC        *TableDesc,
108     UINT32                *TableIndex);

110 ACPI_STATUS
111 AcpiTbStoreTable (
112     ACPI_PHYSICAL_ADDRESS    Address,
113     ACPI_TABLE_HEADER        *Table,
114     UINT32                Length,
115     UINT8                Flags,
116     UINT32                *TableIndex);

118 void
119 AcpiTbDeleteTable (
120     ACPI_TABLE_DESC        *TableDesc);

122 void
123 AcpiTbTerminate (
124     void);
```

new/usr/src/common/acpica/include/actables.h

3

```
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 AcpiTbCheckDsdtdHeader (
181     void);

183 ACPI_TABLE_HEADER *
184 AcpiTbCopyDsdtd (
185     UINT32                TableIndex);

187 void
188 AcpiTbInstallTable (
189     ACPI_PHYSICAL_ADDRESS Address,
190     char                  *Signature,
```

new/usr/src/common/acpica/include/actables.h

4

```
191     UINT32                TableIndex);

193 ACPI_STATUS
194 AcpiTbParseRootTable (
195     ACPI_PHYSICAL_ADDRESS RsdpaAddress);

197 #endif /* __ACTABLES_H__ */
```

new/usr/src/common/acpica/include/actbl.h

1

```
*****
22475 Fri Jan 17 07:31:02 2014
new/usr/src/common/acpica/include/actbl.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Name: actbl.h - Basic ACPI Table Definitions
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACTBL_H__
45 #define __ACTBL_H__
46
47 /*****
48 *
49 * Fundamental ACPI tables
50 *
51 * This file contains definitions for the ACPI tables that are directly consumed
52 * by ACPICA. All other tables are consumed by the OS-dependent ACPI-related
53 * device drivers and other OS support code.
54 *
55 * The RSDP and FACS do not use the common ACPI table header. All other ACPI
56 * tables use the header.
57 */
```

new/usr/src/common/acpica/include/actbl.h

2

```
58 *
59 *****/
60
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_DSDT "DSDT" /* Differentiated System Description
67 #define ACPI_SIG_FADT "FACP" /* Fixed ACPI Description Table */
68 #define ACPI_SIG_FACS "FACS" /* Firmware ACPI Control Structure */
69 #define ACPI_SIG_PSDT "PSDT" /* Persistent System Description Table
70 #define ACPI_SIG_RSDP "RSD PTR " /* Root System Description Pointer */
71 #define ACPI_SIG_RSDT "RSDT" /* Root System Description Table */
72 #define ACPI_SIG_XSDT "XSDT" /* Extended System Description Table
73 #define ACPI_SIG_SSST "SSST" /* Secondary System Description Table
74 #define ACPI_SIG_RSMP "RSMP" /* Short name for RSDP, not signature
75
76
77 /*
78 * All tables and structures must be byte-packed to match the ACPI
79 * specification, since the tables are provided by the system BIOS
80 */
81 #pragma pack(1)
82
83 /*
84 * Note: C bitfields are not used for this reason:
85 * "Bitfields are great and easy to read, but unfortunately the C language
86 * does not specify the layout of bitfields in memory, which means they are
87 * essentially useless for dealing with packed data in on-disk formats or
88 * binary wire protocols." (Or ACPI tables and buffers.) "If you ask me,
89 * this decision was a design error in C. Ritchie could have picked an order
90 * and stuck with it." Norman Ramsey.
91 * See http://stackoverflow.com/a/1053662/41661
92 * Note about bitfields: The UINT8 type is used for bitfields in ACPI tables.
93 * This is the only type that is even remotely portable. Anything else is not
94 * portable, so do not use any other bitfield types.
95 */
96
97 /*****
98 *
99 * Master ACPI Table Header. This common header is used by all ACPI tables
100 * except the RSDP and FACS.
101 *
102 *****/
103
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
109     UINT8 Revision; /* ACPI Specific
110     UINT8 Checksum; /* To make sum o
111     char OemId[ACPI_OEM_ID_SIZE]; /* ASCII OEM ide
112     char OemTableId[ACPI_OEM_TABLE_ID_SIZE]; /* ASCII OEM tab
113     UINT32 OemRevision; /* OEM revision
114     char AslCompilerId[ACPI_NAME_SIZE]; /* ASCII ASL com
115     UINT32 AslCompilerRevision; /* ASL compiler
116 } ACPI_TABLE_HEADER;
117
118 /*****
```

```

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
125 * If misaligned access is not supported by the hardware, accesses to the
126 * 64-bit Address field must be performed with care.
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
268     UINT8      AcpiEnable;    /* Value to write to smi_cmd to
269     UINT8      AcpiDisable;   /* Value to write to smi_cmd to
270     UINT8      S4BiosRequest; /* Value to write to SMI_CMD to
271     UINT8      PstateControl; /* Processor performance state c
272     UINT32      PmlaEventBlock; /* 32-bit port address of Power
273     UINT32      PmlbEventBlock; /* 32-bit port address of Power
274     UINT32      PmlaControlBlock; /* 32-bit port address of Power
275     UINT32      PmlbControlBlock; /* 32-bit port address of Power
276     UINT32      Pm2ControlBlock; /* 32-bit port address of Power
277     UINT32      PmTimerBlock;   /* 32-bit port address of Power
278     UINT32      Gpe0Block;      /* 32-bit port address of Genera
279     UINT32      Gpe1Block;      /* 32-bit port address of Genera
280     UINT32      PmlaEventBlock; /* 32-bit Port address of Power
281     UINT32      PmlbEventBlock; /* 32-bit Port address of Power
282     UINT32      PmlaControlBlock; /* 32-bit Port address of Power
283     UINT32      PmlbControlBlock; /* 32-bit Port address of Power
284     UINT32      Pm2ControlBlock; /* 32-bit Port address of Power
285     UINT32      PmTimerBlock;   /* 32-bit Port address of Power
286     UINT32      Gpe0Block;      /* 32-bit Port address of Genera
287     UINT32      Gpe1Block;      /* 32-bit Port address of Genera
288     UINT8      PmlEventLength; /* Byte Length of ports at PmlxE
289     UINT8      PmlControlLength; /* Byte Length of ports at PmlxC
290     UINT8      Pm2ControlLength; /* Byte Length of ports at Pm2Co
291     UINT8      PmTimerLength;   /* Byte Length of ports at PmTim
292     UINT8      Gpe0BlockLength; /* Byte Length of ports at Gpe0B
293     UINT8      Gpe1BlockLength; /* Byte Length of ports at Gpe1B
294     UINT8      Gpe1Base;        /* Offset in GPE number space wh
295     UINT8      CstControl;      /* Support for the _CST object a
296     UINT8      CstControl;      /* Support for the _CST object a
297     UINT16     C2Latency;       /* Worst case HW latency to ente
298     UINT16     C3Latency;       /* Worst case HW latency to ente
299     UINT16     FlushSize;       /* Processor memory cache line w
300     UINT16     FlushSize;       /* Processor's memory cache line
301     UINT16     FlushStride;     /* Number of flush strides that
302     UINT8      DutyOffset;      /* Processor duty cycle index in
303     UINT8      DutyOffset;      /* Processor duty cycle index in
304     UINT8      DutyWidth;       /* Processor duty cycle value bi
305     UINT8      DayAlarm;        /* Index to day-of-month alarm i
306     UINT8      MonthAlarm;      /* Index to month-of-year alarm
307     UINT8      Century;        /* Index to century in RTC CMOS
308     UINT16     BootFlags;       /* IA-PC Boot Architecture Flags
309     UINT8      Reserved;        /* Reserved, must be zero */
310     UINT32     Flags;           /* Miscellaneous flag bits (see
311     ACPI_GENERIC_ADDRESS      ResetRegister; /* 64-bit address of the Reset r
312     UINT8      ResetValue;      /* Value to write to the ResetRe
313     UINT8      Reserved4[3];      /* Reserved, must be zero */
314     UINT64     XFacs;           /* 64-bit physical address of FA
315     UINT64     XDsdT;          /* 64-bit physical address of DS
316     ACPI_GENERIC_ADDRESS      XPmlaEventBlock; /* 64-bit Extended Power Mgt la

```



```

303     ACPI_GENERIC_ADDRESS    XPmlbEventBlock;    /* 64-bit Extended Power Mgt 1b
304     ACPI_GENERIC_ADDRESS    XPmlaControlBlock;  /* 64-bit Extended Power Mgt 1a
305     ACPI_GENERIC_ADDRESS    XPmlbControlBlock;  /* 64-bit Extended Power Mgt 1b
306     ACPI_GENERIC_ADDRESS    XPm2ControlBlock;   /* 64-bit Extended Power Mgt 2 C
307     ACPI_GENERIC_ADDRESS    XPmTimerBlock;      /* 64-bit Extended Power Mgt Tim
308     ACPI_GENERIC_ADDRESS    XGpe0Block;         /* 64-bit Extended General Purpo
309     ACPI_GENERIC_ADDRESS    XGpelBlock;         /* 64-bit Extended General Purpo
310     ACPI_GENERIC_ADDRESS    SleepControl;        /* 64-bit Sleep Control register
311     ACPI_GENERIC_ADDRESS    SleepStatus;        /* 64-bit Sleep Status register

```

```

313 } ACPI_TABLE_FADT;

```

```

316 /* Masks for FADT Boot Architecture Flags (BootFlags) [Vx]=Introduced in this FA
305 /* Masks for FADT Boot Architecture Flags (BootFlags) */

```

```

318 #define ACPI_FADT_LEGACY_DEVICES    (1)          /* 00: [V2] System has LPC or IS
319 #define ACPI_FADT_8042              (1<<1)      /* 01: [V3] System has an 8042 c
320 #define ACPI_FADT_NO_VGA            (1<<2)      /* 02: [V4] It is not safe to pr
321 #define ACPI_FADT_NO_MSI            (1<<3)      /* 03: [V4] Message Signaled Int
322 #define ACPI_FADT_NO_ASPM           (1<<4)      /* 04: [V4] PCIE ASPM control mu
323 #define ACPI_FADT_NO_CMOS_RTC       (1<<5)      /* 05: [V5] No CMOS real-time cl

```

```

325 /* Masks for FADT flags */

```

```

327 #define ACPI_FADT_WBINVD            (1)          /* 00: [V1] The WBINVD instructi
328 #define ACPI_FADT_WBINVD_FLUSH      (1<<1)      /* 01: [V1] WBINVD flushes but d
315 #define ACPI_FADT_WBINVD           (1)          /* 00: [V1] The wbinvd instructi
316 #define ACPI_FADT_WBINVD_FLUSH      (1<<1)      /* 01: [V1] wbinvd flushes but d
329 #define ACPI_FADT_C1_SUPPORTED      (1<<2)      /* 02: [V1] All processors suppo
330 #define ACPI_FADT_C2_MP_SUPPORTED   (1<<3)      /* 03: [V1] C2 state works on MP
331 #define ACPI_FADT_POWER_BUTTON      (1<<4)      /* 04: [V1] Power button is hand
332 #define ACPI_FADT_SLEEP_BUTTON      (1<<5)      /* 05: [V1] Sleep button is hand
333 #define ACPI_FADT_FIXED_RTC          (1<<6)      /* 06: [V1] RTC wakeup status is
321 #define ACPI_FADT_FIXED_RTC         (1<<6)      /* 06: [V1] RTC wakeup status no
334 #define ACPI_FADT_S4_RTC_WAKE       (1<<7)      /* 07: [V1] RTC alarm can wake s
335 #define ACPI_FADT_32BIT_TIMER        (1<<8)      /* 08: [V1] ACPI timer width is
336 #define ACPI_FADT_DOCKING_SUPPORTED  (1<<9)      /* 09: [V1] Docking supported */
337 #define ACPI_FADT_RESET_REGISTER     (1<<10)     /* 10: [V2] System reset via the
338 #define ACPI_FADT_SEALED_CASE        (1<<11)     /* 11: [V3] No internal expansio
339 #define ACPI_FADT_HEADLESS           (1<<12)     /* 12: [V3] No local video capab
340 #define ACPI_FADT_SLEEP_TYPE         (1<<13)     /* 13: [V3] Must execute native
341 #define ACPI_FADT_PCI_EXPRESS_WAKE   (1<<14)     /* 14: [V4] System supports PCIE
342 #define ACPI_FADT_PLATFORM_CLOCK     (1<<15)     /* 15: [V4] OSPM should use plat
343 #define ACPI_FADT_S4_RTC_VALID       (1<<16)     /* 16: [V4] Contents of RTC_STS
344 #define ACPI_FADT_REMOTE_POWER_ON    (1<<17)     /* 17: [V4] System is compatible
345 #define ACPI_FADT_APIC_CLUSTER       (1<<18)     /* 18: [V4] All local APICs must
346 #define ACPI_FADT_APIC_PHYSICAL      (1<<19)     /* 19: [V4] All local xAPICs mus
347 #define ACPI_FADT_HW_REDUCED         (1<<20)     /* 20: [V5] ACPI hardware is not
348 #define ACPI_FADT_LOW_POWER_S0       (1<<21)     /* 21: [V5] S0 power savings are

```

```

351 /* Values for PreferredProfile (Preferred Power Management Profiles) */
337 /* Values for PreferredProfile (Preferred Power Management Profiles) */

```

```

353 enum AcpiPreferredPmProfiles
339 enum AcpiPreferredPmProfiles
354 {
355     PM_UNSPECIFIED          = 0,
356     PM_DESKTOP              = 1,
357     PM_MOBILE               = 2,
358     PM_WORKSTATION          = 3,
359     PM_ENTERPRISE_SERVER    = 4,
360     PM_SOHO_SERVER          = 5,
361     PM_APPLIANCE_PC         = 6,
362     PM_PERFORMANCE_SERVER   = 7,

```

```

363     PM_TABLET                = 8
347     PM_APPLIANCE_PC         = 6
364 };

```

```

366 /* Values for SleepStatus and SleepControl registers (V5 FADT) */

```

```

368 #define ACPI_X_WAKE_STATUS          0x80
369 #define ACPI_X_SLEEP_TYPE_MASK      0x1C
370 #define ACPI_X_SLEEP_TYPE_POSITION 0x02
371 #define ACPI_X_SLEEP_ENABLE         0x20

```

```

374 /* Reset to default packing */

```

```

376 #pragma pack()

```

```

379 /*
380 * Internal table-related structures
381 */
382 typedef union acpi_name_union
383 {
384     UINT32                Integer;
385     char                   Ascii[4];
387 } ACPI_NAME_UNION;

```

```

390 /* Internal ACPI Table Descriptor. One per ACPI table. */

```

```

392 typedef struct acpi_table_desc
393 {
394     ACPI_PHYSICAL_ADDRESS    Address;
395     ACPI_TABLE_HEADER         *Pointer;
396     UINT32                   Length; /* Length fixed at 32 bits (fixe
373     UINT32                   Length; /* Length fixed at 32 bits */
397     ACPI_NAME_UNION          Signature;
398     ACPI_OWNER_ID            OwnerId;
399     UINT8                    Flags;

```

```

401 } ACPI_TABLE_DESC;

```

```

403 /* Masks for Flags field above */

```

```

405 #define ACPI_TABLE_ORIGIN_UNKNOWN    (0)
406 #define ACPI_TABLE_ORIGIN_MAPPED    (1)
407 #define ACPI_TABLE_ORIGIN_ALLOCATED (2)
408 #define ACPI_TABLE_ORIGIN_OVERRIDE  (4)
409 #define ACPI_TABLE_ORIGIN_MASK      (7)
410 #define ACPI_TABLE_IS_LOADED        (8)

```

```

413 /*
414 * Get the remaining ACPI tables
415 */
416 #include "actbl1.h"
417 #include "actbl2.h"
418 #include "actbl3.h"

```

```

420 /* Macros used to generate offsets to specific table fields */

```

```

422 #define ACPI_FADT_OFFSET(f)          (UINT16) ACPI_OFFSET (ACPI_TABLE_FADT, f)
398 #define ACPI_FADT_OFFSET(f)          (UINT8) ACPI_OFFSET (ACPI_TABLE_FADT, f)

```

```

424 /*
425 * Sizes of the various flavors of FADT. We need to look closely

```

```
426 * 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
437 *
438 *      FADT V1  size: 0x74
439 *      FADT V2  size: 0x84
440 *      FADT V3+ size: 0xF4
441 *
442 */
443 #define ACPI_FADT_V1_SIZE      (UINT32) (ACPI_FADT_OFFSET (Flags) + 4)
444 #define ACPI_FADT_V2_SIZE      (UINT32) (ACPI_FADT_OFFSET (Reserved4[0]) + 3)
445 #define ACPI_FADT_V3_SIZE      (UINT32) (ACPI_FADT_OFFSET (SleepControl))
446 #define ACPI_FADT_V5_SIZE      (UINT32) (sizeof (ACPI_TABLE_FADT))
447 #define ACPI_FADT_V3_SIZE      (UINT32) (sizeof (ACPI_TABLE_FADT))
448
449 #endif /* __ACTBL_H__ */
```

new/usr/src/common/acpica/include/actbl1.h

1

```
*****
36382 Fri Jan 17 07:31:03 2014
new/usr/src/common/acpica/include/actbl1.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Name: actbl1.h - Additional ACPI table definitions
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACTBL1_H__
45 #define __ACTBL1_H__
46
47 /*****
48 *
49 * Additional ACPI Tables (1)
50 *
51 * These tables are not consumed directly by the ACPICA subsystem, but are
52 * included here to support device drivers and the AML disassembler.
53 *
54 * The tables in this file are fully defined within the ACPI specification.
55 *
56 *
57 *****/
```

new/usr/src/common/acpica/include/actbl1.h

2

```
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_ECDD "ECDD" /* 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 Table
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 */
76
77
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)
83
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
94 * Note about bitfields: The UINT8 type is used for bitfields in ACPI tables.
95 * This is the only type that is even remotely portable. Anything else is not
96 * portable, so do not use any other bitfield types.
97 */
98
99 /*****
100 *
101 * Common subtable headers
102 *****/
103
104 /* Generic subtable header (used in MADT, SRAT, etc.) */
105
106 typedef struct acpi_subtable_header
107 {
108     UINT8 Type;
109     UINT8 Length;
110 } ACPI_SUBTABLE_HEADER;
111
112 /*
113 * _____ unchanged portion omitted _____
114
115
116
117
118 /*****
119 *
120 * BERT - Boot Error Record Table (ACPI 4.0)
121 * Version 1
122 *
123 *****/
```

```

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
140     UINT64               Address;         /* Physical address 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
271     ACPI_EINJ_ACTION_RESERVED          = 8, /* 8 and greater are reserved */
272     ACPI_EINJ_TRIGGER_ERROR            = 0xFF /* Except for this value */
273 };

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 */
285     ACPI_EINJ_INSTRUCTION_RESERVED     = 5 /* 5 and greater are reserved */
286 };

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 */
528     UINT16               Device;
529     UINT16               Function;
530     UINT16               DeviceControl;

```

new/usr/src/common/acpica/include/actbl1.h

5

```
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 */
581     ACPI_HEST_NOTIFY_RESERVED    = 5, /* 5 and greater are reserved */
582 };
583 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 */
```

new/usr/src/common/acpica/include/actbl1.h

6

```
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 */
779     ACPI_MADT_TYPE_RESERVED        = 11 /* 11 and greater are reserved */
780 };
781 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 */
969 /* 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
```

new/usr/src/common/acpica/include/actbl2.h

1

```
*****
43681 Fri Jan 17 07:31:04 2014
new/usr/src/common/acpica/include/actbl2.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Name: actbl2.h - ACPI Table Definitions (tables not in ACPI spec)
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACTBL2_H__
45 #define __ACTBL2_H__
46
47 /*****
48 *
49 * Additional ACPI Tables (2)
50 *
51 * These tables are not consumed directly by the ACPI subsystem, but are
52 * included here to support device drivers and the AML disassembler.
53 *
54 * The tables in this file are defined by third-party specifications, and are
55 * not defined directly by the ACPI specification itself.
56 *
57 */
```

new/usr/src/common/acpica/include/actbl2.h

2

```
58 *****/
59
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_ASF "ASF!" /* Alert Standard Format table */
66 #define ACPI_SIG_BOOT "BOOT" /* Simple Boot Flag Table */
67 #define ACPI_SIG_CSRT "CSRT" /* Core System Resource Table */
68 #define ACPI_SIG_DBG2 "DBG2" /* Debug Port table type 2 */
69 #define ACPI_SIG_DBGP "DBGP" /* Debug Port table */
70 #define ACPI_SIG_DMAR "DMAR" /* DMA Remapping table */
71 #define ACPI_SIG_HPET "HPET" /* High Precision Event Timer table */
72 #define ACPI_SIG_IBFT "IBFT" /* iSCSI Boot Firmware Table */
73 #define ACPI_SIG_IVRS "IVRS" /* I/O Virtualization Reporting Structure */
74 #define ACPI_SIG_MCFG "MCFG" /* PCI Memory Mapped Configuration Table */
75 #define ACPI_SIG_MCHI "MCHI" /* Management Controller Host Interface Table */
76 #define ACPI_SIG_MTMR "MTMR" /* MID Timer table */
77 #define ACPI_SIG_SLIC "SLIC" /* Software Licensing Description Table */
78 #define ACPI_SIG_SPCR "SPCR" /* Serial Port Console Redirection Table */
79 #define ACPI_SIG_SPMI "SPMI" /* Server Platform Management Interface Table */
80 #define ACPI_SIG_TCPA "TCPA" /* Trusted Computing Platform Alliance Table */
81 #define ACPI_SIG_UEFI "UEFI" /* UEFI Boot Optimization Table */
82 #define ACPI_SIG_VRTC "VRTC" /* Virtual Real Time Clock Table */
83 #define ACPI_SIG_WAET "WAET" /* Windows ACPI Emulated devices Table */
84 #define ACPI_SIG_WDAT "WDAT" /* Watchdog Action Table */
85 #define ACPI_SIG_WDDT "WDDT" /* Watchdog Timer Description Table */
86 #define ACPI_SIG_WDRT "WDRT" /* Watchdog Resource Table */
87
88 #ifndef ACPI_UNDEFINED_TABLES
89 /*
90 * These tables have been seen in the field, but no definition has been found
91 */
92 #define ACPI_SIG_ATKG "ATKG" /* GMCH SCI table */
93 #define ACPI_SIG_GSCI "GSCI"
94 #define ACPI_SIG_IEIT "IEIT"
95 #endif
96
97 /*
98 * All tables must be byte-packed to match the ACPI specification, since
99 * the tables are provided by the system BIOS.
100 */
101 #pragma pack(1)
102
103 /*
104 * Note: C bitfields are not used for this reason:
105 *
106 * "Bitfields are great and easy to read, but unfortunately the C language
107 * does not specify the layout of bitfields in memory, which means they are
108 * essentially useless for dealing with packed data in on-disk formats or
109 * binary wire protocols." (Or ACPI tables and buffers.) "If you ask me,
110 * this decision was a design error in C. Ritchie could have picked an order
111 * and stuck with it." Norman Ramsey.
112 * See http://stackoverflow.com/a/1053662/41661
113 * Note about bitfields: The UINT8 type is used for bitfields in ACPI tables.
114 * This is the only type that is even remotely portable. Anything else is not
115 * portable, so do not use any other bitfield types.
116 */
117
118 /*
119 * ASF - Alert Standard Format table (Signature "ASF!")
120 * Revision 0x10
121 */
```

```

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;
131 /* unchanged portion omitted */
132
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;
289
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;
342
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
357
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;
375
376 typedef struct acpi_dbg2_header
377 {
378     UINT32                InfoOffset;
379     UINT32                InfoCount;
381 } ACPI_DBG2_HEADER;
383
384 /* Debug Device Information Subtable */
386 typedef struct acpi_dbg2_device
387 {
388     UINT8                 Revision;
389     UINT16                Length;
390     UINT8                 RegisterCount;    /* Number of BaseAddress registers */
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_dbgpp
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_DBGPP;
_____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 MRTCT 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

```

new/usr/src/common/acpica/include/actbl2.h

7

```
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__ */
```

new/usr/src/common/acpica/include/actbl3.h

1

```
*****
20829 Fri Jan 17 07:31:04 2014
new/usr/src/common/acpica/include/actbl3.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Name: actbl3.h - ACPI Table Definitions
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __ACTBL3_H__
45 #define __ACTBL3_H__

48 /*****
49 *
50 * Additional ACPI Tables (3)
51 *
52 * These tables are not consumed directly by the ACPI 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 *****/
```

new/usr/src/common/acpica/include/actbl3.h

2

```
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_BGRT          "BGRT"          /* Boot Graphics Resource Table */
66 #define ACPI_SIG_DRTM          "DRTM"          /* Dynamic Root of Trust for Measure
67 #define ACPI_SIG_FPDt          "FPDT"          /* Firmware Performance Data Table *
68 #define ACPI_SIG_GTDt          "GTDt"          /* Generic Timer Description Table *
69 #define ACPI_SIG_MPST          "MPST"          /* Memory Power State Table */
70 #define ACPI_SIG_PCCT          "PCCT"          /* Platform Communications Channel T
71 #define ACPI_SIG_PMTT          "PMTT"          /* Platform Memory Topology Table */
72 #define ACPI_SIG_RASF          "RASf"          /* RAS Feature table */
73 #define ACPI_SIG_TPM2          "TPM2"          /* Trusted Platform Module 2.0 H/W i

75 #define ACPI_SIG_S3PT          "S3PT"          /* S3 Performance (sub)Table */
76 #define ACPI_SIG_PCCS          "PCC"           /* PCC Shared Memory Region */

78 /* Reserved table signatures */

80 #define ACPI_SIG_MATR          "MATR"          /* Memory Address Translation Table
81 #define ACPI_SIG_MSDM          "MSDM"          /* Microsoft Data Management Table *
82 #define ACPI_SIG_WPBT          "WPBT"          /* Windows Platform Binary Table */

84 /*
85 * All tables must be byte-packed to match the ACPI specification, since
86 * the tables are provided by the system BIOS.
87 */
88 #pragma pack(1)

90 /*
91 * Note: C bitfields are not used for this reason:
92 *
93 * "Bitfields are great and easy to read, but unfortunately the C language
94 * does not specify the layout of bitfields in memory, which means they are
95 * essentially useless for dealing with packed data in on-disk formats or
96 * binary wire protocols." (Or ACPI tables and buffers.) "If you ask me,
97 * this decision was a design error in C. Ritchie could have picked an order
98 * and stuck with it." Norman Ramsey.
99 * See http://stackoverflow.com/a/1053662/41661
100 */

103 /*****
104 *
105 * BGRT - Boot Graphics Resource Table (ACPI 5.0)
106 * Version 1
107 *
108 *****/

110 typedef struct acpi_table_bgrt
111 {
112     ACPI_TABLE_HEADER    Header;          /* Common ACPI table header */
113     UINT16               Version;
114     UINT8                Status;
115     UINT8                ImageType;
116     UINT64               ImageAddress;
117     UINT32               ImageOffsetX;
118     UINT32               ImageOffsetY;

120 } ACPI_TABLE_BGRT;

123 /*****
124 *
125 * DRTM - Dynamic Root of Trust for Measurement table
```

```

126 *
127 *****/

129 typedef struct acpi_table_drtm
130 {
131     ACPI_TABLE_HEADER    Header;          /* Common ACPI table header */
132     UINT64               EntryBaseAddress;
133     UINT64               EntryLength;
134     UINT32               EntryAddress32;
135     UINT64               EntryAddress64;
136     UINT64               ExitAddress;
137     UINT64               LogAreaAddress;
138     UINT32               LogAreaLength;
139     UINT64               ArchDependentAddress;
140     UINT32               Flags;

142 } ACPI_TABLE_DRTM;

144 /* 1) Validated Tables List */

146 typedef struct acpi_drtm_vtl_list
147 {
148     UINT32               ValidatedTableListCount;

150 } ACPI_DRTM_VTL_LIST;

152 /* 2) Resources List */

154 typedef struct acpi_drtm_resource_list
155 {
156     UINT32               ResourceListCount;

158 } ACPI_DRTM_RESOURCE_LIST;

160 /* 3) Platform-specific Identifiers List */

162 typedef struct acpi_drtm_id_list
163 {
164     UINT32               IdListCount;

166 } ACPI_DRTM_ID_LIST;

169 /*****
170 *
171 * FPDT - Firmware Performance Data Table (ACPI 5.0)
172 *      Version 1
173 *
174 *****/

176 typedef struct acpi_table_fpdtd
177 {
178     ACPI_TABLE_HEADER    Header;          /* Common ACPI table header */

180 } ACPI_TABLE_FPDTD;

183 /* FPDT subtable header */

185 typedef struct acpi_fpdtd_header
186 {
187     UINT16               Type;
188     UINT8                Length;
189     UINT8                Revision;

191 } ACPI_FPDTD_HEADER;

```

```

193 /* Values for Type field above */

195 enum AcpiFpdtdType
196 {
197     ACPI_FPDTD_TYPE_BOOT           = 0,
198     ACPI_FPDTD_TYPE_S3PERF        = 1
199 };

202 /*
203 * FPDT subtables
204 */

206 /* 0: Firmware Basic Boot Performance Record */

208 typedef struct acpi_fpdtd_boot
209 {
210     ACPI_FPDTD_HEADER    Header;
211     UINT8                Reserved[4];
212     UINT64               ResetEnd;
213     UINT64               LoadStart;
214     UINT64               StartupStart;
215     UINT64               ExitServicesEntry;
216     UINT64               ExitServicesExit;

218 } ACPI_FPDTD_BOOT;

221 /* 1: S3 Performance Table Pointer Record */

223 typedef struct acpi_fpdtd_s3pt_ptr
224 {
225     ACPI_FPDTD_HEADER    Header;
226     UINT8                Reserved[4];
227     UINT64               Address;

229 } ACPI_FPDTD_S3PT_PTR;

232 /*
233 * S3PT - S3 Performance Table. This table is pointed to by the
234 * FPDT S3 Pointer Record above.
235 */
236 typedef struct acpi_table_s3pt
237 {
238     UINT8                Signature[4]; /* "S3PT" */
239     UINT32               Length;

241 } ACPI_TABLE_S3PT;

244 /*
245 * S3PT Subtables
246 */
247 typedef struct acpi_s3pt_header
248 {
249     UINT16               Type;
250     UINT8                Length;
251     UINT8                Revision;

253 } ACPI_S3PT_HEADER;

255 /* Values for Type field above */

257 enum AcpiS3ptType

```

```

258 {
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;
269 } ACPI_S3PT_RESUME;

272 typedef struct acpi_s3pt_suspend
273 {
274     ACPI_S3PT_HEADER      Header;
275     UINT64                 SuspendStart;
276     UINT64                 SuspendEnd;
277 } 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                 SecurePllInterrupt;
294     UINT32                 SecurePllFlags;
295     UINT32                 NonSecurePllInterrupt;
296     UINT32                 NonSecurePllFlags;
297     UINT32                 VirtualTimerInterrupt;
298     UINT32                 VirtualTimerFlags;
299     UINT32                 NonSecurePl2Interrupt;
300     UINT32                 NonSecurePl2Flags;
301 } 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 Channel Info */
333 } ACPI_TABLE_MPST;

337 /* Memory Platform Communication Channel Info */

339 typedef struct acpi_mpst_channel
340 {
341     ACPI_MPST_CHANNEL_INFO /* Platform Communication Channel Info */
342 } 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;
358 } 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;
374 } ACPI_MPST_POWER_STATE;

378 /* Physical Component ID Structure (follows POWER_STATE above) */

380 typedef struct acpi_mpst_component
381 {
382     UINT16                 ComponentId;
383 } 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;
393 } 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;
405 } 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;
428 } 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;
444 } 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;
477 } 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;
492 } 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;
507 } 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;
520 } 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;
549 } 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;
566 } ACPI_PMTT_CONTROLLER;

569 /* 1a: Proximity Domain substructure */

571 typedef struct acpi_pmtt_domain
572 {
573     UINT32               ProximityDomain;
574 } 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];
602 } 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;
617 } 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;
627 } 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;
640 } 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 {
655     ACPI_RASF_EXECUTE_RASF_COMMAND    = 1
656 };

658 /* Platform RAS Capabilities */

660 enum AcpiRasfCapabilities
661 {
662     ACPI_HW_PATROL_SCRUB_SUPPORTED    = 0,
663     ACPI_SW_PATROL_SCRUB_EXPOSED     = 1
664 };

666 /* Patrol Scrub Commands */

668 enum AcpiRasfPatrolScrubCommands
669 {
670     ACPI_RASF_GET_PATROL_PARAMETERS    = 1,
671     ACPI_RASF_START_PATROL_SCRUBBER    = 2,
672     ACPI_RASF_STOP_PATROL_SCRUBBER     = 3
673 };

675 /* Channel Command flags */

677 #define ACPI_RASF_GENERATE_SCI          (1<<15)

679 /* Status values */

681 enum AcpiRasfStatus
682 {
683     ACPI_RASF_SUCCESS                  = 0,
684     ACPI_RASF_NOT_VALID                = 1,
685     ACPI_RASF_NOT_SUPPORTED            = 2,
686     ACPI_RASF_BUSY                    = 3,
687     ACPI_RASF_FAILED                  = 4,
688     ACPI_RASF_ABORTED                 = 5,
689     ACPI_RASF_INVALID_DATA            = 6
690 };

692 /* Status flags */

694 #define ACPI_RASF_COMMAND_COMPLETE      (1)
695 #define ACPI_RASF_SCI_DOORBELL          (1<<1)
696 #define ACPI_RASF_ERROR                 (1<<2)
697 #define ACPI_RASF_STATUS                (0x1F<<3)

700 /*****
701  *
702  * TPM2 - Trusted Platform Module (TPM) 2.0 Hardware Interface Table
703  *      Version 3
704  *
705  * Conforms to "TPM 2.0 Hardware Interface Table (TPM2)" 29 November 2011
706  *
707  *****/

709 typedef struct acpi_table_tpm2
710 {
711     ACPI_TABLE_HEADER    Header;          /* Common ACPI table header */
712     UINT32               Flags;
713     UINT64               ControlAddress;
714     UINT32               StartMethod;

716 } ACPI_TABLE_TPM2;

718 /* Control area structure (not part of table, pointed to by ControlAddress) */

```

```

720 typedef struct acpi_tpm2_control
721 {
722     UINT32               Reserved;
723     UINT32               Error;
724     UINT32               Cancel;
725     UINT32               Start;
726     UINT64               InterruptControl;
727     UINT32               CommandSize;
728     UINT64               CommandAddress;
729     UINT32               ResponseSize;
730     UINT64               ResponseAddress;

732 } ACPI_TPM2_CONTROL;

735 /* Reset to default packing */

737 #pragma pack()

739 #endif /* __ACTBL3_H__ */

```



new/usr/src/common/acpica/include/actypes.h

1

```
*****
45240 Fri Jan 17 07:31:04 2014
new/usr/src/common/acpica/include/actypes.h
update to acpica-unix2-20140114
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 - 2014, 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 MERCHANTABILITY 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 #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
```

new/usr/src/common/acpica/include/actypes.h

2

```
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
99 * short     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     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
124 *
125 *****/

127 typedef unsigned char          BOOLEAN;
128 typedef unsigned char          UINT8;
129 typedef unsigned short         UINT16;
130 typedef COMPILER_DEPENDENT_UINT64  UINT64;
131 typedef COMPILER_DEPENDENT_INT64   INT64;

133 /*! [End] no source code translation !*/

135 /*
136 * Value returned by AcpiOsGetThreadId. There is no standard "thread_id"
137 * across operating systems or even the various UNIX systems. Since ACPICA
138 * only needs the thread ID as a unique thread identifier, we use a UINT64
139 * as the only common data type - it will accommodate any type of pointer or
140 * any type of integer. It is up to the host-dependent OSL to cast the
141 * native thread ID type to a UINT64 (in AcpiOsGetThreadId).
142 */
143 #define ACPI_THREAD_ID          UINT64

146 /*****
147 *
148 * Types specific to 64-bit targets
149 *
150 *****/

152 #if ACPI_MACHINE_WIDTH == 64

154 /*! [Begin] no source code translation (keep the typedefs as-is) */

156 typedef unsigned int          UINT32;
157 typedef int                   INT32;

159 /*! [End] no source code translation !*/

162 typedef INT64                 ACPI_NATIVE_INT;
163 typedef UINT64                ACPI_SIZE;
164 typedef UINT64                ACPI_IO_ADDRESS;
165 typedef UINT64                ACPI_PHYSICAL_ADDRESS;

167 #define ACPI_MAX_PTR           ACPI_UINT64_MAX
168 #define ACPI_SIZE_MAX          ACPI_UINT64_MAX
169 #define ACPI_USE_NATIVE_DIVIDE /* Has native 64-bit integer support */

171 /*
172 * In the case of the Itanium Processor Family (IPF), the hardware does not
173 * support misaligned memory transfers. Set the MISALIGNMENT_NOT_SUPPORTED flag
174 * to indicate that special precautions must be taken to avoid alignment faults.
175 * (IA64 or ia64 is currently used by existing compilers to indicate IPF.)
176 *
177 * Note: EM64T and other X86-64 processors support misaligned transfers,
178 * so there is no need to define this flag.
179 */
180 #if defined (__IA64__) || defined (__ia64__)
181 #define ACPI_MISALIGNMENT_NOT_SUPPORTED
182 #endif

185 /*****
186 *
187 * Types specific to 32-bit targets
188 */

```

```

189 *****/

191 #elif ACPI_MACHINE_WIDTH == 32

193 /*! [Begin] no source code translation (keep the typedefs as-is) */

195 typedef unsigned int          UINT32;
196 typedef int                   INT32;

198 /*! [End] no source code translation !*/

201 typedef INT32                 ACPI_NATIVE_INT;
202 typedef UINT32                ACPI_SIZE;
203 typedef UINT32                ACPI_IO_ADDRESS;
204 typedef UINT32                ACPI_PHYSICAL_ADDRESS;

206 #define ACPI_MAX_PTR           ACPI_UINT32_MAX
207 #define ACPI_SIZE_MAX          ACPI_UINT32_MAX

209 #else

211 /* ACPI_MACHINE_WIDTH must be either 64 or 32 */

213 #error unknown ACPI_MACHINE_WIDTH
214 #endif

217 /*****
218 *
219 * OS-dependent types
220 *
221 * If the defaults below are not appropriate for the host system, they can
222 * be defined in the OS-specific header, and this will take precedence.
223 *
224 *****/

226 /* Flags for AcpiOsAcquireLock/AcpiOsReleaseLock */

228 #ifndef ACPI_CPU_FLAGS
229 #define ACPI_CPU_FLAGS          ACPI_SIZE
230 #endif

232 /* Object returned from AcpiOsCreateCache */

234 #ifndef ACPI_CACHE_T
235 #ifdef ACPI_USE_LOCAL_CACHE
236 #define ACPI_CACHE_T            ACPI_MEMORY_LIST
237 #else
238 #define ACPI_CACHE_T            void *
239 #endif
240 #endif

242 /*
243 * Synchronization objects - Mutexes, Semaphores, and SpinLocks
244 */
245 #if (ACPI_MUTEX_TYPE == ACPI_BINARY_SEMAPHORE)
246 /*
247 * These macros are used if the host OS does not support a mutex object.
248 * Map the OSL Mutex interfaces to binary semaphores.
249 */
250 #define ACPI_MUTEX              ACPI_SEMAPHORE
251 #define AcpiOsCreateMutex(OutHandle)  AcpiOsCreateSemaphore(1, 1, OutHandle)
252 #define AcpiOsDeleteMutex(Handle)      (void) AcpiOsDeleteSemaphore(Handle)
253 #define AcpiOsAcquireMutex(Handle, Time)  AcpiOsWaitSemaphore(Handle, 1, Time)
254 #define AcpiOsReleaseMutex(Handle)      (void) AcpiOsSignalSemaphore(Handle, 1)

```

```

255 #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
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 the 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.
313  * All ACPICA functions that are available to the rest of the kernel are
314  * tagged with this macro which can be defined as appropriate for the host.
315  */
316 #ifndef ACPI_EXPORT_SYMBOL_INIT
317 #define ACPI_EXPORT_SYMBOL_INIT(Symbol)
318 #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 #ifndef 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)
355 #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/PPN 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
396 #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      100000L
439 #define ACPI_100NSEC_PER_SEC       100000000L

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) (Integer))
489 #define ACPI_HIBYTE(Integer)        ((UINT8) ((Integer) >> 8))
490 #define ACPI_LOWORD(Integer)        ((UINT16) (Integer))
491 #define ACPI_HIWORD(Integer)        ((UINT16) ((Integer) >> 16))
492 #define ACPI_LODWORD(Integer64)     ((UINT32) (Integer64))
493 #define ACPI_HIWORD(Integer64)     ((UINT32) ((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
611 #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, Ref0
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
667 * a Namespace node, only in an ACPI_OPERAND_OBJECT
668 */
669 #define ACPI_TYPE_LOCAL_EXTRA 0x1C
670 #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_XRUPRT_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
772 #define ACPI_MAX_SYS_NOTIFY 0x7f

773 #define ACPI_SYSTEM_HANDLER_LIST 0 /* Used as index, must be SYSTEM_NOTIFY */
774 #define ACPI_DEVICE_HANDLER_LIST 1 /* Used as index, must be DEVICE_NOTIFY */

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
794 #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 address
801 * space type of ACPI_ADR_SPACE_DATA_TABLE.
802 */
803 #define ACPI_ADR_SPACE_DATA_TABLE (ACPI_ADR_SPACE_TYPE) 0x7E /* Internal table */
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;
875 } ACPI_SLEEP_FUNCTIONS;

877 /*
878 * External ACPI object definition
879 */

881 /*
882 * Note: Type == ACPI_TYPE_ANY (0) is used to indicate a NULL package element
883 * or an unresolved named reference.
884 */
885 typedef union acpi_object
886 {
887     struct
888     {
889         ACPI_OBJECT_TYPE Type; /* See definition of AcpiNsType for
890                                */
891         struct
892         {
893             ACPI_OBJECT_TYPE Type; /* ACPI_TYPE_INTEGER */
894             UINT64 Value; /* The actual number */
895         } 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
959 #define ACPI_ALLOCATE_BUFFER (ACPI_SIZE) (-1)
960 #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);

```



new/usr/src/common/acpica/include/actypes.h

17

1097 /\* Table Event handler (Load, LoadTable, etc.) and types \*/

```
1099 typedef
1100 ACPI_STATUS (*ACPI_TABLE_HANDLER) (
1101     UINT32          Event,
1102     void            *Table,
1103     void            *Context);
```

```
1105 #define ACPI_TABLE_LOAD      0x0
1106 #define ACPI_TABLE_UNLOAD    0x1
1107 #define ACPI_NUM_TABLE_EVENTS 2
```

1110 /\* Address Spaces (For Operation Regions) \*/

```
1112 typedef
1113 ACPI_STATUS (*ACPI_ADR_SPACE_HANDLER) (
1114     UINT32          Function,
1115     ACPI_PHYSICAL_ADDRESS Address,
1116     UINT32          BitWidth,
1117     UINT64          *Value,
1118     void            *HandlerContext,
1119     void            *RegionContext);
```

```
1121 #define ACPI_DEFAULT_HANDLER      NULL
```

1123 /\* Special Context data for GenericSerialBus/GeneralPurposeIo (ACPI 5.0) \*/

```
1125 typedef struct acpi_connection_info
1126 {
1127     UINT8          *Connection;
1128     UINT16         Length;
1129     UINT8          AccessLength;
1131 } ACPI_CONNECTION_INFO;
```

```
1134 typedef
1135 ACPI_STATUS (*ACPI_ADR_SPACE_SETUP) (
1136     ACPI_HANDLE     RegionHandle,
1137     UINT32          Function,
1138     void            *HandlerContext,
1139     void            **RegionContext);
```

```
1141 #define ACPI_REGION_ACTIVATE      0
1142 #define ACPI_REGION_DEACTIVATE    1
```

```
1144 typedef
1145 ACPI_STATUS (*ACPI_WALK_CALLBACK) (
1146     ACPI_HANDLE     Object,
1147     UINT32          NestingLevel,
1148     void            *Context,
1149     void            **ReturnValue);
```

```
1151 typedef
1152 UINT32 (*ACPI_INTERFACE_HANDLER) (
1153     ACPI_STRING     InterfaceName,
1154     UINT32          Supported);
```

1157 /\* Interrupt handler return values \*/

```
1159 #define ACPI_INTERRUPT_NOT_HANDLED    0x00
1160 #define ACPI_INTERRUPT_HANDLED        0x01
```

1162 /\* GPE handler return values \*/

new/usr/src/common/acpica/include/actypes.h

18

```
1164 #define ACPI_REENABLE_GPE          0x80
```

1167 /\* Length of 32-bit EISAID values when converted back to a string \*/

```
1169 #define ACPI_EISAID_STRING_SIZE      8 /* Includes null terminator */
```

1171 /\* Length of UUID (string) values \*/

```
1173 #define ACPI_UUID_LENGTH              16
```

1176 /\* Structures used for device/processor HID, UID, CID, and SUB \*/  
1075 /\* Structures used for device/processor HID, UID, CID \*/

```
1178 typedef struct acpi_pnp_device_id
1077 typedef struct acpi_device_id
1179 {
1180     UINT32          Length; /* Length of string + nu
1181     char            *String;
```

```
1183 } ACPI_PNP_DEVICE_ID;
1082 } ACPI_DEVICE_ID;
```

```
1185 typedef struct acpi_pnp_device_id_list
1084 typedef struct acpi_device_id_list
1186 {
1187     UINT32          Count; /* Number of IDs in Ids
1188     UINT32          ListSize; /* Size of list, includi
1189     ACPI_PNP_DEVICE_ID Ids[1]; /* ID array */
1088     ACPI_DEVICE_ID Ids[1]; /* ID array */
```

```
1191 } ACPI_PNP_DEVICE_ID_LIST;
1090 } ACPI_DEVICE_ID_LIST;
```

1193 /\*  
1194 \* Structure returned from AcpiGetObjectInfo.  
1195 \* Optimized for both 32- and 64-bit builds  
1196 \*/

```
1197 typedef struct acpi_device_info
1198 {
1199     UINT32          InfoSize; /* Size of info, includi
1200     UINT32          Name; /* ACPI object Name */
1201     ACPI_OBJECT_TYPE Type; /* ACPI object Type */
1202     UINT8          ParamCount; /* If a method, required
1203     UINT8          Valid; /* Indicates which optio
1204     UINT8          Flags; /* Miscellaneous info */
1205     UINT8          HighestDstates[4]; /* _SxD values: 0xFF ind
1206     UINT8          LowestDstates[5]; /* _SxW values: 0xFF ind
1207     UINT32          CurrentStatus; /* _STA value */
1208     UINT64          Address; /* _ADR value */
1209     ACPI_PNP_DEVICE_ID HardwareId; /* _HID value */
1210     ACPI_PNP_DEVICE_ID UniqueId; /* _UID value */
1211     ACPI_PNP_DEVICE_ID SubsystemId; /* _SUB value */
1212     ACPI_PNP_DEVICE_ID_LIST CompatibleIdList; /* _CID list <must be la
1108     ACPI_DEVICE_ID HardwareId; /* _HID value */
1109     ACPI_DEVICE_ID UniqueId; /* _UID value */
1110     ACPI_DEVICE_ID_LIST CompatibleIdList; /* _CID list <must be la
```

```
1214 } ACPI_DEVICE_INFO;
```

1216 /\* Values for Flags field above (AcpiGetObjectInfo) \*/

```
1218 #define ACPI_PCI_ROOT_BRIDGE          0x01
```

```
1220 /* 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
1230 #define ACPI_VALID_CID          0x10
1231 #define ACPI_VALID_SXDS         0x20
1232 #define ACPI_VALID_SXWS         0x40
```

```
1231 /* Flags for _STA return value (CurrentStatus above) */
1232 /* 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;
1251 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;
1273     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__ */
```

new/usr/src/common/acpica/include/acutils.h

1

```
*****
27305 Fri Jan 17 07:31:05 2014
new/usr/src/common/acpica/include/acutils.h
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Name: acutils.h -- prototypes for the common (subsystem-wide) procedures
4 *
5 *****/
6
7 /*
8  * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #ifndef _ACUTILS_H
46 #define _ACUTILS_H
47
48 extern const UINT8 AcpiGbl_ResourceAmlSizes[];
49 extern const UINT8 AcpiGbl_ResourceAmlSerialBusSizes[];
50
51 /* Strings used by the disassembler and debugger resource dump routines */
52
53 #if defined(ACPI_DEBUG_OUTPUT) || defined(ACPI_DISASSEMBLER) || defined(ACPI_D
54 #if defined(ACPI_DISASSEMBLER) || defined(ACPI_DEBUGGER)
55
56 extern const char *AcpiGbl_BmDecode[];
57 extern const char *AcpiGbl_ConfigDecode[];
```

new/usr/src/common/acpica/include/acutils.h

2

```
58 extern const char *AcpiGbl_ConsumeDecode[];
59 extern const char *AcpiGbl_DecDecode[];
60 extern const char *AcpiGbl_HeDecode[];
61 extern const char *AcpiGbl_IoDecode[];
62 extern const char *AcpiGbl_LlDecode[];
63 extern const char *AcpiGbl_MaxDecode[];
64 extern const char *AcpiGbl_MemDecode[];
65 extern const char *AcpiGbl_MinDecode[];
66 extern const char *AcpiGbl_MtpDecode[];
67 extern const char *AcpiGbl_RngDecode[];
68 extern const char *AcpiGbl_RwDecode[];
69 extern const char *AcpiGbl_ShrDecode[];
70 extern const char *AcpiGbl_SizDecode[];
71 extern const char *AcpiGbl_TrnsDecode[];
72 extern const char *AcpiGbl_TtpDecode[];
73 extern const char *AcpiGbl_TypDecode[];
74 extern const char *AcpiGbl_PpcDecode[];
75 extern const char *AcpiGbl_IorDecode[];
76 extern const char *AcpiGbl_DtsDecode[];
77 extern const char *AcpiGbl_CtDecode[];
78 extern const char *AcpiGbl_SbtDecode[];
79 extern const char *AcpiGbl_AmDecode[];
80 extern const char *AcpiGbl_SmDecode[];
81 extern const char *AcpiGbl_WmDecode[];
82 extern const char *AcpiGbl_CphDecode[];
83 extern const char *AcpiGbl_CpoDecode[];
84 extern const char *AcpiGbl_DpDecode[];
85 extern const char *AcpiGbl_EdDecode[];
86 extern const char *AcpiGbl_BpbDecode[];
87 extern const char *AcpiGbl_SbDecode[];
88 extern const char *AcpiGbl_FcDecode[];
89 extern const char *AcpiGbl_PtDecode[];
90
91 #endif
92
93 /*
94  * For the iASL compiler case, the output is redirected to stderr so that
95  * any of the various ACPI errors and warnings do not appear in the output
96  * files, for either the compiler or disassembler portions of the tool.
97 */
98 #ifndef ACPI_ASL_COMPILER
99
100 #include <stdio.h>
101 extern FILE *AcpiGbl_OutputFile;
102
103 #define ACPI_MSG_REDIRECT_BEGIN \
104     FILE *OutputFile = AcpiGbl_OutputFile; \
105     AcpiOsRedirectOutput (stderr);
106
107 #define ACPI_MSG_REDIRECT_END \
108     AcpiOsRedirectOutput (OutputFile);
109
110 #else
111 /*
112  * non-iASL case - no redirection, nothing to do
113 */
114 #define ACPI_MSG_REDIRECT_BEGIN
115 #define ACPI_MSG_REDIRECT_END
116 #endif
117
118 /*
119  * Common error message prefixes
120 */
121 #define ACPI_MSG_ERROR "ACPI Error: "
122 #define ACPI_MSG_EXCEPTION "ACPI Exception: "
123 #define ACPI_MSG_WARNING "ACPI Warning: "
124 #define ACPI_MSG_INFO "ACPI: "
```

```

124 #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);
148 void
149     *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;
162 } ACPI_PKG_INFO;

163 } ACPI_PKG_INFO;

165 /* Object reference counts */

167 #define REF_INCREMENT      (UINT16) 0
168 #define REF_DECREMENT      (UINT16) 1
169 #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 AcpiUtGetTypeNames (
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 AcpiUtStrncpy (
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 AcpiUtMemcpy (
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 (

```

new/usr/src/common/acpica/include/acutils.h

```
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,
```

7

new/usr/src/common/acpica/include/acutils.h

```
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 (
473     AcpiUtDumpBuffer (
474         UINT8                    *Buffer,
475         UINT32                    Count,
476         UINT32                    Display,
477         UINT32                    ComponentId);
478         componentId);

478 void
479 AcpiUtDumpBuffer (
480     AcpiUtDumpBuffer2 (
481         UINT8                    *Buffer,
482         UINT32                    Count,
483         UINT32                    Display,
484         UINT32                    Offset);
485         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
```

8

```

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
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
774  */
775 ACPI_STATUS
776 AcpiUtDivide (
777     UINT64          InDividend,
778     UINT64          InDivisor,
779     UINT64          *OutQuotient,
780     UINT64          *OutRemainder);

782 ACPI_STATUS
783 AcpiUtShortDivide (
784     UINT64          InDividend,
785     UINT32          Divisor,
786     UINT64          *OutQuotient,
787     UINT32          *OutRemainder);

790 /*
791  * utmisc
792  */
793 const ACPI_EXCEPTION_INFO *
794 const char *
795 AcpiUtValidateException (
796     ACPI_STATUS      Status);

797 BOOLEAN
798 AcpiUtIsPciRootBridge (
799     char             *Id);

801 BOOLEAN
802 AcpiUtIsAmlTable (
803     ACPI_TABLE_HEADER *Table);

805 ACPI_STATUS
806 AcpiUtAllocateOwnerId (
807     ACPI_OWNER_ID     *OwnerId);

808 void
809 AcpiUtReleaseOwnerId (
810     ACPI_OWNER_ID     *OwnerId);

811 ACPI_STATUS
812 AcpiUtWalkPackageTree (
813     ACPI_OPERAND_OBJECT *SourceObject,
814     void                *TargetObject,
815     ACPI_PKG_CALLBACK   WalkCallback,
816     void                *Context);

817 void
818 AcpiUtStrupr (
819     char             *SrcString);

821 void
822 AcpiUtStrlwr (
823     char             *SrcString);

825 void
826 AcpiUtPrintString (
827     char             *String,
828     UINT8            MaxLength);

830 BOOLEAN
831 AcpiUtValidAcpiName (
832     UINT32           Name);

```

```

834 void
835 AcpiUtRepairName (
836     char             *Name);

838 BOOLEAN
839 AcpiUtValidAcpiChar (
840     char             Character,
841     UINT32           Position);

843 ACPI_STATUS
844 AcpiUtStrtol64 (
845     char             *String,
846     UINT32           Base,
847     UINT64           *RetInteger);

849 /* Values for Base above (16=Hex, 10=Decimal) */

851 #define ACPI_ANY_BASE      0

853 UINT32
854 AcpiUtDwordByteSwap (
855     UINT32           Value);

857 void
858 AcpiUtSetIntegerWidth (
859     UINT8            Revision);

861 #ifdef ACPI_DEBUG_OUTPUT
862 void
863 AcpiUtDisplayInitPathname (
864     UINT8            Type,
865     ACPI_NAMESPACE_NODE *ObjHandle,
866     char             *Path);
867 #endif

869 /*
870  * utownerid - Support for Table/Method Owner IDs
871  */
872 ACPI_STATUS
873 AcpiUtAllocateOwnerId (
874     ACPI_OWNER_ID     *OwnerId);

875 void
876 AcpiUtReleaseOwnerId (
877     ACPI_OWNER_ID     *OwnerId);

879 /*
880  * utresrc
881  */
882 ACPI_STATUS
883 AcpiUtWalkAmlResources (
884     ACPI_WALK_STATE   *WalkState,
885     UINT8             Aml,
886     ACPI_SIZE          AmlLength,
887     ACPI_WALK_AML_CALLBACK UserFunction,
888     void                **Context);

889 void
890 AcpiUtValidateResource (
891     ACPI_WALK_STATE   *WalkState,
892     void                Aml,
893     UINT8             *ReturnIndex);

```

```

863 UINT32
864 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);

947 void *
948 AcpiUtAllocate (
949     ACPI_SIZE           Size,
950     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 */

```

new/usr/src/common/acpica/include/amlcode.h

1

```
*****
19946 Fri Jan 17 07:31:06 2014
new/usr/src/common/acpica/include/amlcode.h
update to acpica-unix2-20140114
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 - 2014, Intel Corp.
11 * Copyright (C) 2000 - 2011, Intel Corp.
12 * All rights reserved.
13 *
14 * Redistribution and use in source and binary forms, with or without
15 * modification, are permitted provided that the following conditions
16 * are met:
17 * 1. Redistributions of source code must retain the above copyright
18 *    notice, this list of conditions, and the following disclaimer,
19 *    without modification.
20 * 2. Redistributions in binary form must reproduce at minimum a disclaimer
21 *    substantially similar to the "NO WARRANTY" disclaimer below
22 *    ("Disclaimer") and any redistribution must be conditioned upon
23 *    including a substantially similar Disclaimer requirement for further
24 *    binary redistribution.
25 * 3. Neither the names of the above-listed copyright holders nor the names
26 *    of any contributors may be used to endorse or promote products derived
27 *    from this software without specific prior written permission.
28 *
29 * Alternatively, this software may be distributed under the terms of the
30 * GNU General Public License ("GPL") version 2 as published by the Free
31 * Software Foundation.
32 *
33 * NO WARRANTY
34 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
35 * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
36 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
37 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
38 * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
39 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
40 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
41 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
42 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
43 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
44 * POSSIBILITY OF SUCH DAMAGES.
45 */
46 #ifndef __AMLCODE_H__
47 #define __AMLCODE_H__
48
49 /* primary opcodes */
50
51 #define AML_NULL_CHAR                (UINT16) 0x00
52
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
```

new/usr/src/common/acpica/include/amlcode.h

2

```
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
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
131 #define AML_TO_DECSTRING_OP       (UINT16) 0x97
132 #define AML_TO_HEXSTRING_OP      (UINT16) 0x98
133 #define AML_TO_INTEGER_OP        (UINT16) 0x99
134 #define AML_TO_STRING_OP         (UINT16) 0x9c
135 #define AML_COPY_OP              (UINT16) 0x9d
136 #define AML_MID_OP               (UINT16) 0x9e
137 #define AML_CONTINUE_OP          (UINT16) 0x9f
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_PKGLNGTH           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., ARG_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.

```



```

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_____

```

new/usr/src/common/acpica/include/amlresrc.h

1

```
*****
19141 Fri Jan 17 07:31:06 2014
new/usr/src/common/acpica/include/amlresrc.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: amlresrc.h - AML resource descriptors
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 /* acpisrc:StructDefs -- for acpisrc conversion */

46 #ifndef __AMLRESRC_H
47 #define __AMLRESRC_H

50 /*
51 * Resource descriptor tags, as defined in the ACPI specification.
52 * Used to symbolically reference fields within a descriptor.
53 */
54 #define ACPI_RESTAG_ADDRESS          "_ADR"
55 #define ACPI_RESTAG_ALIGNMENT        "_ALN"
56 #define ACPI_RESTAG_ADDRESSSPACE     "_ASI"
57 #define ACPI_RESTAG_ACCESSSIZE       "_ASZ"
58 #define ACPI_RESTAG_TYPESPECIFICATTRIBUTES "_ATT"
59 #define ACPI_RESTAG_BASEADDRESS      "_BAS"
```

new/usr/src/common/acpica/include/amlresrc.h

2

```
60 #define ACPI_RESTAG_BUSMASTER      "_BM" /* Master(1), Slave(0) */
61 #define ACPI_RESTAG_DEBOUNCETIME    "_DBT"
62 #define ACPI_RESTAG_DECODE          "_DEC"
63 #define ACPI_RESTAG_DEVICEPOLARITY  "_DPL"
64 #define ACPI_RESTAG_DMA             "_DMA"
65 #define ACPI_RESTAG_DMATYP          "_TYP" /* Compatible(0), A(1),
66 #define ACPI_RESTAG_DRIVESTRENGTH   "_DRS"
67 #define ACPI_RESTAG_ENDIANNES       "_END"
68 #define ACPI_RESTAG_FLOWCONTROL     "_FLC"
69 #define ACPI_RESTAG_GRANULARITY     "_GRA"
70 #define ACPI_RESTAG_INTERRUPT       "_INT"
71 #define ACPI_RESTAG_INTERRUPTLEVEL  "_LL" /* ActiveLo(1), ActiveHi
72 #define ACPI_RESTAG_INTERRUPTSHARE  "_SHR" /* Shareable(1), NoShare
73 #define ACPI_RESTAG_INTERRUPTTYPE   "_HE" /* Edge(1), Level(0) */
74 #define ACPI_RESTAG_IORESTRICTION   "_IOR"
75 #define ACPI_RESTAG_LENGTH          "_LEN"
76 #define ACPI_RESTAG_LINE            "_LIN"
77 #define ACPI_RESTAG_MEMATTRIBUTES   "_MTP" /* Memory(0), Reserved(1
78 #define ACPI_RESTAG_MEMTYPE         "_MEM" /* NonCache(0), Cacheabl
79 #define ACPI_RESTAG_MAXADDR         "_MAX"
80 #define ACPI_RESTAG_MINADDR         "_MIN"
81 #define ACPI_RESTAG_MAXTYPE         "_MAF"
82 #define ACPI_RESTAG_MINTYPE         "_MIF"
83 #define ACPI_RESTAG_MODE            "_MOD"
84 #define ACPI_RESTAG_PARITY          "_PAR"
85 #define ACPI_RESTAG_PHASE           "_PHA"
86 #define ACPI_RESTAG_PIN             "_PIN"
87 #define ACPI_RESTAG_PINCONFIG       "_PPI"
88 #define ACPI_RESTAG_POLARITY        "_POL"
89 #define ACPI_RESTAG_REGISTERBITOFFSET "_RBO"
90 #define ACPI_RESTAG_REGISTERBITWIDTH "_RBW"
91 #define ACPI_RESTAG_RANGETYPE       "_RNG"
92 #define ACPI_RESTAG_READWRITETYPE   "_RW" /* ReadOnly(0), Writeabl
93 #define ACPI_RESTAG_LENGTH_RX       "_RXL"
94 #define ACPI_RESTAG_LENGTH_TX       "_TXL"
95 #define ACPI_RESTAG_SLAVEMODE       "_SLV"
96 #define ACPI_RESTAG_SPEED           "_SPE"
97 #define ACPI_RESTAG_STOPBITS        "_STB"
98 #define ACPI_RESTAG_TRANSLATION     "_TRA"
99 #define ACPI_RESTAG_TRANSTYPE       "_TRS" /* Sparse(1), Dense(0) */
100 #define ACPI_RESTAG_TYPE            "_TTP" /* Translation(1), Stati
101 #define ACPI_RESTAG_XFERTYPE        "_SIZ" /* 8(0), 8And16(1), 16(2
102 #define ACPI_RESTAG_VENDORDATA      "_VEN"

105 /* Default sizes for "small" resource descriptors */

107 #define ASL_RDESC_IRQ_SIZE           0x02
108 #define ASL_RDESC_DMA_SIZE           0x02
109 #define ASL_RDESC_ST_DEPEND_SIZE     0x00
110 #define ASL_RDESC_END_DEPEND_SIZE    0x00
111 #define ASL_RDESC_IO_SIZE            0x07
112 #define ASL_RDESC_FIXED_IO_SIZE      0x03
113 #define ASL_RDESC_FIXED_DMA_SIZE     0x05
114 #define ASL_RDESC_END_TAG_SIZE       0x01

117 typedef struct asl_resource_node
118 {
119     UINT32                BufferLength;
120     void                  *Buffer;
121     struct asl_resource_node *Next;
123 } ASL_RESOURCE_NODE;
```



```

126 /* 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      ResourceIndex;
415     UINT16     ResourceOffset;
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      ResourceIndex; \
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
```

new/usr/src/common/acpica/include/platform/accygwin.h

1

```
*****
3547 Fri Jan 17 07:31:06 2014
new/usr/src/common/acpica/include/platform/accygwin.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: accygwin.h - OS specific defines, etc. for cygwin environment
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACCYGWIN_H__
45 #define __ACCYGWIN_H__

47 /*
48 * ACPIA configuration
49 */
50 #define ACPI_USE_SYSTEM_CLIBRARY
51 #define ACPI_USE_DO_WHILE_0
52 #define ACPI_FLUSH_CPU_CACHE()

54 /*
55 * This is needed since sem_timedwait does not appear to work properly
56 * on cygwin (always hangs forever).
57 */
58 #define ACPI_USE_ALTERNATE_TIMEOUT
```

new/usr/src/common/acpica/include/platform/accygwin.h

2

```
61 #include <stdarg.h>
62 #include <string.h>
63 #include <stdlib.h>
64 #include <ctype.h>
65 #include <unistd.h>

67 #if defined(__ia64__) || defined(__x86_64__)
68 #define ACPI_MACHINE_WIDTH 64
69 #define COMPILER_DEPENDENT_INT64 long
70 #define COMPILER_DEPENDENT_UINT64 unsigned long
71 #else
72 #define ACPI_MACHINE_WIDTH 32
73 #define COMPILER_DEPENDENT_INT64 long long
74 #define COMPILER_DEPENDENT_UINT64 unsigned long long
75 #define ACPI_USE_NATIVE_DIVIDE
76 #endif

78 #ifndef __cdecl
79 #define __cdecl
80 #endif

82 #define ACPI_ACQUIRE_GLOBAL_LOCK(GLptr, Acq) if (GLptr) Acq=1; else Acq=0;
83 #define ACPI_RELEASE_GLOBAL_LOCK(GLptr, Pending) Pending = 1

85 /* On Cygwin, pthread_t is a pointer */

87 #define ACPI_CAST_PTHREAD_T(pthread) ((ACPI_THREAD_ID) ACPI_TO_INTEGER (pthread))

89 /* Cygwin uses GCC */

91 #include "acgcc.h"

94 /*
95 * The vsnprintf function is defined by c99, but cygwin/gcc does not
96 * enable this prototype when the -ansi flag is set. Also related to
97 * __STRICT_ANSI__. So, we just declare the prototype here.
98 */
99 int
100 vsnprintf (char *s, size_t n, const char *format, va_list ap);

102 #endif /* __ACCYGWIN_H__ */
```

new/usr/src/common/acpica/include/platform/acefi.h

1

```
*****
2836 Fri Jan 17 07:31:07 2014
new/usr/src/common/acpica/include/platform/acefi.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acefi.h - OS specific defines, etc.
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACEFI_H__
45 #define __ACEFI_H__

47 #include <efi.h>
48 #include <efistdarg.h>
49 #include <efilib.h>

52 /* _int64 works for both IA32 and IA64 */

54 #define COMPILER_DEPENDENT_INT64 __int64
55 #define COMPILER_DEPENDENT_UINT64 unsigned __int64

57 /*
58 * Calling conventions:
```

new/usr/src/common/acpica/include/platform/acefi.h

2

```
59 *
60 * ACPI_SYSTEM_XFACE - Interfaces to host OS (handlers, threads)
61 * ACPI_EXTERNAL_XFACE - External ACPI interfaces
62 * ACPI_INTERNAL_XFACE - Internal ACPI interfaces
63 * ACPI_INTERNAL_VAR_XFACE - Internal variable-parameter list interfaces
64 */
65 #define ACPI_SYSTEM_XFACE
66 #define ACPI_EXTERNAL_XFACE
67 #define ACPI_INTERNAL_XFACE
68 #define ACPI_INTERNAL_VAR_XFACE

70 /* warn C4142: redefinition of type */

72 #pragma warning(disable:4142)

75 #endif /* __ACEFI_H__ */
```

new/usr/src/common/acpica/include/platform/acenv.h

1

```
*****
12451 Fri Jan 17 07:31:07 2014
new/usr/src/common/acpica/include/platform/acenv.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Name: acenv.h - Host and compiler configuration
4 *
5 *****/
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACENV_H__
45 #define __ACENV_H__
46
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 */
52
53 /* Types for ACPI_MUTEX_TYPE */
54
55 #define ACPI_BINARY_SEMAPHORE 0
56 #define ACPI_OSL_MUTEX 1
```

new/usr/src/common/acpica/include/platform/acenv.h

2

```
58 /* Types for DEBUGGER_THREADING */
59
60 #define DEBUGGER_SINGLE_THREADED 0
61 #define DEBUGGER_MULTI_THREADED 1
62
63
64 /*****
65 *
66 * Configuration for ACPI tools and utilities
67 *
68 *****/
69
70 /* iASL configuration */
71
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
81
82 /* AcpiExec configuration. Multithreaded with full AML debugger */
83 /* AcpiExec and AcpiBin configuration */
84
85 #ifdef ACPI_EXEC_APP
86 #define ACPI_APPLICATION
87 #define ACPI_FULL_DEBUG
88 #define ACPI_MUTEX_DEBUG
89 #define ACPI_DBG_TRACK_ALLOCATIONS
90 #endif
91
92 /* AcpiNames configuration. Single threaded with debugger output enabled. */
93
94 #ifdef ACPI_NAMES_APP
95 #define ACPI_DEBUGGER
96 #define ACPI_BIN_APP
97 #define ACPI_APPLICATION
98 #define ACPI_SINGLE_THREADED
99 #endif
100
101 /*
102 * AcpiBin/AcpiDump/AcpiSrc/AcpiXtract/Example configuration. All single
103 * threaded, with no debug output.
104 */
105 #if (defined ACPI_BIN_APP) || \
106     (defined ACPI_DUMP_APP) || \
107     (defined ACPI_SRC_APP) || \
108     (defined ACPI_XTRACT_APP) || \
109     (defined ACPI_EXAMPLE_APP)
110 #define ACPI_APPLICATION
111 #define ACPI_SINGLE_THREADED
112 #endif
113
114 #ifdef ACPI_HELP_APP
115 #define ACPI_APPLICATION
116 #define ACPI_SINGLE_THREADED
117 #define ACPI_NO_ERROR_MESSAGES
118 #endif
119
120 /* Linkable ACPICA library */
121
122 #ifdef ACPI_LIBRARY
123 #define ACPI_USE_LOCAL_CACHE
```

new/usr/src/common/acpica/include/platform/acenv.h

3

```
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"
```

new/usr/src/common/acpica/include/platform/acenv.h

4

```
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)
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 *)s2, (n))
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__ */
```

new/usr/src/common/acpica/include/platform/acfreebsd.h

1

```
*****
3487 Fri Jan 17 07:31:07 2014
new/usr/src/common/acpica/include/platform/acfreebsd.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acfreebsd.h - OS specific defines, etc.
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACFREEBSD_H__
45 #define __ACFREEBSD_H__

48 /* FreeBSD uses GCC */

50 #include "acgcc.h"
51 #include <sys/types.h>
52 #include <machine/acpica_machdep.h>

53 #ifndef __LP64__
54 #define ACPI_MACHINE_WIDTH 64
55 #else
56 #define ACPI_MACHINE_WIDTH 32
57 #endif
```

new/usr/src/common/acpica/include/platform/acfreebsd.h

2

```
59 #define COMPILER_DEPENDENT_INT64 int64_t
60 #define COMPILER_DEPENDENT_UINT64 uint64_t

62 #define ACPI_UINTPTR_T uintptr_t

64 #define ACPI_USE_DO_WHILE_0
65 #define ACPI_USE_LOCAL_CACHE
66 #define ACPI_USE_NATIVE_DIVIDE
67 #define ACPI_USE_SYSTEM_CLIBRARY

69 #ifdef _KERNEL

71 #include <sys/ctype.h>
72 #include <sys/param.h>
73 #include <sys/systm.h>
74 #include <sys/libkern.h>
75 #include <machine/acpica_machdep.h>
76 #include <machine/stdarg.h>

78 #include "opt_acpi.h"

80 #define ACPI_MUTEX_TYPE ACPI_OSL_MUTEX

82 #ifdef ACPI_DEBUG
83 #define ACPI_DEBUG_OUTPUT /* for backward compatibility */
84 #define ACPI_DISASSEMBLER
85 #endif

87 #ifdef ACPI_DEBUG_OUTPUT
88 #include "opt_ddb.h"
89 #ifdef DDB
90 #define ACPI_DEBUGGER
91 #endif /* DDB */
92 #endif /* ACPI_DEBUG_OUTPUT */

94 #ifdef DEBUGGER_THREADING
95 #undef DEBUGGER_THREADING
96 #endif /* DEBUGGER_THREADING */

98 #define DEBUGGER_THREADING 0 /* integrated with DDB */

100 #else /* _KERNEL */

102 #if __STDC_HOSTED__
103 #include <ctype.h>
104 #endif

106 #define ACPI_CAST_PTHREAD_T(pthread) ((ACPI_THREAD_ID) ACPI_TO_INTEGER (pthread))

108 #define ACPI_USE_STANDARD_HEADERS

110 #define ACPI_FLUSH_CPU_CACHE()
111 #define __cdecl

113 #endif /* _KERNEL */

115 #endif /* __ACFREEBSD_H__ */
```

new/usr/src/common/acpica/include/platform/acgcc.h

1

```
*****
2843 Fri Jan 17 07:31:08 2014
new/usr/src/common/acpica/include/platform/acgcc.h
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 / *****
2 *
3 * Name: acgcc.h - GCC specific defines, etc.
4 *
5 *****/
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __ACGCC_H__
45 #define __ACGCC_H__
47 #define ACPI_INLINE __inline__
49 /* Function name is used for debug output. Non-ANSI, compiler-dependent */
51 #define ACPI_GET_FUNCTION_NAME __func__
51 #define ACPI_GET_FUNCTION_NAME __FUNCTION__
53 /*
54 * This macro is used to tag functions as "printf-like" because
55 * some compilers (like GCC) can catch printf format string problems.
56 */
```

new/usr/src/common/acpica/include/platform/acgcc.h

2

```
57 #define ACPI_PRINTF_LIKE(c) __attribute__((__format__ (__printf__, c, c+1)))
59 /*
60 * Some compilers complain about unused variables. Sometimes we don't want to
61 * use all the variables (for example, _AcpiModuleName). This allows us
62 * to tell the compiler warning in a per-variable manner that a variable
62 * to to tell the compiler warning in a per-variable manner that a variable
63 * is unused.
64 */
65 #define ACPI_UNUSED_VAR __attribute__((unused))
67 #endif /* __ACGCC_H__ */
```

new/usr/src/common/acpica/include/platform/achaiku.h

1

```
*****
3764 Fri Jan 17 07:31:08 2014
new/usr/src/common/acpica/include/platform/achaiku.h
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Name: achaiku.h - OS specific defines, etc. for Haiku (www.haiku-os.org)
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #ifndef __ACHAIKU_H
45 #define __ACHAIKU_H
46
47 #include "acgcc.h"
48 #include <KernelExport.h>
49
50 struct mutex;
51
52
53 /* Host-dependent types and defines for user- and kernel-space ACPICA */
54
55 #define ACPI_USE_SYSTEM_CLIBRARY
56 #define ACPI_USE_STANDARD_HEADERS
57
58 #define ACPI_MUTEX_TYPE          ACPI_OSL_MUTEX
59 #define ACPI_MUTEX              struct mutex *
```

new/usr/src/common/acpica/include/platform/achaiku.h

2

```
61 #define ACPI_USE_NATIVE_DIVIDE
62
63 // #define ACPI_THREAD_ID          thread_id
64 #define ACPI_SEMAPHORE            sem_id
65 #define ACPI_SPINLOCK             spinlock *
66 #define ACPI_CPU_FLAGS            cpu_status
67
68 #define COMPILER_DEPENDENT_INT64  int64
69 #define COMPILER_DEPENDENT_UINT64 uint64
70
71
72 #ifdef B_HAIKU_64_BIT
73 #define ACPI_MACHINE_WIDTH        64
74 #else
75 #define ACPI_MACHINE_WIDTH        32
76 #endif
77
78
79 #ifdef _KERNEL_MODE
80 /* Host-dependent types and defines for in-kernel ACPICA */
81
82 /* ACPICA cache implementation is adequate. */
83 #define ACPI_USE_LOCAL_CACHE
84
85 #define ACPI_FLUSH_CPU_CACHE() __asm __volatile("wbinvd");
86
87 /* Based on FreeBSD's due to lack of documentation */
88 extern int AcpiOsAcquireGlobalLock(uint32 *lock);
89 extern int AcpiOsReleaseGlobalLock(uint32 *lock);
90
91 #define ACPI_ACQUIRE_GLOBAL_LOCK(GLptr, Acq) do { \
92     (Acq) = AcpiOsAcquireGlobalLock(&((GLptr)->GlobalLock)); \
93 } while (0)
94
95 #define ACPI_RELEASE_GLOBAL_LOCK(GLptr, Acq) do { \
96     (Acq) = AcpiOsReleaseGlobalLock(&((GLptr)->GlobalLock)); \
97 } while (0)
98
99 #else /* _KERNEL_MODE */
100 /* Host-dependent types and defines for user-space ACPICA */
101
102 #error "We only support kernel mode ACPI atm."
103
104 #endif /* _KERNEL_MODE */
105 #endif /* __ACHAIKU_H */
```

new/usr/src/common/acpica/include/platform/acintel.h

1

```
*****
3196 Fri Jan 17 07:31:08 2014
new/usr/src/common/acpica/include/platform/acintel.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acintel.h - VC specific defines, etc.
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACINTEL_H__
45 #define __ACINTEL_H__

47 /* Configuration specific to Intel 64-bit C compiler */

49 #define COMPILER_DEPENDENT_INT64    __int64
50 #define COMPILER_DEPENDENT_UINT64   unsigned __int64
51 #define ACPI_INLINE                 __inline

53 /*
54 * Calling conventions:
55 *
56 * ACPI_SYSTEM_XFACE        - Interfaces to host OS (handlers, threads)
57 * ACPI_EXTERNAL_XFACE      - External ACPI interfaces
58 * ACPI_INTERNAL_XFACE      - Internal ACPI interfaces
```

new/usr/src/common/acpica/include/platform/acintel.h

2

```
59 * ACPI_INTERNAL_VAR_XFACE - Internal variable-parameter list interfaces
60 */
61 #define ACPI_SYSTEM_XFACE
62 #define ACPI_EXTERNAL_XFACE
63 #define ACPI_INTERNAL_XFACE
64 #define ACPI_INTERNAL_VAR_XFACE

66 /* remark 981 - operands evaluated in no particular order */
67 #pragma warning(disable:981)

69 /* warn C4100: unreferenced formal parameter */
70 #pragma warning(disable:4100)

72 /* warn C4127: conditional expression is constant */
73 #pragma warning(disable:4127)

75 /* warn C4706: assignment within conditional expression */
76 #pragma warning(disable:4706)

78 /* warn C4214: bit field types other than int */
79 #pragma warning(disable:4214)

82 #endif /* __ACINTEL_H__ */
```

new/usr/src/common/acpica/include/platform/aclinux.h

1

```
*****
7844 Fri Jan 17 07:31:09 2014
new/usr/src/common/acpica/include/platform/aclinux.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: aclinux.h - OS specific defines, etc. for Linux
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #ifndef __ACLINUX_H__
45 #define __ACLINUX_H__
46
47 /* Common (in-kernel/user-space) ACPICA configuration */
48
49 #define ACPI_USE_SYSTEM_CLIBRARY
50 #define ACPI_USE_DO_WHILE_0
51 #define ACPI_MUTEX_TYPE          ACPI_BINARY_SEMAPHORE
52
53
54 #ifdef __KERNEL__
55
56 #include <linux/string.h>
```

new/usr/src/common/acpica/include/platform/aclinux.h

2

```
57 #include <linux/kernel.h>
58 #include <linux/ctype.h>
59 #include <linux/sched.h>
60 #include <linux/atomic.h>
61 #include <linux/math64.h>
62 #include <linux/slab.h>
63 #include <linux/spinlock_types.h>
64 #ifdef EXPORT_ACPI_INTERFACES
65 #include <linux/export.h>
66 #endif
67 #include <asm/acpi.h>
68
69 /* Host-dependent types and defines for in-kernel ACPICA */
70
71 #define ACPI_MACHINE_WIDTH      BITS_PER_LONG
72 #define ACPI_EXPORT_SYMBOL(symbol) EXPORT_SYMBOL(symbol);
73 #define strtoul                simple_strtoul
74
75 #define ACPI_CACHE_T            struct kmem_cache
76 #define ACPI_SPINLOCK           spinlock_t *
77 #define ACPI_CPU_FLAGS          unsigned long
78
79 #else /* !__KERNEL__ */
80
81 #include <stdarg.h>
82 #include <string.h>
83 #include <stdlib.h>
84 #include <ctype.h>
85 #include <unistd.h>
86
87 /* Host-dependent types and defines for user-space ACPICA */
88
89 #define ACPI_FLUSH_CPU_CACHE()
90 #define ACPI_CAST_PTHREAD_T(Pthread) ((ACPI_THREAD_ID) (Pthread))
91
92 #if defined(__ia64__) || defined(__x86_64__) || defined(__aarch64__)
93 #define ACPI_MACHINE_WIDTH      64
94 #define COMPILER_DEPENDENT_INT64 long
95 #define COMPILER_DEPENDENT_UINT64 unsigned long
96 #else
97 #define ACPI_MACHINE_WIDTH      32
98 #define COMPILER_DEPENDENT_INT64 long long
99 #define COMPILER_DEPENDENT_UINT64 unsigned long long
100 #define ACPI_USE_NATIVE_DIVIDE
101 #endif
102
103 #ifndef __cdecl
104 #define __cdecl
105 #endif
106
107 #endif /* __KERNEL__ */
108
109 /* Linux uses GCC */
110
111 #include "acgcc.h"
112
113
114 #ifdef __KERNEL__
115
116 /*
117 * FIXME: Inclusion of actypes.h
118 * Linux kernel need this before defining inline OSL interfaces as
119 * actypes.h need to be included to find ACPICA type definitions.
120 * Since from ACPICA's perspective, the actypes.h should be included after
121 * acenv.h (aclinux.h), this leads to a inclusion mis-ordering issue.
122 */
```

```

123 #include <acpi/actypes.h>

125 /*
126  * Overrides for in-kernel ACPIA
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 kcalloc() and resume does not.
145  */
146 static inline void *
147 AcpiOsAllocate (
148     ACPI_SIZE          Size)
149 {
150     return kcalloc (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 kcalloc (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 ACPIA 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 ACPIA 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  */

```



new/usr/src/common/acpica/include/platform/aclinux.h

5

```
255 void
256 EarlyAcpiOsUnmapMemory (
257     void __iomem      *Virt,
258     ACPI_SIZE          Size);

260 #endif /* __KERNEL__ */

262 #endif /* __ACLINUX_H__ */
```

new/usr/src/common/acpica/include/platform/acmacosx.h

1

\*\*\*\*\*

2399 Fri Jan 17 07:31:09 2014

new/usr/src/common/acpica/include/platform/acmacosx.h

update to acpica-unix2-20140114

acpica-unix2-20130823

\*\*\*\*\*

```
1 /*****
2 *
3 * Name: acmacosx.h - OS specific defines, etc. for Mac OS X
4 *
5 *****/
```

```
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __ACMACOSX_H__
45 #define __ACMACOSX_H__
```

```
47 #include "aclinux.h"
```

```
49 #ifdef __APPLE__
50 #define sem_destroy sem_close
51 #define ACPI_USE_ALTERNATE_TIMEOUT
52 #endif /* __APPLE__ */
```

```
54 #ifdef __clang__
55 #pragma clang diagnostic ignored "-Wformat-nonliteral"
56 #endif
```

```
58 #endif /* __ACMACOSX_H__ */
```

new/usr/src/common/acpica/include/platform/acmsvc.h

1

```
*****
5413 Fri Jan 17 07:31:09 2014
new/usr/src/common/acpica/include/platform/acmsvc.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acmsvc.h - VC specific defines, etc.
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACMSVC_H__
45 #define __ACMSVC_H__
46
47 /*
48 * Map low I/O functions for MS. This allows us to disable MS language
49 * extensions for maximum portability.
50 */
51 #define open _open
52 #define read _read
53 #define write _write
54 #define close _close
55 #define stat _stat
56 #define fstat _fstat
57 #define mkdir _mkdir
```

new/usr/src/common/acpica/include/platform/acmsvc.h

2

```
59 #define strtowl _strtowl
60 #define O_RDONLY _O_RDONLY
61 #define O_BINARY _O_BINARY
62 #define O_CREAT _O_CREAT
63 #define O_WRONLY _O_WRONLY
64 #define O_TRUNC _O_TRUNC
65 #define S_IREAD _S_IREAD
66 #define S_IWRITE _S_IWRITE
67 #define S_IFDIR _S_IFDIR
68
69 /* Eliminate warnings for "old" (non-secure) versions of clib functions */
70
71 #ifndef _CRT_SECURE_NO_WARNINGS
72 #define _CRT_SECURE_NO_WARNINGS
73 #endif
74
75 /* Eliminate warnings for POSIX clib function names (open, write, etc.) */
76
77 #ifndef _CRT_NONSTDC_NO_DEPRECATED
78 #define _CRT_NONSTDC_NO_DEPRECATED
79 #endif
80
81 #define COMPILER_DEPENDENT_INT64 __int64
82 #define COMPILER_DEPENDENT_UINT64 unsigned __int64
83 #define ACPI_INLINE __inline
84
85 /*
86 * Calling conventions:
87 *
88 * ACPI_SYSTEM_XFACE - Interfaces to host OS (handlers, threads)
89 * ACPI_EXTERNAL_XFACE - External ACPI interfaces
90 * ACPI_INTERNAL_XFACE - Internal ACPI interfaces
91 * ACPI_INTERNAL_VAR_XFACE - Internal variable-parameter list interfaces
92 */
93 #define ACPI_SYSTEM_XFACE __cdecl
94 #define ACPI_EXTERNAL_XFACE
95 #define ACPI_INTERNAL_XFACE
96 #define ACPI_INTERNAL_VAR_XFACE __cdecl
97
98 #ifndef _LINT
99 /*
100 * Math helper functions
101 */
102 #define ACPI_DIV_64_BY_32(n_hi, n_lo, d32, q32, r32) \
103 { \
104     __asm mov edx, n_hi \
105     __asm mov eax, n_lo \
106     __asm div d32 \
107     __asm mov q32, eax \
108     __asm mov r32, edx \
109 }
110 #define unchanged_portion_omitted
111 #endif
112
113 /* warn C4100: unreferenced formal parameter */
114 #pragma warning(disable:4100)
115
116 /* warn C4127: conditional expression is constant */
117 #pragma warning(disable:4127)
118
119 /* warn C4706: assignment within conditional expression */
120 #pragma warning(disable:4706)
121
122 /* warn C4131: uses old-style declarator (iASL compiler only) */
123 #pragma warning(disable:4131)
```

new/usr/src/common/acpica/include/platform/acmsvc.h

3

```
145 #if _MSC_VER > 1200 /* Versions above VC++ 6 */
146 #pragma warning( disable : 4295 ) /* needed for acpredef.h array */
147 #endif

150 /* Debug support. Must be last in this file, do not move. */

152 #ifdef _DEBUG
153 #include <crtdbg.h>

155 /*
156  * Debugging memory corruption issues with windows:
157  * Add #include <crtdbg.h> to accomon.h if necessary.
158  * Add _ASSERT(_CrtCheckMemory()); where needed to test memory integrity.
159  * This can quickly localize the memory corruption.
160  */
161 #define ACPI_DEBUG_INITIALIZE() \
162     _CrtSetDbgFlag (_CRTDBG_CHECK_ALWAYS_DF | \
163         _CRTDBG_ALLOC_MEM_DF | _CRTDBG_CHECK_CRT_DF | \
164         _CrtSetDbgFlag(_CRTDBG_REPORT_FLAG))
165 #endif

167 #endif /* __ACMSVC_H__ */
```

new/usr/src/common/acpica/include/platform/acnetbsd.h

1

```
*****
3518 Fri Jan 17 07:31:10 2014
new/usr/src/common/acpica/include/platform/acnetbsd.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acnetbsd.h - OS specific defines, etc.
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACNETBSD_H__
45 #define __ACNETBSD_H__

47 /* NetBSD uses GCC */

49 #include "acgcc.h"

51 #ifdef __LP64__
52 #define ACPI_MACHINE_WIDTH 64
53 #else
54 #define ACPI_MACHINE_WIDTH 32
55 #endif

57 #define COMPILER_DEPENDENT_INT64 int64_t
58 #define COMPILER_DEPENDENT_UINT64 uint64_t
```

new/usr/src/common/acpica/include/platform/acnetbsd.h

2

```
60 #ifdef __KERNEL__
61 #include "opt_acpi.h" /* collect build-time options here */

63 #include <sys/param.h>
64 #include <sys/system.h>
65 #include <machine/stdarg.h>
66 #include <machine/acpi_func.h>

68 #define asm __asm

70 #define ACPI_USE_NATIVE_DIVIDE

72 #define ACPI_SYSTEM_XFACE
73 #define ACPI_EXTERNAL_XFACE
74 #define ACPI_INTERNAL_XFACE
75 #define ACPI_INTERNAL_VAR_XFACE

77 #ifdef ACPI_DEBUG
78 #define ACPI_DEBUG_OUTPUT
79 #define ACPI_DBG_TRACK_ALLOCATIONS
80 #ifdef DEBUGGER_THREADING
81 #undef DEBUGGER_THREADING
82 #endif /* DEBUGGER_THREADING */
83 #define DEBUGGER_THREADING 0 /* integrated with DDB */
84 #include "opt_ddb.h"
85 #ifdef DDB
86 #define ACPI_DISASSEMBLER
87 #define ACPI_DEBUGGER
88 #endif /* DDB */
89 #endif /* ACPI_DEBUG */

91 static __inline int
92 isprint(int ch)
93 {
94     return(isspace(ch) || isascii(ch));
95 }

_____unchanged_portion_omitted_____
```

new/usr/src/common/acpica/include/platform/acos2.h

1

```
*****
3565 Fri Jan 17 07:31:10 2014
new/usr/src/common/acpica/include/platform/acos2.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acos2.h - OS/2 specific defines, etc.
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #ifndef __ACOS2_H__
45 #define __ACOS2_H__
46 #define INCL_LONGLONG
47 #include <os2.h>

50 #define ACPI_MACHINE_WIDTH      32

52 #define COMPILER_DEPENDENT_INT64  long long
53 #define COMPILER_DEPENDENT_UINT64 unsigned long long
54 #define ACPI_USE_NATIVE_DIVIDE

56 #define ACPI_SYSTEM_XFACE        APIENTRY
57 #define ACPI_EXTERNAL_XFACE      APIENTRY
58 #define ACPI_INTERNAL_XFACE      APIENTRY
```

new/usr/src/common/acpica/include/platform/acos2.h

2

```
59 #define ACPI_INTERNAL_VAR_XFACE    APIENTRY

61 /*
62 * Some compilers complain about unused variables. Sometimes we don't want to
63 * use all the variables (most specifically for _THIS_MODULE). This allow us
64 * to tell the compiler warning in a per-variable manner that a variable
65 * is unused.
66 */
67 #define ACPI_UNUSED_VAR

69 #define ACPI_USE_STANDARD_HEADERS
70 #include <io.h>

72 #define ACPI_FLUSH_CPU_CACHE() Wbinvd()
73 void Wbinvd(void);

75 #define ACPI_ACQUIRE_GLOBAL_LOCK(GLptr, Acq)    Acq = OSPMAcquireGlobalLock(G
76 #define ACPI_RELEASE_GLOBAL_LOCK(GLptr, Pnd)    Pnd = OSPMReleaseGlobalLock(G
77 unsigned short OSPMAcquireGlobalLock (void *);
78 unsigned short OSPMReleaseGlobalLock (void *);

80 #define ACPI_SHIFT_RIGHT_64(n_hi, n_lo) \
81 { \
82     unsigned long long val = 0LL; \
83     val = n_lo | ( ((unsigned long long)h_hi) << 32 ); \
84     __llrotr (val,1); \
85     n_hi = (unsigned long)((val >> 32 ) & 0xffffffff ); \
86     n_lo = (unsigned long)(val & 0xffffffff); \
87 }

unchanged_portion_omitted
```

new/usr/src/common/acpica/include/platform/acsolaris.h

1

```
*****
2832 Fri Jan 17 07:31:10 2014
new/usr/src/common/acpica/include/platform/acsolaris.h
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 2011 Nexenta Systems, Inc. All rights reserved.
24  * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26  * Copyright 2013 Nexenta Systems, Inc. All rights reserved.
27 */

28 #ifndef _ACSOLARIS_H_
29 #define _ACSOLARIS_H_

31 #ifdef __cplusplus
32 extern "C" {
33 #endif

35 #ifdef _KERNEL
36 #include <sys/types.h>
37 #include <sys/sunddi.h>
38 #include <sys/varargs.h>
39 #include <sys/cpu.h>
40 #include <sys/thread.h>
41 #else
42 #include <ctype.h>
43 #include <stdarg.h>
44 #include <stdlib.h>
45 #include <string.h>
46 #include <unistd.h>
47 #endif

49 #ifdef __GNUC__
50 #include "acgcc.h"
51 #else
52 /* Function name used for debug output. */
53 #define ACPI_GET_FUNCTION_NAME __func__
54 #define ACPI_UNUSED_VAR
55 #endif

57 uint32_t __acpi_acquire_global_lock(void *);
58 uint32_t __acpi_release_global_lock(void *);
59 void __acpi_wbinvd(void);
```

new/usr/src/common/acpica/include/platform/acsolaris.h

2

```
61 #ifdef _ILP32
62 #define ACPI_MACHINE_WIDTH 32
63 #elif defined(_LP64)
64 #define ACPI_MACHINE_WIDTH 64
65 #endif

67 #define COMPILER_DEPENDENT_INT64 int64_t
68 #define COMPILER_DEPENDENT_UINT64 uint64_t

70 #define ACPI_CAST_PTHREAD_T(pthread) ((ACPI_THREAD_ID) (pthread))

72 #define ACPI_PRINTF_LIKE_FUNC

59 #define ACPI_UNUSED_VAR
74 #define ACPI_USE_NATIVE_DIVIDE
75 #define ACPI_FLUSH_CPU_CACHE() (__acpi_wbinvd())

77 #ifdef DEBUG
78 #define ACPI_DEBUG_OUTPUT
79 #define ACPI_DISASSEMBLER
80 #endif

82 #define ACPI_PACKED_POINTERS_NOT_SUPPORTED

84 /*
85  * Calling conventions:
86  *
87  * ACPI_SYSTEM_XFACE - Interfaces to host OS (handlers, threads)
88  * ACPI_EXTERNAL_XFACE - External ACPI interfaces
89  * ACPI_INTERNAL_XFACE - Internal ACPI interfaces
90  * ACPI_INTERNAL_VAR_XFACE - Internal variable-parameter list interfaces
91  */
92 #define ACPI_SYSTEM_XFACE
93 #define ACPI_EXTERNAL_XFACE
94 #define ACPI_INTERNAL_XFACE
95 #define ACPI_INTERNAL_VAR_XFACE

97 #define ACPI_ASM_MACROS
98 #define BREAKPOINT3
99 #define ACPI_DISABLE_IRQS() cli()
100 #define ACPI_ENABLE_IRQS() sti()
101 #define ACPI_ACQUIRE_GLOBAL_LOCK(Facs, Acq) \
102 ((Acq) = __acpi_acquire_global_lock(Facs))

104 #define ACPI_RELEASE_GLOBAL_LOCK(Facs, Acq) \
105 ((Acq) = __acpi_release_global_lock(Facs))

107 #ifdef __cplusplus
108 }
_____unchanged_portion_omitted_____
```

new/usr/src/common/acpica/include/platform/acwin.h

1

```
*****
5683 Fri Jan 17 07:31:11 2014
new/usr/src/common/acpica/include/platform/acwin.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acwin.h - OS specific defines, etc.
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #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
```

new/usr/src/common/acpica/include/platform/acwin.h

2

```
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, 0xFFFFFFFF
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_
```



new/usr/src/common/acpica/include/platform/acwin64.h

1

```
*****
3121 Fri Jan 17 07:31:11 2014
new/usr/src/common/acpica/include/platform/acwin64.h
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Name: acwin64.h - OS specific defines, etc.
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __ACWIN64_H__
45 #define __ACWIN64_H__

47 /*! [Begin] no source code translation (Keep the include) */

49 #include "acintel.h"
50 /*! [End] no source code translation */

52 #define ACPI_MACHINE_WIDTH 64

54 #define ACPI_USE_STANDARD_HEADERS

56 /*
57 * Handle platform- and compiler-specific assembly language differences.
58 */
```

new/usr/src/common/acpica/include/platform/acwin64.h

2

```
59 * Notes:
60 * 1) Interrupt 3 is used to break into a debugger
61 * 2) Interrupts are turned off during ACPI register setup
62 */

64 /*! [Begin] no source code translation */

66 #define ACPI_FLUSH_CPU_CACHE()

68 /*
69 * For Acpi applications, we don't want to try to access the global lock
70 */
71 #ifdef ACPI_APPLICATION
72 #define ACPI_ACQUIRE_GLOBAL_LOCK(GLptr, Acq) if (AcpiGbl_GlobalLockPresent
73 #define ACPI_RELEASE_GLOBAL_LOCK(GLptr, Pnd) if (AcpiGbl_GlobalLockPresent
74 #else

76 #define ACPI_ACQUIRE_GLOBAL_LOCK(GLptr, Acq)

78 #define ACPI_RELEASE_GLOBAL_LOCK(GLptr, Pnd)

80 #endif

82 /*! [End] no source code translation */

84 #endif /* __ACWIN_H__ */
```

new/usr/src/common/acpica/os\_specific/service\_layers/osfreebsdtbl.c

1

```
*****
24958 Fri Jan 17 07:31:11 2014
new/usr/src/common/acpica/os_specific/service_layers/osfreebsdtbl.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: osfreebsdtbl - FreeBSD OSL for obtaining ACPI tables
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acpidump.h"

46 #include <kenv.h>
47 #include <unistd.h>
48 #include <sys/param.h>
49 #include <sys/sysctl.h>

52 #define _COMPONENT          ACPI_OS_SERVICES
53 #define ACPI_MODULE_NAME    ("osfreebsdtbl")

56 /* Local prototypes */

58 static ACPI_STATUS
59 OslTableInitialize (
60     void);
```

new/usr/src/common/acpica/os\_specific/service\_layers/osfreebsdtbl.c

2

```
62 static ACPI_STATUS
63 OslMapTable (
64     ACPI_SIZE              Address,
65     char                    *Signature,
66     ACPI_TABLE_HEADER      **Table);

68 static ACPI_STATUS
69 OslAddTablesToList (
70     void);

72 static ACPI_STATUS
73 OslGetTableViaRoot (
74     char                    *Signature,
75     UINT32                  Instance,
76     ACPI_TABLE_HEADER      **Table,
77     ACPI_PHYSICAL_ADDRESS  *Address);

80 /* Hints for RSDP */

82 #define SYSTEM_KENV          "hint.acpi.0.rsdp"
83 #define SYSTEM_SYSCTL       "machdep.acpi_root"

85 /* Initialization flags */

87 UINT8                      Gbl_TableListInitialized = FALSE;
88 UINT8                      Gbl_MainTableObtained = FALSE;

90 /* Local copies of main ACPI tables */

92 ACPI_TABLE_RSDP             Gbl_Rsdp;
93 ACPI_TABLE_FADT             *Gbl_Fadt;
94 ACPI_TABLE_RSDT             *Gbl_Rsdt;
95 ACPI_TABLE_XSDT             *Gbl_Xsdt;

97 /* Fadt address */

99 ACPI_PHYSICAL_ADDRESS      Gbl_FadtAddress;

101 /* Revision of RSD PTR */

103 UINT8                      Gbl_Revision;

105 /* List of information about obtained ACPI tables */

107 typedef struct              table_info
108 {
109     struct table_info        *Next;
110     char                      Signature[4];
111     UINT32                    Instance;
112     ACPI_PHYSICAL_ADDRESS    Address;
113 } OSL_TABLE_INFO;

114 } OSL_TABLE_INFO;

116 OSL_TABLE_INFO              *Gbl_TableListHead = NULL;

119 /*****
120 *
121 * FUNCTION:      AcpiOsGetTableByAddress
122 *
123 * PARAMETERS:    Address      - Physical address of the ACPI table
124 *                Table        - Where a pointer to the table is returned
125 *
126 * RETURN:        Status; Table buffer is returned if AE_OK.
```

```

127 *          AE_NOT_FOUND: A valid table was not found at the address
128 *
129 * DESCRIPTION: Get an ACPI table via a physical memory address.
130 *
131 *****/
132
133 ACPI_STATUS
134 AcpiOsGetTableByAddress (
135     ACPI_PHYSICAL_ADDRESS    Address,
136     ACPI_TABLE_HEADER        **Table)
137 {
138     ACPI_TABLE_HEADER        *MappedTable;
139     ACPI_TABLE_HEADER        *LocalTable;
140     ACPI_STATUS               Status;
141
142     /* Validate the input physical address to avoid program crash */
143     if (Address < ACPI_HI_RSDP_WINDOW_BASE)
144     {
145         fprintf (stderr, "Invalid table address: 0x%8.8X%8.8X\n",
146             ACPI_FORMAT_UINT64 (Address));
147         return (AE_BAD_ADDRESS);
148     }
149
150     /* Map the table and validate it */
151     Status = OsMapTable (Address, NULL, &MappedTable);
152     if (ACPI_FAILURE (Status))
153     {
154         return (Status);
155     }
156
157     /* Copy table to local buffer and return it */
158     LocalTable = calloc (1, MappedTable->Length);
159     if (!LocalTable)
160     {
161         AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
162         return (AE_NO_MEMORY);
163     }
164
165     ACPI_MEMCPY (LocalTable, MappedTable, MappedTable->Length);
166     AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
167
168     *Table = LocalTable;
169     return (AE_OK);
170 }
171
172 /*****
173 *
174 * FUNCTION:    AcpiOsGetTableByName
175 *
176 * PARAMETERS:  Signature      - ACPI Signature for desired table. Must be
177 *                          a null terminated 4-character string.
178 *              Instance       - Multiple table support for SSDT/UEFI (0...n)
179 *                          Must be 0 for other tables.
180 *              Table          - Where a pointer to the table is returned
181 *              Address        - Where the table physical address is returned
182 *
183 * RETURN:     Status; Table buffer and physical address returned if AE_OK.
184 *             AE_LIMIT: Instance is beyond valid limit
185 *             AE_NOT_FOUND: A table with the signature was not found
186 *
187 * NOTE:       Assumes the input signature is uppercase.
188 *****/

```

```

193 *
194 *****/
195
196 ACPI_STATUS
197 AcpiOsGetTableByName (
198     char                *Signature,
199     UINT32              Instance,
200     ACPI_TABLE_HEADER    **Table,
201     ACPI_PHYSICAL_ADDRESS *Address)
202 {
203     ACPI_STATUS          Status;
204
205     /* Instance is only valid for SSDT/UEFI tables */
206
207     if (Instance &&
208         !ACPI_COMPARE_NAME (Signature, ACPI_SIG SSDT) &&
209         !ACPI_COMPARE_NAME (Signature, ACPI_SIG UEFI))
210     {
211         return (AE_LIMIT);
212     }
213
214     /* Initialize main tables */
215     Status = OsTableInitialize ();
216     if (ACPI_FAILURE (Status))
217     {
218         return (Status);
219     }
220
221     /*
222      * If one of the main ACPI tables was requested (RSDT/XSDT/FADT),
223      * simply return it immediately.
224      */
225     if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_XSDT))
226     {
227         if (!Gbl_Revision)
228         {
229             return (AE_NOT_FOUND);
230         }
231
232         *Address = Gbl_Rsdp.XsdtPhysicalAddress;
233         *Table = (ACPI_TABLE_HEADER *) Gbl_Xsdt;
234         return (AE_OK);
235     }
236
237     if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_RSDT))
238     {
239         if (!Gbl_Rsdp.RsdtPhysicalAddress)
240         {
241             return (AE_NOT_FOUND);
242         }
243
244         *Address = Gbl_Rsdp.RsdtPhysicalAddress;
245         *Table = (ACPI_TABLE_HEADER *) Gbl_Rsdt;
246         return (AE_OK);
247     }
248
249     if (ACPI_COMPARE_NAME (Signature, ACPI_SIG_FADT))
250     {
251         *Address = Gbl_FadtAddress;
252         *Table = (ACPI_TABLE_HEADER *) Gbl_Fadt;
253         return (AE_OK);
254     }
255
256     /* 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. */
393
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  }
419
420  /* Get RSDP from memory */
421
422  RsdpAddress = AcpiOsMapMemory (RsdpBase, RsdpSize);
423  if (!RsdpAddress)
424  {
425      return (AE_BAD_ADDRESS);
426  }
427
428  /* Search low memory for the RSDP */
429
430  TableAddress = AcpiTbScanMemoryForRsdp (RsdpAddress, RsdpSize);
431  if (!TableAddress)
432  {
433      AcpiOsUnmapMemory (RsdpAddress, RsdpSize);
434      return (AE_ERROR);
435  }
436
437  ACPI_MEMCPY (&Gbl_Rsdp, TableAddress, sizeof (Gbl_Rsdp));
438  AcpiOsUnmapMemory (RsdpAddress, RsdpSize);
439
440  /* Get XSDT from memory */
441
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      }
450
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  }
461
462  ACPI_MEMCPY (Gbl_Xsdt, MappedTable, MappedTable->Length);
463  AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
464  }
465
466  /* Get RSDT from memory */
467
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      }
476
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      }
486
487      ACPI_MEMCPY (Gbl_Rsdt, MappedTable, MappedTable->Length);
488      AcpiOsUnmapMemory (MappedTable, MappedTable->Length);
489  }
490
491  /* Get FADT from memory */
492
493  if (Gbl_Revision)
494  {
495      Gbl_FadtAddress = Gbl_Xsdt->TableOffsetEntry[0];
496  }
497  else
498  {
499      Gbl_FadtAddress = Gbl_Rsdt->TableOffsetEntry[0];
500  }
501
502  if (!Gbl_FadtAddress)
503  {
504      fprintf (stderr, "FADT: Table could not be found\n");
505      return (AE_ERROR);
506  }
507
508  Status = OslMapTable (Gbl_FadtAddress, ACPI_SIG_FADT, &MappedTable);
509  if (ACPI_FAILURE (Status))
510  {
511      return (Status);
512  }
513
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_XSDT) &&
581                 Gbl_Fadt->Xsdt)
582             {
583                 TableAddress = (ACPI_PHYSICAL_ADDRESS) Gbl_Fadt->Xsdt;
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     }
619 }

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     {
643         /* 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 }

```



new/usr/src/common/acpica/os\_specific/service\_layers/oslinuxtbl.c

1

```
*****
39193 Fri Jan 17 07:31:12 2014
new/usr/src/common/acpica/os_specific/service_layers/oslinuxtbl.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: oslinuxtbl - Linux OSL for obtaining ACPI tables
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acpidump.h"

47 #define _COMPONENT          ACPI_OS_SERVICES
48 #define ACPI_MODULE_NAME    ("oslinuxtbl")

51 #ifndef PATH_MAX
52 #define PATH_MAX 256
53 #endif

56 /* List of information about obtained ACPI tables */

58 typedef struct          table_info
59 {
60     struct table_info    *Next;
```

new/usr/src/common/acpica/os\_specific/service\_layers/oslinuxtbl.c

2

```
61     UINT32                Instance;
62     char                   Signature[ACPI_NAME_SIZE];

64 } OSL_TABLE_INFO;

66 /* Local prototypes */

68 static ACPI_STATUS
69 OslTableInitialize (
70     void);

72 static ACPI_STATUS
73 OslTableNameFromFile (
74     char                   *Filename,
75     char                   *Signature,
76     UINT32                 *Instance);

78 static ACPI_STATUS
79 OslAddTableToList (
80     char                   *Signature,
81     UINT32                 Instance);

83 static ACPI_STATUS
84 OslReadTableFromFile (
85     char                   *Filename,
86     ACPI_SIZE              FileOffset,
87     char                   *Signature,
88     ACPI_TABLE_HEADER      **Table);

90 static ACPI_STATUS
91 OslMapTable (
92     ACPI_SIZE              Address,
93     char                   *Signature,
94     ACPI_TABLE_HEADER      **Table);

96 static void
97 OslUnmapTable (
98     ACPI_TABLE_HEADER      *Table);

100 static ACPI_PHYSICAL_ADDRESS
101 OslFindRsdpViaEfi (
102     void);

104 static ACPI_STATUS
105 OslLoadRsdp (
106     void);

108 static ACPI_STATUS
109 OslListCustomizedTables (
110     char                   *Directory);

112 static ACPI_STATUS
113 OslGetCustomizedTable (
114     char                   *Pathname,
115     char                   *Signature,
116     UINT32                 Instance,
117     ACPI_TABLE_HEADER      **Table,
118     ACPI_PHYSICAL_ADDRESS *Address);

120 static ACPI_STATUS
121 OslListBiosTables (
122     void);

124 static ACPI_STATUS
125 OslGetBiosTable (
126     char                   *Signature,
```

```

127     UINT32      Instance,
128     ACPI_TABLE_HEADER **Table,
129     ACPI_PHYSICAL_ADDRESS *Address);

131 static ACPI_STATUS
132 OslGetLastStatus (
133     ACPI_STATUS      DefaultStatus);

136 /* File locations */

138 #define DYNAMIC_TABLE_DIR    "/sys/firmware/acpi/tables/dynamic"
139 #define STATIC_TABLE_DIR    "/sys/firmware/acpi/tables"
140 #define EFI_SYSTAB          "/sys/firmware/efi/systab"

142 /* Should we get dynamically loaded SSDTs from DYNAMIC_TABLE_DIR? */

144 UINT8      Gbl_DumpDynamicTables = TRUE;

146 /* Initialization flags */

148 UINT8      Gbl_TableListInitialized = FALSE;

150 /* Local copies of main ACPI tables */

152 ACPI_TABLE_RSDP      Gbl_Rsdp;
153 ACPI_TABLE_FADT      *Gbl_Fadt = NULL;
154 ACPI_TABLE_RSDT      *Gbl_Rsdt = NULL;
155 ACPI_TABLE_XSDT      *Gbl_Xsdt = NULL;

157 /* Table addresses */

159 ACPI_PHYSICAL_ADDRESS Gbl_FadtAddress = 0;
160 ACPI_PHYSICAL_ADDRESS Gbl_RsdpAddress = 0;

162 /* Revision of RSD PTR */

164 UINT8      Gbl_Revision = 0;

166 OSL_TABLE_INFO *Gbl_TableListHead = NULL;
167 UINT32      Gbl_TableCount = 0;

170 /*****
171  *
172  * FUNCTION:    OslGetLastStatus
173  *
174  * PARAMETERS:  DefaultStatus - Default error status to return
175  *
176  * RETURN:      Status; Converted from errno.
177  *
178  * DESCRIPTION: Get last errno and conver it to ACPI_STATUS.
179  *
180  *****/

182 static ACPI_STATUS
183 OslGetLastStatus (
184     ACPI_STATUS      DefaultStatus)
185 {
187     switch (errno)
188     {
189     case EACCES:
190     case EPERM:
192         return (AE_ACCESS);

```

```

194     case ENOENT:

196         return (AE_NOT_FOUND);

198     case ENOMEM:

200         return (AE_NO_MEMORY);

202     default:

204         return (DefaultStatus);
205     }
206 }

209 /*****
210  *
211  * FUNCTION:    AcpiOsGetTableByAddress
212  *
213  * PARAMETERS:  Address      - Physical address of the ACPI table
214  *              Table        - Where a pointer to the table is returned
215  *
216  * RETURN:      Status; Table buffer is returned if AE_OK.
217  *              AE_NOT_FOUND: A valid table was not found at the address
218  *
219  * DESCRIPTION: Get an ACPI table via a physical memory address.
220  *
221  *****/

223 ACPI_STATUS
224 AcpiOsGetTableByAddress (
225     ACPI_PHYSICAL_ADDRESS Address,
226     ACPI_TABLE_HEADER **Table)
227 {
228     UINT32      TableLength;
229     ACPI_TABLE_HEADER *MappedTable;
230     ACPI_TABLE_HEADER *LocalTable = NULL;
231     ACPI_STATUS Status = AE_OK;

234     /* Get main ACPI tables from memory on first invocation of this function */

236     Status = OslTableInitialize ();
237     if (ACPI_FAILURE (Status))
238     {
239         return (Status);
240     }

242     /* Map the table and validate it */

244     Status = OslMapTable (Address, NULL, &MappedTable);
245     if (ACPI_FAILURE (Status))
246     {
247         return (Status);
248     }

250     /* Copy table to local buffer and return it */

252     TableLength = ApGetTableLength (MappedTable);
253     if (TableLength == 0)
254     {
255         Status = AE_BAD_HEADER;
256         goto ErrorExit;
257     }

```

```

259     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     }
327
328     if (ACPI_FAILURE (Status) && Status == AE_LIMIT)
329     {
330         if (Gbl_DumpDynamicTables)
331         {
332             /* Attempt to get a dynamic table */
333
334             Status = OslGetCustomizedTable (DYNAMIC_TABLE_DIR, Signature,
335                                             Instance, Table, Address);
336         }
337     }
338
339     return (Status);
340 }
341
342
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  *****/
356
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;
366
367     NewInfo = calloc (1, sizeof (OSL_TABLE_INFO));
368     if (!NewInfo)
369     {
370         return (AE_NO_MEMORY);
371     }
372
373     ACPI_MOVE_NAME (NewInfo->Signature, Signature);
374
375     if (!Gbl_TableListHead)
376     {
377         Gbl_TableListHead = NewInfo;
378     }
379     else
380     {
381         Next = Gbl_TableListHead;
382         while (1)
383         {
384             if (ACPI_COMPARE_NAME (Next->Signature, Signature))
385             {
386                 if (Next->Instance == Instance)
387                 {
388                     Found = TRUE;
389                 }
390             }
391             Next = Next->Next;
392         }
393     }
394     if (!Found)
395     {
396         NextInstance = NextInstance + 1;
397         NewInfo->Instance = NextInstance;
398         NewInfo->Next = Gbl_TableListHead;
399         Gbl_TableListHead = NewInfo;
400     }
401     return (AE_OK);
402 }
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

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%x", &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_RSMT, 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 (ACPI_SIG_RSDP, 0);
686     if (ACPI_FAILURE (Status))
687     {
688         return (Status);
689     }

691     Status = OslAddTableToList (ACPI_SIG_RSMT, 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/RSMT/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, ACPI_RSDP_NAME) ||
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)
1184     {
1185         AcpiOsUnmapMemory (Table, ApGetTableLength (Table));
1186     }
1187 }

1190 /*****
1191  *
1192  * FUNCTION:      OslTableNameFromFile
1193  *
1194  * PARAMETERS:    Filename          - File that contains the desired table
1195  *                Signature         - Pointer to 4-character buffer to store
1196  *                                extracted table signature.
1197  *                Instance          - Pointer to integer to store extracted
1198  *                                table instance number.
1199  *
1200  * RETURN:        Status; Table name is extracted if AE_OK.
1201  *
1202  * DESCRIPTION:    Extract table signature and instance number from a table file
1203  *                name.
1204  *
1205  *****/

1207 static ACPI_STATUS
1208 OslTableNameFromFile (
1209     char          *Filename,
1210     char          *Signature,
1211     UINT32        *Instance)
1212 {
1213
1214     /* Ignore meaningless files */
1215
1216     if (strlen (Filename) < ACPI_NAME_SIZE)
1217     {
1218         return (AE_BAD_SIGNATURE);
1219     }
1220
1221     /* Extract instance number */
1222
1223     if (isdigit ((int) Filename[ACPI_NAME_SIZE]))
1224     {
1225         sscanf (&Filename[ACPI_NAME_SIZE], "%d", Instance);
1226     }
1227     else if (strlen (Filename) != ACPI_NAME_SIZE)
1228     {
1229         return (AE_BAD_SIGNATURE);
1230     }
1231     else
1232     {
1233         *Instance = 0;
1234     }
1235
1236     /* Extract signature */
1237
1238     ACPI_MOVE_NAME (Signature, Filename);
1239     return (AE_OK);
1240 }

1243 /*****
1244  *
1245  * FUNCTION:      OslReadTableFromFile
1246  *
1247  * PARAMETERS:    Filename          - File that contains the desired table
1248  *                FileOffset        - Offset of the table in file

```

```

1249  *                Signature          - Optional ACPI Signature for desired table.
1250  *                                A null terminated 4-character string.
1251  *                Table              - Where a pointer to the table is returned
1252  *
1253  * RETURN:        Status; Table buffer is returned if AE_OK.
1254  *
1255  * DESCRIPTION:    Read a ACPI table from a file.
1256  *
1257  *****/

1259 static ACPI_STATUS
1260 OslReadTableFromFile (
1261     char          *Filename,
1262     ACPI_SIZE     FileOffset,
1263     char          *Signature,
1264     ACPI_TABLE_HEADER **Table)
1265 {
1266     FILE          *TableFile;
1267     ACPI_TABLE_HEADER Header;
1268     ACPI_TABLE_HEADER *LocalTable = NULL;
1269     UINT32        TableLength;
1270     INT32         Count;
1271     UINT32        Total = 0;
1272     ACPI_STATUS    Status = AE_OK;

1275     /* Open the file */
1276
1277     TableFile = fopen (Filename, "rb");
1278     if (TableFile == NULL)
1279     {
1280         fprintf (stderr, "Could not open table file: %s\n", Filename);
1281         return (OslGetLastStatus (AE_NOT_FOUND));
1282     }
1283
1284     fseek (TableFile, FileOffset, SEEK_SET);
1285
1286     /* Read the Table header to get the table length */
1287
1288     Count = fread (&Header, 1, sizeof (ACPI_TABLE_HEADER), TableFile);
1289     if (Count != sizeof (ACPI_TABLE_HEADER))
1290     {
1291         fprintf (stderr, "Could not read table header: %s\n", Filename);
1292         Status = AE_BAD_HEADER;
1293         goto ErrorExit;
1294     }
1295
1296     /* If signature is specified, it must match the table */
1297
1298     if (Signature &&
1299         !ACPI_COMPARE_NAME (Signature, Header.Signature))
1300     {
1301         fprintf (stderr, "Incorrect signature: Expecting %4.4s, found %4.4s\n",
1302             Signature, Header.Signature);
1303         Status = AE_BAD_SIGNATURE;
1304         goto ErrorExit;
1305     }
1306
1307     TableLength = ApGetTableLength (&Header);
1308     if (TableLength == 0)
1309     {
1310         Status = AE_BAD_HEADER;
1311         goto ErrorExit;
1312     }
1313
1314     /* Read the entire table into a local buffer */

```

```

1316 LocalTable = calloc (1, TableLength);
1317 if (!LocalTable)
1318 {
1319     fprintf (stderr,
1320             "%4.4s: Could not allocate buffer for table of length %X\n",
1321             Header.Signature, TableLength);
1322     Status = AE_NO_MEMORY;
1323     goto ErrorExit;
1324 }
1326 fseek (TableFile, FileOffset, SEEK_SET);
1328 while (!feof (TableFile) && Total < TableLength)
1329 {
1330     Count = fread (LocalTable, 1, TableLength-Total, TableFile);
1331     if (Count < 0)
1332     {
1333         fprintf (stderr, "%4.4s: Could not read table content\n",
1334                 Header.Signature);
1335         Status = AE_INVALID_TABLE_LENGTH;
1336         goto ErrorExit;
1337     }
1339     Total += Count;
1340 }
1342 /* Validate checksum */
1344 (void) ApIsValidChecksum (LocalTable);
1346 ErrorExit:
1347     fclose (TableFile);
1348     *Table = LocalTable;
1349     return (Status);
1350 }
1353 /*****
1354  *
1355  * FUNCTION:      OslGetCustomizedTable
1356  *
1357  * PARAMETERS:    Pathname      - Directory to find Linux customized table
1358  *                Signature     - ACPI Signature for desired table. Must be
1359  *                               a null terminated 4-character string.
1360  *                Instance      - Multiple table support for SSDT/UEFI (0...n)
1361  *                               Must be 0 for other tables.
1362  *                Table         - Where a pointer to the table is returned
1363  *                Address       - Where the table physical address is returned
1364  *
1365  * RETURN:        Status; Table buffer is returned if AE_OK.
1366  *                AE_LIMIT: Instance is beyond valid limit
1367  *                AE_NOT_FOUND: A table with the signature was not found
1368  *
1369  * DESCRIPTION:   Get an OS customized table.
1370  *
1371  *****/
1373 static ACPI_STATUS
1374 OslGetCustomizedTable (
1375     char                *Pathname,
1376     char                *Signature,
1377     UINT32              Instance,
1378     ACPI_TABLE_HEADER **Table,
1379     ACPI_PHYSICAL_ADDRESS *Address)
1380 {

```

```

1381     void                *TableDir;
1382     UINT32              CurrentInstance = 0;
1383     char                TempName[ACPI_NAME_SIZE];
1384     char                TableFilename[PATH_MAX];
1385     char                *Filename;
1386     ACPI_STATUS          Status;
1389     /* Open the directory for customized tables */
1391     TableDir = AcpiOsOpenDirectory (Pathname, "", REQUEST_FILE_ONLY);
1392     if (!TableDir)
1393     {
1394         return (OslGetLastStatus (AE_NOT_FOUND));
1395     }
1397     /* Attempt to find the table in the directory */
1399     while ((Filename = AcpiOsGetNextFilename (TableDir)))
1400     {
1401         /* Ignore meaningless files */
1403         if (!ACPI_COMPARE_NAME (Filename, Signature))
1404         {
1405             continue;
1406         }
1408         /* Extract table name and instance number */
1410         Status = OslTableNameFromFile (Filename, TempName, &CurrentInstance);
1412         /* Ignore meaningless files */
1414         if (ACPI_FAILURE (Status) || CurrentInstance != Instance)
1415         {
1416             continue;
1417         }
1419         /* Create the table pathname */
1421         if (Instance != 0)
1422         {
1423             sprintf (TableFilename, "%s/%4.4s%d", Pathname, TempName, Instance);
1424         }
1425         else
1426         {
1427             sprintf (TableFilename, "%s/%4.4s", Pathname, TempName);
1428         }
1429         break;
1430     }
1432     AcpiOsCloseDirectory (TableDir);
1434     if (!Filename)
1435     {
1436         return (AE_LIMIT);
1437     }
1439     /* There is no physical address saved for customized tables, use zero */
1441     *Address = 0;
1442     Status = OslReadTableFromFile (TableFilename, 0, NULL, Table);
1444     return (Status);
1445 }

```

new/usr/src/common/acpica/os\_specific/service\_layers/osunixdir.c

1

```
*****
7065 Fri Jan 17 07:31:12 2014
new/usr/src/common/acpica/os_specific/service_layers/osunixdir.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: osunixdir - Unix directory access interfaces
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include <acpi.h>
45
46 #include <stdio.h>
47 #include <stdlib.h>
48 #include <string.h>
49 #include <dirent.h>
50 #include <fnmatch.h>
51 #include <ctype.h>
52 #include <sys/stat.h>
53
54 /*
55 * Allocated structure returned from OsOpenDirectory
56 */
57 typedef struct ExternalFindInfo
58 {
```

new/usr/src/common/acpica/os\_specific/service\_layers/osunixdir.c

2

```
59 char *DirPathname;
60 DIR *DirPtr;
61 char temp_buffer[256];
62 char *WildcardSpec;
63 char RequestedFileType;
64
65 } EXTERNAL_FIND_INFO;
66
67 /*****
68 *
69 * FUNCTION: AcpiOsOpenDirectory
70 *
71 * PARAMETERS: DirPathname - Full pathname to the directory
72 * WildcardSpec - string of the form "*.c", etc.
73 *
74 * RETURN: A directory "handle" to be used in subsequent search operations.
75 * NULL returned on failure.
76 *
77 * DESCRIPTION: Open a directory in preparation for a wildcard search
78 *
79 *
80 *****/
81
82 void *
83 AcpiOsOpenDirectory (
84 char *DirPathname,
85 char *WildcardSpec,
86 char RequestedFileType)
87 {
88 EXTERNAL_FIND_INFO *ExternalInfo;
89 DIR *dir;
90
91 /* Allocate the info struct that will be returned to the caller */
92
93 ExternalInfo = calloc (1, sizeof (EXTERNAL_FIND_INFO));
94 if (!ExternalInfo)
95 {
96 return (NULL);
97 }
98
99 /* Get the directory stream */
100
101 dir = opendir (DirPathname);
102 if (!dir)
103 {
104 fprintf (stderr, "Cannot open directory - %s\n", DirPathname);
105 free (ExternalInfo);
106 return (NULL);
107 }
108
109 /* Save the info in the return structure */
110
111 ExternalInfo->WildcardSpec = WildcardSpec;
112 ExternalInfo->RequestedFileType = RequestedFileType;
113 ExternalInfo->DirPathname = DirPathname;
114 ExternalInfo->DirPtr = dir;
115 return (ExternalInfo);
116 }
117
118 /*****
119 *
120 * FUNCTION: AcpiOsGetNextFilename
121 *
122 * PARAMETERS: DirHandle - Created via AcpiOsOpenDirectory
```

```

125 *
126 * RETURN:      Next filename matched. NULL if no more matches.
127 *
128 * DESCRIPTION: Get the next file in the directory that matches the wildcard
129 *               specification.
130 *
131 *****/

133 char *
134 AcpiOsGetNextFilename (
135     void                *DirHandle)
136 {
137     EXTERNAL_FIND_INFO *ExternalInfo = DirHandle;
138     struct dirent      *dir_entry;
139     char               *temp_str;
140     int                str_len;
141     struct stat        temp_stat;
142     int                err;

145     while ((dir_entry = readdir (ExternalInfo->DirPtr))
146     {
147         if (!fnmatch (ExternalInfo->WildcardSpec, dir_entry->d_name, 0))
148         {
149             if (dir_entry->d_name[0] == '.')
150             {
151                 continue;
152             }

154             str_len = strlen (dir_entry->d_name) +
155                 strlen (ExternalInfo->DirPathname) + 2;

157             temp_str = calloc (str_len, 1);
158             if (!temp_str)
159             {
160                 fprintf (stderr,
161                     "Could not allocate buffer for temporary string\n");
162                 return (NULL);
163             }

165             strcpy (temp_str, ExternalInfo->DirPathname);
166             strcat (temp_str, "/");
167             strcat (temp_str, dir_entry->d_name);

169             err = stat (temp_str, &temp_stat);
170             if (err == -1)
171             {
172                 fprintf (stderr,
173                     "Cannot stat file (should not happen) - %s\n",
174                     temp_str);
175                 free (temp_str);
176                 return (NULL);
177             }

179             free (temp_str);

181             if ((S_ISDIR (temp_stat.st_mode)
182                 && (ExternalInfo->RequestedFileType == REQUEST_DIR_ONLY))
183                 ||
184                 (!S_ISDIR (temp_stat.st_mode)
185                 && ExternalInfo->RequestedFileType == REQUEST_FILE_ONLY))
186             {
187                 /* copy to a temp buffer because dir_entry struct is on the stack */

189                 strcpy (ExternalInfo->temp_buffer, dir_entry->d_name);
190                 return (ExternalInfo->temp_buffer);

```

```

191     }
192     }
193 }

195     return (NULL);
196 }

199 *****/
200 *
201 * FUNCTION:      AcpiOsCloseDirectory
202 *
203 * PARAMETERS:    DirHandle          - Created via AcpiOsOpenDirectory
204 *
205 * RETURN:        None.
206 *
207 * DESCRIPTION:    Close the open directory and cleanup.
208 *
209 *****/

211 void
212 AcpiOsCloseDirectory (
213     void                *DirHandle)
214 {
215     EXTERNAL_FIND_INFO *ExternalInfo = DirHandle;

218     /* Close the directory and free allocations */

220     closedir (ExternalInfo->DirPtr);
221     free (DirHandle);
222 }

```

new/usr/src/common/acpica/os\_specific/service\_layers/osunixmap.c

1

```
*****
5179 Fri Jan 17 07:31:12 2014
new/usr/src/common/acpica/os_specific/service_layers/osunixmap.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: osunixmap - Unix OSL for file mappings
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acpidump.h"
45 #include <unistd.h>
46 #include <sys/mman.h>
47 #ifdef _FreeBSD
48 #include <sys/param.h>
49 #endif

51 #define _COMPONENT          ACPI_OS_SERVICES
52 #define ACPI_MODULE_NAME    ("osunixmap")

55 #ifndef O_BINARY
56 #define O_BINARY 0
57 #endif

59 #ifdef _FreeBSD
60 #define MMAP_FLAGS          MAP_SHARED
```

new/usr/src/common/acpica/os\_specific/service\_layers/osunixmap.c

2

```
61 #else
62 #define MMAP_FLAGS          MAP_PRIVATE
63 #endif

65 #define SYSTEM_MEMORY      "/dev/mem"

68 /*****
69 *
70 * FUNCTION:      AcpiOsGetPageSize
71 *
72 * PARAMETERS:    None
73 *
74 * RETURN:        Page size of the platform.
75 *
76 * DESCRIPTION:   Obtain page size of the platform.
77 *
78 *****/

80 static ACPI_SIZE
81 AcpiOsGetPageSize (
82     void)
83 {
84
85     #ifdef PAGE_SIZE
86         return PAGE_SIZE;
87     #else
88         return sysconf (_SC_PAGESIZE);
89     #endif
90 }

93 /*****
94 *
95 * FUNCTION:      AcpiOsMapMemory
96 *
97 * PARAMETERS:    Where - Physical address of memory to be mapped
98 *                Length - How much memory to map
99 *
100 * RETURN:        Pointer to mapped memory. Null on error.
101 *
102 * DESCRIPTION:   Map physical memory into local address space.
103 *
104 *****/

106 void *
107 AcpiOsMapMemory (
108     ACPI_PHYSICAL_ADDRESS Where,
109     ACPI_SIZE Length)
110 {
111     UINT8 *MappedMemory;
112     ACPI_PHYSICAL_ADDRESS Offset;
113     ACPI_SIZE PageSize;
114     int fd;

117     fd = open (SYSTEM_MEMORY, O_RDONLY | O_BINARY);
118     if (fd < 0)
119     {
120         fprintf (stderr, "Cannot open %s\n", SYSTEM_MEMORY);
121         return (NULL);
122     }

124     /* Align the offset to use mmap */

126     PageSize = AcpiOsGetPageSize ();
```

```
127     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 }
```

new/usr/src/common/acpica/os\_specific/service\_layers/osunixxf.c

1

```
*****
35731 Fri Jan 17 07:31:12 2014
new/usr/src/common/acpica/os_specific/service_layers/osunixxf.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: osunixxf - UNIX OSL interfaces
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 /*
46 * These interfaces are required in order to compile the ASL compiler and the
47 * various ACPICA tools under Linux or other Unix-like system.
48 */
49 #include "acpi.h"
50 #include "accommon.h"
51 #include "amlcode.h"
52 #include "acparser.h"
53 #include "acdebug.h"

55 #include <stdio.h>
56 #include <stdlib.h>
```

new/usr/src/common/acpica/os\_specific/service\_layers/osunixxf.c

2

```
57 #include <stdarg.h>
58 #include <unistd.h>
59 #include <sys/time.h>
60 #include <semaphore.h>
61 #include <pthread.h>
62 #include <errno.h>

64 #define _COMPONENT ACPI_OS_SERVICES
65     ACPI_MODULE_NAME    ("osunixxf")

68 FILE                *AcpiGbl_OutputFile;
69 BOOLEAN              AcpiGbl_DebugTimeout = FALSE;

72 /* Upcalls to AcpiExec */

74 ACPI_PHYSICAL_ADDRESS
75 AeLocalGetRootPointer (
76     void);

78 void
79 AeTableOverride (
80     ACPI_TABLE_HEADER    *ExistingTable,
81     ACPI_TABLE_HEADER    **NewTable);

83 typedef void* (*PTHREAD_CALLBACK) (void *);

85 /* Buffer used by AcpiOsVprintf */

87 #define ACPI_VPRINTF_BUFFER_SIZE    512
88 #define _ASCII_NEWLINE              '\n'

90 /* Terminal support for AcpiExec only */

92 #ifdef ACPI_EXEC_APP
93 #include <termios.h>

95 struct termios        OriginalTermAttributes;

97 ACPI_STATUS
98 AcpiUtReadLine (
99     char                *Buffer,
100     UINT32              BufferLength,
101     UINT32              *BytesRead);

103 static void
104 OsEnterLineEditMode (
105     void);

107 static void
108 OsExitLineEditMode (
109     void);

112 /*****
113 *
114 * FUNCTION:      OsEnterLineEditMode, OsExitLineEditMode
115 *
116 * PARAMETERS:    None
117 *
118 * RETURN:        None
119 *
120 * DESCRIPTION:   Enter/Exit the raw character input mode for the terminal.
121 *
122 * Interactive line-editing support for the AML debugger. Used with the
```



```

123 * common/acgetline module.
124 *
125 * readline() is not used because of non-portability. It is not available
126 * on all systems, and if it is, often the package must be manually installed.
127 *
128 * Therefore, we use the POSIX tcgetattr/tcsetattr and do the minimal line
129 * editing that we need in AcpiOsGetLine.
130 *
131 * If the POSIX tcgetattr/tcsetattr interfaces are unavailable, these
132 * calls will also work:
133 *   For OsEnterLineEditMode: system ("stty cbreak -echo")
134 *   For OsExitLineEditMode: system ("stty cooked echo")
135 *
136 *****/

138 static void
139 OsEnterLineEditMode (
140     void)
141 {
142     struct termios      LocalTermAttributes;

145     /* Get and keep the original attributes */

147     if (tcgetattr (STDIN_FILENO, &OriginalTermAttributes))
148     {
149         fprintf (stderr, "Could not get/set terminal attributes!\n");
150         return;
151     }

153     /* Set the new attributes to enable raw character input */

155     memcpy (&LocalTermAttributes, &OriginalTermAttributes,
156         sizeof (struct termios));

158     LocalTermAttributes.c_lflag &= ~(ICANON | ECHO);
159     LocalTermAttributes.c_cc[VMIN] = 1;
160     LocalTermAttributes.c_cc[VTIME] = 0;

162     tcsetattr (STDIN_FILENO, TCSANOW, &LocalTermAttributes);
163 }

165 static void
166 OsExitLineEditMode (
167     void)
168 {
169     /* Set terminal attributes back to the original values */

171     tcsetattr (STDIN_FILENO, TCSANOW, &OriginalTermAttributes);
172 }

175 #else

177 /* These functions are not needed for other ACPICA utilities */

179 #define OsEnterLineEditMode()
180 #define OsExitLineEditMode()
181 #endif

184 *****/
185 *
186 * FUNCTION:      AcpiOsInitialize, AcpiOsTerminate
187 *
188 * PARAMETERS:    None

```

```

189 *
190 * RETURN:        Status
191 *
192 * DESCRIPTION:    Initialize and terminate this module.
193 *
194 *****/

196 ACPI_STATUS
197 AcpiOsInitialize (
198     void)
199 {
201     AcpiGbl_OutputFile = stdout;

203     OsEnterLineEditMode ();
204     return (AE_OK);
205 }

207 ACPI_STATUS
208 AcpiOsTerminate (
209     void)
210 {
212     OsExitLineEditMode ();
213     return (AE_OK);
214 }

217 *****/
218 *
219 * FUNCTION:      AcpiOsGetRootPointer
220 *
221 * PARAMETERS:    None
222 *
223 * RETURN:        RSDP physical address
224 *
225 * DESCRIPTION:    Gets the ACPI root pointer (RSDP)
226 *
227 *****/

229 ACPI_PHYSICAL_ADDRESS
230 AcpiOsGetRootPointer (
231     void)
232 {
234     return (AeLocalGetRootPointer ());
235 }

238 *****/
239 *
240 * FUNCTION:      AcpiOsPredefinedOverride
241 *
242 * PARAMETERS:    InitVal          - Initial value of the predefined object
243 *                NewVal           - The new value for the object
244 *
245 * RETURN:        Status, pointer to value. Null pointer returned if not
246 *                overriding.
247 *
248 * DESCRIPTION:    Allow the OS to override predefined names
249 *
250 *****/

252 ACPI_STATUS
253 AcpiOsPredefinedOverride (
254     const ACPI_PREDEFINED_NAMES *InitVal,

```

```

255     ACPI_STRING          *NewVal)
256 {
257
258     if (!InitVal || !NewVal)
259     {
260         return (AE_BAD_PARAMETER);
261     }
262
263     *NewVal = NULL;
264     return (AE_OK);
265 }
266
267 /*****
268 *
269 * FUNCTION:      AcpiOsTableOverride
270 *
271 * PARAMETERS:    ExistingTable      - Header of current table (probably
272 *                                     firmware)
273 *                                     NewTable      - Where an entire new table is returned.
274 *
275 * RETURN:        Status, pointer to new table. Null pointer returned if no
276 *                                     table is available to override
277 *
278 * DESCRIPTION:    Return a different version of a table if one is available
279 *
280 *
281 *****/
282
283 ACPI_STATUS
284 AcpiOsTableOverride (
285     ACPI_TABLE_HEADER      *ExistingTable,
286     ACPI_TABLE_HEADER      **NewTable)
287 {
288
289     if (!ExistingTable || !NewTable)
290     {
291         return (AE_BAD_PARAMETER);
292     }
293
294     *NewTable = NULL;
295
296 #ifdef ACPI_EXEC_APP
297
298     AeTableOverride (ExistingTable, NewTable);
299     return (AE_OK);
300 #else
301
302     return (AE_NO_ACPI_TABLES);
303 #endif
304 }
305
306 /*****
307 *
308 * FUNCTION:      AcpiOsPhysicalTableOverride
309 *
310 * PARAMETERS:    ExistingTable      - Header of current table (probably firmware)
311 *                                     NewAddress      - Where new table address is returned
312 *                                     (Physical address)
313 *                                     NewTableLength    - Where new table length is returned
314 *
315 * RETURN:        Status, address/length of new table. Null pointer returned
316 *                                     if no table is available to override.
317 *
318 * DESCRIPTION:    Returns AE_SUPPORT, function not used in user space.
319 *
320 *****/

```

```

321 *****/
322
323 ACPI_STATUS
324 AcpiOsPhysicalTableOverride (
325     ACPI_TABLE_HEADER      *ExistingTable,
326     ACPI_PHYSICAL_ADDRESS  *NewAddress,
327     UINT32                  *NewTableLength)
328 {
329
330     return (AE_SUPPORT);
331 }
332
333 /*****
334 *
335 * FUNCTION:      AcpiOsRedirectOutput
336 *
337 * PARAMETERS:    Destination      - An open file handle/pointer
338 *
339 * RETURN:        None
340 *
341 * DESCRIPTION:    Causes redirect of AcpiOsPrintf and AcpiOsVprintf
342 *
343 *****/
344
345 void
346 AcpiOsRedirectOutput (
347     void              *Destination)
348 {
349
350     AcpiGbl_OutputFile = Destination;
351 }
352
353 /*****
354 *
355 * FUNCTION:      AcpiOsPrintf
356 *
357 * PARAMETERS:    fmt, ...          - Standard printf format
358 *
359 * RETURN:        None
360 *
361 * DESCRIPTION:    Formatted output. Note: very similar to AcpiOsVprintf
362 *                                     (performance), changes should be tracked in both functions.
363 *
364 *****/
365
366 void ACPI_INTERNAL_VAR_XFACE
367 AcpiOsPrintf (
368     const char      *Fmt,
369     ...)
370 {
371     va_list          Args;
372     UINT8             Flags;
373
374     Flags = AcpiGbl_DbOutputFlags;
375     if (Flags & ACPI_DB_REDIRECTABLE_OUTPUT)
376     {
377         /* Output is redirectable to either a file (if open) or the console */
378
379         if (AcpiGbl_DebugFile)
380         {
381             /* Output file is open, send the output there */
382
383             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!) */
393
394     Flags |= ACPI_DB_CONSOLE_OUTPUT;
395 }
396 }
397
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 }
405
406 /*****
407 *
408 * FUNCTION:      AcpiOsVprintf
409 *
410 * PARAMETERS:    fmt          - Standard printf format
411 *                args         - Argument list
412 *
413 * RETURN:        None
414 *
415 * DESCRIPTION:   Formatted output with argument list pointer. Note: very
416 *                similar to AcpiOsPrintf, changes should be tracked in both
417 *                functions.
418 *
419 *
420 *****/
421
422 void
423 AcpiOsVprintf (
424     const char    *Fmt,
425     va_list       Args)
426 {
427     UINT8         Flags;
428     char          Buffer[ACPI_VPRINTF_BUFFER_SIZE];
429
430     /*
431     * We build the output string in a local buffer because we may be
432     * outputting the buffer twice. Using vfprintf is problematic because
433     * some implementations modify the args pointer/structure during
434     * execution. Thus, we use the local buffer for portability.
435     *
436     * Note: Since this module is intended for use by the various ACPICA
437     * utilities/applications, we can safely declare the buffer on the stack.
438     * Also, This function is used for relatively small error messages only.
439     */
440     vsnprintf (Buffer, ACPI_VPRINTF_BUFFER_SIZE, Fmt, Args);
441
442     Flags = AcpiGbl_DbOutputFlags;
443     if (Flags & ACPI_DB_REDIRECTABLE_OUTPUT)
444     {
445         /* Output is directable to either a file (if open) or the console */
446
447         if (AcpiGbl_DebugFile)
448         {
449             /* Output file is open, send the output there */
450
451             fputs (Buffer, AcpiGbl_DebugFile);

```

```

453     }
454     else
455     {
456         /* No redirection, send output to console (once only!) */
457
458         Flags |= ACPI_DB_CONSOLE_OUTPUT;
459     }
460 }
461
462 if (Flags & ACPI_DB_CONSOLE_OUTPUT)
463 {
464     fputs (Buffer, AcpiGbl_OutputFile);
465 }
466 }
467
468 #ifndef ACPI_EXEC_APP
469 /*****
470 *
471 * FUNCTION:      AcpiOsGetLine
472 *
473 * PARAMETERS:    Buffer          - Where to return the command line
474 *                BufferLength    - Maximum length of Buffer
475 *                BytesRead      - Where the actual byte count is returned
476 *
477 * RETURN:        Status and actual bytes read
478 *
479 * DESCRIPTION:   Get the next input line from the terminal. NOTE: For the
480 *                AcpiExec utility, we use the acgetline module instead to
481 *                provide line-editing and history support.
482 *
483 *
484 *****/
485
486 ACPI_STATUS
487 AcpiOsGetLine (
488     char          *Buffer,
489     UINT32        BufferLength,
490     UINT32        *BytesRead)
491 {
492     int           InputChar;
493     UINT32        EndOfLine;
494
495     /* Standard AcpiOsGetLine for all utilities except AcpiExec */
496
497     for (EndOfLine = 0; ; EndOfLine++)
498     {
499         if (EndOfLine >= BufferLength)
500         {
501             return (AE_BUFFER_OVERFLOW);
502         }
503
504         if ((InputChar = getchar ()) == EOF)
505         {
506             return (AE_ERROR);
507         }
508
509         if (!InputChar || InputChar == _ASCII_NEWLINE)
510         {
511             break;
512         }
513
514         Buffer[EndOfLine] = (char) InputChar;
515     }
516
517     /* 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);
652 }

654 ACPI_STATUS
655 AcpiOsWaitSemaphore (
656     ACPI_HANDLE      Handle,
657     UINT32            Units,
658     UINT16            Timeout)
659 {
660     return (AE_OK);
661 }

663 ACPI_STATUS
664 AcpiOsSignalSemaphore (
665     ACPI_HANDLE      Handle,
666     UINT32            Units)
667 {
668     return (AE_OK);
669 }

671 #else
672 /*****
673  *
674  * FUNCTION:      AcpiOsCreateSemaphore
675  *
676  * PARAMETERS:    InitialUnits      - Units to be assigned to the new semaphore
677  *                OutHandle          - Where a handle will be returned
678  *
679  * RETURN:        Status
680  *
681  * DESCRIPTION:   Create an OS semaphore
682  *
683  *****/
685 ACPI_STATUS
686 AcpiOsCreateSemaphore (
687     UINT32            MaxUnits,
688     UINT32            InitialUnits,
689     ACPI_HANDLE        *OutHandle)
690 {
691     sem_t              *Sem;

694     if (!OutHandle)
695     {
696         return (AE_BAD_PARAMETER);
697     }

699 #ifdef __APPLE__
700     {
701         char              *SemaphoreName = tmpnam (NULL);

703         Sem = sem_open (SemaphoreName, O_EXCL|O_CREAT, 0755, InitialUnits);
704         if (!Sem)
705         {
706             return (AE_NO_MEMORY);
707         }
708         sem_unlink (SemaphoreName); /* This just deletes the name */
709     }
711 #else
712     Sem = AcpiOsAllocate (sizeof (sem_t));
713     if (!Sem)
714     {
715         return (AE_NO_MEMORY);
716     }

```

```

718     if (sem_init (Sem, 0, InitialUnits) == -1)
719     {
720         AcpiOsFree (Sem);
721         return (AE_BAD_PARAMETER);
722     }
723 #endif

725     *OutHandle = (ACPI_HANDLE) Sem;
726     return (AE_OK);
727 }

730 /*****
731  *
732  * FUNCTION:      AcpiOsDeleteSemaphore
733  *
734  * PARAMETERS:    Handle              - Handle returned by AcpiOsCreateSemaphore
735  *
736  * RETURN:        Status
737  *
738  * DESCRIPTION:   Delete an OS semaphore
739  *
740  *****/
742 ACPI_STATUS
743 AcpiOsDeleteSemaphore (
744     ACPI_HANDLE        Handle)
745 {
746     sem_t              *Sem = (sem_t *) Handle;

749     if (!Sem)
750     {
751         return (AE_BAD_PARAMETER);
752     }

754     if (sem_destroy (Sem) == -1)
755     {
756         return (AE_BAD_PARAMETER);
757     }

759     return (AE_OK);
760 }

763 /*****
764  *
765  * FUNCTION:      AcpiOsWaitSemaphore
766  *
767  * PARAMETERS:    Handle              - Handle returned by AcpiOsCreateSemaphore
768  *                Units              - How many units to wait for
769  *                MsecTimeout         - How long to wait (milliseconds)
770  *
771  * RETURN:        Status
772  *
773  * DESCRIPTION:   Wait for units
774  *
775  *****/
777 ACPI_STATUS
778 AcpiOsWaitSemaphore (
779     ACPI_HANDLE        Handle,
780     UINT32            Units,
781     UINT16            MsecTimeout)
782 {

```

```

783     ACPI_STATUS      Status = AE_OK;
784     sem_t            *Sem = (sem_t *) Handle;
785 #ifndef ACPI_USE_ALTERNATE_TIMEOUT
786     struct timespec   Time;
787     int               RetVal;
788 #endif

791     if (!Sem)
792     {
793         return (AE_BAD_PARAMETER);
794     }

796     switch (MsecTimeout)
797     {
798     /*
799      * No Wait:
800      * -----
801      * A zero timeout value indicates that we shouldn't wait - just
802      * acquire the semaphore if available otherwise return AE_TIME
803      * (a.k.a. 'would block').
804      */
805     case 0:
806         if (sem_trywait(Sem) == -1)
807         {
808             Status = (AE_TIME);
809         }
810         break;

813     /* Wait Indefinitely */

815     case ACPI_WAIT_FOREVER:

817         if (sem_wait (Sem))
818         {
819             Status = (AE_TIME);
820         }
821         break;

823     /* Wait with MsecTimeout */

825     default:

827 #ifdef ACPI_USE_ALTERNATE_TIMEOUT
828     /*
829      * Alternate timeout mechanism for environments where
830      * sem_timedwait is not available or does not work properly.
831      */
832     while (MsecTimeout)
833     {
834         if (sem_trywait (Sem) == 0)
835         {
836             /* Got the semaphore */
837             return (AE_OK);
838         }

840         if (MsecTimeout >= 10)
841         {
842             MsecTimeout -= 10;
843             usleep (10 * ACPI_USEC_PER_MSEC); /* ten milliseconds */
844         }
845         else
846         {
847             MsecTimeout--;
848             usleep (ACPI_USEC_PER_MSEC); /* one millisecond */

```

```

849     }
850     }
851     Status = (AE_TIME);
852 #else
853     /*
854      * The interface to sem_timedwait is an absolute time, so we need to
855      * get the current time, then add in the millisecond Timeout value.
856      */
857     if (clock_gettime (CLOCK_REALTIME, &Time) == -1)
858     {
859         perror ("clock_gettime");
860         return (AE_TIME);
861     }

863     Time.tv_sec += (MsecTimeout / ACPI_MSEC_PER_SEC);
864     Time.tv_nsec += ((MsecTimeout % ACPI_MSEC_PER_SEC) * ACPI_NSEC_PER_MSEC)

866     /* Handle nanosecond overflow (field must be less than one second) */

868     if (Time.tv_nsec >= ACPI_NSEC_PER_SEC)
869     {
870         Time.tv_sec += (Time.tv_nsec / ACPI_NSEC_PER_SEC);
871         Time.tv_nsec = (Time.tv_nsec % ACPI_NSEC_PER_SEC);
872     }

874     while (((RetVal = sem_timedwait (Sem, &Time)) == -1) && (errno == EINTR))
875     {
876         continue;
877     }

879     if (RetVal != 0)
880     {
881         if (errno != ETIMEDOUT)
882         {
883             perror ("sem_timedwait");
884         }
885         Status = (AE_TIME);
886     }
887 #endif
888     break;
889 }

891     return (Status);
892 }

895 /*****
896  *
897  * FUNCTION:      AcpiOsSignalSemaphore
898  *
899  * PARAMETERS:    Handle          - Handle returned by AcpiOsCreateSemaphore
900  *                Units           - Number of units to send
901  *
902  * RETURN:        Status
903  *
904  * DESCRIPTION:    Send units
905  *
906  *****/

908 ACPI_STATUS
909 AcpiOsSignalSemaphore (
910     ACPI_HANDLE      Handle,
911     UINT32           Units)
912 {
913     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 {
944
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 {
996
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 {
1018
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 {
1039
1040     if (microseconds)
1041     {
1042         usleep (microseconds);
1043     }
1044 }

```

```

1047 /*****
1048 *
1049 * FUNCTION:    AcpiOsSleep
1050 *
1051 * PARAMETERS:  milliseconds          - Time to sleep
1052 *
1053 * RETURN:      Blocks until sleep is completed.
1054 *
1055 * DESCRIPTION: Sleep at millisecond granularity
1056 *
1057 *****/
1059 void
1060 AcpiOsSleep (
1061     UINT64          milliseconds)
1062 {
1064     /* Sleep for whole seconds */
1066     sleep (milliseconds / ACPI_MSEC_PER_SEC);
1068     /*
1069     * Sleep for remaining microseconds.
1070     * Arg to usleep() is in usecs and must be less than 1,000,000 (1 second).
1071     */
1072     usleep ((milliseconds % ACPI_MSEC_PER_SEC) * ACPI_USEC_PER_MSEC);
1073 }
1076 /*****
1077 *
1078 * FUNCTION:    AcpiOsGetTimer
1079 *
1080 * PARAMETERS:  None
1081 *
1082 * RETURN:      Current time in 100 nanosecond units
1083 *
1084 * DESCRIPTION: Get the current system time
1085 *
1086 *****/
1088 UINT64
1089 AcpiOsGetTimer (
1090     void)
1091 {
1092     struct timeval    time;
1095     /* This timer has sufficient resolution for user-space application code */
1097     gettimeofday (&time, NULL);
1099     /* (Seconds * 10^7 = 100ns(10^-7)) + (Microseconds(10^-6) * 10^1 = 100ns) */
1101     return (((UINT64) time.tv_sec * ACPI_100NSEC_PER_SEC) +
1102         ((UINT64) time.tv_usec * ACPI_100NSEC_PER_USEC));
1103 }
1106 /*****
1107 *
1108 * FUNCTION:    AcpiOsReadPciConfiguration
1109 *
1110 * PARAMETERS:  PciId          - Seg/Bus/Dev
1111 *              Register        - Device Register
1112 *              Value           - Buffer where value is placed

```

```

1113 *              Width           - Number of bits
1114 *
1115 * RETURN:      Status
1116 *
1117 * DESCRIPTION: Read data from PCI configuration space
1118 *
1119 *****/
1121 ACPI_STATUS
1122 AcpiOsReadPciConfiguration (
1123     ACPI_PCI_ID      *PciId,
1124     UINT32           Register,
1125     UINT64           *Value,
1126     UINT32           Width)
1127 {
1129     *Value = 0;
1130     return (AE_OK);
1131 }
1134 /*****
1135 *
1136 * FUNCTION:    AcpiOsWritePciConfiguration
1137 *
1138 * PARAMETERS:  PciId          - Seg/Bus/Dev
1139 *              Register        - Device Register
1140 *              Value           - Value to be written
1141 *              Width           - Number of bits
1142 *
1143 * RETURN:      Status.
1144 *
1145 * DESCRIPTION: Write data to PCI configuration space
1146 *
1147 *****/
1149 ACPI_STATUS
1150 AcpiOsWritePciConfiguration (
1151     ACPI_PCI_ID      *PciId,
1152     UINT32           Register,
1153     UINT64           Value,
1154     UINT32           Width)
1155 {
1157     return (AE_OK);
1158 }
1161 /*****
1162 *
1163 * FUNCTION:    AcpiOsReadPort
1164 *
1165 * PARAMETERS:  Address        - Address of I/O port/register to read
1166 *              Value           - Where value is placed
1167 *              Width           - Number of bits
1168 *
1169 * RETURN:      Value read from port
1170 *
1171 * DESCRIPTION: Read data from an I/O port or register
1172 *
1173 *****/
1175 ACPI_STATUS
1176 AcpiOsReadPort (
1177     ACPI_IO_ADDRESS  Address,
1178     UINT32           *Value,

```



```

1179     UINT32          Width)
1180 {
1181
1182     switch (Width)
1183     {
1184     case 8:
1185
1186         *Value = 0xFF;
1187         break;
1188
1189     case 16:
1190
1191         *Value = 0xFFFF;
1192         break;
1193
1194     case 32:
1195
1196         *Value = 0xFFFFFFFF;
1197         break;
1198
1199     default:
1200
1201         return (AE_BAD_PARAMETER);
1202     }
1203
1204     return (AE_OK);
1205 }
1206
1207 /*****
1208 *
1209 * FUNCTION:      AcpiOsWritePort
1210 *
1211 * PARAMETERS:    Address      - Address of I/O port/register to write
1212 *                Value        - Value to write
1213 *                Width        - Number of bits
1214 *
1215 * RETURN:        None
1216 *
1217 * DESCRIPTION:   Write data to an I/O port or register
1218 *
1219 * *****/
1220
1221
1222 ACPI_STATUS
1223 AcpiOsWritePort (
1224     ACPI_IO_ADDRESS    Address,
1225     UINT32             Value,
1226     UINT32             Width)
1227 {
1228
1229     return (AE_OK);
1230 }
1231
1232 /*****
1233 *
1234 * FUNCTION:      AcpiOsReadMemory
1235 *
1236 * PARAMETERS:    Address      - Physical Memory Address to read
1237 *                Value        - Where value is placed
1238 *                Width        - Number of bits (8,16,32, or 64)
1239 *
1240 * RETURN:        Value read from physical memory address. Always returned
1241 *                as a 64-bit integer, regardless of the read width.
1242 *
1243 * DESCRIPTION:   Read data from a physical memory address
1244 */

```

```

1245 *
1246 *****/
1247
1248 ACPI_STATUS
1249 AcpiOsReadMemory (
1250     ACPI_PHYSICAL_ADDRESS    Address,
1251     UINT64                   *Value,
1252     UINT32                   Width)
1253 {
1254
1255     switch (Width)
1256     {
1257     case 8:
1258     case 16:
1259     case 32:
1260     case 64:
1261
1262         *Value = 0;
1263         break;
1264
1265     default:
1266
1267         return (AE_BAD_PARAMETER);
1268     }
1269     return (AE_OK);
1270 }
1271
1272 /*****
1273 *
1274 * FUNCTION:      AcpiOsWriteMemory
1275 *
1276 * PARAMETERS:    Address      - Physical Memory Address to write
1277 *                Value        - Value to write
1278 *                Width        - Number of bits (8,16,32, or 64)
1279 *
1280 * RETURN:        None
1281 *
1282 * DESCRIPTION:   Write data to a physical memory address
1283 *
1284 * *****/
1285
1286
1287 ACPI_STATUS
1288 AcpiOsWriteMemory (
1289     ACPI_PHYSICAL_ADDRESS    Address,
1290     UINT64                   Value,
1291     UINT32                   Width)
1292 {
1293
1294     return (AE_OK);
1295 }
1296
1297 /*****
1298 *
1299 * FUNCTION:      AcpiOsReadable
1300 *
1301 * PARAMETERS:    Pointer      - Area to be verified
1302 *                Length       - Size of area
1303 *
1304 * RETURN:        TRUE if readable for entire length
1305 *
1306 * DESCRIPTION:   Verify that a pointer is valid for reading
1307 *
1308 * *****/
1309

```

```

1311 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 }
```

new/usr/src/common/acpica/os\_specific/service\_layers/oswindir.c

1

```
*****
7547 Fri Jan 17 07:31:13 2014
new/usr/src/common/acpica/os_specific/service_layers/oswindir.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: oswindir - Windows directory access interfaces
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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>

46 #include <stdio.h>
47 #include <stdlib.h>
48 #include <string.h>
49 #include <io.h>

51 typedef struct ExternalFindInfo
52 {
53     struct _finddata_t      DosInfo;
54     char                    *FullWildcardSpec;
55     long                    FindHandle;
56     char                    State;
57     char                    RequestedFileType;
58 }
59 } EXTERNAL_FIND_INFO;
```

new/usr/src/common/acpica/os\_specific/service\_layers/oswindir.c

2

```
62 /*****
63 *
64 * FUNCTION:      AcpiOsOpenDirectory
65 *
66 * PARAMETERS:    DirPathname      - Full pathname to the directory
67 *                WildcardSpec     - string of the form "*.c", etc.
68 *                RequestedFileType - Either a directory or normal file
69 *
70 * RETURN:        A directory "handle" to be used in subsequent search operations.
71 *                NULL returned on failure.
72 *
73 * DESCRIPTION:   Open a directory in preparation for a wildcard search
74 *
75 *****/

77 void *
78 AcpiOsOpenDirectory (
79     char                *DirPathname,
80     char                *WildcardSpec,
81     char                RequestedFileType)
82 {
83     long                FindHandle;
84     char                *FullWildcardSpec;
85     EXTERNAL_FIND_INFO *SearchInfo;

88     /* No directory path means "use current directory" - use a dot */

90     if (!DirPathname || strlen (DirPathname) == 0)
91     {
92         DirPathname = ".";
93     }

95     /* Allocate the info struct that will be returned to the caller */

97     SearchInfo = calloc (sizeof (EXTERNAL_FIND_INFO), 1);
98     if (!SearchInfo)
99     {
100         return (NULL);
101     }

103     /* Allocate space for the full wildcard path */

105     FullWildcardSpec = calloc (strlen (DirPathname) + strlen (WildcardSpec) + 2,
106                               sizeof (char));
107     if (!FullWildcardSpec)
108     {
109         printf ("Could not allocate buffer for wildcard pathname\n");
110         return (NULL);
111     }

112     /* Create the full wildcard path */

114     strcpy (FullWildcardSpec, DirPathname);
115     strcat (FullWildcardSpec, "/");
116     strcat (FullWildcardSpec, WildcardSpec);

118     /* Initialize the find functions, get first match */

120     FindHandle = _findfirst (FullWildcardSpec, &SearchInfo->DosInfo);
121     if (FindHandle == -1)
122     {
123         /* Failure means that no match was found */

125         free (FullWildcardSpec);
```

```

126     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     }

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 }

```

new/usr/src/common/acpica/os\_specific/service\_layers/oswintbl.c

1

```
*****
12518 Fri Jan 17 07:31:13 2014
new/usr/src/common/acpica/os_specific/service_layers/oswintbl.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: oswintbl - Windows OSL for obtaining ACPI tables
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acutils.h"
47 #include <stdio.h>

49 #ifdef WIN32
50 #pragma warning(disable:4115) /* warning C4115: (caused by rpcasynch.h) */
51 #include <windows.h>

53 #elif WIN64
54 #include <windowsx.h>
55 #endif

57 #define _COMPONENT          ACPI_OS_SERVICES
58 #define ACPI_MODULE_NAME    ("oswintbl")
```

new/usr/src/common/acpica/os\_specific/service\_layers/oswintbl.c

2

```
60 /* Local prototypes */

62 static char *
63 WindowsFormatException (
64     LONG                WinStatus);

66 /* Globals */

68 #define LOCAL_BUFFER_SIZE        64

70 static char                KeyBuffer[LOCAL_BUFFER_SIZE];
71 static char                ErrorBuffer[LOCAL_BUFFER_SIZE];

73 /*
74 * Tables supported in the Windows registry. SSDTs are not placed into
75 * the registry, a limitation.
76 */
77 static char                *SupportedTables[] =
78 {
79     "DSDT",
80     "RSDT",
81     "FACS",
82     "FACP"
83 };

85 /* Max index for table above */

87 #define ACPI_OS_MAX_TABLE_INDEX    3

90 /*****
91 *
92 * FUNCTION:        WindowsFormatException
93 *
94 * PARAMETERS:      WinStatus          - Status from a Windows system call
95 *
96 * RETURN:          Formatted (ascii) exception code. Front-end to Windows
97 *                  FormatMessage interface.
98 *
99 * DESCRIPTION:      Decode a windows exception
100 *
101 *****/

103 static char *
104 WindowsFormatException (
105     LONG                WinStatus)
106 {
108     ErrorBuffer[0] = 0;
109     FormatMessage (FORMAT_MESSAGE_FROM_SYSTEM, NULL, WinStatus, 0,
110                 ErrorBuffer, LOCAL_BUFFER_SIZE, NULL);

112     return (ErrorBuffer);
113 }

116 /*****
117 *
118 * FUNCTION:        AcpiOsGetTableByAddress
119 *
120 * PARAMETERS:      Address            - Physical address of the ACPI table
121 *                  Table              - Where a pointer to the table is returned
122 *
123 * RETURN:          Status; Table buffer is returned if AE_OK.
124 *                  AE_NOT_FOUND: A valid table was not found at the address
```

```

125 *
126 * DESCRIPTION: Get an ACPI table via a physical memory address.
127 *
128 * NOTE:      Cannot be implemented without a Windows device driver.
129 *
130 *****/
132 ACPI_STATUS
133 AcpiOsGetTableByAddress (
134     ACPI_PHYSICAL_ADDRESS    Address,
135     ACPI_TABLE_HEADER        **Table)
136 {
137
138     fprintf (stderr, "Get table by address is not supported on Windows\n");
139     return (AE_SUPPORT);
140 }
141
142
143 /*****
144 *
145 * FUNCTION:    AcpiOsGetTableByIndex
146 *
147 * PARAMETERS:  Index            - Which table to get
148 *              Table            - Where a pointer to the table is returned
149 *              Instance         - Where a pointer to the table instance no. is
150 *                               returned
151 *              Address          - Where the table physical address is returned
152 *
153 * RETURN:      Status; Table buffer and physical address returned if AE_OK.
154 *              AE_LIMIT: Index is beyond valid limit
155 *
156 * DESCRIPTION: Get an ACPI table via an index value (0 through n). Returns
157 *              AE_LIMIT when an invalid index is reached. Index is not
158 *              necessarily an index into the RSDT/XSDT.
159 *              Table is obtained from the Windows registry.
160 *
161 * NOTE:        Cannot get the physical address from the windows registry;
162 *              zero is returned instead.
163 *
164 *****/
166 ACPI_STATUS
167 AcpiOsGetTableByIndex (
168     UINT32                Index,
169     ACPI_TABLE_HEADER      **Table,
170     UINT32                *Instance,
171     ACPI_PHYSICAL_ADDRESS *Address)
172 {
173     ACPI_STATUS    Status;
174
175     if (Index > ACPI_OS_MAX_TABLE_INDEX)
176     {
177         return (AE_LIMIT);
178     }
179
180     Status = AcpiOsGetTableByName (SupportedTables[Index], 0, Table, Address);
181     return (Status);
182 }
183
184
185 /*****
186 *
187 * FUNCTION:    AcpiOsGetTableByName
188 *
189 * PARAMETERS:  Signature        - ACPI Signature for desired table. Must be

```

```

191 *              a null terminated 4-character string.
192 *              Instance        - For SSDTs (0...n). Use 0 otherwise.
193 *              Table           - Where a pointer to the table is returned
194 *              Address         - Where the table physical address is returned
195 *
196 * RETURN:      Status; Table buffer and physical address returned if AE_OK.
197 *              AE_LIMIT: Instance is beyond valid limit
198 *              AE_NOT_FOUND: A table with the signature was not found
199 *
200 * DESCRIPTION: Get an ACPI table via a table signature (4 ASCII characters).
201 *              Returns AE_LIMIT when an invalid instance is reached.
202 *              Table is obtained from the Windows registry.
203 *
204 * NOTE:        Assumes the input signature is uppercase.
205 *              Cannot get the physical address from the windows registry;
206 *              zero is returned instead.
207 *
208 *****/
210 ACPI_STATUS
211 AcpiOsGetTableByName (
212     char                *Signature,
213     UINT32              Instance,
214     ACPI_TABLE_HEADER    **Table,
215     ACPI_PHYSICAL_ADDRESS *Address)
216 {
217     HKEY                Handle = NULL;
218     LONG                WinStatus;
219     ULONG               Type;
220     ULONG               NameSize;
221     ULONG               DataSize;
222     HKEY                SubKey;
223     ULONG               i;
224     ACPI_TABLE_HEADER    *ReturnTable;
225
226     /*
227     * Windows has no SSDTs in the registry, so multiple instances are
228     * not supported.
229     */
230     if (Instance > 0)
231     {
232         return (AE_LIMIT);
233     }
234
235     /* Get a handle to the table key */
236
237     while (1)
238     {
239         ACPI_STRCPY (KeyBuffer, "HARDWARE\\ACPI\\");
240         if (AcpiUtSafeStrcat (KeyBuffer, sizeof (KeyBuffer), Signature))
241         {
242             return (AE_BUFFER_OVERFLOW);
243         }
244
245         WinStatus = RegOpenKeyEx (HKEY_LOCAL_MACHINE, KeyBuffer,
246                                 0L, KEY_READ, &Handle);
247
248         if (WinStatus != ERROR_SUCCESS)
249         {
250             /*
251             * Somewhere along the way, MS changed the registry entry for
252             * the FADT from
253             * HARDWARE/ACPI/FACP to
254             * HARDWARE/ACPI/FADT.
255             */

```

```

257     * 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
```

new/usr/src/common/acpica/os\_specific/service\_layers/oswinxf.c

1

```
*****
37033 Fri Jan 17 07:31:13 2014
new/usr/src/common/acpica/os_specific/service_layers/oswinxf.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: oswinxf - Windows OSL
4 *
5 *****/
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"
47 #ifdef WIN32
48 #pragma warning(disable:4115) /* warning C4115: named type definition in paren
50 #include <windows.h>
51 #include <winbase.h>
53 #elif WIN64
54 #include <windowsx.h>
55 #endif
57 #include <stdio.h>
```

new/usr/src/common/acpica/os\_specific/service\_layers/oswinxf.c

2

```
58 #include <stdlib.h>
59 #include <stdarg.h>
60 #include <process.h>
61 #include <time.h>
63 #define _COMPONENT          ACPI_OS_SERVICES
64         ACPI_MODULE_NAME    ("oswinxf")
67 FILE          *AcpiGbl_OutputFile;
68 UINT64        TimerFrequency;
69 char          TableName[ACPI_NAME_SIZE + 1];
71 #define ACPI_OS_DEBUG_TIMEOUT    30000 /* 30 seconds */
74 /* Upcalls to AcpiExec application */
76 ACPI_PHYSICAL_ADDRESS
77 AeLocalGetRootPointer (
78     void);
80 void
81 AeTableOverride (
82     ACPI_TABLE_HEADER *ExistingTable,
83     ACPI_TABLE_HEADER **NewTable);
85 /*
86 * Real semaphores are only used for a multi-threaded application
87 */
88 #ifndef ACPI_SINGLE_THREADED
90 /* Semaphore information structure */
92 typedef struct acpi_os_semaphore_info
93 {
94     UINT16        MaxUnits;
95     UINT16        CurrentUnits;
96     void          *OsHandle;
98 } ACPI_OS_SEMAPHORE_INFO;
100 /* Need enough semaphores to run the large aslts suite */
102 #define ACPI_OS_MAX_SEMAPHORES    256
104 ACPI_OS_SEMAPHORE_INFO            AcpiGbl_Semaphores[ACPI_OS_MAX_SEMAPHORES];
106 #endif /* ACPI_SINGLE_THREADED */
108 BOOLEAN                            AcpiGbl_DebugTimeout = FALSE;
110 /*****
111 *
112 * FUNCTION:      AcpiOsTerminate
113 *
114 * PARAMETERS:    None
115 *
116 * RETURN:        Status
117 *
118 * DESCRIPTION:    Nothing to do for windows
119 *
120 *****/
122 ACPI_STATUS
123 AcpiOsTerminate (
```

```

124     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     #ifndef 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 #ifndef ACPI_EXEC_APP

```

```

256  /* Call back up to AcpiExec */
258  AeTableOverride (ExistingTable, NewTable);
259  #endif

262  #ifdef ACPI_AS_L_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
389 *
390 *****/

392 BOOLEAN
393 AcpiOsWrittable (
394     void                *Pointer,
395     ACPI_SIZE           Length)
396 {
397
398     return ((BOOLEAN) !IsBadWritePtr (Pointer, Length));
399 }

402 /*****
403 *
404 * FUNCTION:    AcpiOsRedirectOutput
405 *
406 * PARAMETERS: Destination        - An open file handle/pointer
407 *
408 * RETURN:      None
409 *
410 * DESCRIPTION: Causes redirect of AcpiOsPrintf and AcpiOsVprintf
411 *
412 *****/

414 void
415 AcpiOsRedirectOutput (
416     void                *Destination)
417 {
418
419     AcpiGbl_OutputFile = Destination;
420 }

423 /*****
424 *
425 * FUNCTION:    AcpiOsPrintf
426 *
427 * PARAMETERS: Fmt, ...          - Standard printf format
428 *
429 * RETURN:      None
430 *
431 * DESCRIPTION: Formatted output
432 *
433 *****/

435 void ACPI_INTERNAL_VAR_XFACE
436 AcpiOsPrintf (
437     const char          *Fmt,
438     ...)
439 {
440     va_list              Args;
441     UINT8                Flags;

442     Flags = AcpiGbl_DbOutputFlags;
443     if (Flags & ACPI_DB_REDIRECTABLE_OUTPUT)
444     {
445         /* Output is directable to either a file (if open) or the console */

446         if (AcpiGbl_DebugFile)
447         {
448             /* Output file is open, send the output there */

449             va_start (Args, Fmt);

```

```

454     vfprintf (AcpiGbl_DebugFile, Fmt, Args);
455     va_end (Args);
456 }
457 else
458 {
459     /* No redirection, send output to console (once only!) */

461     Flags |= ACPI_DB_CONSOLE_OUTPUT;
462 }
463 }

465 if (Flags & ACPI_DB_CONSOLE_OUTPUT)
466 {
467     va_start (Args, Fmt);
468     vfprintf (AcpiGbl_OutputFile, Fmt, Args);
469     va_end (Args);
470 }

472 return;
473 }

476 /*****
477 *
478 * FUNCTION:    AcpiOsVprintf
479 *
480 * PARAMETERS: Fmt                - Standard printf format
481 *              Args              - Argument list
482 *
483 * RETURN:      None
484 *
485 * DESCRIPTION: Formatted output with argument list pointer
486 *
487 *****/

489 void
490 AcpiOsVprintf (
491     const char          *Fmt,
492     va_list              Args)
493 {
494     INT32                Count = 0;
495     UINT8                Flags;

496     Flags = AcpiGbl_DbOutputFlags;
497     if (Flags & ACPI_DB_REDIRECTABLE_OUTPUT)
498     {
499         /* Output is directable to either a file (if open) or the console */

501         if (AcpiGbl_DebugFile)
502         {
503             /* Output file is open, send the output there */

504             Count = vfprintf (AcpiGbl_DebugFile, Fmt, Args);
505         }
506         else
507         {
508             /* No redirection, send output to console (once only!) */

509             Flags |= ACPI_DB_CONSOLE_OUTPUT;
510         }
511     }

513     if (Flags & ACPI_DB_CONSOLE_OUTPUT)
514     {
515         Count = vfprintf (AcpiGbl_OutputFile, Fmt, Args);

```

```

520     }
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);
653 }

656 /*****
657  *
658  * FUNCTION:      AcpiOsFree
659  *
660  * PARAMETERS:    Mem                - Pointer to previously allocated memory
661  *
662  * RETURN:        None.
663  *
664  * DESCRIPTION:   Free memory allocated via AcpiOsAllocate
665  *
666  *****/

668 void
669 AcpiOsFree (
670     void                *Mem)
671 {
672
673     free (Mem);
674 }

677 #ifdef ACPI_SINGLE_THREADED
678 /*****
679  *
680  * FUNCTION:      Semaphore stub functions
681  *
682  * DESCRIPTION:   Stub functions used for single-thread applications that do
683  *               not require semaphore synchronization. Full implementations
684  *               of these functions appear after the stubs.
685  *
686  *****/

688 ACPI_STATUS
689 AcpiOsCreateSemaphore (
690     UINT32              MaxUnits,
691     UINT32              InitialUnits,
692     ACPI_HANDLE         *OutHandle)
693 {
694     *OutHandle = (ACPI_HANDLE) 1;
695     return (AE_OK);
696 }

698 ACPI_STATUS
699 AcpiOsDeleteSemaphore (
700     ACPI_HANDLE         Handle)
701 {
702     return (AE_OK);
703 }

705 ACPI_STATUS
706 AcpiOsWaitSemaphore (
707     ACPI_HANDLE         Handle,
708     UINT32              Units,
709     UINT16              Timeout)
710 {
711     return (AE_OK);
712 }

714 ACPI_STATUS
715 AcpiOsSignalSemaphore (
716     ACPI_HANDLE         Handle,
717     UINT32              Units)

```

```

718 {
719     return (AE_OK);
720 }

722 #else
723 /*****
724  *
725  * FUNCTION:      AcpiOsCreateSemaphore
726  *
727  * PARAMETERS:    MaxUnits            - Maximum units that can be sent
728  *               InitialUnits        - Units to be assigned to the new semaphore
729  *               OutHandle            - Where a handle will be returned
730  *
731  * RETURN:        Status
732  *
733  * DESCRIPTION:   Create an OS semaphore
734  *
735  *****/

737 ACPI_STATUS
738 AcpiOsCreateSemaphore (
739     UINT32              MaxUnits,
740     UINT32              InitialUnits,
741     ACPI_SEMAPHORE      *OutHandle)
742 {
743     void                *Mutex;
744     UINT32              i;

746     ACPI_FUNCTION_NAME (OsCreateSemaphore);

749     if (MaxUnits == ACPI_UINT32_MAX)
750     {
751         MaxUnits = 255;
752     }

754     if (InitialUnits == ACPI_UINT32_MAX)
755     {
756         InitialUnits = MaxUnits;
757     }

759     if (InitialUnits > MaxUnits)
760     {
761         return (AE_BAD_PARAMETER);
762     }

764     /* Find an empty slot */

766     for (i = 0; i < ACPI_OS_MAX_SEMAPHORES; i++)
767     {
768         if (!AcpiGbl_Semaphores[i].OsHandle)
769         {
770             break;
771         }
772     }
773     if (i >= ACPI_OS_MAX_SEMAPHORES)
774     {
775         ACPI_EXCEPTION ((AE_INFO, AE_LIMIT,
776             "Reached max semaphores (%u), could not create", ACPI_OS_MAX_SEMAPHO
777         return (AE_LIMIT);
778     }

780     /* Create an OS semaphore */

782     Mutex = CreateSemaphore (NULL, InitialUnits, MaxUnits, NULL);
783     if (!Mutex)

```

```

784 {
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=%u",
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
917 *
918 * PARAMETERS:    Handle          - Handle returned by AcpiOsCreateSemaphore
919 *                Units           - Number of units to send
920 *
921 * RETURN:        Status
922 *
923 * DESCRIPTION:    Send units
924 *
925 *****/

927 ACPI_STATUS
928 AcpiOsSignalSemaphore (
929     ACPI_SEMAPHORE    Handle,
930     UINT32             Units)
931 {
932     UINT32             Index = (UINT32) Handle;

935     ACPI_FUNCTION_ENTRY ();

938     if (Index >= ACPI_OS_MAX_SEMAPHORES)
939     {
940         printf ("SignalSemaphore: Index/Handle out of range: %2.2X\n", Index);
941         return (AE_BAD_PARAMETER);
942     }

944     if (!AcpiGbl_Semaphores[Index].OsHandle)
945     {
946         printf ("SignalSemaphore: Null OS handle, Index %2.2X\n", Index);
947         return (AE_BAD_PARAMETER);
948     }

950     if (Units > 1)
951     {
952         printf ("SignalSemaphore: Attempt to signal %u units, Index %2.2X\n", Units, Index);
953         return (AE_NOT_IMPLEMENTED);
954     }

956     if ((AcpiGbl_Semaphores[Index].CurrentUnits + 1) >
957         AcpiGbl_Semaphores[Index].MaxUnits)
958     {
959         ACPI_ERROR ((AE_INFO,
960             "Oversignalled semaphore[%u]! Current %u Max %u",
961             Index, AcpiGbl_Semaphores[Index].CurrentUnits,
962             AcpiGbl_Semaphores[Index].MaxUnits));

964         return (AE_LIMIT);
965     }

967     AcpiGbl_Semaphores[Index].CurrentUnits++;
968     ReleaseSemaphore (AcpiGbl_Semaphores[Index].OsHandle, Units, NULL);

970     return (AE_OK);
971 }

973 #endif /* ACPI_SINGLE_THREADED */

976 /*****
977 *
978 * FUNCTION:      Spinlock interfaces
979 *
980 * DESCRIPTION:    Map these interfaces to semaphore interfaces
981 *

```

```

982 *****/

984 ACPI_STATUS
985 AcpiOsCreateLock (
986     ACPI_SPINLOCK      *OutHandle)
987 {
988     return (AcpiOsCreateSemaphore (1, 1, OutHandle));
989 }

991 void
992 AcpiOsDeleteLock (
993     ACPI_SPINLOCK      Handle)
994 {
995     AcpiOsDeleteSemaphore (Handle);
996 }

998 ACPI_CPU_FLAGS
999 AcpiOsAcquireLock (
1000     ACPI_SPINLOCK      Handle)
1001 {
1002     AcpiOsWaitSemaphore (Handle, 1, 0xFFFF);
1003     return (0);
1004 }

1006 void
1007 AcpiOsReleaseLock (
1008     ACPI_SPINLOCK      Handle,
1009     ACPI_CPU_FLAGS      Flags)
1010 {
1011     AcpiOsSignalSemaphore (Handle, 1);
1012 }

1015 #if ACPI_FUTURE_IMPLEMENTATION

1017 /* Mutex interfaces, just implement with a semaphore */

1019 ACPI_STATUS
1020 AcpiOsCreateMutex (
1021     ACPI_MUTEX          *OutHandle)
1022 {
1023     return (AcpiOsCreateSemaphore (1, 1, OutHandle));
1024 }

1026 void
1027 AcpiOsDeleteMutex (
1028     ACPI_MUTEX          Handle)
1029 {
1030     AcpiOsDeleteSemaphore (Handle);
1031 }

1033 ACPI_STATUS
1034 AcpiOsAcquireMutex (
1035     ACPI_MUTEX          Handle,
1036     UINT16               Timeout)
1037 {
1038     AcpiOsWaitSemaphore (Handle, 1, Timeout);
1039     return (0);
1040 }

1042 void
1043 AcpiOsReleaseMutex (
1044     ACPI_MUTEX          Handle)
1045 {
1046     AcpiOsSignalSemaphore (Handle, 1);
1047 }

```

```

1048 #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 {
1072
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 {
1094
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 {
1115     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 {
1137
1138     /* Add 10ms to account for clock tick granularity */
1139
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 {
1167
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
1181 *
1182 * RETURN:      Status
1183 *
1184 * DESCRIPTION: Write data to PCI configuration space
1185 *
1186 *****/
1188 ACPI_STATUS
1189 AcpiOsWritePciConfiguration (
1190     ACPI_PCI_ID      *PciId,
1191     UINT32            Register,
1192     UINT64            Value,
1193     UINT32            Width)
1194 {
1196     return (AE_OK);
1197 }

1200 /*****
1201 *
1202 * FUNCTION:      AcpiOsReadPort
1203 *
1204 * PARAMETERS:    Address      - Address of I/O port/register to read
1205 *                Value        - Where value is placed
1206 *                Width        - Number of bits
1207 *
1208 * RETURN:        Value read from port
1209 *
1210 * DESCRIPTION:   Read data from an I/O port or register
1211 *
1212 *****/
1214 ACPI_STATUS
1215 AcpiOsReadPort (
1216     ACPI_IO_ADDRESS    Address,
1217     UINT32             *Value,
1218     UINT32             Width)
1219 {
1220     ACPI_FUNCTION_NAME (OsReadPort);

1223     switch (Width)
1224     {
1225     case 8:

1227         *Value = 0xFF;
1228         break;

1230     case 16:

1232         *Value = 0xFFFF;
1233         break;

1235     case 32:

1237         *Value = 0xFFFFFFFF;
1238         break;

1240     default:

1242         ACPI_ERROR ((AE_INFO, "Bad width parameter: %X", Width));
1243         return (AE_BAD_PARAMETER);
1244     }

```

```

1246     return (AE_OK);
1247 }

1250 /*****
1251 *
1252 * FUNCTION:      AcpiOsWritePort
1253 *
1254 * PARAMETERS:    Address      - Address of I/O port/register to write
1255 *                Value        - Value to write
1256 *                Width        - Number of bits
1257 *
1258 * RETURN:        None
1259 *
1260 * DESCRIPTION:   Write data to an I/O port or register
1261 *
1262 *****/
1264 ACPI_STATUS
1265 AcpiOsWritePort (
1266     ACPI_IO_ADDRESS    Address,
1267     UINT32             Value,
1268     UINT32             Width)
1269 {
1270     ACPI_FUNCTION_NAME (OsWritePort);

1273     if ((Width == 8) || (Width == 16) || (Width == 32))
1274     {
1275         return (AE_OK);
1276     }

1278     ACPI_ERROR ((AE_INFO, "Bad width parameter: %X", Width));
1279     return (AE_BAD_PARAMETER);
1280 }

1283 /*****
1284 *
1285 * FUNCTION:      AcpiOsReadMemory
1286 *
1287 * PARAMETERS:    Address      - Physical Memory Address to read
1288 *                Value        - Where value is placed
1289 *                Width        - Number of bits (8,16,32, or 64)
1290 *
1291 * RETURN:        Value read from physical memory address. Always returned
1292 *                as a 64-bit integer, regardless of the read width.
1293 *
1294 * DESCRIPTION:   Read data from a physical memory address
1295 *
1296 *****/
1298 ACPI_STATUS
1299 AcpiOsReadMemory (
1300     ACPI_PHYSICAL_ADDRESS Address,
1301     UINT64                *Value,
1302     UINT32                Width)
1303 {
1305     switch (Width)
1306     {
1307     case 8:
1308     case 16:
1309     case 32:
1310     case 64:

```

```

1312     *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;

```

```

1445     NewObject = malloc (((ACPI_MEMORY_LIST *) Cache)->ObjectSize);
1446     memset (NewObject, 0, ((ACPI_MEMORY_LIST *) Cache)->ObjectSize);
1448     return (NewObject);
1449 }

1451 ACPI_STATUS
1452 AcpiOsReleaseObject (
1453     ACPI_CACHE_T          *Cache,
1454     void                  *Object)
1455 {
1456     free (Object);
1457     return (AE_OK);
1458 }

1460 #endif /* ACPI_USE_LOCAL_CACHE */

1463 /* Optional multi-thread support */

1465 #ifndef ACPI_SINGLE_THREADED
1466 /*****
1467  *
1468  * FUNCTION:      AcpiOsGetThreadId
1469  *
1470  * PARAMETERS:    None
1471  *
1472  * RETURN:        Id of the running thread
1473  *
1474  * DESCRIPTION:   Get the Id of the current (running) thread
1475  *
1476  *****/
1478 ACPI_THREAD_ID
1479 AcpiOsGetThreadId (
1480     void)
1481 {
1482     DWORD          ThreadId;

1484     /* Ensure ID is never 0 */

1486     ThreadId = GetCurrentThreadId ();
1487     return ((ACPI_THREAD_ID) (ThreadId + 1));
1488 }

1491 /*****
1492  *
1493  * FUNCTION:      AcpiOsExecute
1494  *
1495  * PARAMETERS:    Type                - Type of execution
1496  *                Function            - Address of the function to execute
1497  *                Context              - Passed as a parameter to the function
1498  *
1499  * RETURN:        Status
1500  *
1501  * DESCRIPTION:   Execute a new thread
1502  *
1503  *****/
1505 ACPI_STATUS
1506 AcpiOsExecute (
1507     ACPI_EXECUTE_TYPE    Type,
1508     ACPI_OSD_EXEC_CALLBACK Function,
1509     void                  *Context)

```

```

1510 {
1512     _beginthread (Function, (unsigned) 0, Context);
1513     return (0);
1514 }

1516 #endif /* ACPI_SINGLE_THREADED */

1519 /*****
1520  *
1521  * FUNCTION:      AcpiOsWaitEventsComplete
1522  *
1523  * PARAMETERS:    None
1524  *
1525  * RETURN:        None
1526  *
1527  * DESCRIPTION:   Wait for all asynchronous events to complete. This
1528  *                implementation does nothing.
1529  *
1530  *****/
1532 void
1533 AcpiOsWaitEventsComplete (
1534     void)
1535 {
1536     return;
1537 }

```

new/usr/src/common/acpica/tools/acpibin/abcompare.c

1

```
*****
21798 Fri Jan 17 07:31:13 2014
new/usr/src/common/acpica/tools/acpibin/abcompare.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: abcompare - compare AML files
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acpibin.h"

47 FILE *File1;
48 FILE *File2;
49 ACPI_TABLE_HEADER Header1;
50 ACPI_TABLE_HEADER Header2;

52 #define BUFFER_SIZE 256
53 char Buffer[BUFFER_SIZE];

56 /* Local prototypes */

58 static BOOLEAN
59 AbValidateHeader (
60     ACPI_TABLE_HEADER *Header);
```

new/usr/src/common/acpica/tools/acpibin/abcompare.c

2

```
62 static UINT8
63 AcpiTbSumTable (
64     void *Buffer,
65     UINT32 Length);

67 static char *
68 AbGetFile (
69     char *Filename,
70     UINT32 *FileSize);

72 static UINT32
73 AbGetFileSize (
74     FILE *File);

76 static void
77 AbPrintHeaderInfo (
78     ACPI_TABLE_HEADER *Header);

80 static void
81 AbPrintHeadersInfo (
82     ACPI_TABLE_HEADER *Header,
83     ACPI_TABLE_HEADER *Header2);

85 ACPI_PHYSICAL_ADDRESS
86 AeLocalGetRootPointer (
87     void);

90 /*****
91 *
92 * FUNCTION: UtHexCharToValue
93 *
94 * PARAMETERS: HexChar - Hex character in Ascii
95 *
96 * RETURN: The binary value of the hex character
97 *
98 * DESCRIPTION: Perform ascii-to-hex translation
99 *
100 *****/

102 static UINT8
103 UtHexCharToValue (
104     int HexChar,
105     UINT8 *OutBinary)
106 {
108     if (HexChar >= 0x30 && HexChar <= 0x39)
109     {
110         *OutBinary = (UINT8) (HexChar - 0x30);
111         return (1);
112     }
114     else if (HexChar >= 0x41 && HexChar <= 0x46)
115     {
116         *OutBinary = (UINT8) (HexChar - 0x37);
117         return (1);
118     }
120     else if (HexChar >= 0x61 && HexChar <= 0x66)
121     {
122         *OutBinary = (UINT8) (HexChar - 0x57);
123         return (1);
124     }
125     return (0);
126 }
```

```

128 static UINT8
129 AbHexByteToBinary (
130     char          *HexString,
131     char          *OutBinary)
132 {
133     UINT8         Local1;
134     UINT8         Local2;
135
136     if (!UtHexCharToValue (HexString[0], &Local1))
137     {
138         return (0);
139     }
140     if (!UtHexCharToValue (HexString[1], &Local2))
141     {
142         return (0);
143     }
144
145     *OutBinary = (UINT8) ((Local1 << 4) | Local2);
146     return (2);
147
148 }
149
150 /*****
151 *
152 * FUNCTION:      AbValidateHeader
153 *
154 * DESCRIPTION: Check for valid ACPI table header
155 *
156 * *****/
157
158 static BOOLEAN
159 AbValidateHeader (
160     ACPI_TABLE_HEADER *Header)
161 {
162     if (!AcpiUtValidAcpiName (Header->Signature))
163     {
164         printf ("Header signature is invalid\n");
165         return (FALSE);
166     }
167
168     return (TRUE);
169
170 }
171
172 /*****
173 *
174 * FUNCTION:      AcpiTbSumTable
175 *
176 * PARAMETERS: Buffer          - Buffer to checksum
177 *              Length        - Size of the buffer
178 *
179 * RETURNS:      8 bit checksum of buffer
180 *
181 * DESCRIPTION: Computes an 8 bit checksum of the buffer(length) and returns it.
182 *
183 * *****/
184
185 static UINT8
186 AcpiTbSumTable (
187     void          *Buffer,
188     UINT32        Length)
189 {

```

```

193     const UINT8    *limit;
194     const UINT8    *rover;
195     UINT8          sum = 0;
196
197     if (Buffer && Length)
198     {
199         /* Buffer and Length are valid */
200
201         limit = (UINT8 *) Buffer + Length;
202
203         for (rover = Buffer; rover < limit; rover++)
204         {
205             sum = (UINT8) (sum + *rover);
206         }
207     }
208     return (sum);
209 }
210
211 /*****
212 *
213 * FUNCTION:      AbPrintHeaderInfo
214 *
215 * PARAMETERS: Header          - An ACPI table header
216 *
217 * RETURNS:      None.
218 *
219 * DESCRIPTION: Format and display header contents.
220 *
221 * *****/
222
223 static void
224 AbPrintHeaderInfo (
225     ACPI_TABLE_HEADER *Header)
226 {
227     /* Display header information */
228
229     printf ("Signature      : %4.4s\n", Header->Signature);
230     printf ("Length          : %8.8X\n", Header->Length);
231     printf ("Revision         : %2.2X\n", Header->Revision);
232     printf ("Checksum          : %2.2X\n", Header->Checksum);
233     printf ("OEM ID            : %6.6s\n", Header->OemId);
234     printf ("OEM Table ID      : %8.8s\n", Header->OemTableId);
235     printf ("OEM Revision      : %8.8X\n", Header->OemRevision);
236     printf ("ASL Compiler ID   : %4.4s\n", Header->AslCompilerId);
237     printf ("Compiler Revision : %8.8X\n", Header->AslCompilerRevision);
238     printf ("\n");
239
240 static void
241 AbPrintHeadersInfo (
242     ACPI_TABLE_HEADER *Header,
243     ACPI_TABLE_HEADER *Header2)
244 {
245     /* Display header information for both headers */
246
247     printf ("Signature      %8.4s : %4.4s\n", Header->Signature, Header2->Signature);
248     printf ("Length          %8.8X : %8.8X\n", Header->Length, Header2->Length);
249     printf ("Revision         %8.2X : %2.2X\n", Header->Revision, Header2->Revision);
250     printf ("Checksum          %8.2X : %2.2X\n", Header->Checksum, Header2->Checksum);
251     printf ("OEM ID            %8.6s : %6.6s\n", Header->OemId, Header2->OemId);
252     printf ("OEM Table ID      %8.8s : %8.8s\n", Header->OemTableId, Header2->OemTableId);
253     printf ("OEM Revision      %8.8X : %8.8X\n", Header->OemRevision, Header2->OemRevision);

```

```

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                *FilePath)
276 {
277     UINT32              Actual;

280     File1 = fopen (FilePath, "rb");
281     if (!File1)
282     {
283         printf ("Could not open file %s\n", FilePath);
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", FilePath);
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                *FilePath)
314 {
315     UINT32              Actual;
316     ACPI_TABLE_HEADER  *Table;
317     UINT8               Checksum;

320     File1 = fopen (FilePath, "rb");
321     if (!File1)
322     {
323         printf ("Could not open file %s\n", FilePath);
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", FilePath);
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", 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 (FilePath, "r+b");
380     if (!File1)
381     {
382         printf ("Could not open file %s for writing\n", FilePath);
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",
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 }
```

new/usr/src/common/acpica/tools/acpibin/abmain.c

1

```
*****
5650 Fri Jan 17 07:31:13 2014
new/usr/src/common/acpica/tools/acpibin/abmain.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: abmain - Main module for the acpi binary utility
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 _DECLARE_GLOBALS
46 #include "acpibin.h"
47 #include "acapps.h"

49 /* Local prototypes */

51 static void
52 AbDisplayUsage (
53     UINT8                OptionCount);

56 #define AB_UTILITY_NAME        "ACPI Binary Table Dump Utility"
57 #define AB_SUPPORTED_OPTIONS   "c:d:e:h:s:tv"

60 /*****
```

new/usr/src/common/acpica/tools/acpibin/abmain.c

2

```
61 *
62 * FUNCTION:    AbDisplayUsage
63 *
64 * DESCRIPTION: Usage message
65 *
66 *****/

68 static void
69 AbDisplayUsage (
70     UINT8                OptionCount)
71 {
72     if (OptionCount)
73     {
74         printf ("Option requires %u arguments\n\n", OptionCount);
75     }
76
77     ACPI_USAGE_HEADER ("acpibin [options]");
78
79     ACPI_OPTION ("-c <File1><File2>", "Compare two binary AML files");
80     ACPI_OPTION ("-d <In><Out>", "Dump AML binary to text file");
81     ACPI_OPTION ("-e <Sig><In><Out>", "Extract binary AML table from AcpiD");
82     ACPI_OPTION ("-h <File>", "Display table header for binary AML");
83     ACPI_OPTION ("-s <File>", "Update checksum for binary AML file");
84     ACPI_OPTION ("-t", "Terse mode");
85     ACPI_OPTION ("-v", "Display version information");
86
87 }

90 /*****
91 *
92 * FUNCTION:    main
93 *
94 * DESCRIPTION: C main function
95 *
96 *****/

98 int ACPI_SYSTEM_XFACE
99 main (
100     int                argc,
101     char                *argv[])
102 {
103     int                j;
104     int                Status = AE_OK;

107     ACPI_DEBUG_INITIALIZE (); /* For debug version only */

109     AcpiGbl_DebugFile = NULL;
110     AcpiGbl_DbOutputFlags = DB_CONSOLE_OUTPUT;

112     AcpiOsInitialize ();
113     printf (ACPI_COMMON_SIGNON (AB_UTILITY_NAME));

115     if (argc < 2)
116     {
117         AbDisplayUsage (0);
118         return (0);
119     }

121     /* Command line options */

123     while ((j = AcpiGetopt (argc, argv, AB_SUPPORTED_OPTIONS)) != EOF) switch(j)
124     {
125         case 'c':    /* Compare Files */
```

```
127     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 }
```

new/usr/src/common/acpica/tools/acpibin/acpibin.h

1

```
*****
3116 Fri Jan 17 07:31:14 2014
new/usr/src/common/acpica/tools/acpibin/acpibin.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: acpibinh - Include file for AcpiBin utility
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"

47 #include <stdio.h>
48 #include <fcntl.h>
49 #include <errno.h>

51 #define DB_CONSOLE_OUTPUT 0x02
52 #define ACPI_DB_REDIRECTABLE_OUTPUT 0x01

54 /*
55 * Global variables. Defined in main.c only, externed in all other files
56 */
57 #ifndef _DECLARE_GLOBALS
58 #define EXTERN
59 #define INIT_GLOBAL(a,b) a=b
```

new/usr/src/common/acpica/tools/acpibin/acpibin.h

2

```
60 #else
61 #define EXTERN extern
62 #define INIT_GLOBAL(a,b) a
63 #endif

66 /* Globals */

68 EXTERN BOOLEAN INIT_GLOBAL (Gbl_TerseMode, FALSE);

71 /* Prototypes */

73 int
74 AbCompareAmlFiles (
75 char *File1Path,
76 char *File2Path);

78 int
79 AbExtractAmlFile (
80 char *TableSig,
81 char *File1Path,
82 char *File2Path);

84 int
85 AbDumpAmlFile (
86 char *File1Path,
87 char *File2Path);

89 void
90 AbComputeChecksum (
91 char *File1Path);

93 void
94 AbDisplayHeader (
95 char *File1Path);
```

new/usr/src/common/acpica/tools/acpidump/acpidump.h

1

```
*****
4963 Fri Jan 17 07:31:14 2014
new/usr/src/common/acpica/tools/acpidump/acpidump.h
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: acpidump.h - Include file for AcpiDump utility
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "acpi.h"
45 #include "accommon.h"
46 #include "actables.h"
47
48 #include <stdio.h>
49 #include <fcntl.h>
50 #include <errno.h>
51 #include <sys/stat.h>
52
53 /*
54 * Global variables. Defined in main.c only, externed in all other files
55 */
56 #ifndef _DECLARE_GLOBALS
57 #define EXTERN
58 #define INIT_GLOBAL(a,b)      a=b
59 #else
60 #define EXTERN      extern
```

new/usr/src/common/acpica/tools/acpidump/acpidump.h

2

```
61 #define INIT_GLOBAL(a,b)      a
62 #endif
63
64
65 /* Globals */
66
67 EXTERN BOOLEAN      INIT_GLOBAL (Gbl_SummaryMode, FALSE);
68 EXTERN BOOLEAN      INIT_GLOBAL (Gbl_VerboseMode, FALSE);
69 EXTERN BOOLEAN      INIT_GLOBAL (Gbl_BinaryMode, FALSE);
70 EXTERN BOOLEAN      INIT_GLOBAL (Gbl_DumpCustomizedTables, FALSE);
71 EXTERN FILE         INIT_GLOBAL (*Gbl_OutputFile, NULL);
72 EXTERN char          INIT_GLOBAL (*Gbl_OutputFilename, NULL);
73 EXTERN UINT64        INIT_GLOBAL (Gbl_RsdpBase, 0);
74
75 /* Globals required for use with ACPICA modules */
76
77 #ifdef _DECLARE_GLOBALS
78 UINT8               AcpiGbl_EnableInterpreterSlack = FALSE;
79 UINT8               AcpiGbl_IntegerByteWidth = 8;
80 UINT32              AcpiDbgLevel = 0;
81 UINT32              AcpiDbgLayer = 0;
82 #endif
83
84 /* Action table used to defer requested options */
85
86 typedef struct ap_dump_action
87 {
88     char              *Argument;
89     UINT32            ToBeDone;
90 } AP_DUMP_ACTION;
91
92
93 #define AP_MAX_ACTIONS      32
94
95 #define AP_DUMP_ALL_TABLES      0
96 #define AP_DUMP_TABLE_BY_ADDRESS      1
97 #define AP_DUMP_TABLE_BY_NAME      2
98 #define AP_DUMP_TABLE_BY_FILE      3
99
100 #define AP_MAX_ACPI_FILES      256 /* Prevent infinite loops */
101
102 /* Minimum FADT sizes for various table addresses */
103
104 #define MIN_FADT_FOR_DSDT      (ACPI_FADT_OFFSET (Dsdt) + sizeof (UINT32))
105 #define MIN_FADT_FOR_FACS      (ACPI_FADT_OFFSET (Facs) + sizeof (UINT32))
106 #define MIN_FADT_FOR_XDSMT      (ACPI_FADT_OFFSET (XDsdT) + sizeof (UINT64))
107 #define MIN_FADT_FOR_XFACS      (ACPI_FADT_OFFSET (XFacs) + sizeof (UINT64))
108
109
110 /*
111 * apdump - Table get/dump routines
112 */
113 int
114 ApDumpTableFromFile (
115     char              *Pathname);
116
117 int
118 ApDumpTableByName (
119     char              *Signature);
120
121 int
122 ApDumpTableByAddress (
123     char              *AsciiAddress);
124
125 int
126 ApDumpAllTables (
```



```
127     void);

129     BOOLEAN
130     ApIsValidHeader (
131         ACPI_TABLE_HEADER    *Table);

133     BOOLEAN
134     ApIsValidChecksum (
135         ACPI_TABLE_HEADER    *Table);

137     UINT32
138     ApGetTableLength (
139         ACPI_TABLE_HEADER    *Table);

142     /*
143     * apfiles - File I/O utilities
144     */
145     UINT32
146     ApGetFileSize (
147         FILE                  *File);

149     int
150     ApOpenOutputFile (
151         char                  *Pathname);

153     int
154     ApWriteToBinaryFile (
155         ACPI_TABLE_HEADER    *Table,
156         UINT32                Instance);

158     ACPI_TABLE_HEADER *
159     ApGetTableFromFile (
160         char                  *Pathname,
161         UINT32                *FileSize);
```

new/usr/src/common/acpica/tools/acpidump/apdump.c

1

```
*****
13864 Fri Jan 17 07:31:14 2014
new/usr/src/common/acpica/tools/acpidump/apdump.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: apdump - Dump routines for ACPI tables (acpidump)
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "acpidump.h"
45
46
47 /* Local prototypes */
48
49 static int
50 ApDumpTableBuffer (
51     ACPI_TABLE_HEADER *Table,
52     UINT32 Instance,
53     ACPI_PHYSICAL_ADDRESS Address);
54
55
56 /*****
57 *
58 * FUNCTION: ApIsValidHeader
59 *
60 *****/
```

new/usr/src/common/acpica/tools/acpidump/apdump.c

2

```
60 * PARAMETERS: Table - Pointer to table to be validated
61 *
62 * RETURN: TRUE if the header appears to be valid. FALSE otherwise
63 *
64 * DESCRIPTION: Check for a valid ACPI table header
65 *
66 *****/
67
68 BOOLEAN
69 ApIsValidHeader (
70     ACPI_TABLE_HEADER *Table)
71 {
72     if (!ACPI_VALIDATE_RSDP_SIG (Table->Signature))
73     {
74         /* Make sure signature is all ASCII and a valid ACPI name */
75
76         if (!AcpiUtValidAcpiName (Table->Signature))
77         {
78             fprintf (stderr, "Table signature (0x%8.8X) is invalid\n",
79                 *(UINT32 *) Table->Signature);
80             return (FALSE);
81         }
82
83         /* Check for minimum table length */
84
85         if (Table->Length < sizeof (ACPI_TABLE_HEADER))
86         {
87             fprintf (stderr, "Table length (0x%8.8X) is invalid\n",
88                 Table->Length);
89             return (FALSE);
90         }
91     }
92
93     return (TRUE);
94 }
95
96
97 /*****
98 *
99 * FUNCTION: ApIsValidChecksum
100 *
101 * PARAMETERS: Table - Pointer to table to be validated
102 *
103 * RETURN: TRUE if the checksum appears to be valid. FALSE otherwise
104 *
105 * DESCRIPTION: Check for a valid ACPI table checksum
106 *
107 *****/
108
109 BOOLEAN
110 ApIsValidChecksum (
111     ACPI_TABLE_HEADER *Table)
112 {
113     ACPI_STATUS Status;
114     ACPI_TABLE_RSDP *Rsdp;
115
116
117     if (ACPI_VALIDATE_RSDP_SIG (Table->Signature))
118     {
119         /*
120          * Checksum for RSDP.
121          * Note: Other checksums are computed during the table dump.
122          */
123
124         Rsdp = ACPI_CAST_PTR (ACPI_TABLE_RSDP, Table);
125         Status = AcpiTbValidateRsdp (Rsdp);
```

```

126     }
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 */
261
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 */
268
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      }
286
287      if (ApDumpTableBuffer (Table, Instance, Address))
288      {
289          return (-1);
290      }
291      free (Table);
292  }
293
294  /* Something seriously bad happened if the loop terminates here */
295
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     }
332
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     }
342
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     }
380
381     /* Table signatures are expected to be uppercase */
382
383     strcpy (LocalSignature, Signature);
384     AcpiUtStrupr (LocalSignature);
385
386     /* To be friendly, handle tables whose signatures do not match the name */
387
388     if (ACPI_COMPARE_NAME (LocalSignature, "FADT"))
389     {

```

```

390     strcpy (LocalSignature, ACPI_SIG_FADT);
391 }
392 else if (ACPI_COMPARE_NAME (LocalSignature, "MADT"))
393 {
394     strcpy (LocalSignature, ACPI_SIG_MADT);
395 }
396
397 /* Dump all instances of this signature (to handle multiple SSDTs) */
398
399 for (Instance = 0; Instance < AP_MAX_ACPI_FILES; Instance++)
400 {
401     Status = AcpiOsGetTableByName (LocalSignature, Instance,
402                                   &Table, &Address);
403     if (ACPI_FAILURE (Status))
404     {
405         /* AE_LIMIT means that no more tables are available */
406
407         if (Status == AE_LIMIT)
408         {
409             return (0);
410         }
411
412         fprintf (stderr,
413                 "Could not get ACPI table with signature [%s], %s\n",
414                 LocalSignature, AcpiFormatException (Status));
415         return (-1);
416     }
417
418     if (ApDumpTableBuffer (Table, Instance, Address))
419     {
420         return (-1);
421     }
422     free (Table);
423 }
424
425 /* Something seriously bad happened if the loop terminates here */
426
427 return (-1);
428 }
429
430
431 /*****
432 *
433 * FUNCTION:      ApDumpTableFromFile
434 *
435 * PARAMETERS:    Pathname          - File containing the binary ACPI table
436 *
437 * RETURN:        Status
438 *
439 * DESCRIPTION:   Dump an ACPI table from a binary file
440 *
441 *****/
442
443 int
444 ApDumpTableFromFile (
445     char                *Pathname)
446 {
447     ACPI_TABLE_HEADER    *Table;
448     UINT32               FileSize = 0;
449     int                  TableStatus;
450
451     /* Get the entire ACPI table from the file */
452
453     Table = ApGetTableFromFile (Pathname, &FileSize);
454     if (!Table)

```

```

455 {
456     return (-1);
457 }
458
459 /* File must be at least as long as the table length */
460
461 if (Table->Length > FileSize)
462 {
463     fprintf (stderr,
464             "Table length (0x%X) is too large for input file (0x%X) %s\n",
465             Table->Length, FileSize, Pathname);
466     return (-1);
467 }
468
469 if (Gbl_VerboseMode)
470 {
471     fprintf (stderr,
472             "Input file: %s contains table [%4.4s], 0x%X (%u) bytes\n",
473             Pathname, Table->Signature, FileSize, FileSize);
474 }
475
476 TableStatus = ApDumpTableBuffer (Table, 0, 0);
477 free (Table);
478 return (TableStatus);
479 }
480
481 /*****
482 *
483 * FUNCTION:      AcpiOs* print functions
484 *
485 * DESCRIPTION:   Used for linkage with ACPICA modules
486 *
487 *****/
488
489 void ACPI_INTERNAL_VAR_XFACE
490 AcpiOsPrintf (
491     const char          *Fmt,
492     ...)
493 {
494     va_list              Args;
495
496     va_start (Args, Fmt);
497     vfprintf (stdout, Fmt, Args);
498     va_end (Args);
499 }
500
501 void
502 AcpiOsVprintf (
503     const char          *Fmt,
504     va_list              Args)
505 {
506     vfprintf (stdout, Fmt, Args);
507 }

```

new/usr/src/common/acpica/tools/acpidump/apfiles.c

1

```
*****
8016 Fri Jan 17 07:31:14 2014
new/usr/src/common/acpica/tools/acpidump/apfiles.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: apfiles - File-related functions for acpidump utility
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acpidump.h"
45 #include "acapps.h"

48 /*****
49 *
50 * FUNCTION: ApOpenOutputFile
51 *
52 * PARAMETERS: Pathname - Output filename
53 *
54 * RETURN: Open file handle
55 *
56 * DESCRIPTION: Open a text output file for acpidump. Checks if file already
57 * exists.
58 *
59 *****/
```

new/usr/src/common/acpica/tools/acpidump/apfiles.c

2

```
61 int
62 ApOpenOutputFile (
63     char *Pathname)
64 {
65     struct stat StatInfo;
66     FILE *File;

69     /* If file exists, prompt for overwrite */

71     if (!stat (Pathname, &StatInfo))
72     {
73         fprintf (stderr, "Target path already exists, overwrite? [y|n] ");

75         if (getchar () != 'y')
76         {
77             return (-1);
78         }
79     }

81     /* Point stdout to the file */

83     File = freopen (Pathname, "w", stdout);
84     if (!File)
85     {
86         perror ("Could not open output file");
87         return (-1);
88     }

90     /* Save the file and path */

92     Gbl_OutputFile = File;
93     Gbl_OutputFilename = Pathname;
94     return (0);
95 }

98 /*****
99 *
100 * FUNCTION: ApWriteToBinaryFile
101 *
102 * PARAMETERS: Table - ACPI table to be written
103 *             Instance - ACPI table instance no. to be written
104 *
105 * RETURN: Status
106 *
107 * DESCRIPTION: Write an ACPI table to a binary file. Builds the output
108 * filename from the table signature.
109 *
110 *****/

112 int
113 ApWriteToBinaryFile (
114     ACPI_TABLE_HEADER *Table,
115     UINT32 Instance)
116 {
117     char Filename[ACPI_NAME_SIZE + 16];
118     char InstanceStr [16];
119     FILE *File;
120     size_t Actual;
121     UINT32 TableLength;

124     /* Obtain table length */

126     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, ACPI_RSDP_NAME);
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 *
260 * PARAMETERS:  File           - Open file descriptor
261 *
262 * RETURN:      File size in bytes
263 *
264 * DESCRIPTION: Get the size of an open file
265 *
266 *****/

268 UINT32
269 ApGetFileSize (
270     FILE           *File)
271 {
272     UINT32         FileSize;
273     long           Offset;

276     Offset = ftell (File);
277     if (fseek (File, 0, SEEK_END))
278     {
279         return (0);
280     }

282     /* Get size and restore file pointer */

284     FileSize = (UINT32) ftell (File);
285     if (fseek (File, Offset, SEEK_SET))
286     {
287         return (0);
288     }

290     return (FileSize);
291 }
```



new/usr/src/common/acpica/tools/acpidump/apmain.c

1

```
*****
10358 Fri Jan 17 07:31:14 2014
new/usr/src/common/acpica/tools/acpidump/apmain.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: apmain - Main module for the acpidump utility
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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
45 #include "acpidump.h"
46 #include "acapps.h"

49 /*
50 * acpidump - A portable utility for obtaining system ACPI tables and dumping
51 * them in an ASCII hex format suitable for binary extraction via acpixtract.
52 *
53 * Obtaining the system ACPI tables is an OS-specific operation.
54 *
55 * This utility can be ported to any host operating system by providing a
56 * module containing system-specific versions of these interfaces:
57 *
58 *     AcpiOsGetTableByAddress
59 *     AcpiOsGetTableByIndex
60 *     AcpiOsGetTableByName
```

new/usr/src/common/acpica/tools/acpidump/apmain.c

2

```
61 *
62 * See the ACPICA Reference Guide for the exact definitions of these
63 * interfaces. Also, see these ACPICA source code modules for example
64 * implementations:
65 *
66 *     source/os_specific/service_layers/oswintbl.c
67 *     source/os_specific/service_layers/oslinuxtbl.c
68 */

71 /* Local prototypes */

73 static void
74 ApDisplayUsage (
75     void);

77 static int
78 ApDoOptions (
79     int             argc,
80     char            **argv);

82 static void
83 ApInsertAction (
84     char            *Argument,
85     UINT32          ToBeDone);

88 /* Table for deferred actions from command line options */

90 AP_DUMP_ACTION     ActionTable [AP_MAX_ACTIONS];
91 UINT32             CurrentAction = 0;

94 #define AP_UTILITY_NAME           "ACPI Binary Table Dump Utility"
95 #define AP_SUPPORTED_OPTIONS      "?a:bcf:hn:o:r:svz"

98 /*****
99 *
100 * FUNCTION:     ApDisplayUsage
101 *
102 * DESCRIPTION:  Usage message for the AcpiDump utility
103 *
104 *****/

106 static void
107 ApDisplayUsage (
108     void)
109 {

111     ACPI_USAGE_HEADER ("acpidump [options]");

113     ACPI_OPTION ("-b",             "Dump tables to binary files");
114     ACPI_OPTION ("-c",             "Dump customized tables");
115     ACPI_OPTION ("-h -?",          "This help message");
116     ACPI_OPTION ("-o <File>",      "Redirect output to file");
117     ACPI_OPTION ("-r <Address>",   "Dump tables from specified RSDP");
118     ACPI_OPTION ("-s",             "Print table summaries only");
119     ACPI_OPTION ("-v",             "Display version information");
120     ACPI_OPTION ("-z",             "Verbose mode");

122     printf ("\nTable Options:\n");

124     ACPI_OPTION ("-a <Address>",   "Get table via a physical address");
125     ACPI_OPTION ("-f <BinaryFile>", "Get table via a binary file");
126     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 {
153
154     /* Insert action and check for table overflow */
155
156     ActionTable [CurrentAction].Argument = Argument;
157     ActionTable [CurrentAction].ToBeDone = ToBeDone;
158
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 }

```

new/usr/src/common/acpica/tools/acpiexec/aecommon.h

1

```
*****
5484 Fri Jan 17 07:31:14 2014
new/usr/src/common/acpica/tools/acpiexec/aecommon.h
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: aecommon - common include for the AcpiExec utility
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 _AECOMMON
45 #define _AECOMMON

47 #ifdef _MSC_VER          /* disable some level-4 warnings */
48 #pragma warning(disable:4100) /* warning C4100: unreferenced formal parameter
49 #endif

51 #include "acpi.h"
52 #include "aecommon.h"
53 #include "acparser.h"
54 #include "amlcode.h"
55 #include "acnamesp.h"
56 #include "acdebug.h"
57 #include "actables.h"
58 #include "acinterp.h"
```

new/usr/src/common/acpica/tools/acpiexec/aecommon.h

2

```
59 #include "acapps.h"

61 #include <stdio.h>
62 #include <stdlib.h>
63 #include <string.h>
64 #include <signal.h>

66 extern BOOLEAN      AcpiGbl_IgnoreErrors;
67 extern UINT8        AcpiGbl_RegionFillValue;
68 extern UINT8        AcpiGbl_UseHwReducedFadt;
69 extern BOOLEAN      AcpiGbl_DisplayRegionAccess;
70 extern BOOLEAN      AcpiGbl_DoInterfaceTests;
71 extern BOOLEAN      AcpiGbl_LoadTestTables;

73 /* Check for unexpected exceptions */

75 #define AE_CHECK_STATUS(Name, Status, Expected) \
76     if (Status != Expected) \
77     { \
78         AcpiOsPrintf ("Unexpected %s from %s (%s-%d)\n", \
79             AcpiFormatException (Status), #Name, _AcpiModuleName, __LINE__); \
80     }

82 /* Check for unexpected non-AE_OK errors */

84 #define AE_CHECK_OK(Name, Status)    AE_CHECK_STATUS (Name, Status, AE_OK);

86 typedef struct ae_table_desc
87 {
88     ACPI_TABLE_HEADER    *Table;
89     struct ae_table_desc *Next;

91 } AE_TABLE_DESC;

93 /*
94 * Debug Regions
95 */
96 typedef struct ae_region
97 {
98     ACPI_PHYSICAL_ADDRESS Address;
99     UINT32                 Length;
100     void                   *Buffer;
101     void                   *NextRegion;
102     UINT8                  SpaceId;

104 } AE_REGION;

106 typedef struct ae_debug_regions
107 {
108     UINT32                 NumberOfRegions;
109     AE_REGION              *RegionList;

111 } AE_DEBUG_REGIONS;

114 #define TEST_OUTPUT_LEVEL(lvl)    if ((lvl) & OutputLevel)

116 #define OSD_PRINT(lvl,fp)         TEST_OUTPUT_LEVEL(lvl) {\
117     AcpiOsPrintf PARAM_LIST(fp);}

119 void ACPI_SYSTEM_XFACE
120 AeCtrlHandler (
121     int                Sig);

123 ACPI_STATUS
124 AeBuildLocalTables (
```

```

125     UINT32      TableCount,
126     AE_TABLE_DESC *TableList);

128 ACPI_STATUS
129 AeInstallTables (
130     void);

132 void
133 AeDumpNamespace (
134     void);

136 void
137 AeDumpObject (
138     char          *MethodName,
139     ACPI_BUFFER    *ReturnObj);

141 void
142 AeDumpBuffer (
143     UINT32          Address);

145 void
146 AeExecute (
147     char          *Name);

149 void
150 AeSetScope (
151     char          *Name);

153 void
154 AeCloseDebugFile (
155     void);

157 void
158 AeOpenDebugFile (
159     char          *Name);

161 ACPI_STATUS
162 AeDisplayAllMethods (
163     UINT32          DisplayCount);

165 ACPI_STATUS
166 AeInstallEarlyHandlers (
167     void);

169 ACPI_STATUS
170 AeInstallLateHandlers (
171     void);

173 void
174 AeMiscellaneousTests (
175     void);

177 ACPI_STATUS
178 AeRegionHandler (
179     UINT32          Function,
180     ACPI_PHYSICAL_ADDRESS Address,
181     UINT32          BitWidth,
182     UINT64          *Value,
183     void            *HandlerContext,
184     void            *RegionContext);

186 UINT32
187 AeGpeHandler (
188     ACPI_HANDLE      GpeDevice,
189     UINT32          GpeNumber,
190     void            *Context);

```

```

192 void
193 AeGlobalEventHandler (
194     UINT32          Type,
195     ACPI_HANDLE      GpeDevice,
196     UINT32          EventNumber,
197     void            *Context);

199 #endif /* _AECOMMON */

```

new/usr/src/common/acpica/tools/acpiexec/aeexec.c

1

```
*****
22774 Fri Jan 17 07:31:15 2014
new/usr/src/common/acpica/tools/acpiexec/aeexec.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: aeexec - Support routines for AcpiExec utility
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "aecommon.h"
45
46 #define _COMPONENT ACPI_TOOLS
47 ACPI_MODULE_NAME ("aeexec")
48
49 /* Local prototypes */
50
51 static ACPI_STATUS
52 AeSetupConfiguration (
53     void *RegionAddr);
54
55 static void
56 AeTestBufferArgument (
57     void);
```

new/usr/src/common/acpica/tools/acpiexec/aeexec.c

2

```
59 static void
60 AeTestPackageArgument (
61     void);
62
63 static ACPI_STATUS
64 AeGetDevices (
65     ACPI_HANDLE ObjHandle,
66     UINT32 NestingLevel,
67     void *Context,
68     void **ReturnValue);
69
70 static ACPI_STATUS
71 ExecuteOSI (
72     char *OsString,
73     UINT32 ExpectedResult);
74
75 static void
76 AeMutexInterfaces (
77     void);
78
79 static void
80 AeHardwareInterfaces (
81     void);
82
83 static void
84 AeGenericRegisters (
85     void);
86
87 #if (!ACPI_REDUCED_HARDWARE)
88 static void
89 AfInstallGpeBlock (
90     void);
91 #endif /* !ACPI_REDUCED_HARDWARE */
92
93 extern unsigned char Ssdt2Code[];
94 extern unsigned char Ssdt3Code[];
95 extern unsigned char Ssdt4Code[];
96
97
98 /*****
99 *
100 * FUNCTION: AeSetupConfiguration
101 *
102 * PARAMETERS: RegionAddr - Address for an ACPI table to be loaded
103 * dynamically. Test purposes only.
104 *
105 * RETURN: Status
106 *
107 * DESCRIPTION: Call AML _CFG configuration control method
108 *
109 *****/
110
111 static ACPI_STATUS
112 AeSetupConfiguration (
113     void *RegionAddr)
114 {
115     ACPI_OBJECT_LIST ArgList;
116     ACPI_OBJECT Arg[3];
117
118     /*
119     * Invoke _CFG method if present
120     */
121     ArgList.Count = 1;
122     ArgList.Pointer = Arg;
123 }
```

```

125     Arg[0].Type = ACPI_TYPE_INTEGER;
126     Arg[0].Integer.Value = ACPI_TO_INTEGER (RegionAddr);

128     (void) AcpiEvaluateObject (NULL, "\\_CFG", &ArgList, NULL);
129     return (AE_OK);
130 }

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  *****/

147 static void
148 AfInstallGpeBlock (
149     void)
150 {
151     ACPI_STATUS          Status;
152     ACPI_HANDLE          Handle;
153     ACPI_GENERIC_ADDRESS BlockAddress;
154     ACPI_HANDLE          GpeDevice;
155     ACPI_OBJECT_TYPE     Type;

158     /* _GPE should always exist */

160     Status = AcpiGetHandle (NULL, "\\_GPE", &Handle);
161     AE_CHECK_OK (AcpiGetHandle, Status);
162     if (ACPI_FAILURE (Status))
163     {
164         return;
165     }

167     ACPI_MEMSET (&BlockAddress, 0, sizeof (ACPI_GENERIC_ADDRESS));
168     BlockAddress.SpaceId = ACPI_ADR_SPACE_SYSTEM_MEMORY;
169     BlockAddress.Address = 0x76540000;

171     /* Attempt to install a GPE block on GPE2 (if present) */

173     Status = AcpiGetHandle (NULL, "\\GPE2", &Handle);
174     if (ACPI_SUCCESS (Status))
175     {
176         Status = AcpiGetType (Handle, &Type);
177         if (ACPI_FAILURE (Status) ||
178             (Type != ACPI_TYPE_DEVICE))
179         {
180             return;
181         }

183         Status = AcpiInstallGpeBlock (Handle, &BlockAddress, 7, 8);
184         AE_CHECK_OK (AcpiInstallGpeBlock, Status);

186         Status = AcpiInstallGpeHandler (Handle, 8,
187             ACPI_GPE_LEVEL_TRIGGERED, AeGpeHandler, NULL);
188         AE_CHECK_OK (AcpiInstallGpeHandler, Status);

190         Status = AcpiEnableGpe (Handle, 8);

```

```

191     AE_CHECK_OK (AcpiEnableGpe, Status);

193     Status = AcpiGetGpeDevice (0x30, &GpeDevice);
194     AE_CHECK_OK (AcpiGetGpeDevice, Status);

196     Status = AcpiGetGpeDevice (0x42, &GpeDevice);
197     AE_CHECK_OK (AcpiGetGpeDevice, Status);

199     Status = AcpiGetGpeDevice (AcpiCurrentGpeCount-1, &GpeDevice);
200     AE_CHECK_OK (AcpiGetGpeDevice, Status);

202     Status = AcpiGetGpeDevice (AcpiCurrentGpeCount, &GpeDevice);
203     AE_CHECK_STATUS (AcpiGetGpeDevice, Status, AE_NOT_EXIST);

205     Status = AcpiRemoveGpeHandler (Handle, 8, AeGpeHandler);
206     AE_CHECK_OK (AcpiRemoveGpeHandler, Status);
207 }

209 /* Attempt to install a GPE block on GPE3 (if present) */

211 Status = AcpiGetHandle (NULL, "\\GPE3", &Handle);
212 if (ACPI_SUCCESS (Status))
213 {
214     Status = AcpiGetType (Handle, &Type);
215     if (ACPI_FAILURE (Status) ||
216         (Type != ACPI_TYPE_DEVICE))
217     {
218         return;
219     }

221     Status = AcpiInstallGpeBlock (Handle, &BlockAddress, 8, 11);
222     AE_CHECK_OK (AcpiInstallGpeBlock, Status);
223 }
224 }
225 #endif /* !ACPI_REDUCED_HARDWARE */

228 /* Test using a Buffer object as a method argument */

230 static void
231 AeTestBufferArgument (
232     void)
233 {
234     ACPI_OBJECT_LIST     Params;
235     ACPI_OBJECT          BufArg;
236     UINT8                Buffer[] = {
237         0,0,0,0,
238         4,0,0,0,
239         1,2,3,4};

242     BufArg.Type = ACPI_TYPE_BUFFER;
243     BufArg.Buffer.Length = 12;
244     BufArg.Buffer.Pointer = Buffer;

246     Params.Count = 1;
247     Params.Pointer = &BufArg;

249     (void) AcpiEvaluateObject (NULL, "\\BUF", &Params, NULL);
250 }

253 static ACPI_OBJECT          PkgArg;
254 static ACPI_OBJECT          PkgElements[5];
255 static ACPI_OBJECT          Pkg2Elements[5];
256 static ACPI_OBJECT_LIST     Params;

```

```

259 /*
260  * Test using a Package object as an method argument
261  */
262 static void
263 AeTestPackageArgument (
264     void)
265 {
266     /* Main package */
267
268     PkgArg.Type = ACPI_TYPE_PACKAGE;
269     PkgArg.Package.Count = 4;
270     PkgArg.Package.Elements = PkgElements;
271
272     /* Main package elements */
273
274     PkgElements[0].Type = ACPI_TYPE_INTEGER;
275     PkgElements[0].Integer.Value = 0x22228888;
276
277     PkgElements[1].Type = ACPI_TYPE_STRING;
278     PkgElements[1].String.Length = sizeof ("Top-level package");
279     PkgElements[1].String.Pointer = "Top-level package";
280
281     PkgElements[2].Type = ACPI_TYPE_BUFFER;
282     PkgElements[2].Buffer.Length = sizeof ("XXXX");
283     PkgElements[2].Buffer.Pointer = (UINT8 *) "XXXX";
284
285     PkgElements[3].Type = ACPI_TYPE_PACKAGE;
286     PkgElements[3].Package.Count = 2;
287     PkgElements[3].Package.Elements = Pkg2Elements;
288
289     /* Sub-package elements */
290
291     Pkg2Elements[0].Type = ACPI_TYPE_INTEGER;
292     Pkg2Elements[0].Integer.Value = 0xAAAABBBB;
293
294     Pkg2Elements[1].Type = ACPI_TYPE_STRING;
295     Pkg2Elements[1].String.Length = sizeof ("Nested Package");
296     Pkg2Elements[1].String.Pointer = "Nested Package";
297
298     /* Parameter object */
299
300     Params.Count = 1;
301     Params.Pointer = &PkgArg;
302
303     (void) AcpiEvaluateObject (NULL, "\\_PKG", &Params, NULL);
304 }
305
306 static ACPI_STATUS
307 AeGetDevices (
308     ACPI_HANDLE          ObjHandle,
309     UINT32               NestingLevel,
310     void                 *Context,
311     void                 **ReturnValue)
312 {
313     return (AE_OK);
314 }
315
316
317
318
319
320 /*****
321  *
322  * FUNCTION:      ExecuteOSI

```

```

323  *
324  * PARAMETERS:   OsiString          - String passed to _OSI method
325  *               ExpectedResult     - 0 (FALSE) or 0xFFFFFFFF (TRUE)
326  *
327  * RETURN:       Status
328  *
329  * DESCRIPTION:  Execute the internally implemented (in ACPICA) _OSI method.
330  *
331  *****/
332
333 static ACPI_STATUS
334 ExecuteOSI (
335     char          *OsiString,
336     UINT32        ExpectedResult)
337 {
338     ACPI_STATUS    Status;
339     ACPI_OBJECT_LIST ArgList;
340     ACPI_OBJECT    Arg[1];
341     ACPI_BUFFER     ReturnValue;
342     ACPI_OBJECT     *Obj;
343
344     /* Setup input argument */
345
346     ArgList.Count = 1;
347     ArgList.Pointer = Arg;
348
349     Arg[0].Type = ACPI_TYPE_STRING;
350     Arg[0].String.Pointer = OsiString;
351     Arg[0].String.Length = strlen (Arg[0].String.Pointer);
352
353     /* Ask ACPICA to allocate space for the return object */
354
355     ReturnValue.Length = ACPI_ALLOCATE_BUFFER;
356
357     Status = AcpiEvaluateObject (NULL, "\\_OSI", &ArgList, &ReturnValue);
358
359     if (ACPI_FAILURE (Status))
360     {
361         AcpiOsPrintf ("Could not execute _OSI method, %s\n",
362             AcpiFormatException (Status));
363         return (Status);
364     }
365
366     Status = AE_ERROR;
367
368     if (ReturnValue.Length < sizeof (ACPI_OBJECT))
369     {
370         AcpiOsPrintf ("Return value from _OSI method too small, %.8X\n",
371             ReturnValue.Length);
372         goto ErrorExit;
373     }
374
375     Obj = ReturnValue.Pointer;
376     if (Obj->Type != ACPI_TYPE_INTEGER)
377     {
378         AcpiOsPrintf ("Invalid return type from _OSI method, %.2X\n", Obj->Type);
379         goto ErrorExit;
380     }
381
382     if (Obj->Integer.Value != ExpectedResult)
383     {
384         AcpiOsPrintf ("Invalid return value from _OSI, expected %.8X found %.8X\n",
385             ExpectedResult, (UINT32) Obj->Integer.Value);
386         goto ErrorExit;
387     }
388 }

```



```

390     Status = AE_OK;

392     /* Reset the OSI data */

394     AcpiGbl_OsiData = 0;

396 ErrorExit:

398     /* Free a buffer created via ACPI_ALLOCATE_BUFFER */

400     AcpiOsFree (ReturnValue.Pointer);

402     return (Status);
403 }

406 /*****
407  *
408  * FUNCTION:     AeGenericRegisters
409  *
410  * DESCRIPTION:  Call the AcpiRead/Write interfaces.
411  *
412  *****/

414 static ACPI_GENERIC_ADDRESS      GenericRegister;

416 static void
417 AeGenericRegisters (
418     void)
419 {
420     ACPI_STATUS      Status;
421     UINT64           Value;

424     GenericRegister.Address = 0x1234;
425     GenericRegister.BitWidth = 64;
426     GenericRegister.BitOffset = 0;
427     GenericRegister.SpaceId = ACPI_ADR_SPACE_SYSTEM_IO;

429     Status = AcpiRead (&Value, &GenericRegister);
430     AE_CHECK_OK (AcpiRead, Status);

432     Status = AcpiWrite (Value, &GenericRegister);
433     AE_CHECK_OK (AcpiWrite, Status);

435     GenericRegister.Address = 0x12345678;
436     GenericRegister.BitOffset = 0;
437     GenericRegister.SpaceId = ACPI_ADR_SPACE_SYSTEM_MEMORY;

439     Status = AcpiRead (&Value, &GenericRegister);
440     AE_CHECK_OK (AcpiRead, Status);

442     Status = AcpiWrite (Value, &GenericRegister);
443     AE_CHECK_OK (AcpiWrite, Status);
444 }

447 /*****
448  *
449  * FUNCTION:     AeMutexInterfaces
450  *
451  * DESCRIPTION:  Exercise the AML mutex access interfaces
452  *
453  *****/

```

```

455 static void
456 AeMutexInterfaces (
457     void)
458 {
459     ACPI_STATUS      Status;
460     ACPI_HANDLE      MutexHandle;

463     /* Get a handle to an AML mutex */

465     Status = AcpiGetHandle (NULL, "\\MTX1", &MutexHandle);
466     if (Status == AE_NOT_FOUND)
467     {
468         return;
469     }

471     AE_CHECK_OK (AcpiGetHandle, Status);
472     if (ACPI_FAILURE (Status))
473     {
474         return;
475     }

477     /* Acquire the mutex */

479     Status = AcpiAcquireMutex (NULL, "\\MTX1", 0xFFFF);
480     AE_CHECK_OK (AcpiAcquireMutex, Status);
481     if (ACPI_FAILURE (Status))
482     {
483         return;
484     }

486     /* Release mutex with different parameters */

488     Status = AcpiReleaseMutex (MutexHandle, NULL);
489     AE_CHECK_OK (AcpiReleaseMutex, Status);
490 }

493 /*****
494  *
495  * FUNCTION:     AeHardwareInterfaces
496  *
497  * DESCRIPTION:  Call various hardware support interfaces
498  *
499  *****/

501 static void
502 AeHardwareInterfaces (
503     void)
504 {
505     #if (!ACPI_REDUCED_HARDWARE)

507     ACPI_STATUS      Status;
508     UINT32           Value;

511     /* If Hardware Reduced flag is set, we are all done */

513     if (AcpiGbl_ReducedHardware)
514     {
515         return;
516     }

518     Status = AcpiWriteBitRegister (ACPI_BITREG_WAKE_STATUS, 1);
519     AE_CHECK_OK (AcpiWriteBitRegister, Status);

```

```

521 Status = AcpiWriteBitRegister (ACPI_BITREG_GLOBAL_LOCK_ENABLE, 1);
522 AE_CHECK_OK (AcpiWriteBitRegister, Status);

524 Status = AcpiWriteBitRegister (ACPI_BITREG_SLEEP_ENABLE, 1);
525 AE_CHECK_OK (AcpiWriteBitRegister, Status);

527 Status = AcpiWriteBitRegister (ACPI_BITREG_ARB_DISABLE, 1);
528 AE_CHECK_OK (AcpiWriteBitRegister, Status);

531 Status = AcpiReadBitRegister (ACPI_BITREG_WAKE_STATUS, &Value);
532 AE_CHECK_OK (AcpiReadBitRegister, Status);

534 Status = AcpiReadBitRegister (ACPI_BITREG_GLOBAL_LOCK_ENABLE, &Value);
535 AE_CHECK_OK (AcpiReadBitRegister, Status);

537 Status = AcpiReadBitRegister (ACPI_BITREG_SLEEP_ENABLE, &Value);
538 AE_CHECK_OK (AcpiReadBitRegister, Status);

540 Status = AcpiReadBitRegister (ACPI_BITREG_ARB_DISABLE, &Value);
541 AE_CHECK_OK (AcpiReadBitRegister, Status);

543 #endif /* !ACPI_REduced_HARDWARE */
544 }

547 /*****
548  *
549  * FUNCTION:      AeMiscellaneousTests
550  *
551  * DESCRIPTION:  Various ACPICA validation tests.
552  *
553  *****/

555 void
556 AeMiscellaneousTests (
557     void)
558 {
559     ACPI_BUFFER      ReturnBuf;
560     char              Buffer[32];
561     ACPI_STATUS       Status;
562     ACPI_STATISTICS   Stats;
563     ACPI_HANDLE       Handle;

565     #if (!ACPI_REduced_HARDWARE)
566     ACPI_VENDOR_UUID Uuid = {0, {ACPI_INIT_UUID (0,0,0,0,0,0,0,0,0,0)}}
567     UINT32            LockHandle1;
568     UINT32            LockHandle2;
569     #endif /* !ACPI_REduced_HARDWARE */

572     Status = AcpiGetHandle (NULL, "\\_", &Handle);
573     AE_CHECK_OK (AcpiGetHandle, Status);

575     if (AcpiGbl_DoInterfaceTests)
576     {
577         /*
578          * Tests for AcpiLoadTable and AcpiUnloadParentTable
579          */

581         /* Attempt unload of DSDT, should fail */

583         Status = AcpiGetHandle (NULL, "\\_SB_", &Handle);
584         AE_CHECK_OK (AcpiGetHandle, Status);

586         Status = AcpiUnloadParentTable (Handle);

```

```

587     AE_CHECK_STATUS (AcpiUnloadParentTable, Status, AE_TYPE);

589     /* Load and unload SSDT4 */

591     Status = AcpiLoadTable ((ACPI_TABLE_HEADER *) Ssdt4Code);
592     AE_CHECK_OK (AcpiLoadTable, Status);

594     Status = AcpiGetHandle (NULL, "\\_T96", &Handle);
595     AE_CHECK_OK (AcpiGetHandle, Status);

597     Status = AcpiUnloadParentTable (Handle);
598     AE_CHECK_OK (AcpiUnloadParentTable, Status);

600     /* Re-load SSDT4 */

602     Status = AcpiLoadTable ((ACPI_TABLE_HEADER *) Ssdt4Code);
603     AE_CHECK_OK (AcpiLoadTable, Status);

605     /* Unload and re-load SSDT2 (SSDT2 is in the XSDT) */

607     Status = AcpiGetHandle (NULL, "\\_T99", &Handle);
608     AE_CHECK_OK (AcpiGetHandle, Status);

610     Status = AcpiUnloadParentTable (Handle);
611     AE_CHECK_OK (AcpiUnloadParentTable, Status);

613     Status = AcpiLoadTable ((ACPI_TABLE_HEADER *) Ssdt2Code);
614     AE_CHECK_OK (AcpiLoadTable, Status);

616     /* Load OEM9 table (causes table override) */

618     Status = AcpiLoadTable ((ACPI_TABLE_HEADER *) Ssdt3Code);
619     AE_CHECK_OK (AcpiLoadTable, Status);
620 }

622 AeHardwareInterfaces ();
623 AeGenericRegisters ();
624 AeSetupConfiguration (Ssdt3Code);

626 AeTestBufferArgument();
627 AeTestPackageArgument ();
628 AeMutexInterfaces ();

630 /* Test _OSI install/remove */

632 Status = AcpiInstallInterface ("");
633 AE_CHECK_STATUS (AcpiInstallInterface, Status, AE_BAD_PARAMETER);

635 Status = AcpiInstallInterface ("TestString");
636 AE_CHECK_OK (AcpiInstallInterface, Status);

638 Status = AcpiInstallInterface ("TestString");
639 AE_CHECK_STATUS (AcpiInstallInterface, Status, AE_ALREADY_EXISTS);

641 Status = AcpiRemoveInterface ("Windows 2006");
642 AE_CHECK_OK (AcpiRemoveInterface, Status);

644 Status = AcpiRemoveInterface ("TestString");
645 AE_CHECK_OK (AcpiRemoveInterface, Status);

647 Status = AcpiRemoveInterface ("XXXXXX");
648 AE_CHECK_STATUS (AcpiRemoveInterface, Status, AE_NOT_EXIST);

650 Status = AcpiInstallInterface ("AnotherTestString");
651 AE_CHECK_OK (AcpiInstallInterface, Status);

```

```

653  /* Test _OSI execution */
655  Status = ExecuteOSI ("Extended Address Space Descriptor", 0xFFFFFFFF);
656  AE_CHECK_OK (ExecuteOSI, Status);

658  Status = ExecuteOSI ("Windows 2001", 0xFFFFFFFF);
659  AE_CHECK_OK (ExecuteOSI, Status);

661  Status = ExecuteOSI ("MichiganTerminalSystem", 0);
662  AE_CHECK_OK (ExecuteOSI, Status);

665  ReturnBuf.Length = 32;
666  ReturnBuf.Pointer = Buffer;

668  Status = AcpiGetName (ACPI_ROOT_OBJECT, ACPI_FULL_PATHNAME, &ReturnBuf);
669  AE_CHECK_OK (AcpiGetName, Status);

671  /* Get Devices */

673  Status = AcpiGetDevices (NULL, AeGetDevices, NULL, NULL);
674  AE_CHECK_OK (AcpiGetDevices, Status);

676  Status = AcpiGetStatistics (&Stats);
677  AE_CHECK_OK (AcpiGetStatistics, Status);

680 #if (!ACPI_REDUCED_HARDWARE)

682  Status = AcpiInstallGlobalEventHandler (AeGlobalEventHandler, NULL);
683  AE_CHECK_OK (AcpiInstallGlobalEventHandler, Status);

685  /* If Hardware Reduced flag is set, we are all done */

687  if (AcpiGbl_ReducedHardware)
688  {
689      return;
690  }

692  Status = AcpiEnableEvent (ACPI_EVENT_GLOBAL, 0);
693  AE_CHECK_OK (AcpiEnableEvent, Status);

695  /*
696   * GPEs: Handlers, enable/disable, etc.
697   */
698  Status = AcpiInstallGpeHandler (NULL, 0, ACPI_GPE_LEVEL_TRIGGERED, AeGpeHand
699  AE_CHECK_OK (AcpiInstallGpeHandler, Status);

701  Status = AcpiEnableGpe (NULL, 0);
702  AE_CHECK_OK (AcpiEnableGpe, Status);

704  Status = AcpiRemoveGpeHandler (NULL, 0, AeGpeHandler);
705  AE_CHECK_OK (AcpiRemoveGpeHandler, Status);

707  Status = AcpiInstallGpeHandler (NULL, 0, ACPI_GPE_LEVEL_TRIGGERED, AeGpeHand
708  AE_CHECK_OK (AcpiInstallGpeHandler, Status);

710  Status = AcpiEnableGpe (NULL, 0);
711  AE_CHECK_OK (AcpiEnableGpe, Status);

713  Status = AcpiSetGpe (NULL, 0, ACPI_GPE_DISABLE);
714  AE_CHECK_OK (AcpiSetGpe, Status);

716  Status = AcpiSetGpe (NULL, 0, ACPI_GPE_ENABLE);
717  AE_CHECK_OK (AcpiSetGpe, Status);

```

```

720  Status = AcpiInstallGpeHandler (NULL, 1, ACPI_GPE_EDGE_TRIGGERED, AeGpeHandl
721  AE_CHECK_OK (AcpiInstallGpeHandler, Status);

723  Status = AcpiEnableGpe (NULL, 1);
724  AE_CHECK_OK (AcpiEnableGpe, Status);

727  Status = AcpiInstallGpeHandler (NULL, 2, ACPI_GPE_LEVEL_TRIGGERED, AeGpeHand
728  AE_CHECK_OK (AcpiInstallGpeHandler, Status);

730  Status = AcpiEnableGpe (NULL, 2);
731  AE_CHECK_OK (AcpiEnableGpe, Status);

734  Status = AcpiInstallGpeHandler (NULL, 3, ACPI_GPE_EDGE_TRIGGERED, AeGpeHandl
735  AE_CHECK_OK (AcpiInstallGpeHandler, Status);

737  Status = AcpiInstallGpeHandler (NULL, 4, ACPI_GPE_LEVEL_TRIGGERED, AeGpeHand
738  AE_CHECK_OK (AcpiInstallGpeHandler, Status);

740  Status = AcpiInstallGpeHandler (NULL, 5, ACPI_GPE_EDGE_TRIGGERED, AeGpeHandl
741  AE_CHECK_OK (AcpiInstallGpeHandler, Status);

743  Status = AcpiGetHandle (NULL, "\\_SB", &Handle);
744  AE_CHECK_OK (AcpiGetHandle, Status);

746  Status = AcpiSetupGpeForWake (Handle, NULL, 5);
747  AE_CHECK_OK (AcpiSetupGpeForWake, Status);

749  Status = AcpiSetGpeWakeMask (NULL, 5, ACPI_GPE_ENABLE);
750  AE_CHECK_OK (AcpiSetGpeWakeMask, Status);

752  Status = AcpiSetupGpeForWake (Handle, NULL, 6);
753  AE_CHECK_OK (AcpiSetupGpeForWake, Status);

755  Status = AcpiSetupGpeForWake (ACPI_ROOT_OBJECT, NULL, 6);
756  AE_CHECK_OK (AcpiSetupGpeForWake, Status);

758  Status = AcpiSetupGpeForWake (Handle, NULL, 9);
759  AE_CHECK_OK (AcpiSetupGpeForWake, Status);

761  Status = AcpiInstallGpeHandler (NULL, 0x19, ACPI_GPE_LEVEL_TRIGGERED, AeGpeH
762  AE_CHECK_OK (AcpiInstallGpeHandler, Status);

764  Status = AcpiEnableGpe (NULL, 0x19);
765  AE_CHECK_OK (AcpiEnableGpe, Status);

768  Status = AcpiInstallGpeHandler (NULL, 0x62, ACPI_GPE_LEVEL_TRIGGERED, AeGpeH
769  AE_CHECK_OK (AcpiInstallGpeHandler, Status);

771  Status = AcpiEnableGpe (NULL, 0x62);
772  AE_CHECK_OK (AcpiEnableGpe, Status);

774  Status = AcpiDisableGpe (NULL, 0x62);
775  AE_CHECK_OK (AcpiDisableGpe, Status);

777  AfInstallGpeBlock ();

779  /* Here is where the GPEs are actually "enabled" */

781  Status = AcpiUpdateAllGpes ();
782  AE_CHECK_OK (AcpiUpdateAllGpes, Status);

784  Status = AcpiGetHandle (NULL, "RSRC", &Handle);

```

```
785     if (ACPI_SUCCESS (Status))
786     {
787         ReturnBuf.Length = ACPI_ALLOCATE_BUFFER;
788
789         Status = AcpiGetVendorResource (Handle, "_CRS", &Uuid, &ReturnBuf);
790         if (ACPI_SUCCESS (Status))
791         {
792             AcpiOsFree (ReturnBuf.Pointer);
793         }
794     }
795
796     /* Test global lock */
797
798     Status = AcpiAcquireGlobalLock (0xFFFF, &LockHandle1);
799     AE_CHECK_OK (AcpiAcquireGlobalLock, Status);
800
801     Status = AcpiAcquireGlobalLock (0x5, &LockHandle2);
802     AE_CHECK_OK (AcpiAcquireGlobalLock, Status);
803
804     Status = AcpiReleaseGlobalLock (LockHandle1);
805     AE_CHECK_OK (AcpiReleaseGlobalLock, Status);
806
807     Status = AcpiReleaseGlobalLock (LockHandle2);
808     AE_CHECK_OK (AcpiReleaseGlobalLock, Status);
809
810 #endif /* !ACPI_REduced_HARDWARE */
811 }
```

new/usr/src/common/acpica/tools/acpiexec/aehandlers.c

1

```
*****
53161 Fri Jan 17 07:31:15 2014
new/usr/src/common/acpica/tools/acpiexec/aehandlers.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20131115
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: aehandlers - Various handlers for acpiexec
4 *
5 *****/
7 /*****
8 *
9 * 1. Copyright Notice
10 *
11 * Some or all of this work - Copyright (c) 1999 - 2014, Intel Corp.
12 * All rights reserved.
13 *
14 * 2. License
15 *
16 * 2.1. This is your license from Intel Corp. under its intellectual property
17 * rights. You may have additional license terms from the party that provided
18 * you this software, covering your right to use that party's intellectual
19 * property rights.
20 *
21 * 2.2. Intel grants, free of charge, to any person ("Licensee") obtaining a
22 * copy of the source code appearing in this file ("Covered Code") an
23 * irrevocable, perpetual, worldwide license under Intel's copyrights in the
24 * base code distributed originally by Intel ("Original Intel Code") to copy,
25 * make derivatives, distribute, use and display any portion of the Covered
26 * Code in any form, with the right to sublicense such rights; and
27 *
28 * 2.3. Intel grants Licensee a non-exclusive and non-transferable patent
29 * license (with the right to sublicense), under only those claims of Intel
30 * patents that are infringed by the Original Intel Code, to make, use, sell,
31 * offer to sell, and import the Covered Code and derivative works thereof
32 * solely to the minimum extent necessary to exercise the above copyright
33 * license, and in no event shall the patent license extend to any additions
34 * to or modifications of the Original Intel Code. No other license or right
35 * is granted directly or by implication, estoppel or otherwise;
36 *
37 * The above copyright and patent license is granted only if the following
38 * conditions are met:
39 *
40 * 3. Conditions
41 *
42 * 3.1. Redistribution of Source with Rights to Further Distribute Source.
43 * Redistribution of source code of any substantial portion of the Covered
44 * Code or modification with rights to further distribute source must include
45 * the above Copyright Notice, the above License, this list of Conditions,
46 * and the following Disclaimer and Export Compliance provision. In addition,
47 * Licensee must cause all Covered Code to which Licensee contributes to
48 * contain a file documenting the changes Licensee made to create that Covered
49 * Code and the date of any change. Licensee must include in that file the
50 * documentation of any changes made by any predecessor Licensee. Licensee
51 * must include a prominent statement that the modification is derived,
52 * directly or indirectly, from Original Intel Code.
53 *
54 * 3.2. Redistribution of Source with no Rights to Further Distribute Source.
55 * Redistribution of source code of any substantial portion of the Covered
56 * Code or modification without rights to further distribute source must
```

new/usr/src/common/acpica/tools/acpiexec/aehandlers.c

2

```
57 * include the following Disclaimer and Export Compliance provision in the
58 * documentation and/or other materials provided with distribution. In
59 * addition, Licensee may not authorize further sublicense of source of any
60 * portion of the Covered Code, and must include terms to the effect that the
61 * license from Licensee to its licensee is limited to the intellectual
62 * property embodied in the software Licensee provides to its licensee, and
63 * not to intellectual property embodied in modifications its licensee may
64 * make.
65 *
66 * 3.3. Redistribution of Executable. Redistribution in executable form of any
67 * substantial portion of the Covered Code or modification must reproduce the
68 * above Copyright Notice, and the following Disclaimer and Export Compliance
69 * provision in the documentation and/or other materials provided with the
70 * distribution.
71 *
72 * 3.4. Intel retains all right, title, and interest in and to the Original
73 * Intel Code.
74 *
75 * 3.5. Neither the name Intel nor any other trademark owned or controlled by
76 * Intel shall be used in advertising or otherwise to promote the sale, use or
77 * other dealings in products derived from or relating to the Covered Code
78 * without prior written authorization from Intel.
79 *
80 * 4. Disclaimer and Export Compliance
81 *
82 * 4.1. INTEL MAKES NO WARRANTY OF ANY KIND REGARDING ANY SOFTWARE PROVIDED
83 * HERE. ANY SOFTWARE ORIGINATING FROM INTEL OR DERIVED FROM INTEL SOFTWARE
84 * IS PROVIDED "AS IS," AND INTEL WILL NOT PROVIDE ANY SUPPORT, ASSISTANCE,
85 * INSTALLATION, TRAINING OR OTHER SERVICES. INTEL WILL NOT PROVIDE ANY
86 * UPDATES, ENHANCEMENTS OR EXTENSIONS. INTEL SPECIFICALLY DISCLAIMS ANY
87 * IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A
88 * PARTICULAR PURPOSE.
89 *
90 * 4.2. IN NO EVENT SHALL INTEL HAVE ANY LIABILITY TO LICENSEE, ITS LICENSEES
91 * OR ANY OTHER THIRD PARTY, FOR ANY LOST PROFITS, LOST DATA, LOSS OF USE OR
92 * COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY INDIRECT,
93 * SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS AGREEMENT, UNDER ANY
94 * CAUSE OF ACTION OR THEORY OF LIABILITY, AND IRRESPECTIVE OF WHETHER INTEL
95 * HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES. THESE LIMITATIONS
96 * SHALL APPLY NOTWITHSTANDING THE FAILURE OF THE ESSENTIAL PURPOSE OF ANY
97 * LIMITED REMEDY.
98 *
99 * 4.3. Licensee shall not export, either directly or indirectly, any of this
100 * software or system incorporating such software without first obtaining any
101 * required license or other approval from the U. S. Department of Commerce or
102 * any other agency or department of the United States Government. In the
103 * event Licensee exports any such software from the United States or
104 * re-exports any such software from a foreign destination, Licensee shall
105 * ensure that the distribution and export/re-export of the software is in
106 * compliance with all laws, regulations, orders, or other restrictions of the
107 * U.S. Export Administration Regulations. Licensee agrees that neither it nor
108 * any of its subsidiaries will export/re-export any technical data, process,
109 * software, or service, directly or indirectly, to any country for which the
110 * United States government or any agency thereof requires an export license,
111 * other governmental approval, or letter of assurance, without first obtaining
112 * such license, approval or letter.
113 *
114 *****/
116 #include "aecommon.h"
118 #define _COMPONENT          ACPI_TOOLS
119         ACPI_MODULE_NAME    ("aehandlers")
121 /* Local prototypes */
```

```

123 static void
124 AeNotifyHandler1 (
125     ACPI_HANDLE      Device,
126     UINT32            Value,
127     void              *Context);

129 static void
130 AeNotifyHandler2 (
131     ACPI_HANDLE      Device,
132     UINT32            Value,
133     void              *Context);

135 static void
136 AeCommonNotifyHandler (
137     ACPI_HANDLE      Device,
138     UINT32            Value,
139     UINT32            HandlerId);

141 static void
142 AeDeviceNotifyHandler (
143     ACPI_HANDLE      Device,
144     UINT32            Value,
145     void              *Context);

147 static ACPI_STATUS
148 AeExceptionHandler (
149     ACPI_STATUS      AmlStatus,
150     ACPI_NAME         Name,
151     UINT16            Opcode,
152     UINT32            AmlOffset,
153     void              *Context);

155 static ACPI_STATUS
156 AeTableHandler (
157     UINT32            Event,
158     void              *Table,
159     void              *Context);

161 static ACPI_STATUS
162 AeRegionInit (
163     ACPI_HANDLE      RegionHandle,
164     UINT32            Function,
165     void              *HandlerContext,
166     void              **RegionContext);

168 static void
169 AeAttachedDataHandler (
170     ACPI_HANDLE      Object,
171     void              *Data);

173 static void
174 AeAttachedDataHandler2 (
175     ACPI_HANDLE      Object,
176     void              *Data);

178 static UINT32
179 AeInterfaceHandler (
180     ACPI_STRING       InterfaceName,
181     UINT32            Supported);

183 static ACPI_STATUS
184 AeInstallEcHandler (
185     ACPI_HANDLE      ObjHandle,
186     UINT32            Level,
187     void              *Context,
188     void              **ReturnValue);

```

```

190 static ACPI_STATUS
191 AeInstallPciHandler (
192     ACPI_HANDLE      ObjHandle,
193     UINT32            Level,
194     void              *Context,
195     void              **ReturnValue);

197 static ACPI_STATUS
198 AeInstallDeviceHandlers (
199     void);

201 #if (!ACPI_REduced_HARDWARE)
202 static UINT32
203 AeEventHandler (
204     void              *Context);

206 static UINT32
207 AeSciHandler (
208     void              *Context);

210 static char          *TableEvents[] =
211 {
212     "LOAD",
213     "UNLOAD",
214     "UNKNOWN"
215 };
216 #endif /* !ACPI_REduced_HARDWARE */

219 static UINT32          SigintCount = 0;
220 static AE_DEBUG_REGIONS AeRegions;
221 BOOLEAN                AcpiGbl_DisplayRegionAccess = FALSE;

223 /*
224  * We will override some of the default region handlers, especially
225  * the SystemMemory handler, which must be implemented locally.
226  * These handlers are installed "early" - before any _REG methods
227  * are executed - since they are special in the sense that the ACPI spec
228  * declares that they must "always be available". Cannot override the
229  * DataTable region handler either -- needed for test execution.
230  *
231  * NOTE: The local region handler will simulate access to these address
232  * spaces by creating a memory buffer behind each operation region.
233  */
234 static ACPI_ADR_SPACE_TYPE DefaultSpaceIdList[] =
235 {
236     ACPI_ADR_SPACE_SYSTEM_MEMORY,
237     ACPI_ADR_SPACE_SYSTEM_IO,
238     ACPI_ADR_SPACE_PCI_CONFIG,
239     ACPI_ADR_SPACE_EC
240 };

242 /*
243  * We will install handlers for some of the various address space IDs.
244  * Test one user-defined address space (used by aslts).
245  */
246 #define ACPI_ADR_SPACE_USER_DEFINED1    0x80
247 #define ACPI_ADR_SPACE_USER_DEFINED2    0xE4

249 static ACPI_ADR_SPACE_TYPE SpaceIdList[] =
250 {
251     ACPI_ADR_SPACE_SMBUS,
252     ACPI_ADR_SPACE_CMOS,
253     ACPI_ADR_SPACE_PCI_BAR_TARGET,
254     ACPI_ADR_SPACE_IPMI,

```

```

255     ACPI_ADR_SPACE_GPIO,
256     ACPI_ADR_SPACE_GSBUS,
257     ACPI_ADR_SPACE_FIXED_HARDWARE,
258     ACPI_ADR_SPACE_USER_DEFINED1,
259     ACPI_ADR_SPACE_USER_DEFINED2
260 };

262 static ACPI_CONNECTION_INFO  AeMyContext;

265 /*****
266  *
267  * FUNCTION:      AeCtrlCHandler
268  *
269  * PARAMETERS:    Sig
270  *
271  * RETURN:        none
272  *
273  * DESCRIPTION:   Control-C handler. Abort running control method if any.
274  *
275  *****/

277 void ACPI_SYSTEM_XFACE
278 AeCtrlCHandler (
279     int                Sig)
280 {
282     signal (SIGINT, SIG_IGN);
283     SigintCount++;

285     AcpiOsPrintf ("Caught a ctrl-c (%u)\n\n", SigintCount);

287     if (AcpiGbl_MethodExecuting)
288     {
289         AcpiGbl_AbortMethod = TRUE;
290         signal (SIGINT, AeCtrlCHandler);

292         if (SigintCount < 10)
293         {
294             return;
295         }
296     }

298     (void) AcpiOsTerminate ();
299     exit (0);
300 }

303 /*****
304  *
305  * FUNCTION:      AeNotifyHandler(s)
306  *
307  * PARAMETERS:    Standard notify handler parameters
308  *
309  * RETURN:        Status
310  *
311  * DESCRIPTION:   Notify handlers for AcpiExec utility. Used by the ASL
312  *               test suite(s) to communicate errors and other information to
313  *               this utility via the Notify() operator. Tests notify handling
314  *               and multiple notify handler support.
315  *
316  *****/

318 static void
319 AeNotifyHandler1 (
320     ACPI_HANDLE         Device,

```

```

321     UINT32              Value,
322     void                *Context)
323 {
324     AeCommonNotifyHandler (Device, Value, 1);
325 }

327 static void
328 AeNotifyHandler2 (
329     ACPI_HANDLE         Device,
330     UINT32              Value,
331     void                *Context)
332 {
333     AeCommonNotifyHandler (Device, Value, 2);
334 }

336 static void
337 AeCommonNotifyHandler (
338     ACPI_HANDLE         Device,
339     UINT32              Value,
340     UINT32              HandlerId)
341 {
342     char                *Type;

345     Type = "Device";
346     if (Value <= ACPI_MAX_SYS_NOTIFY)
347     {
348         Type = "System";
349     }

351     switch (Value)
352     {
353     #if 0
354     case 0:

356         printf ("[AcpiExec] Method Error 0x%X: Results not equal\n", Value);
357         if (AcpiGbl_DebugFile)
358         {
359             AcpiOsPrintf ("[AcpiExec] Method Error: Results not equal\n");
360         }
361         break;

363     case 1:

365         printf ("[AcpiExec] Method Error: Incorrect numeric result\n");
366         if (AcpiGbl_DebugFile)
367         {
368             AcpiOsPrintf ("[AcpiExec] Method Error: Incorrect numeric result\n");
369         }
370         break;

372     case 2:

374         printf ("[AcpiExec] Method Error: An operand was overwritten\n");
375         if (AcpiGbl_DebugFile)
376         {
377             AcpiOsPrintf ("[AcpiExec] Method Error: An operand was overwritten\n");
378         }
379         break;

381     #endif

383     default:

385         printf ("[AcpiExec] Handler %u: Received a %s Notify on [%4.4s] %p Value
386             HandlerId, Type, AcpiUtGetNodeName (Device), Device, Value,

```

```

387     AcpiUtGetNotifyName (Value));
388     if (AcpiGbl_DebugFile)
389     {
390         AcpiOsPrintf ("[AcpiExec] Handler %u: Received a %s notify, Value 0x
391             HandlerId, Type, Value);
392     }
393
394     (void) AcpiEvaluateObject (Device, "_NOT", NULL, NULL);
395     break;
396 }
397 }

400 /*****
401 *
402 * FUNCTION:     AeSystemNotifyHandler
403 *
404 * PARAMETERS:  Standard notify handler parameters
405 *
406 * RETURN:      Status
407 *
408 * DESCRIPTION: System notify handler for AcpiExec utility. Used by the ASL
409 *               test suite(s) to communicate errors and other information to
410 *               this utility via the Notify() operator.
411 *
412 *****/
413
414 static void
415 AeSystemNotifyHandler (
416     ACPI_HANDLE      Device,
417     UINT32            Value,
418     void              *Context)
419 {
420
421     printf ("[AcpiExec] Global:   Received a System Notify on [%4.4s] %p Value
422         AcpiUtGetNodeName (Device), Device, Value,
423         AcpiUtGetNotifyName (Value));
424     if (AcpiGbl_DebugFile)
425     {
426         AcpiOsPrintf ("[AcpiExec] Global:   Received a System Notify, Value 0x%
427     }
428
429     (void) AcpiEvaluateObject (Device, "_NOT", NULL, NULL);
430 }

433 /*****
434 *
435 * FUNCTION:     AeDeviceNotifyHandler
436 *
437 * PARAMETERS:  Standard notify handler parameters
438 *
439 * RETURN:      Status
440 *
441 * DESCRIPTION: Device notify handler for AcpiExec utility. Used by the ASL
442 *               test suite(s) to communicate errors and other information to
443 *               this utility via the Notify() operator.
444 *
445 *****/
446
447 static void
448 AeDeviceNotifyHandler (
449     ACPI_HANDLE      Device,
450     UINT32            Value,
451     void              *Context)
452 {

```

```

454     printf ("[AcpiExec] Global:   Received a Device Notify on [%4.4s] %p Value
455         AcpiUtGetNodeName (Device), Device, Value,
456         AcpiUtGetNotifyName (Value));
457     if (AcpiGbl_DebugFile)
458     {
459         AcpiOsPrintf ("[AcpiExec] Global:   Received a Device Notify, Value 0x%
460     }
461
462     (void) AcpiEvaluateObject (Device, "_NOT", NULL, NULL);
463 }

466 /*****
467 *
468 * FUNCTION:     AeExceptionHandler
469 *
470 * PARAMETERS:  Standard exception handler parameters
471 *
472 * RETURN:      Status
473 *
474 * DESCRIPTION: System exception handler for AcpiExec utility.
475 *
476 *****/
477
478 static ACPI_STATUS
479 AeExceptionHandler (
480     ACPI_STATUS      AmlStatus,
481     ACPI_NAME        Name,
482     UINT16            Opcode,
483     UINT32            AmlOffset,
484     void              *Context)
485 {
486     ACPI_STATUS      NewAmlStatus = AmlStatus;
487     ACPI_STATUS      Status;
488     ACPI_BUFFER       ReturnObj;
489     ACPI_OBJECT_LIST  ArgList;
490     ACPI_OBJECT        Arg[3];
491     const char        *Exception;
492
493     Exception = AcpiFormatException (AmlStatus);
494     AcpiOsPrintf ("[AcpiExec] Exception %s during execution ", Exception);
495     if (Name)
496     {
497         AcpiOsPrintf ("of method [%4.4s]", (char *) &Name);
498     }
499     else
500     {
501         AcpiOsPrintf ("at module level (table load)");
502     }
503     AcpiOsPrintf (" Opcode [%s] @%X\n", AcpiPsGetOpcodeName (Opcode), AmlOffset);
504
505     /*
506     * Invoke the _ERR method if present
507     *
508     * Setup parameter object
509     */
510     ArgList.Count = 3;
511     ArgList.Pointer = Arg;
512
513     Arg[0].Type = ACPI_TYPE_INTEGER;
514     Arg[0].Integer.Value = AmlStatus;
515
516     Arg[1].Type = ACPI_TYPE_STRING;
517     Arg[1].String.Pointer = ACPI_CAST_PTR (char, Exception);

```



```

519     Arg[1].String.Length = ACPI_STRLen (Exception);

521     Arg[2].Type = ACPI_TYPE_INTEGER;
522     Arg[2].Integer.Value = AcpiOsGetThreadId();

524     /* Setup return buffer */

526     ReturnObj.Pointer = NULL;
527     ReturnObj.Length = ACPI_ALLOCATE_BUFFER;

529     Status = AcpiEvaluateObject (NULL, "\\_ERR", &ArgList, &ReturnObj);
530     if (ACPI_SUCCESS (Status))
531     {
532         if (ReturnObj.Pointer)
533         {
534             /* Override original status */

536             NewAmlStatus = (ACPI_STATUS)
537                 ((ACPI_OBJECT *) ReturnObj.Pointer)->Integer.Value;

539             /* Free a buffer created via ACPI_ALLOCATE_BUFFER */

541             AcpiOsFree (ReturnObj.Pointer);
542         }
543     }
544     else if (Status != AE_NOT_FOUND)
545     {
546         AcpiOsPrintf ("[AcpiExec] Could not execute _ERR method, %s\n",
547             AcpiFormatException (Status));
548     }

550     /* Global override */

552     if (AcpiGbl_IgnoreErrors)
553     {
554         NewAmlStatus = AE_OK;
555     }

557     if (NewAmlStatus != AmlStatus)
558     {
559         AcpiOsPrintf ("[AcpiExec] Exception override, new status %s\n",
560             AcpiFormatException (NewAmlStatus));
561     }

563     return (NewAmlStatus);
564 }

567 /*****
568  *
569  * FUNCTION:     AeTableHandler
570  *
571  * PARAMETERS:  Table handler
572  *
573  * RETURN:      Status
574  *
575  * DESCRIPTION: System table handler for AcpiExec utility.
576  *
577  *****/

579 static ACPI_STATUS
580 AeTableHandler (
581     UINT32          Event,
582     void            *Table,
583     void            *Context)
584 {

```

```

585 #if (!ACPI_REDUCED_HARDWARE)
586     ACPI_STATUS      Status;
587 #endif /* !ACPI_REDUCED_HARDWARE */

590     if (Event > ACPI_NUM_TABLE_EVENTS)
591     {
592         Event = ACPI_NUM_TABLE_EVENTS;
593     }

595 #if (!ACPI_REDUCED_HARDWARE)
596     /* Enable any GPEs associated with newly-loaded GPE methods */

598     Status = AcpiUpdateAllGpes ();
599     AE_CHECK_OK (AcpiUpdateAllGpes, Status);

601     printf ("[AcpiExec] Table Event %s, [%4.4s] %p\n",
602         TableEvents[Event], ((ACPI_TABLE_HEADER *) Table)->Signature, Table);
603 #endif /* !ACPI_REDUCED_HARDWARE */

605     return (AE_OK);
606 }

609 /*****
610  *
611  * FUNCTION:     AeGpeHandler
612  *
613  * DESCRIPTION:  Common GPE handler for acpiexec
614  *
615  *****/

617 UINT32
618 AeGpeHandler (
619     ACPI_HANDLE      GpeDevice,
620     UINT32           GpeNumber,
621     void             *Context)
622 {
623     ACPI_NAMESPACE_NODE *DeviceNode = (ACPI_NAMESPACE_NODE *) GpeDevice;

626     AcpiOsPrintf ("[AcpiExec] GPE Handler received GPE%02X (GPE block %4.4s)\n",
627         GpeNumber, GpeDevice ? DeviceNode->Name.Ascii : "FADT");

629     return (ACPI_REENABLE_GPE);
630 }

633 /*****
634  *
635  * FUNCTION:     AeGlobalEventHandler
636  *
637  * DESCRIPTION:  Global GPE/Fixed event handler
638  *
639  *****/

641 void
642 AeGlobalEventHandler (
643     UINT32          Type,
644     ACPI_HANDLE      Device,
645     UINT32          EventNumber,
646     void            *Context)
647 {
648     char            *TypeName;

```

```

651     switch (Type)
652     {
653     case ACPI_EVENT_TYPE_GPE:

655         TypeName = "GPE";
656         break;

658     case ACPI_EVENT_TYPE_FIXED:

660         TypeName = "FixedEvent";
661         break;

663     default:

665         TypeName = "UNKNOWN";
666         break;
667     }

669     AcpiOsPrintf ("[AcpiExec] Global Event Handler received: Type %s Number %.2X\n",
670                 TypeName, EventNumber, Device);
671 }

674 /*****
675  *
676  * FUNCTION:      AeAttachedDataHandler
677  *
678  * DESCRIPTION:  Handler for deletion of nodes with attached data (attached via
679  *               AcpiAttachData)
680  *
681  *****/

683 static void
684 AeAttachedDataHandler (
685     ACPI_HANDLE      Object,
686     void              *Data)
687 {
688     ACPI_NAMESPACE_NODE *Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Data);

691     AcpiOsPrintf ("Received an attached data deletion (1) on %4.4s\n",
692                 Node->Name.Ascii);
693 }

696 /*****
697  *
698  * FUNCTION:      AeAttachedDataHandler2
699  *
700  * DESCRIPTION:  Handler for deletion of nodes with attached data (attached via
701  *               AcpiAttachData)
702  *
703  *****/

705 static void
706 AeAttachedDataHandler2 (
707     ACPI_HANDLE      Object,
708     void              *Data)
709 {
710     ACPI_NAMESPACE_NODE *Node = ACPI_CAST_PTR (ACPI_NAMESPACE_NODE, Data);

713     AcpiOsPrintf ("Received an attached data deletion (2) on %4.4s\n",
714                 Node->Name.Ascii);
715 }

```

```

718 /*****
719  *
720  * FUNCTION:      AeInterfaceHandler
721  *
722  * DESCRIPTION:  Handler for _OSI invocations
723  *
724  *****/

726 static UINT32
727 AeInterfaceHandler (
728     ACPI_STRING      InterfaceName,
729     UINT32            Supported)
730 {
731     ACPI_FUNCTION_NAME (AeInterfaceHandler);

734     ACPI_DEBUG_PRINT ((ACPI_DB_INFO,
735                 "Received _OSI (%s), is %ssupported\n",
736                 InterfaceName, Supported == 0 ? "not " : ""));

738     return (Supported);
739 }

742 #if (!ACPI_REDUCED_HARDWARE)
743 /*****
744  *
745  * FUNCTION:      AeEventHandler, AeSciHandler
746  *
747  * DESCRIPTION:  Handler for Fixed Events and SCIs
748  *
749  *****/

751 static UINT32
752 AeEventHandler (
753     void              *Context)
754 {
755     return (0);
756 }

758 static UINT32
759 AeSciHandler (
760     void              *Context)
761 {
763     AcpiOsPrintf ("[AcpiExec] Received an SCI at handler\n");
764     return (0);
765 }

767 #endif /* !ACPI_REDUCED_HARDWARE */

770 /*****
771  *
772  * FUNCTION:      AeRegionInit
773  *
774  * PARAMETERS:  None
775  *
776  * RETURN:      Status
777  *
778  * DESCRIPTION:  Opreion init function.
779  *
780  *****/

782 static ACPI_STATUS

```

```

783 AeRegionInit (
784     ACPI_HANDLE      RegionHandle,
785     UINT32            Function,
786     void              *HandlerContext,
787     void              **RegionContext)
788 {
789     if (Function == ACPI_REGION_DEACTIVATE)
790     {
791         *RegionContext = NULL;
792     }
793     else
794     {
795         *RegionContext = RegionHandle;
796     }
797 }
798
799 return (AE_OK);
800 }
801
802 /*****
803 *
804 * FUNCTION:      AeInstallSciHandler
805 * PARAMETERS:    None
806 * RETURN:        Status
807 * DESCRIPTION:   Install handler for SCIs. Exercise the code by doing an
808 *                install/remove/install.
809 * *****/
810
811 static ACPI_STATUS
812 AeInstallSciHandler (
813     void)
814 {
815     ACPI_STATUS      Status;
816
817     Status = AcpiInstallSciHandler (AeSciHandler, &AeMyContext);
818     if (ACPI_FAILURE (Status))
819     {
820         ACPI_EXCEPTION ((AE_INFO, Status,
821             "Could not install an SCI handler (1)"));
822     }
823
824     Status = AcpiRemoveSciHandler (AeSciHandler);
825     if (ACPI_FAILURE (Status))
826     {
827         ACPI_EXCEPTION ((AE_INFO, Status,
828             "Could not remove an SCI handler"));
829     }
830
831     Status = AcpiInstallSciHandler (AeSciHandler, &AeMyContext);
832     if (ACPI_FAILURE (Status))
833     {
834         ACPI_EXCEPTION ((AE_INFO, Status,
835             "Could not install an SCI handler (2)"));
836     }
837
838     return (Status);
839 }
840
841 /*****

```

```

842 *
843 * FUNCTION:      AeInstallDeviceHandlers, AeInstallEcHandler,
844 *                AeInstallPciHandler
845 *
846 * PARAMETERS:    ACPI_WALK_NAMESPACE callback
847 *
848 * RETURN:        Status
849 *
850 * DESCRIPTION:   Walk entire namespace, install a handler for every EC
851 *                and PCI device found.
852 * *****/
853
854 static ACPI_STATUS
855 AeInstallEcHandler (
856     ACPI_HANDLE      ObjHandle,
857     UINT32            Level,
858     void              *Context,
859     void              **ReturnValue)
860 {
861     ACPI_STATUS      Status;
862
863     /* Install the handler for this EC device */
864
865     Status = AcpiInstallAddressSpaceHandler (ObjHandle, ACPI_ADR_SPACE_EC,
866         AeRegionHandler, AeRegionInit, &AeMyContext);
867     if (ACPI_FAILURE (Status))
868     {
869         ACPI_EXCEPTION ((AE_INFO, Status,
870             "Could not install an OpRegion handler for EC device (%p)",
871             ObjHandle));
872     }
873
874     return (Status);
875 }
876
877 static ACPI_STATUS
878 AeInstallPciHandler (
879     ACPI_HANDLE      ObjHandle,
880     UINT32            Level,
881     void              *Context,
882     void              **ReturnValue)
883 {
884     ACPI_STATUS      Status;
885
886     /* Install memory and I/O handlers for the PCI device */
887
888     Status = AcpiInstallAddressSpaceHandler (ObjHandle, ACPI_ADR_SPACE_SYSTEM_IO,
889         AeRegionHandler, AeRegionInit, &AeMyContext);
890     if (ACPI_FAILURE (Status))
891     {
892         ACPI_EXCEPTION ((AE_INFO, Status,
893             "Could not install an OpRegion handler for PCI device (%p)",
894             ObjHandle));
895     }
896
897     Status = AcpiInstallAddressSpaceHandler (ObjHandle, ACPI_ADR_SPACE_SYSTEM_ME,
898         AeRegionHandler, AeRegionInit, &AeMyContext);
899     if (ACPI_FAILURE (Status))
900     {
901         ACPI_EXCEPTION ((AE_INFO, Status,
902             "Could not install an OpRegion handler for PCI device (%p)",
903             ObjHandle));
904     }
905 }

```

```

916     return (AE_CTRL_TERMINATE);
917 }

919 static ACPI_STATUS
920 AeInstallDeviceHandlers (
921     void)
922 {
923
924     /* Find all Embedded Controller devices */
925
926     AcpiGetDevices ("PNP0C09", AeInstallEcHandler, NULL, NULL);
927
928     /* Install a PCI handler */
929
930     AcpiGetDevices ("PNP0A08", AeInstallPciHandler, NULL, NULL);
931     return (AE_OK);
932 }

935 /*****
936  *
937  * FUNCTION:    AeInstallLateHandlers
938  *
939  * PARAMETERS:  None
940  *
941  * RETURN:      Status
942  *
943  * DESCRIPTION: Install handlers for the AcpiExec utility.
944  *
945  *****/

947 ACPI_STATUS
948 AeInstallLateHandlers (
949     void)
950 {
951     ACPI_STATUS      Status;
952     UINT32            i;

955 #if (!ACPI_REDUCED_HARDWARE)
956     if (!AcpiGbl_ReducedHardware)
957     {
958         /* Install a user SCI handler */

960         Status = AeInstallSciHandler ();
961         AE_CHECK_OK (AeInstallSciHandler, Status);

963         /* Install some fixed event handlers */

965         Status = AcpiInstallFixedEventHandler (ACPI_EVENT_GLOBAL, AeEventHandler);
966         AE_CHECK_OK (AcpiInstallFixedEventHandler, Status);

968         Status = AcpiInstallFixedEventHandler (ACPI_EVENT_RTC, AeEventHandler, N
969         AE_CHECK_OK (AcpiInstallFixedEventHandler, Status);
970     }
971 #endif /* !ACPI_REDUCED_HARDWARE */

973     AeMyContext.Connection = NULL;
974     AeMyContext.AccessLength = 0xA5;

976     /*
977     * We will install a handler for each EC device, directly under the EC
978     * device definition. This is unlike the other handlers which we install
979     * at the root node. Also install memory and I/O handlers at any PCI
980     * devices.

```

```

981     */
982     AeInstallDeviceHandlers ();

984     /*
985     * Install handlers for some of the "device driver" address spaces
986     * such as SMBus, etc.
987     */
988     for (i = 0; i < ACPI_ARRAY_LENGTH (SpaceIdList); i++)
989     {
990         /* Install handler at the root object */

992         Status = AcpiInstallAddressSpaceHandler (ACPI_ROOT_OBJECT,
993         SpaceIdList[i], AeRegionHandler,
994         AeRegionInit, &AeMyContext);
995         if (ACPI_FAILURE (Status))
996         {
997             ACPI_EXCEPTION ((AE_INFO, Status,
998             "Could not install an OpRegion handler for %s space(%u)",
999             AcpiUtGetRegionName((UINT8) SpaceIdList[i]), SpaceIdList[i]));
1000             return (Status);
1001         }
1002     }

1004     return (AE_OK);
1005 }

1008 /*****
1009  *
1010  * FUNCTION:    AeInstallEarlyHandlers
1011  *
1012  * PARAMETERS:  None
1013  *
1014  * RETURN:      Status
1015  *
1016  * DESCRIPTION: Install handlers for the AcpiExec utility.
1017  *
1018  * Notes:       Don't install handler for PCI_Config, we want to use the
1019  *              default handler to exercise that code.
1020  *
1021  *****/

1023 ACPI_STATUS
1024 AeInstallEarlyHandlers (
1025     void)
1026 {
1027     ACPI_STATUS      Status;
1028     UINT32            i;
1029     ACPI_HANDLE       Handle;

1032     ACPI_FUNCTION_ENTRY ();

1035     Status = AcpiInstallInterfaceHandler (AeInterfaceHandler);
1036     if (ACPI_FAILURE (Status))
1037     {
1038         printf ("Could not install interface handler, %s\n",
1039         AcpiFormatException (Status));
1040     }

1042     Status = AcpiInstallTableHandler (AeTableHandler, NULL);
1043     if (ACPI_FAILURE (Status))
1044     {
1045         printf ("Could not install table handler, %s\n",
1046         AcpiFormatException (Status));

```

```

1047 }
1049 Status = AcpiInstallExceptionHandler (AeExceptionHandler);
1050 if (ACPI_FAILURE (Status))
1051 {
1052     printf ("Could not install exception handler, %s\n",
1053         AcpiFormatException (Status));
1054 }
1056 /* Install global notify handlers */
1058 Status = AcpiInstallNotifyHandler (ACPI_ROOT_OBJECT, ACPI_SYSTEM_NOTIFY,
1059     AeSystemNotifyHandler, NULL);
1060 if (ACPI_FAILURE (Status))
1061 {
1062     printf ("Could not install a global system notify handler, %s\n",
1063         AcpiFormatException (Status));
1064 }
1066 Status = AcpiInstallNotifyHandler (ACPI_ROOT_OBJECT, ACPI_DEVICE_NOTIFY,
1067     AeDeviceNotifyHandler, NULL);
1068 if (ACPI_FAILURE (Status))
1069 {
1070     printf ("Could not install a global notify handler, %s\n",
1071         AcpiFormatException (Status));
1072 }
1074 Status = AcpiGetHandle (NULL, "\\_SB", &Handle);
1075 if (ACPI_SUCCESS (Status))
1076 {
1077     Status = AcpiInstallNotifyHandler (Handle, ACPI_SYSTEM_NOTIFY,
1078         AeNotifyHandler1, NULL);
1079     if (ACPI_FAILURE (Status))
1080     {
1081         printf ("Could not install a notify handler, %s\n",
1082             AcpiFormatException (Status));
1083     }
1085     Status = AcpiRemoveNotifyHandler (Handle, ACPI_SYSTEM_NOTIFY,
1086         AeNotifyHandler1);
1087     if (ACPI_FAILURE (Status))
1088     {
1089         printf ("Could not remove a notify handler, %s\n",
1090             AcpiFormatException (Status));
1091     }
1093     Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
1094         AeNotifyHandler1, NULL);
1095     AE_CHECK_OK (AcpiInstallNotifyHandler, Status);
1097     Status = AcpiRemoveNotifyHandler (Handle, ACPI_ALL_NOTIFY,
1098         AeNotifyHandler1);
1099     AE_CHECK_OK (AcpiRemoveNotifyHandler, Status);
1101 #if 0
1102     Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
1103         AeNotifyHandler1, NULL);
1104     if (ACPI_FAILURE (Status))
1105     {
1106         printf ("Could not install a notify handler, %s\n",
1107             AcpiFormatException (Status));
1108     }
1109 #endif
1111 /* Install two handlers for _SB_ */

```

```

1113 Status = AcpiInstallNotifyHandler (Handle, ACPI_SYSTEM_NOTIFY,
1114     AeNotifyHandler1, ACPI_CAST_PTR (void, 0x01234567));
1116 Status = AcpiInstallNotifyHandler (Handle, ACPI_SYSTEM_NOTIFY,
1117     AeNotifyHandler2, ACPI_CAST_PTR (void, 0x89ABCDEF));
1119 /* Attempt duplicate handler installation, should fail */
1121 Status = AcpiInstallNotifyHandler (Handle, ACPI_SYSTEM_NOTIFY,
1122     AeNotifyHandler1, ACPI_CAST_PTR (void, 0x77777777));
1124 Status = AcpiAttachData (Handle, AeAttachedDataHandler, Handle);
1125 AE_CHECK_OK (AcpiAttachData, Status);
1127 Status = AcpiDetachData (Handle, AeAttachedDataHandler);
1128 AE_CHECK_OK (AcpiDetachData, Status);
1130 Status = AcpiAttachData (Handle, AeAttachedDataHandler, Handle);
1131 AE_CHECK_OK (AcpiAttachData, Status);
1133 /* Test support for multiple attaches */
1135 Status = AcpiAttachData (Handle, AeAttachedDataHandler2, Handle);
1136 AE_CHECK_OK (AcpiAttachData, Status);
1137 }
1138 else
1139 {
1140     printf ("No _SB found, %s\n", AcpiFormatException (Status));
1141 }
1144 Status = AcpiGetHandle (NULL, "\\_TZ.TZ1", &Handle);
1145 if (ACPI_SUCCESS (Status))
1146 {
1147     Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
1148         AeNotifyHandler1, ACPI_CAST_PTR (void, 0x01234567));
1150     Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
1151         AeNotifyHandler2, ACPI_CAST_PTR (void, 0x89ABCDEF));
1153     Status = AcpiRemoveNotifyHandler (Handle, ACPI_ALL_NOTIFY,
1154         AeNotifyHandler1);
1155     Status = AcpiRemoveNotifyHandler (Handle, ACPI_ALL_NOTIFY,
1156         AeNotifyHandler2);
1158     Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
1159         AeNotifyHandler2, ACPI_CAST_PTR (void, 0x89ABCDEF));
1161     Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
1162         AeNotifyHandler1, ACPI_CAST_PTR (void, 0x01234567));
1163 }
1165 Status = AcpiGetHandle (NULL, "\\_PR.CPU0", &Handle);
1166 if (ACPI_SUCCESS (Status))
1167 {
1168     Status = AcpiInstallNotifyHandler (Handle, ACPI_ALL_NOTIFY,
1169         AeNotifyHandler1, ACPI_CAST_PTR (void, 0x01234567));
1171     Status = AcpiInstallNotifyHandler (Handle, ACPI_SYSTEM_NOTIFY,
1172         AeNotifyHandler2, ACPI_CAST_PTR (void, 0x89ABCDEF));
1173 }
1175 /*
1176  * Install handlers that will override the default handlers for some of
1177  * the space IDs.
1178  */

```

```

1179     for (i = 0; i < ACPI_ARRAY_LENGTH (DefaultSpaceIdList); i++)
1180     {
1181         /* Install handler at the root object */
1182
1183         Status = AcpiInstallAddressSpaceHandler (ACPI_ROOT_OBJECT,
1184             DefaultSpaceIdList[i], AeRegionHandler,
1185             AeRegionInit, &AeMyContext);
1186         if (ACPI_FAILURE (Status))
1187         {
1188             ACPI_EXCEPTION ((AE_INFO, Status,
1189                 "Could not install a default OpRegion handler for %s space(%u)",
1190                 AcpiUtGetRegionName ((UINT8) DefaultSpaceIdList[i]),
1191                 DefaultSpaceIdList[i]));
1192             return (Status);
1193         }
1194     }
1195
1196     /*
1197     * Initialize the global Region Handler space
1198     * MCW 3/23/00
1199     */
1200     AeRegions.NumberOfRegions = 0;
1201     AeRegions.RegionList = NULL;
1202     return (Status);
1203 }

```

```

1206 /*****
1207 *
1208 * FUNCTION:      AeRegionHandler
1209 *
1210 * PARAMETERS:    Standard region handler parameters
1211 *
1212 * RETURN:        Status
1213 *
1214 * DESCRIPTION:   Test handler - Handles some dummy regions via memory that can
1215 *                be manipulated in Ring 3. Simulates actual reads and writes.
1216 *
1217 *****/

```

```

1219 ACPI_STATUS
1220 AeRegionHandler (
1221     UINT32             Function,
1222     ACPI_PHYSICAL_ADDRESS Address,
1223     UINT32             BitWidth,
1224     UINT64             *Value,
1225     void               *HandlerContext,
1226     void               *RegionContext)
1227 {
1228
1229     ACPI_OPERAND_OBJECT *RegionObject = ACPI_CAST_PTR (ACPI_OPERAND_OBJECT,
1230     UINT8               *Buffer = ACPI_CAST_PTR (UINT8, Value);
1231     UINT8               *OldBuffer;
1232     UINT8               *NewBuffer;
1233     ACPI_PHYSICAL_ADDRESS BaseAddress;
1234     ACPI_PHYSICAL_ADDRESS BaseAddressEnd;
1235     ACPI_PHYSICAL_ADDRESS RegionAddress;
1236     ACPI_PHYSICAL_ADDRESS RegionAddressEnd;
1237     ACPI_SIZE           Length;
1238     BOOLEAN             BufferExists;
1239     BOOLEAN             BufferResize;
1240     AE_REGION           *RegionElement;
1241     void               *BufferValue;
1242     ACPI_STATUS         Status;
1243     UINT32             ByteWidth;
1244     UINT32             RegionLength;

```

```

1245     UINT32             i;
1246     UINT8             SpaceId;
1247     ACPI_CONNECTION_INFO *MyContext;
1248     UINT32             Value1;
1249     UINT32             Value2;
1250     ACPI_RESOURCE       *Resource;
1251
1252     ACPI_FUNCTION_NAME (AeRegionHandler);
1253
1254     /*
1255     * If the object is not a region, simply return
1256     */
1257     if (RegionObject->Region.Type != ACPI_TYPE_REGION)
1258     {
1259         return (AE_OK);
1260     }
1261
1262     /* Check that we actually got back our context parameter */
1263
1264     if (HandlerContext != &AeMyContext)
1265     {
1266         printf ("Region handler received incorrect context %p, should be %p\n",
1267             HandlerContext, &AeMyContext);
1268     }
1269
1270     MyContext = ACPI_CAST_PTR (ACPI_CONNECTION_INFO, HandlerContext);
1271
1272     /*
1273     * Find the region's address space and length before searching
1274     * the linked list.
1275     */
1276     BaseAddress = RegionObject->Region.Address;
1277     Length = (ACPI_SIZE) RegionObject->Region.Length;
1278     SpaceId = RegionObject->Region.SpaceId;
1279
1280     ACPI_DEBUG_PRINT ((ACPI_DB_OPREGION, "Operation Region request on %s at 0x%X
1281     AcpiUtGetRegionName (RegionObject->Region.SpaceId),
1282     (UINT32) Address));
1283
1284     /*
1285     * Region support can be disabled with the -do option.
1286     * We use this to support dynamically loaded tables where we pass a valid
1287     * address to the AML.
1288     */
1289     if (AcpiGbl_DbOpt_NoRegionSupport)
1290     {
1291         BufferValue = ACPI_TO_POINTER (Address);
1292         ByteWidth = (BitWidth / 8);
1293
1294         if (BitWidth % 8)
1295         {
1296             ByteWidth += 1;
1297         }
1298         goto DoFunction;
1299     }
1300
1301     switch (SpaceId)
1302     {
1303     case ACPI_ADR_SPACE_SYSTEM_IO:
1304         /*
1305         * For I/O space, exercise the port validation
1306         * Note: ReadPort currently always returns all ones, length=BitLength
1307         */
1308         switch (Function & ACPI_IO_MASK)
1309         {

```

```

1311     case ACPI_READ:
1312
1313         if (BitWidth == 64)
1314         {
1315             /* Split the 64-bit request into two 32-bit requests */
1316
1317             Status = AcpiHwReadPort (Address, &Value1, 32);
1318             AE_CHECK_OK (AcpiHwReadPort, Status);
1319             Status = AcpiHwReadPort (Address+4, &Value2, 32);
1320             AE_CHECK_OK (AcpiHwReadPort, Status);
1321
1322             *Value = Value1 | ((UINT64) Value2 << 32);
1323         }
1324         else
1325         {
1326             Status = AcpiHwReadPort (Address, &Value1, BitWidth);
1327             AE_CHECK_OK (AcpiHwReadPort, Status);
1328             *Value = (UINT64) Value1;
1329         }
1330         break;
1331
1332     case ACPI_WRITE:
1333
1334         if (BitWidth == 64)
1335         {
1336             /* Split the 64-bit request into two 32-bit requests */
1337
1338             Status = AcpiHwWritePort (Address, ACPI_LODWORD (*Value), 32);
1339             AE_CHECK_OK (AcpiHwWritePort, Status);
1340             Status = AcpiHwWritePort (Address+4, ACPI_HIWORD (*Value), 32);
1341             AE_CHECK_OK (AcpiHwWritePort, Status);
1342         }
1343         else
1344         {
1345             Status = AcpiHwWritePort (Address, (UINT32) *Value, BitWidth);
1346             AE_CHECK_OK (AcpiHwWritePort, Status);
1347         }
1348         break;
1349
1350     default:
1351
1352         Status = AE_BAD_PARAMETER;
1353         break;
1354     }
1355
1356     if (ACPI_FAILURE (Status))
1357     {
1358         return (Status);
1359     }
1360
1361     /* Now go ahead and simulate the hardware */
1362     break;
1363
1364     /*
1365     * SMBus and GenericSerialBus support the various bidirectional
1366     * protocols.
1367     */
1368     case ACPI_ADR_SPACE_SMBUS:
1369     case ACPI_ADR_SPACE_GSBUS: /* ACPI 5.0 */
1370
1371         Length = 0;
1372
1373         switch (Function & ACPI_IO_MASK)
1374         {
1375             case ACPI_READ:

```

```

1377         switch (Function >> 16)
1378         {
1379             case AML_FIELD_ATTRIB_QUICK:
1380             case AML_FIELD_ATTRIB_SEND_RCV:
1381             case AML_FIELD_ATTRIB_BYTE:
1382
1383                 Length = 1;
1384                 break;
1385
1386             case AML_FIELD_ATTRIB_WORD:
1387             case AML_FIELD_ATTRIB_WORD_CALL:
1388
1389                 Length = 2;
1390                 break;
1391
1392             case AML_FIELD_ATTRIB_BLOCK:
1393             case AML_FIELD_ATTRIB_BLOCK_CALL:
1394
1395                 Length = 32;
1396                 break;
1397
1398             case AML_FIELD_ATTRIB_MULTIBYTE:
1399             case AML_FIELD_ATTRIB_RAW_BYTES:
1400             case AML_FIELD_ATTRIB_RAW_PROCESS:
1401
1402                 /* (-2) for status/length */
1403                 Length = MyContext->AccessLength - 2;
1404                 break;
1405
1406             default:
1407
1408                 break;
1409         }
1410         break;
1411
1412     case ACPI_WRITE:
1413
1414         switch (Function >> 16)
1415         {
1416             case AML_FIELD_ATTRIB_QUICK:
1417             case AML_FIELD_ATTRIB_SEND_RCV:
1418             case AML_FIELD_ATTRIB_BYTE:
1419             case AML_FIELD_ATTRIB_WORD:
1420             case AML_FIELD_ATTRIB_BLOCK:
1421
1422                 Length = 0;
1423                 break;
1424
1425             case AML_FIELD_ATTRIB_WORD_CALL:
1426                 Length = 2;
1427                 break;
1428
1429             case AML_FIELD_ATTRIB_BLOCK_CALL:
1430                 Length = 32;
1431                 break;
1432
1433             case AML_FIELD_ATTRIB_MULTIBYTE:
1434             case AML_FIELD_ATTRIB_RAW_BYTES:
1435             case AML_FIELD_ATTRIB_RAW_PROCESS:
1436
1437                 /* (-2) for status/length */
1438                 Length = MyContext->AccessLength - 2;
1439                 break;
1440
1441             default:

```

```

1443         break;
1444     }
1445     break;
1447 default:
1449     break;
1450 }

1452 if (AcpiGbl_DisplayRegionAccess)
1453 {
1454     AcpiOsPrintf ("AcpiExec: %s "
1455         "%s: Attr %X Addr %4X BaseAddr %4X Len %2X Width %X BufLen %X\n",
1456         AcpiUtGetRegionName (SpaceId),
1457         (Function & ACPI_IO_MASK) ? "Write" : "Read",
1458         (UINT32) (Function >> 16),
1459         (UINT32) Address, (UINT32) BaseAddress,
1460         Length, BitWidth, Buffer[1]);

1462     /* GenericSerialBus has a Connection() parameter */

1464     if (SpaceId == ACPI_ADR_SPACE_GSBUS)
1465     {
1466         Status = AcpiBufferToResource (MyContext->Connection,
1467             MyContext->Length, &Resource);

1469         AcpiOsPrintf (" [AccLen %2X Conn %p]",
1470             MyContext->AccessLength, MyContext->Connection);
1471     }
1472     AcpiOsPrintf ("\n");
1473 }

1475 /* Setup the return buffer. Note: ASLTS depends on these fill values */

1477 for (i = 0; i < Length; i++)
1478 {
1479     Buffer[i+2] = (UINT8) (0xA0 + i);
1480 }

1482 Buffer[0] = 0x7A;
1483 Buffer[1] = (UINT8) Length;
1484 return (AE_OK);

1487 case ACPI_ADR_SPACE_IPMI: /* ACPI 4.0 */

1489     if (AcpiGbl_DisplayRegionAccess)
1490     {
1491         AcpiOsPrintf ("AcpiExec: IPMI "
1492             "%s: Attr %X Addr %4X BaseAddr %4X Len %2X Width %X BufLen %X\n",
1493             (Function & ACPI_IO_MASK) ? "Write" : "Read",
1494             (UINT32) (Function >> 16), (UINT32) Address, (UINT32) BaseAddress,
1495             Length, BitWidth, Buffer[1]);
1496     }

1498     /*
1499     * Regardless of a READ or WRITE, this handler is passed a 66-byte
1500     * buffer in which to return the IPMI status/length/data.
1501     *
1502     * Return some example data to show use of the bidirectional buffer
1503     */
1504     Buffer[0] = 0;          /* Status byte */
1505     Buffer[1] = 64;         /* Return buffer data length */
1506     Buffer[2] = 0;         /* Completion code */
1507     Buffer[3] = 0;         /* Reserved */

```

```

1509     /*
1510     * Fill the 66-byte buffer with the return data.
1511     * Note: ASLTS depends on these fill values.
1512     */
1513     for (i = 4; i < 66; i++)
1514     {
1515         Buffer[i] = (UINT8) (i);
1516     }
1517     return (AE_OK);

1519 default:
1520     break;
1521 }

1523 /*
1524 * Search through the linked list for this region's buffer
1525 */
1526 BufferExists = FALSE;
1527 BufferResize = FALSE;
1528 RegionElement = AeRegions.RegionList;

1530 if (AeRegions.NumberOfRegions)
1531 {
1532     BaseAddressEnd = BaseAddress + Length - 1;
1533     while (!BufferExists && RegionElement)
1534     {
1535         RegionAddress = RegionElement->Address;
1536         RegionAddressEnd = RegionElement->Address + RegionElement->Length -
1537             RegionLength = RegionElement->Length;

1539         /*
1540         * Overlapping Region Support
1541         *
1542         * While searching through the region buffer list, determine if an
1543         * overlap exists between the requested buffer space and the current
1544         * RegionElement space. If there is an overlap then replace the old
1545         * buffer with a new buffer of increased size before continuing to
1546         * do the read or write
1547         */
1548         if (RegionElement->SpaceId != SpaceId ||
1549             BaseAddressEnd < RegionAddress ||
1550             BaseAddress > RegionAddressEnd)
1551         {
1552             /*
1553             * Requested buffer is outside of the current RegionElement
1554             * bounds
1555             */
1556             RegionElement = RegionElement->NextRegion;
1557         }
1558         else
1559         {
1560             /*
1561             * Some amount of buffer space sharing exists. There are 4 cases
1562             * to consider:
1563             *
1564             * 1. Right overlap
1565             * 2. Left overlap
1566             * 3. Left and right overlap
1567             * 4. Fully contained - no resizing required
1568             */
1569             BufferExists = TRUE;

1571             if ((BaseAddress >= RegionAddress) &&
1572                 (BaseAddress <= RegionAddressEnd) &&
1573                 (BaseAddressEnd > RegionAddressEnd))
1574             {

```



```

1575      /* Right overlap */
1577      RegionElement->Length = BaseAddress -
1578      RegionAddress + Length;
1579      BufferResize = TRUE;
1580  }
1582  else if ((BaseAddressEnd >= RegionAddress) &&
1583      (BaseAddressEnd <= RegionAddressEnd) &&
1584      (BaseAddress < RegionAddress))
1585  {
1586      /* Left overlap */
1588      RegionElement->Address = BaseAddress;
1589      RegionElement->Length = RegionAddress -
1590      BaseAddress + RegionElement->Length;
1591      BufferResize = TRUE;
1592  }
1594  else if ((BaseAddress < RegionAddress) &&
1595      (BaseAddressEnd > RegionAddressEnd))
1596  {
1597      /* Left and right overlap */
1599      RegionElement->Address = BaseAddress;
1600      RegionElement->Length = Length;
1601      BufferResize = TRUE;
1602  }
1604  /*
1605   * only remaining case is fully contained for which we don't
1606   * need to do anything
1607   */
1608  if (BufferResize)
1609  {
1610      NewBuffer = AcpiOsAllocate (RegionElement->Length);
1611      if (!NewBuffer)
1612      {
1613          return (AE_NO_MEMORY);
1614      }
1616      OldBuffer = RegionElement->Buffer;
1617      RegionElement->Buffer = NewBuffer;
1618      NewBuffer = NULL;
1620      /* Initialize the region with the default fill value */
1622      ACPI_MEMSET (RegionElement->Buffer,
1623      AcpiGbl_RegionFillValue, RegionElement->Length);
1625      /*
1626       * Get BufferValue to point (within the new buffer) to the
1627       * base address of the old buffer
1628       */
1629      BufferValue = (UINT8 *) RegionElement->Buffer +
1630      (UINT64) RegionAddress -
1631      (UINT64) RegionElement->Address;
1633      /*
1634       * Copy the old buffer to its same location within the new
1635       * buffer
1636       */
1637      ACPI_MEMCPY (BufferValue, OldBuffer, RegionLength);
1638      AcpiOsFree (OldBuffer);
1639  }
1640  }

```

```

1641      }
1642  }
1644  /*
1645   * If the Region buffer does not exist, create it now
1646   */
1647  if (!BufferExists)
1648  {
1649      /* Do the memory allocations first */
1651      RegionElement = AcpiOsAllocate (sizeof (AE_REGION));
1652      if (!RegionElement)
1653      {
1654          return (AE_NO_MEMORY);
1655      }
1657      RegionElement->Buffer = AcpiOsAllocate (Length);
1658      if (!RegionElement->Buffer)
1659      {
1660          AcpiOsFree (RegionElement);
1661          return (AE_NO_MEMORY);
1662      }
1664      /* Initialize the region with the default fill value */
1666      ACPI_MEMSET (RegionElement->Buffer, AcpiGbl_RegionFillValue, Length);
1668      RegionElement->Address      = BaseAddress;
1669      RegionElement->Length      = Length;
1670      RegionElement->SpaceId     = SpaceId;
1671      RegionElement->NextRegion  = NULL;
1673      /*
1674       * Increment the number of regions and put this one
1675       * at the head of the list as it will probably get accessed
1676       * more often anyway.
1677       */
1678      AeRegions.NumberOfRegions += 1;
1680      if (AeRegions.RegionList)
1681      {
1682          RegionElement->NextRegion = AeRegions.RegionList;
1683      }
1685      AeRegions.RegionList = RegionElement;
1686  }
1688  /* Calculate the size of the memory copy */
1690  ByteWidth = (BitWidth / 8);
1692  if (BitWidth % 8)
1693  {
1694      ByteWidth += 1;
1695  }
1697  /*
1698   * The buffer exists and is pointed to by RegionElement.
1699   * We now need to verify the request is valid and perform the operation.
1700   *
1701   * NOTE: RegionElement->Length is in bytes, therefore it we compare against
1702   * ByteWidth (see above)
1703   */
1704  if (((UINT64) Address + ByteWidth) >
1705      ((UINT64) (RegionElement->Address) + RegionElement->Length))
1706  {

```

```

1707     ACPI_WARNING ((AE_INFO,
1708         "Request on [%4.4s] is beyond region limit Req=0x%X+0x%X, Base=0x%X,
1709         (RegionObject->Region.Node)->Name.Ascii, (UINT32) Address,
1710         ByteWidth, (UINT32)(RegionElement->Address),
1711         RegionElement->Length));

1713     return (AE_AML_REGION_LIMIT);
1714 }

1716 /*
1717  * Get BufferValue to point to the "address" in the buffer
1718  */
1719 BufferValue = ((UINT8 *) RegionElement->Buffer +
1720     ((UINT64) Address - (UINT64) RegionElement->Address));

1722 DoFunction:
1723 /*
1724  * Perform a read or write to the buffer space
1725  */
1726 switch (Function)
1727 {
1728 case ACPI_READ:
1729     /*
1730      * Set the pointer Value to whatever is in the buffer
1731      */
1732     ACPI_MEMCPY (Value, BufferValue, ByteWidth);
1733     break;

1735 case ACPI_WRITE:
1736     /*
1737      * Write the contents of Value to the buffer
1738      */
1739     ACPI_MEMCPY (BufferValue, Value, ByteWidth);
1740     break;

1742 default:

1744     return (AE_BAD_PARAMETER);
1745 }

1747 if (AcpiGbl_DisplayRegionAccess)
1748 {
1749     switch (SpaceId)
1750     {
1751     case ACPI_ADR_SPACE_SYSTEM_MEMORY:

1753         AcpiOsPrintf ("AcpiExec: SystemMemory "
1754             "%s: Val %.8X Addr %.4X Width %X [REGION: BaseAddr %.4X Len %.2X
1755             (Function & ACPI_IO_MASK) ? "Write" : "Read ",
1756             (UINT32) *Value, (UINT32) Address, BitWidth, (UINT32) BaseAddress
1757             break;

1759     case ACPI_ADR_SPACE_GPIO: /* ACPI 5.0 */

1761         /* This space is required to always be ByteAcc */

1763         Status = AcpiBufferToResource (MyContext->Connection,
1764             MyContext->Length, &Resource);

1766         AcpiOsPrintf ("AcpiExec: GeneralPurposeIo "
1767             "%s: Val %.8X Addr %.4X BaseAddr %.4X Len %.2X Width %X AccLen %
1768             (Function & ACPI_IO_MASK) ? "Write" : "Read ", (UINT32) *Value,
1769             (UINT32) Address, (UINT32) BaseAddress, Length, BitWidth,
1770             MyContext->AccessLength, MyContext->Connection);
1771         break;

```

```

1773     default:

1775         break;
1776     }
1777 }

1779     return (AE_OK);
1780 }

```

new/usr/src/common/acpica/tools/acpiexec/aemain.c

1

```
*****
16416 Fri Jan 17 07:31:15 2014
new/usr/src/common/acpica/tools/acpiexec/aemain.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: aemain - Main routine for the AcpiExec utility
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "aeccommon.h"
45
46 #define _COMPONENT ACPI_TOOLS
47 ACPI_MODULE_NAME ("aemain")
48
49 /*
50 * Main routine for the ACPI user-space execution utility.
51 *
52 * Portability note: The utility depends upon the host for command-line
53 * wildcard support - it is not implemented locally. For example:
54 *
55 * Linux/Unix systems: Shell expands wildcards automatically.
56 *
57 * Windows: The setargv.obj module must be linked in to automatically
```

new/usr/src/common/acpica/tools/acpiexec/aemain.c

2

```
59 * expand wildcards.
60 */
61 extern BOOLEAN AcpiGbl_DebugTimeout;
62
63 /* Local prototypes */
64
65 static int
66 AeDoOptions (
67     int argc,
68     char **argv);
69
70 static ACPI_STATUS
71 AcpiDbgRunBatchMode (
72     void);
73
74
75 #define AE_BUFFER_SIZE 1024
76 #define ASL_MAX_FILES 256
77
78 /* Execution modes */
79
80 #define AE_MODE_COMMAND_LOOP 0 /* Normal command execution loop */
81 #define AE_MODE_BATCH_MULTIPLE 1 /* -b option to execute a command line */
82 #define AE_MODE_BATCH_SINGLE 2 /* -m option to execute a single control
83
84
85 /* Globals */
86
87 UINT8 AcpiGbl_RegionFillValue = 0;
88 BOOLEAN AcpiGbl_IgnoreErrors = FALSE;
89 BOOLEAN AcpiGbl_DbOpt_NoRegionSupport = FALSE;
90 UINT8 AcpiGbl_UseHwReducedFadt = FALSE;
91 BOOLEAN AcpiGbl_DoInterfaceTests = FALSE;
92 BOOLEAN AcpiGbl_LoadTestTables = 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;
96
97 #define ACPIEXEC_NAME "AML Execution/Debug Utility"
98 #define AE_SUPPORTED_OPTIONS "?b:d:e:f:ghm^orv^:x:"
99
100
101 /*****
102 *
103 * FUNCTION: usage
104 *
105 * PARAMETERS: None
106 *
107 * RETURN: None
108 *
109 * DESCRIPTION: Print a usage message
110 *
111 *****/
112
113 static void
114 usage (
115     void)
116 {
117
118     ACPI_USAGE_HEADER ("acpiexec [options] AMLfile1 AMLfile2 ...");
119
120     ACPI_OPTION ("-b \"CommandLine\"", "Batch mode command line execution (cmdl
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 ("-el",      "Enable loading of additional test table
135     ACPI_OPTION ("-em",      "Enable Interpreter Serialized Mode");
136     ACPI_OPTION ("-es",      "Enable Interpreter Slack Mode");
137     ACPI_OPTION ("-et",      "Enable debug semaphore timeout");
138     printf ("\n");

140     ACPI_OPTION ("-f <Value>", "Operation Region initialization fill va
141     ACPI_OPTION ("-r",      "Use hardware-reduced FADT V5");
142     ACPI_OPTION ("-v",      "Display version information");
143     ACPI_OPTION ("-vi",      "Verbose initialization output");
144     ACPI_OPTION ("-vr",      "Verbose region handler output");
145     ACPI_OPTION ("-x <DebugLevel>", "Debug output level");
146 }

```

```

149 /*****
150 *
151 * FUNCTION:     AeDoOptions
152 *
153 * PARAMETERS:  argc/argv      - Standard argc/argv
154 *
155 * RETURN:      Status
156 *
157 * DESCRIPTION: Command line option processing
158 *
159 *****/

```

```

161 static int
162 AeDoOptions (
163     int          argc,
164     char         **argv)
165 {
166     int          j;

169     while ((j = AcpiGetopt (argc, argv, AE_SUPPORTED_OPTIONS)) != EOF) switch (j)
170     {
171     case 'b':

173         if (strlen (AcpiGbl_Optarg) > (AE_BUFFER_SIZE - 1))
174         {
175             printf ("**** The length of command line (%u) exceeded maximum (%u)\n",
176                 (UINT32) strlen (AcpiGbl_Optarg), (AE_BUFFER_SIZE - 1));
177             return (-1);
178         }
179         AcpiGbl_ExecutionMode = AE_MODE_BATCH_MULTIPLE;
180         strcpy (BatchBuffer, AcpiGbl_Optarg);
181         break;

183     case 'd':

185         switch (AcpiGbl_Optarg[0])
186         {
187         case 'a':

189             AcpiGbl_IgnoreErrors = TRUE;
190             break;

```

```

192     case 'i':

194         AcpiGbl_DbOpt_ini_methods = FALSE;
195         break;

197     case 'o':

199         AcpiGbl_DbOpt_NoRegionSupport = TRUE;
200         break;

202     case 'r':

204         AcpiGbl_DisableAutoRepair = TRUE;
205         break;

207     case 't':

209         #ifdef ACPI_DBG_TRACK_ALLOCATIONS
210             AcpiGbl_DisableMemTracking = TRUE;
211         #endif
212         break;

214     default:

216         printf ("Unknown option: -d%s\n", AcpiGbl_Optarg);
217         return (-1);
218     }
219     break;

221     case 'e':

223         switch (AcpiGbl_Optarg[0])
224         {
225         case 'f':

227             #ifdef ACPI_DBG_TRACK_ALLOCATIONS
228                 AcpiGbl_DisplayFinalMemStats = TRUE;
229             #endif
230             break;

232     case 'i':

234         AcpiGbl_DoInterfaceTests = TRUE;
235         break;

237     case 'l':

239         AcpiGbl_LoadTestTables = TRUE;
240         break;

242     case 'm':

244         AcpiGbl_AllMethodsSerialized = TRUE;
245         printf ("Enabling AML Interpreter serialized mode\n");
246         break;

248     case 's':

250         AcpiGbl_EnableInterpreterSlack = TRUE;
251         printf ("Enabling AML Interpreter slack mode\n");
252         break;

254     case 't':

256         AcpiGbl_DebugTimeout = TRUE;

```

```

257         break;

259     default:

261         printf ("Unknown option: -e%s\n", AcpiGbl_Optarg);
262         return (-1);
263     }
264     break;

266     case 'f':

268         AcpiGbl_RegionFillValue = (UINT8) strtoul (AcpiGbl_Optarg, NULL, 0);
269         break;

271     case 'g':

273         AcpiGbl_DbOpt_tables = TRUE;
274         AcpiGbl_DbFilename = NULL;
275         break;

277     case 'h':
278     case '?':

280         usage();
281         return (0);

283     case 'm':

285         AcpiGbl_ExecutionMode = AE_MODE_BATCH_SINGLE;
286         switch (AcpiGbl_Optarg[0])
287         {
288             case '^':

290                 strcpy (BatchBuffer, "MAIN");
291                 break;

293             default:

295                 strcpy (BatchBuffer, AcpiGbl_Optarg);
296                 break;
297         }
298         break;

300     case 'o':

302         AcpiGbl_DbOpt_disasm = TRUE;
303         AcpiGbl_DbOpt_stats = TRUE;
304         break;

306     case 'r':

308         AcpiGbl_UseHwReducedFadt = TRUE;
309         printf ("Using ACPI 5.0 Hardware Reduced Mode via version 5 FADT\n");
310         break;

312     case 'v':

314         switch (AcpiGbl_Optarg[0])
315         {
316             case '^': /* -v: (Version): signon already emitted, just exit */

318                 exit (0);

320             case 'i':

322                 AcpiDbgLevel |= ACPI_LV_INIT_NAMES;

```

```

323         break;

325     case 'r':

327         AcpiGbl_DisplayRegionAccess = TRUE;
328         break;

330     default:

332         printf ("Unknown option: -v%s\n", AcpiGbl_Optarg);
333         return (-1);
334     }
335     break;

337     case 'x':

339         AcpiDbgLevel = strtoul (AcpiGbl_Optarg, NULL, 0);
340         AcpiGbl_DbConsoleDebugLevel = AcpiDbgLevel;
341         printf ("Debug Level: 0x%8.8X\n", AcpiDbgLevel);
342         break;

344     default:

346         usage();
347         return (-1);
348     }

350     return (0);
351 }

354 /*****
355  *
356  * FUNCTION:    main
357  *
358  * PARAMETERS:  argc, argv
359  *
360  * RETURN:      Status
361  *
362  * DESCRIPTION: Main routine for AcpiExec utility
363  *
364  *****/

366 int ACPI_SYSTEM_XFACE
367 main (
368     int             argc,
369     char            **argv)
370 {
371     ACPI_STATUS      Status;
372     UINT32           InitFlags;
373     ACPI_TABLE_HEADER *Table = NULL;
374     UINT32           TableCount;
375     AE_TABLE_DESC     *TableDesc;

378     ACPI_DEBUG_INITIALIZE (); /* For debug version only */

380     printf (ACPI_COMMON_SIGNON (ACPIEXEC_NAME));
381     if (argc < 2)
382     {
383         usage ();
384         return (0);
385     }

387     signal (SIGINT, AeCtrlCHandler);

```

```

389  /* Init globals */
391  AcpiDbgLevel = ACPI_NORMAL_DEFAULT;
392  AcpiDbgLayer = 0xFFFFFFFF;
394  /* Init ACPI and start debugger thread */
396  Status = AcpiInitializeSubsystem ();
397  AE_CHECK_OK (AcpiInitializeSubsystem, Status);
398  if (ACPI_FAILURE (Status))
399  {
400      goto ErrorExit;
401  }
403  /* Get the command line options */
405  if (AeDoOptions (argc, argv))
406  {
407      goto ErrorExit;
408  }
410  /* The remaining arguments are filenames for ACPI tables */
412  if (!argv[AcpiGbl_Optind])
413  {
414      goto EnterDebugger;
415  }
417  AcpiGbl_DbOpt_tables = TRUE;
418  TableCount = 0;
420  /* Get each of the ACPI table files on the command line */
422  while (argv[AcpiGbl_Optind])
423  {
424      /* Get one entire table */
426      Status = AcpiDbReadTableFromFile (argv[AcpiGbl_Optind], &Table);
427      if (ACPI_FAILURE (Status))
428      {
429          printf ("**** Could not get table from file %s, %s\n",
430              argv[AcpiGbl_Optind], AcpiFormatException (Status));
431          goto ErrorExit;
432      }
434      /* Ignore non-AML tables, we can't use them. Except for an FADT */
436      if (!ACPI_COMPARE_NAME (Table->Signature, ACPI_SIG_FADT) &&
437          !AcpiUtIsAmlTable (Table))
438      {
439          ACPI_INFO ((AE_INFO,
440              "Table [%4.4s] is not an AML table, ignoring",
441              Table->Signature));
442          AcpiOsFree (Table);
443      }
444      else
445      {
446          /* Allocate and link a table descriptor */
448          TableDesc = AcpiOsAllocate (sizeof (AE_TABLE_DESC));
449          TableDesc->Table = Table;
450          TableDesc->Next = AeTableListHead;
451          AeTableListHead = TableDesc;
453          TableCount++;
454      }

```

```

456      AcpiGbl_Optind++;
457  }
459  printf ("\n");
461  /* Build a local RSDT with all tables and let ACPICA process the RSDT */
463  Status = AeBuildLocalTables (TableCount, AeTableListHead);
464  if (ACPI_FAILURE (Status))
465  {
466      goto ErrorExit;
467  }
469  Status = AeInstallTables ();
470  if (ACPI_FAILURE (Status))
471  {
472      printf ("**** Could not load ACPI tables, %s\n",
473          AcpiFormatException (Status));
474      goto EnterDebugger;
475  }
477  /*
478   * Install most of the handlers.
479   * Override some default region handlers, especially SystemMemory
480   */
481  Status = AeInstallEarlyHandlers ();
482  if (ACPI_FAILURE (Status))
483  {
484      goto EnterDebugger;
485  }
487  /* Setup initialization flags for ACPICA */
489  InitFlags = (ACPI_NO_HANDLER_INIT | ACPI_NO_ACPI_ENABLE);
490  if (!AcpiGbl_DbOpt_ini_methods)
491  {
492      InitFlags |= (ACPI_NO_DEVICE_INIT | ACPI_NO_OBJECT_INIT);
493  }
495  /*
496   * Main initialization for ACPICA subsystem
497   * TBD: Need a way to call this after the ACPI table "LOAD" command
498   */
499  Status = AcpiEnableSubsystem (InitFlags);
500  if (ACPI_FAILURE (Status))
501  {
502      printf ("**** Could not EnableSubsystem, %s\n",
503          AcpiFormatException (Status));
504      goto EnterDebugger;
505  }
507  /*
508   * Install handlers for "device driver" space IDs (EC,SMBus, etc.)
509   * and fixed event handlers
510   */
511  AeInstallLateHandlers ();
513  /* Finish the ACPICA initialization */
515  Status = AcpiInitializeObjects (InitFlags);
516  if (ACPI_FAILURE (Status))
517  {
518      printf ("**** Could not InitializeObjects, %s\n",
519          AcpiFormatException (Status));
520      goto EnterDebugger;

```

```

521     }
523     AeMiscellaneousTests ();

526 EnterDebugger:
528     /* Exit if error above and we are in one of the batch modes */
530     if (ACPI_FAILURE (Status) && (AcpiGbl_ExecutionMode > 0))
531     {
532         goto ErrorExit;
533     }

535     /* Run a batch command or enter the command loop */
537     switch (AcpiGbl_ExecutionMode)
538     {
539     default:
540     case AE_MODE_COMMAND_LOOP:

542         AcpiDbUserCommands (ACPI_DEBUGGER_COMMAND_PROMPT, NULL);
543         break;

545     case AE_MODE_BATCH_MULTIPLE:

547         AcpiDbRunBatchMode ();
548         break;

550     case AE_MODE_BATCH_SINGLE:

552         AcpiDbExecute (BatchBuffer, NULL, NULL, EX_NO_SINGLE_STEP);
553         Status = AcpiTerminate ();
554         break;
555     }

557     return (0);

560 ErrorExit:
562     (void) AcpiOsTerminate ();
563     return (-1);
564 }

567 /*****
568  *
569  * FUNCTION:      AcpiDbRunBatchMode
570  *
571  * PARAMETERS:    BatchCommandLine - A semicolon separated list of commands
572  *                to be executed.
573  *                Use only commas to separate elements of
574  *                particular command.
575  * RETURN:        Status
576  *
577  * DESCRIPTION:   For each command of list separated by ';' prepare the command
578  *                buffer and pass it to AcpiDbCommandDispatch.
579  *
580  *****/

582 static ACPI_STATUS
583 AcpiDbRunBatchMode (
584     void)
585 {
586     ACPI_STATUS      Status;

```

```

587     char            *Ptr = BatchBuffer;
588     char            *Cmd = Ptr;
589     UINT8           Run = 0;

592     AcpiGbl_MethodExecuting = FALSE;
593     AcpiGbl_StepToNextCall = FALSE;

595     while (*Ptr)
596     {
597         if (*Ptr == ',')
598         {
599             /* Convert commas to spaces */
600             *Ptr = ' ';
601         }
602         else if (*Ptr == ';')
603         {
604             *Ptr = '\0';
605             Run = 1;
606         }

608         Ptr++;

610         if (Run || (*Ptr == '\0'))
611         {
612             (void) AcpiDbCommandDispatch (Cmd, NULL, NULL);
613             Run = 0;
614             Cmd = Ptr;
615         }
616     }

618     Status = AcpiTerminate ();
619     return (Status);
620 }

```

new/usr/src/common/acpica/tools/acpiexec/aetables.c

1

```
*****
17954 Fri Jan 17 07:31:15 2014
new/usr/src/common/acpica/tools/acpiexec/aetables.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: aetables - ACPI table setup/install for acpiexec utility
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "aeccommon.h"
45 #include "aetables.h"
46
47 #define _COMPONENT          ACPI_TOOLS
48 #define ACPI_MODULE_NAME    ("aetables")
49
50 /* Local prototypes */
51
52 static void
53 AeInitializeTableHeader (
54     ACPI_TABLE_HEADER *Header,
55     char *Signature,
56     UINT32 Length);
57
58 void
```

new/usr/src/common/acpica/tools/acpiexec/aetables.c

2

```
59 AeTableOverride (
60     ACPI_TABLE_HEADER *ExistingTable,
61     ACPI_TABLE_HEADER **NewTable);
62
63 ACPI_PHYSICAL_ADDRESS
64 AeLocalGetRootPointer (
65     void);
66
67 /* User table (DSDT) */
68
69 static ACPI_TABLE_HEADER *DsdtToInstallOverride;
70
71 /* Non-AML tables that are constructed locally and installed */
72
73 static ACPI_TABLE_RSDP LocalRSDP;
74 static ACPI_TABLE_FACS LocalFACS;
75 static ACPI_TABLE_HEADER LocalTEST;
76 static ACPI_TABLE_HEADER LocalBADTABLE;
77
78 /*
79 * We need a local FADT so that the hardware subcomponent will function,
80 * even though the underlying OSD HW access functions don't do anything.
81 */
82 static ACPI_TABLE_FADT LocalFADT;
83
84 /*
85 * Use XSDT so that both 32- and 64-bit versions of this utility will
86 * function automatically.
87 */
88 static ACPI_TABLE_XSDT *LocalXSDT;
89
90 #define BASE_XSDT_TABLES 9
91 #define BASE_XSDT_SIZE ((BASE_XSDT_TABLES) * sizeof (UINT64))
92
93 #define ACPI_MAX_INIT_TABLES (32)
94 static ACPI_TABLE_DESC Tables[ACPI_MAX_INIT_TABLES];
95
96 /*****
97 *
98 * FUNCTION: AeTableOverride
99 *
100 * DESCRIPTION: Local implementation of AcpiOsTableOverride.
101 * Exercise the override mechanism
102 *
103 *
104 *****/
105
106 void
107 AeTableOverride (
108     ACPI_TABLE_HEADER *ExistingTable,
109     ACPI_TABLE_HEADER **NewTable)
110 {
111     if (!AcpiGbl_LoadTestTables)
112     {
113         *NewTable = NULL;
114         return;
115     }
116
117 /* This code exercises the table override mechanism in the core */
118
119 if (ACPI_COMPARE_NAME (ExistingTable->Signature, ACPI_SIG_DSDT))
120 {
121     *NewTable = DsdtToInstallOverride;
122 }
123 }
```



```

125  /* This code tests override of dynamically loaded tables */
127  else if (ACPI_COMPARE_NAME (ExistingTable->Signature, "OEM9"))
128  {
129      *NewTable = ACPI_CAST_PTR (ACPI_TABLE_HEADER, Ssdt3Code);
130  }
131 }

134 /*****
135  *
136  * FUNCTION:      AeInitializeTableHeader
137  *
138  * PARAMETERS:    Header          - A valid standard ACPI table header
139  *                Signature       - Signature to insert
140  *                Length         - Length of the table
141  *
142  * RETURN:        None. Header is modified.
143  *
144  * DESCRIPTION:   Initialize the table header for a local ACPI table.
145  *
146  *****/

148 static void
149 AeInitializeTableHeader (
150     ACPI_TABLE_HEADER *Header,
151     char *Signature,
152     UINT32 Length)
153 {
155     ACPI_MOVE_NAME (Header->Signature, Signature);
156     Header->Length = Length;

158     Header->OemRevision = 0x1001;
159     ACPI_STRNCPY (Header->OemId, "Intel", ACPI_OEM_ID_SIZE);
160     ACPI_STRNCPY (Header->OemTableId, "AcpiExec", ACPI_OEM_TABLE_ID_SIZE);
161     ACPI_STRNCPY (Header->AslCompilerId, "INTL", ACPI_NAME_SIZE);
162     Header->AslCompilerRevision = 0x20131218;
163 }

166 /*****
167  *
168  * FUNCTION:      AeBuildLocalTables
169  *
170  * PARAMETERS:    TableCount      - Number of tables on the command line
171  *                TableList       - List of actual tables from files
172  *
173  * RETURN:        Status
174  *
175  * DESCRIPTION:   Build a complete ACPI table chain, with a local RSDP, XSDT,
176  *                FADT, and several other test tables.
177  *
178  *****/

180 ACPI_STATUS
181 AeBuildLocalTables (
182     UINT32 TableCount,
183     AE_TABLE_DESC *TableList)
184 {
185     ACPI_PHYSICAL_ADDRESS DsdtAddress = 0;
186     UINT32 XsdtSize;
187     AE_TABLE_DESC *NextTable;
188     UINT32 NextIndex;
189     ACPI_TABLE_FADT *ExternalFadt = NULL;

```

```

192  /*
193  * Update the table count. For the DSDT, it is not put into the XSDT.
194  * For the FADT, this table is already accounted for since we usually
195  * install a local FADT.
196  */
197  NextTable = TableList;
198  while (NextTable)
199  {
200      if (ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_DSDT) ||
201          ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_FADT))
202      {
203          TableCount--;
204      }
205      NextTable = NextTable->Next;
206  }

208  XsdtSize = (((TableCount + 1) * sizeof (UINT64)) + sizeof (ACPI_TABLE_HEADER
209  if (AcpiGbl_LoadTestTables)
210  {
211      XsdtSize += BASE_XSDT_SIZE;
212  }

214  /* Build an XSDT */

216  LocalXSDT = AcpiOsAllocate (XsdtSize);
217  if (!LocalXSDT)
218  {
219      return (AE_NO_MEMORY);
220  }

222  ACPI_MEMSET (LocalXSDT, 0, XsdtSize);
223  AeInitializeTableHeader ((void *) LocalXSDT, ACPI_SIG_XSDT, XsdtSize);

225  LocalXSDT->TableOffsetEntry[0] = ACPI_PTR_TO_PHYSADDR (&LocalFadt);
226  NextIndex = 1;

228  /*
229  * Install the user tables. The DSDT must be installed in the FADT.
230  * All other tables are installed directly into the XSDT.
231  *
232  * Note: The tables are loaded in reverse order from the incoming
233  * input, which makes it match the command line order.
234  */
235  NextTable = TableList;
236  while (NextTable)
237  {
238      /*
239      * Incoming DSDT or FADT are special cases. All other tables are
240      * just immediately installed into the XSDT.
241      */
242      if (ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_DSDT))
243      {
244          if (DsdtAddress)
245          {
246              printf ("Already found a DSDT, only one allowed\n");
247              return (AE_ALREADY_EXISTS);
248          }

250          /* The incoming user table is a DSDT */

252          DsdtAddress = ACPI_PTR_TO_PHYSADDR (NextTable->Table);
253          DsdtToInstallOverride = NextTable->Table;
254      }
255      else if (ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_FADT))
256      {

```

```

257     ExternalFadt = ACPI_CAST_PTR (ACPI_TABLE_FADT, NextTable->Table);
258     LocalXSDT->TableOffsetEntry[0] = ACPI_PTR_TO_PHYSADDR (NextTable->Ta
259 }
260 else
261 {
262     /* Install the table in the XSDT */
263
264     LocalXSDT->TableOffsetEntry[TableCount - NextIndex + 1] =
265         ACPI_PTR_TO_PHYSADDR (NextTable->Table);
266     NextIndex++;
267 }
268
269     NextTable = NextTable->Next;
270 }
271
272 /* Install the optional extra local tables */
273
274 if (AcpiGbl_LoadTestTables)
275 {
276     LocalXSDT->TableOffsetEntry[NextIndex++] = ACPI_PTR_TO_PHYSADDR (&LocalT
277     LocalXSDT->TableOffsetEntry[NextIndex++] = ACPI_PTR_TO_PHYSADDR (&LocalB
278
279     /* Install two SSDTs to test multiple table support */
280
281     LocalXSDT->TableOffsetEntry[NextIndex++] = ACPI_PTR_TO_PHYSADDR (&Ssdt1C
282     LocalXSDT->TableOffsetEntry[NextIndex++] = ACPI_PTR_TO_PHYSADDR (&Ssdt2C
283
284     /* Install the OEM1 table to test LoadTable */
285
286     LocalXSDT->TableOffsetEntry[NextIndex++] = ACPI_PTR_TO_PHYSADDR (&Oem1Co
287
288     /* Install the OEMx table to test LoadTable */
289
290     LocalXSDT->TableOffsetEntry[NextIndex++] = ACPI_PTR_TO_PHYSADDR (&OemxCo
291
292     /* Install the EC DT table to test _REG */
293
294     LocalXSDT->TableOffsetEntry[NextIndex++] = ACPI_PTR_TO_PHYSADDR (&EcDtCo
295
296     /* Install two UEFIs to test multiple table support */
297
298     LocalXSDT->TableOffsetEntry[NextIndex++] = ACPI_PTR_TO_PHYSADDR (&Uefi1C
299     LocalXSDT->TableOffsetEntry[NextIndex++] = ACPI_PTR_TO_PHYSADDR (&Uefi2C
300 }
301
302 /* Build an RSDP. Contains a valid XSDT only, no RSDT */
303
304 ACPI_MEMSET (&LocalRSDP, 0, sizeof (ACPI_TABLE_RSDP));
305 ACPI_MAKE_RSDP_SIG (LocalRSDP.Signature);
306 ACPI_MEMCPY (LocalRSDP.OemId, "Intel", 6);
307
308 LocalRSDP.Revision = 2;
309 LocalRSDP.XsdtPhysicalAddress = ACPI_PTR_TO_PHYSADDR (LocalXSDT);
310 LocalRSDP.Length = sizeof (ACPI_TABLE_RSDP);
311
312 /* Set checksums for both XSDT and RSDP */
313
314 LocalXSDT->Header.Checksum = 0;
315 LocalXSDT->Header.Checksum = (UINT8) -AcpiTbChecksum (
316     (void *) LocalXSDT, LocalXSDT->Header.Length);
317
318 LocalRSDP.Checksum = 0;
319 LocalRSDP.Checksum = (UINT8) -AcpiTbChecksum (
320     (void *) &LocalRSDP, ACPI_RSDP_CHECKSUM_LENGTH);
321
322 if (!DsdtAddress)

```

```

323 {
324     /* Use the local DSDT because incoming table(s) are all SSDT(s) */
325
326     DsdtAddress = ACPI_PTR_TO_PHYSADDR (LocalDsdtCode);
327     DsdtToInstallOverride = ACPI_CAST_PTR (ACPI_TABLE_HEADER, LocalDsdtCode)
328 }
329
330 /*
331  * Build an FADT. There are three options for the FADT:
332  * 1) Incoming external FADT specified on the command line
333  * 2) A "hardware reduced" local FADT
334  * 3) A fully featured local FADT
335  */
336 if (ExternalFadt)
337 {
338     /*
339      * Use the external FADT, but we must update the DSDT/FACS addresses
340      * as well as the checksum
341      */
342     ExternalFadt->Dsdt = DsdtAddress;
343     if (!AcpiGbl_ReducedHardware)
344     {
345         ExternalFadt->Facs = ACPI_PTR_TO_PHYSADDR (&LocalFACS);
346     }
347
348     /* Is there room in the FADT for the XDsdst and XFacs 64-bit pointers? */
349
350     if (ExternalFadt->Header.Length > ACPI_PTR_DIFF (&ExternalFadt->XDsdst, E
351     {
352         ExternalFadt->XDsdst = DsdtAddress;
353
354         if (!AcpiGbl_ReducedHardware)
355         {
356             ExternalFadt->XFacs = ACPI_PTR_TO_PHYSADDR (&LocalFACS);
357         }
358     }
359
360     /* Complete the FADT with the checksum */
361
362     ExternalFadt->Header.Checksum = 0;
363     ExternalFadt->Header.Checksum = (UINT8) -AcpiTbChecksum (
364         (void *) ExternalFadt, ExternalFadt->Header.Length);
365 }
366 else if (AcpiGbl_UseHwReducedFadt)
367 {
368     ACPI_MEMCPY (&LocalFADT, HwReducedFadtCode, sizeof (ACPI_TABLE_FADT));
369     LocalFADT.Dsdt = DsdtAddress;
370     LocalFADT.XDsdst = DsdtAddress;
371
372     LocalFADT.Header.Checksum = 0;
373     LocalFADT.Header.Checksum = (UINT8) -AcpiTbChecksum (
374         (void *) &LocalFADT, LocalFADT.Header.Length);
375 }
376 else
377 {
378     /*
379      * Build a local FADT so we can test the hardware/event init
380      */
381     ACPI_MEMSET (&LocalFADT, 0, sizeof (ACPI_TABLE_FADT));
382     LocalFADT.Header.Revision = 5;
383     AeInitializeTableHeader ((void *) &LocalFADT, ACPI_SIG_FADT, sizeof (ACP
384
385     /* Setup FADT header and DSDT/FACS addresses */
386
387     LocalFADT.Dsdt = 0;
388     LocalFADT.Facs = 0;

```

```

390     LocalFADT.XDsdtd = DsdtdAddress;
391     LocalFADT.XFacs = ACPI_PTR_TO_PHYSADDR (&LocalFACS);

393     /* Miscellaneous FADT fields */

395     LocalFADT.Gpe0BlockLength = 16;
396     LocalFADT.Gpe0Block = 0x00001234;

398     LocalFADT.Gpe1BlockLength = 6;
399     LocalFADT.Gpe1Block = 0x00005678;
400     LocalFADT.Gpe1Base = 96;

402     LocalFADT.Pm1EventLength = 4;
403     LocalFADT.Pm1aEventBlock = 0x00001aaa;
404     LocalFADT.Pm1bEventBlock = 0x00001bbb;

406     LocalFADT.Pm1ControlLength = 2;
407     LocalFADT.Pm1aControlBlock = 0xB0;

409     LocalFADT.PmTimerLength = 4;
410     LocalFADT.PmTimerBlock = 0xA0;

412     LocalFADT.Pm2ControlBlock = 0xC0;
413     LocalFADT.Pm2ControlLength = 1;

415     /* Setup one example X-64 GAS field */

417     LocalFADT.XPm1bEventBlock.SpaceId = ACPI_ADR_SPACE_SYSTEM_IO;
418     LocalFADT.XPm1bEventBlock.Address = LocalFADT.Pm1bEventBlock;
419     LocalFADT.XPm1bEventBlock.BitWidth = (UINT8) ACPI_MUL_8 (LocalFADT.Pm1Ev

421     /* Complete the FADT with the checksum */

423     LocalFADT.Header.Checksum = 0;
424     LocalFADT.Header.Checksum = (UINT8) -AcpiTbChecksum (
425         (void *) &LocalFADT, LocalFADT.Header.Length);
426 }

428 /* Build a FACS */

430 ACPI_MEMSET (&LocalFACS, 0, sizeof (ACPI_TABLE_FACS));
431 ACPI_MOVE_NAME (LocalFACS.Signature, ACPI_SIG_FACS);

433 LocalFACS.Length = sizeof (ACPI_TABLE_FACS);
434 LocalFACS.GlobalLock = 0x11AA0011;

436 /* Build the optional local tables */

438 if (AcpiGbl_LoadTestTables)
439 {
440     /*
441      * Build a fake table [TEST] so that we make sure that the
442      * ACPI core ignores it
443      */
444     ACPI_MEMSET (&LocalTEST, 0, sizeof (ACPI_TABLE_HEADER));
445     ACPI_MOVE_NAME (LocalTEST.Signature, "TEST");

447     LocalTEST.Revision = 1;
448     LocalTEST.Length = sizeof (ACPI_TABLE_HEADER);
449     LocalTEST.Checksum = (UINT8) -AcpiTbChecksum (
450         (void *) &LocalTEST, LocalTEST.Length);

452     /*
453      * Build a fake table with a bad signature [BAD!] so that we make
454      * sure that the ACPI core ignores it

```

```

455     */
456     ACPI_MEMSET (&LocalBADTABLE, 0, sizeof (ACPI_TABLE_HEADER));
457     ACPI_MOVE_NAME (LocalBADTABLE.Signature, "BAD!");

459     LocalBADTABLE.Revision = 1;
460     LocalBADTABLE.Length = sizeof (ACPI_TABLE_HEADER);
461     LocalBADTABLE.Checksum = (UINT8) -AcpiTbChecksum (
462         (void *) &LocalBADTABLE, LocalBADTABLE.Length);
463 }

465     return (AE_OK);
466 }

469 /*****
470  *
471  * FUNCTION:      AeInstallTables
472  *
473  * PARAMETERS:    None
474  *
475  * RETURN:        Status
476  *
477  * DESCRIPTION:   Install the various ACPI tables
478  *
479  *****/

481 ACPI_STATUS
482 AeInstallTables (
483     void)
484 {
485     ACPI_STATUS          Status;
486     ACPI_TABLE_HEADER    Header;
487     ACPI_TABLE_HEADER    *Table;

490     Status = AcpiInitializeTables (Tables, ACPI_MAX_INIT_TABLES, TRUE);
491     AE_CHECK_OK (AcpiInitializeTables, Status);

493     Status = AcpiReallocateRootTable ();
494     AE_CHECK_OK (AcpiReallocateRootTable, Status);

496     Status = AcpiLoadTables ();
497     AE_CHECK_OK (AcpiLoadTables, Status);

499     /*
500      * Test run-time control method installation. Do it twice to test code
501      * for an existing name.
502      */
503     Status = AcpiInstallMethod (MethodCode);
504     if (ACPI_FAILURE (Status))
505     {
506         AcpiOsPrintf ("%s, Could not install method\n",
507             AcpiFormatException (Status));
508     }

510     Status = AcpiInstallMethod (MethodCode);
511     if (ACPI_FAILURE (Status))
512     {
513         AcpiOsPrintf ("%s, Could not install method\n",
514             AcpiFormatException (Status));
515     }

517     if (AcpiGbl_LoadTestTables)
518     {
519         /* Test multiple table/UEFI support. First, get the headers */

```

```
521     Status = AcpiGetTableHeader (ACPI_SIG_UEFI, 1, &Header);
522     AE_CHECK_OK (AcpiGetTableHeader, Status);

524     Status = AcpiGetTableHeader (ACPI_SIG_UEFI, 2, &Header);
525     AE_CHECK_OK (AcpiGetTableHeader, Status);

527     Status = AcpiGetTableHeader (ACPI_SIG_UEFI, 3, &Header);
528     AE_CHECK_STATUS (AcpiGetTableHeader, Status, AE_NOT_FOUND);

530     /* Now get the actual tables */

532     Status = AcpiGetTable (ACPI_SIG_UEFI, 1, &Table);
533     AE_CHECK_OK (AcpiGetTable, Status);

535     Status = AcpiGetTable (ACPI_SIG_UEFI, 2, &Table);
536     AE_CHECK_OK (AcpiGetTable, Status);

538     Status = AcpiGetTable (ACPI_SIG_UEFI, 3, &Table);
539     AE_CHECK_STATUS (AcpiGetTable, Status, AE_NOT_FOUND);
540 }

542     return (AE_OK);
543 }

546 /*****
547  *
548  * FUNCTION:      AeLocalGetRootPointer
549  *
550  * PARAMETERS:    Flags          - not used
551  *                Address        - Where the root pointer is returned
552  *
553  * RETURN:        Status
554  *
555  * DESCRIPTION:    Return a local RSDP, used to dynamically load tables via the
556  *                standard ACPI mechanism.
557  *
558  *****/

560 ACPI_PHYSICAL_ADDRESS
561 AeLocalGetRootPointer (
562     void)
563 {

565     return ((ACPI_PHYSICAL_ADDRESS) &LocalRSDP);
566 }
```

new/usr/src/common/acpica/tools/acpiexec/aetables.h

1

```
*****
21982 Fri Jan 17 07:31:16 2014
new/usr/src/common/acpica/tools/acpiexec/aetables.h
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: aetables.h - Precompiled AML ACPI tables for acpiexec
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #ifndef AETABLES_H
45 #define AETABLES_H
46
47
48 /*
49 * Miscellaneous pre-compiled AML ACPI tables to be installed
50 */
51
52 /* Local DSDT used only if not present in the input */
53
54 static unsigned char LocalDsdtdCode[] =
55 {
56     0x44,0x53,0x44,0x54,0x24,0x00,0x00,0x00, /* 00000000 "DSDT$..." */
57     0x02,0x2C,0x49,0x6E,0x74,0x65,0x6C,0x00, /* 00000008 "...Intel." */
58     0x4C,0x6F,0x63,0x61,0x6C,0x00,0x00,0x00, /* 00000010 "Local..." */
59 }
```

new/usr/src/common/acpica/tools/acpiexec/aetables.h

2

```
59     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "...INTL" */
60     0x30,0x07,0x09,0x20,
61 };
62
63 /* Several example SSDTs */
64
65 /* SSDT1 is used by ASLTS; if changed here, must also be changed in dtregions.as
66
67 static unsigned char Ssdtd1Code[] = /* Has method _T98 */
68 {
69     0x53,0x53,0x44,0x54,0x3E,0x00,0x00,0x00, /* 00000000 "SSDT>..." */
70     0x02,0x08,0x49,0x6E,0x74,0x65,0x6C,0x00, /* 00000008 "...Intel." */
71     0x73,0x73,0x64,0x74,0x31,0x00,0x00,0x00, /* 00000010 "ssdt1..." */
72     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "...INTL" */
73     0x20,0x06,0x12,0x20,0x14,0x19,0x5F,0x54, /* 00000020 "..._T" */
74     0x39,0x38,0x01,0x70,0x0D,0x53,0x53,0x44, /* 00000028 "98.p.SSD" */
75     0x54,0x31,0x20,0x2D,0x20,0x5F,0x54,0x39, /* 00000030 "T1 - _T9" */
76     0x38,0x00,0x5B,0x31,0xA4,0x00             /* 00000038 "8.[1..." */
77 };
78
79 unsigned char Ssdtd2Code[] = /* Has method _T99 */
80 {
81     0x53,0x53,0x44,0x54,0x3E,0x00,0x00,0x00, /* 00000000 "SSDT>..." */
82     0x02,0xFE,0x49,0x6E,0x74,0x65,0x6C,0x00, /* 00000008 "...Intel." */
83     0x73,0x73,0x64,0x74,0x32,0x00,0x00,0x00, /* 00000010 "ssdt2..." */
84     0x02,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "...INTL" */
85     0x20,0x06,0x12,0x20,0x14,0x19,0x5F,0x54, /* 00000020 "..._T" */
86     0x39,0x39,0x06,0x70,0x0D,0x53,0x53,0x44, /* 00000028 "99.p.SSD" */
87     0x54,0x32,0x20,0x2D,0x20,0x5F,0x54,0x39, /* 00000030 "T2 - _T9" */
88     0x39,0x00,0x5B,0x31,0xA4,0x00             /* 00000038 "9.[1..." */
89 };
90
91 unsigned char Ssdtd3Code[] = /* OEM9: Has method _T97 */
92 {
93     0x4F,0x45,0x4D,0x39,0x30,0x00,0x00,0x00, /* 00000000 "OEM10..." */
94     0x01,0xDD,0x49,0x6E,0x74,0x65,0x6C,0x00, /* 00000008 "...Intel." */
95     0x4D,0x61,0x6E,0x79,0x00,0x00,0x00,0x00, /* 00000010 "Many..." */
96     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "...INTL" */
97     0x24,0x04,0x03,0x20,0x14,0x0B,0x5F,0x54, /* 00000020 "$..._T" */
98     0x39,0x37,0x00,0x70,0x0A,0x04,0x60,0xA4, /* 00000028 "97.p...' */
99 };
100
101 unsigned char Ssdtd4Code[] = /* Has method _T96 */
102 {
103     0x53,0x53,0x44,0x54,0x2D,0x00,0x00,0x00, /* 00000000 "SSDT-..." */
104     0x02,0x2B,0x49,0x6E,0x74,0x65,0x6C,0x00, /* 00000008 "...Intel." */
105     0x73,0x73,0x64,0x74,0x34,0x00,0x00,0x00, /* 00000010 "ssdt4..." */
106     0x04,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "...INTL" */
107     0x20,0x06,0x12,0x20,0x14,0x08,0x5F,0x54, /* 00000020 "..._T" */
108     0x39,0x36,0x05,0xA4,0x00                 /* 00000028 "96..." */
109 };
110
111 /* "Hardware-Reduced" ACPI 5.0 FADT (No FACS, no ACPI hardware) */
112
113 unsigned char HwReducedFadtCode[] =
114 {
115     0x46,0x41,0x43,0x50,0x0C,0x01,0x00,0x00, /* 00000000 "FACP..." */
116     0x05,0x8C,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL" */
117     0x41,0x43,0x50,0x49,0x35,0x30,0x20,0x20, /* 00000010 "ACPI50" */
118     0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "...INTL" */
119     0x13,0x04,0x11,0x20,0x00,0x00,0x00,0x00, /* 00000020 "......" */
120     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "..." */
121     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000030 "..." */
122     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "..." */
123     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000040 "..." */
124     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000048 "..." */
125 }
```

```

125 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000050 "....." */
126 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000058 "....." */
127 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000060 "....." */
128 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000068 "....." */
129 0x00,0x00,0x78,0x00,0x01,0x08,0x00,0x01, /* 00000070 ".x....." */
130 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000078 "....." */
131 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000080 "....." */
132 0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00, /* 00000088 "....." */
133 0x00,0x00,0x00,0x00,0x01,0x20,0x00,0x02, /* 00000090 "....." */
134 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000098 "....." */
135 0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000A0 "....." */
136 0x00,0x00,0x00,0x00,0x01,0x10,0x00,0x02, /* 000000A8 "....." */
137 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000B0 "....." */
138 0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000B8 "....." */
139 0x00,0x00,0x00,0x00,0x01,0x08,0x00,0x00, /* 000000C0 "....." */
140 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000C8 "....." */
141 0x01,0x20,0x00,0x03,0x00,0x00,0x00,0x00, /* 000000D0 ". ...." */
142 0x00,0x00,0x00,0x00,0x01,0x80,0x00,0x01, /* 000000D8 "....." */
143 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000E0 "....." */
144 0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000E8 "....." */
145 0x00,0x00,0x00,0x00,0x01,0x08,0x00,0x01, /* 000000F0 "....." */
146 0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000F8 "....." */
147 0x01,0x08,0x00,0x01,0x01,0x00,0x00,0x00, /* 00000100 "....." */
148 0x00,0x00,0x00,0x00 /* 00000108 "....." */
149 };

151 /* Example OEM table */

153 static unsigned char Oem1Code[] =
154 {
155     0x4F,0x45,0x4D,0x31,0x38,0x00,0x00,0x00, /* 00000000 "OEM18..." */
156     0x01,0x4B,0x49,0x6E,0x74,0x65,0x6C,0x00, /* 00000008 ".KIntel." */
157     0x4D,0x61,0x6E,0x79,0x00,0x00,0x00,0x00, /* 00000010 "Many...." */
158     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
159     0x18,0x09,0x03,0x20,0x08,0x5F,0x58,0x54, /* 00000020 "..._XT" */
160     0x32,0x0A,0x04,0x14,0x0C,0x5F,0x58,0x54, /* 00000028 "2..._XT" */
161     0x31,0x00,0x70,0x01,0x5F,0x58,0x54,0x32, /* 00000030 "1.p._XT2" */
162 };

164 /* ASL source for this table is at the end of this file */

166 static unsigned char OemxCode[] =
167 {
168     0x4F,0x45,0x4D,0x58,0xB0,0x00,0x00,0x00, /* 00000000 "OEMX...." */
169     0x02,0x54,0x4D,0x79,0x4F,0x45,0x4D,0x00, /* 00000008 ".TMyOEM." */
170     0x54,0x65,0x73,0x74,0x00,0x00,0x00,0x00, /* 00000010 "Test...." */
171     0x32,0x04,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "2...INTL" */
172     0x31,0x03,0x10,0x20,0x14,0x1D,0x5F,0x49, /* 00000020 "1..._I" */
173     0x4E,0x49,0x00,0x70,0x0D,0x54,0x61,0x62, /* 00000028 "NI.p.Tab" */
174     0x6C,0x65,0x20,0x4F,0x45,0x4D,0x58,0x20, /* 00000030 "le OEMX" */
175     0x72,0x75,0x6E,0x6E,0x69,0x6E,0x67,0x00, /* 00000038 "running." */
176     0x5B,0x31,0x10,0x22,0x5C,0x5F,0x47,0x50, /* 00000040 "[1."_GP" */
177     0x45,0x14,0x06,0x5F,0x45,0x30,0x37,0x00, /* 00000048 "E...E07." */
178     0x14,0x06,0x5F,0x45,0x32,0x32,0x00,0x14, /* 00000050 "...E22." */
179     0x06,0x5F,0x4C,0x33,0x31,0x00,0x14,0x06, /* 00000058 "...L31..." */
180     0x5F,0x4C,0x36,0x36,0x00,0x5B,0x82,0x10, /* 00000060 "...L66.[..." */
181     0x4F,0x45,0x4D,0x31,0x08,0x5F,0x50,0x52, /* 00000068 "OEM1.PR" */
182     0x57,0x12,0x05,0x02,0x0A,0x07,0x00,0x5B, /* 00000070 "W.....[" */
183     0x82,0x10,0x4F,0x45,0x4D,0x32,0x08,0x5F, /* 00000078 "...OEM2..." */
184     0x50,0x52,0x57,0x12,0x05,0x02,0x0A,0x66, /* 00000080 "PRW....f" */
185     0x00,0x10,0x26,0x5C,0x47,0x50,0x45,0x32, /* 00000088 "...&GPE2" */
186     0x14,0x06,0x5F,0x4C,0x30,0x31,0x00,0x14, /* 00000090 "...L01..." */
187     0x06,0x5F,0x45,0x30,0x37,0x00,0x08,0x5F, /* 00000098 "...E07..." */
188     0x50,0x52,0x57,0x12,0x0C,0x02,0x12,0x08, /* 000000A0 "PRW....." */
189     0x02,0x5C,0x47,0x50,0x45,0x32,0x01,0x00 /* 000000A8 "...GPE2..." */
190 };

```

```

192 /* Example ECDT */

194 unsigned char EcdtCode[] =
195 {
196     0x45,0x43,0x44,0x54,0x4E,0x00,0x00,0x00, /* 00000000 "ECDTN..." */
197     0x01,0x94,0x20,0x49,0x6E,0x74,0x65,0x6C, /* 00000008 "... Intel" */
198     0x54,0x65,0x6D,0x70,0x6C,0x61,0x74,0x65, /* 00000010 "Template" */
199     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
200     0x16,0x03,0x11,0x20,0x01,0x08,0x00,0x00, /* 00000020 "f....." */
201     0x66,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "...b..." */
202     0x01,0x08,0x00,0x00,0x62,0x00,0x00,0x00, /* 00000030 "....." */
203     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "..._SB.PC" */
204     0x09,0x5C,0x5F,0x53,0x42,0x2E,0x50,0x43, /* 00000040 "IO.EC." */
205     0x49,0x30,0x2E,0x45,0x43,0x00 /* 00000048 */
206 };

208 /* Test for multiple UEFI tables */

210 unsigned char Uefi1Code[] =
211 {
212     0x55,0x45,0x46,0x49,0x36,0x00,0x00,0x00, /* 00000000 "UEFI6..." */
213     0x01,0x6E,0x20,0x49,0x6E,0x74,0x65,0x6C, /* 00000008 "...n Intel" */
214     0x20,0x20,0x20,0x55,0x45,0x46,0x49,0x31, /* 00000010 "UEFI1" */
215     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
216     0x23,0x08,0x13,0x20,0x06,0x07,0x08,0x09, /* 00000020 "#....." */
217     0x04,0x05,0x06,0x07,0x08,0x09,0x0A,0x0B, /* 00000028 "....." */
218     0x0C,0x0D,0x0E,0x0F,0x36,0x00 /* 00000030 "....6." */
219 };

221 unsigned char Uefi2Code[] =
222 {
223     0x55,0x45,0x46,0x49,0xAA,0x00,0x00,0x00, /* 00000000 "UEFI...." */
224     0x01,0xE0,0x20,0x49,0x6E,0x74,0x65,0x6C, /* 00000008 "... Intel" */
225     0x20,0x20,0x20,0x55,0x45,0x46,0x49,0x32, /* 00000010 "UEFI2" */
226     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
227     0x23,0x08,0x13,0x20,0x67,0x45,0x23,0x01, /* 00000020 "#...gE#" */
228     0x04,0x05,0x06,0x07,0x08,0x09,0x0A,0x0B, /* 00000028 "....." */
229     0x0C,0x0D,0x0E,0x0F,0x36,0x00,0x04,0x19, /* 00000030 "....6..." */
230     0x00,0x56,0x34,0xF2,0x04,0x03,0x02,0x01, /* 00000038 "...V4..." */
231     0x77,0x66,0x55,0x44,0x33,0x22,0x11,0x1E, /* 00000040 "wFUD3..." */
232     0x1C,0x1F,0x14,0x10,0x0C,0x08,0x04,0xAB, /* 00000048 "....." */
233     0x54,0x68,0x69,0x73,0x20,0x69,0x73,0x20, /* 00000050 "This is" */
234     0x61,0x20,0x73,0x74,0x72,0x69,0x6E,0x67, /* 00000058 "a string" */
235     0x00,0x5C,0x50,0x43,0x49,0x30,0x5C,0x41, /* 00000060 "...PCI0\A" */
236     0x42,0x43,0x44,0x00,0x36,0x00,0x55,0x00, /* 00000068 "BCD.6.U." */
237     0x6E,0x00,0x69,0x00,0x63,0x00,0x6F,0x00, /* 00000070 "n.i.c.o." */
238     0x64,0x00,0x65,0x00,0x20,0x00,0x53,0x00, /* 00000078 "d.e..s." */
239     0x74,0x00,0x72,0x00,0x69,0x00,0x6E,0x00, /* 00000080 "t.r.i.n." */
240     0x67,0x00,0x00,0x00,0x58,0x5B,0x00,0x00, /* 00000088 "g...Xl..." */
241     0x00,0x00,0x00,0x00,0x41,0x42,0x43,0x44, /* 00000090 "....ABCD" */
242     0x45,0x00,0x00,0x01,0x02,0x03,0x04,0x05, /* 00000098 "E....." */
243     0x06,0x07,0x08,0x09,0x0A,0x0B,0x0C,0x0D, /* 000000A0 "....." */
244     0x0E,0x0F /* 000000A8 "... " */
245 };

248 /*
249 * Example installable control method
250 *
251 * DefinitionBlock ("", "DSDT", 2, "Intel", "MTHDTEST", 0x20090512)
252 * {
253 *     Method (\_SI._T97, 1, Serialized)
254 *     {
255 *         Store ("Example installed method", Debug)
256 *         Store (Arg0, Debug)

```

```

257 *      Return ()
258 *    }
259 * }
260 *
261 * Compiled byte code below.
262 */
263 static unsigned char MethodCode[] =
264 {
265     0x44,0x53,0x44,0x54,0x53,0x00,0x00,0x00, /* 00000000 "DSDTS..." */
266     0x02,0xF9,0x49,0x6E,0x74,0x65,0x6C,0x00, /* 00000008 "...Intel." */
267     0x4D,0x54,0x48,0x44,0x54,0x45,0x53,0x54, /* 00000010 "MTHDTEST" */
268     0x12,0x05,0x09,0x20,0x49,0x4E,0x54,0x4C, /* 00000018 "...INTL." */
269     0x22,0x04,0x09,0x20,0x14,0x2E,0x2E,0x5F, /* 00000020 "... .." */
270     0x54,0x49,0x5F,0x5F,0x54,0x39,0x37,0x09, /* 00000028 "SI_T97." */
271     0x70,0x0D,0x45,0x78,0x61,0x6D,0x70,0x6C, /* 00000030 "p.Exampl" */
272     0x65,0x20,0x69,0x6E,0x73,0x74,0x61,0x6C, /* 00000038 "e instal" */
273     0x6C,0x65,0x64,0x20,0x6D,0x65,0x74,0x68, /* 00000040 "led meth" */
274     0x6F,0x64,0x00,0x5B,0x31,0x70,0x68,0x5B, /* 00000048 "od.[lph[" */
275     0x31,0xA4,0x00,
276 };
277
279 #if 0
280 /*****
281 *
282 * DESCRIPTION: ASL tables that are used in RSDT/XSDT, also used to test
283 *              Load/LoadTable operators.
284 *
285 *****/
286
287 DefinitionBlock ("", "OEMX", 2, "MyOEM", "Test", 0x00000432)
288 {
289     External (GPE2, DeviceObj)
290
291     Method (_INI)
292     {
293         Store ("Table OEMX running", Debug)
294     }
295
296     Scope (\_GPE)
297     {
298         Method (_E07) {}
299         Method (_E22) {}
300         Method (_L31) {}
301         Method (_L66) {}
302     }
303
304     Device (OEM1)
305     {
306         Name (_PRW, Package() {7,0})
307     }
308     Device (OEM2)
309     {
310         Name (_PRW, Package() {0x66,0})
311     }
312
313     Scope (\GPE2)
314     {
315         Method (_L01) {}
316         Method (_E07) {}
317
318         Name (_PRW, Package() {Package() {\GPE2, 1}, 0})
319     }
320 }
322 /* Parent gr.asl file */

```

```

324 DefinitionBlock ("", "SSDT", 2, "Intel", "Many", 0x00000001)
325 {
326     Name (BUF1, Buffer())
327     {
328         0x4F,0x45,0x4D,0x58,0xB0,0x00,0x00,0x00, /* 00000000 "OEMX..." */
329         0x02,0x54,0x4D,0x79,0x4F,0x45,0x4D,0x00, /* 00000008 ".TMyOEM." */
330         0x54,0x65,0x73,0x74,0x00,0x00,0x00,0x00, /* 00000010 "Test..." */
331         0x32,0x04,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "2...INTL." */
332         0x31,0x03,0x10,0x20,0x14,0x1D,0x5F,0x49, /* 00000020 "1... ..I" */
333         0x4E,0x49,0x00,0x70,0x0D,0x54,0x61,0x62, /* 00000028 "NI.p.Tab" */
334         0x6C,0x65,0x20,0x4F,0x45,0x4D,0x58,0x20, /* 00000030 "le OEMX " */
335         0x72,0x75,0x6E,0x6E,0x69,0x6E,0x67,0x00, /* 00000038 "running." */
336         0x5B,0x31,0x10,0x22,0x5C,0x5F,0x47,0x50, /* 00000040 "[1." _GP" */
337         0x45,0x14,0x06,0x5F,0x45,0x30,0x37,0x00, /* 00000048 "E..E07." */
338         0x14,0x06,0x5F,0x45,0x32,0x32,0x00,0x14, /* 00000050 "...E22.." */
339         0x06,0x5F,0x4C,0x33,0x31,0x00,0x14,0x06, /* 00000058 "...L31..." */
340         0x5F,0x4C,0x36,0x36,0x00,0x5B,0x82,0x10, /* 00000060 "...L66.[..." */
341         0x4F,0x45,0x4D,0x31,0x08,0x5F,0x50,0x52, /* 00000068 "OEM1._PR" */
342         0x57,0x12,0x05,0x02,0x0A,0x07,0x00,0x5B, /* 00000070 "W.....[" */
343         0x82,0x10,0x4F,0x45,0x4D,0x32,0x08,0x5F, /* 00000078 "...OEM2.." */
344         0x50,0x52,0x57,0x12,0x05,0x02,0x0A,0x66, /* 00000080 "PRW....f" */
345         0x00,0x10,0x26,0x5C,0x47,0x50,0x45,0x32, /* 00000088 "...&\GPE2" */
346         0x14,0x06,0x5F,0x4C,0x30,0x31,0x00,0x14, /* 00000090 "...L01..." */
347         0x06,0x5F,0x45,0x30,0x37,0x00,0x08,0x5F, /* 00000098 "...E07..." */
348         0x50,0x52,0x57,0x12,0x0C,0x02,0x12,0x08, /* 000000A0 "PRW....." */
349         0x02,0x5C,0x47,0x50,0x45,0x32,0x01,0x00 /* 000000A8 "...GPE2..." */
350     }
351
352     Name (HNDL, 0)
353     Method (LD)
354     {
355         Load (BUF1, HNDL)
356         Store ("Load operator, handle:", Debug)
357         Store (HNDL, Debug)
358     }
359
360     Method (MAIN, 0, NotSerialized)
361     {
362         Store ("Loading OEMX table", Debug)
363         Store (LoadTable ("OEMX", "MyOEM", "Test"), Debug)
364     }
365
366     Scope (\_GPE)
367     {
368         Method (_L08) {}
369         Method (_E08) {}
370         Method (_L0B) {}
371     }
372
373     Device (DEV0)
374     {
375         Name (_PRW, Package() {0x11, 0})
376     }
377
378     Device (\GPE2)
379     {
380         Method (_L00) {}
381     }
382 }
384 /* SSDT1 */
386 DefinitionBlock ("ssdt1.aml", "SSDT", 2, "Intel", "ssdt1", 0x00000001)
387 {
388     Method (_T98, 1, NotSerialized)

```

```

389 {
390     Store ("SSDT1 - _T98", Debug)
391     Return (Zero)
392 }
393 }

395 /* SSDT2 */

397 DefinitionBlock ("ssdt2.aml", "SSDT", 2, "Intel", "ssdt2", 0x00000002)
398 {
399     Method (_T99, 6, NotSerialized)
400     {
401         Store ("SSDT2 - _T99", Debug)
402         Return (Zero)
403     }
404 }

406 /* SSDT4 */

408 DefinitionBlock ("ssdt4.aml", "SSDT", 2, "Intel", "ssdt4", 0x00000004)
409 {
410     Method (_T96, 5, NotSerialized)
411     {
412         Return (Zero)
413     }
414 }

416 /* Example ECDT */

418 [000h 0000 4]          Signature : "ECDT"          /* Embedded Controller
419 [004h 0004 4]          Table Length : 0000004E
420 [008h 0008 1]          Revision : 01
421 [009h 0009 1]          Checksum : 14
422 [00Ah 0010 6]          OEM ID : " Intel"
423 [010h 0016 8]          OEM Table ID : "Template"
424 [018h 0024 4]          OEM Revision : 00000001
425 [01Ch 0028 4]          Asl Compiler ID : "INTL"
426 [020h 0032 4]          Asl Compiler Revision : 20110316

429 [024h 0036 12]        Command/Status Register : <Generic Address Structure>
430 [024h 0036 1]          Space ID : 01 (SystemIO)
431 [025h 0037 1]          Bit Width : 08
432 [026h 0038 1]          Bit Offset : 00
433 [027h 0039 1]          Encoded Access Width : 00 (Undefined/Legacy)
434 [028h 0040 8]          Address : 0000000000000066

436 [030h 0048 12]        Data Register : <Generic Address Structure>
437 [030h 0048 1]          Space ID : 01 (SystemIO)
438 [031h 0049 1]          Bit Width : 08
439 [032h 0050 1]          Bit Offset : 00
440 [033h 0051 1]          Encoded Access Width : 00 (Undefined/Legacy)
441 [034h 0052 8]          Address : 0000000000000062

443 [03Ch 0060 4]          UID : 00000000
444 [040h 0064 1]          GPE Number : 09
445 [041h 0065 13]         Namepath : "\_SB.PCI0.EC"

448 /* Test multiple UEFI support */

450 [0004]          Signature : "UEFI"          [UEFI Boot Optimization Ta
451 [0004]          Table Length : 00000036
452 [0001]          Revision : 01
453 [0001]          Checksum : 9B
454 [0006]          OEM ID : " Intel"

```

```

455 [0008]          OEM Table ID : " UEFI1"
456 [0004]          OEM Revision : 00000001
457 [0004]          Asl Compiler ID : "INTL"
458 [0004]          Asl Compiler Revision : 20100528

460 [0016]          UUID Identifier : 09080706-0504-0706-0809-0A0B0C0D0E0F
461 [0002]          Data Offset : 0000

464 [004]          Signature : "UEFI"          /* UEFI Boot Optimization T
465 [004]          Table Length : 00000036
466 [001]          Revision : 01
467 [001]          Checksum : 9B
468 [006]          OEM ID : " Intel"
469 [008]          OEM Table ID : " UEFI2"
470 [004]          OEM Revision : 00000001
471 [004]          Asl Compiler ID : "INTL"
472 [004]          Asl Compiler Revision : 20100528

474 [016]          UUID Identifier : 01234567-0504-0706-0809-0A0B0C0D0E0F
475 [002]          Data Offset : 0000

477          Label : StartRecord
478          UINT8 : 4
479          UINT16 : $EndRecord - $StartRecord /* Should
480          UINT24 : 123456 | F00000
481          UINT32 : 01020304
482          UINT56 : 11223344556677
483          UINT64 : 0102030405060708 * 4 - 200 / 100 | F0
484          Label : EndRecord

486          UINT8 : AB
487          String : "This is a string"
488          DevicePath : "\PCI0\ABCD"
489          UINT16 : $StartRecord
490          Unicode : "Unicode String"
491          UINT64 : $EndRecord * 128

493          Buffer : 41 42 43 44 45
494          String : ""
495          GUID : 03020100-0504-0706-0809-0A0B0C0D0E0F
496 #endif

498 #endif /* __AETABLES_H__ */

```



new/usr/src/common/acpica/tools/acpihelp/acpihelp.h

1

```
*****
4277 Fri Jan 17 07:31:16 2014
new/usr/src/common/acpica/tools/acpihelp/acpihelp.h
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: acpihelp.h - Include file for AcpiHelp utility
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __ACPIHELP_H
45 #define __ACPIHELP_H

48 #include "acpi.h"
49 #include "accommon.h"
50 #include "acapps.h"

52 #include <stdio.h>
53 #include <sys/stat.h>
54 #include <sys/types.h>
55 #include <fcntl.h>
56 #include <ctype.h>
57 #include <string.h>
58 #ifdef WIN32
59 #include <io.h>
60 #include <direct.h>
```

new/usr/src/common/acpica/tools/acpihelp/acpihelp.h

2

```
61 #endif
62 #include <errno.h>

65 #define AH_DECODE_DEFAULT 0
66 #define AH_DECODE_ASL 1
67 #define AH_DECODE_ASL_KEYWORD 2
68 #define AH_DECODE_PREDEFINED_NAME 3
69 #define AH_DECODE_AML 4
70 #define AH_DECODE_AML_OPCODE 5
71 #define AH_DISPLAY_DEVICE_IDS 6
72 #define AH_DECODE_EXCEPTION 7

74 #define AH_MAX_ASL_LINE_LENGTH 70
75 #define AH_MAX_AML_LINE_LENGTH 100

78 typedef struct ah_aml_opcode
79 {
80     UINT16 OpcodeRangeStart;
81     UINT16 OpcodeRangeEnd;
82     char *OpcodeString;
83     char *OpcodeName;
84     char *Type;
85     char *FixedArguments;
86     char *VariableArguments;
87     char *Grammar;
88 } AH_AML_OPCODE;

91 typedef struct ah_asl_operator
92 {
93     char *Name;
94     char *Syntax;
95     char *Description;
96 } AH_ASL_OPERATOR;

99 typedef struct ah_asl_keyword
100 {
101     char *Name;
102     char *Description;
103     char *KeywordList;
104 } AH_ASL_KEYWORD;

107 typedef struct ah_device_id
108 {
109     char *Name;
110     char *Description;
111 } AH_DEVICE_ID;

112 } AH_DEVICE_ID;

115 extern const AH_AML_OPCODE AmlOpcodeInfo[];
116 extern const AH_ASL_OPERATOR AslOperatorInfo[];
117 extern const AH_ASL_KEYWORD AslKeywordInfo[];
118 extern BOOLEAN AhDisplayAll;

120 void
121 AhStrupr (
122     char *SrcString);

124 void
125 AhFindAmlOpcode (
126     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 */
```

new/usr/src/common/acpica/tools/acpihelp/ahamlops.c

1

```
*****
34670 Fri Jan 17 07:31:16 2014
new/usr/src/common/acpica/tools/acpihelp/ahamlops.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: ahamlops - Table of all known AML opcodes
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "acpihelp.h"
45
46 /*
47 * AML opcodes with related syntax and grammar information.
48 * This table was extracted from the ACPI specification.
49 */
50 const AH_AML_OPCODE AmlOpcodeInfo[] =
51 {
52     {0x00, 0x00, "0x00", "ZeroOp", "DataObject",
53     {0x01, 0x01, "0x01", "OneOp", "DataObject",
54     {0x02, 0x05, "0x02-0x05", NULL, "DataObject",
55     {0x06, 0x06, "0x06", "AliasOp", "TermObject",
56     {0x07, 0x07, "0x07", NULL, NULL,
57     {0x08, 0x08, "0x08", "NameOp", "TermObject",
58     {0x09, 0x09, "0x09", NULL, "DefName := Name",
59     {0x0A, 0x0A, "0x0A", "BytePrefix", "DataObject",
60     {0x0B, 0x0B, "0x0B", "WordPrefix", "DataObject",
61     {0x0C, 0x0C, "0x0C", "DWordPrefix", "DataObject",
62     {0x0D, 0x0D, "0x0D", "StringPrefix", "DataObject",
63     {0x0E, 0x0E, "0x0E", "QWordPrefix", "DataObject",
64     {0x0F, 0x0F, "0x0F", NULL, "QWordConst := Q",
65     {0x10, 0x10, "0x10", "ScopeOp", "TermObject",
66     {0x11, 0x11, "0x11", "BufferOp", "TermObject",
67     {0x12, 0x12, "0x12", "PackageOp", "TermObject",
68     {0x13, 0x13, "0x13", "VarPackageOp", "TermObject",
69     {0x14, 0x14, "0x14", "MethodOp", "TermObject",
70     {0x15, 0x2D, "0x15-0x2D", NULL, "DefMethod := Me",
71     {0x2E, 0x2E, "0x2E", "DualNamePrefix", "NameObject",
72     {0x2F, 0x2F, "0x2F", "MultiNamePrefix", "DualNamePath :=",
73     {0x30, 0x39, "0x30-0x39", "DigitChar", "NameObject",
74     {0x3A, 0x40, "0x3A-0x40", NULL, "NameChar",
75     {0x41, 0x5A, "0x41-0x5A", "NameChar", "NameObject",
76     {0x5B, 0x5B, "0x5B", "ExtOpPrefix", "DataObject",
77     {0x5B00, 0x5B00, "0x5B00", NULL, "DataObject",
78     {0x5B01, 0x5B01, "0x5B01", "MutexOp", "TermObject",
79     {0x5B02, 0x5B02, "0x5B02", "EventOp", "TermObject",
80     {0x5B12, 0x5B12, "0x5B12", "CondRefOfOp", "DefEvent := Eve",
81     {0x5B13, 0x5B13, "0x5B13", "CreateFieldOp", "DefCondRefOf :=",
82     {0x5B1F, 0x5B1F, "0x5B1F", "LoadTableOp", "TermObject",
83     {0x5B20, 0x5B20, "0x5B20", "LoadOp", "DefLoadTable :=",
84     {0x5B21, 0x5B21, "0x5B21", "StallOp", "TermObject",
85     {0x5B22, 0x5B22, "0x5B22", "SleepOp", "DefStall := Sta",
86     {0x5B23, 0x5B23, "0x5B23", "AcquireOp", "TermObject",
87     {0x5B24, 0x5B24, "0x5B24", "SignalOp", "DefAcquire := A",
88     {0x5B25, 0x5B25, "0x5B25", "WaitOp", "DefSignal := Si",
89     {0x5B26, 0x5B26, "0x5B26", "ResetOp", "TermObject",
90     {0x5B27, 0x5B27, "0x5B27", "WaitOp", "DefWait := Wait",
91     {0x5B28, 0x5B28, "0x5B28", "ResetOp", "TermObject",
92     {0x5B29, 0x5B29, "0x5B29", "WaitOp", "DefWait := Wait",
93     {0x5B2A, 0x5B2A, "0x5B2A", "ResetOp", "TermObject",
94     {0x5B2B, 0x5B2B, "0x5B2B", "WaitOp", "DefWait := Wait",
95     {0x5B2C, 0x5B2C, "0x5B2C", "ResetOp", "TermObject",
96     {0x5B2D, 0x5B2D, "0x5B2D", "WaitOp", "DefWait := Wait",
97     {0x5B2E, 0x5B2E, "0x5B2E", "ResetOp", "TermObject",
98     {0x5B2F, 0x5B2F, "0x5B2F", "WaitOp", "DefWait := Wait",
99     {0x5B30, 0x5B30, "0x5B30", "ResetOp", "TermObject",
100     {0x5B31, 0x5B31, "0x5B31", "WaitOp", "DefWait := Wait",
101     {0x5B32, 0x5B32, "0x5B32", "ResetOp", "TermObject",
102     {0x5B33, 0x5B33, "0x5B33", "WaitOp", "DefWait := Wait",
103     {0x5B34, 0x5B34, "0x5B34", "ResetOp", "TermObject",
104     {0x5B35, 0x5B35, "0x5B35", "WaitOp", "DefWait := Wait",
105     {0x5B36, 0x5B36, "0x5B36", "ResetOp", "TermObject",
106     {0x5B37, 0x5B37, "0x5B37", "WaitOp", "DefWait := Wait",
107     {0x5B38, 0x5B38, "0x5B38", "ResetOp", "TermObject",
108     {0x5B39, 0x5B39, "0x5B39", "WaitOp", "DefWait := Wait",
109     {0x5B3A, 0x5B3A, "0x5B3A", "ResetOp", "TermObject",
110     {0x5B3B, 0x5B3B, "0x5B3B", "WaitOp", "DefWait := Wait",
111     {0x5B3C, 0x5B3C, "0x5B3C", "ResetOp", "TermObject",
112     {0x5B3D, 0x5B3D, "0x5B3D", "WaitOp", "DefWait := Wait",
113     {0x5B3E, 0x5B3E, "0x5B3E", "ResetOp", "TermObject",
114     {0x5B3F, 0x5B3F, "0x5B3F", "WaitOp", "DefWait := Wait",
115     {0x5B40, 0x5B40, "0x5B40", "ResetOp", "TermObject",
116     {0x5B41, 0x5B41, "0x5B41", "WaitOp", "DefWait := Wait",
117     {0x5B42, 0x5B42, "0x5B42", "ResetOp", "TermObject",
118     {0x5B43, 0x5B43, "0x5B43", "WaitOp", "DefWait := Wait",
119     {0x5B44, 0x5B44, "0x5B44", "ResetOp", "TermObject",
120     {0x5B45, 0x5B45, "0x5B45", "WaitOp", "DefWait := Wait",
121     {0x5B46, 0x5B46, "0x5B46", "ResetOp", "TermObject",
122     {0x5B47, 0x5B47, "0x5B47", "WaitOp", "DefWait := Wait",
123     {0x5B48, 0x5B48, "0x5B48", "ResetOp", "TermObject",
124     {0x5B49, 0x5B49, "0x5B49", "WaitOp", "DefWait := Wait",
125     {0x5B4A, 0x5B4A, "0x5B4A", "ResetOp", "TermObject",
126     {0x5B4B, 0x5B4B, "0x5B4B", "WaitOp", "DefWait := Wait",
127     {0x5B4C, 0x5B4C, "0x5B4C", "ResetOp", "TermObject",
128     {0x5B4D, 0x5B4D, "0x5B4D", "WaitOp", "DefWait := Wait",
129     {0x5B4E, 0x5B4E, "0x5B4E", "ResetOp", "TermObject",
130     {0x5B4F, 0x5B4F, "0x5B4F", "WaitOp", "DefWait := Wait",
131     {0x5B50, 0x5B50, "0x5B50", "ResetOp", "TermObject",
132     {0x5B51, 0x5B51, "0x5B51", "WaitOp", "DefWait := Wait",
133     {0x5B52, 0x5B52, "0x5B52", "ResetOp", "TermObject",
134     {0x5B53, 0x5B53, "0x5B53", "WaitOp", "DefWait := Wait",
135     {0x5B54, 0x5B54, "0x5B54", "ResetOp", "TermObject",
136     {0x5B55, 0x5B55, "0x5B55", "WaitOp", "DefWait := Wait",
137     {0x5B56, 0x5B56, "0x5B56", "ResetOp", "TermObject",
138     {0x5B57, 0x5B57, "0x5B57", "WaitOp", "DefWait := Wait",
139     {0x5B58, 0x5B58, "0x5B58", "ResetOp", "TermObject",
140     {0x5B59, 0x5B59, "0x5B59", "WaitOp", "DefWait := Wait",
141     {0x5B5A, 0x5B5A, "0x5B5A", "ResetOp", "TermObject",
142     {0x5B5B, 0x5B5B, "0x5B5B", "WaitOp", "DefWait := Wait",
143     {0x5B5C, 0x5B5C, "0x5B5C", "ResetOp", "TermObject",
144     {0x5B5D, 0x5B5D, "0x5B5D", "WaitOp", "DefWait := Wait",
145     {0x5B5E, 0x5B5E, "0x5B5E", "ResetOp", "TermObject",
146     {0x5B5F, 0x5B5F, "0x5B5F", "WaitOp", "DefWait := Wait",
147     {0x5B60, 0x5B60, "0x5B60", "ResetOp", "TermObject",
148     {0x5B61, 0x5B61, "0x5B61", "WaitOp", "DefWait := Wait",
149     {0x5B62, 0x5B62, "0x5B62", "ResetOp", "TermObject",
150     {0x5B63, 0x5B63, "0x5B63", "WaitOp", "DefWait := Wait",
151     {0x5B64, 0x5B64, "0x5B64", "ResetOp", "TermObject",
152     {0x5B65, 0x5B65, "0x5B65", "WaitOp", "DefWait := Wait",
153     {0x5B66, 0x5B66, "0x5B66", "ResetOp", "TermObject",
154     {0x5B67, 0x5B67, "0x5B67", "WaitOp", "DefWait := Wait",
155     {0x5B68, 0x5B68, "0x5B68", "ResetOp", "TermObject",
156     {0x5B69, 0x5B69, "0x5B69", "WaitOp", "DefWait := Wait",
157     {0x5B6A, 0x5B6A, "0x5B6A", "ResetOp", "TermObject",
158     {0x5B6B, 0x5B6B, "0x5B6B", "WaitOp", "DefWait := Wait",
159     {0x5B6C, 0x5B6C, "0x5B6C", "ResetOp", "TermObject",
160     {0x5B6D, 0x5B6D, "0x5B6D", "WaitOp", "DefWait := Wait",
161     {0x5B6E, 0x5B6E, "0x5B6E", "ResetOp", "TermObject",
162     {0x5B6F, 0x5B6F, "0x5B6F", "WaitOp", "DefWait := Wait",
163     {0x5B70, 0x5B70, "0x5B70", "ResetOp", "TermObject",
164     {0x5B71, 0x5B71, "0x5B71", "WaitOp", "DefWait := Wait",
165     {0x5B72, 0x5B72, "0x5B72", "ResetOp", "TermObject",
166     {0x5B73, 0x5B73, "0x5B73", "WaitOp", "DefWait := Wait",
167     {0x5B74, 0x5B74, "0x5B74", "ResetOp", "TermObject",
168     {0x5B75, 0x5B75, "0x5B75", "WaitOp", "DefWait := Wait",
169     {0x5B76, 0x5B76, "0x5B76", "ResetOp", "TermObject",
170     {0x5B77, 0x5B77, "0x5B77", "WaitOp", "DefWait := Wait",
171     {0x5B78, 0x5B78, "0x5B78", "ResetOp", "TermObject",
172     {0x5B79, 0x5B79, "0x5B79", "WaitOp", "DefWait := Wait",
173     {0x5B7A, 0x5B7A, "0x5B7A", "ResetOp", "TermObject",
174     {0x5B7B, 0x5B7B, "0x5B7B", "WaitOp", "DefWait := Wait",
175     {0x5B7C, 0x5B7C, "0x5B7C", "ResetOp", "TermObject",
176     {0x5B7D, 0x5B7D, "0x5B7D", "WaitOp", "DefWait := Wait",
177     {0x5B7E, 0x5B7E, "0x5B7E", "ResetOp", "TermObject",
178     {0x5B7F, 0x5B7F, "0x5B7F", "WaitOp", "DefWait := Wait",
179     {0x5B80, 0x5B80, "0x5B80", "ResetOp", "TermObject",
180     {0x5B81, 0x5B81, "0x5B81", "WaitOp", "DefWait := Wait",
181     {0x5B82, 0x5B82, "0x5B82", "ResetOp", "TermObject",
182     {0x5B83, 0x5B83, "0x5B83", "WaitOp", "DefWait := Wait",
183     {0x5B84, 0x5B84, "0x5B84", "ResetOp", "TermObject",
184     {0x5B85, 0x5B85, "0x5B85", "WaitOp", "DefWait := Wait",
185     {0x5B86, 0x5B86, "0x5B86", "ResetOp", "TermObject",
186     {0x5B87, 0x5B87, "0x5B87", "WaitOp", "DefWait := Wait",
187     {0x5B88, 0x5B88, "0x5B88", "ResetOp", "TermObject",
188     {0x5B89, 0x5B89, "0x5B89", "WaitOp", "DefWait := Wait",
189     {0x5B8A, 0x5B8A, "0x5B8A", "ResetOp", "TermObject",
190     {0x5B8B, 0x5B8B, "0x5B8B", "WaitOp", "DefWait := Wait",
191     {0x5B8C, 0x5B8C, "0x5B8C", "ResetOp", "TermObject",
192     {0x5B8D, 0x5B8D, "0x5B8D", "WaitOp", "DefWait := Wait",
193     {0x5B8E, 0x5B8E, "0x5B8E", "ResetOp", "TermObject",
194     {0x5B8F, 0x5B8F, "0x5B8F", "WaitOp", "DefWait := Wait",
195     {0x5B90, 0x5B90, "0x5B90", "ResetOp", "TermObject",
196     {0x5B91, 0x5B91, "0x5B91", "WaitOp", "DefWait := Wait",
197     {0x5B92, 0x5B92, "0x5B92", "ResetOp", "TermObject",
198     {0x5B93, 0x5B93, "0x5B93", "WaitOp", "DefWait := Wait",
199     {0x5B94, 0x5B94, "0x5B94", "ResetOp", "TermObject",
200     {0x5B95, 0x5B95, "0x5B95", "WaitOp", "DefWait := Wait",
201     {0x5B96, 0x5B96, "0x5B96", "ResetOp", "TermObject",
202     {0x5B97, 0x5B97, "0x5B97", "WaitOp", "DefWait := Wait",
203     {0x5B98, 0x5B98, "0x5B98", "ResetOp", "TermObject",
204     {0x5B99, 0x5B99, "0x5B99", "WaitOp", "DefWait := Wait",
205     {0x5B9A, 0x5B9A, "0x5B9A", "ResetOp", "TermObject",
206     {0x5B9B, 0x5B9B, "0x5B9B", "WaitOp", "DefWait := Wait",
207     {0x5B9C, 0x5B9C, "0x5B9C", "ResetOp", "TermObject",
208     {0x5B9D, 0x5B9D, "0x5B9D", "WaitOp", "DefWait := Wait",
209     {0x5B9E, 0x5B9E, "0x5B9E", "ResetOp", "TermObject",
210     {0x5B9F, 0x5B9F, "0x5B9F", "WaitOp", "DefWait := Wait",
211     {0x5BA0, 0x5BA0, "0x5BA0", "ResetOp", "TermObject",
212     {0x5BA1, 0x5BA1, "0x5BA1", "WaitOp", "DefWait := Wait",
213     {0x5BA2, 0x5BA2, "0x5BA2", "ResetOp", "TermObject",
214     {0x5BA3, 0x5BA3, "0x5BA3", "WaitOp", "DefWait := Wait",
215     {0x5BA4, 0x5BA4, "0x5BA4", "ResetOp", "TermObject",
216     {0x5BA5, 0x5BA5, "0x5BA5", "WaitOp", "DefWait := Wait",
217     {0x5BA6, 0x5BA6, "0x5BA6", "ResetOp", "TermObject",
218     {0x5BA7, 0x5BA7, "0x5BA7", "WaitOp", "DefWait := Wait",
219     {0x5BA8, 0x5BA8, "0x5BA8", "ResetOp", "TermObject",
220     {0x5BA9, 0x5BA9, "0x5BA9", "WaitOp", "DefWait := Wait",
221     {0x5BAA, 0x5BAA, "0x5BAA", "ResetOp", "TermObject",
222     {0x5BAB, 0x5BAB, "0x5BAB", "WaitOp", "DefWait := Wait",
223     {0x5BAC, 0x5BAC, "0x5BAC", "ResetOp", "TermObject",
224     {0x5BAD, 0x5BAD, "0x5BAD", "WaitOp", "DefWait := Wait",
225     {0x5BAE, 0x5BAE, "0x5BAE", "ResetOp", "TermObject",
226     {0x5BAF, 0x5BAF, "0x5BAF", "WaitOp", "DefWait := Wait",
227     {0x5BB0, 0x5BB0, "0x5BB0", "ResetOp", "TermObject",
228     {0x5BB1, 0x5BB1, "0x5BB1", "WaitOp", "DefWait := Wait",
229     {0x5BB2, 0x5BB2, "0x5BB2", "ResetOp", "TermObject",
230     {0x5BB3, 0x5BB3, "0x5BB3", "WaitOp", "DefWait := Wait",
231     {0x5BB4, 0x5BB4, "0x5BB4", "ResetOp", "TermObject",
232     {0x5BB5, 0x5BB5, "0x5BB5", "WaitOp", "DefWait := Wait",
233     {0x5BB6, 0x5BB6, "0x5BB6", "ResetOp", "TermObject",
234     {0x5BB7, 0x5BB7, "0x5BB7", "WaitOp", "DefWait := Wait",
235     {0x5BB8, 0x5BB8, "0x5BB8", "ResetOp", "TermObject",
236     {0x5BB9, 0x5BB9, "0x5BB9", "WaitOp", "DefWait := Wait",
237     {0x5BBA, 0x5BBA, "0x5BBA", "ResetOp", "TermObject",
238     {0x5BBB, 0x5BBB, "0x5BBB", "WaitOp", "DefWait := Wait",
239     {0x5BBC, 0x5BBC, "0x5BBC", "ResetOp", "TermObject",
240     {0x5BBD, 0x5BBD, "0x5BBD", "WaitOp", "DefWait := Wait",
241     {0x5BBE, 0x5BBE, "0x5BBE", "ResetOp", "TermObject",
242     {0x5BBF, 0x5BBF, "0x5BBF", "WaitOp", "DefWait := Wait",
243     {0x5BC0, 0x5BC0, "0x5BC0", "ResetOp", "TermObject",
244     {0x5BC1, 0x5BC1, "0x5BC1", "WaitOp", "DefWait := Wait",
245     {0x5BC2, 0x5BC2, "0x5BC2", "ResetOp", "TermObject",
246     {0x5BC3, 0x5BC3, "0x5BC3", "WaitOp", "DefWait := Wait",
247     {0x5BC4, 0x5BC4, "0x5BC4", "ResetOp", "TermObject",
248     {0x5BC5, 0x5BC5, "0x5BC5", "WaitOp", "DefWait := Wait",
249     {0x5BC6, 0x5BC6, "0x5BC6", "ResetOp", "TermObject",
250     {0x5BC7, 0x5BC7, "0x5BC7", "WaitOp", "DefWait := Wait",
251     {0x5BC8, 0x5BC8, "0x5BC8", "ResetOp", "TermObject",
252     {0x5BC9, 0x5BC9, "0x5BC9", "WaitOp", "DefWait := Wait",
253     {0x5BCA, 0x5BCA, "0x5BCA", "ResetOp", "TermObject",
254     {0x5BCB, 0x5BCB, "0x5BCB", "WaitOp", "DefWait := Wait",
255     {0x5BCC, 0x5BCC, "0x5BCC", "ResetOp", "TermObject",
256     {0x5BCD, 0x5BCD, "0x5BCD", "WaitOp", "DefWait := Wait",
257     {0x5BCE, 0x5BCE, "0x5BCE", "ResetOp", "TermObject",
258     {0x5BCF, 0x5BCF, "0x5BCF", "WaitOp", "DefWait := Wait",
259     {0x5BD0, 0x5BD0, "0x5BD0", "ResetOp", "TermObject",
260     {0x5BD1, 0x5BD1, "0x5BD1", "WaitOp", "DefWait := Wait",
261     {0x5BD2, 0x5BD2, "0x5BD2", "ResetOp", "TermObject",
262     {0x5BD3, 0x5BD3, "0x5BD3", "WaitOp", "DefWait := Wait",
263     {0x5BD4, 0x5BD4, "0x5BD4", "ResetOp", "TermObject",
264     {0x5BD5, 0x5BD5, "0x5BD5", "WaitOp", "DefWait := Wait",
265     {0x5BD6, 0x5BD6, "0x5BD6", "ResetOp", "TermObject",
266     {0x5BD7, 0x5BD7, "0x5BD7", "WaitOp", "DefWait := Wait",
267     {0x5BD8, 0x5BD8, "0x5BD8", "ResetOp", "TermObject",
268     {0x5BD9, 0x5BD9, "0x5BD9", "WaitOp", "DefWait := Wait",
269     {0x5BDA, 0x5BDA, "0x5BDA", "ResetOp", "TermObject",
270     {0x5BDB, 0x5BDB, "0x5BDB", "WaitOp", "DefWait := Wait",
271     {0x5BDC, 0x5BDC, "0x5BDC", "ResetOp", "TermObject",
272     {0x5BDD, 0x5BDD, "0x5BDD", "WaitOp", "DefWait := Wait",
273     {0x5BDE, 0x5BDE, "0x5BDE", "ResetOp", "TermObject",
274     {0x5BDF, 0x5BDF, "0x5BDF", "WaitOp", "DefWait := Wait",
275     {0x5BE0, 0x5BE0, "0x5BE0", "ResetOp", "TermObject",
276     {0x5BE1, 0x5BE1, "0x5BE1", "WaitOp", "DefWait := Wait",
277     {0x5BE2, 0x5BE2, "0x5BE2", "ResetOp", "TermObject",
278     {0x5BE3, 0x5BE3, "0x5BE3", "WaitOp", "DefWait := Wait",
279     {0x5BE4, 0x5BE4, "0x5BE4", "ResetOp", "TermObject",
280     {0x5BE5, 0x5BE5, "0x5BE5", "WaitOp", "DefWait := Wait",
281     {0x5BE6, 0x5BE6, "0x5BE6", "ResetOp", "TermObject",
282     {0x5BE7, 0x5BE7, "0x5BE7", "WaitOp", "DefWait := Wait",
283     {0x5BE8, 0x5BE8, "0x5BE8", "ResetOp", "TermObject",
284     {0x5BE9, 0x5BE9, "0x5BE9", "WaitOp", "DefWait := Wait",
285     {0x5BEA, 0x5BEA, "0x5BEA", "ResetOp", "TermObject",
286     {0x5BED, 0x5BED, "0x5BED", "WaitOp", "DefWait := Wait",
287     {0x5BEC, 0x5BEC, "0x5BEC", "ResetOp", "TermObject",
288     {0x5BED, 0x5BED, "0x5BED", "WaitOp", "DefWait := Wait",
289     {0x5BEE, 0x5BEE, "0x5BEE", "ResetOp", "TermObject",
290     {0x5BEF, 0x5BEF, "0x5BEF", "WaitOp", "DefWait := Wait",
291     {0x5BF0, 0x5BF0, "0x5BF0", "ResetOp", "TermObject",
292     {0x5BF1, 0x5BF1, "0x5BF1", "WaitOp", "DefWait := Wait",
293     {0x5BF2, 0x5BF2, "0x5BF2", "ResetOp", "TermObject",
294     {0x5BF3, 0x5BF3, "0x5BF3", "WaitOp", "DefWait := Wait",
295     {0x5BF4, 0x5BF4, "0x5BF4", "ResetOp", "TermObject",
296     {0x5BF5, 0x5BF5, "0x5BF5", "WaitOp", "DefWait := Wait",
297     {0x5BF6, 0x5BF6, "0x5BF6", "ResetOp", "TermObject",
298     {0x5BF7, 0x5BF7, "0x5BF7", "WaitOp", "DefWait := Wait",
299     {0x5BF8, 0x5BF8, "0x5BF8", "ResetOp", "TermObject",
300     {0x5BF9, 0x5BF9, "0x5BF9", "WaitOp", "DefWait := Wait",
301     {0x5BFA, 0x5BFA, "0x5BFA", "ResetOp", "TermObject",
302     {0x5BFB, 0x5BFB, "0x5BFB", "WaitOp", "DefWait := Wait",
303     {0x5BFC, 0x5BFC, "0x5BFC", "ResetOp", "TermObject",
304     {0x5BFD, 0x5BFD, "0x5BFD", "WaitOp", "DefWait := Wait",
305     {0x5BFE, 0x5BFE, "0x5BFE", "ResetOp", "TermObject",
306     {0x5BFF, 0x5BFF, "0x5BFF", "WaitOp", "DefWait := Wait",
307     {0x5C00, 0x5C00, "0x5C00", "ResetOp", "TermObject",
308     {0x5C01, 0x5C01, "0x5C01", "WaitOp", "DefWait := Wait",
309     {0x5C02, 0x5C02, "0x5C02", "ResetOp", "TermObject",
310     {0x5C03, 0x5C03, "0x5C03", "WaitOp", "DefWait := Wait",
311     {0x5C04, 0x5C04, "0x5C04", "ResetOp", "TermObject",
312     {0x5C05, 0x5C05, "0x5C05", "WaitOp", "DefWait := Wait",
313     {0x5C06, 0x5C06, "0x5C06", "ResetOp", "TermObject",
314     {0x5C07, 0x5C07, "0x5C07", "WaitOp", "DefWait := Wait",
315     {0x5C08, 0x5C08, "0x5C08", "ResetOp", "TermObject",
316     {0x5C09, 0x5C09, "0x5C09", "WaitOp", "DefWait := Wait",
317     {0x5C0A, 0x5C0A, "0x5C0A", "ResetOp", "TermObject",
318     {0x5C0B, 0x5C0B, "0x5C0B", "WaitOp", "DefWait := Wait",
319     {0x5C0C, 0x5C0C, "0x5C0C", "ResetOp", "TermObject",
320     {0x5C0D, 0x5C0D, "0x5C0D", "WaitOp", "DefWait := Wait",
321     {0x5C0E, 0x5C0E, "0x5C0E", "ResetOp", "TermObject",
322     {0x5C0F, 0x5C0F, "0x5C0F", "WaitOp", "DefWait := Wait",
323     {0x5C10, 0x5C10, "0x5C10", "ResetOp", "TermObject",
324     {0x5C11, 0x5C11, "0x5C11", "WaitOp", "DefWait := Wait",
325     {0x5C12, 0x5C12, "0x5C12", "ResetOp", "TermObject",
326     {0x5C13, 0x5C13, "0x5C13", "WaitOp", "DefWait := Wait",
327     {0x5C14, 0x5C14, "0x5C14", "ResetOp", "TermObject",
328     {0x5C15, 0x5C15, "0x5C15", "WaitOp", "DefWait := Wait",
329     {0x5C16, 0x5C16, "0x5C16", "ResetOp", "TermObject",
330     {0x5C17, 0x5C17, "0x5C17", "WaitOp", "DefWait := Wait",
331     {0x5C18, 0x5C18, "0x5C18", "ResetOp", "TermObject",
332     {0x5C19, 0x5C19, "0x5C19", "WaitOp", "DefWait := Wait",
333     {0x5C1A, 0x5C1A, "0x5C1A", "ResetOp", "TermObject",
334     {0x5C1B, 0x5C1B, "0x5C1B", "WaitOp", "DefWait := Wait",
335     {0x5C1C, 0x5C1C, "0x5C1C", "ResetOp", "TermObject",
336     {0x5C1D, 0x5C1D, "0x5C1D", "WaitOp", "DefWait := Wait",
337     {0x5C1E, 0x5C1E, "0x5C1E", "ResetOp", "TermObject",
338     {0x5C1F, 0x5C1F, "0x5C1F", "WaitOp", "DefWait := Wait",
339     {0x5C20, 0x5C20, "0x5C20", "ResetOp", "TermObject",
340     {0x5C21, 0x5C21, "0x5C21", "WaitOp", "DefWait := Wait",
341     {0x5C22, 0x5C22, "0x5C22", "ResetOp", "TermObject",
342     {0x5C23, 0x5C23, "0x5C23", "WaitOp", "DefWait := Wait",
343     {0x5C24, 0x5C24, "0x5C24", "ResetOp", "TermObject",
344     {0x5C25, 0x5C25, "0x5C25", "WaitOp", "DefWait := Wait",
345     {0x5C26, 0x5C26, "0x5C26", "ResetOp", "TermObject",
346     {0x5C27, 0x5C27, "0x5C27", "WaitOp", "DefWait := Wait",
347     {0x5C28, 0x5C28, "0x5C28", "ResetOp", "TermObject",
348     {0x5C29, 0x5C29, "0x5C29", "WaitOp", "DefWait := Wait",
349     {0x5C2A, 0x5C2A, "0x5C2A", "ResetOp", "TermObject",
350     {0x5C2B, 0x5C2B, "0x5C2B", "WaitOp", "DefWait := Wait",
351     {0x5C2C, 0x5C2C, "0x5C2C", "ResetOp", "TermObject",
352     {0x5C2D, 0x5C2D, "0x5C2D", "WaitOp", "DefWait := Wait",
353     {0x5C2E, 0x5C2E, "0x5C2E", "ResetOp", "TermObject",
354     {0x5C2F, 0x5C2F, "0x5C2F", "WaitOp", "DefWait := Wait",
355     {0x5C30, 0x5C30, "0x5C30", "ResetOp", "TermObject",
356     {0x5C31, 0x5C31, "0x5C31", "WaitOp", "DefWait := Wait",
357     {0x5C32, 0x5C32, "0x5C32", "ResetOp", "TermObject",
358     {0x5C33, 0x5C33, "0x5C33", "WaitOp", "DefWait := Wait",
359     {0x5C34, 0x5C34, "0x5C34", "ResetOp", "TermObject",
360     {0x5C35, 0x5C35, "0x5C35", "WaitOp", "DefWait := Wait",
361     {0x5C36, 0x5C36, "0x5C36", "ResetOp", "TermObject",
362     {0x5C37, 0x5C37, "0x5C37", "WaitOp", "DefWait := Wait",
363     {0x5C38, 0x5C38, "0x5C38", "ResetOp", "TermObject",
364     {0x5C39, 0x5C39, "0x5C39", "WaitOp", "DefWait := Wait",
365     {0x5C3A, 0x5C3A, "0x5C3A", "ResetOp", "TermObject",
366     {0x5C3B, 0x5C3B, "0x5C3B", "WaitOp", "DefWait := Wait",
367     {0x5C3C, 0x5C3C, "0x5C3C", "ResetOp", "TermObject",
368     {0x5C3D, 0x5C3D, "0x5C3D", "WaitOp", "DefWait := Wait",
369     {0x5C3E, 0x5C3E, "0x5C3E", "ResetOp", "TermObject",
370     {0x5C3F, 0x5C3F, "0x5C3F", "WaitOp", "DefWait := Wait
```

```

127 {0x5B27, 0x5B27, "0x5B27", "ReleaseOp", "DefReset := Res
128 {0x5B28, 0x5B28, "0x5B28", "FromBCDOp", "TermObject",
129 {0x5B29, 0x5B29, "0x5B29", "ToBCD", "DefRelease := R
130 {0x5B2A, 0x5B2A, "0x5B2A", "UnloadOp", "TermObject",
131 {0x5B30, 0x5B30, "0x5B30", "RevisionOp", "DefFromBCD := F
132 {0x5B31, 0x5B31, "0x5B31", "DebugOp", "TermObject",
133 {0x5B32, 0x5B32, "0x5B32", "FatalOp", "DefToBCD := ToB
134 {0x5B33, 0x5B33, "0x5B33", "TimerOp", "TermObject",
135 {0x5B80, 0x5B80, "0x5B80", "OpRegionOp", "DefUnload := Un
136 {0x5B81, 0x5B81, "0x5B81", "FieldOp", "DataObject",
137 {0x5B82, 0x5B82, "0x5B82", "DeviceOp", NULL,
138 {0x5B83, 0x5B83, "0x5B83", "ProcessorOp", "DebugObject",
139 {0x5B84, 0x5B84, "0x5B84", "PowerResOp", NULL,
140 {0x5B85, 0x5B85, "0x5B85", "ThermalZoneOp", "TermObject",
141 {0x5B86, 0x5B86, "0x5B86", "IndexFieldOp", "DefFatal := Fat
142 {0x5B87, 0x5B87, "0x5B87", "BankFieldOp", "TermObject",
143 {0x5B88, 0x5B88, "0x5B88", "DataRegionOp", "DefTimer := Tim
144 {0x5B89, 0x5B89, "0x5B89-0x5BFF", NULL, "DefOpRegion :=
145 {0x5C, 0x5C, "0x5C", "RootChar", "TermObject",
146 {0x5D, 0x5D, "0x5D", NULL, "DefField := Fie
147 {0x5E, 0x5E, "0x5E", "ParentPrefixChar", "TermObject",
148 {0x5F, 0x5F, "0x5F", "NameChar", "DefDevice := De
149 {0x60, 0x60, "0x60", "Local0Op", "TermObject",
150 {0x61, 0x61, "0x61", "Local1Op", "DefProcessor :=
151 {0x62, 0x62, "0x62", "Local2Op", "DefPowerRes :=
152 {0x63, 0x63, "0x63", "Local3Op", "TermObject",
153 {0x64, 0x64, "0x64", "Local4Op", "DefThermalZone
154 {0x65, 0x65, "0x65", "Local5Op", "TermObject",
155 {0x66, 0x66, "0x66", "Local6Op", "DefIndexField :
156 {0x67, 0x67, "0x67", "Local7Op", "TermObject",
157 {0x68, 0x68, "0x68", "Arg0Op", "DefBankField :=
158 {0x69, 0x69, "0x69", "Arg1Op", "TermObject",
159 {0x6A, 0x6A, "0x6A", "Arg2Op", "DefDataRegion :

```

```

193 {0x6B, 0x6B, "0x6B", "Arg3Op", NULL,
194 {0x6C, 0x6C, "0x6C", "Arg4Op", "ArgObject",
195 {0x6D, 0x6D, "0x6D", "Arg5Op", NULL,
196 {0x6E, 0x6E, "0x6E", "Arg6Op", "ArgObject",
197 {0x6F, 0x6F, "0x6F", NULL, NULL,
198 {0x70, 0x70, "0x70", "StoreOp", "TermObject",
199 {0x71, 0x71, "0x71", "RefOfOp", "DefStore := Sto
200 {0x72, 0x72, "0x72", "AddOp", "TermObject",
201 {0x73, 0x73, "0x73", "ConcatOp", "DefRefOf := Ref
202 {0x74, 0x74, "0x74", "SubtractOp", "DefAdd := AddOp
203 {0x75, 0x75, "0x75", "IncrementOp", "TermObject",
204 {0x76, 0x76, "0x76", "DecrementOp", "DefConcat := Co
205 {0x77, 0x77, "0x77", "MultiplyOp", "TermObject",
206 {0x78, 0x78, "0x78", "DivideOp", "DefSubtract :=
207 {0x79, 0x79, "0x79", "ShiftLeftOp", "TermObject",
208 {0x7A, 0x7A, "0x7A", "ShiftRightOp", "DefMultiply :=
209 {0x7B, 0x7B, "0x7B", "AndOp", "TermObject",
210 {0x7C, 0x7C, "0x7C", "NandOp", "DefDivide := Di
211 {0x7D, 0x7D, "0x7D", "OrOp", "TermObject",
212 {0x7E, 0x7E, "0x7E", "NorOp", "DefShiftLeft :=
213 {0x7F, 0x7F, "0x7F", "XorOp", "TermObject",
214 {0x80, 0x80, "0x80", "NotOp", "DefShiftRight :
215 {0x81, 0x81, "0x81", "FindSetLeftBitOp", "TermObject",
216 {0x82, 0x82, "0x82", "FindSetRightBitOp", "DefFindSetLeftB
217 {0x83, 0x83, "0x83", "DerefOfOp", "TermObject",
218 {0x84, 0x84, "0x84", "ConcatResOp", "DefFindSetRight
219 {0x85, 0x85, "0x85", "ModOp", "DefDerefOf := D
220 {0x86, 0x86, "0x86", "NotifyOp", "TermObject",
221 {0x87, 0x87, "0x87", "SizeOfOp", "DefConcatRes :=
222 {0x88, 0x88, "0x88", "IndexOp", "TermObject",
223 {0x89, 0x89, "0x89", "MatchOp", "DefMod := ModOp
224 {0x8A, 0x8A, "0x8A", "CreateDWordFieldOp", "TermObject",
225 {0x8B, 0x8B, "0x8B", "CreateWordFieldOp", "DefNotify := No

```

```

259 {0x8C, 0x8C, "0x8C", "CreateByteFieldOp", "DefCreateWordFi
260 {0x8D, 0x8D, "0x8D", "CreateBitFieldOp", "TermObject",
261 {0x8E, 0x8E, "0x8E", "ObjectTypeOp", "DefCreateByteFi
262 {0x8F, 0x8F, "0x8F", "CreateQWordFieldOp", "TermObject",
263 {0x90, 0x90, "0x90", "LAndOp", "DefCreateQWordF
264 {0x91, 0x91, "0x91", "LOrOp", "TermObject",
265 {0x92, 0x92, "0x92", "LNotOp", "DefLAnd := Land
266 {0x9293, 0x9293, "0x9293", "LNotEqualOp", "TermObject",
267 {0x9294, 0x9294, "0x9294", "LLessEqualOp", "DefLNot := Lnot
268 {0x9295, 0x9295, "0x9295", "LGreaterEqualOp", "TermObject",
269 {0x93, 0x93, "0x93", "LEqualOp", "DefLLessEqual :=
270 {0x94, 0x94, "0x94", "LGreaterOp", "TermObject",
271 {0x95, 0x95, "0x95", "LLessOp", "DefLGreaterEqua
272 {0x96, 0x96, "0x96", "ToBufferOp", "TermObject",
273 {0x97, 0x97, "0x97", "ToDecimalStringOp", "DefLLess := Lle
274 {0x98, 0x98, "0x98", "ToHexStringOp", "TermObject",
275 {0x99, 0x99, "0x99", "ToIntegerOp", "DefToBuffer :=
276 {0x9A, 0x9B, "0x9A-0x9B", NULL, "TermObject",
277 {0x9C, 0x9C, "0x9C", "ToStringOp", "DefToDecimalStr
278 {0x9D, 0x9D, "0x9D", "CopyObjectOp", "TermObject",
279 {0x9E, 0x9E, "0x9E", "MidOp", "DefToHexString
280 {0x9F, 0x9F, "0x9F", "ContinueOp", "TermObject",
281 {0xA0, 0xA0, "0xA0", "IfOp", "DefContinue :=
282 {0xA1, 0xA1, "0xA1", "ElseOp", "TermObject",
283 {0xA2, 0xA2, "0xA2", "WhileOp", "DefIfElse := If
284 {0xA3, 0xA3, "0xA3", "NoopOp", "TermObject",
285 {0xA4, 0xA4, "0xA4", "ReturnOp", "DefElse := Noth
286 {0xA5, 0xA5, "0xA5", "BreakOp", "TermObject",
287 {0xA6, 0xCB, "0xA6-0xCB", NULL, "DefWhile := Whi
288 {0xCC, 0xCC, "0xCC", "BreakPointOp", "TermObject",
289 {0xCD, 0xFE, "0xCD-0xFE", NULL, "DefNoop := Noop
290 {0xFF, 0xFF, "0xFF", "OnesOp", "TermObject",
291 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, "DefReturn := Re
292 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, "TermObject",
293 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, "DefBreak := Bre
294 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
295 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
296 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
297 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
298 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
299 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
300 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
301 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
302 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
303 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
304 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
305 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
306 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
307 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
308 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
309 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
310 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
311 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
312 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
313 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
314 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
315 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
316 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
317 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
318 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
319 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
320 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
321 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
322 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
323 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
324 {0, 0, NULL, NULL, NULL, NULL, NULL, NULL, NULL,

```

```

325 };

```

```

*****
7842 Fri Jan 17 07:31:16 2014
new/usr/src/common/acpica/tools/acpihelp/ahaslkey.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: ahaslkey - Table of all known ASL non-operator keywords
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "acpihelp.h"
45
46 /*
47 * ASL Keyword types and associated actual keywords.
48 * This table was extracted from the ACPI specification.
49 */
50 const AH_AS_L_KEYWORD AslKeywordInfo[] =
51 {
52     {"AccessAttribKeyword", "Serial Bus Attributes (with legacy SMBus aliases)",
53      ":= AttribQuick (SMBusQuick) | AttribSendReceive (SMBusSendReceive) | "
54      "AttribByte (SMBusByte) | AttribWord (SMBusWord) | "
55      "AttribBlock (SMBusBlock) | AttribProcessCall (SMBusProcessCall) | "
56      "AttribBlockProcessCall (SMBusProcessCall)"},
57     {"AccessTypeKeyword", "Field Access Types",
58      ":= AnyAcc | ByteAcc | WordAcc | DWordAcc | QWordAcc | BufferAcc"},
59     {"AddressingModeKeyword", "Mode - Resource Descriptors",
60      ":= AddressingMode7Bit | AddressingMode10Bit"},

```

```

61     {"AddressKeyword", "ACPI memory range types",
62      ":= AddressRangeMemory | AddressRangeReserved | "
63      "AddressRangeNVS | AddressRangeACPI"},
64     {"AddressSpaceKeyword", "Operation Region Address Space Types",
65      ":= RegionSpaceKeyword | FixedHW"},
66     {"BusMasterKeyword", "DMA Bus Mastering",
67      ":= BusMaster | NotBusMaster"},
68     {"ByteLengthKeyword", "Bits per Byte - Resource Descriptors",
69      ":= DataBitsFive | DataBitsSix | DataBitsSeven | DataBitsEight | DataBit"},
70     {"ClockPhaseKeyword", "Resource Descriptors",
71      ":= ClockPhaseFirst | ClockPhaseSecond"},
72     {"ClockPolarityKeyword", "Resource Descriptors",
73      ":= ClockPolarityLow | ClockPolarityHigh"},
74     {"DecodeKeyword", "Type of Memory Decoding - Resource Descriptors",
75      ":= SubDecode | PosDecode"},
76     {"DmaTypeKeyword", "DMA Types - DMA Resource Descriptor",
77      ":= Compatibility | TypeA | TypeB | TypeF"},
78     {"EndianKeyword", "Endian type - Resource Descriptor",
79      ":= BigEndian | LittleEndian"},
80     {"ExtendedAttribKeyword", "Extended Bus Attributes",
81      ":= AttribBytes (AccessLength) | AttribRawBytes (AccessLength) | "
82      "AttribRawProcessBytes (AccessLength)"},
83     {"FlowControlKeyword", "Resource Descriptor",
84      ":= FlowControlNone | FlowControlXon | FlowControlHardware"},
85     {"InterruptLevelKeyword", "Interrupt Active Types",
86      ":= ActiveHigh | ActiveLow | ActiveBoth"},
87     {"InterruptTypeKeyword", "Interrupt Types",
88      ":= Edge | Level"},
89     {"IoDecodeKeyword", "I/O Decoding - IO Resource Descriptor",
90      ":= Decode16 | Decode10"},
91     {"IoRestrictionKeyword", "I/O Restriction - GPIO Resource Descriptors",
92      ":= IoRestrictionNone | IoRestrictionInputOnly | "
93      "IoRestrictionOutputOnly | IoRestrictionNoneAndPreserve"},
94     {"LockRuleKeyword", "Global Lock use for Field Operator",
95      ":= Lock | NoLock"},
96     {"MatchOpKeyword", "Types for Match Operator",
97      ":= MTR | MEQ | MLE | MLT | MGE | MGT"},
98     {"MaxKeyword", "Max Range Type - Resource Descriptors",
99      ":= MaxFixed | MaxNotFixed"},
100     {"MemTypeKeyword", "Memory Types - Resource Descriptors",
101      ":= Cacheable | WriteCombining | Prefetchable | NonCacheable"},
102     {"MinKeyword", "Min Range Type - Resource Descriptors",
103      ":= MinFixed | MinNotFixed"},
104     {"ObjectTypeKeyword", "ACPI Object Types",
105      ":= UnknownObj | IntObj | StrObj | BuffObj | PkgObj | FieldUnitObj | "
106      "DeviceObj | EventObj | MethodObj | MutexObj | OpRegionObj | PowerResObj | "
107      "ProcessorObj | ThermalZoneObj | BuffFieldObj | DDBHandleObj"},
108     {"ParityKeyword", "Resource Descriptors",
109      ":= ParityTypeNone | ParityTypeSpace | ParityTypeMark | "
110      "ParityTypeOdd | ParityTypeEven"},
111     {"PinConfigKeyword", "Pin Configuration - GPIO Resource Descriptors",
112      ":= PullDefault | PullUp | PullDown | PullNone"},
113     {"PolarityKeyword", "Resource Descriptors",
114      ":= PolarityHigh | PolarityLow"},
115     {"RangeTypeKeyword", "I/O Range Types - Resource Descriptors",
116      ":= ISAOnlyRanges | NonISAOnlyRanges | EntireRange"},
117     {"ReadWriteKeyword", "Memory Access Types - Resource Descriptors",
118      ":= ReadWrite | ReadOnly"},
119     {"RegionSpaceKeyword", "Operation Region Address Space Types",
120      ":= UserDefRegionSpace | SystemIO | SystemMemory | PCI_Config | "
121      "EmbeddedControl | SMBus | SystemCMOS | PciBarTarget | IPMI | "
122      "GeneralPurposeIo, GenericSerialBus"},
123     {"ResourceTypeKeyword", "Resource Usage - Resource Descriptors",
124      ":= ResourceConsumer | ResourceProducer"},
125     {"SerializeRuleKeyword", "Control Method Serialization",
126      ":= Serialized | NotSerialized"},

```

```
127 { "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 };
```

new/usr/src/common/acpica/tools/acpihelp/ahaslops.c

1

```
*****
28994 Fri Jan 17 07:31:16 2014
new/usr/src/common/acpica/tools/acpihelp/ahaslops.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: ahaslops - Table of all known ASL operators
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "acpihelp.h"
45
46 /*
47 * ASL operators with syntax (directly from ACPI specification).
48 * Note: All tokens require a space separator.
49 * Long lines are automatically split during output.
50 */
51 const AH_AS_L_OPERATOR AslOperatorInfo[] =
52 {
53     {"AccessAs", "(AccessType, AccessAttribKeyword | "
54     "ExtendedAttribKeyword (AccessLength))",
55     "ChangeFieldUnitAccess"},
56     {"Acquire", "(SyncObject, TimeoutValue) => Boolean",
57     "Acquire a mutex"},
58     {"Add", "(Addend1, Addend2, Result) => Integer",
59     "Integer Add"},
60     {"Alias", "(SourceObject, AliasObject)",
```

new/usr/src/common/acpica/tools/acpihelp/ahaslops.c

2

```
61 "Define a name alias"},
62 {"And", "(Source1, Source2, Result) => Integer",
63 "Integer Bitwise And"},
64 {"Arg", "Arg0 - Arg6",
65 "Method argument data objects"},
66 {"BankField", "(RegionName, BankName, BankValue, "
67 "AccessTypeKeyword, LockRuleKeyword, "
68 "UpdateRuleKeyword) {FieldUnitList}",
69 "Declare fields in a banked configuration object"},
70 {"Break", "No parameters",
71 "Continue following the innermost enclosing Whil"},
72 {"BreakPoint", "No parameters",
73 "Used for debugging, stops execution in the debu"},
74 {"Buffer", "(BufferSize) {String or ByteList} => Buffer",
75 "Declare Buffer object"},
76 {"Case", "(Value) {TermList}",
77 "Expression for conditional execution"},
78 {"Concatenate", "(Source1, Source2, Result) => ComputationalData",
79 "Concatenate two strings, integers or buffers"},
80 {"ConcatenateResTemplate", "(Source1, Source2, Result) => Buffer",
81 "Concatenate two resource templates"},
82 {"CondRefOf", "(Source, Result) => Boolean",
83 "Conditional reference to an object"},
84 {"Connection", "(ResourceMacro)",
85 "Associate connection with FieldUnits within a F"},
86 {"Continue", "No parameters",
87 "Continue innermost enclosing While loop"},
88 {"CopyObject", "(Source, Destination) => DataRefObject",
89 "Copy and existing object"},
90 {"CreateBitField", "(SourceBuffer, BitIndex, BitFieldName)",
91 "Declare a bit field object of a buffer object"},
92 {"CreateByteField", "(SourceBuffer, ByteIndex, ByteFieldName)",
93 "Declare a byte field object of a buffer object"},
94 {"CreateDWordField", "(SourceBuffer, ByteIndex, DWordFieldName)",
95 "Declare a DWord field object of a buffer object"},
96 {"CreateField", "(SourceBuffer, BitIndex, NumBits, FieldName)",
97 "Declare an arbitrary length bit field of a buff"},
98 {"CreateQWordField", "(SourceBuffer, ByteIndex, QWordFieldName)",
99 "Declare a QWord field object of a buffer object"},
100 {"CreateWordField", "(SourceBuffer, ByteIndex, WordFieldName)",
101 "Declare a Word field object of a buffer object"},
102 {"DataTableRegion", "(RegionName, SignatureString, OemIDString, OemT",
103 "Declare a Data Table Region"},
104 {"Debug", "No parameters",
105 "Debugger output"},
106 {"Decrement", "(Minuend) => Integer",
107 "Decrement an Integer"},
108 {"Default", "{TermList}",
109 "Default execution path in Switch()"},
110 {"DefinitionBlock", "(AmlFileName, TableSignature, ComplianceRevisio",
111 "OemId, TableId, OemRevision) {TermList}",
112 "Declare a Definition Block"},
113 {"DerefOf", "(Source) => Object",
114 "Dereference an object reference"},
115 {"Device", "(DeviceName) {ObjectList}",
116 "Declare a bus/device object"},
117 {"Divide", "(Dividend, Divisor, Remainder, Result) => Integ",
118 "Integer Divide"},
119 {"DMA", "(DmaTypeKeyword, BusMasterKeyword, XferTypeKeyw",
120 "DescriptorName) {DmaChannelList} => Buffer",
121 "DMA Resource Descriptor macro"},
122 {"DWordIO", "(ResourceTypeKeyword, MinKeyword, MaxKeyword, "
123 "DecodeKeyword, RangeTypeKeyword, AddressGranula",
124 "AddressMinimum, AddressMaximum, AddressTranslat",
125 "RangeLength, ResourceSourceIndex, "
126 "ResourceSource, DescriptorName, TypeKeyword, Tr
```



```

127 "DWord IO Resource Descriptor macro",
128 {"DWordMemory", "(ResourceTypeKeyword, DecodeKeyword, MinKeyword
129 "MaxKeyword, MemTypeKeyword, ReadWriteKeyword, "
130 "AddressGranularity, AddressMinimum, AddressMaxi
131 "RangeLength, ResourceSourceIndex, ResourceSourc
132 "TypeKeyword)",
133 "DWord Memory Resource Descriptor macro",
134 {"DWordSpace", "(ResourceType, ResourceTypeKeyword, DecodeKeywo
135 "MinKeyword, MaxKeyword, TypeSpecificFlags, "
136 "AddressGranularity, AddressMinimum, AddressMaxi
137 "AddressTranslation, RangeLength, "
138 "ResourceSourceIndex, ResourceSource, Descriptor
139 "DWord Space Resource Descriptor macro",
140 {"EISAID", "(EisaIdString) => DWordConst",
141 "EISA ID String to Integer conversion macro",
142 {"Else", "{TermList}",
143 "Alternate conditional execution",
144 {"ElseIf", "(Predicate)",
145 "Conditional execution",
146 {"EndDependentFn", "()" => Buffer",
147 "End Dependent Function Resource Descriptor macr
148 {"Event", "(EventName)",
149 "Declare an event synchronization object",
150 {"ExtendedIO", "(ResourceTypeKeyword, MinKeyword, MaxKeyword, "
151 "DecodeKeyword, RangeTypeKeyword, AddressGranula
152 "AddressMinimum, AddressMaximum, AddressTranslat
153 "TypeSpecificAttributes, DescriptorName, TypeKey
154 "Extended IO Resource Descriptor macro",
155 {"ExtendedMemory", "(ResourceTypeKeyword, DecodeKeyword, MinKeyword
156 "MaxKeyword, MemTypeKeyword, ReadWriteKeyword, "
157 "AddressGranularity, AddressMinimum, AddressMaxi
158 "RangeLength, TypeSpecificAttributes, Descriptor
159 "AddressKeyword, TypeKeyword)",
160 "Extended Memory Resource Descriptor macro",
161 {"ExtendedSpace", "(ResourceType, ResourceTypeKeyword, DecodeKeywo
162 "MinKeyword, MaxKeyword, TypeSpecificFlags, "
163 "AddressGranularity, AddressMinimum, AddressMaxi
164 "RangeLength, TypeSpecificAttributes, Descriptor
165 "Extended Space Resource Descriptor macro",
166 {"External", "(ObjectName, ObjectTypeKeyword, ReturnTypes, Par
167 "Declare external objects",
168 {"Fatal", "(Type, Code, Arg)",
169 "Fatal error check",
170 {"Field", "(RegionName, AccessTypeKeyword, LockRuleKeyword
171 "UpdateRuleKeyword) {FieldUnitList}",
172 "Declare fields of an operation region object",
173 {"FindSetLeftBit", "(Source, Result) => Integer",
174 "Index of first least significant bit set",
175 {"FindSetRightBit", "(Source, Result) => Integer",
176 "Index of first most significant bit set",
177 {"FixedDMA", "(DmaRequestLine, Channel, TransferWidthKeyword,
178 "Fixed DMA Resource Descriptor macro",
179 {"FixedIO", "(AddressBase, RangeLength, DescriptorName) => B
180 "Fixed I/O Resource Descriptor macro",
181 {"FromBCD", "(BCDValue, Result) => Integer",
182 "Convert from BCD to numeric",
183 {"Function", "(FunctionName, ReturnTypes, ParameterTypes) {Ter
184 "Declare control method",
185 {"GpioInt", "(InterruptTypeKeyword, InterruptLevelKeyword, "
186 "ShareTypeKeyword, PinConfigKeyword, "
187 "DebounceTimeout, ResourceSource, "
188 "ResourceSourceIndex, ResourceTypeKeyword, Descr
189 "RawDataBuffer() {VendorData}) {Pin}",
190 "GPIO Interrupt Connection Resource Descriptor M
191 {"GpioIo", "(ShareTypeKeyword, PinConfigKeyword, DebounceTi
192 "IoRestrictionKeyword, ResourceSource, "

```

```

193 "ResourceSourceIndex, ResourceTypeKeyword, Descr
194 "RawDataBuffer() {VendorData}) {PinList}",
195 "GPIO I/O Connection Resource Descriptor Macro"
196 {"I2cSerialBus", "(SlaveAddress, SlaveModeKeyword, ConnectionSpee
197 "AddressingModeKeyword, ResourceSource, "
198 "ResourceSourceIndex, ResourceTypeKeyword, Descr
199 "RawDataBuffer() {VendorData})",
200 "I2C Serial Bus Connection Resource Descriptor M
201 {"If", "(Predicate) {TermList}",
202 "Conditional execution",
203 {"Include", "(FilePathName)",
204 "Include another ASL file",
205 {"Increment", "(Addend) => Integer",
206 "Increment a Integer",
207 {"Index", "(Source, Index, Destination) => ObjectReference
208 "Indexed Reference to member object",
209 {"IndexField", "(IndexName, DataName, AccessTypeKeyword, LockRu
210 "UpdateRuleKeyword) {FieldUnitList}",
211 "Declare Index/Data Fields",
212 {"Interrupt", "(ResourceTypeKeyword, InterruptTypeKeyword, Int
213 "ShareTypeKeyword, ResourceSourceIndex, "
214 "ResourceSource, DescriptorName) {InterruptList}
215 "Interrupt Resource Descriptor macro",
216 {"IO", "(IoDecodeKeyword, AddressMin, AddressMax, Addre
217 "RangeLength, DescriptorName) => Buffer",
218 "IO Resource Descriptor macro",
219 {"IRQ", "(InterruptTypeKeyword, InterruptLevelKeyword, S
220 "DescriptorName) {InterruptList} => Buffer",
221 "Interrupt Resource Descriptor macro",
222 {"IRQNoFlags", "(DescriptorName) {InterruptList} => Buffer",
223 "Short Interrupt Resource Descriptor macro",
224 {"LAnd", "(Source1, Source2) => Boolean",
225 "Logical And",
226 {"LEqual", "(Source1, Source2) => Boolean",
227 "Logical Equal",
228 {"LGreater", "(Source1, Source2) => Boolean",
229 "Logical Greater",
230 {"LGreaterEqual", "(Source1, Source2) => Boolean",
231 "Logical Not less",
232 {"LLess", "(Source1, Source2) => Boolean",
233 "Logical Less",
234 {"LLessEqual", "(Source1, Source2) => Boolean",
235 "Logical Not greater",
236 {"LNot", "(Source) => Boolean",
237 "Logical Not",
238 {"LNotEqual", "(Source1, Source2) => Boolean",
239 "Logical Not equal",
240 {"Load", "(Object, DDBHandle)",
241 "Load differentiating definition block",
242 {"LoadTable", "(SignatureString, OemIdString, OemTableIdString
243 "ParameterPathString, ParameterData) => DDBHandl
244 "Load Table from RSDT/XSDT",
245 {"Local", "Local0 - Local7",
246 "Method local data objects",
247 {"LOr", "(Source1, Source2) => Boolean",
248 "Logical Or",
249 {"Match", "(SearchPackage, MatchOpKeyword, MatchObject1, M
250 "MatchObject2, StartIndex) => Ones | Integer",
251 "Search for match in package array",
252 {"Memory24", "(ReadWriteKeyword, AddressMinimum, AddressMaxim
253 "RangeLength, DescriptorName)",
254 "Memory Resource Descriptor macro",
255 {"Memory32", "(ReadWriteKeyword, AddressMinimum, AddressMaxim
256 "RangeLength, DescriptorName)",
257 "Memory Resource Descriptor macro",
258 {"Memory32Fixed", "(ReadWriteKeyword, AddressBase, RangeLength, De

```

```
new/usr/src/common/acpica/tools/acpihelp/ahaslops.c
```

```

325 "RegisterBitOffset, RegisterAddress, "  

326 "AccessSize, DescriptorName)",  

327 "Generic register Resource Descriptor macro",  

328 {"Release",  

329 "Release a synchronization object",  

330 {"Reset",  

331 "Reset a synchronization object",  

332 {"ResourceTemplate",  

333 "Resource to buffer conversion macro",  

334 {"Return",  

335 "Return from method execution",  

336 {"Revision",  

337 "Constant revision object",  

338 {"Scope",  

339 "Open named scope ",  

340 {"ShiftLeft",  

341 "Integer shift value left",  

342 {"ShiftRight",  

343 "Integer shift value right",  

344 {"Signal",  

345 "Signal a synchronization object",  

346 {"SizeOf",  

347 "Get the size of a buffer", string}}, or package"  

348 {"Sleep",  

349 "Sleep n milliseconds (yields the processor)",  

350 {"SpiSerialBus",  

351 "DeviceSelection, PolarityKeyword, WireModeKeyw  

352 "DataBitLength, SlaveModeKeyword, "  

353 "ConnectionSpeed, ClockPolarityKeyword, ClockPha  

354 "ResourceSource, ResourceSourceIndex, "  

355 "ResourceTypeKeyword, DescriptorName, RawDataBuf  

356 "SPI Serial Bus Connection Resource Descriptor M  

357 {"Stall",  

358 "Delay n microseconds (does not yield the proces  

359 {"StartDependentFn",  

360 {"StartDependentFnNoPri",  

361 "Start Dependent Function Resource Descriptor ma  

362 {"Store",  

363 "Store object",  

364 {"Subtract",  

365 "Integer Subtract",  

366 {"Switch",  

367 "Select code to execute based on expression valu  

368 {"ThermalZone",  

369 "Declare a thermal zone package",  

370 {"Timer",  

371 "Get 64-bit timer value",  

372 {"ToBCD",  

373 "Convert Integer to BCD",  

374 {"ToBuffer",  

375 "Convert data type to buffer",  

376 {"ToDecimalString",  

377 "Convert data type to decimal string",  

378 {"ToHexString",  

379 "Convert data type to hexadecimal string",  

380 {"ToInteger",  

381 "Convert data type to integer",  

382 {"ToString",  

383 "Copy ASCII string from buffer",  

384 {"ToUUID",  

385 "Convert Ascii string to UUID",  

386 {"UartSerialBus",  

387 "LinesInUse, EndianKeyword, ParityKeyword, "  

388 "FlowControlKeyword, ReceiveBufferSize, Transmit  

389 "ResourceSourceIndex, ResourceTypeKeyword, Descr  

390 "RawDataBuffer() {VendorData}}",

```

```
391     "UART Serial Bus Connection Resource Descriptor
392     {"Unicode",
393     "(String) => Buffer",
394     "String to Unicode conversion macro",
395     {"Unload",
396     "(Handle)",
397     "Unload definition block",
398     {"VendorLong",
399     "(DescriptorName) {VendorByteList}",
400     "Vendor Resource Descriptor",
401     {"VendorShort",
402     "(DescriptorName) {VendorByteList}",
403     "Vendor Resource Descriptor",
404     {"Wait",
405     "(SyncObject, TimeoutValue) => Boolean",
406     "Wait on an Event",
407     {"While",
408     "(Predicate) {TermList}",
409     "Conditional loop",
410     {"WordBusNumber",
411     "(ResourceTypeKeyword, MinKeyword, MaxKeyword, D
412     AddressGranularity, AddressMinimum, "
413     AddressMaximum, AddressTranslation, RangeLength
414     ResourceSource, DescriptorName)",
415     "Word Bus number Resource Descriptor macro",
416     {"WordIO",
417     "(ResourceTypeKeyword, MinKeyword, MaxKeyword, D
418     RangeTypeKeyword, AddressGranularity, "
419     AddressMinimum, AddressMaximum, AddressTranslat
420     ResourceSourceIndex, ResourceSource, Descriptor
421     TranslationKeyword)",
422     "Word IO Resource Descriptor macro",
423     {"WordSpace",
424     "(ResourceType, ResourceTypeKeyword, DecodeKeywo
425     MaxKeyword, TypeSpecificFlags, "
426     AddressGranularity, AddressMinimum, AddressMaxi
427     RangeLength, ResourceSourceIndex, ResourceSourc
428     Word Space Resource Descriptor macro",
429     {"XOr",
430     "(Source1, Source2, Result) => Integer",
431     "Integer Bitwise Xor",
432     {"Zero",
433     "=> Integer",
434     "Constant Zero object (0)",
435     {NULL, NULL, NULL}
436     };
```

new/usr/src/common/acpica/tools/acpihelp/ahdecode.c

1

```
*****
23886 Fri Jan 17 07:31:16 2014
new/usr/src/common/acpica/tools/acpihelp/ahdecode.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: ahdecode - Operator/Opcode decoding for acpihelp utility
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define ACPI_CREATE_PREDEFINED_TABLE
45 #define ACPI_CREATE_RESOURCE_TABLE
46
47 #include "acpihelp.h"
48 #include "acpredef.h"
49
50
51 /* Device IDs defined in the ACPI specification */
52
53 static const AH_DEVICE_ID AhDeviceIds[] =
54 {
55     {"PNP0A05", "Generic Container Device"},
56     {"PNP0A06", "Generic Container Device"},
57     {"PNP0C08", "ACPI core hardware"},
58     {"PNP0C09", "Embedded Controller Device"},
59     {"PNP0C0A", "Control Method Battery"},
60     {"PNP0C0B", "Fan"},

```

new/usr/src/common/acpica/tools/acpihelp/ahdecode.c

2

```
61     {"PNP0C0C", "Power Button Device"},
62     {"PNP0C0D", "Lid Device"},
63     {"PNP0C0E", "Sleep Button Device"},
64     {"PNP0C0F", "PCI Interrupt Link Device"},
65     {"PNP0C80", "Memory Device"},
66
67     {"ACPI0001", "SMBus 1.0 Host Controller"},
68     {"ACPI0002", "Smart Battery Subsystem"},
69     {"ACPI0003", "Power Source Device"},
70     {"ACPI0004", "Module Device"},
71     {"ACPI0005", "SMBus 2.0 Host Controller"},
72     {"ACPI0006", "GPE Block Device"},
73     {"ACPI0007", "Processor Device"},
74     {"ACPI0008", "Ambient Light Sensor Device"},
75     {"ACPI0009", "I/O xAPIC Device"},
76     {"ACPI000A", "I/O APIC Device"},
77     {"ACPI000B", "I/O xAPIC Device"},
78     {"ACPI000C", "Processor Aggregator Device"},
79     {"ACPI000D", "Power Meter Device"},
80     {"ACPI000E", "Time/Alarm Device"},
81     {"ACPI000F", "User Presence Detection Device"},
82
83     {NULL, NULL}
84 };
85
86 #define AH_DISPLAY_EXCEPTION(Status, Name) \
87     printf ("%4X: %s\n", Status, Name)
88
89 #define AH_DISPLAY_EXCEPTION_TEXT(Status, Exception) \
90     printf ("%4X: %-28s (%s)\n", Status, Exception->Name, Exception->Descriptio
91
92 #define BUFFER_LENGTH 128
93 #define LINE_BUFFER_LENGTH 512
94
95 static char Gbl_Buffer[BUFFER_LENGTH];
96 static char Gbl_LineBuffer[LINE_BUFFER_LENGTH];
97
98 /* Local prototypes */
99
100 static BOOLEAN
101 AhDisplayPredefinedName (
102     char *Name,
103     UINT32 Length);
104
105 static void
106 AhDisplayPredefinedInfo (
107     char *Name);
108
109 static void
110 AhDisplayResourceName (
111     const ACPI_PREDEFINED_INFO *ThisName);
112
113 static void
114 AhDisplayAmlOpcode (
115     const AH_AML_OPCODE *Op);
116
117 static void
118 AhDisplayAslOperator (
119     const AH_AS_L_OPERATOR *Op);
120
121 static void
122 AhDisplayOperatorKeywords (
123     const AH_AS_L_OPERATOR *Op);
124
125 static void
126 AhDisplayAslKeyword (

```

```

127  const AH_AS_L_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      /* Construct 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\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\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.
260 *
261 * RETURN:      None
262 *
263 * DESCRIPTION: Find the name in the main ACPIA predefined info table and
264 *              display the # of arguments and the return value type.
265 *
266 *              Note: Resource Descriptor field names do not appear in this
267 *              table -- thus, nothing will be displayed for them.
268 *
269 *****/
271 static void
272 AhDisplayPredefinedInfo (
273     char                *Name)
274 {
275     const ACPI_PREDEFINED_INFO *ThisName;

278     /* NOTE: we check both tables always because there are some dupes */

280     /* Check against the predefine methods first */

282     ThisName = AcpiUtMatchPredefinedMethod (Name);
283     if (ThisName)
284     {
285         AcpiUtDisplayPredefinedMethod (Gbl_Buffer, ThisName, TRUE);
286     }

288     /* Check against the predefined resource descriptor names */

290     ThisName = AcpiUtMatchResourceName (Name);
291     if (ThisName)
292     {
293         AhDisplayResourceName (ThisName);
294     }
295 }

298 *****/
299 *
300 * FUNCTION:      AhDisplayResourceName
301 *
302 * PARAMETERS:    ThisName           - Entry in the predefined method/name table
303 *
304 * RETURN:        None
305 *
306 * DESCRIPTION:   Display information about a resource descriptor name.
307 *
308 *****/
310 static void
311 AhDisplayResourceName (
312     const ACPI_PREDEFINED_INFO *ThisName)
313 {
314     UINT32                NumTypes;

317     NumTypes = AcpiUtGetResourceBitWidth (Gbl_Buffer,
318     ThisName->Info.ArgumentList);

320     printf ("          %4.4s resource descriptor field is %s bits wide%s\n",
321     ThisName->Info.Name,
322     Gbl_Buffer,
323     (NumTypes > 1) ? " (depending on descriptor type)" : "");
324 }

```

```

327 *****/
328 *
329 * FUNCTION:      AhFindAmlOpcode (entry point for AML opcode name search)
330 *
331 * PARAMETERS:    Name           - Name or prefix for an AML opcode.
332 *              NULL means "find all"
333 *
334 * RETURN:        None
335 *
336 * DESCRIPTION:   Find all AML opcodes that match the input Name or name
337 *              prefix.
338 *
339 *****/
341 void
342 AhFindAmlOpcode (
343     char                *Name)
344 {
345     const AH_AML_OPCODE *Op;
346     BOOLEAN              Found = FALSE;

349     AhStrupr (Name);

351     /* Find/display all opcode names that match the input name prefix */

353     for (Op = AmlOpcodeInfo; Op->OpcodeString; Op++)
354     {
355         if (!Op->OpcodeName) /* Unused opcodes */
356         {
357             continue;
358         }

360         if (!Name)
361         {
362             AhDisplayAmlOpcode (Op);
363             Found = TRUE;
364             continue;
365         }

367         /* Upper case the opcode name before substring compare */

369         strcpy (Gbl_Buffer, Op->OpcodeName);
370         AhStrupr (Gbl_Buffer);

372         if (strstr (Gbl_Buffer, Name) == Gbl_Buffer)
373         {
374             AhDisplayAmlOpcode (Op);
375             Found = TRUE;
376         }
377     }

379     if (!Found)
380     {
381         printf ("%s, no matching AML operators\n", Name);
382     }
383 }

386 *****/
387 *
388 * FUNCTION:      AhDecodeAmlOpcode (entry point for AML opcode search)
389 *
390 * PARAMETERS:    OpcodeString     - String version of AML opcode

```

```

391 *
392 * RETURN:      None
393 *
394 * DESCRIPTION: Display information about the input AML opcode
395 *
396 *****/
398 void
399 AhDecodeAmlOpcode (
400     char                *OpcodeString)
401 {
402     const AH_AML_OPCODE *Op;
403     UINT32               Opcode;
404     UINT8                Prefix;
405
406     if (!OpcodeString)
407     {
408         AhFindAmlOpcode (NULL);
409         return;
410     }
411
412     Opcode = ACPI_STRTOUL (OpcodeString, NULL, 16);
413     if (Opcode > ACPI_UINT16_MAX)
414     {
415         printf ("Invalid opcode (more than 16 bits)\n");
416         return;
417     }
418
419     /* Only valid opcode extension is 0x5B */
420
421     Prefix = (Opcode & 0x0000FF00) >> 8;
422     if (Prefix && (Prefix != 0x5B))
423     {
424         printf ("Invalid opcode (invalid extension prefix 0x%X)\n",
425             Prefix);
426         return;
427     }
428
429     /* Find/Display the opcode. May fall within an opcode range */
430
431     for (Op = AmlOpcodeInfo; Op->OpcodeString; Op++)
432     {
433         if ((Opcode >= Op->OpcodeRangeStart) &&
434             (Opcode <= Op->OpcodeRangeEnd))
435         {
436             AhDisplayAmlOpcode (Op);
437         }
438     }
439 }
440
441 *****/
442
443 *
444 * FUNCTION:      AhDisplayAmlOpcode
445 *
446 * PARAMETERS:    Op
447 *                - An opcode info struct
448 *
449 * RETURN:        None
450 *
451 * DESCRIPTION:   Display the contents of an AML opcode information struct
452 *
453 *****/
454
455 static void
456 AhDisplayAmlOpcode (

```

```

457     const AH_AML_OPCODE *Op)
458 {
459     if (!Op->OpcodeName)
460     {
461         printf ("%18s: Opcode=%-9s\n", "Reserved opcode", Op->OpcodeString);
462         return;
463     }
464
465     /* Opcode name and value(s) */
466
467     printf ("%18s: Opcode=%-9s Type (%s)",
468         Op->OpcodeName, Op->OpcodeString, Op->Type);
469
470     /* Optional fixed/static arguments */
471
472     if (Op->FixedArguments)
473     {
474         printf (" FixedArgs (");
475         AhPrintOneField (37, 36 + 7 + strlen (Op->Type) + 12,
476             AH_MAX_AML_LINE_LENGTH, Op->FixedArguments);
477         printf (");");
478     }
479
480     /* Optional variable-length argument list */
481
482     if (Op->VariableArguments)
483     {
484         if (Op->FixedArguments)
485         {
486             printf ("\n%s", 36, " ");
487         }
488         printf (" VariableArgs (");
489         AhPrintOneField (37, 15, AH_MAX_AML_LINE_LENGTH, Op->VariableArguments);
490         printf (");");
491     }
492     printf ("\n");
493
494     /* Grammar specification */
495
496     if (Op->Grammar)
497     {
498         AhPrintOneField (37, 0, AH_MAX_AML_LINE_LENGTH, Op->Grammar);
499         printf ("\n");
500     }
501 }
502
503 *****/
504
505 *
506 * FUNCTION:      AhFindAslKeywords (entry point for ASL keyword search)
507 *
508 * PARAMETERS:    Name
509 *                - Name or prefix for an ASL keyword.
510 *                NULL means "find all"
511 *
512 * RETURN:        None
513 *
514 * DESCRIPTION:   Find all ASL keywords that match the input Name or name
515 *                prefix.
516 *
517 *****/
518
519 void
520 AhFindAslKeywords (
521     char                *Name)
522 {

```

```

523     const AH_AS_L_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_AS_L_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_AS_L_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_AS_L_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 *****/
657 static void
658 AhDisplayAslOperator (
659     const AH_AS_L_OPERATOR *Op)
660 {
662     /* ASL operator name and description */
664     printf ("%16s: %s\n", Op->Name, Op->Description);
665     if (!Op->Syntax)
666     {
667         return;
668     }
670     /* Syntax for the operator */
672     AhPrintOneField (18, 0, AH_MAX_AS_L_LINE_LENGTH, Op->Syntax);
673     printf ("\n");
675     AhDisplayOperatorKeywords (Op);
676     printf ("\n");
677 }
680 /*****
681  *
682  * FUNCTION:      AhDisplayOperatorKeywords
683  *
684  * PARAMETERS:    Op                - Pointer to ASL keyword with syntax info
685  *
686  * RETURN:        None
687  *
688  * DESCRIPTION:   Display any/all keywords that are associated with the ASL
689  *                operator.
690  *
691  *****/
693 static void
694 AhDisplayOperatorKeywords (
695     const AH_AS_L_OPERATOR *Op)
696 {
697     char *Token;
698     char *Separators = "(){} , ";
699     BOOLEAN FirstKeyword = TRUE;
702     if (!Op || !Op->Syntax)
703     {
704         return;
705     }
707     /*
708      * Find all parameters that have the word "keyword" within, and then
709      * display the info about that keyword
710      */
711     strcpy (Gbl_LineBuffer, Op->Syntax);
712     Token = strtok (Gbl_LineBuffer, Separators);
713     while (Token)
714     {
715         if (strstr (Token, "Keyword"))
716         {
717             if (FirstKeyword)
718             {
719                 printf ("\n");
720                 FirstKeyword = FALSE;

```

```

721     }
723     /* Found a keyword, display keyword information */
725     AhFindAslKeywords (Token);
726 }
728     Token = strtok (NULL, Separators);
729 }
730 }
733 /*****
734  *
735  * FUNCTION:      AhPrintOneField
736  *
737  * PARAMETERS:    Indent            - Indent length for new line(s)
738  *                CurrentPosition    - Position on current line
739  *                MaxPosition        - Max allowed line length
740  *                Field              - Data to output
741  *
742  * RETURN:        Line position after field is written
743  *
744  * DESCRIPTION:   Split long lines appropriately for ease of reading.
745  *
746  *****/
748 static void
749 AhPrintOneField (
750     UINT32 Indent,
751     UINT32 CurrentPosition,
752     UINT32 MaxPosition,
753     const char *Field)
754 {
755     UINT32 Position;
756     UINT32 TokenLength;
757     const char *This;
758     const char *Next;
759     const char *Last;
762     This = Field;
763     Position = CurrentPosition;
765     if (Position == 0)
766     {
767         printf ("%s", (int) Indent, " ");
768         Position = Indent;
769     }
771     Last = This + strlen (This);
772     while ((Next = strpbrk (This, " "))
773     {
774         TokenLength = Next - This;
775         Position += TokenLength;
777         /* Split long lines */
779         if (Position > MaxPosition)
780         {
781             printf ("\n%s", (int) Indent, " ");
782             Position = TokenLength;
783         }
785         printf ("%.*s ", (int) TokenLength, This);
786         This = Next + 1;

```

```

787     }

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 ACPI exception codes:\n\n");
861         AH_DISPLAY_EXCEPTION (0, "AE_OK", "No error occurred");

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++;
876             } while (ExceptionInfo);
877         }
878         return;
879     }
880 }

882 /* Decode a single user-supplied exception code */

884 Status = ACPI_STRTOL (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 }

```

new/usr/src/common/acpica/tools/acpihelp/ahmain.c

1

```
*****
7002 Fri Jan 17 07:31:17 2014
new/usr/src/common/acpica/tools/acpihelp/ahmain.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: ahmain - Main module for the acpi help utility
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "acpihelp.h"
45
46 /* Local prototypes */
47
48 static void
49 AhDisplayUsage (
50     void);
51
52 #define AH_UTILITY_NAME      "ACPI Help Utility"
53 #define AH_SUPPORTED_OPTIONS "ehikmopsv"
54
55 /*****
56 *
57 * FUNCTION:    AhDisplayUsage
58 *
59 *****/
```

new/usr/src/common/acpica/tools/acpihelp/ahmain.c

2

```
61 * DESCRIPTION: Usage message
62 *
63 *****/
64
65 static void
66 AhDisplayUsage (
67     void)
68 {
69
70     ACPI_USAGE_HEADER ("acpihelp <options> [NamePrefix | HexValue]");
71     ACPI_OPTION ("-h", "Display help");
72     ACPI_OPTION ("-v", "Display version information");
73
74     printf ("\nACPI Names and Symbols:\n");
75     ACPI_OPTION ("-k [NamePrefix]", "Find/Display ASL non-operator keywo");
76     ACPI_OPTION ("-m [NamePrefix]", "Find/Display AML opcode name(s)");
77     ACPI_OPTION ("-p [NamePrefix]", "Find/Display ASL predefined method");
78     ACPI_OPTION ("-s [NamePrefix]", "Find/Display ASL operator name(s)");
79
80     printf ("\nACPI Values:\n");
81     ACPI_OPTION ("-e [HexValue]", "Decode ACPICA exception code");
82     ACPI_OPTION ("-i", "Display known ACPI Device IDs (_HID");
83     ACPI_OPTION ("-o [HexValue]", "Decode hex AML opcode");
84
85     printf ("\nNamePrefix/HexValue not specified means \"Display All\"\n");
86     printf ("\nDefault search with NamePrefix and no options:\n");
87     printf ("    Find ASL operator names - if NamePrefix does not start with und");
88     printf ("    Find ASL predefined method names - if NamePrefix starts with un");
89 }
90
91 /*****
92 *
93 * FUNCTION:    main
94 *
95 * DESCRIPTION: C main function for AcpiHelp utility.
96 *
97 *
98 *****/
99
100 int ACPI_SYSTEM_XFACE
101 main (
102     int      argc,
103     char     *argv[])
104 {
105     char     *Name;
106     UINT32   DecodeType;
107     int      j;
108
109     ACPI_DEBUG_INITIALIZE (); /* For debug version only */
110     printf (ACPI_COMMON_SIGNON (AH_UTILITY_NAME));
111     DecodeType = AH_DECODE_DEFAULT;
112
113     if (argc < 2)
114     {
115         AhDisplayUsage ();
116         return (0);
117     }
118
119     /* Command line options */
120
121     while ((j = AcpiGetopt (argc, argv, AH_SUPPORTED_OPTIONS)) != EOF) switch (j)
122     {
123     case 'e':
124         DecodeType = AH_DECODE_EXCEPTION;
125     }
```

```

127     break;
129     case 'i':
131         DecodeType = AH_DISPLAY_DEVICE_IDS;
132         break;
134     case 'k':
136         DecodeType = AH_DECODE_AS_L_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_AS_L;
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_AS_L:

```

```

193         AhFindAslOperators (Name);
194         break;
196     case AH_DECODE_AS_L_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;
254     if (!SrcString)
255     {
256         return;
257     }
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 }
```

new/usr/src/common/acpica/tools/acpinames/acpinames.h

1

\*\*\*\*\*

2317 Fri Jan 17 07:31:17 2014

new/usr/src/common/acpica/tools/acpinames/acpinames.h

update to acpica-unix2-20140114

acpica-unix2-20130823

PANKOVs restructure

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: acpinames.h - Common include for AcpiNames utility
4 *
5 *****/
```

```
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 _ACPINAMES_H
45 #define _ACPINAMES_H
46
47 #include "acpi.h"
48 #include "accommon.h"
49 #include "acapps.h"
50 #include "../acpiexec/aeccommon.h"
51
52 #include <stdio.h>
53
54 #define ACPI_MAX_INIT_TABLES (32)
55
56 #endif
```

new/usr/src/common/acpica/tools/acpinames/anmain.c

1

```
*****
8211 Fri Jan 17 07:31:17 2014
new/usr/src/common/acpica/tools/acpinames/anmain.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: anmain - Main routine for the AcpiNames utility
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "acpinames.h"
45
46 #define _COMPONENT          ACPI_TOOLS
47         ACPI_MODULE_NAME    ("anmain")
48
49
50 extern ACPI_TABLE_DESC  Tables[];
51
52 static AE_TABLE_DESC     *AeTableListHead = NULL;
53
54
55 #define AN_UTILITY_NAME      "ACPI Namespace Dump Utility"
56 #define AN_SUPPORTED_OPTIONS "?hv"
```

new/usr/src/common/acpica/tools/acpinames/anmain.c

2

```
59 /*****
60 *
61 * FUNCTION:      usage
62 *
63 * PARAMETERS:    None
64 *
65 * RETURN:        None
66 *
67 * DESCRIPTION:   Print a usage message
68 *
69 *****/
70
71 static void
72 usage (
73     void)
74 {
75
76     ACPI_USAGE_HEADER ("AcpiNames [options] Amlfile");
77     ACPI_OPTION ("-?", "Display this message");
78     ACPI_OPTION ("-v", "Display version information");
79 }
80
81 /*****
82 *
83 * FUNCTION:      NsDumpEntireNamespace
84 *
85 * PARAMETERS:    AmlFilename - Filename for DSDT or SSDT AML table
86 *
87 * RETURN:        Status (pass/fail)
88 *
89 * DESCRIPTION:   Build an ACPI namespace for the input AML table, and dump the
90 *               formatted namespace contents.
91 *
92 *
93 *****/
94
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;
104
105
106     /* Open the binary AML file and read the entire table */
107
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     }
115
116     /* Table must be a DSDT. SSDTs are not currently supported */
117
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     }
124 }
```

```

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);

260     /* Get the command line options */

262     while ((j = AcpiGetopt (argc, argv, AN_SUPPORTED_OPTIONS)) != EOF) switch(j)
263     {
264     case 'v': /* -v: (Version): signon already emitted, just exit */
266         return (0);

268     case '?':
269     case 'h':
270     default:

272         usage();
273         return (0);
274     }

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 }
```

new/usr/src/common/acpica/tools/acpinames/anstubs.c

1

```
*****
8820 Fri Jan 17 07:31:17 2014
new/usr/src/common/acpica/tools/acpinames/anstubs.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: anstubs - Stub routines for the AcpiNames utility
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 #include <acutils.h>
47 #include <acevents.h>
48 #include <acdispat.h>

50 #define _COMPONENT          ACPI_TOOLS
51 #define ACPI_MODULE_NAME    ("anstubs")

54 /*****
55 *
56 * DESCRIPTION: Stubs used to facilitate linkage of the NsDump utility.
57 *
58 *****/
```

new/usr/src/common/acpica/tools/acpinames/anstubs.c

2

```
61 /* Utilities */

63 void
64 AcpiUtSubsystemShutdown (
65     void)
66 {
67 }

69 ACPI_STATUS
70 AcpiUtExecute_STA (
71     ACPI_NAMESPACE_NODE *DeviceNode,
72     UINT32 *StatusFlags)
73 {
74     return (AE_NOT_IMPLEMENTED);
75 }

77 ACPI_STATUS
78 AcpiUtExecute_HID (
79     ACPI_NAMESPACE_NODE *DeviceNode,
80     ACPI_PNP_DEVICE_ID **ReturnId)
81 {
82     return (AE_NOT_IMPLEMENTED);
83 }

85 ACPI_STATUS
86 AcpiUtExecute_CID (
87     ACPI_NAMESPACE_NODE *DeviceNode,
88     ACPI_PNP_DEVICE_ID_LIST **ReturnCidList)
89 {
90     return (AE_NOT_IMPLEMENTED);
91 }

93 ACPI_STATUS
94 AcpiUtExecute_UID (
95     ACPI_NAMESPACE_NODE *DeviceNode,
96     ACPI_PNP_DEVICE_ID **ReturnId)
97 {
98     return (AE_NOT_IMPLEMENTED);
99 }

101 ACPI_STATUS
102 AcpiUtExecute_SUB (
103     ACPI_NAMESPACE_NODE *DeviceNode,
104     ACPI_PNP_DEVICE_ID **ReturnId)
105 {
106     return (AE_NOT_IMPLEMENTED);
107 }

109 ACPI_STATUS
110 AcpiUtExecutePowerMethods (
111     ACPI_NAMESPACE_NODE *DeviceNode,
112     const char **MethodNames,
113     UINT8 MethodCount,
114     UINT8 *OutValues)
115 {
116     return (AE_NOT_IMPLEMENTED);
117 }

119 ACPI_STATUS
120 AcpiUtEvaluateNumericObject (
121     char *ObjectName,
122     ACPI_NAMESPACE_NODE *DeviceNode,
123     UINT64 *Value)
124 {
125     return (AE_NOT_IMPLEMENTED);
126 }
```

```

126 }

128 ACPI_STATUS
129 AcpiUtCopyIobjectToEobject (
130     ACPI_OPERAND_OBJECT *Obj,
131     ACPI_BUFFER          *RetBuffer)
132 {
133     return (AE_NOT_IMPLEMENTED);
134 }

136 ACPI_STATUS
137 AcpiUtCopyEobjectToIobject (
138     ACPI_OBJECT          *Obj,
139     ACPI_OPERAND_OBJECT **InternalObj)
140 {
141     return (AE_NOT_IMPLEMENTED);
142 }

144 ACPI_STATUS
145 AcpiUtCopyIobjectToIobject (
146     ACPI_OPERAND_OBJECT *SourceDesc,
147     ACPI_OPERAND_OBJECT **DestDesc,
148     ACPI_WALK_STATE      *WalkState)
149 {
150     return (AE_NOT_IMPLEMENTED);
151 }

154 /* Hardware manager */

156 UINT32
157 AcpiHwGetMode (
158     void)
159 {
160     return (0);
161 }

164 /* Event manager */

166 ACPI_STATUS
167 AcpiEvInstallXruptHandlers (
168     void)
169 {
170     return (AE_OK);
171 }

173 ACPI_STATUS
174 AcpiEvInitializeEvents (
175     void)
176 {
177     return (AE_OK);
178 }

180 ACPI_STATUS
181 AcpiEvInstallRegionHandlers (
182     void)
183 {
184     return (AE_OK);
185 }

187 ACPI_STATUS
188 AcpiEvInitializeOpRegions (
189     void)
190 {
191     return (AE_OK);

```

```

192 }

194 ACPI_STATUS
195 AcpiEvInitializeRegion (
196     ACPI_OPERAND_OBJECT *RegionObj,
197     BOOLEAN              AcpiNsLocked)
198 {
199     return (AE_OK);
200 }

202 #if (!ACPI_REDUCED_HARDWARE)
203 ACPI_STATUS
204 AcpiEvDeleteGpeBlock (
205     ACPI_GPE_BLOCK_INFO *GpeBlock)
206 {
207     return (AE_OK);
208 }

210 ACPI_STATUS
211 AcpiEnable (
212     void)
213 {
214     return (AE_OK);
215 }
216 #endif /* !ACPI_REDUCED_HARDWARE */

219 /* AML Interpreter */

221 void
222 AcpiExUnlinkMutex (
223     ACPI_OPERAND_OBJECT *ObjDesc)
224 {
225 }

227 void
228 AcpiExReleaseAllMutexes (
229     ACPI_THREAD_STATE *Thread)
230 {
231 }

233 ACPI_STATUS
234 AcpiExReadDataFromField (
235     ACPI_WALK_STATE *WalkState,
236     ACPI_OPERAND_OBJECT *ObjDesc,
237     ACPI_OPERAND_OBJECT **RetBufferDesc)
238 {
239     return (AE_NOT_IMPLEMENTED);
240 }

242 ACPI_STATUS
243 AcpiExWriteDataToField (
244     ACPI_OPERAND_OBJECT *SourceDesc,
245     ACPI_OPERAND_OBJECT *ObjDesc,
246     ACPI_OPERAND_OBJECT **ResultDesc)
247 {
248     return (AE_NOT_IMPLEMENTED);
249 }

251 ACPI_STATUS
252 AcpiExPrepFieldValue (
253     ACPI_CREATE_FIELD_INFO *Info)
254 {
255     return (AE_OK);
256 }

```

```

258 ACPI_STATUS
259 AcpiExAcquireMutexObject (
260     UINT16          Timeout,
261     ACPI_OPERAND_OBJECT *ObjDesc,
262     ACPI_THREAD_ID   ThreadId)
263 {
264     return (AE_OK);
265 }
267 ACPI_STATUS
268 AcpiExReleaseMutexObject (
269     ACPI_OPERAND_OBJECT *ObjDesc)
270 {
271     return (AE_OK);
272 }
274 ACPI_STATUS
275 AcpiExStoreObjectToNode (
276     ACPI_OPERAND_OBJECT *SourceDesc,
277     ACPI_NAMESPACE_NODE *Node,
278     ACPI_WALK_STATE      *WalkState,
279     UINT8                ImplicitConversion)
280 {
281     return (AE_NOT_IMPLEMENTED);
282 }
285 /* Namespace manager */
287 ACPI_STATUS
288 AcpiNsEvaluate (
289     ACPI_EVALUATE_INFO *Info)
290 {
291     return (AE_NOT_IMPLEMENTED);
292 }
294 void
295 AcpiNsExecModuleCodeList (
296     void)
297 {
298 }
301 /* Dispatcher */
303 ACPI_STATUS
304 AcpiDsInitializeObjects (
305     UINT32          TableIndex,
306     ACPI_NAMESPACE_NODE *StartNode)
307 {
308     return (AE_OK);
309 }
311 ACPI_STATUS
312 AcpiDsCallControlMethod (
313     ACPI_THREAD_STATE *Thread,
314     ACPI_WALK_STATE   *WalkState,
315     ACPI_PARSE_OBJECT *Op)
316 {
317     return (AE_NOT_IMPLEMENTED);
318 }
320 ACPI_STATUS
321 AcpiDsRestartControlMethod (
322     ACPI_WALK_STATE *WalkState,
323     ACPI_OPERAND_OBJECT *ReturnDesc)

```

```

324 {
325     return (AE_NOT_IMPLEMENTED);
326 }
328 void
329 AcpiDsTerminateControlMethod (
330     ACPI_OPERAND_OBJECT *MethodDesc,
331     ACPI_WALK_STATE      *WalkState)
332 {
333 }
335 ACPI_STATUS
336 AcpiDsMethodError (
337     ACPI_STATUS          Status,
338     ACPI_WALK_STATE      *WalkState)
339 {
340     return (AE_NOT_IMPLEMENTED);
341 }
343 ACPI_STATUS
344 AcpiDsBeginMethodExecution (
345     ACPI_NAMESPACE_NODE *MethodNode,
346     ACPI_OPERAND_OBJECT *ObjDesc,
347     ACPI_WALK_STATE      *WalkState)
348 {
349     return (AE_NOT_IMPLEMENTED);
350 }
352 ACPI_STATUS
353 AcpiDsGetPredicateValue (
354     ACPI_WALK_STATE      *WalkState,
355     ACPI_OPERAND_OBJECT *ResultObj)
356 {
357     return (AE_NOT_IMPLEMENTED);
358 }
360 ACPI_STATUS
361 AcpiDsGetBufferFieldArguments (
362     ACPI_OPERAND_OBJECT *ObjDesc)
363 {
364     return (AE_OK);
365 }
367 ACPI_STATUS
368 AcpiDsGetBankFieldArguments (
369     ACPI_OPERAND_OBJECT *ObjDesc)
370 {
371     return (AE_OK);
372 }
374 ACPI_STATUS
375 AcpiDsGetRegionArguments (
376     ACPI_OPERAND_OBJECT *RgnDesc)
377 {
378     return (AE_OK);
379 }
381 ACPI_STATUS
382 AcpiDsGetBufferArguments (
383     ACPI_OPERAND_OBJECT *ObjDesc)
384 {
385     return (AE_OK);
386 }
388 ACPI_STATUS
389 AcpiDsGetPackageArguments (

```

```
390     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 }
```

new/usr/src/common/acpica/tools/acpinames/antables.c

1

```
*****
9903 Fri Jan 17 07:31:17 2014
new/usr/src/common/acpica/tools/acpinames/antables.c
update to acpica-unix2-20140114
acpica-unix2-20130823
PANKOVs restructure
*****
1 /*****
2 *
3 * Module Name: antables - ACPI table setup/install for AcpiNames utility
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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        ("antables")

49 /* Local prototypes */

51 ACPI_PHYSICAL_ADDRESS
52 AeLocalGetRootPointer (
53     void);

55 /* Non-AML tables that are constructed locally and installed */

57 static ACPI_TABLE_RSDP      LocalRSDP;
58 static ACPI_TABLE_FACS      LocalFACS;
```

new/usr/src/common/acpica/tools/acpinames/antables.c

2

```
60 /*
61 * We need a local FADT so that the hardware subcomponent will function,
62 * even though the underlying OSD HW access functions don't do anything.
63 */
64 static ACPI_TABLE_FADT      LocalFADT;

66 /*
67 * Use XSDT so that both 32- and 64-bit versions of this utility will
68 * function automatically.
69 */
70 static ACPI_TABLE_XSDT      *LocalXSDT;

72 #define BASE_XSDT_TABLES      1
73 #define BASE_XSDT_SIZE        (sizeof (ACPI_TABLE_XSDT) + \
74                                 ((BASE_XSDT_TABLES - 1) * sizeof (UINT64)))

76 ACPI_TABLE_DESC              Tables[ACPI_MAX_INIT_TABLES];

79 /*****
80 *
81 * FUNCTION:      AeBuildLocalTables
82 *
83 * PARAMETERS:    TableCount      - Number of tables on the command line
84 *                TableList       - List of actual tables from files
85 *
86 * RETURN:        Status
87 *
88 * DESCRIPTION:   Build a complete ACPI table chain, with a local RSDP, XSDT,
89 *                FADT, FACS, and the input DSDT/SSDT.
90 *
91 *****/

93 ACPI_STATUS
94 AeBuildLocalTables (
95     UINT32              TableCount,
96     AE_TABLE_DESC        *TableList)
97 {
98     ACPI_PHYSICAL_ADDRESS DsdtAddress = 0;
99     UINT32                XsdtSize;
100     AE_TABLE_DESC          *NextTable;
101     UINT32                 NextIndex;
102     ACPI_TABLE_FADT        *ExternalFadt = NULL;

105     /*
106     * Update the table count. For DSDT, it is not put into the XSDT. For
107     * FADT, this is already accounted for since we usually install a
108     * local FADT.
109     */
110     NextTable = TableList;
111     while (NextTable)
112     {
113         if (ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_DSDT) ||
114             ACPI_COMPARE_NAME (NextTable->Table->Signature, ACPI_SIG_FADT))
115         {
116             TableCount--;
117         }
118         NextTable = NextTable->Next;
119     }

121     XsdtSize = BASE_XSDT_SIZE + (TableCount * sizeof (UINT64));

123     /* Build an XSDT */

125     LocalXSDT = AcpiOsAllocate (XsdtSize);
```

```

126     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.PmlEventLength = 4;
248         LocalFADT.PmlaEventBlock = 0x00001aaa;
249         LocalFADT.PmlbEventBlock = 0x00001bbb;

251         LocalFADT.PmlControlLength = 2;
252         LocalFADT.PmlaControlBlock = 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.XPmlbEventBlock.SpaceId = ACPI_ADR_SPACE_SYSTEM_IO;
263     LocalFADT.XPmlbEventBlock.Address = LocalFADT.PmlbEventBlock;
264     LocalFADT.XPmlbEventBlock.BitWidth = (UINT8) ACPI_MUL_8 (LocalFADT.PmlEv
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 }
```



new/usr/src/common/acpica/tools/acpirc/acpirc.h

1

```
*****
12847 Fri Jan 17 07:31:18 2014
new/usr/src/common/acpica/tools/acpirc/acpirc.h
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: acpirc.h - Include file for AcpiSrc utility
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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"

47 #include <stdio.h>
48 #include <sys/stat.h>
49 #include <errno.h>

51 /* mkdir support */

53 #ifdef WIN32
54 #include <direct.h>
55 #else
56 #define mkdir(x) mkdir(x, 0770)
57 #endif
```

new/usr/src/common/acpica/tools/acpirc/acpirc.h

2

```
60 /* Constants */

62 #define LINES_IN_LEGAL_HEADER          105 /* See above */
63 #define LEGAL_HEADER_SIGNATURE         " * 2.1. This is your license from I
64 #define LINES_IN_LINUX_HEADER         34
65 #define LINUX_HEADER_SIGNATURE        " * THIS SOFTWARE IS PROVIDED BY THE
66 #define LINES_IN_AS_L_HEADER          29 /* Header as output from disassem

68 #define ASRC_MAX_FILE_SIZE             (1024 * 100)

70 #define FILE_TYPE_SOURCE                1
71 #define FILE_TYPE_HEADER                2
72 #define FILE_TYPE_DIRECTORY             3
73 #define FILE_TYPE_PATCH                 4

75 #define CVT_COUNT_TABS                  0x00000001
76 #define CVT_COUNT_NON_ANSI_COMMENTS     0x00000002
77 #define CVT_TRIM_LINES                   0x00000004
78 #define CVT_CHECK_BRACES                 0x00000008
79 #define CVT_COUNT_LINES                  0x00000010
80 #define CVT_BRACES_ON_SAME_LINE          0x00000020
81 #define CVT_MIXED_CASE_TO_UNDERSCORES   0x00000040
82 #define CVT_LOWER_CASE_IDENTIFIERS       0x00000080
83 #define CVT_REMOVE_DEBUG_MACROS          0x00000100
84 #define CVT_TRIM_WHITESPACE              0x00000200 /* Should be after all l
85 #define CVT_REMOVE_EMPTY_BLOCKS          0x00000400 /* Should be after trimm
86 #define CVT_REDUCE_TYPEDEFS              0x00000800
87 #define CVT_COUNT_SHORTMULTILINE_COMMENTS 0x00001000
88 #define CVT_SPACES_TO_TABS4              0x40000000 /* Tab conversion should
89 #define CVT_SPACES_TO_TABS8              0x80000000 /* Tab conversion should

91 #define FLG_DEFAULT_FLAGS                0x00000000
92 #define FLG_NO_CARRIAGE_RETURNS          0x00000001
93 #define FLG_NO_FILE_OUTPUT                0x00000002
94 #define FLG_LOWERCASE_DIRNAMES           0x00000004

96 #define AS_START_IGNORE                  "/*!"
97 #define AS_STOP_IGNORE                   "!*/"

100 /* Globals */

102 extern UINT32          Gbl_Files;
103 extern UINT32          Gbl_MissingBraces;
104 extern UINT32          Gbl_Tabs;
105 extern UINT32          Gbl_NonAnsiComments;
106 extern UINT32          Gbl_SourceLines;
107 extern UINT32          Gbl_WhiteLines;
108 extern UINT32          Gbl_CommentLines;
109 extern UINT32          Gbl_LongLines;
110 extern UINT32          Gbl_TotalLines;
111 extern UINT32          Gbl_HeadersSize;
112 extern UINT32          Gbl_HeaderLines;
113 extern struct stat      Gbl_StatBuf;
114 extern char             *Gbl_FileBuffer;
115 extern UINT32          Gbl_TotalSize;
116 extern UINT32          Gbl_FileSize;
117 extern UINT32          Gbl_FileType;
118 extern BOOLEAN          Gbl_VerboseMode;
119 extern BOOLEAN          Gbl_QuietMode;
120 extern BOOLEAN          Gbl_BatchMode;
121 extern BOOLEAN          Gbl_MadeChanges;
122 extern BOOLEAN          Gbl_Overwrite;
123 extern BOOLEAN          Gbl_WidenDeclarations;
124 extern BOOLEAN          Gbl_IgnoreLoneLineFeeds;
125 extern BOOLEAN          Gbl_HasLoneLineFeeds;
```

```

126 extern BOOLEAN          Gbl_Cleanup;
127 extern BOOLEAN          Gbl_IgnoreTranslationEscapes;
128 extern void              *Gbl_StructDefs;

130 #define PARAM_LIST(pl)    pl
131 #define TERSE_PRINT(a)    if (!Gbl_VerboseMode) printf PARAM_LIST(a)
132 #define VERBOSE_PRINT(a)  if (Gbl_VerboseMode) printf PARAM_LIST(a)

134 #define REPLACE_WHOLE_WORD 0x00
135 #define REPLACE_SUBSTRINGS 0x01
136 #define REPLACE_MASK       0x01

138 #define EXTRA_INDENT_C     0x02

141 /* Conversion table structs */

143 typedef struct acpi_string_table
144 {
145     char          *Target;
146     char          *Replacement;
147     UINT8         Type;
148 } ACPI_STRING_TABLE;

149 } ACPI_STRING_TABLE;

152 typedef struct acpi_typed_identifier_table
153 {
154     char          *Identifier;
155     UINT8         Type;
156 } ACPI_TYPED_IDENTIFIER_TABLE;

157 } ACPI_TYPED_IDENTIFIER_TABLE;

159 #define SRC_TYPE_SIMPLE      0
160 #define SRC_TYPE_STRUCT     1
161 #define SRC_TYPE_UNION       2

164 typedef struct acpi_identifier_table
165 {
166     char          *Identifier;
167 } ACPI_IDENTIFIER_TABLE;

168 } ACPI_IDENTIFIER_TABLE;

170 typedef struct acpi_conversion_table
171 {
172     char          *NewHeader;
173     UINT32         Flags;
174 } ACPI_CONVERSION_TABLE;

175 ACPI_TYPED_IDENTIFIER_TABLE *LowerCaseTable;

177 ACPI_STRING_TABLE          *SourceStringTable;
178 ACPI_IDENTIFIER_TABLE      *SourceLineTable;
179 ACPI_IDENTIFIER_TABLE      *SourceConditionalTable;
180 ACPI_IDENTIFIER_TABLE      *SourceMacroTable;
181 ACPI_TYPED_IDENTIFIER_TABLE *SourceStructTable;
182 ACPI_IDENTIFIER_TABLE      *SourceSpecialMacroTable;
183 UINT32                     SourceFunctions;

185 ACPI_STRING_TABLE          *HeaderStringTable;
186 ACPI_IDENTIFIER_TABLE      *HeaderLineTable;
187 ACPI_IDENTIFIER_TABLE      *HeaderConditionalTable;
188 ACPI_IDENTIFIER_TABLE      *HeaderMacroTable;
189 ACPI_TYPED_IDENTIFIER_TABLE *HeaderStructTable;
190 ACPI_IDENTIFIER_TABLE      *HeaderSpecialMacroTable;
191 UINT32                     HeaderFunctions;

```

```

193 ACPI_STRING_TABLE          *PatchStringTable;
194 ACPI_IDENTIFIER_TABLE      *PatchLineTable;
195 ACPI_IDENTIFIER_TABLE      *PatchConditionalTable;
196 ACPI_IDENTIFIER_TABLE      *PatchMacroTable;
197 ACPI_TYPED_IDENTIFIER_TABLE *PatchStructTable;
198 ACPI_IDENTIFIER_TABLE      *PatchSpecialMacroTable;
199 UINT32                     PatchFunctions;

201 } ACPI_CONVERSION_TABLE;

204 /* Conversion tables */

206 extern ACPI_CONVERSION_TABLE LinuxConversionTable;
207 extern ACPI_CONVERSION_TABLE CleanupConversionTable;
208 extern ACPI_CONVERSION_TABLE StatsConversionTable;
209 extern ACPI_CONVERSION_TABLE CustomConversionTable;
210 extern ACPI_CONVERSION_TABLE LicenseConversionTable;
211 extern ACPI_CONVERSION_TABLE IndentConversionTable;

214 /* Prototypes */

216 char *
217 AsSkipUntilChar (
218     char          *Buffer,
219     char          Target);

221 char *
222 AsSkipPastChar (
223     char          *Buffer,
224     char          Target);

226 char *
227 AsReplaceData (
228     char          *Buffer,
229     UINT32         LengthToRemove,
230     char          *BufferToAdd,
231     UINT32         LengthToAdd);

233 int
234 AsReplaceString (
235     char          *Target,
236     char          *Replacement,
237     UINT8         Type,
238     char          *Buffer);

240 int
241 AsLowerCaseString (
242     char          *Target,
243     char          *Buffer);

245 void
246 AsRemoveLine (
247     char          *Buffer,
248     char          *Keyword);

250 void
251 AsRemoveMacro (
252     char          *Buffer,
253     char          *Keyword);

255 void
256 AsCheckForBraces (
257     char          *Buffer,

```

```

258     char                *Filename);

260 void
261 AsTrimLines (
262     char                *Buffer,
263     char                *Filename);

265 void
266 AsMixedCaseToUnderscores (
267     char                *Buffer,
268     char                *Filename);

270 void
271 AsCountTabs (
272     char                *Buffer,
273     char                *Filename);

275 void
276 AsBracesOnSameLine (
277     char                *Buffer);

279 void
280 AsLowerCaseIdentifiers (
281     char                *Buffer);

283 void
284 AsReduceTypedefs (
285     char                *Buffer,
286     char                *Keyword);

288 void
289 AsRemoveDebugMacros (
290     char                *Buffer);

292 void
293 AsRemoveEmptyBlocks (
294     char                *Buffer,
295     char                *Filename);

297 void
298 AsCleanupSpecialMacro (
299     char                *Buffer,
300     char                *Keyword);

302 void
303 AsCountSourceLines (
304     char                *Buffer,
305     char                *Filename);

307 void
308 AsCountNonAnsiComments (
309     char                *Buffer,
310     char                *Filename);

312 void
313 AsTrimWhitespace (
314     char                *Buffer);

316 void
317 AsTabify4 (
318     char                *Buffer);

320 void
321 AsTabify8 (
322     char                *Buffer);

```

```

324 void
325 AsRemoveConditionalCompile (
326     char                *Buffer,
327     char                *Keyword);

329 ACPI_NATIVE_INT
330 AsProcessTree (
331     ACPI_CONVERSION_TABLE *ConversionTable,
332     char                *SourcePath,
333     char                *TargetPath);

335 int
336 AsGetFile (
337     char                *FileName,
338     char                *FileBuffer,
339     UINT32              *FileSize);

341 int
342 AsPutFile (
343     char                *Pathname,
344     char                *FileBuffer,
345     UINT32              SystemFlags);

347 void
348 AsReplaceHeader (
349     char                *Buffer,
350     char                *NewHeader);

352 void
353 AsConvertFile (
354     ACPI_CONVERSION_TABLE *ConversionTable,
355     char                *FileBuffer,
356     char                *Filename,
357     ACPI_NATIVE_INT      FileType);

359 ACPI_NATIVE_INT
360 AsProcessOneFile (
361     ACPI_CONVERSION_TABLE *ConversionTable,
362     char                *SourcePath,
363     char                *TargetPath,
364     int                MaxPathLength,
365     char                *Filename,
366     ACPI_NATIVE_INT      FileType);

368 ACPI_NATIVE_INT
369 AsCheckForDirectory (
370     char                *SourceDirPath,
371     char                *TargetDirPath,
372     char                *Filename,
373     char                **SourcePath,
374     char                **TargetPath);

376 void
377 AsRemoveExtraLines (
378     char                *FileBuffer,
379     char                *Filename);

381 void
382 AsRemoveSpacesAfterPeriod (
383     char                *FileBuffer,
384     char                *Filename);

386 BOOLEAN
387 AsMatchExactWord (
388     char                *Word,
389     UINT32              WordLength);

```

```
391 void
392 AsPrint (
393     char                *Message,
394     UINT32              Count,
395     char                *Filename);
396
397 void
398 AsInsertPrefix (
399     char                *Buffer,
400     char                *Keyword,
401     UINT8              Type);
402
403 char *
404 AsInsertData (
405     char                *Buffer,
406     char                *BufferToAdd,
407     UINT32              LengthToAdd);
408
409 char *
410 AsRemoveData (
411     char                *StartPointer,
412     char                *EndPointer);
413
414 void
415 AsInsertCarriageReturns (
416     char                *Buffer);
417
418 void
419 AsConvertToLineFeeds (
420     char                *Buffer);
421
422 void
423 AsStrlwr (
424     char                *SrcString);
```

new/usr/src/common/acpica/tools/acpiscrc/ascase.c

1

```
*****
18586 Fri Jan 17 07:31:18 2014
new/usr/src/common/acpica/tools/acpiscrc/ascase.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: ascase - Source conversion - lower/upper case utilities
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acpiscrc.h"

46 /* Local prototypes */

48 void
49 AsUpperCaseTokens (
50     char                *Buffer,
51     char                *PrefixString);

54 /*****
55 *
56 * FUNCTION:    AsLowerCaseString
57 *
58 * DESCRIPTION: LowerCase all instances of a target string with a replacement
59 * string. Returns count of the strings replaced.
60 */
```

new/usr/src/common/acpica/tools/acpiscrc/ascase.c

2

```
60 *
61 *****/

63 int
64 AsLowerCaseString (
65     char                *Target,
66     char                *Buffer)
67 {
68     char                *SubString1;
69     char                *SubString2;
70     char                *SubBuffer;
71     int                 TargetLength;
72     int                 LowerCaseCount = 0;
73     int                 i;

76     TargetLength = strlen (Target);

78     SubBuffer = Buffer;
79     SubString1 = Buffer;

81     while (SubString1)
82     {
83         /* Find the target string */

85         SubString1 = strstr (SubBuffer, Target);
86         if (!SubString1)
87         {
88             return (LowerCaseCount);
89         }

91         /*
92          * Check for translation escape string -- means to ignore
93          * blocks of code while replacing
94          */
95         if (Gbl_IgnoreTranslationEscapes)
96         {
97             SubString2 = NULL;
98         }
99         else
100         {
101             SubString2 = strstr (SubBuffer, AS_START_IGNORE);
102         }

104         if ((SubString2) &&
105             (SubString2 < SubString1))
106         {
107             /* Find end of the escape block starting at "SubString2" */

109             SubString2 = strstr (SubString2, AS_STOP_IGNORE);
110             if (!SubString2)
111             {
112                 /* Didn't find terminator */

114                 return (LowerCaseCount);
115             }

117             /* Move buffer to end of escape block and continue */

119             SubBuffer = SubString2;
120         }

122         /* Do the actual replace if the target was found */

124     else
125     {
```

```

126     if (!AsMatchExactWord (SubString1, TargetLength))
127     {
128         SubBuffer = SubString1 + 1;
129         continue;
130     }
131
132     for (i = 0; i < TargetLength; i++)
133     {
134         SubString1[i] = (char) tolower ((int) SubString1[i]);
135     }
136
137     SubBuffer = SubString1 + TargetLength;
138
139     if ((Gbl_WidenDeclarations) && (!Gbl_StructDefs))
140     {
141         if ((SubBuffer[0] == ' ') && (SubBuffer[1] == ' '))
142         {
143             AsInsertData (SubBuffer, "      ", 8);
144         }
145     }
146
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         }
201
202         /* Ignore standard escape sequences (\n, \r, etc.) Not Hex or Octal esc
203
204         if (*SubBuffer == '\\')
205         {
206             SubBuffer += 2;
207             continue;
208         }
209
210         /* Ignore single-quoted characters */
211
212         if (*SubBuffer == '\')
213         {
214             SubBuffer += 3;
215             continue;
216         }
217
218         /* Ignore standard double-quoted strings */
219
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                 }
234             }
235
236             /* Handle escape sequences */
237
238             if (*SubBuffer == '\\')
239             {
240                 SubBuffer++;
241             }
242             SubBuffer++;
243             SubBuffer++;
244             continue;
245         }
246     }
247
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 Uppercase/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                 break;
362             }
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      }
397
398      /* Find the end of this identifier (token) */
399
400      TokenEnd = SubBuffer - 1;
401      while ((isalnum ((int) *TokenEnd)) ||
402             (*TokenEnd == '_'))
403      {
404          TokenEnd++;
405      }
406
407      SubString = TokenEnd;
408      Length = 0;
409
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          }
423
424          SubString++;
425      }
426
427      if (!Length)
428      {
429          Length = strlen (&SubBuffer[0]);
430      }
431
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;
441
442      /* Insert the underscore */
443
444      memmove (&SubBuffer[1], &SubBuffer[0], Length + 1);
445      SubBuffer[0] = '_';
446
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      }
457
458      /* Lower case the original upper case letter */
459
460      SubBuffer[1] = (char) tolower ((int) SubBuffer[1]);
461      SubBuffer += 2;
462      }
463      }
464
465      SubBuffer++;
466      }
467      }
468
469      /*****
470       *
471       * FUNCTION:      AsLowerCaseIdentifiers
472       * DESCRIPTION:  Converts mixed case identifiers to lower case. Leaves comments,
473       *               quoted strings, and all-upper-case macros alone.
474       *
475       *****/
476
477      void
478      AsLowerCaseIdentifiers (
479          char          *Buffer)
480      {
481          char          *SubBuffer = Buffer;
482
483          while (*SubBuffer)
484          {
485              /*
486               * Check for translation escape string -- means to ignore
487               * blocks of code while replacing
488               */
489              if ((SubBuffer[0] == '/') &&
490                  (SubBuffer[1] == '*') &&
491                  (SubBuffer[2] == '!'))
492              {
493                  SubBuffer = strstr (SubBuffer, "/*/");
494                  if (!SubBuffer)
495                  {
496                      return;
497                  }
498              }
499
500              /* Ignore comments */
501
502              if ((SubBuffer[0] == '/') &&
503                  (SubBuffer[1] == '*'))
504              {
505                  SubBuffer = strstr (SubBuffer, "*/");
506                  if (!SubBuffer)
507                  {
508                      return;
509                  }
510              }
511
512              SubBuffer += 2;
513          }
514
515          /* Ignore quoted strings */
516
517          if ((SubBuffer[0] == '\"') && (SubBuffer[1] != '\"'))
518          {
519              SubBuffer++;

```



```

523      /* Find the closing quote */
524
525      while (SubBuffer[0])
526      {
527          /* Ignore escaped quote characters */
528
529          if (SubBuffer[0] == '\\')
530          {
531              SubBuffer++;
532          }
533          else if (SubBuffer[0] == '"')
534          {
535              SubBuffer++;
536              break;
537          }
538          SubBuffer++;
539      }
540
541      if (!SubBuffer[0])
542      {
543          return;
544      }
545
546      /*
547       * Only lower case if we have an upper followed by a lower
548       * This leaves the all-uppercase things (macros, etc.) intact
549       */
550
551      if ((isupper ((int) SubBuffer[0])) &&
552          (islower ((int) SubBuffer[1])))
553      {
554          Gbl_MadeChanges = TRUE;
555          *SubBuffer = (char) tolower ((int) *SubBuffer);
556      }
557
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  *****/
571
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;
582
583     SubBuffer = Buffer;
584
585     while (SubBuffer)
586     {

```

```

588     SubBuffer = strstr (SubBuffer, PrefixString);
589     if (SubBuffer)
590     {
591         TokenEnd = SubBuffer;
592         while ((isalnum ((int) *TokenEnd)) || (*TokenEnd == '_'))
593         {
594             TokenEnd++;
595         }
596
597         for (i = 0; i < (TokenEnd - SubBuffer); i++)
598         {
599             if ((islower ((int) SubBuffer[i])) &&
600                 (isupper ((int) SubBuffer[i+1])))
601             {
602
603                 SubString = TokenEnd;
604                 Length = 0;
605
606                 while (*SubString != '\n')
607                 {
608                     if ((SubString[0] == ' ') &&
609                         (SubString[1] == ' '))
610                     {
611                         Length = SubString - &SubBuffer[i] - 2;
612                         break;
613                     }
614
615                     SubString++;
616                 }
617
618                 if (!Length)
619                 {
620                     Length = strlen (&SubBuffer[i+1]);
621                 }
622
623                 memmove (&SubBuffer[i+2], &SubBuffer[i+1], (Length+1));
624                 SubBuffer[i+1] = '_';
625                 i += 2;
626                 TokenEnd++;
627             }
628         }
629
630         for (i = 0; i < (TokenEnd - SubBuffer); i++)
631         {
632             SubBuffer[i] = (char) toupper ((int) SubBuffer[i]);
633         }
634
635         SubBuffer = TokenEnd;
636     }
637 }
638 }

```

new/usr/src/common/acpica/tools/acpiscrc/asconVRT.c

1

```
*****
39065 Fri Jan 17 07:31:18 2014
new/usr/src/common/acpica/tools/acpiscrc/asconVRT.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: asconVRT - Source conversion code
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acpiscrc.h"

46 /* Local prototypes */

48 char *
49 AsCheckAndSkipLiterals (
50     char          *Buffer,
51     UINT32         *TotalLines);

53 UINT32
54 AsCountLines (
55     char          *Buffer,
56     char          *Filename);

58 /* Opening signature of the Intel legal header */
```

new/usr/src/common/acpica/tools/acpiscrc/asconVRT.c

2

```
60 char          *HeaderBegin = "/******

63 /*****
64 *
65 * FUNCTION:      AsRemoveExtraLines
66 *
67 * DESCRIPTION: Remove all extra lines at the start and end of the file.
68 *
69 *****/

71 void
72 AsRemoveExtraLines (
73     char          *FileBuffer,
74     char          *Filename)
75 {
76     char          *FileEnd;
77     int           Length;

80     /* Remove any extra lines at the start of the file */

82     while (*FileBuffer == '\n')
83     {
84         printf ("Removing extra line at start of file: %s\n", Filename);
85         AsRemoveData (FileBuffer, FileBuffer + 1);
86     }

88     /* Remove any extra lines at the end of the file */

90     Length = strlen (FileBuffer);
91     FileEnd = FileBuffer + (Length - 2);

93     while (*FileEnd == '\n')
94     {
95         printf ("Removing extra line at end of file: %s\n", Filename);
96         AsRemoveData (FileEnd, FileEnd + 1);
97         FileEnd--;
98     }
99 }

102 /*****
103 *
104 * FUNCTION:      AsRemoveSpacesAfterPeriod
105 *
106 * DESCRIPTION: Remove an extra space after a period.
107 *
108 *****/

110 void
111 AsRemoveSpacesAfterPeriod (
112     char          *FileBuffer,
113     char          *Filename)
114 {
115     int           ReplaceCount = 0;
116     char          *Possible;

119     Possible = FileBuffer;
120     while (Possible)
121     {
122         Possible = strstr (Possible, ". ");
123         if (Possible)
124         {
125             if ((*Possible - 1) == '.') ||
```

```

126         (*(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 if (ReplaceCount)
138 {
139     printf ("Removed %d extra blanks after a period: %s\n",
140           ReplaceCount, Filename);
141 }
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:      AsCheckForBraces
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 (!(strcmp (" 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                         , SubBuffer);
313                 }
314             }
315             else if (!(strcmp (" 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                     , SubBuffer);
332             }
333         }
334         else if (!(strcmp (" 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                         , SubBuffer);
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     }
398
399     SubBuffer++;
400 }
401
402 if (StartWhiteSpace)
403 {
404     SpaceCount += (SubBuffer - StartWhiteSpace);
405
406     /* Remove the spaces */
407
408     SubBuffer = AsRemoveData (StartWhiteSpace, SubBuffer);
409     StartWhiteSpace = NULL;
410 }
411
412 SubBuffer++;
413 }
414
415 Exit:
416 if (SpaceCount)
417 {
418     Gbl_MadeChanges = TRUE;
419     AsPrint ("Extraneous spaces removed", SpaceCount, Filename);
420 }
421 }
422 }
423
424 /*****
425 *
426 * FUNCTION:    AsTrimWhitespace
427 *
428 * DESCRIPTION: Remove "excess" blank lines - any more than 2 blank lines.
429 *              this can happen during the translation when lines are removed.
430 *
431 *
432 *****/
433
434 void
435 AsTrimWhitespace (
436     char        *Buffer)
437 {
438     int          ReplaceCount = 1;
439
440     while (ReplaceCount)
441     {
442         ReplaceCount = AsReplaceString ("\n\n\n", "\n\n", REPLACE_SUBSTRINGS);
443     }
444 }
445
446 /*****
447 *
448 * FUNCTION:    AsReplaceHeader
449 *
450 * DESCRIPTION: Replace the default Intel legal header with a new header
451 *
452 *
453 *****/

```

```

456 void
457 AsReplaceHeader (
458     char        *Buffer,
459     char        *NewHeader)
460 {
461     char        *SubBuffer;
462     char        *TokenEnd;
463
464     /* Find the original header */
465
466     SubBuffer = strstr (Buffer, HeaderBegin);
467     if (!SubBuffer)
468     {
469         return;
470     }
471
472     /* Find the end of the original header */
473
474     TokenEnd = strstr (SubBuffer, "**/");
475     TokenEnd = AsSkipPastChar (TokenEnd, '\n');
476
477     /* Delete old header, insert new one */
478
479     AsReplaceData (SubBuffer, TokenEnd - SubBuffer, NewHeader, strlen (NewHeader));
480 }
481
482 /*****
483 *
484 * FUNCTION:    AsReplaceString
485 *
486 * DESCRIPTION: Replace all instances of a target string with a replacement
487 *              string. Returns count of the strings replaced.
488 *
489 *****/
490
491 int
492 AsReplaceString (
493     char        *Target,
494     char        *Replacement,
495     UINT8       Type,
496     char        *Buffer)
497 {
498     char        *SubString1;
499     char        *SubString2;
500     char        *SubBuffer;
501     int          TargetLength;
502     int          ReplacementLength;
503     int          ReplaceCount = 0;
504
505     TargetLength = strlen (Target);
506     ReplacementLength = strlen (Replacement);
507
508     SubBuffer = Buffer;
509     SubString1 = Buffer;
510
511     while (SubString1)
512     {
513         /* Find the target string */
514
515         SubString1 = strstr (SubBuffer, Target);
516         if (!SubString1)
517         {
518             return (ReplaceCount);
519         }
520     }

```

```

522     }
523
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     }
536
537     if ((SubString2) &&
538         (SubString2 < SubString1))
539     {
540         /* Find end of the escape block starting at "SubString2" */
541
542         SubString2 = strstr (SubString2, AS_STOP_IGNORE);
543         if (!SubString2)
544         {
545             /* Didn't find terminator */
546
547             return (ReplaceCount);
548         }
549
550         /* Move buffer to end of escape block and continue */
551
552         SubBuffer = SubString2;
553     }
554
555     /* Do the actual replace if the target was found */
556
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
567             SubBuffer = AsReplaceData (SubString1, TargetLength, Replacement, Re
568
569             if ((Type & EXTRA_INDENT_C) &&
570                 (!Gbl_StructDefs))
571             {
572                 SubBuffer = AsInsertData (SubBuffer, "      ", 8);
573             }
574
575             ReplaceCount++;
576         }
577     }
578
579     return (ReplaceCount);
580 }
581
582 /*****
583 *
584 * FUNCTION:    AsConvertToLineFeeds
585 */

```

```

588 * DESCRIPTION:
589 *
590 *****/
591
592 void
593 AsConvertToLineFeeds (
594     char                *Buffer)
595 {
596     char                *SubString;
597     char                *SubBuffer;
598
599     SubBuffer = Buffer;
600     SubString = Buffer;
601
602     while (SubString)
603     {
604         /* Find the target string */
605
606         SubString = strstr (SubBuffer, "\r\n");
607         if (!SubString)
608         {
609             return;
610         }
611
612         SubBuffer = AsReplaceData (SubString, 1, NULL, 0);
613     }
614     return;
615 }
616
617 /*****
618 *
619 * FUNCTION:    AsInsertCarriageReturns
620 *
621 * DESCRIPTION:
622 *
623 *****/
624
625 void
626 AsInsertCarriageReturns (
627     char                *Buffer)
628 {
629     char                *SubString;
630     char                *SubBuffer;
631
632     SubBuffer = Buffer;
633     SubString = Buffer;
634
635     while (SubString)
636     {
637         /* Find the target string */
638
639         SubString = strstr (SubBuffer, "\n");
640         if (!SubString)
641         {
642             return;
643         }
644
645         SubBuffer = AsInsertData (SubString, "\r", 1);
646         SubBuffer += 1;
647     }
648     return;
649 }

```

```

655 /*****
656 *
657 * FUNCTION:    AsBracesOnSameLine
658 *
659 * DESCRIPTION: Move opening braces up to the same line as an if, for, else,
660 *              or while statement (leave function opening brace on separate
661 *              line).
662 *
663 *****/
664
665 void
666 AsBracesOnSameLine (
667     char          *Buffer)
668 {
669     char          *SubBuffer = Buffer;
670     char          *Beginning;
671     char          *StartOfThisLine;
672     char          *Next;
673     BOOLEAN       BlockBegin = TRUE;
674
675     while (*SubBuffer)
676     {
677         /* Ignore comments */
678
679         if ((SubBuffer[0] == '/') &&
680             (SubBuffer[1] == '*'))
681         {
682             SubBuffer = strstr (SubBuffer, "*/");
683             if (!SubBuffer)
684             {
685                 return;
686             }
687
688             SubBuffer += 2;
689             continue;
690         }
691
692         /* Ignore quoted strings */
693
694         if (*SubBuffer == '\\"')
695         {
696             SubBuffer++;
697             SubBuffer = AsSkipPastChar (SubBuffer, '\\');
698             if (!SubBuffer)
699             {
700                 return;
701             }
702         }
703
704         if (!strcmp ("\n", SubBuffer, 2))
705         {
706             /*
707              * A newline followed by a closing brace closes a function
708              * or struct or initializer block
709              */
710             BlockBegin = TRUE;
711         }
712
713         /*
714          * Move every standalone brace up to the previous line
715          * Check for digit will ignore initializer lists surrounded by braces.
716          * This will work until we need more complex detection.
717          */
718         if ((*SubBuffer == '{') && !isdigit ((int) SubBuffer[1]))

```

```

720     {
721         if (BlockBegin)
722         {
723             BlockBegin = FALSE;
724         }
725         else
726         {
727             /*
728              * Backup to previous non-whitespace
729              */
730             Beginning = SubBuffer - 1;
731             while ((*Beginning == ' ') ||
732                 (*Beginning == '\n'))
733             {
734                 Beginning--;
735             }
736
737             StartOfThisLine = Beginning;
738             while (*StartOfThisLine != '\n')
739             {
740                 StartOfThisLine--;
741             }
742
743             /*
744              * Move the brace up to the previous line, UNLESS:
745              *
746              * 1) There is a conditional compile on the line (starts with '#
747              * 2) Previous line ends with an '=' (Start of initializer block
748              * 3) Previous line ends with a comma (part of an init list)
749              * 4) Previous line ends with a backslash (part of a macro)
750              */
751             if ((StartOfThisLine[1] != '#') &&
752                 (*Beginning != '\\') &&
753                 (*Beginning != '/') &&
754                 (*Beginning != '{') &&
755                 (*Beginning != '=') &&
756                 (*Beginning != '\\'))
757             {
758                 Beginning++;
759                 SubBuffer++;
760             }
761
762             Gbl_MadeChanges = TRUE;
763
764 #ifdef ADD_EXTRA_WHITESPACE
765             AsReplaceData (Beginning, SubBuffer - Beginning, " {\n", 3);
766 #else
767             /* Find non-whitespace start of next line */
768
769             Next = SubBuffer + 1;
770             while ((*Next == ' ') ||
771                 (*Next == '\t'))
772             {
773                 Next++;
774             }
775
776             /* Find non-whitespace start of this line */
777
778             StartOfThisLine++;
779             while ((*StartOfThisLine == ' ') ||
780                 (*StartOfThisLine == '\t'))
781             {
782                 StartOfThisLine++;
783             }
784
785             /*
786              * Must be a single-line comment to need more whitespace

```

```

786         * 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             (!strcmp (StartOfThisLine, "else if", 7) ||
794              !strcmp (StartOfThisLine, "else while", 10) ||
795              strcmp (StartOfThisLine, "else", 4)))
796         {
797             AsReplaceData (Beginning, SubBuffer - Beginning, " {\n",
798                             2);
799         }
800         else
801         {
802             AsReplaceData (Beginning, SubBuffer - Beginning, " {", 2);
803         }
804     }
805 }
806
807 SubBuffer++;
808 }
809 }
810 }

```

```

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     while (*SubBuffer)
832     {
833         if (*SubBuffer == '\n')
834         {
835             Column = 0;
836         }
837         else
838         {
839             Column++;
840         }
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 }

```

```

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;
919     UINT32              SpaceCount = 0;
920     UINT32              Column = 0;
921     UINT32              TabCount = 0;
922     UINT32              LastLineTabCount = 0;
923     UINT32              LastLineColumnStart = 0;
924     UINT32              ThisColumnStart = 0;
925     UINT32              ThisTabCount = 0;
926     char                *FirstNonBlank = NULL;

929     while (*SubBuffer)
930     {
931         if (*SubBuffer == '\n')
932         {
933             /* This is a standalone blank line */

935             FirstNonBlank = NULL;
936             Column = 0;
937             SpaceCount = 0;
938             TabCount = 0;
939             SubBuffer++;
940             continue;
941         }

943         if (!FirstNonBlank)
944         {
945             /* Find the first non-blank character on this line */

947             FirstNonBlank = SubBuffer;
948             while (*FirstNonBlank == ' ')
949             {
950                 FirstNonBlank++;
951             }

953             /*
954              * This mechanism limits the difference in tab counts from
955              * line to line. It helps avoid the situation where a second
956              * continuation line (which was indented correctly for tabs=4) would
957              * get indented off the screen if we just blindly converted to tabs.
958              */
959             ThisColumnStart = FirstNonBlank - SubBuffer;

961             if (LastLineTabCount == 0)
962             {
963                 ThisTabCount = 0;
964             }
965             else if (ThisColumnStart == LastLineColumnStart)
966             {
967                 ThisTabCount = LastLineTabCount - 1;
968             }
969             else
970             {
971                 ThisTabCount = LastLineTabCount + 1;
972             }
973         }

975         Column++;

977         /* Check if we are in a comment */

979         if ((SubBuffer[0] == '/') &&
980             (SubBuffer[1] == '/'))
981         {
982             SpaceCount = 0;
983             SubBuffer += 2;

```

```

985         if (*SubBuffer == '\n')
986         {
987             if (TabCount > 0)
988             {
989                 LastLineTabCount = TabCount;
990                 TabCount = 0;
991             }
992             FirstNonBlank = NULL;
993             LastLineColumnStart = ThisColumnStart;
994             SubBuffer++;
995         }

997         continue;
998     }

1000     /* Check for comment open */

1002     if ((SubBuffer[0] == '/') &&
1003         (SubBuffer[1] == '/'))
1004     {
1005         /* Find the end of the comment, it must exist */

1007         CommentEnd = strstr (SubBuffer, "/*");
1008         if (!CommentEnd)
1009         {
1010             return;
1011         }

1013         /* Toss the rest of this line or single-line comment */

1015         while ((SubBuffer < CommentEnd) &&
1016             (*SubBuffer != '\n'))
1017         {
1018             SubBuffer++;
1019         }

1021         if (*SubBuffer == '\n')
1022         {
1023             if (TabCount > 0)
1024             {
1025                 LastLineTabCount = TabCount;
1026                 TabCount = 0;
1027             }
1028             FirstNonBlank = NULL;
1029             LastLineColumnStart = ThisColumnStart;
1030         }

1032         SpaceCount = 0;
1033         continue;
1034     }

1036     /* Ignore quoted strings */

1038     if ((!CommentEnd) && (*SubBuffer == '\"'))
1039     {
1040         SubBuffer++;
1041         SubBuffer = AsSkipPastChar (SubBuffer, '\"');
1042         if (!SubBuffer)
1043         {
1044             return;
1045         }
1046         SpaceCount = 0;
1047     }

1049     if (*SubBuffer != ' ')

```

```

1050     {
1051         /* Not a space, skip to end of line */
1052
1053         SubBuffer = AsSkipUntilChar (SubBuffer, '\n');
1054         if (!SubBuffer)
1055         {
1056             return;
1057         }
1058         if (TabCount > 0)
1059         {
1060             LastLineTabCount = TabCount;
1061             TabCount = 0;
1062         }
1063
1064         FirstNonBlank = NULL;
1065         LastLineColumnStart = ThisColumnStart;
1066         Column = 0;
1067         SpaceCount = 0;
1068     }
1069     else
1070     {
1071         /* Another space */
1072
1073         SpaceCount++;
1074
1075         if (SpaceCount >= 4)
1076         {
1077             /* Replace this group of spaces with a tab character */
1078
1079             SpaceCount = 0;
1080
1081             NewSubBuffer = SubBuffer - 3;
1082
1083             if (TabCount <= ThisTabCount ? (ThisTabCount + 1) : 0)
1084             {
1085                 *NewSubBuffer = '\t';
1086                 NewSubBuffer++;
1087                 SubBuffer++;
1088                 TabCount++;
1089             }
1090
1091             /* Remove the spaces */
1092
1093             SubBuffer = AsRemoveData (NewSubBuffer, SubBuffer);
1094             continue;
1095         }
1096     }
1097     SubBuffer++;
1098 }
1099
1100
1103 /*****
1104  *
1105  * FUNCTION:    AsCountLines
1106  *
1107  * DESCRIPTION: Count the number of lines in the input buffer. Also count
1108  *              the number of long lines (lines longer than 80 chars).
1109  *
1110  *****/
1111
1112 UINT32
1113 AsCountLines (
1114     char          *Buffer,
1115     char          *Filename)

```

```

1116 {
1117     char          *SubBuffer = Buffer;
1118     char          *EndOfLine;
1119     UINT32        LineCount = 0;
1120     UINT32        LongLineCount = 0;
1121
1122     while (*SubBuffer)
1123     {
1124         EndOfLine = AsSkipUntilChar (SubBuffer, '\n');
1125         if (!EndOfLine)
1126         {
1127             Gbl_TotalLines += LineCount;
1128             return (LineCount);
1129         }
1130
1131         if ((EndOfLine - SubBuffer) > 80)
1132         {
1133             LongLineCount++;
1134             VERBOSE_PRINT (("long: %.80s\n", SubBuffer));
1135         }
1136
1137         LineCount++;
1138         SubBuffer = EndOfLine + 1;
1139     }
1140
1141     if (LongLineCount)
1142     {
1143         VERBOSE_PRINT ("%u Lines longer than 80 found in %s\n", LongLineCount,
1144             Gbl_LongLines += LongLineCount);
1145     }
1146
1147     Gbl_TotalLines += LineCount;
1148     return (LineCount);
1149 }
1150
1153 /*****
1154  *
1155  * FUNCTION:    AsCountTabs
1156  *
1157  * DESCRIPTION: Simply count the number of tabs in the input file buffer
1158  *
1159  *****/
1160
1161 void
1162 AsCountTabs (
1163     char          *Buffer,
1164     char          *Filename)
1165 {
1166     UINT32        i;
1167     UINT32        TabCount = 0;
1168
1169     for (i = 0; Buffer[i]; i++)
1170     {
1171         if (Buffer[i] == '\t')
1172         {
1173             TabCount++;
1174         }
1175     }
1176
1177     if (TabCount)
1178     {
1179         AsPrint ("Tabs found", TabCount, Filename);
1180         Gbl_Tabs += TabCount;
1181     }

```

```

1182     }
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-ansi) */

```

```

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 (!strcmp (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 }

1448 #ifdef ACPI_FUTURE_IMPLEMENTATION
1449 /*****
1450 *
1451 * FUNCTION:      AsTrimComments
1452 *
1453 * DESCRIPTION: Finds 3-line comments with only a single line of text
1454 *
1455 *****/
1457 void
1458 AsTrimComments (
1459     char          *Buffer,
1460     char          *Filename)
1461 {
1462     char          *SubBuffer = Buffer;
1463     char          *Ptr1;
1464     char          *Ptr2;
1465     UINT32        LineCount;
1466     UINT32        ShortCommentCount = 0;

1469     while (1)
1470     {
1471         /* Find comment open, within procedure level */

1473         SubBuffer = strstr (SubBuffer, " /*");
1474         if (!SubBuffer)
1475         {
1476             goto Exit;
1477         }

1479         /* Find comment terminator */

1481         Ptr1 = strstr (SubBuffer, "*/");
1482         if (!Ptr1)
1483         {
1484             goto Exit;
1485         }

1487         /* Find next EOL (from original buffer) */

1489         Ptr2 = strstr (SubBuffer, "\n");
1490         if (!Ptr2)
1491         {
1492             goto Exit;
1493         }

1495         /* Ignore one-line comments */

1497         if (Ptr1 < Ptr2)
1498         {
1499             /* Normal comment, ignore and continue; */

1501             SubBuffer = Ptr2;
1502             continue;
1503         }

1505         /* Examine multi-line comment */

1507         LineCount = 1;
1508         while (Ptr1 > Ptr2)
1509         {
1510             /* Find next EOL */

```

```

1512         Ptr2++;
1513         Ptr2 = strstr (Ptr2, "\n");
1514         if (!Ptr2)
1515         {
1516             goto Exit;
1517         }

1519         LineCount++;
1520     }

1522     SubBuffer = Ptr1;

1524     if (LineCount <= 3)
1525     {
1526         ShortCommentCount++;
1527     }
1528 }

1531 Exit:

1533     if (ShortCommentCount)
1534     {
1535         AsPrint ("Short Comments found", ShortCommentCount, Filename);
1536     }
1537 }
1538 #endif

```

new/usr/src/common/acpica/tools/acpiscrc/asfile.c

1

```
*****
22883 Fri Jan 17 07:31:18 2014
new/usr/src/common/acpica/tools/acpiscrc/asfile.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: asfile - Main module for the acpi source processor utility
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acpiscrc.h"

46 /* Local prototypes */

48 void
49 AsDoWildcard (
50     ACPI_CONVERSION_TABLE *ConversionTable,
51     char *SourcePath,
52     char *TargetPath,
53     int MaxPathLength,
54     int FileType,
55     char *WildcardSpec);

57 BOOLEAN
58 AsDetectLoneLineFeeds (
59     char *Filename,
```

new/usr/src/common/acpica/tools/acpiscrc/asfile.c

2

```
60     char *Buffer);

62 static ACPI_INLINE int
63 AsMaxInt (int a, int b)
64 {
65     return (a > b ? a : b);
66 }

69 /*****
70 *
71 * FUNCTION:    AsDoWildcard
72 *
73 * DESCRIPTION: Process files via wildcards
74 *
75 *****/

77 void
78 AsDoWildcard (
79     ACPI_CONVERSION_TABLE *ConversionTable,
80     char *SourcePath,
81     char *TargetPath,
82     int MaxPathLength,
83     int FileType,
84     char *WildcardSpec)
85 {
86     void *DirInfo;
87     char *Filename;
88     char *SourceDirPath;
89     char *TargetDirPath;
90     char RequestedFileType;

93     if (FileType == FILE_TYPE_DIRECTORY)
94     {
95         RequestedFileType = REQUEST_DIR_ONLY;
96     }
97     else
98     {
99         RequestedFileType = REQUEST_FILE_ONLY;
100     }

102     VERBOSE_PRINT (("Checking for %s source files in directory \"%s\"\n",
103         WildcardSpec, SourcePath));

105     /* Open the directory for wildcard search */

107     DirInfo = AcpiOsOpenDirectory (SourcePath, WildcardSpec, RequestedFileType);
108     if (DirInfo)
109     {
110         /*
111          * Get all of the files that match both the
112          * wildcard and the requested file type
113          */
114         while ((Filename = AcpiOsGetNextFilename (DirInfo)))
115         {
116             /* Looking for directory files, must check file type */

118             switch (RequestedFileType)
119             {
120                 case REQUEST_DIR_ONLY:

122                     /* If we actually have a dir, process the subtree */

124                     if (!AsCheckForDirectory (SourcePath, TargetPath, Filename,
125                         &SourceDirPath, &TargetDirPath))
```

```

126     {
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     {
244         UINT32 i = 1;
245         UINT32 LfCount = 0;
246         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     case FILE_TYPE_PATCH:

339         Functions = ConversionTable->PatchFunctions;
340         StringTable = ConversionTable->PatchStringTable;
341         LineTable = ConversionTable->PatchLineTable;
342         ConditionalTable = ConversionTable->PatchConditionalTable;
343         MacroTable = ConversionTable->PatchMacroTable;
344         StructTable = ConversionTable->PatchStructTable;
345         SpecialMacroTable = ConversionTable->PatchSpecialMacroTable;
346         break;

348     default:

350         printf ("Unknown file type, cannot process\n");
351         return;
352     }

355     Gbl_StructDefs = strstr (FileBuffer, "/* acpiscrc:StructDefs");
356     Gbl_Files++;
357     VERBOSE_PRINT (("Processing %u bytes\n",
358         (unsigned int) strlen (FileBuffer)));

360     if (Gbl_Cleanup)
361     {
362         AsRemoveExtraLines (FileBuffer, Filename);
363         AsRemoveSpacesAfterPeriod (FileBuffer, Filename);
364     }

366     if (ConversionTable->LowerCaseTable)
367     {
368         for (i = 0; ConversionTable->LowerCaseTable[i].Identifier; i++)
369         {
370             AsLowerCaseString (ConversionTable->LowerCaseTable[i].Identifier,
371                 FileBuffer);
372         }
373     }

375     /* Process all the string replacements */

377     if (StringTable)
378     {
379         for (i = 0; StringTable[i].Target; i++)
380         {
381             AsReplaceString (StringTable[i].Target, StringTable[i].Replacement,
382                 StringTable[i].Type, FileBuffer);
383         }
384     }

386     if (LineTable)
387     {
388         for (i = 0; LineTable[i].Identifier; i++)
389         {

```



```

390     AsRemoveLine (FileBuffer, LineTable[i].Identifier);
391   }
392 }

394 if (ConditionalTable)
395 {
396   for (i = 0; ConditionalTable[i].Identifier; i++)
397   {
398     AsRemoveConditionalCompile (FileBuffer, ConditionalTable[i].Identifi
399   }
400 }

402 if (MacroTable)
403 {
404   for (i = 0; MacroTable[i].Identifier; i++)
405   {
406     AsRemoveMacro (FileBuffer, MacroTable[i].Identifier);
407   }
408 }

410 if (StructTable)
411 {
412   for (i = 0; StructTable[i].Identifier; i++)
413   {
414     AsInsertPrefix (FileBuffer, StructTable[i].Identifier, StructTable[i
415   }
416 }

418 if (SpecialMacroTable)
419 {
420   for (i = 0; SpecialMacroTable[i].Identifier; i++)
421   {
422     AsCleanupSpecialMacro (FileBuffer, SpecialMacroTable[i].Identifier);
423   }
424 }

426 /* Process the function table */

428 for (i = 0; i < 32; i++)
429 {
430   /* Decode the function bitmap */

432   switch ((1 << i) & Functions)
433   {
434     case 0:

436       /* This function not configured */
437       break;

439     case CVT_COUNT_TABS:

441       AsCountTabs (FileBuffer, Filename);
442       break;

444     case CVT_COUNT_NON_ANSI_COMMENTS:

446       AsCountNonAnsiComments (FileBuffer, Filename);
447       break;

449     case CVT_CHECK_BRACES:

451       AsCheckForBraces (FileBuffer, Filename);
452       break;

454     case CVT_TRIM_LINES:

```

```

456     AsTrimLines (FileBuffer, Filename);
457     break;

459     case CVT_COUNT_LINES:

461       AsCountSourceLines (FileBuffer, Filename);
462       break;

464     case CVT_BRACES_ON_SAME_LINE:

466       AsBracesOnSameLine (FileBuffer);
467       break;

469     case CVT_MIXED_CASE_TO_UNDERSCORES:

471       AsMixedCaseToUnderscores (FileBuffer, Filename);
472       break;

474     case CVT_LOWER_CASE_IDENTIFIERS:

476       AsLowerCaseIdentifiers (FileBuffer);
477       break;

479     case CVT_REMOVE_DEBUG_MACROS:

481       AsRemoveDebugMacros (FileBuffer);
482       break;

484     case CVT_TRIM_WHITESPACE:

486       AsTrimWhitespace (FileBuffer);
487       break;

489     case CVT_REMOVE_EMPTY_BLOCKS:

491       AsRemoveEmptyBlocks (FileBuffer, Filename);
492       break;

494     case CVT_REDUCE_TYPEDEFS:

496       AsReduceTypedefs (FileBuffer, "typedef union");
497       AsReduceTypedefs (FileBuffer, "typedef struct");
498       break;

500     case CVT_SPACES_TO_TABS4:

502       AsTabify4 (FileBuffer);
503       break;

505     case CVT_SPACES_TO_TABS8:

507       AsTabify8 (FileBuffer);
508       break;

510     case CVT_COUNT_SHORTMULTILINE_COMMENTS:

512     #ifdef ACPI_FUTURE_IMPLEMENTATION
513       AsTrimComments (FileBuffer, Filename);
514     #endif
515     break;

517     default:

519       printf ("Unknown conversion subfunction opcode\n");
520       break;
521   }

```

```

522     }
523
524     if (ConversionTable->NewHeader)
525     {
526         AsReplaceHeader (FileBuffer, ConversionTable->NewHeader);
527     }
528 }
529
530
531 /*****
532  *
533  * FUNCTION:      AsProcessOneFile
534  *
535  * DESCRIPTION: Process one source file. The file is opened, read entirely
536  *              into a buffer, converted, then written to a new file.
537  *
538  *****/
539
540 ACPI_NATIVE_INT
541 AsProcessOneFile (
542     ACPI_CONVERSION_TABLE *ConversionTable,
543     char *SourcePath,
544     char *TargetPath,
545     int MaxPathLength,
546     char *Filename,
547     ACPI_NATIVE_INT FileType)
548 {
549     char *Pathname;
550     char *OutPathname = NULL;
551
552     /* Allocate a file pathname buffer for both source and target */
553
554     Pathname = calloc (MaxPathLength + strlen (Filename) + 2, 1);
555     if (!Pathname)
556     {
557         printf ("Could not allocate buffer for file pathnames\n");
558         return (-1);
559     }
560
561     Gbl_FileType = FileType;
562
563     /* Generate the source pathname and read the file */
564
565     if (SourcePath)
566     {
567         strcpy (Pathname, SourcePath);
568         strcat (Pathname, "/");
569     }
570
571     strcat (Pathname, Filename);
572
573     if (AsGetFile (Pathname, &Gbl_FileBuffer, &Gbl_FileSize))
574     {
575         return (-1);
576     }
577
578     Gbl_HeaderSize = 0;
579     if (strstr (Filename, ".asl"))
580     {
581         Gbl_HeaderSize = LINES_IN_ASX_HEADER; /* Lines in default ASL header */
582     }
583     else if (strstr (Gbl_FileBuffer, LEGAL_HEADER_SIGNATURE))
584     {
585         Gbl_HeaderSize = LINES_IN_LEGAL_HEADER; /* Normal C file and H header */
586     }
587 }

```

```

588     else if (strstr (Gbl_FileBuffer, LINUX_HEADER_SIGNATURE))
589     {
590         Gbl_HeaderSize = LINES_IN_LINUX_HEADER; /* Linuxized C file and H header
591     }
592
593     /* Process the file in the buffer */
594
595     Gbl_MadeChanges = FALSE;
596     if (!Gbl_IgnoreLoneLineFeeds && Gbl_HasLoneLineFeeds)
597     {
598         /*
599          * All lone LFs will be converted to CR/LF
600          * (when file is written, Windows version only)
601          */
602         printf ("Converting lone linefeeds\n");
603         Gbl_MadeChanges = TRUE;
604     }
605
606     AsConvertFile (ConversionTable, Gbl_FileBuffer, Pathname, FileType);
607
608     if (!(ConversionTable->Flags & FLG_NO_FILE_OUTPUT))
609     {
610         if (!(Gbl_Overwrite && !Gbl_MadeChanges))
611         {
612             /* Generate the target pathname and write the file */
613
614             OutPathname = calloc (MaxPathLength + strlen (Filename) + 2 + strlen
615             if (!OutPathname)
616             {
617                 printf ("Could not allocate buffer for file pathnames\n");
618                 return (-1);
619             }
620
621             strcpy (OutPathname, TargetPath);
622             if (SourcePath)
623             {
624                 strcat (OutPathname, "/");
625                 strcat (OutPathname, Filename);
626             }
627
628             AsPutFile (OutPathname, Gbl_FileBuffer, ConversionTable->Flags);
629         }
630     }
631
632     free (Gbl_FileBuffer);
633     free (Pathname);
634     if (OutPathname)
635     {
636         free (OutPathname);
637     }
638
639     return (0);
640 }
641
642
643 /*****
644  *
645  * FUNCTION:      AsCheckForDirectory
646  *
647  * DESCRIPTION: Check if the current file is a valid directory. If not,
648  *              construct the full pathname for the source and target paths.
649  *              Checks for the dot and dot-dot files (they are ignored)
650  *
651  *****/
652
653 ACPI_NATIVE_INT

```

```

654 AsCheckForDirectory (
655     char          *SourceDirPath,
656     char          *TargetDirPath,
657     char          *Filename,
658     char          **SourcePath,
659     char          **TargetPath)
660 {
661     char          *SrcPath;
662     char          *TgtPath;

665     if (!(strcmp (Filename, ".") ||
666         !(strcmp (Filename, ".."))))
667     {
668         return (-1);
669     }

671     SrcPath = calloc (strlen (SourceDirPath) + strlen (Filename) + 2, 1);
672     if (!SrcPath)
673     {
674         printf ("Could not allocate buffer for directory source pathname\n");
675         return (-1);
676     }

678     TgtPath = calloc (strlen (TargetDirPath) + strlen (Filename) + 2, 1);
679     if (!TgtPath)
680     {
681         printf ("Could not allocate buffer for directory target pathname\n");
682         free (SrcPath);
683         return (-1);
684     }

686     strcpy (SrcPath, SourceDirPath);
687     strcat (SrcPath, "/");
688     strcat (SrcPath, Filename);

690     strcpy (TgtPath, TargetDirPath);
691     strcat (TgtPath, "/");
692     strcat (TgtPath, Filename);

694     *SourcePath = SrcPath;
695     *TargetPath = TgtPath;
696     return (0);
697 }

700 /*****
701  *
702  * FUNCTION:    AsGetFile
703  *
704  * DESCRIPTION: Open a file and read it entirely into a an allocated buffer
705  *
706  *****/

708 int
709 AsGetFile (
710     char          *Filename,
711     char          **FileBuffer,
712     UINT32        *FileSize)
713 {
714     FILE          *File;
715     UINT32        Size;
716     char          *Buffer;
717     int           Seek1;
718     int           Seek2;
719     size_t        Actual;

```

```

722     /* Binary mode leaves CR/LF pairs */

724     File = fopen (Filename, "rb");
725     if (!File)
726     {
727         printf ("Could not open file %s\n", Filename);
728         return (-1);
729     }

731     /* Need file size to allocate a buffer */

733     Seek1 = fseek (File, 0L, SEEK_END);
734     Size = ftell (File);
735     Seek2 = fseek (File, 0L, SEEK_SET);

737     if (Seek1 || Seek2 || (Size == -1))
738     {
739         printf ("Could not get file size for %s\n", Filename);
740         goto ErrorExit;
741     }

743     /*
744      * Create a buffer for the entire file
745      * Add plenty extra buffer to accommodate string replacements
746      */
747     Gbl_TotalSize += Size;

749     Buffer = calloc (Size * 2, 1);
750     if (!Buffer)
751     {
752         printf ("Could not allocate buffer of size %u\n", Size * 2);
753         goto ErrorExit;
754     }

756     /* Read the entire file */

758     Actual = fread (Buffer, 1, Size, File);
759     if (Actual != Size)
760     {
761         printf ("Could not read the input file %s (%u bytes)\n",
762             Filename, Size);
763         goto ErrorExit;
764     }

766     Buffer [Size] = 0;          /* Null terminate the buffer */
767     fclose (File);

769     /* Check for unix contamination */

771     Gbl_HasLoneLineFeeds = AsDetectLoneLineFeeds (Filename, Buffer);

773     /*
774      * Convert all CR/LF pairs to LF only. We do this locally so that
775      * this code is portable across operating systems.
776      */
777     AsConvertToLineFeeds (Buffer);

779     *FileBuffer = Buffer;
780     *FileSize = Size;
781     return (0);

784 ErrorExit:

```

```
786     fclose (File);
787     return (-1);
788 }

791 /*****
792  *
793  * FUNCTION:    AsPutFile
794  *
795  * DESCRIPTION: Create a new output file and write the entire contents of the
796  *              buffer to the new file. Buffer must be a zero terminated string
797  *
798  *****/

800 int
801 AsPutFile (
802     char          *Pathname,
803     char          *FileBuffer,
804     UINT32        SystemFlags)
805 {
806     FILE          *File;
807     UINT32        FileSize;
808     size_t        Actual;
809     int           Status = 0;

812     /* Create the target file */

814     if (!(SystemFlags & FLG_NO_CARRIAGE_RETURNS))
815     {
816         /* Put back the CR before each LF */

818         AsInsertCarriageReturns (FileBuffer);
819     }

821     File = fopen (Pathname, "w+b");
822     if (!File)
823     {
824         perror ("Could not create destination file");
825         printf ("Could not create destination file \"%s\\n\"", Pathname);
826         return (-1);
827     }

829     /* Write the buffer to the file */

831     FileSize = strlen (FileBuffer);
832     Actual = fwrite (FileBuffer, 1, FileSize, File);
833     if (Actual != FileSize)
834     {
835         printf ("Error writing output file \"%s\\n\"", Pathname);
836         Status = -1;
837     }

839     fclose (File);
840     return (Status);
841 }
```

new/usr/src/common/acpica/tools/acpiscrc/asmmain.c

1

```
*****
13677 Fri Jan 17 07:31:19 2014
new/usr/src/common/acpica/tools/acpiscrc/asmmain.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: asmmain - Main module for the acpi source processor utility
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acpiscrc.h"
46 #include "acapps.h"

48 /* Local prototypes */

50 int
51 AsStricmp (
52     char          *String1,
53     char          *String2);

55 int
56 AsExaminePaths (
57     ACPI_CONVERSION_TABLE *ConversionTable,
58     char                  *Source,
59     char                  *Target,
```

new/usr/src/common/acpica/tools/acpiscrc/asmmain.c

2

```
60     UINT32          *SourceFileType);

62 void
63 AsDisplayStats (
64     void);

66 void
67 AsDisplayUsage (
68     void);

70 /* Globals */

72 UINT32          Gbl_Tabs = 0;
73 UINT32          Gbl_MissingBraces = 0;
74 UINT32          Gbl_NonAnsiComments = 0;
75 UINT32          Gbl_Files = 0;
76 UINT32          Gbl_WhiteLines = 0;
77 UINT32          Gbl_CommentLines = 0;
78 UINT32          Gbl_SourceLines = 0;
79 UINT32          Gbl_LongLines = 0;
80 UINT32          Gbl_TotalLines = 0;
81 UINT32          Gbl_TotalSize = 0;
82 UINT32          Gbl_HeaderLines = 0;
83 UINT32          Gbl_HeaderSize = 0;
84 void            *Gbl_StructDefs = NULL;

86 struct stat      Gbl_StatBuf;
87 char             *Gbl_FileBuffer;
88 UINT32          Gbl_FileSize;
89 UINT32          Gbl_FileType;
90 BOOLEAN          Gbl_VerboseMode = FALSE;
91 BOOLEAN          Gbl_QuietMode = FALSE;
92 BOOLEAN          Gbl_BatchMode = FALSE;
93 BOOLEAN          Gbl_DebugStatementsMode = FALSE;
94 BOOLEAN          Gbl_MadeChanges = FALSE;
95 BOOLEAN          Gbl_Overwrite = FALSE;
96 BOOLEAN          Gbl_WidenDeclarations = FALSE;
97 BOOLEAN          Gbl_IgnoreLoneLineFeeds = FALSE;
98 BOOLEAN          Gbl_HasLoneLineFeeds = FALSE;
99 BOOLEAN          Gbl_Cleanup = FALSE;
100 BOOLEAN          Gbl_IgnoreTranslationEscapes = FALSE;

102 #define AS_UTILITY_NAME          "ACPI Source Code Conversion Utility"
103 #define AS_SUPPORTED_OPTIONS     "cdhilgsuv^y"

106 /*****
107 *
108 * FUNCTION:      AsStricmp
109 *
110 * DESCRIPTION:   Implementation of the non-ANSI strcmp function (compare
111 * strings with no case sensitivity)
112 *
113 *****/

115 int
116 AsStricmp (
117     char          *String1,
118     char          *String2)
119 {
120     int          c1;
121     int          c2;

124     do
125     {
```

```

126     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 if (strstr (SourcePath, ".c"))
499             {
500                 AsProcessOneFile (ConversionTable, NULL, TargetPath, 0, SourcePath,
501             }
502             else if (strstr (SourcePath, ".patch"))
503             {
504                 AsProcessOneFile (ConversionTable, NULL, TargetPath, 0, SourcePath,
505             }
506             else
507             {
508                 printf ("Unknown file type - %s\n", SourcePath);
509             }
510         }

512     /* Always display final summary and stats */

514     AsDisplayStats ();

516     return (0);
517 }

```



new/usr/src/common/acpica/tools/acpiscrc/asremove.c

1

```
*****
19063 Fri Jan 17 07:31:19 2014
new/usr/src/common/acpica/tools/acpiscrc/asremove.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: asremove - Source conversion - removal functions
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "acpiscrc.h"
45
46 /* Local prototypes */
47
48 void
49 AsRemoveStatement (
50     char          *Buffer,
51     char          *Keyword,
52     UINT32        Type);
53
54
55 /*****
56 *
57 * FUNCTION:    AsRemoveStatement
58 *
59 *****/
```

new/usr/src/common/acpica/tools/acpiscrc/asremove.c

2

```
59 * DESCRIPTION: Remove all statements that contain the given keyword.
60 * Limitations: Removes text from the start of the line that
61 * contains the keyword to the next semicolon. Currently
62 * doesn't ignore comments.
63 *
64 *****/
65
66 void
67 AsRemoveStatement (
68     char          *Buffer,
69     char          *Keyword,
70     UINT32        Type)
71 {
72     char          *SubString;
73     char          *SubBuffer;
74     int           KeywordLength;
75
76
77     KeywordLength = strlen (Keyword);
78     SubBuffer = Buffer;
79     SubString = Buffer;
80
81
82     while (SubString)
83     {
84         SubString = strstr (SubBuffer, Keyword);
85
86         if (SubString)
87         {
88             SubBuffer = SubString;
89
90             if ((Type == REPLACE_WHOLE_WORD) &&
91                 (!AsMatchExactWord (SubString, KeywordLength)))
92             {
93                 SubBuffer++;
94                 continue;
95             }
96
97             /* Find start of this line */
98
99             while (*SubString != '\n')
100             {
101                 SubString--;
102             }
103             SubString++;
104
105             /* Find end of this statement */
106
107             SubBuffer = AsSkipPastChar (SubBuffer, ';');
108             if (!SubBuffer)
109             {
110                 return;
111             }
112
113             /* Find end of this line */
114
115             SubBuffer = AsSkipPastChar (SubBuffer, '\n');
116             if (!SubBuffer)
117             {
118                 return;
119             }
120
121             /* If next line is blank, remove it too */
122
123             if (*SubBuffer == '\n')
124             {
```

```

125         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 "#ifxxx" */

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 */
262
263         SubBuffer = AsSkipPastChar (ElsePtr, '\n');
264         if (!SubBuffer)
265         {
266             return;
267         }
268
269         /* Remove the #ifdef .... #else code */
270
271         AsRemoveData (SubString, SubBuffer);
272
273         /* Next, we will remove the #endif statement */
274
275         EndifPtr = strstr (SubString, "#endif");
276         if (!EndifPtr)
277         {
278             /* There has to be an #endif */
279
280             return;
281         }
282
283         SubString = EndifPtr;
284     }
285
286     /* Remove the ... #endif part */
287     /* Find end of this line */
288
289     SubBuffer = AsSkipPastChar (EndifPtr, '\n');
290     if (!SubBuffer)
291     {
292         return;
293     }
294
295     /* Remove the lines */
296
297     SubBuffer = AsRemoveData (SubString, SubBuffer);
298 }
299
300 /*****
301 *
302 * FUNCTION:    AsRemoveMacro
303 *
304 * DESCRIPTION: Remove every line that contains the keyword. Does not
305 *              skip comments.
306 *
307 *****/
308
309 void
310 AsRemoveMacro (
311     char          *Buffer,
312     char          *Keyword)
313 {
314     char          *SubString;
315     char          *SubBuffer;
316     int           NestLevel;
317
318     SubBuffer = Buffer;
319     SubString = Buffer;

```

```

320     while (SubString)
321     {
322         SubString = strstr (SubBuffer, Keyword);
323
324         if (SubString)
325         {
326             SubBuffer = SubString;
327
328             /* Find start of the macro parameters */
329
330             while (*SubString != '(')
331             {
332                 SubString++;
333             }
334             SubString++;
335
336             /* Remove the macro name and opening paren */
337
338             SubString = AsRemoveData (SubBuffer, SubString);
339
340             NestLevel = 1;
341             while (*SubString)
342             {
343                 if (*SubString == '(')
344                 {
345                     NestLevel++;
346                 }
347                 else if (*SubString == ')')
348                 {
349                     NestLevel--;
350                 }
351                 SubString++;
352
353                 if (NestLevel == 0)
354                 {
355                     break;
356                 }
357             }
358
359             /* Remove the closing paren */
360
361             SubBuffer = AsRemoveData (SubString-1, SubString);
362         }
363     }
364
365     SubBuffer = AsRemoveData (SubString-1, SubString);
366 }
367
368 /*****
369 *
370 * FUNCTION:    AsRemoveLine
371 *
372 * DESCRIPTION: Remove every line that contains the keyword. Does not
373 *              skip comments.
374 *
375 *****/
376
377 void
378 AsRemoveLine (
379     char          *Buffer,
380     char          *Keyword)
381 {
382     char          *SubString;
383     char          *SubBuffer;
384
385     SubBuffer = Buffer;
386     SubString = Buffer;

```

```

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 */
656
657         while (*SubString != '(')
658         {
659             SubString++;
660         }
661         SubString++;
662
663         NestLevel = 1;
664         while (*SubString)
665         {
666             if (*SubString == '(')
667             {
668                 NestLevel++;
669             }
670             else if (*SubString == ')')
671             {
672                 NestLevel--;
673             }
674
675             SubString++;
676
677             if (NestLevel == 0)
678             {
679                 break;
680             }
681         }
682
683     SkipLine:
684
685     /* Find end of the line */
686
687     NewLine = FALSE;
688     while (!NewLine && *SubString)
689     {
690         if (*SubString == '\n' && *(SubString - 1) != '\\')
691         {
692             NewLine = TRUE;
693         }
694         SubString++;
695     }
696
697     /* Find end of the line */
698
699     if (*SubString == '#' || *SubString == '\n')
700     {
701         goto SkipLine;
702     }
703
704     SubBuffer = SubString;
705
706     /* Find start of the non-space */
707
708     while (*SubString == ' ')
709     {
710         SubString++;
711     }
712
713     /* Find end of the line */
714
715     if (*SubString == '#' || *SubString == '\n')
716     {
717         goto SkipLine;
718     }

```

```

720     /* Find end of the line */
721
722     if (*SubString == '/' || *SubString == '*')
723     {
724         CommentEnd = strstr (SubString, "/*");
725         if (CommentEnd)
726         {
727             SubString = CommentEnd + 2;
728             goto SkipLine;
729         }
730     }
731
732     SubString = AsRemoveData (SubBuffer, SubString);
733 }
734 }
735 }

```

new/usr/src/common/acpica/tools/acpirc/astable.c

1

```
*****
47627 Fri Jan 17 07:31:19 2014
new/usr/src/common/acpica/tools/acpirc/astable.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: astable - Tables used for source conversion
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acpirc.h"
46 #include "acapps.h"

49 /*****
50 *
51 * Standard/Common translation tables
52 *
53 *****/

56 ACPI_STRING_TABLE      StandardDataTypes[] = {

58     /* Declarations first */
```

new/usr/src/common/acpica/tools/acpirc/astable.c

2

```
60     { "UINT32",      "unsigned int",    REPLACE_SUBSTRINGS},
61     { "UINT16",      "unsigned short",   REPLACE_SUBSTRINGS},
62     { "UINT8",       "unsigned char",    REPLACE_SUBSTRINGS},
63     { "BOOLEAN",     "unsigned char",    REPLACE_SUBSTRINGS},

65     /* Now do embedded typecasts */

67     { "UINT32",      "unsigned int",    REPLACE_SUBSTRINGS},
68     { "UINT16",      "unsigned short",   REPLACE_SUBSTRINGS},
69     { "UINT8",       "unsigned char",    REPLACE_SUBSTRINGS},
70     { "BOOLEAN",     "unsigned char",    REPLACE_SUBSTRINGS},

72     { "INT32",       "int",             REPLACE_SUBSTRINGS},
73     { "INT32",       "int",             REPLACE_SUBSTRINGS},
74     { "INT16",       "short",           REPLACE_SUBSTRINGS},
75     { "INT8",        "char",            REPLACE_SUBSTRINGS},

77     /* Put back anything we broke (such as anything with _INT32_ in it) */

79     { "_int_",       "_INT32_",         REPLACE_SUBSTRINGS},
80     { "unsigned int_", "UINT32_",       REPLACE_SUBSTRINGS},
81     { NULL,          NULL,              0},
82 };

85 /*****
86 *
87 * Linux-specific translation tables
88 *
89 *****/

91 char      DualLicenseHeader[] =
92 "/*\n"
93 " * Copyright (C) 2000 - 2014, Intel Corp.\n"
94 " * All rights reserved.\n"
95 " *\n"
96 " * Redistribution and use in source and binary forms, with or without\n"
97 " * modification, are permitted provided that the following conditions\n"
98 " * are met:\n"
99 " * 1. Redistributions of source code must retain the above copyright\n"
100 " * notice, this list of conditions, and the following disclaimer,\n"
101 " * without modification.\n"
102 " * 2. Redistributions in binary form must reproduce at minimum a disclaimer\n"
103 " * substantially similar to the \"NO WARRANTY\" disclaimer below\n"
104 " * (\"Disclaimer\") and any redistribution must be conditioned upon\n"
105 " * including a substantially similar Disclaimer requirement for further\n"
106 " * binary redistribution.\n"
107 " * 3. Neither the names of the above-listed copyright holders nor the names\n"
108 " * of any contributors may be used to endorse or promote products derived\n"
109 " * from this software without specific prior written permission.\n"
110 " *\n"
111 " * Alternatively, this software may be distributed under the terms of the\n"
112 " * GNU General Public License (\"GPL\") version 2 as published by the Free\n"
113 " * Software Foundation.\n"
114 " *\n"
115 " * NO WARRANTY\n"
116 " * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS\n"
117 " * \"AS IS\" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT\n"
118 " * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR\n"
119 " * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT\n"
120 " * HOLDERS OR CONTRIBUTORS BE LIABLE FOR SPECIAL, EXEMPLARY, OR CONSEQUENTIAL\n"
121 " * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS\n"
122 " * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)\n"
123 " * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,\n"
124 " * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING\n"
125 " * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE\n"
```

```

126 " * POSSIBILITY OF SUCH DAMAGES.\n"
127 " */\n";

129 ACPI_STRING_TABLE      LinuxDataTypes[] = {

131 /*
132  * Extra space is added after the type so there is room to add "struct", "union"
133  * etc. when the existing struct typedefs are eliminated.
134  */

136 /* Declarations first - ACPI types and standard C types */

138 {"INT64", "s64", "REPLACE_WHOLE_WORD", EXTRA_I
139 {"UINT64", "u64", "REPLACE_WHOLE_WORD", EXTRA_I
140 {"UINT32", "u32", "REPLACE_WHOLE_WORD", EXTRA_I
141 {"INT32", "s32", "REPLACE_WHOLE_WORD", EXTRA_I
142 {"UINT16", "u16", "REPLACE_WHOLE_WORD", EXTRA_I
143 {"INT16", "s16", "REPLACE_WHOLE_WORD", EXTRA_I
144 {"UINT8", "u8", "REPLACE_WHOLE_WORD", EXTRA_I
145 {"BOOLEAN", "u8", "REPLACE_WHOLE_WORD", EXTRA_I
146 {"char", "char", "REPLACE_WHOLE_WORD", EXTRA_I
147 {"void", "void", "REPLACE_WHOLE_WORD", EXTRA_I
148 {"char *", "char *", "REPLACE_WHOLE_WORD", EXTRA_I
149 {"void *", "void *", "REPLACE_WHOLE_WORD", EXTRA_I
150 {"int", "int", "REPLACE_WHOLE_WORD", EXTRA_I
151 {"FILE", "FILE", "REPLACE_WHOLE_WORD", EXTRA_I
152 {"size_t", "size_t", "REPLACE_WHOLE_WORD", EXTRA_I

154 /* Now do embedded typecasts */

156 {"UINT64", "u64", "REPLACE_WHOLE_WORD",
157 {"UINT32", "u32", "REPLACE_WHOLE_WORD",
158 {"UINT16", "u16", "REPLACE_WHOLE_WORD",
159 {"UINT8", "u8", "REPLACE_WHOLE_WORD",
160 {"BOOLEAN", "u8", "REPLACE_WHOLE_WORD",

162 {"INT64", "s64", "REPLACE_WHOLE_WORD",
163 {"INT64", "s64", "REPLACE_WHOLE_WORD",
164 {"INT32", "s32", "REPLACE_WHOLE_WORD",
165 {"INT32", "s32", "REPLACE_WHOLE_WORD",
166 {"INT16", "s16", "REPLACE_WHOLE_WORD",
167 {"INT8", "s8", "REPLACE_WHOLE_WORD",
168 {"INT16", "s16", "REPLACE_WHOLE_WORD",
169 {"INT8", "s8", "REPLACE_WHOLE_WORD",

171 {NULL, NULL, 0}
172 };

174 ACPI_TYPED_IDENTIFIER_TABLE      AcpiIdentifiers[] = {

176 {"ACPI_ADDRESS_RANGE", SRC_TYPE_STRUCT,
177 {"ACPI_ADR_SPACE_HANDLER", SRC_TYPE_SIMPLE,
178 {"ACPI_ADR_SPACE_SETUP", SRC_TYPE_SIMPLE,
179 {"ACPI_ADR_SPACE_TYPE", SRC_TYPE_SIMPLE,
180 {"ACPI_AML_OPERANDS", SRC_TYPE_UNION,
181 {"ACPI_BIT_REGISTER_INFO", SRC_TYPE_STRUCT,
182 {"ACPI_BUFFER", SRC_TYPE_STRUCT,
183 {"ACPI_BUS_ATTRIBUTE", SRC_TYPE_STRUCT,
184 {"ACPI_CACHE_T", SRC_TYPE_SIMPLE,
185 {"ACPI_CMTHANDLER", SRC_TYPE_SIMPLE,
186 {"ACPI_COMMON_FACS", SRC_TYPE_STRUCT,
187 {"ACPI_COMMON_STATE", SRC_TYPE_STRUCT,
188 {"ACPI_COMMON_DESCRIPTOR", SRC_TYPE_STRUCT,
189 {"ACPI_COMPATIBLE_ID", SRC_TYPE_STRUCT,
190 {"ACPI_CONNECTION_INFO", SRC_TYPE_STRUCT,
191 {"ACPI_CONTROL_STATE", SRC_TYPE_STRUCT,

```

```

192 {"ACPI_CONVERSION_TABLE", SRC_TYPE_STRUCT,
193 {"ACPI_CPU_FLAGS", SRC_TYPE_SIMPLE,
194 {"ACPI_CREATE_FIELD_INFO", SRC_TYPE_STRUCT,
195 {"ACPI_DB_ARGUMENT_INFO", SRC_TYPE_STRUCT,
196 {"ACPI_DB_COMMAND_HELP", SRC_TYPE_STRUCT,
197 {"ACPI_DB_COMMAND_INFO", SRC_TYPE_STRUCT,
198 {"ACPI_DB_EXECUTE_WALK", SRC_TYPE_STRUCT,
199 {"ACPI_DB_METHOD_INFO", SRC_TYPE_STRUCT,
200 {"ACPI_DEBUG_MEM_BLOCK", SRC_TYPE_STRUCT,
201 {"ACPI_DEBUG_MEM_HEADER", SRC_TYPE_STRUCT,
202 {"ACPI_DEBUG_PRINT_INFO", SRC_TYPE_STRUCT,
203 {"ACPI_DESCRIPTOR", SRC_TYPE_UNION,
204 {"ACPI_DEVICE_INFO", SRC_TYPE_STRUCT,
205 {"ACPI_DEVICE_WALK_INFO", SRC_TYPE_STRUCT,
206 {"ACPI_DMTHANDLER", SRC_TYPE_STRUCT,
207 {"ACPI_DMTHANDLER_HANDLER", SRC_TYPE_SIMPLE,
208 {"ACPI_EVALUATE_INFO", SRC_TYPE_STRUCT,
209 {"ACPI_EVENT_HANDLER", SRC_TYPE_SIMPLE,
210 {"ACPI_EVENT_STATUS", SRC_TYPE_SIMPLE,
211 {"ACPI_EVENT_TYPE", SRC_TYPE_SIMPLE,
212 {"ACPI_EXCEPTION_HANDLER", SRC_TYPE_SIMPLE,
213 {"ACPI_EXCEPTION_INFO", SRC_TYPE_STRUCT,
214 {"ACPI_EXDUMP_INFO", SRC_TYPE_STRUCT,
215 {"ACPI_EXECUTE_OP", SRC_TYPE_SIMPLE,
216 {"ACPI_EXECUTE_TYPE", SRC_TYPE_SIMPLE,
217 {"ACPI_EXTERNAL_LIST", SRC_TYPE_STRUCT,
218 {"ACPI_EXTERNAL_FILE", SRC_TYPE_STRUCT,
219 {"ACPI_FADT_INFO", SRC_TYPE_STRUCT,
220 {"ACPI_FADT_PM_INFO", SRC_TYPE_STRUCT,
221 {"ACPI_FIELD_INFO", SRC_TYPE_STRUCT,
222 {"ACPI_FIND_CONTEXT", SRC_TYPE_STRUCT,
223 {"ACPI_FIXED_EVENT_HANDLER", SRC_TYPE_STRUCT,
224 {"ACPI_FIXED_EVENT_INFO", SRC_TYPE_STRUCT,
225 {"ACPI_GBL_EVENT_HANDLER", SRC_TYPE_SIMPLE,
226 {"ACPI_GENERIC_ADDRESS", SRC_TYPE_STRUCT,
227 {"ACPI_GENERIC_STATE", SRC_TYPE_UNION,
228 {"ACPI_GET_DEVICES_INFO", SRC_TYPE_STRUCT,
229 {"ACPI_GLOBAL_NOTIFY_HANDLER", SRC_TYPE_STRUCT,
230 {"ACPI_GPE_BLOCK_INFO", SRC_TYPE_STRUCT,
231 {"ACPI_GPE_CALLBACK", SRC_TYPE_SIMPLE,
232 {"ACPI_GPE_DEVICE_INFO", SRC_TYPE_STRUCT,
233 {"ACPI_GPE_EVENT_INFO", SRC_TYPE_STRUCT,
234 {"ACPI_GPE_HANDLER", SRC_TYPE_SIMPLE,
235 {"ACPI_GPE_HANDLER_INFO", SRC_TYPE_STRUCT,
236 {"ACPI_GPE_INDEX_INFO", SRC_TYPE_STRUCT,
237 {"ACPI_GPE_NOTIFY_INFO", SRC_TYPE_STRUCT,
238 {"ACPI_GPE_REGISTER_INFO", SRC_TYPE_STRUCT,
239 {"ACPI_GPE_WALK_INFO", SRC_TYPE_STRUCT,
240 {"ACPI_GPE_XRUPT_INFO", SRC_TYPE_STRUCT,
241 {"ACPI_HANDLE", SRC_TYPE_SIMPLE,
242 {"ACPI_HANDLER_INFO", SRC_TYPE_STRUCT,
243 {"ACPI_INIT_HANDLER", SRC_TYPE_SIMPLE,
244 {"ACPI_INTERFACE_HANDLER", SRC_TYPE_SIMPLE,
245 {"ACPI_IDENTIFIER_TABLE", SRC_TYPE_STRUCT,
246 {"ACPI_INIT_WALK_INFO", SRC_TYPE_STRUCT,
247 {"ACPI_INTEGER", SRC_TYPE_SIMPLE,
248 {"ACPI_INTEGER_OVERLAY", SRC_TYPE_STRUCT,
249 {"ACPI_INTEGRITY_INFO", SRC_TYPE_STRUCT,
250 {"ACPI_INTERFACE_INFO", SRC_TYPE_STRUCT,
251 {"ACPI_INTERNAL_METHOD", SRC_TYPE_SIMPLE,
252 {"ACPI_INTERPRETER_MODE", SRC_TYPE_SIMPLE,
253 {"ACPI_IO_ADDRESS", SRC_TYPE_SIMPLE,
254 {"ACPI_IO_ATTRIBUTE", SRC_TYPE_STRUCT,
255 {"ACPI_MEM_SPACE_CONTEXT", SRC_TYPE_STRUCT,
256 {"ACPI_MEMORY_ATTRIBUTE", SRC_TYPE_STRUCT,

```



```

258 {"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_RS_CONVERT_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_MTM", 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_FPD", 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_FPDТ_HEADER", SRC_TYPE_STRUCT},
557 {"ACPI_FPDТ_BOOT", SRC_TYPE_STRUCT},
558 {"ACPI_FPDТ_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_MADТ_ADDRESS_OVERRIDE", SRC_TYPE_STRUCT},

```

```

588 {"ACPI_MADТ_HEADER", SRC_TYPE_STRUCT},
589 {"ACPI_MADТ_IO_APIC", SRC_TYPE_STRUCT},
590 {"ACPI_MADТ_IO_SAPIC", SRC_TYPE_STRUCT},
591 {"ACPI_MADТ_LOCAL_APIC", SRC_TYPE_STRUCT},
592 {"ACPI_MADТ_LOCAL_APIC_NMI", SRC_TYPE_STRUCT},
593 {"ACPI_MADТ_LOCAL_APIC_OVERRIDE", SRC_TYPE_STRUCT},
594 {"ACPI_MADТ_LOCAL_SAPIC", SRC_TYPE_STRUCT},
595 {"ACPI_MADТ_LOCAL_X2APIC", SRC_TYPE_STRUCT},
596 {"ACPI_MADТ_LOCAL_X2APIC_NMI", SRC_TYPE_STRUCT},
597 {"ACPI_MADТ_GENERIC_DISTRIBUTOR", SRC_TYPE_STRUCT},
598 {"ACPI_MADТ_GENERIC_INTERRUPT", SRC_TYPE_STRUCT},
599 {"ACPI_MADТ_INTERRUPT_OVERRIDE", SRC_TYPE_STRUCT},
600 {"ACPI_MADТ_INTERRUPT_SOURCE", SRC_TYPE_STRUCT},
601 {"ACPI_MADТ_NMI_SOURCE", SRC_TYPE_STRUCT},
602 {"ACPI_MADТ_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_TYDEFS | CVT_SPACES_TO_TABS8),

765     /* Patch files */

767     LinuxDataTypes,
768     NULL,
769     NULL,
770     NULL,
771     AcpiIdentifiers,
772     NULL,
773     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
774     CVT_MIXED_CASE_TO_UNDERSCORES),
775 };

778 /*****
779  *
780  * Code cleanup translation tables
781  *
782  *****/

784 ACPI_CONVERSION_TABLE    CleanupConversionTable = {

```

```

786     NULL,
787     FLG_DEFAULT_FLAGS,
788     NULL,
789     /* C source 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_CHECK_BRACES | CVT_TRIM_LINES | CVT_TRIM_WHITESPACE),

800     /* C header files */

802     NULL,
803     NULL,
804     NULL,
805     NULL,
806     NULL,
807     NULL,
808     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
809      CVT_TRIM_LINES | CVT_TRIM_WHITESPACE),

811     /* Patch files */

813     NULL,
814     NULL,
815     NULL,
816     NULL,
817     NULL,
818     NULL,
819     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES),
820 };

823 ACPI_CONVERSION_TABLE      StatsConversionTable = {

825     NULL,
826     FLG_NO_FILE_OUTPUT,
827     NULL,

829     /* C source files */

831     NULL,
832     NULL,
833     NULL,
834     NULL,
835     NULL,
836     NULL,
837     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
838      CVT_COUNT_SHORTMULTILINE_COMMENTS),

840     /* C header files */

842     NULL,
843     NULL,
844     NULL,
845     NULL,
846     NULL,
847     NULL,
848     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
849      CVT_COUNT_SHORTMULTILINE_COMMENTS),

851     /* Patch files */

```

```

853     NULL,
854     NULL,
855     NULL,
856     NULL,
857     NULL,
858     NULL,
859     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
860      CVT_COUNT_SHORTMULTILINE_COMMENTS),
861 };

864 /*****
865  *
866  * Dual License injection translation table
867  *
868  *****/

870 ACPI_CONVERSION_TABLE      LicenseConversionTable = {

872     DualLicenseHeader,
873     FLG_DEFAULT_FLAGS,
874     NULL,

876     /* C source files */

878     NULL,
879     NULL,
880     NULL,
881     NULL,
882     NULL,
883     NULL,
884     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
885      CVT_COUNT_SHORTMULTILINE_COMMENTS),

887     /* C header files */

889     NULL,
890     NULL,
891     NULL,
892     NULL,
893     NULL,
894     NULL,
895     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
896      CVT_COUNT_SHORTMULTILINE_COMMENTS),

898     /* Patch files */

900     NULL,
901     NULL,
902     NULL,
903     NULL,
904     NULL,
905     NULL,
906     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
907      CVT_COUNT_SHORTMULTILINE_COMMENTS),
908 };

911 /*****
912  *
913  * Customizable translation tables
914  *
915  *****/

917 ACPI_STRING_TABLE          CustomReplacements[] = {

```

```

920  {"(c) 1999 - 2013",      "(c) 1999 - 2014",      REPLACE_WHOLE_WORD}, /* M
921  {"(c) 2006 - 2013",      "(c) 2006 - 2014",      REPLACE_WHOLE_WORD}, /* T

923 #if 0
924  {"SUPPORT, ASSISTANCE", "SUPPORT, ASSISTANCE",      REPLACE_WHOLE_WORD}, /* F

926  {"(ACPI_INTEGER)", "(UINT64)",      REPLACE_WHOLE_WORD},
927  {"ACPI_INTEGER",      "", "UINT64",      REPLACE_WHOLE_WORD},
928  {"ACPI_INTEGER", "UINT64",      REPLACE_WHOLE_WORD},
929  {"ACPI_INTEGER_MAX", "ACPI_UINT64_MAX",      REPLACE_WHOLE_WORD},
930  {"#include \"acpi.h\"",      "#include \"acpi.h\"\\n#include \"accommon.h\"", R
931  {"AcpiTbSumTable", "AcpiTbSumTable",      REPLACE_WHOLE_WORD},
932  {"ACPI_SIG_BOOT", "ACPI_SIG_BOOT",      REPLACE_WHOLE_WORD},
933  {"ACPI_SIG_DBGP", "ACPI_SIG_DBGP",      REPLACE_WHOLE_WORD},
934  {"ACPI_SIG_DSDT", "ACPI_SIG_DSDT",      REPLACE_WHOLE_WORD},
935  {"ACPI_SIG_ECDT", "ACPI_SIG_ECDT",      REPLACE_WHOLE_WORD},
936  {"ACPI_SIG_FACS", "ACPI_SIG_FACS",      REPLACE_WHOLE_WORD},
937  {"ACPI_SIG_FADT", "ACPI_SIG_FADT",      REPLACE_WHOLE_WORD},
938  {"ACPI_SIG_HPET", "ACPI_SIG_HPET",      REPLACE_WHOLE_WORD},
939  {"ACPI_SIG_MADT", "ACPI_SIG_MADT",      REPLACE_WHOLE_WORD},
940  {"ACPI_SIG_MCFG", "ACPI_SIG_MCFG",      REPLACE_WHOLE_WORD},
941  {"ACPI_SIG_PSDT", "ACPI_SIG_PSDT",      REPLACE_WHOLE_WORD},
942  {"ACPI_NAME_RSDP", "ACPI_NAME_RSDP",      REPLACE_WHOLE_WORD},
943  {"ACPI_SIG_RSDP", "ACPI_SIG_RSDP",      REPLACE_WHOLE_WORD},
944  {"ACPI_SIG_RSDT", "ACPI_SIG_RSDT",      REPLACE_WHOLE_WORD},
945  {"ACPI_SIG_SBST", "ACPI_SIG_SBST",      REPLACE_WHOLE_WORD},
946  {"ACPI_SIG_SLIT", "ACPI_SIG_SLIT",      REPLACE_WHOLE_WORD},
947  {"ACPI_SIG_SPCR", "ACPI_SIG_SPCR",      REPLACE_WHOLE_WORD},
948  {"ACPI_SIG_SPIC", "ACPI_SIG_SPIC",      REPLACE_WHOLE_WORD},
949  {"ACPI_SIG_SPMI", "ACPI_SIG_SPMI",      REPLACE_WHOLE_WORD},
950  {"ACPI_SIG_SRAT", "ACPI_SIG_SRAT",      REPLACE_WHOLE_WORD},
951  {"ACPI_SIG_SSDT", "ACPI_SIG_SSDT",      REPLACE_WHOLE_WORD},
952  {"ACPI_SIG_TCPA", "ACPI_SIG_TCPA",      REPLACE_WHOLE_WORD},
953  {"ACPI_SIG_WDRT", "ACPI_SIG_WDRT",      REPLACE_WHOLE_WORD},
954  {"ACPI_SIG_XSDT", "ACPI_SIG_XSDT",      REPLACE_WHOLE_WORD},

956  {"ACPI_ALLOCATE_ZEROED", "ACPI_ALLOCATE_ZEROED", REPLACE_WHOLE_WORD},
957  {"ACPI_ALLOCATE", "ACPI_ALLOCATE",      REPLACE_WHOLE_WORD},
958  {"ACPI_FREE", "ACPI_FREE",      REPLACE_WHOLE_WORD},

960  "ACPI_NATIVE_UINT", "ACPI_NATIVE_UINT",      REPLACE_WHOLE_WORD,
961  "ACPI_NATIVE_UINT **", "ACPI_NATIVE_UINT **", REPLACE_WHOLE_WORD,
962  "ACPI_NATIVE_UINT", "ACPI_NATIVE_UINT",      REPLACE_WHOLE_WORD,
963  "ACPI_NATIVE_INT", "ACPI_NATIVE_INT",      REPLACE_WHOLE_WORD,
964  "ACPI_NATIVE_INT **", "ACPI_NATIVE_INT **", REPLACE_WHOLE_WORD,
965  "ACPI_NATIVE_INT", "ACPI_NATIVE_INT",      REPLACE_WHOLE_WORD,
966 #endif

968  {NULL,      NULL, 0}
969 };

972 ACPI_CONVERSION_TABLE      CustomConversionTable = {

974  NULL,
975  FLG_DEFAULT_FLAGS,
976  NULL,

978  /* C source files */

980  CustomReplacements,
981  LinuxEliminateLines_H,
982  NULL,
983  NULL,

```

```

984  NULL,
985  NULL,
986  (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
987  CVT_TRIM_LINES | CVT_TRIM_WHITESPACE),

989  /* C header files */

991  CustomReplacements,
992  LinuxEliminateLines_H,
993  NULL,
994  NULL,
995  NULL,
996  NULL,
997  (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
998  CVT_TRIM_LINES | CVT_TRIM_WHITESPACE),

1000  /* C header files */

1002  NULL,
1003  NULL,
1004  NULL,
1005  NULL,
1006  NULL,
1007  NULL,
1008  (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES),
1009 };

1012  /*****
1013  *
1014  * Indentation result fixup table
1015  *
1016  *****/

1018  ACPI_CONVERSION_TABLE      IndentConversionTable = {

1020  NULL,
1021  FLG_NO_CARRIAGE_RETURNS,

1023  NULL,

1025  /* C source files */

1027  LinuxSpecialStrings,
1028  NULL,
1029  NULL,
1030  NULL,
1031  NULL,
1032  LinuxSpecialMacros,
1033  (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
1034  CVT_TRIM_LINES | CVT_TRIM_WHITESPACE),

1036  /* C header files */

1038  LinuxSpecialStrings,
1039  NULL,
1040  NULL,
1041  NULL,
1042  NULL,
1043  LinuxSpecialMacros,
1044  (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES |
1045  CVT_TRIM_LINES | CVT_TRIM_WHITESPACE),

1047  /* C header files */

1049  NULL,

```

```
1050     NULL,  
1051     NULL,  
1052     NULL,  
1053     NULL,  
1054     NULL,  
1055     (CVT_COUNT_TABS | CVT_COUNT_NON_ANSI_COMMENTS | CVT_COUNT_LINES),  
1056 };
```

new/usr/src/common/acpica/tools/acpisc/asutils.c

1

```
*****
7817 Fri Jan 17 07:31:19 2014
new/usr/src/common/acpica/tools/acpisc/asutils.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: asutils - common utilities
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acpisc.h"

47 /*****
48 *
49 * FUNCTION: AsStrlwr (strlwr)
50 *
51 * PARAMETERS: SrcString - The source string to convert
52 *
53 * RETURN: None
54 *
55 * DESCRIPTION: Convert string to lowercase
56 *
57 * NOTE: This is not a POSIX function, so it appears here so that we don't have
58 * header file issues with the various hosts/compilers/clibs.
59 *
60 *****/
```

new/usr/src/common/acpica/tools/acpisc/asutils.c

2

```
62 void
63 AsStrlwr (
64     char          *SrcString)
65 {
66     char          *String;

69     /* Walk entire string, lowercasing the letters */

71     if (SrcString)
72     {
73         for (String = SrcString; *String; String++)
74         {
75             *String = (char) ACPI_TOLOWER (*String);
76         }
77     }
78 }

81 /*****
82 *
83 * FUNCTION: AsSkipUntilChar
84 *
85 * DESCRIPTION: Find the next instance of the input character
86 *
87 *****/

89 char *
90 AsSkipUntilChar (
91     char          *Buffer,
92     char          Target)
93 {
94
95     while (*Buffer != Target)
96     {
97         if (!*Buffer)
98         {
99             return (NULL);
100         }
101
102         Buffer++;
103     }

105     return (Buffer);
106 }

109 /*****
110 *
111 * FUNCTION: AsSkipPastChar
112 *
113 * DESCRIPTION: Find the next instance of the input character, return a buffer
114 * pointer to this character+1.
115 *
116 *****/

118 char *
119 AsSkipPastChar (
120     char          *Buffer,
121     char          Target)
122 {
123
124     while (*Buffer != Target)
125     {
126         if (!*Buffer)
```



```

127     {
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

```

new/usr/src/common/acpica/tools/acpisc/asutils.c

5

```
259     */
260     BufferLength = strlen (EndPoint) + 1;

262     Gbl_MadeChanges = TRUE;
263     memmove (StartPointer, EndPointer, BufferLength);

265     return (StartPointer);
266 }
```

new/usr/src/common/acpica/tools/acpixtract/acpixtract.c

1

```
*****
21590 Fri Jan 17 07:31:19 2014
new/usr/src/common/acpica/tools/acpixtract/acpixtract.c
update to acpica-unix2-20140114
update to acpica-unix2-20130927
acpica-unix2-20130823
PANKOV's restructure
*****
1 /*****
2 *
3 * Module Name: acpixtract - convert ascii ACPI tables to binary
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acapps.h"

48 #include <stdio.h>
49 #include <stdlib.h>
50 #include <string.h>
51 #include <ctype.h>

53 /* Local prototypes */

55 static void
56 AxStrlwr (
57     char                *String);
```

new/usr/src/common/acpica/tools/acpixtract/acpixtract.c

2

```
59 static void
60 AxCheckAscii (
61     char                *Name,
62     int                 Count);

64 static void
65 AxNormalizeSignature (
66     char                *Signature);

68 static unsigned int
69 AxGetInstance (
70     char                *InputPathname,
71     char                *Signature);

73 static size_t
74 AxGetTableHeader (
75     FILE                *InputFile,
76     unsigned char       *OutputData);

78 static unsigned int
79 AxCountTableInstances (
80     char                *InputPathname,
81     char                *Signature);

83 int
84 AxExtractTables (
85     char                *InputPathname,
86     char                *Signature,
87     unsigned int        MinimumInstances);

89 int
90 AxListTables (
91     char                *InputPathname);

93 static size_t
94 AxConvertLine (
95     char                *InputLine,
96     unsigned char       *OutputData);

98 static int
99 AxIsEmptyLine (
100    char                *Buffer);

102 typedef struct AxTableInfo
103 {
104     UINT32              Signature;
105     unsigned int        Instances;
106     unsigned int        NextInstance;
107     struct AxTableInfo *Next;
108 } AX_TABLE_INFO;

109 } AX_TABLE_INFO;

111 /* Extraction states */

113 #define AX_STATE_FIND_HEADER      0
114 #define AX_STATE_EXTRACT_DATA    1

116 /* Miscellaneous constants */

118 #define AX_LINE_BUFFER_SIZE      256
119 #define AX_MIN_TABLE_NAME_LENGTH 6 /* strlen ("DSDT @") */

122 static AX_TABLE_INFO *AxTableListHead = NULL;
123 static char Filename[16];
124 static unsigned char Data[16];
```

```

125 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:      AxCtrlwr
133  *
134  * PARAMETERS:    String          - Ascii string
135  *
136  * RETURN:        None
137  *
138  * DESCRIPTION:    String lowercase function.
139  *
140  *****/

142 static void
143 AxCtrlwr (
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 (!strcmp (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;
268
269     /* Terminate the input line at the end of the actual data (for sscanf) */
270
271     End = strstr (InputLine + 2, " ");
272     if (!End)
273     {
274         return (0); /* Don't understand the format */
275     }
276     *End = 0;
277
278     /*
279      * Convert one line of table data, of the form:
280      * <offset>: <up to 16 bytes of hex data> <ASCII representation> <newline>
281      *
282      * Example:
283      * 02C0: 5F 53 42 5F 4C 4E 4B 44 00 12 13 04 0C FF FF 08 _SB_LNKD.....
284      */
285     BytesConverted = sscanf (InputLine,
286         "%s %x %x %x %x %x %x %x %x %x %x %x %x %x %x",
287         &Converted[0], &Converted[1], &Converted[2], &Converted[3],
288         &Converted[4], &Converted[5], &Converted[6], &Converted[7],
289         &Converted[8], &Converted[9], &Converted[10], &Converted[11],
290         &Converted[12], &Converted[13], &Converted[14], &Converted[15]);
291
292     /* Pack converted data into a byte array */
293
294     for (i = 0; i < BytesConverted; i++)
295     {
296         OutputData[i] = (unsigned char) Converted[i];
297     }
298
299     return ((size_t) BytesConverted);
300 }
301
302 *****/
303
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  *****/
316
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;
325
326     /* Get the full 36 byte ACPI table header, requires 3 input text lines */
327
328     for (i = 0; i < 3; i++)
329     {
330         if (!fgets (HeaderBuffer, AX_LINE_BUFFER_SIZE, InputFile))
331         {
332             return (TotalConverted);
333         }
334
335         BytesConverted = AxConvertLine (HeaderBuffer, OutputData);
336         TotalConverted += BytesConverted;
337         OutputData += 16;
338
339         if (BytesConverted != 16)
340         {
341             return (TotalConverted);
342         }
343     }
344
345     return (TotalConverted);
346 }
347
348 *****/
349
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  *****/
363
364 static unsigned int
365 AxCountTableInstances (
366     char          *InputPathname,
367     char          *Signature)
368 {
369     FILE          *InputFile;
370     unsigned int  Instances = 0;
371
372     InputFile = fopen (InputPathname, "rt");
373     if (!InputFile)
374     {
375         printf ("Could not open file %s\n", InputPathname);
376         return (0);
377     }
378
379     /* Count the number of instances of this signature */
380
381     while (fgets (InstanceBuffer, AX_LINE_BUFFER_SIZE, InputFile))
382     {
383         /* Ignore empty lines and lines that start with a space */
384
385         if (AxIsEmptyLine (InstanceBuffer) ||
386             (InstanceBuffer[0] == ' '))
387         {
388

```

```

389     continue;
390 }

392     AxNormalizeSignature (InstanceBuffer);
393     if (ACPI_COMPARE_NAME (InstanceBuffer, Signature))
394     {
395         Instances++;
396     }
397 }

399     fclose (InputFile);
400     return (Instances);
401 }

404 /*****
405 *
406 * FUNCTION:      AxGetNextInstance
407 *
408 * PARAMETERS:    InputPathname      - Filename for acpidump file
409 *                Signature          - Requested ACPI signature
410 *
411 * RETURN:        The next instance number for this signature. Zero if this
412 *                is the first instance of this signature.
413 *
414 * DESCRIPTION:   Get the next instance number of the specified table. If this
415 *                is the first instance of the table, create a new instance
416 *                block. Note: only SSDT and PSDDT tables can have multiple
417 *                instances.
418 *
419 *****/

421 static unsigned int
422 AxGetNextInstance (
423     char          *InputPathname,
424     char          *Signature)
425 {
426     AX_TABLE_INFO *Info;

429     Info = AxTableListHead;
430     while (Info)
431     {
432         if (*(UINT32 *) Signature == Info->Signature)
433         {
434             break;
435         }

437         Info = Info->Next;
438     }

440     if (!Info)
441     {
442         /* Signature not found, create new table info block */

444         Info = malloc (sizeof (AX_TABLE_INFO));
445         if (!Info)
446         {
447             printf ("Could not allocate memory\n");
448             exit (0);
449         }

451         Info->Signature = *(UINT32 *) Signature;
452         Info->Instances = AxCountTableInstances (InputPathname, Signature);
453         Info->NextInstance = 1;
454         Info->Next = AxTableListHead;

```

```

455     AxTableListHead = Info;
456 }

458     if (Info->Instances > 1)
459     {
460         return (Info->NextInstance++);
461     }

463     return (0);
464 }

467 /*****
468 *
469 * FUNCTION:      AxExtractTables
470 *
471 * PARAMETERS:    InputPathname      - Filename for acpidump file
472 *                Signature          - Requested ACPI signature to extract.
473 *                                NULL means extract ALL tables.
474 *                MinimumInstances    - Min instances that are acceptable
475 *
476 * RETURN:        Status
477 *
478 * DESCRIPTION:   Convert text ACPI tables to binary
479 *
480 *****/

482 int
483 AxExtractTables (
484     char          *InputPathname,
485     char          *Signature,
486     unsigned int   MinimumInstances)
487 {
488     FILE          *InputFile;
489     FILE          *OutputFile = NULL;
490     size_t        BytesWritten;
491     size_t        TotalBytesWritten = 0;
492     size_t        BytesConverted;
493     unsigned int   State = AX_STATE_FIND_HEADER;
494     unsigned int   FoundTable = 0;
495     unsigned int   Instances = 0;
496     unsigned int   ThisInstance;
497     char          ThisSignature[4];
498     int            Status = 0;

501     /* Open input in text mode, output is in binary mode */

503     InputFile = fopen (InputPathname, "rt");
504     if (!InputFile)
505     {
506         printf ("Could not open file %s\n", InputPathname);
507         return (-1);
508     }

510     if (Signature)
511     {
512         /* Are there enough instances of the table to continue? */

514         AxNormalizeSignature (Signature);

516         Instances = AxCountTableInstances (InputPathname, Signature);
517         if (Instances < MinimumInstances)
518         {
519             printf ("Table %s was not found in %s\n", Signature, InputPathname);
520             Status = -1;

```

```

521     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          * DSDT @ 0x737e4000
557          * XSDT @ 0x737f2fff
558          * RSD PTR @ 0xf6cd0
559          * SSDT @ (nil)
560          */
561         if (!strstr (LineBuffer, " @ "))
562         {
563             continue;
564         }

566         AxNormalizeSignature (LineBuffer);
567         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 (!strcmp (TableHeader->Signature, "RSD PTR ", 8))
737     {
738         AxCheckAscii ((char *) &Header[9], 6);
739         printf ("%7.4s          \"%6.6s\"\n", "RSDP", &Header[9]);
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 }

```



new/usr/src/common/acpica/tools/acpixtract/axmain.c

1

```
*****
5867 Fri Jan 17 07:31:20 2014
new/usr/src/common/acpica/tools/acpixtract/axmain.c
update to acpica-unix2-20140114
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: axmain - main module for acpixtract utility
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 "acapps.h"
47 #include <stdio.h>

50 static void
51 DisplayUsage (
52     void);

54 int
55 AxExtractTables (
56     char                *InputPathname,
57     char                *Signature,
58     unsigned int        MinimumInstances);

60 int
```

new/usr/src/common/acpica/tools/acpixtract/axmain.c

2

```
61 AxListTables (
62     char                *InputPathname);

65 /* Options */

67 #define AX_EXTRACT_ALL        0
68 #define AX_LIST_ALL          1
69 #define AX_EXTRACT_SIGNATURE  2
70 #define AX_EXTRACT_AML_TABLES 3

72 static int                AxAction = AX_EXTRACT_AML_TABLES; /* DSDT & SSDTs */

74 #define AX_OPTIONAL_TABLES    0
75 #define AX_REQUIRED_TABLE     1

77 #define AX_UTILITY_NAME       "ACPI Binary Table Extraction Utility"
78 #define AX_SUPPORTED_OPTIONS  "ahls:v"

81 /*****
82 *
83 * FUNCTION:    DisplayUsage
84 *
85 * DESCRIPTION: Usage message
86 *
87 *****/

89 static void
90 DisplayUsage (
91     void)
92 {
93
94     ACPI_USAGE_HEADER ("acpixtract [option] <InputFile>");
95
96     ACPI_OPTION ("-a",                "Extract all tables, not just DSDT/SSDT");
97     ACPI_OPTION ("-l",                "List table summaries, do not extract");
98     ACPI_OPTION ("-s <signature>",    "Extract all tables with <signature>");
99     ACPI_OPTION ("-v",                "Display version information");

101     printf ("\nExtract binary ACPI tables from text acpidump output\n");
102     printf ("Default invocation extracts the DSDT and all SSDTs\n");
103 }

106 /*****
107 *
108 * FUNCTION:    main
109 *
110 * DESCRIPTION: C main function
111 *
112 *****/

114 int
115 main (
116     int                argc,
117     char                *argv[])
118 {
119     char                *Filename;
120     int                Status;
121     int                j;

124     ACPI_DEBUG_INITIALIZE (); /* For debug version only */
125     printf (ACPI_COMMON_SIGNON (AX_UTILITY_NAME));
```

```

127     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 }

```

new/usr/src/common/acpica/tools/examples/examples.c

1

```
*****
13245 Fri Jan 17 07:31:20 2014
new/usr/src/common/acpica/tools/examples/examples.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
acpica-unix2-20130823
*****
1 /*****
2 *
3 * Module Name: examples - Example ACPICA initialization and execution code
4 *
5 *****/

7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 __EXAMPLES_C__
45 #include "examples.h"

47 #define _COMPONENT          ACPI_EXAMPLE
48         ACPI_MODULE_NAME    ("examples")

51 /*****
52 *
53 * ACPICA Example Code
54 *
55 * This module contains examples of how the host OS should interface to the
56 * ACPICA subsystem.
57 *
58 * 1) How to use the platform/acenv.h file and how to set configuration
59 * options.
```

new/usr/src/common/acpica/tools/examples/examples.c

2

```
60 *
61 * 2) main - using the debug output mechanism and the error/warning output
62 * macros.
63 *
64 * 3) Two examples of the ACPICA initialization sequence. The first is a
65 * initialization with no "early" ACPI table access. The second shows
66 * how to use ACPICA to obtain the tables very early during kernel
67 * initialization, even before dynamic memory is available.
68 *
69 * 4) How to invoke a control method, including argument setup and how to
70 * access the return value.
71 *
72 *****/

75 /* Local Prototypes */

77 static ACPI_STATUS
78 InitializeFullAcpica (void);

80 static ACPI_STATUS
81 InstallHandlers (void);

83 static void
84 NotifyHandler (
85     ACPI_HANDLE          Device,
86     UINT32               Value,
87     void                 *Context);

89 static void
90 ExecuteMAIN (void);

92 static void
93 ExecuteOSI (void);

95 ACPI_STATUS
96 InitializeAcpiTables (
97     void);

99 ACPI_STATUS
100 InitializeAcpi (
101     void);

104 /*****
105 *
106 * FUNCTION:    main
107 *
108 * PARAMETERS:  argc, argv
109 *
110 * RETURN:      Status
111 *
112 * DESCRIPTION: Main routine. Shows the use of the various output macros, as
113 * well as the use of the debug layer/level globals.
114 *
115 *****/

117 int ACPI_SYSTEM_XFACE
118 main (
119     int             argc,
120     char            **argv)
121 {

123     ACPI_DEBUG_INITIALIZE (); /* For debug version only */

125     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
259 * the ACPICA subsystem.
260 */
261 ACPI_STATUS
262 InitializeAcpi (
263     void)
264 {
265     ACPI_STATUS          Status;

268     /* Initialize the ACPICA subsystem */

270     Status = AcpiInitializeSubsystem ();
271     if (ACPI_FAILURE (Status))
272     {
273         return (Status);
274     }

276     /* Copy the root table list to dynamic memory */

278     Status = AcpiReallocateRootTable ();
279     if (ACPI_FAILURE (Status))
280     {
281         return (Status);
282     }

284     /* Create the ACPI namespace from ACPI tables */

286     Status = AcpiLoadTables ();
287     if (ACPI_FAILURE (Status))
288     {
289         return (Status);
290     }

292     /* Install local handlers */

294     Status = InstallHandlers ();
295     if (ACPI_FAILURE (Status))
296     {
297         ACPI_EXCEPTION ((AE_INFO, Status, "While installing handlers"));
298         return (Status);
299     }

301     /* Initialize the ACPI hardware */

303     Status = AcpiEnableSubsystem (ACPI_FULL_INITIALIZATION);
304     if (ACPI_FAILURE (Status))
305     {
306         return (Status);
307     }

309     /* Complete the ACPI namespace object initialization */

311     Status = AcpiInitializeObjects (ACPI_FULL_INITIALIZATION);
312     if (ACPI_FAILURE (Status))
313     {
314         return (Status);
315     }

317     return (AE_OK);
318 }

321 /*****
322  *
323  * Example ACPICA handler and handler installation

```

```

324 *
325 *****/

327 static void
328 NotifyHandler (
329     ACPI_HANDLE          Device,
330     UINT32               Value,
331     void                 *Context)
332 {
334     ACPI_INFO ((AE_INFO, "Received a notify 0x%X", Value));
335 }

338 static ACPI_STATUS
339 InstallHandlers (void)
340 {
341     ACPI_STATUS          Status;

344     /* Install global notify handler */

346     Status = AcpiInstallNotifyHandler (ACPI_ROOT_OBJECT, ACPI_SYSTEM_NOTIFY,
347                                         NotifyHandler, NULL);
348     if (ACPI_FAILURE (Status))
349     {
350         ACPI_EXCEPTION ((AE_INFO, Status, "While installing Notify handler"));
351         return (Status);
352     }

354     return (AE_OK);
355 }

358 /*****
359  *
360  * Examples of control method execution.
361  *
362  * _OSI is a predefined method that is implemented internally within ACPICA.
363  *
364  * Shows the following elements:
365  *
366  * 1) How to setup a control method argument and argument list
367  * 2) How to setup the return value object
368  * 3) How to invoke AcpiEvaluateObject
369  * 4) How to check the returned ACPI_STATUS
370  * 5) How to analyze the return value
371  *
372  *****/

374 static void
375 ExecuteOSI (void)
376 {
377     ACPI_STATUS          Status;
378     ACPI_OBJECT_LIST     ArgList;
379     ACPI_OBJECT           Arg[1];
380     ACPI_BUFFER           ReturnValue;
381     ACPI_OBJECT           *Object;

384     ACPI_INFO ((AE_INFO, "Executing _OSI reserved method"));

386     /* Setup input argument */

388     ArgList.Count = 1;
389     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 }

```

new/usr/src/common/acpica/tools/examples/examples.h

1

\*\*\*\*\*

2318 Fri Jan 17 07:31:20 2014

new/usr/src/common/acpica/tools/examples/examples.h

update to acpica-unix2-20140114

update to acpica-unix2-20131218

\*\*\*\*\*

```
1 /*****
2 *
3 * Module Name: examples.h - Common include for Examples program
4 *
5 *****/
```

```
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 _EXAMPLES_H
45 #define _EXAMPLES_H
```

```
47 #include "acpi.h"
48 #include "aecommon.h"
49 #include "acapps.h"
50 #include "../acpiexec/aecommon.h"
```

```
52 #include <stdio.h>
```

```
55 void
56 ExInitializeAcpiTables (
57     void);
```

```
59 #endif
```

new/usr/src/common/acpica/tools/examples/exstubs.c

1

```
*****
7187 Fri Jan 17 07:31:20 2014
new/usr/src/common/acpica/tools/examples/exstubs.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
*****
1 /*****
2 *
3 * Module Name: exstubs - Stub routines for the Example program
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #include "examples.h"
45
46 #include <acutils.h>
47 #include <acevents.h>
48 #include <acdispat.h>
49
50 #define _COMPONENT          ACPI_EXAMPLE
51         ACPI_MODULE_NAME    ("exstubs")
52
53
54 /*****
55 *
56 * DESCRIPTION: Stubs used to facilitate linkage of the example program
57 *
58 *****/
```

new/usr/src/common/acpica/tools/examples/exstubs.c

2

```
61 /* Utilities */
62
63 void
64 AcpiUtSubsystemShutdown (
65     void)
66 {
67 }
68
69 ACPI_STATUS
70 AcpiUtExecute_STA (
71     ACPI_NAMESPACE_NODE    *DeviceNode,
72     UINT32                  *StatusFlags)
73 {
74     return (AE_NOT_IMPLEMENTED);
75 }
76
77 ACPI_STATUS
78 AcpiUtExecute_HID (
79     ACPI_NAMESPACE_NODE    *DeviceNode,
80     ACPI_PNP_DEVICE_ID      **ReturnId)
81 {
82     return (AE_NOT_IMPLEMENTED);
83 }
84
85 ACPI_STATUS
86 AcpiUtExecute_CID (
87     ACPI_NAMESPACE_NODE    *DeviceNode,
88     ACPI_PNP_DEVICE_ID_LIST **ReturnCidList)
89 {
90     return (AE_NOT_IMPLEMENTED);
91 }
92
93 ACPI_STATUS
94 AcpiUtExecute_UID (
95     ACPI_NAMESPACE_NODE    *DeviceNode,
96     ACPI_PNP_DEVICE_ID      **ReturnId)
97 {
98     return (AE_NOT_IMPLEMENTED);
99 }
100
101 ACPI_STATUS
102 AcpiUtExecute_SUB (
103     ACPI_NAMESPACE_NODE    *DeviceNode,
104     ACPI_PNP_DEVICE_ID      **ReturnId)
105 {
106     return (AE_NOT_IMPLEMENTED);
107 }
108
109 ACPI_STATUS
110 AcpiUtExecutePowerMethods (
111     ACPI_NAMESPACE_NODE    *DeviceNode,
112     const char              **MethodNames,
113     UINT8                   MethodCount,
114     UINT8                   *OutValues)
115 {
116     return (AE_NOT_IMPLEMENTED);
117 }
118
119 ACPI_STATUS
120 AcpiUtEvaluateNumericObject (
121     char                    *ObjectName,
122     ACPI_NAMESPACE_NODE    *DeviceNode,
123     UINT64                  *Value)
124 {
125     return (AE_NOT_IMPLEMENTED);
126 }
```



```

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 }

261 ACPI_STATUS
262 AcpiEvQueueNotifyRequest (
263     ACPI_NAMESPACE_NODE    *Node,
264     UINT32                  NotifyValue)
265 {
266     return (AE_OK);
267 }

269 BOOLEAN
270 AcpiEvIsNotifyObject (
271     ACPI_NAMESPACE_NODE    *Node)
272 {
273     return (TRUE);
274 }

277 /* Namespace manager */

279 ACPI_STATUS
280 AcpiNsCheckReturnValue (
281     ACPI_NAMESPACE_NODE    *Node,
282     ACPI_EVALUATE_INFO     *Info,
283     UINT32                  UserParamCount,
284     ACPI_STATUS             ReturnStatus,
285     ACPI_OPERAND_OBJECT    **ReturnObjectPtr)
286 {
287     return (AE_OK);
288 }

290 void
291 AcpiNsCheckArgumentTypes (
292     ACPI_EVALUATE_INFO     *Info)
293 {
294     return;
295 }

297 void
298 AcpiNsCheckArgumentCount (
299     char                    *Pathname,
300     ACPI_NAMESPACE_NODE    *Node,
301     UINT32                  UserParamCount,
302     const ACPI_PREDEFINED_INFO *Predefined)
303 {
304     return;
305 }

307 void
308 AcpiNsCheckAcpiCompliance (
309     char                    *Pathname,
310     ACPI_NAMESPACE_NODE    *Node,
311     const ACPI_PREDEFINED_INFO *Predefined)
312 {
313     return;
314 }

316 const ACPI_PREDEFINED_INFO *
317 AcpiUtMatchPredefinedMethod (
318     char                    *Name)
319 {
320     return (NULL);
321 }

323 /* OSL interfaces */

```

```

325 ACPI_THREAD_ID
326 AcpiOsGetThreadId (
327     void)
328 {
329     return (1);
330 }

332 ACPI_STATUS
333 AcpiOsExecute (
334     ACPI_EXECUTE_TYPE      Type,
335     ACPI_OSD_EXEC_CALLBACK Function,
336     void                    *Context)
337 {
338     return (AE_SUPPORT);
339 }

```

new/usr/src/common/acpica/tools/examples/extables.c

1

```
*****
21052 Fri Jan 17 07:31:20 2014
new/usr/src/common/acpica/tools/examples/extables.c
update to acpica-unix2-20140114
update to acpica-unix2-20131218
*****
1 /*****
2 *
3 * Module Name: extables - ACPI tables for Example program
4 *
5 *****/
6
7 /*
8 * Copyright (C) 2000 - 2014, 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 MERCHANTABILITY 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 */
43
44 #define __EXTABLES_C__
45
46 #include "examples.h"
47 #include "actables.h"
48
49 #define _COMPONENT          ACPI_EXAMPLE
50         ACPI_MODULE_NAME    ("extables")
51
52 ACPI_PHYSICAL_ADDRESS
53 AeLocalGetRootPointer (
54     void);
55
56
57 /*****
58 *
59 * ACPICA Example tables and table setup
60 */
```

new/usr/src/common/acpica/tools/examples/extables.c

2

```
61 * This module contains the ACPI tables used for the example program. The
62 * original source code for the tables appears at the end of the module.
63 *
64 *****/
65
66
67 /* These tables will be modified at runtime */
68
69 unsigned char RsdpCode[] =
70 {
71     0x52,0x53,0x44,0x20,0x50,0x54,0x52,0x20, /* 00000000 "RSD PTR " */
72     0x43,0x49,0x4E,0x54,0x45,0x4C,0x20,0x02, /* 00000008 "CINTEL ." */
73     0x00,0x00,0x00,0x00,0x24,0x00,0x00,0x00, /* 00000010 "....$..." */
74     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000018 "....." */
75     0xDC,0x00,0x00,0x00 /* 00000020 "...." */
76 };
77
78 unsigned char RsdtCode[] =
79 {
80     0x52,0x53,0x44,0x54,0x28,0x00,0x00,0x00, /* 00000000 "RSDT(...)" */
81     0x01,0x10,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL " */
82     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
83     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
84     0x15,0x11,0x13,0x20,0x01,0x00,0x00,0x00 /* 00000020 "... .." */
85 };
86
87 unsigned char XsdtCode[] =
88 {
89     0x58,0x53,0x44,0x54,0x2C,0x00,0x00,0x00, /* 00000000 "XSDT,..." */
90     0x01,0x06,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...INTEL " */
91     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
92     0x01,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
93     0x15,0x11,0x13,0x20,0x01,0x00,0x00,0x00, /* 00000020 "... .." */
94     0x00,0x00,0x00,0x00 /* 00000028 "...." */
95 };
96
97 unsigned char FadtCode[] =
98 {
99     0x46,0x41,0x43,0x50,0x0C,0x01,0x00,0x00, /* 00000000 "FACP...." */
100     0x05,0x64,0x49,0x4E,0x54,0x45,0x4C,0x20, /* 00000008 "...GINTEL " */
101     0x54,0x45,0x4D,0x50,0x4C,0x41,0x54,0x45, /* 00000010 "TEMPLATE" */
102     0x00,0x00,0x00,0x00,0x49,0x4E,0x54,0x4C, /* 00000018 "....INTL" */
103     0x15,0x11,0x13,0x20,0x01,0x00,0x00,0x00, /* 00000020 "... .." */
104     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000028 "....." */
105     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000030 "....." */
106     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000038 "....." */
107     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000040 "....." */
108     0x01,0x00,0x00,0x00,0x01,0x00,0x00,0x00, /* 00000048 "....." */
109     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000050 "....." */
110     0x04,0x02,0x01,0x04,0x08,0x00,0x00,0x00, /* 00000058 "....." */
111     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000060 "....." */
112     0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000068 "....." */
113     0x00,0x00,0x00,0x00,0x01,0x08,0x00,0x01, /* 00000070 "....." */
114     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000078 "....." */
115     0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00, /* 00000080 "....." */
116     0x00,0x00,0x00,0x00,0x01,0x00,0x00,0x00, /* 00000088 "....." */
117     0x00,0x00,0x00,0x00,0x01,0x20,0x00,0x02, /* 00000090 "....." */
118     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 00000098 "....." */
119     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000A0 "....." */
120     0x00,0x00,0x00,0x00,0x01,0x10,0x00,0x02, /* 000000A8 "....." */
121     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000B0 "....." */
122     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000B8 "....." */
123     0x00,0x00,0x00,0x00,0x01,0x08,0x00,0x00, /* 000000C0 "....." */
124     0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00, /* 000000C8 "....." */
125     0x01,0x20,0x00,0x03,0x01,0x00,0x00,0x00, /* 000000D0 ". ...." */
126     0x00,0x00,0x00,0x00,0x01,0x40,0x00,0x01, /* 000000D8 ".....@..." */

```

```

127 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 0x01,0x00,0x00,0x00,0x00,0x49,0x4E,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 *Rsdtd = ACPI_CAST_PTR (ACPI_TABLE_RSDT, RsdtdCode);
171 ACPI_TABLE_XSDT *Xsdtd = ACPI_CAST_PTR (ACPI_TABLE_XSDT, XsdtdCode);
172 ACPI_TABLE_FADT *Fadtd = ACPI_CAST_PTR (ACPI_TABLE_FADT, FadtdCode);

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->RsdtdPhysicalAddress = (UINT32) ACPI_TO_INTEGER (RsdtdCode);
194 Rsdp->XsdtdPhysicalAddress = (UINT64) ACPI_TO_INTEGER (XsdtdCode);

196 /* RSDT and XSDT */

198 Rsdtd->TableOffsetEntry[0] = (UINT32) ACPI_TO_INTEGER (FadtdCode);
199 Xsdtd->TableOffsetEntry[0] = (UINT64) ACPI_TO_INTEGER (FadtdCode);

201 /* FADT */

203 Fadtd->Facs = 0;
204 Fadtd->Dsdt = 0;
205 Fadtd->XFacs = (UINT64) ACPI_TO_INTEGER (FacsCode);
206 Fadtd->XDsdtd = (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 Rsdtd->Header.Checksum = 0;
215 Rsdtd->Header.Checksum = (UINT8) -AcpiTbChecksum (
216     (void *) Rsdtd, Rsdtd->Header.Length);

218 Xsdtd->Header.Checksum = 0;
219 Xsdtd->Header.Checksum = (UINT8) -AcpiTbChecksum (
220     (void *) Xsdtd, Xsdtd->Header.Length);

222 Fadtd->Header.Checksum = 0;
223 Fadtd->Header.Checksum = (UINT8) -AcpiTbChecksum (
224     (void *) Fadtd, Fadtd->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 #ifndef 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
260 [0008]          XSDT Address : 0000000000000000
261 [0001]          Extended Checksum : DC
262 [0003]          Reserved : 000000

265 /* RSDT */

267 [0004]          Signature : "RSDT"      [Root System Description T
268 [0004]          Table Length : 00000044
269 [0001]          Revision : 01
270 [0001]          Checksum : B1
271 [0006]          Oem ID : "INTEL "
272 [0008]          Oem Table ID : "TEMPLATE"
273 [0004]          Oem Revision : 00000001
274 [0004]          Asl Compiler ID : "INTL"
275 [0004]          Asl Compiler Revision : 20100528

277 [0004]          ACPI Table Address 0 : 00000001

280 /* XSDT */

282 [0004]          Signature : "XSDT"      [Extended System Descripti
283 [0004]          Table Length : 00000064
284 [0001]          Revision : 01
285 [0001]          Checksum : 8B
286 [0006]          Oem ID : "INTEL "
287 [0008]          Oem Table ID : "TEMPLATE"
288 [0004]          Oem Revision : 00000001
289 [0004]          Asl Compiler ID : "INTL"
290 [0004]          Asl Compiler Revision : 20100528

292 [0008]          ACPI Table Address 0 : 0000000000000001

295 /* FADT */

297 [0004]          Signature : "FACP"      [Fixed ACPI Description Ta
298 [0004]          Table Length : 0000010C
299 [0001]          Revision : 05
300 [0001]          Checksum : 18
301 [0006]          Oem ID : "INTEL "
302 [0008]          Oem Table ID : "TEMPLATE"
303 [0004]          Oem Revision : 00000000
304 [0004]          Asl Compiler ID : "INTL"
305 [0004]          Asl Compiler Revision : 20111123

307 [0004]          FACS Address : 00000001
308 [0004]          DSDT Address : 00000001
309 [0001]          Model : 00
310 [0001]          PM Profile : 00 [Unspecified]
311 [0002]          SCI Interrupt : 0000
312 [0004]          SMI Command Port : 00000000
313 [0001]          ACPI Enable Value : 00
314 [0001]          ACPI Disable Value : 00
315 [0001]          S4BIOS Command : 00
316 [0001]          P-State Control : 00
317 [0004]          PM1A Event Block Address : 00000001
318 [0004]          PM1B Event Block Address : 00000000
319 [0004]          PM1A Control Block Address : 00000001
320 [0004]          PM1B Control Block Address : 00000000
321 [0004]          PM2 Control Block Address : 00000001
322 [0004]          PM Timer Block Address : 00000001
323 [0004]          GPE0 Block Address : 00000001
324 [0004]          GPE1 Block Address : 00000000
```

```
325 [0001]          PM1 Event Block Length : 04
326 [0001]          PM1 Control Block Length : 02
327 [0001]          PM2 Control Block Length : 01
328 [0001]          PM Timer Block Length : 04
329 [0001]          GPE0 Block Length : 08
330 [0001]          GPE1 Block Length : 00
331 [0001]          GPE1 Base Offset : 00
332 [0001]          _CST Support : 00
333 [0002]          C2 Latency : 0000
334 [0002]          C3 Latency : 0000
335 [0002]          CPU Cache Size : 0000
336 [0002]          Cache Flush Stride : 0000
337 [0001]          Duty Cycle Offset : 00
338 [0001]          Duty Cycle Width : 00
339 [0001]          RTC Day Alarm Index : 00
340 [0001]          RTC Month Alarm Index : 00
341 [0001]          RTC Century Index : 00
342 [0002]          Boot Flags (decoded below) : 0000
343          Legacy Devices Supported (V2) : 0
344          8042 Present on ports 60/64 (V2) : 0
345          VGA Not Present (V4) : 0
346          MSI Not Supported (V4) : 0
347          PCIe ASPM Not Supported (V4) : 0
348          CMOS RTC Not Present (V5) : 0
349 [0001]          Reserved : 00
350 [0004]          Flags (decoded below) : 00000000
351          WBINVD instruction is operational (V1) : 0
352          WBINVD flushes all caches (V1) : 0
353          All CPUs support C1 (V1) : 0
354          C2 works on MP system (V1) : 0
355          Control Method Power Button (V1) : 0
356          Control Method Sleep Button (V1) : 0
357          RTC wake not in fixed reg space (V1) : 0
358          RTC can wake system from S4 (V1) : 0
359          32-bit PM Timer (V1) : 0
360          Docking Supported (V1) : 0
361          Reset Register Supported (V2) : 0
362          Sealed Case (V3) : 0
363          Headless - No Video (V3) : 0
364          Use native instr after SLP_TYPx (V3) : 0
365          PCIEXP_WAK Bits Supported (V4) : 0
366          Use Platform Timer (V4) : 0
367          RTC_STS valid on S4 wake (V4) : 0
368          Remote Power-on capable (V4) : 0
369          Use APIC Cluster Model (V4) : 0
370          Use APIC Physical Destination Mode (V4) : 0
371          Hardware Reduced (V5) : 0
372          Low Power S0 Idle (V5) : 0

374 [0012]          Reset Register : [Generic Address Structure]
375 [0001]          Space ID : 01 [SystemIO]
376 [0001]          Bit Width : 08
377 [0001]          Bit Offset : 00
378 [0001]          Encoded Access Width : 01 [Byte Access:8]
379 [0008]          Address : 0000000000000001

381 [0001]          Value to cause reset : 00
382 [0003]          Reserved : 000000
383 [0008]          FACS Address : 0000000000000001
384 [0008]          DSDT Address : 0000000000000001
385 [0012]          PM1A Event Block : [Generic Address Structure]
386 [0001]          Space ID : 01 [SystemIO]
387 [0001]          Bit Width : 20
388 [0001]          Bit Offset : 00
389 [0001]          Encoded Access Width : 02 [Word Access:16]
390 [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

```

new/usr/src/pkg/manifests/diagnostic-acpitools.mf

1

\*\*\*\*\*

1032 Fri Jan 17 07:31:21 2014

new/usr/src/pkg/manifests/diagnostic-acpitools.mf

PANKOVs restructure

\*\*\*\*\*

```
1 #
2 # This file and its contents are supplied under the terms of the
3 # Common Development and Distribution License ("CDDL"), version 1.0.
4 # You may only use this file in accordance with the terms of version
5 # 1.0 of the CDDL.
6 #
7 # A full copy of the text of the CDDL should have accompanied this
8 # source. A copy of the CDDL is also available via the Internet at
9 # http://www.illumos.org/license/CDDL.
10 #
11 #
12 #
13 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
14 #
15 #
16 set name=pkg.fmri value=pkg:/diagnostic/acpitools@$(PKGVERS)
17 set name=pkg.description value=acpitools
18 set name=pkg.summary value=acpitools
19 set name=info.classification \
20     value="org.opensolaris.category.2008:Applications/System Utilities"
21 set name=variant.arch value=$(ARCH)
22 dir path=usr group=sys
23 dir path=usr/bin
24 file path=usr/bin/acpiexec mode=0555
25 file path=usr/bin/acpixtract mode=0555
26 file path=usr/bin/iasl mode=0555
27 license lic_CDDL license=lic_CDDL
28 license usr/src/common/acpica/THIRDPARTYLICENSE \
29     license=usr/src/common/acpica/THIRDPARTYLICENSE
```

new/usr/src/pkg/manifests/system-kernel.mf

1

\*\*\*\*\*

45175 Fri Jan 17 07:31:21 2014

new/usr/src/pkg/manifests/system-kernel.mf

acpica-unix2-20130823

\*\*\*\*\*

```
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 #
23 # Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
24 #
25 #
26 #
27 # The default for payload-bearing actions in this package is to appear in the
28 # global zone only. See the include file for greater detail, as well as
29 # information about overriding the defaults.
30 #
31 <include global_zone_only_component>
32 <include system-kernel.man1m.inc>
33 <include system-kernel.man2.inc>
34 <include system-kernel.man4.inc>
35 <include system-kernel.man5.inc>
36 <include system-kernel.man7.inc>
37 <include system-kernel.man7d.inc>
38 <include system-kernel.man7fs.inc>
39 <include system-kernel.man7m.inc>
40 <include system-kernel.man7p.inc>
41 <include system-kernel.man9.inc>
42 <include system-kernel.man9e.inc>
43 <include system-kernel.man9f.inc>
44 <include system-kernel.man9p.inc>
45 <include system-kernel.man9s.inc>
46 set name=pkg.fmri value=pkg:/system/kernel@$(PKGVERS)
47 set name=pkg.description \
48     value="core kernel software for a specific instruction-set architecture"
49 set name=pkg.summary value="Core Solaris Kernel"
50 set name=info.classification value=org.opensolaris.category.2008:System/Core
51 set name=variant.arch value=$(ARCH)
52 dir path=boot group=sys
53 $(i386_ONLY)dir path=boot/acpi group=sys
54 $(i386_ONLY)dir path=boot/acpi/tables group=sys
55 dir path=boot/solaris group=sys
56 dir path=boot/solaris/bin group=sys
57 dir path=etc group=sys
58 dir path=etc/crypto group=sys
59 dir path=etc/sock2path.d group=sys
60 dir path=kernel group=sys
61 $(i386_ONLY)dir path=kernel/$(ARCH64) group=sys
```

new/usr/src/pkg/manifests/system-kernel.mf

2

```
62 dir path=kernel/crypto group=sys
63 dir path=kernel/crypto/$(ARCH64) group=sys
64 dir path=kernel/dacf group=sys
65 dir path=kernel/dacf/$(ARCH64) group=sys
66 dir path=kernel/drv group=sys
67 dir path=kernel/drv/$(ARCH64) group=sys
68 dir path=kernel/exec group=sys
69 dir path=kernel/exec/$(ARCH64) group=sys
70 dir path=kernel/fs group=sys
71 dir path=kernel/fs/$(ARCH64) group=sys
72 dir path=kernel/ipp group=sys
73 dir path=kernel/ipp/$(ARCH64) group=sys
74 dir path=kernel/kiconv group=sys
75 dir path=kernel/kiconv/$(ARCH64) group=sys
76 dir path=kernel/mac group=sys
77 dir path=kernel/mac/$(ARCH64) group=sys
78 dir path=kernel/misc group=sys
79 dir path=kernel/misc/$(ARCH64) group=sys
80 dir path=kernel/misc/scsi_vhci group=sys
81 dir path=kernel/misc/scsi_vhci/$(ARCH64) group=sys
82 dir path=kernel/sched group=sys
83 dir path=kernel/sched/$(ARCH64) group=sys
84 dir path=kernel/socketmod group=sys
85 dir path=kernel/socketmod/$(ARCH64) group=sys
86 dir path=kernel/strmod group=sys
87 dir path=kernel/strmod/$(ARCH64) group=sys
88 dir path=kernel/sys group=sys
89 dir path=kernel/sys/$(ARCH64) group=sys
90 dir path=lib
91 dir path=lib/svc
92 dir path=lib/svc/manifest group=sys
93 dir path=lib/svc/manifest/system group=sys
94 dir path=lib/svc/method
95 dir path=usr/share/man
96 dir path=usr/share/man/man1m
97 dir path=usr/share/man/man2
98 dir path=usr/share/man/man3
99 dir path=usr/share/man/man4
100 dir path=usr/share/man/man5
101 dir path=usr/share/man/man7d
102 dir path=usr/share/man/man7fs
103 dir path=usr/share/man/man7m
104 dir path=usr/share/man/man7p
105 dir path=usr/share/man/man9
106 dir path=usr/share/man/man9e
107 dir path=usr/share/man/man9f
108 dir path=usr/share/man/man9p
109 dir path=usr/share/man/man9s
110 $(i386_ONLY)driver name=acpi_drv perms="* 0666 root sys"
111 driver name=aggr perms="* 0666 root sys"
112 driver name=arp perms="arp 0666 root sys"
113 driver name=bl perms="* 0666 root sys"
114 driver name=bridge clone_perms="bridge 0666 root sys" \
115     policy="read_priv_set=net_rawaccess write_priv_set=net_rawaccess"
116 $(sparc_ONLY)driver name=bscbus alias=SUNW,bscbus
117 $(i386_ONLY)driver name=bscbus alias=SVI0101
118 $(sparc_ONLY)driver name=bscv alias=SUNW,bscv perms="* 0644 root sys"
119 $(i386_ONLY)driver name=bscv
120 driver name=clone
121 driver name=cn perms="* 0620 root tty"
122 driver name=conskbd perms="kbd 0666 root sys"
123 driver name=consms perms="mouse 0666 root sys"
124 driver name=cpuid perms="self 0644 root sys"
125 $(i386_ONLY)driver name=cpunex alias=cpus
126 driver name=crypto perms="crypto 0666 root sys"
127 driver name=cryptoadm perms="cryptoadm 0644 root sys"
```



```

128 $(sparc_ONLY)driver name=dad alias=ide-disk perms="* 0640 root sys"
129 driver name=devinfo perms="devinfo 0640 root sys" \
130 perms="devinfo,ro 0444 root sys"
131 driver name=dld perms="* 0666 root sys"
132 driver name=dlpistub perms="* 0666 root sys"
133 $(sparc_ONLY)driver name=i8042 alias=8042
134 $(i386_ONLY)driver name=i8042
135 driver name=icmp perms="icmp 0666 root sys" \
136 policy="read_priv_set=net_icmpaccess write_priv_set=net_icmpaccess"
137 driver name=icmp6 perms="icmp6 0666 root sys" \
138 policy="read_priv_set=net_icmpaccess write_priv_set=net_icmpaccess"
139 $(i386_ONLY)driver name=intel_nb5000 \
140 alias=pci8086,25c0 \
141 alias=pci8086,25d0 \
142 alias=pci8086,25d4 \
143 alias=pci8086,25d8 \
144 alias=pci8086,3600 \
145 alias=pci8086,4000 \
146 alias=pci8086,4001 \
147 alias=pci8086,4003 \
148 alias=pci8086,65c0
149 $(i386_ONLY)driver name=intel_nhm \
150 alias=pci8086,3423 \
151 alias=pci8086,372a
152 $(i386_ONLY)driver name=intel_nhmex alias=pci8086,3438
153 driver name=ip perms="ip 0666 root sys" \
154 policy="read_priv_set=net_rawaccess write_priv_set=net_rawaccess"
155 driver name=ip6 perms="ip6 0666 root sys" \
156 policy="read_priv_set=net_rawaccess write_priv_set=net_rawaccess"
157 driver name=ipnet perms="lo0 0666 root sys" \
158 policy="read_priv_set=net_observability write_priv_set=net_observability"
159 driver name=ippctl
160 driver name=ipsecah perms="ipsecah 0666 root sys" \
161 policy="read_priv_set=sys_ip_config write_priv_set=sys_ip_config"
162 driver name=ipsecesp perms="ipsecesp 0666 root sys" \
163 policy="read_priv_set=sys_ip_config write_priv_set=sys_ip_config"
164 driver name=iptun
165 driver name=iwscn
166 driver name=kb8042 alias=pnpPNP,303
167 driver name=keysock perms="keysock 0666 root sys" \
168 policy="read_priv_set=sys_ip_config write_priv_set=sys_ip_config"
169 driver name=kmdb
170 driver name=kssl perms="* 0666 root sys"
171 driver name=llcl clone_perms="llcl 0666 root sys"
172 driver name=lofi perms="* 0600 root sys" perms="ctl 0644 root sys"
173 driver name=log perms="conslog 0666 root sys" perms="log 0640 root sys"
174 $(i386_ONLY)driver name=mc-amd \
175 alias=pci1022,1100 \
176 alias=pci1022,1101 \
177 alias=pci1022,1102
178 driver name=mm perms="allkmem 0600 root sys" perms="kmem 0640 root sys" \
179 perms="mem 0640 root sys" perms="null 0666 root sys" \
180 perms="zero 0666 root sys" \
181 policy="allkmem read_priv_set=all write_priv_set=all" \
182 policy="kmem read_priv_set=none write_priv_set=all" \
183 policy="mem read_priv_set=none write_priv_set=all"
184 driver name=mouse8042 alias=pnpPNP,f03
185 $(i386_ONLY)driver name=mpt class=scsi \
186 alias=pci1000,30 \
187 alias=pci1000,50 \
188 alias=pci1000,54 \
189 alias=pci1000,56 \
190 alias=pci1000,58 \
191 alias=pci1000,62 \
192 alias=pciex1000,56 \
193 alias=pciex1000,58 \

```

```

194 alias=pciex1000,62
195 driver name=nulldriver \
196 alias=scsa,nodev \
197 alias=scsa,probe
198 driver name=openeepr perms="openprom 0640 root sys" policy=write_priv_set=all
199 driver name=options
200 $(sparc_ONLY)driver name=pci_pci class=pci \
201 alias=pci1011,1 \
202 alias=pci1011,21 \
203 alias=pci1011,24 \
204 alias=pci1011,25 \
205 alias=pci1011,26 \
206 alias=pci1014,22 \
207 alias=pciclass,060400
208 $(i386_ONLY)driver name=pci_pci class=pci \
209 alias=pci1011,1 \
210 alias=pci1011,21 \
211 alias=pci1014,22 \
212 alias=pciclass,060400 \
213 alias=pciclass,060401
214 $(sparc_ONLY)driver name=pcieb \
215 alias=pciex108e,9010 \
216 alias=pciex108e,9020 \
217 alias=pciex10b5,8114 \
218 alias=pciex10b5,8516 \
219 alias=pciex10b5,8517 \
220 alias=pciex10b5,8518 \
221 alias=pciex10b5,8532 \
222 alias=pciex10b5,8533 \
223 alias=pciex10b5,8548 \
224 alias=pciexclass,060400
225 $(i386_ONLY)driver name=pcieb \
226 alias=pciexclass,060400 \
227 alias=pciexclass,060401
228 $(sparc_ONLY)driver name=pcieb_bcm alias=pciexl166,103
229 driver name=physmem perms="* 0600 root sys"
230 driver name=poll perms="* 0666 root sys"
231 $(sparc_ONLY)driver name=power alias=ali1535d+-power
232 $(i386_ONLY)driver name=power
233 driver name=pseudo alias=zconsnex
234 driver name=ptc perms="* 0666 root sys"
235 driver name=ptsl perms="* 0666 root sys"
236 $(sparc_ONLY)driver name=ramdisk alias=SUNW,ramdisk perms="* 0600 root sys" \
237 perms="ctl 0644 root sys"
238 $(i386_ONLY)driver name=ramdisk perms="* 0600 root sys" \
239 perms="ctl 0644 root sys"
240 driver name=random perms="* 0644 root sys" policy=write_priv_set=sys_devices
241 driver name=rts perms="rts 0666 root sys"
242 driver name=sad perms="admin 0666 root sys" perms="user 0666 root sys"
243 driver name=scsi_vhci class=scsi-self-identifying perms="* 0666 root sys" \
244 policy="devctl write_priv_set=sys_devices"
245 $(sparc_ONLY)driver name=sd perms="* 0640 root sys" \
246 alias=ide-cdrom \
247 alias=scsiclass,00 \
248 alias=scsiclass,05
249 $(i386_ONLY)driver name=sd perms="* 0640 root sys" \
250 alias=scsiclass,00 \
251 alias=scsiclass,05
252 driver name=sugen perms="* 0600 root sys" \
253 alias=scsa,08.bfcp \
254 alias=scsa,08.bvhci
255 driver name=simnet clone_perms="simnet 0666 root sys" perms="* 0666 root sys"
256 $(i386_ONLY)driver name=smbios perms="smbios 0444 root sys"
257 driver name=softmac
258 driver name=spdssock perms="spdssock 0666 root sys" \
259 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=vnuc clone_perms="vnuc 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/$(ARCH64)/ipsec group=sys
360 file path=kernel/drv/$(ARCH64)/ipsecsp group=sys
361 file path=kernel/drv/$(ARCH64)/iptun group=sys
362 file path=kernel/drv/$(ARCH64)/iwsn 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/$(ARCH64)/llcl group=sys
368 file path=kernel/drv/$(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)/spdsoc 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/ipsec group=sys
470 file path=kernel/drv/ipsec.conf group=sys
471 $(i386_ONLY)file path=kernel/drv/ipsec group=sys
472 file path=kernel/drv/ipsec.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/llcl group=sys
485 file path=kernel/drv/llcl.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/openeep group=sys
501 file path=kernel/drv/openeep.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/phymem group=sys
508 file path=kernel/drv/phymem.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 $(i386_ONLY)file path=kernel/misc/tlimod group=sys mode=0755
749 file path=kernel/misc/sched/$(ARCH64)/SDC group=sys mode=0755
750 file path=kernel/misc/sched/$(ARCH64)/TS group=sys mode=0755
751 file path=kernel/misc/sched/$(ARCH64)/TS_DPTBL group=sys mode=0755
752 $(i386_ONLY)file path=kernel/misc/sched/SDC group=sys mode=0755
753 $(i386_ONLY)file path=kernel/misc/sched/TS group=sys mode=0755
754 $(i386_ONLY)file path=kernel/misc/sched/TS_DPTBL group=sys mode=0755
755 file path=kernel/misc/socketmod/$(ARCH64)/ksslf group=sys mode=0755
756 file path=kernel/misc/socketmod/$(ARCH64)/socksctp group=sys mode=0755
757 file path=kernel/misc/socketmod/$(ARCH64)/trill group=sys mode=0755
758 $(i386_ONLY)file path=kernel/misc/socketmod/ksslf group=sys mode=0755
759 $(i386_ONLY)file path=kernel/misc/socketmod/socksctp group=sys mode=0755
760 $(i386_ONLY)file path=kernel/misc/socketmod/trill group=sys mode=0755
761 file path=kernel/misc/strmod/$(ARCH64)/bufmod group=sys mode=0755
762 file path=kernel/misc/strmod/$(ARCH64)/connld group=sys mode=0755
763 file path=kernel/misc/strmod/$(ARCH64)/dedump group=sys mode=0755
764 file path=kernel/misc/strmod/$(ARCH64)/drcompat group=sys mode=0755
765 file path=kernel/misc/strmod/$(ARCH64)/ldterm group=sys mode=0755
766 $(sparc_ONLY)file path=kernel/misc/strmod/$(ARCH64)/ms group=sys mode=0755
767 file path=kernel/misc/strmod/$(ARCH64)/pckt group=sys mode=0755
768 file path=kernel/misc/strmod/$(ARCH64)/pfmod group=sys mode=0755
769 file path=kernel/misc/strmod/$(ARCH64)/pipemod group=sys mode=0755
770 file path=kernel/misc/strmod/$(ARCH64)/ptem group=sys mode=0755
771 file path=kernel/misc/strmod/$(ARCH64)/redirmod group=sys mode=0755
772 file path=kernel/misc/strmod/$(ARCH64)/rpcmod group=sys mode=0755
773 file path=kernel/misc/strmod/$(ARCH64)/timod group=sys mode=0755
774 file path=kernel/misc/strmod/$(ARCH64)/tirdwr group=sys mode=0755
775 file path=kernel/misc/strmod/$(ARCH64)/ttcompat group=sys mode=0755
776 $(sparc_ONLY)file path=kernel/misc/strmod/$(ARCH64)/vuid3ps2 group=sys mode=0755
777 $(i386_ONLY)file path=kernel/misc/strmod/bufmod group=sys mode=0755
778 $(i386_ONLY)file path=kernel/misc/strmod/connld group=sys mode=0755
779 $(i386_ONLY)file path=kernel/misc/strmod/dedump group=sys mode=0755
780 $(i386_ONLY)file path=kernel/misc/strmod/drcompat group=sys mode=0755
781 $(i386_ONLY)file path=kernel/misc/strmod/ldterm group=sys mode=0755
782 $(i386_ONLY)file path=kernel/misc/strmod/pckt group=sys mode=0755
783 $(i386_ONLY)file path=kernel/misc/strmod/pfmod group=sys mode=0755
784 $(i386_ONLY)file path=kernel/misc/strmod/pipemod group=sys mode=0755
785 $(i386_ONLY)file path=kernel/misc/strmod/ptem group=sys mode=0755
786 $(i386_ONLY)file path=kernel/misc/strmod/redirmod group=sys mode=0755
787 $(i386_ONLY)file path=kernel/misc/strmod/rpcmod group=sys mode=0755

```

```

788 $(i386_ONLY)file path=kernel/strmod/timod group=sys mode=0755
789 $(i386_ONLY)file path=kernel/strmod/tirdwr group=sys mode=0755
790 $(i386_ONLY)file path=kernel/strmod/ttcompat group=sys mode=0755
791 file path=kernel/sys/$(ARCH64)/c2audit group=sys mode=0755
792 file path=kernel/sys/$(ARCH64)/doorfs group=sys mode=0755
793 file path=kernel/sys/$(ARCH64)/inst_sync group=sys mode=0755
794 file path=kernel/sys/$(ARCH64)/kaio group=sys mode=0755
795 file path=kernel/sys/$(ARCH64)/msgsys group=sys mode=0755
796 file path=kernel/sys/$(ARCH64)/pipe group=sys mode=0755
797 file path=kernel/sys/$(ARCH64)/portfs group=sys mode=0755
798 file path=kernel/sys/$(ARCH64)/pset group=sys mode=0755
799 file path=kernel/sys/$(ARCH64)/semsys group=sys mode=0755
800 file path=kernel/sys/$(ARCH64)/shmsys group=sys mode=0755
801 $(i386_ONLY)file path=kernel/sys/c2audit group=sys mode=0755
802 $(i386_ONLY)file path=kernel/sys/doorfs group=sys mode=0755
803 $(i386_ONLY)file path=kernel/sys/inst_sync group=sys mode=0755
804 $(i386_ONLY)file path=kernel/sys/kaio group=sys mode=0755
805 $(i386_ONLY)file path=kernel/sys/msgsys group=sys mode=0755
806 $(i386_ONLY)file path=kernel/sys/pipe group=sys mode=0755
807 $(i386_ONLY)file path=kernel/sys/portfs group=sys mode=0755
808 $(i386_ONLY)file path=kernel/sys/pset group=sys mode=0755
809 $(i386_ONLY)file path=kernel/sys/semsys group=sys mode=0755
810 $(i386_ONLY)file path=kernel/sys/shmsys group=sys mode=0755
811 file path=lib/svc/manifest/system/dumpadm.xml group=sys mode=0444
812 file path=lib/svc/manifest/system/intrd.xml group=sys mode=0444
813 file path=lib/svc/manifest/system/scheduler.xml group=sys mode=0444
814 file path=lib/svc/method/svc-dumpadm mode=0555
815 file path=lib/svc/method/svc-intrd mode=0555
816 file path=lib/svc/method/svc-scheduler mode=0555
817 $(sparc_ONLY)file path=usr/share/man/man1m/monitor.1m
818 $(sparc_ONLY)file path=usr/share/man/man1m/obpsym.1m
819 # On SPARC driver/bcscv is Serverbladel specific, and in system/kernel/platform
820 # We keep the manual page generic
821 $(sparc_ONLY)file path=usr/share/man/man7d/dad.7d
822 $(i386_ONLY)file path=usr/share/man/man7d/smbios.7d
823 # Sadly vuid mouse support is in different packages on different platforms
824 # While kstat(7D) is in SUNWcs, the structures are general
825 hardlink path=kernel/misc/$(ARCH64)/md5 \
826     target=../../../../kernel/crypto/$(ARCH64)/md5
827 hardlink path=kernel/misc/$(ARCH64)/sha1 \
828     target=../../../../kernel/crypto/$(ARCH64)/sha1
829 hardlink path=kernel/misc/$(ARCH64)/sha2 \
830     target=../../../../kernel/crypto/$(ARCH64)/sha2
831 $(i386_ONLY)hardlink path=kernel/misc/md5 target=../../../../kernel/crypto/md5
832 $(i386_ONLY)hardlink path=kernel/misc/sha1 target=../../../../kernel/crypto/sha1
833 $(i386_ONLY)hardlink path=kernel/misc/sha2 target=../../../../kernel/crypto/sha2
834 hardlink path=kernel/socketmod/$(ARCH64)/icmp \
835     target=../../../../kernel/drv/$(ARCH64)/icmp
836 hardlink path=kernel/socketmod/$(ARCH64)/rts \
837     target=../../../../kernel/drv/$(ARCH64)/rts
838 hardlink path=kernel/socketmod/$(ARCH64)/tcp \
839     target=../../../../kernel/drv/$(ARCH64)/tcp
840 hardlink path=kernel/socketmod/$(ARCH64)/udp \
841     target=../../../../kernel/drv/$(ARCH64)/udp
842 $(i386_ONLY)hardlink path=kernel/socketmod/icmp target=../../../../kernel/drv/icmp
843 $(i386_ONLY)hardlink path=kernel/socketmod/rts target=../../../../kernel/drv/rts
844 $(i386_ONLY)hardlink path=kernel/socketmod/tcp target=../../../../kernel/drv/tcp
845 $(i386_ONLY)hardlink path=kernel/socketmod/udp target=../../../../kernel/drv/udp
846 hardlink path=kernel/strmod/$(ARCH64)/arp \
847     target=../../../../kernel/drv/$(ARCH64)/arp
848 hardlink path=kernel/strmod/$(ARCH64)/icmp \
849     target=../../../../kernel/drv/$(ARCH64)/icmp
850 hardlink path=kernel/strmod/$(ARCH64)/ip \
851     target=../../../../kernel/drv/$(ARCH64)/ip
852 hardlink path=kernel/strmod/$(ARCH64)/ipsec \
853     target=../../../../kernel/drv/$(ARCH64)/ipsec

```

```

854 hardlink path=kernel/strmod/$(ARCH64)/ipsec \
855     target=../../../../kernel/drv/$(ARCH64)/ipsec
856 hardlink path=kernel/strmod/$(ARCH64)/keysock \
857     target=../../../../kernel/drv/$(ARCH64)/keysock
858 hardlink path=kernel/strmod/$(ARCH64)/tcp \
859     target=../../../../kernel/drv/$(ARCH64)/tcp
860 hardlink path=kernel/strmod/$(ARCH64)/udp \
861     target=../../../../kernel/drv/$(ARCH64)/udp
862 $(i386_ONLY)hardlink path=kernel/strmod/arp target=../../../../kernel/drv/arp
863 $(i386_ONLY)hardlink path=kernel/strmod/icmp target=../../../../kernel/drv/icmp
864 $(i386_ONLY)hardlink path=kernel/strmod/ip target=../../../../kernel/drv/ip
865 $(i386_ONLY)hardlink path=kernel/strmod/ipsec \
866     target=../../../../kernel/drv/ipsec
867 $(i386_ONLY)hardlink path=kernel/strmod/ipsec \
868     target=../../../../kernel/drv/ipsec
869 $(i386_ONLY)hardlink path=kernel/strmod/keysock \
870     target=../../../../kernel/drv/keysock
871 $(i386_ONLY)hardlink path=kernel/strmod/tcp target=../../../../kernel/drv/tcp
872 $(i386_ONLY)hardlink path=kernel/strmod/udp target=../../../../kernel/drv/udp
873 hardlink path=kernel/sys/$(ARCH64)/autofs \
874     target=../../../../kernel/fs/$(ARCH64)/autofs
875 hardlink path=kernel/sys/$(ARCH64)/rpcmod \
876     target=../../../../kernel/strmod/$(ARCH64)/rpcmod
877 $(i386_ONLY)hardlink path=kernel/sys/autofs target=../../../../kernel/fs/autofs
878 $(i386_ONLY)hardlink path=kernel/sys/rpcmod target=../../../../kernel/strmod/rpcmod
879 legacy pkg=SUNWckr \
880     desc="core kernel software for a specific instruction-set architecture" \
881     name="Core Solaris Kernel (Root)"
882 license cr_Sun license=cr_Sun
883 license lic_CDDL license=lic_CDDL
884 license usr/src/cmd/mdb/common/libstand/THIRDPARTYLICENSE \
885     license=usr/src/cmd/mdb/common/libstand/THIRDPARTYLICENSE
886 $(i386_ONLY)license usr/src/common/acpica/THIRDPARTYLICENSE \
887     license=usr/src/common/acpica/THIRDPARTYLICENSE
888 license usr/src/common/bzip2/LICENSE license=usr/src/common/bzip2/LICENSE
889 license usr/src/common/crypto/THIRDPARTYLICENSE.cryptogams \
890     license=usr/src/common/crypto/THIRDPARTYLICENSE.cryptogams
891 $(i386_ONLY)license usr/src/common/crypto/aes/amd64/THIRDPARTYLICENSE.gladman \
892     license=usr/src/common/crypto/aes/amd64/THIRDPARTYLICENSE.gladman
893 $(i386_ONLY)license usr/src/common/crypto/aes/amd64/THIRDPARTYLICENSE.openssl \
894     license=usr/src/common/crypto/aes/amd64/THIRDPARTYLICENSE.openssl
895 license usr/src/common/crypto/ecc/THIRDPARTYLICENSE \
896     license=usr/src/common/crypto/ecc/THIRDPARTYLICENSE
897 $(i386_ONLY)license usr/src/common/crypto/md5/amd64/THIRDPARTYLICENSE \
898     license=usr/src/common/crypto/md5/amd64/THIRDPARTYLICENSE
899 license usr/src/common/mpi/THIRDPARTYLICENSE \
900     license=usr/src/common/mpi/THIRDPARTYLICENSE
901 license usr/src/uts/common/inet/ip/THIRDPARTYLICENSE.rts \
902     license=usr/src/uts/common/inet/ip/THIRDPARTYLICENSE.rts
903 license usr/src/uts/common/inet/tcp/THIRDPARTYLICENSE \
904     license=usr/src/uts/common/inet/tcp/THIRDPARTYLICENSE
905 license usr/src/uts/common/io/THIRDPARTYLICENSE.etheraddr \
906     license=usr/src/uts/common/io/THIRDPARTYLICENSE.etheraddr
907 license usr/src/uts/common/sys/THIRDPARTYLICENSE.icu \
908     license=usr/src/uts/common/sys/THIRDPARTYLICENSE.icu
909 license usr/src/uts/common/sys/THIRDPARTYLICENSE.unicode \
910     license=usr/src/uts/common/sys/THIRDPARTYLICENSE.unicode
909 $(i386_ONLY)license usr/src/uts/intel/io/acpica/THIRDPARTYLICENSE \
910     license=usr/src/uts/intel/io/acpica/THIRDPARTYLICENSE
911 $(i386_ONLY)link path=boot/solaris/bin/root_archive \
912     target=../../../../usr/sbin/root_archive
913 link path=dev/dld target=../../../../devices/pseudo/dld@0:ctl
914 link path=kernel/misc/$(ARCH64)/des \
915     target=../../../../kernel/crypto/$(ARCH64)/des
916 $(i386_ONLY)link path=kernel/misc/des target=../../../../kernel/crypto/des

```

new/usr/src/tools/findunref/exception\_list.open

1

```
*****
7434 Fri Jan 17 07:31:21 2014
new/usr/src/tools/findunref/exception_list.open
acpica-unix2-20130823
*****
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 #
23 # Copyright (c) 2001, 2010, Oracle and/or its affiliates. All rights reserved.
24 #
25 #
26 #
27 # open-tree exception list
28 #
29 # See README.exception_lists for details
30 #
31 #
32 #
33 # Ignore oddly-named text files scattered about -- someday these should be
34 # suffixed with .txt so we don't have to list them.
35 #
36 ./usr/src/cmd/oawk/EXPLAIN
37 ./usr/src/cmd/vi/port/ex.news
38 ./usr/src/cmd/ssh/doc
39 #
40 #
41 # Ignore everything under trees that may be resynched from outside ON.
42 #
43 ./usr/src/cmd/perl
44 ./usr/src/cmd/sqlite
45 ./usr/src/lib/libsqlite
46 ./usr/src/cmd/tcpdump
47 ./usr/src/common/openssl
48 ./usr/src/grub
49 ./usr/src/cmd/acpitools
50 ./usr/src/common/acpica
49 ./usr/src/uts/intel/sys/acpi
51 #
52 #
53 # Ignore ksh93/ast-related files that are only used to resync our build
54 # configuration with upstream.
55 #
56 ./usr/src/lib/libast/*/src/lib/libast/FEATURE
57 ./usr/src/lib/libast/*/src/lib/libast/ast_namval.h
58 ./usr/src/lib/libast/common/comp/conf.*
59 ./usr/src/lib/libast/common/features
60 ./usr/src/lib/libast/common/include/ast_windows.h
```

new/usr/src/tools/findunref/exception\_list.open

2

```
61 ./usr/src/lib/libast/common/port/lc.tab
62 ./usr/src/lib/libast/common/port/lcgen.c
63 ./usr/src/lib/libcmd/*/src/lib/libcmd/FEATURE
64 ./usr/src/lib/libcmd/common/features
65 ./usr/src/lib/libdll/*/src/lib/libdll/FEATURE
66 ./usr/src/lib/libdll/common/features
67 ./usr/src/lib/libpp/*/*pp.*
68 ./usr/src/lib/libpp/common/gentab.sh
69 ./usr/src/lib/libpp/common/ppsym.c
70 ./usr/src/lib/libpp/i386/ppdebug.h
71 ./usr/src/lib/libpp/sparc/ppdebug.h
72 ./usr/src/lib/libshell/*/src/cmd/ksh93/FEATURE
73 ./usr/src/lib/libshell/common/data/math.tab
74 ./usr/src/lib/libshell/common/features
75 ./usr/src/lib/libshell/misc/buildksh93.sh
76 ./usr/src/lib/libshell/misc/buildksh93.readme
77 #
78 #
79 # Ignore ksh93/ast-related "iffe" (if feature enabled) probe
80 #
81 ./usr/src/lib/libsum/common/features/sum
82 #
83 #
84 # Ignore ksh93/ast-related upstream source, currently superseded by
85 # a per-platform version of sum.h, since we use libmd.so.1 for some
86 # ciphers.
87 #
88 ./usr/src/lib/libsum/common/sum.h
89 #
90 #
91 # Ignore ksh93/ast-related test programs.
92 #
93 ./usr/src/cmd/ast/msgcc/msgcc.tst
94 ./usr/src/lib/libast/common/port/astmath.c
95 #
96 #
97 # Ignore ksh93/ast-related source components that are not currently
98 # used but may be useful later.
99 #
100 ./usr/src/lib/libcmd/common/cksum.c
101 ./usr/src/lib/libcmd/common/md5sum.c
102 ./usr/src/lib/libcmd/common/sum.c
103 ./usr/src/lib/libshell/common/btins/mksservice.c
104 ./usr/src/lib/libshell/common/data/bash_pre_rc.sh
105 ./usr/src/lib/libshell/common/include/env.h
106 ./usr/src/lib/libshell/common/sh/bash.c
107 ./usr/src/lib/libshell/common/sh/env.c
108 ./usr/src/lib/libshell/common/sh/shcomp.c
109 ./usr/src/lib/libshell/common/sh/suid_exec.c
110 #
111 #
112 # Ignore any files built as part of the nightly program itself.
113 #
114 # ISUSED - let checkpaths know that the next entry is good.
115 ./usr/src/*.out
116 # ISUSED - let checkpaths know that the next entry is good.
117 ./usr/src/*.ref
118 #
119 #
120 # Ignore internal test directories and test programs.
121 #
122 */tests
123 */test
124 *Test.java
125 *_test.[ch]
126 ./usr/src/cmd/ldap/common/*test.c
```

## new/usr/src/tools/findunref/exception\_list.open

3

```
127 ./usr/src/cmd/logadm/tester
128 ./usr/src/cmd/print/printmgr/com/sun/admin/pm/client/helptools/extract
129 ./usr/src/cmd/print/printmgr/com/sun/admin/pm/server/pmtest
130 ./usr/src/cmd/sendmail/libsm/t-*.c
131 ./usr/src/cmd/sort/common/convert.c
132 ./usr/src/cmd/sort/common/invoke.c
133 ./usr/src/lib/crypt_modules/sha256/test.c
134 ./usr/src/lib/efcode/fcode_test
135 ./usr/src/lib/libkvm/common/test.c
136 ./usr/src/lib/libsaveargs/tests/

138 #
139 # Ignore debugging code.
140 #
141 ./usr/src/cmd/fs.d/pcfs/fsck/inject.c
142 ./usr/src/cmd/sort/common/statistics.c

144 #
145 # Ignore internal packages, scripts, and tools that are intentionally not
146 # built or used during a nightly.
147 #
148 ./usr/src/cmd/sgs/packages
149 ./usr/src/cmd/sgs/rtld.4.x
150 ./usr/src/prototypes
151 ./usr/src/cmd/pools/poold/com/sun/solaris/*/package.html
152 ./usr/src/uts/intel/io/acpica/cmp_ca.sh

153 #
154 # Ignore files that are only used by internal packages.
155 #
156 ./usr/src/cmd/sgs/*/chk.msg

158 #
159 # Ignore files that get used during a EXPORT_SRC or CRYPT_SRC build only.
160 #
161 ./usr/src/lib/gss_mechs/mech_dh/backend/mapfile-vers
162 ./usr/src/lib/gss_mechs/mech_dh/dh1024/mapfile-vers
163 ./usr/src/lib/gss_mechs/mech_dh/dh192/mapfile-vers
164 ./usr/src/lib/gss_mechs/mech_dh/dh640/mapfile-vers

166 #
167 # Ignore Makefiles which are used by developers but not used by nightly
168 # itself. This is a questionable practice, since they tend to rot.
169 #
170 ./usr/src/cmd/syslogd/sparcv9/Makefile
171 ./usr/src/uts/sparc/uhci/Makefile
172 ./usr/src/lib/pam_modules/smb/amd64/Makefile
173 ./usr/src/lib/pam_modules/smb/sparcv9/Makefile
174 ./usr/src/cmd/isns/isnsd/xml_def/isnsmgmtSchema.xsd

176 #
177 # Ignore dtrace scripts only used by developers
178 #
179 ./usr/src/cmd/vscan/vscand/vscan.d

181 #
182 # Ignore sample source code.
183 #
184 ./usr/src/cmd/sendmail/libmilter/example.c
185 ./usr/src/lib/libdhcpcsvc/modules/templates

187 #
188 # Ignore .xcl files that aren't used because the program is statically linked.
189 #
190 ./usr/src/cmd/cmd-inet/sbin/dhcpagent/dhcpagent.xcl
```

## new/usr/src/tools/findunref/exception\_list.open

4

```
192 #
193 # Ignore sendmail files included for completeness' sake, but which won't
194 # be used until certain _FFR (for future release) #define's go live.
195 #
196 ./usr/src/cmd/sendmail/src/statusd_shm.h

198 #
199 # Ignore files originally supplied by ISC (Internet Software Consortium)
200 # as part of a BIND release.
201 #
202 ./usr/src/lib/libresolv2/common/irs/getaddrinfo.c
203 ./usr/src/lib/libresolv2/common/irs/nis_p.h
204 ./usr/src/lib/libresolv2/common/resolv/res_mkupdate.h
205 ./usr/src/lib/libresolv2/include/err.h

207 #
208 # Ignore mont_mulf.c. It is used as a starting point for some hand optimized
209 # assembly files. We keep it around for future reference.
210 #
211 ./usr/src/common/bignum/mont_mulf.c

213 #
214 # Ignore the sparc Makefiles for x86-only drivers;
215 # they're used to build warlock only.
216 #
217 ./usr/src/uts/sparc/sata/Makefile
218 ./usr/src/uts/sparc/si3124/Makefile
219 ./usr/src/uts/sparc/nv_sata/Makefile
220 ./usr/src/uts/sparc/ahci/Makefile

222 #
223 # Ignore uttrack.c. It is provided as part of the standard
224 # ACPI CA source code but provides optional resource tracking
225 # functionality which is not used.
226 #
227 ./usr/src/uts/intel/io/acpica/utilities/uttrack.c

229 #
230 # Ignore any files that get used during a gcc build only.
231 #
232 ./usr/src/cmd/sgs/rtld/common/mapfile-order-gcc

234 #
235 # The sharemgr command is built 32-bit only by default, but support
236 # for building 64-bit is latent in the Makefiles.
237 #

239 #
240 # The sharemgr command is built 32-bit only by default, but support
241 # for building 64-bit is latent in the Makefiles.
242 #
243 ./usr/src/cmd/dfs.cmds/sharemgr/amd64/Makefile
244 ./usr/src/cmd/dfs.cmds/sharemgr/sparcv9/Makefile

246 #
247 # Legitimately unreferenced license/copying files. Please include
248 # explanatory comments when adding items here.
249 #

251 #
252 # OPENSOLARIS.LICENSE needs to remain in usr/src as long as it is
253 # referenced in the CDDL headers.
254 #
255 ./usr/src/OPENSOLARIS.LICENSE

257 #
258 # Though "COPYING" is usually used as a filename for GPL, the license
259 # information for openssl is actually found in usr/src/cmd/ssh/doc/LICENSE.
260 # The COPYING.Ylonen file is merely additional information.
261 #
262 ./usr/src/cmd/ssh/doc/COPYING.Ylonen
```



**new/usr/src/tools/findunref/exception\_list.open**

**5**

```
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
```

new/usr/src/uts/common/io/power.c

1

\*\*\*\*\*

32149 Fri Jan 17 07:31:22 2014

new/usr/src/uts/common/io/power.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 * Copyright 2011 Joyent, Inc. All rights reserved.
25 * Copyright 2011 Nexenta Systems, Inc. All rights reserved.
26 */

28 /*
29  *      Power Button Driver
30  *
31  *      This driver handles interrupt generated by the power button on
32  *      platforms with "power" device node which has "button" property.
33  *      Currently, these platforms are:
34  *
35  *          ACPI-enabled x86/x64 platforms
36  *          Ultra-5_10, Ultra-80, Sun-Blade-100, Sun-Blade-150,
37  *          Sun-Blade-1500, Sun-Blade-2500,
38  *          Sun-Fire-V210, Sun-Fire-V240, Netra-240
39  *
40  *      Only one instance is allowed to attach. In order to know when
41  *      an application that has opened the device is going away, a new
42  *      minor clone is created for each open(9E) request. There are
43  *      allocations for creating minor clones between 1 and 255. The iocctl
44  *      interface is defined by pbio(7I) and approved as part of
45  *      PSARC/1999/393 case.
46 */

48 #include <sys/types.h>
49 #include <sys/conf.h>
50 #include <sys/ddi.h>
51 #include <sys/sunddi.h>
52 #include <sys/ddi_impldefs.h>
53 #include <sys/cmn_err.h>
54 #include <sys/errno.h>
55 #include <sys/modctl.h>
56 #include <sys/open.h>
57 #include <sys/stat.h>
58 #include <sys/poll.h>
59 #include <sys/pbio.h>
60 #include <sys/sysevent/eventdefs.h>
61 #include <sys/sysevent/pwrctl.h>
```

new/usr/src/uts/common/io/power.c

2

```
63 #if defined(__sparc)
64 #include <sys/machsystem.h>
65 #endif

67 #ifdef ACPI_POWER_BUTTON

69 #include <acpica/include/acpi.h>
69 #include <sys/acpi/acpi.h>
70 #include <sys/acpica.h>

72 #else

74 #include <sys/epic.h>
75 /*
76  * Some #defs that must be here as they differ for power.c
77  * and epic.c
78 */
79 #define EPIC_REGS_OFFSET      0x00
80 #define EPIC_REGS_LEN        0x82

83 /*
84  * This flag, which is set for platforms, that have EPIC processor
85  * to process power button interrupt, helps in executing platform
86  * specific code.
87 */
88 static char      hasEPIC = B_FALSE;
89 #endif /* ACPI_POWER_BUTTON */

91 /*
92  * Maximum number of clone minors that is allowed. This value
93  * is defined relatively low to save memory.
94 */
95 #define POWER_MAX_CLONE 256

97 /*
98  * Minor number is instance < 8 + clone minor from range 1-255; clone 0
99  * is reserved for "original" minor.
100 */
101 #define POWER_MINOR_TO_CLONE(minor) ((minor) & (POWER_MAX_CLONE - 1))

103 /*
104  * Power Button Abort Delay
105 */
106 #define ABORT_INCREMENT_DELAY 10

108 /*
109  * FWARC 2005/687: power device compatible property
110 */
111 #define POWER_DEVICE_TYPE "power-device-type"

113 /*
114  * Driver global variables
115 */
116 static void *power_state;
117 static int power_inst = -1;

119 static hrtime_t power_button_debounce = NANOSEC/MILLISEC*10;
120 static hrtime_t power_button_abort_interval = 1.5 * NANOSEC;
121 static int power_button_abort_presses = 3;
122 static int power_button_abort_enable = 1;
123 static int power_button_enable = 1;

125 static int power_button_pressed = 0;
126 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 occurred 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

new/usr/src/uts/common/sys/hotkey\_drv.h

1

```
*****
3541 Fri Jan 17 07:31:22 2014
new/usr/src/uts/common/sys/hotkey_drv.h
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 /*
23  * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25 */
26
27 #ifndef _HOTKEY_DRV_H
28 #define _HOTKEY_DRV_H
29
30 #ifdef __cplusplus
31 extern "C" {
32 #endif
33
34 #include <sys/types.h>
35 #include <sys/conf.h>
36 #include <sys/stat.h>
37 #include <sys/note.h>
38 #include <sys/modctl.h>
39 #include <sys/ddi.h>
40 #include <sys/sunddi.h>
41 #include <acpica/include/acpi.h>
42 #include <sys/acpi/acpi.h>
43 #include <sys/sysevent/eventdefs.h>
44 #include <sys/acpi_drv.h>
45
46
47 #define ID_LEN          9
48
49 struct acpi_drv_dev {
50     ACPI_HANDLE hdl;
51     char hid[ID_LEN];      /* ACPI HardwareId */
52     char uid[ID_LEN];      /* ACPI UniqueId */
53     ACPI_INTEGER adr;      /* Bus device Id */
54     int valid;             /* the device state is valid */
55
56     /*
57      * Unlike most other devices, when a battery is inserted or
58      * removed from the system, the device itself(the battery bay)
59      * is still considered to be present in the system.
60      */
61 }
```

new/usr/src/uts/common/sys/hotkey\_drv.h

2

```
61     * Value:
62     *      0 -- Off-line
63     *      1 -- On-line
64     *     -1 -- Unknown
65     */
66     int present;
67     enum acpi_drv_type type;
68     int index;      /* device index */
69     int minor;
70 };
_____unchanged_portion_omitted_____
```

new/usr/src/uts/i86pc/cpu/amd\_opteron/ao\_mca.c

1

```
*****
27764 Fri Jan 17 07:31:22 2014
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/system.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)
```

new/usr/src/uts/i86pc/cpu/amd\_opteron/ao\_mca.c

2

```
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_____
```

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_container.c

1

\*\*\*\*\*  
8962 Fri Jan 17 07:31:23 2014

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_container.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 */
29
30 /*
31 * There are three types of container objects defined in the ACPI Spec as below.
32 * PNP0A05: Generic Container Device
33 *   A device whose settings are totally controlled by its ACPI resource
34 *   information, and otherwise needs no device or bus-specific driver support.
35 *   This was originally known as Generic ISA Bus Device.
36 *   This ID should only be used for containers that do not produce resources
37 *   for consumption by child devices. Any system resources claimed by a PNP0A05
38 *   device's _CRS object must be consumed by the container itself.
39 * PNP0A06: Generic Container Device
40 *   This device behaves exactly the same as the PNP0A05 device.
41 *   This was originally known as Extended I/O Bus.
42 *   This ID should only be used for containers that do not produce resources
43 *   for consumption by child devices. Any system resources claimed by a PNP0A06
44 *   device's _CRS object must be consumed by the container itself.
45 * ACPI0004: Module Device.
46 *   This device is a container object that acts as a bus node in a namespace.
47 *   A Module Device without any of the _CRS, _PRS and _SRS methods behaves
48 *   the same way as the Generic Container Devices (PNP0A05 or PNP0A06).
49 *   If the Module Device contains a _CRS method, only the resources
50 *   described in the _CRS are available for consumption by its child devices.
51 *   Also, the Module Device can support _PRS and _SRS methods if _CRS is
52 *   supported.
53 */
54
55 #include <sys/types.h>
56 #include <sys/atomic.h>
57 #include <sys/note.h>
58 #include <sys/sunddi.h>
59 #include <sys/sunndi.h>
60 #include <acpica/include/acpi.h>
61 #include <sys/acpi/acpi.h>
```

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_container.c

2

```
61 #include <sys/acpica.h>
62 #include <sys/acpidev.h>
63 #include <sys/acpidev_dr.h>
64 #include <sys/acpidev_impl.h>
65
66 static ACPI_STATUS acpidev_container_probe(acpidev_walk_info_t *infop);
67 static acpidev_filter_result_t acpidev_container_filter(
68     acpidev_walk_info_t *infop, char *devname, int maxlen);
69 static ACPI_STATUS acpidev_container_init(acpidev_walk_info_t *infop);
70 static acpidev_filter_result_t acpidev_container_filter_func(
71     acpidev_walk_info_t *infop, ACPI_HANDLE hdl, acpidev_filter_rule_t *rulep,
72     char *devname, int devnamelen);
73
74 /*
75  * Default class driver for ACPI container objects.
76 */
77 acpidev_class_t acpidev_class_container = {
78     0, /* adc_refcnt */
79     ACPIDEV_CLASS_REV1, /* adc_version */
80     ACPIDEV_CLASS_ID_CONTAINER, /* adc_class_id */
81     "ACPI Container", /* adc_class_name */
82     ACPIDEV_TYPE_CONTAINER, /* adc_dev_type */
83     NULL, /* adc_private */
84     NULL, /* adc_pre_probe */
85     NULL, /* adc_post_probe */
86     acpidev_container_probe, /* adc_probe */
87     acpidev_container_filter, /* adc_filter */
88     acpidev_container_init, /* adc_init */
89     NULL, /* adc_fini */
90 };
91
92 _____unchanged_portion_omitted_____
```

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_cpu.c

1

\*\*\*\*\*

21688 Fri Jan 17 07:31:23 2014

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_cpu.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 /*
23  * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25 */
26 /*
27  * Copyright (c) 2009-2010, Intel Corporation.
28  * All rights reserved.
29 */
30
31 /*
32  * [Support of X2APIC]
33  * According to the ACPI Spec, when using the X2APIC interrupt model, logical
34  * processors with APIC ID values of 255 and greater are required to have a
35  * Processor Device object and must convey the Processor's APIC information to
36  * OSPM using the Processor Local X2APIC structure. Logical Processors with APIC
37  * ID values less than 255 must use the Processor Local XAPIC structure to
38  * convey their APIC information to OSPM.
39 */
40
41 #include <sys/types.h>
42 #include <sys/atomic.h>
43 #include <sys/bootconf.h>
44 #include <sys/cpuvar.h>
45 #include <sys/machsystem.h>
46 #include <sys/note.h>
47 #include <sys/psm_types.h>
48 #include <sys/x86_archext.h>
49 #include <sys/sunddi.h>
50 #include <sys/sunndi.h>
51 #include <acpica/include/acpi.h>
52 #include <sys/acpi/acpi.h>
53 #include <sys/acpica.h>
54 #include <sys/acpidev.h>
55 #include <sys/acpidev_impl.h>
56
57 struct acpidev_cpu_map_item {
58     uint32_t      proc_id;
59     uint32_t      apic_id;
60 };
61
62 unchanged portion omitted
```

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_device.c

1

\*\*\*\*\*

5107 Fri Jan 17 07:31:24 2014

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_device.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) 2009-2010, Intel Corporation.
23 * All rights reserved.
24 */

26 #include <sys/types.h>
27 #include <sys/atomic.h>
28 #include <sys/sunddi.h>
29 #include <sys/sunndi.h>
30 #include <acpica/include/acpi.h>
30 #include <sys/acpi/acpi.h>
31 #include <sys/acpica.h>
32 #include <sys/acpidev.h>
33 #include <sys/acpidev_impl.h>

35 static ACPI_STATUS acpidev_device_probe(acpidev_walk_info_t *infop);
36 static acpidev_filter_result_t acpidev_device_filter(acpidev_walk_info_t *infop,
37     char *devname, int maxlen);
38 static ACPI_STATUS acpidev_device_init(acpidev_walk_info_t *infop);

40 static uint32_t acpidev_device_unitaddr = 0;

42 /*
43  * Default class driver for ACPI DEVICE objects.
44  * The default policy for DEVICE objects is to scan child objects without
45  * creating device nodes. But some special DEVICE objects will have device
46  * nodes created for them.
47  */
48 acpidev_class_t acpidev_class_device = {
49     0, /* adc_refcnt */
50     ACPIDEV_CLASS_REV1, /* adc_version */
51     ACPIDEV_CLASS_ID_DEVICE, /* adc_class_id */
52     "ACPI Device", /* adc_class_name */
53     ACPIDEV_TYPE_DEVICE, /* adc_dev_type */
54     NULL, /* adc_private */
55     NULL, /* adc_pre_probe */
56     NULL, /* adc_post_probe */
57     acpidev_device_probe, /* adc_probe */
58     acpidev_device_filter, /* adc_filter */
59     acpidev_device_init, /* adc_init */
60     NULL, /* adc_fini */

```

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_device.c

2

61 };

unchanged\_portion\_omitted



new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_dr.c

1

\*\*\*\*\*

73056 Fri Jan 17 07:31:24 2014

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_dr.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 */

22 /*
23  * Copyright (c) 2010, Intel Corporation.
24  * All rights reserved.
25  */
```

```
27 #include <sys/types.h>
28 #include <sys/atomic.h>
29 #include <sys/cmn_err.h>
30 #include <sys/cpuvar.h>
31 #include <sys/memlist.h>
32 #include <sys/memlist_impl.h>
33 #include <sys/note.h>
34 #include <sys/obpdefs.h>
35 #include <sys/synch.h>
36 #include <sys/sysmacros.h>
37 #include <sys/sunddi.h>
38 #include <sys/sunndi.h>
39 #include <sys/x86_archext.h>
40 #include <sys/machsystem.h>
41 #include <sys/memnode.h>          /* for lgrp_plat_node_cnt */
42 #include <sys/psm_types.h>
43 #include <acpica/include/acpi.h>
44 #include <sys/acpi/acpi.h>
45 #include <sys/acpica.h>
46 #include <sys/acpidev.h>
47 #include <sys/acpidev_rsc.h>
48 #include <sys/acpidev_dr.h>
49 #include <sys/acpidev_impl.h>
```

```
50 struct acpidev_dr_set_prop_arg {
51     uint32_t    level;
52     uint32_t    bdnum;
53     uint32_t    cpu_id;
54     uint32_t    mem_id;
55     uint32_t    io_id;
56     uint32_t    mod_id;
57 };
```

\_\_\_\_\_unchanged portion omitted\_\_\_\_\_

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_drv.c

1

\*\*\*\*\*

35905 Fri Jan 17 07:31:25 2014

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_drv.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) 2009-2010, Intel Corporation.
23 * All rights reserved.
24 */

26 /*
27 * Platform specific device enumerator for ACPI specific devices.
28 * "x86 system devices" refers to the suite of hardware components which are
29 * common to the x86 platform and play important roles in the system
30 * architecture but can't be enumerated/discovered through industry-standard
31 * bus specifications. Examples of these x86 system devices include:
32 *   * Logical processor/CPU
33 *   * Memory device
34 *   * Non-PCI discoverable IOMMU or DMA Remapping Engine
35 *   * Non-PCI discoverable IOxAPIC
36 *   * Non-PCI discoverable HPET (High Precision Event Timer)
37 *   * ACPI defined devices, including power button, sleep button, battery etc.
38 *
39 * X86 system devices may be discovered through BIOS/Firmware interfaces, such
40 * as SMBIOS tables, MPS tables and ACPI tables since their discovery isn't
41 * covered by any industry-standard bus specifications.
42 *
43 * In order to aid Solaris in flexibly managing x86 system devices,
44 * x86 system devices are placed into a specific firmware device
45 * subtree whose device path is '/devices/fw'.
46 *
47 * This driver populates the firmware device subtree with ACPI-discoverable
48 * system devices if possible. To achieve that, the ACPI object
49 * namespace is abstracted as ACPI virtual buses which host system devices.
50 * Another nexus driver for the ACPI virtual bus will manage all devices
51 * connected to it.
52 *
53 * For more detailed information, please refer to PSARC/2009/104.
54 */

56 #include <sys/types.h>
57 #include <sys/bitmap.h>
58 #include <sys/cmn_err.h>
59 #include <sys/ddi_subrdefs.h>
60 #include <sys/errno.h>
61 #include <sys/modctl.h>
```

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_drv.c

2

```
62 #include <sys/mutex.h>
63 #include <sys/note.h>
64 #include <sys/obpdefs.h>
65 #include <sys/sunddi.h>
66 #include <sys/sunndi.h>
67 #include <acpica/include/acpi.h>
67 #include <sys/acpi/acpi.h>
68 #include <sys/acpica.h>
69 #include <sys/acpidev.h>
70 #include <sys/acpidev_dr.h>
71 #include <sys/acpidev_impl.h>

73 /* Patchable through /etc/system */
74 int acpidev_options = 0;
75 int acpidev_debug = 0;

77 krwlock_t acpidev_class_lock;
78 acpidev_class_list_t *acpidev_class_list_root = NULL;
79 ulong_t acpidev_object_type_mask[BT_BITOUL(ACPI_TYPE_NS_NODE_MAX + 1)];

81 /* ACPI device autoconfig global status */
82 typedef enum acpidev_status {
83     ACPIDEV_STATUS_FAILED = -2,      /* ACPI device autoconfig failed */
84     ACPIDEV_STATUS_DISABLED = -1,    /* ACPI device autoconfig disabled */
85     ACPIDEV_STATUS_UNKNOWN = 0,      /* initial status */
86     ACPIDEV_STATUS_INITIALIZED,      /* ACPI device autoconfig initialized */
87     ACPIDEV_STATUS_FIRST_PASS,       /* first probing finished */
88     ACPIDEV_STATUS_READY              /* second probing finished */
89 } acpidev_status_t;
    unchanged_portion_omitted
```

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_memory.c

1

\*\*\*\*\*

5203 Fri Jan 17 07:31:25 2014

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_memory.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) 2009-2010, Intel Corporation.
23 * All rights reserved.
24 */
```

```
26 #include <sys/types.h>
27 #include <sys/atomic.h>
28 #include <sys/sunddi.h>
29 #include <sys/sunndi.h>
30 #include <acpica/include/acpi.h>
30 #include <sys/acpi/acpi.h>
31 #include <sys/acpica.h>
32 #include <sys/acpidev.h>
33 #include <sys/acpidev_rsc.h>
34 #include <sys/acpidev_dr.h>
35 #include <sys/acpidev_impl.h>
```

```
37 static ACPI_STATUS acpidev_memory_probe(acpidev_walk_info_t *infop);
38 static acpidev_filter_result_t acpidev_memory_filter(
39     acpidev_walk_info_t *infop, char *devname, int maxlen);
40 static ACPI_STATUS acpidev_memory_init(acpidev_walk_info_t *infop);
```

```
42 /*
43  * Default class driver for ACPI memory objects.
44  */
45 acpidev_class_t acpidev_class_memory = {
46     0, /* adc_refcnt */
47     ACPIDEV_CLASS_REV1, /* adc_version */
48     ACPIDEV_CLASS_ID_MEMORY, /* adc_class_id */
49     "ACPI memory", /* adc_class_name */
50     ACPIDEV_TYPE_MEMORY, /* adc_dev_type */
51     NULL, /* adc_private */
52     NULL, /* adc_pre_probe */
53     NULL, /* adc_post_probe */
54     acpidev_memory_probe, /* adc_probe */
55     acpidev_memory_filter, /* adc_filter */
56     acpidev_memory_init, /* adc_init */
57     NULL, /* adc_fini */
58 };
```

unchanged portion omitted

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_pci.c

1

\*\*\*\*\*

4851 Fri Jan 17 07:31:26 2014

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_pci.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 /*
23  * Copyright (c) 2010, Intel Corporation.
24  * All rights reserved.
25  */
```

```
27 #include <sys/types.h>
28 #include <sys/atomic.h>
29 #include <sys/sunddi.h>
30 #include <sys/sunndi.h>
31 #include <acpica/include/acpi.h>
31 #include <sys/acpi/acpi.h>
32 #include <sys/acpica.h>
33 #include <sys/acpidev.h>
34 #include <sys/acpidev_rsc.h>
35 #include <sys/acpidev_dr.h>
36 #include <sys/acpidev_impl.h>
```

```
38 static ACPI_STATUS acpidev_pci_probe(acpidev_walk_info_t *infop);
```

```
40 /*
41  * Default class driver for PCI/PCIEX Host Bridge devices.
42  */
43 acpidev_class_t acpidev_class_pci = {
44     0, /* adc_refcnt */
45     ACPIDEV_CLASS_REV1, /* adc_version */
46     ACPIDEV_CLASS_ID_PCI, /* adc_class_id */
47     "PCI/PCIex Host Bridge", /* adc_class_name */
48     ACPIDEV_TYPE_PCI, /* adc_dev_type */
49     NULL, /* adc_private */
50     NULL, /* adc_pre_probe */
51     NULL, /* adc_post_probe */
52     acpidev_pci_probe, /* adc_probe */
53     NULL, /* adc_filter */
54     NULL, /* adc_init */
55     NULL, /* adc_fini */
56 };
```

\_\_\_\_\_unchanged\_portion\_omitted\_\_\_\_\_

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_resource.c

1

\*\*\*\*\*

30644 Fri Jan 17 07:31:26 2014

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_resource.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/types.h>
31 #include <sys/cmn_err.h>
32 #include <sys/sysmacros.h>
33 #include <sys/sunddi.h>
34 #include <sys/sunndi.h>
35 #include <acpica/include/acpi.h>
36 #include <sys/acpi/acpi.h>
37 #include <sys/acpica.h>
38 #include <sys/acpidev.h>
39 #include <sys/acpidev_rsc.h>
40 #include <sys/acpidev_impl.h>

41 #define ACPIDEV_RES_INIT_ITEMS      8
42 #define ACPIDEV_RES_INCR_ITEMS      8

44 /* Data structure to hold parsed resources during walking. */
45 struct acpidev_resource_handle {
46     boolean_t
47     int
48     int
49     acpidev_phys_spec_t
50     acpidev_phys_spec_t
51     int
52     int
53     acpidev_ranges_t
54     acpidev_ranges_t
55     int
56     int
57     acpidev_bus_range_t
58     acpidev_bus_range_t
59     int
60     int
61     acpidev_consumer;
62     acpidev_reg_count;
63     acpidev_reg_max;
64     *acpidev_regpp;
65     acpidev_regs[ACPIDEV_RES_INIT_ITEMS];
66     acpidev_range_count;
67     acpidev_range_max;
68     *acpidev_rangepp;
69     acpidev_ranges[ACPIDEV_RES_INIT_ITEMS];
70     acpidev_bus_count;
71     acpidev_bus_max;
72     *acpidev_busp;
73     acpidev_buses[ACPIDEV_RES_INIT_ITEMS];
74     acpidev_irq_count;
75     acpidev_irqp[ACPIDEV_RES_IRQ_MAX];
```

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_resource.c

2

```
61     int
62     int
63 };
_____unchanged_portion_omitted_
```

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_scope.c

1

\*\*\*\*\*

5148 Fri Jan 17 07:31:27 2014

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_scope.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) 2009-2010, Intel Corporation.
23 * All rights reserved.
24 */

26 #include <sys/types.h>
27 #include <sys/atomic.h>
28 #include <sys/sunddi.h>
29 #include <sys/sunndi.h>
30 #include <acpica/include/acpi.h>
30 #include <sys/acpi/acpi.h>
31 #include <sys/acpica.h>
32 #include <sys/acpidev.h>
33 #include <sys/acpidev_impl.h>

35 static ACPI_STATUS acpidev_scope_probe(acpidev_walk_info_t *infop);
36 static acpidev_filter_result_t acpidev_scope_filter(acpidev_walk_info_t *infop,
37     char *devname, int maxlen);
38 static ACPI_STATUS acpidev_scope_init(acpidev_walk_info_t *infop);

40 /*
41 * Default class driver for ACPI scope objects.
42 * This class driver is used to handle predefined ACPI SCOPE objects
43 * under the ACPI root object, such as _PR_, _SB_ and _TZ_ etc.
44 * The default policy for ACPI SCOPE objects is SKIP.
45 */
46 acpidev_class_t acpidev_class_scope = {
47     0, /* adc_refcnt */
48     ACPIDEV_CLASS_REV1, /* adc_version */
49     ACPIDEV_CLASS_ID_SCOPE, /* adc_class_id */
50     "ACPI Scope", /* adc_class_name */
51     ACPIDEV_TYPE_SCOPE, /* adc_dev_type */
52     NULL, /* adc_private */
53     NULL, /* adc_pre_probe */
54     NULL, /* adc_post_probe */
55     acpidev_scope_probe, /* adc_probe */
56     acpidev_scope_filter, /* adc_filter */
57     acpidev_scope_init, /* adc_init */
58     NULL, /* adc_fini */
59 };
```

unchanged\_portion\_omitted

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_util.c

1

\*\*\*\*\*

24027 Fri Jan 17 07:31:27 2014

new/usr/src/uts/i86pc/io/acpi/acpidev/acpidev\_util.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 */

22 /*
23  * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25 */
26 /*
27  * Copyright (c) 2009-2010, Intel Corporation.
28  * All rights reserved.
29 */

31 #include <sys/types.h>
32 #include <sys/cmn_err.h>
33 #include <sys/note.h>
34 #include <sys/sysmacros.h>
35 #include <sys/sunddi.h>
36 #include <sys/sunndi.h>
37 #include <acpica/include/acpi.h>
37 #include <sys/acpi/acpi.h>
38 #include <sys/acpica.h>
39 #include <sys/acpidev.h>
40 #include <sys/acpidev_impl.h>
41 #include <util/sscanf.h>

43 /* Data structures used to extract the numeric unit address from string _UID. */
44 static acpidev_pseudo_uid_head_t acpidev_uid_heads[ACPIDEV_CLASS_ID_MAX];
45 static char *acpidev_uid_formats[] = {
46     "%u",
47 };
48
49 unchanged_portion_omitted
```

new/usr/src/uts/i86pc/io/acpi/acpinex/acpinex\_event.c

1

```
*****
21526 Fri Jan 17 07:31:27 2014
new/usr/src/uts/i86pc/io/acpi/acpinex/acpinex_event.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 /*
23  * Copyright 2011 Nexenta Systems, Inc. All rights reserved.
24  * Copyright (c) 2010, Intel Corporation.
25  * All rights reserved.
26 */
27
28 #include <sys/types.h>
29 #include <sys/atomic.h>
30 #include <sys/bitmap.h>
31 #include <sys/cmn_err.h>
32 #include <sys/note.h>
33 #include <sys/sunndi.h>
34 #include <sys/fastboot_impl.h>
35 #include <sys/sysevent.h>
36 #include <sys/sysevent/dr.h>
37 #include <sys/sysevent/eventdefs.h>
38 #include <acpica/include/acpi.h>
39 #include <sys/acpi/acpi.h>
40 #include <sys/acpica.h>
41 #include <sys/acpidev.h>
42 #include <sys/acpidev_dr.h>
43 #include <sys/acpinex.h>
44
45 int acpinex_event_support_remove = 0;
46
47 static volatile uint_t acpinex_dr_event_cnt = 0;
48 static ulong_t acpinex_object_type_mask[BT_BITOUL(ACPI_TYPE_NS_NODE_MAX + 1)];
49
50 /*
51  * Generate DR_REQ event to syseventd.
52  * Please refer to sys/sysevent/dr.h for message definition.
53  */
54 static int
55 acpinex_event_generate_event(dev_info_t *dip, ACPI_HANDLE hdl, int req,
56                             int event, char *objname)
57 {
58     int rv = 0;
59     sysevent_id_t eid;
60     sysevent_value_t evnt_val;
61     sysevent_attr_list_t *evnt_attr_list = NULL;
```

new/usr/src/uts/i86pc/io/acpi/acpinex/acpinex\_event.c

2

```
61     char *attach_pnt;
62     char event_type[32];
63
64     /* Add "attachment point" attribute. */
65     attach_pnt = kmem_zalloc(MAXPATHLEN, KM_SLEEP);
66     if (ACPI_FAILURE(acpidev_dr_get_attachment_point(hdl,
67                                                     attach_pnt, MAXPATHLEN))) {
68         cmn_err(CE_WARN,
69                 "!acpinex: failed to generate AP name for %s.", objname);
70         kmem_free(attach_pnt, MAXPATHLEN);
71         return (-1);
72     }
73     ASSERT(attach_pnt[0] != '\0');
74     evnt_val.value_type = SE_DATA_TYPE_STRING;
75     evnt_val.value.sv_string = attach_pnt;
76     rv = sysevent_add_attr(&evnt_attr_list, DR_AP_ID, &evnt_val, KM_SLEEP);
77     if (rv != 0) {
78         cmn_err(CE_WARN,
79                 "!acpinex: failed to add attr [%s] for %s event.",
80                 DR_AP_ID, EC_DR);
81         kmem_free(attach_pnt, MAXPATHLEN);
82         return (rv);
83     }
84
85     /* Add "request type" attribute. */
86     evnt_val.value_type = SE_DATA_TYPE_STRING;
87     evnt_val.value.sv_string = SE_REQ2STR(req);
88     rv = sysevent_add_attr(&evnt_attr_list, DR_REQ_TYPE, &evnt_val,
89                           KM_SLEEP);
90     if (rv != 0) {
91         cmn_err(CE_WARN,
92                 "!acpinex: failed to add attr [%s] for %s event.",
93                 DR_REQ_TYPE, EC_DR);
94         sysevent_free_attr(evnt_attr_list);
95         kmem_free(attach_pnt, MAXPATHLEN);
96         return (rv);
97     }
98
99     /* Add "acpi-event-type" attribute. */
100     switch (event) {
101     case ACPI_NOTIFY_BUS_CHECK:
102         (void) snprintf(event_type, sizeof (event_type),
103                         ACPIDEV_EVENT_TYPE_BUS_CHECK);
104         break;
105     case ACPI_NOTIFY_DEVICE_CHECK:
106         (void) snprintf(event_type, sizeof (event_type),
107                         ACPIDEV_EVENT_TYPE_DEVICE_CHECK);
108         break;
109     case ACPI_NOTIFY_DEVICE_CHECK_LIGHT:
110         (void) snprintf(event_type, sizeof (event_type),
111                         ACPIDEV_EVENT_TYPE_DEVICE_CHECK_LIGHT);
112         break;
113     case ACPI_NOTIFY_EJECT_REQUEST:
114         (void) snprintf(event_type, sizeof (event_type),
115                         ACPIDEV_EVENT_TYPE_EJECT_REQUEST);
116         break;
117     default:
118         cmn_err(CE_WARN,
119                 "!acpinex: unknown ACPI event type %d.", event);
120         sysevent_free_attr(evnt_attr_list);
121         kmem_free(attach_pnt, MAXPATHLEN);
122         return (-1);
123     }
124     evnt_val.value_type = SE_DATA_TYPE_STRING;
125     evnt_val.value.sv_string = event_type;
126     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     }
136
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     }
144
145     nvlist_free(evnt_attr_list);
146     kmem_free(attach_pnt, MAXPATHLEN);
147
148     return (rv);
149 }
```

unchanged portion omitted

new/usr/src/uts/i86pc/io/acpi/drmach\_acpi/drmach\_acpi.c

1

```
*****
66613 Fri Jan 17 07:31:28 2014
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 */
29
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/onttrap.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>
```

new/usr/src/uts/i86pc/io/acpi/drmach\_acpi/drmach\_acpi.c

2

```
62 #include <sys/sysevent.h>
63 #include <sys/sysevent/dr.h>
64 #include <sys/sysevent/eventdefs.h>
65 #include <acpica/include/acpi.h>
66 #include <sys/acpi/acpi.h>
67 #include <sys/acpica.h>
68 #include <sys/acpidev.h>
69 #include <sys/acpidev_rsc.h>
70 #include <sys/acpidev_dr.h>
71 #include <sys/dr.h>
72 #include <sys/dr_util.h>
73 #include <sys/drmach.h>
74 #include "drmach_acpi.h"
75
76 /* utility */
77 #define MBYTE (1048576ull)
78 #define ptob64(p) ((uint64_t)(p) << PAGESHIFT)
79 #define b64top(b) ((pgcnt_t)((b) >> PAGESHIFT))
80
81 static int drmach_init(void);
82 static void drmach_fini(void);
83 static int drmach_name2type_idx(char *);
84 static sbd_error_t *drmach_mem_update_lgrp(drmachid_t);
85
86 static void drmach_board_dispose(drmachid_t id);
87 static sbd_error_t *drmach_board_release(drmachid_t);
88 static sbd_error_t *drmach_board_status(drmachid_t, drmach_status_t *);
89
90 static void drmach_io_dispose(drmachid_t);
91 static sbd_error_t *drmach_io_release(drmachid_t);
92 static sbd_error_t *drmach_io_status(drmachid_t, drmach_status_t *);
93
94 static void drmach_cpu_dispose(drmachid_t);
95 static sbd_error_t *drmach_cpu_release(drmachid_t);
96 static sbd_error_t *drmach_cpu_status(drmachid_t, drmach_status_t *);
97
98 static void drmach_mem_dispose(drmachid_t);
99 static sbd_error_t *drmach_mem_release(drmachid_t);
100 static sbd_error_t *drmach_mem_status(drmachid_t, drmach_status_t *);
101
102 #ifdef DEBUG
103 int drmach_debug = 1; /* set to non-zero to enable debug messages */
104 #endif /* DEBUG */
105
106 drmach_domain_info_t drmach_domain;
107
108 static char *drmach_ie_fmt = "drmach_acpi.c %d";
109 static drmach_array_t *drmach_boards;
110
111 /* rwlock to protect drmach_boards. */
112 static krwlock_t drmach_boards_rwlock;
113
114 /* rwlock to block out CPR thread. */
115 static krwlock_t drmach_cpr_rwlock;
116
117 /* CPR callb id. */
118 static callb_id_t drmach_cpr_cid;
119
120 static struct {
121     const char *name;
122     const char *type;
123     sbd_error_t *(*new)(drmach_device_t *, drmachid_t *);
124 } drmach_name2type[] = {
125     /* unchanged portion omitted */
126 }
```

new/usr/src/uts/i86pc/io/acpi/drmach\_acpi/drmach\_acpi.h

1

```
*****
5136 Fri Jan 17 07:31:28 2014
new/usr/src/uts/i86pc/io/acpi/drmach_acpi/drmach_acpi.h
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 #ifndef _SYS_DRMACH ACPI_H
31 #define _SYS_DRMACH ACPI_H
32 #include <sys/types.h>
33 #include <sys/cmn_err.h>
34 #include <sys/param.h>
35 #include <sys/sunddi.h>
36 #include <acpica/include/acpi.h>
37 #include <sys/acpi/acpi.h>
38 #include <sys/acpidev.h>
39 #include <sys/drmach.h>

41 #ifdef __cplusplus
42 extern "C" {
43 #endif

45 #ifdef _KERNEL

47 /* Use ACPI handle as DRMACH handle on x86 systems. */
48 #define DRMACH_HANDLE ACPI_HANDLE

50 /* Macros to deal with object type. */
51 #define DRMACH_OBJ(id) ((drmach_common_t *)id)

53 #define DRMACH_NULL_ID(id) ((id) == 0)

55 #define DRMACH_IS_BOARD_ID(id) \
56 ((id != 0) && (DRMACH_OBJ(id)->isa == (void *)drmach_board_new))

58 #define DRMACH_IS_CPU_ID(id) \
59 ((id != 0) && (DRMACH_OBJ(id)->isa == (void *)drmach_cpu_new))
```

new/usr/src/uts/i86pc/io/acpi/drmach\_acpi/drmach\_acpi.h

2

```
61 #define DRMACH_IS_MEM_ID(id) \
62 ((id != 0) && (DRMACH_OBJ(id)->isa == (void *)drmach_mem_new))

64 #define DRMACH_IS_IO_ID(id) \
65 ((id != 0) && (DRMACH_OBJ(id)->isa == (void *)drmach_io_new))

67 #define DRMACH_IS_DEVICE_ID(id) \
68 ((id != 0) && \
69 (DRMACH_OBJ(id)->isa == (void *)drmach_cpu_new || \
70 DRMACH_OBJ(id)->isa == (void *)drmach_mem_new || \
71 DRMACH_OBJ(id)->isa == (void *)drmach_io_new))

73 #define DRMACH_IS_ID(id) \
74 ((id != 0) && \
75 (DRMACH_OBJ(id)->isa == (void *)drmach_board_new || \
76 DRMACH_OBJ(id)->isa == (void *)drmach_cpu_new || \
77 DRMACH_OBJ(id)->isa == (void *)drmach_mem_new || \
78 DRMACH_OBJ(id)->isa == (void *)drmach_io_new))

80 #define DRMACH_INTERNAL_ERROR() \
81 drerr_new(1, EX86_INTERNAL, drmach_ie_fmt, __LINE__)

83 #ifdef DEBUG
84 extern int drmach_debug;

86 #define DRMACH_PR if (drmach_debug) printf
87 #else
88 #define DRMACH_PR _NOTE(CONSTANTCONDITION) if (0) printf
89 #endif /* DEBUG */

91 typedef struct {
92     struct drmach_node *node;
93     void *data;
94     void *func;
95 } drmach_node_walk_args_t;
unchanged portion omitted
```

new/usr/src/uts/i86pc/io/amd\_iommu/amd\_iommu\_acpi.h

1

```
*****
7428 Fri Jan 17 07:31:29 2014
new/usr/src/uts/i86pc/io/amd_iommu/amd_iommu_acpi.h
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) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
23 */

25 #ifndef _AMD_IOMMU_ACPI_H
26 #define _AMD_IOMMU_ACPI_H

28 #ifdef __cplusplus
29 extern "C" {
30 #endif

32 #include <sys/sunddi.h>
33 #include <acpica/include/acpi.h>
34 #include <sys/acpi/acpi.h>
35 #include <sys/acpica.h>
36 #include <sys/amd_iommu.h>
37 #include "amd_iommu_impl.h"

38 #ifdef _KERNEL

40 #define IVRS_SIG          "IVRS"

42 /*
43  * IVINFO settings
44  */
45 #define AMD_IOMMU_ACPI_IVINFO_RSV1      (31 << 16 | 23)
46 #define AMD_IOMMU_ACPI_HT_ATSRSV       (22 << 16 | 22)
47 #define AMD_IOMMU_ACPI_VA_SIZE          (21 << 16 | 15)
48 #define AMD_IOMMU_ACPI_PA_SIZE          (14 << 16 | 8)
49 #define AMD_IOMMU_ACPI_IVINFO_RSV2     (7 << 16 | 0)

51 /*
52  * IVHD Device entry len field
53  */
54 #define AMD_IOMMU_ACPI_DEVENTRY_LEN     (7 << 16 | 6)

56 /*
57  * IVHD flag fields definition
58  */
59 #define AMD_IOMMU_ACPI_IVHD_FLAGS_RSV   (7 << 16 | 5)
60 #define AMD_IOMMU_ACPI_IVHD_FLAGS_IOTLBSUP (4 << 16 | 4)
```

new/usr/src/uts/i86pc/io/amd\_iommu/amd\_iommu\_acpi.h

2

```
61 #define AMD_IOMMU_ACPI_IVHD_FLAGS_ISOC      (3 << 16 | 3)
62 #define AMD_IOMMU_ACPI_IVHD_FLAGS_RESPASSPW (2 << 16 | 2)
63 #define AMD_IOMMU_ACPI_IVHD_FLAGS_PASSPW     (1 << 16 | 1)
64 #define AMD_IOMMU_ACPI_IVHD_FLAGS_HTTUNEN    (0 << 16 | 0)

66 /*
67  * IVHD IOMMU info fields
68  */
69 #define AMD_IOMMU_ACPI_IOMMU_INFO_RSV1      (15 << 16 | 13)
70 #define AMD_IOMMU_ACPI_IOMMU_INFO_UNITID    (12 << 16 | 8)
71 #define AMD_IOMMU_ACPI_IOMMU_INFO_RSV2      (7 << 16 | 5)
72 #define AMD_IOMMU_ACPI_IOMMU_INFO_MSINUM    (4 << 16 | 0)

74 /*
75  * IVHD deventry data settings
76  */
77 #define AMD_IOMMU_ACPI_LINT1PASS            (7 << 16 | 7)
78 #define AMD_IOMMU_ACPI_LINT0PASS            (6 << 16 | 6)
79 #define AMD_IOMMU_ACPI_SYSMGT              (5 << 16 | 4)
80 #define AMD_IOMMU_ACPI_DATRSV              (3 << 16 | 3)
81 #define AMD_IOMMU_ACPI_NMIPASS             (2 << 16 | 2)
82 #define AMD_IOMMU_ACPI_EXTINTPASS          (1 << 16 | 1)
83 #define AMD_IOMMU_ACPI_INITPASS            (0 << 16 | 0)

85 /*
86  * IVHD deventry extended data settings
87  */
88 #define AMD_IOMMU_ACPI_ATSDISABLED          (31 << 16 | 31)
89 #define AMD_IOMMU_ACPI_EXTDATRSV            (30 << 16 | 0)

91 /*
92  * IVMD flags fields settings
93  */
94 #define AMD_IOMMU_ACPI_IVMD_RSV             (7 << 16 | 4)
95 #define AMD_IOMMU_ACPI_IVMD_EXCL_RANGE      (3 << 16 | 3)
96 #define AMD_IOMMU_ACPI_IVMD_IW             (2 << 16 | 2)
97 #define AMD_IOMMU_ACPI_IVMD_IR             (1 << 16 | 1)
98 #define AMD_IOMMU_ACPI_IVMD_UNITY           (0 << 16 | 0)

100 #define AMD_IOMMU_ACPI_INFO_HASH_SZ         (256)

102 /*
103  * Deventry special device "variety"
104  */
105 #define AMD_IOMMU_ACPI_SPECIAL_APIC         0x1
106 #define AMD_IOMMU_ACPI_SPECIAL_HPET        0x2

108 typedef enum {
109     DEVENTRY_INVALID = 0,
110     DEVENTRY_ALL = 1,
111     DEVENTRY_SELECT,
112     DEVENTRY_RANGE,
113     DEVENTRY_RANGE_END,
114     DEVENTRY_ALIAS_SELECT,
115     DEVENTRY_ALIAS_RANGE,
116     DEVENTRY_EXTENDED_SELECT,
117     DEVENTRY_EXTENDED_RANGE,
118     DEVENTRY_SPECIAL_DEVICE
119 } ivhd_deventry_type_t;
unchanged_portion_omitted
```

new/usr/src/uts/i86pc/io/amd\_iommu/amd\_iommu\_page\_tables.c

1

\*\*\*\*\*

45674 Fri Jan 17 07:31:29 2014

new/usr/src/uts/i86pc/io/amd\_iommu/amd\_iommu\_page\_tables.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 */

22 /*
23  * Copyright (c) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
24 */

26 #include <sys/sunddi.h>
27 #include <sys/sunndi.h>
28 #include <acpica/include/acpi.h>
29 #include <sys/acpi/acpi.h>
30 #include <sys/acpica.h>
31 #include <sys/amd_iommu.h>
32 #include <sys/bootconf.h>
33 #include <sys/sysmacros.h>
34 #include <sys/ddidmareg.h>

35 #include "amd_iommu_impl.h"
36 #include "amd_iommu_acpi.h"
37 #include "amd_iommu_page_tables.h"

39 ddi_dma_attr_t amd_iommu_phtable_dma_attr = {
40     DMA_ATTR_V0,
41     0U, /* dma_attr_addr_lo */
42     0xffffffffffffffffULL, /* dma_attr_addr_hi */
43     0xffffffffU, /* dma_attr_count_max */
44     (uint64_t)4096, /* dma_attr_align */
45     1, /* dma_attr_burstsizes */
46     64, /* dma_attr_minxfer */
47     0xffffffffU, /* dma_attr_maxxfer */
48     0xffffffffU, /* dma_attr_seg */
49     1, /* dma_attr_sgllen, variable */
50     64, /* dma_attr_granular */
51     0 /* dma_attr_flags */
52 };
53
54 _____
55 unchanged portion omitted
56 _____
```

new/usr/src/uts/i86pc/io/apix/apix.c

1

```
*****
65839 Fri Jan 17 07:31:30 2014
new/usr/src/uts/i86pc/io/apix/apix.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 */

22 /*
23  * Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
24 */
25 /*
26  * Copyright (c) 2010, Intel Corporation.
27  * All rights reserved.
28 */
29 /*
30  * Copyright (c) 2013, Joyent, Inc. All rights reserved.
31 */

33 /*
34  * To understand how the apix module interacts with the interrupt subsystem read
35  * the theory statement in uts/i86pc/os/intr.c.
36 */

38 /*
39  * PSMT 1.1 extensions are supported only in 2.6 and later versions.
40  * PSMT 1.2 extensions are supported only in 2.7 and later versions.
41  * PSMT 1.3 and 1.4 extensions are supported in Solaris 10.
42  * PSMT 1.5 extensions are supported in Solaris Nevada.
43  * PSMT 1.6 extensions are supported in Solaris Nevada.
44  * PSMT 1.7 extensions are supported in Solaris Nevada.
45 */
46 #define PSMT_1_7

48 #include <sys/processor.h>
49 #include <sys/time.h>
50 #include <sys/psm.h>
51 #include <sys/smp_impldefs.h>
52 #include <sys/cram.h>
53 #include <acpica/include/acpi.h>
54 #include <sys/acpi/acpi.h>
55 #include <sys/psm_common.h>
56 #include <sys/pit.h>
57 #include <sys/ddi.h>
58 #include <sys/sunddi.h>
59 #include <sys/ddi_impldefs.h>
60 #include <sys/pci.h>
```

new/usr/src/uts/i86pc/io/apix/apix.c

2

```
61 #include <sys/promif.h>
62 #include <sys/x86_archext.h>
63 #include <sys/cpc_impl.h>
64 #include <sys/uadmin.h>
65 #include <sys/panic.h>
66 #include <sys/debug.h>
67 #include <sys/archsystem.h>
68 #include <sys/trap.h>
69 #include <sys/machsystem.h>
70 #include <sys/sysmacros.h>
71 #include <sys/cpuvar.h>
72 #include <sys/rm_platter.h>
73 #include <sys/privregs.h>
74 #include <sys/note.h>
75 #include <sys/pci_intr_lib.h>
76 #include <sys/spl.h>
77 #include <sys/clock.h>
78 #include <sys/cyclic.h>
79 #include <sys/dditypes.h>
80 #include <sys/sunddi.h>
81 #include <sys/x_call.h>
82 #include <sys/reboot.h>
83 #include <sys/mach_intr.h>
84 #include <sys/apix.h>
85 #include <sys/apix_irm_impl.h>

87 static int apix_probe();
88 static void apix_init();
89 static void apix_picinit(void);
90 static int apix_intr_enter(int, int *);
91 static void apix_intr_exit(int, int);
92 static void apix_setspl(int);
93 static int apix_disable_intr(processorid_t);
94 static void apix_enable_intr(processorid_t);
95 static int apix_get_clkvect(int);
96 static int apix_get_ipivect(int, int);
97 static void apix_post_cyclic_setup(void *);
98 static int apix_post_cpu_start();
99 static int apix_intr_ops(dev_info_t *, ddi_intr_handle_impl_t *,
100     psm_intr_op_t, int *);

102 /*
103  * Helper functions for apix_intr_ops()
104 */
105 static void apix_redistribute_compute(void);
106 static int apix_get_pending(apix_vector_t *);
107 static apix_vector_t *apix_get_req_vector(ddi_intr_handle_impl_t *, ushort_t);
108 static int apix_get_intr_info(ddi_intr_handle_impl_t *, apic_get_intr_t *);
109 static char *apix_get_apic_type(void);
110 static int apix_intx_get_pending(int);
111 static void apix_intx_set_mask(int irqno);
112 static void apix_intx_clear_mask(int irqno);
113 static int apix_intx_get_shared(int irqno);
114 static void apix_intx_set_shared(int irqno, int delta);
115 static apix_vector_t *apix_intx_xlate_vector(dev_info_t *, int,
116     struct intrspec *);
117 static int apix_intx_alloc_vector(dev_info_t *, int, struct intrspec *);

119 extern int apic_clkinit(int);

121 /* IRM initialization for APIX PSM module */
122 extern void apix_irm_init(void);

124 extern int irm_enable;

126 /*
```

```
127 *      Local static data
128 */
129 static struct psm_ops apix_ops = {
130     apix_probe,
131
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 */
143
144     apic_set_idlecpu,
145     apic_unset_idlecpu,
146
147     apic_clkinit,
148     apix_get_clkvect,
149     NULL, /* psm_hrttimeinit */
150     apic_gethrtime,
151
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,
158
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 };
171
172 _____unchanged_portion_omitted_____
```

new/usr/src/uts/i86pc/io/apix/apix\_utils.c

1

```
*****
48694 Fri Jan 17 07:31:30 2014
new/usr/src/uts/i86pc/io/apix/apix_utils.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 /*
23  * Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
24 */
25 /*
26  * Copyright (c) 2010, Intel Corporation.
27  * All rights reserved.
28 */
29 /*
30  * Copyright 2013 Nexenta Systems, Inc. All rights reserved.
31  * Copyright 2013 Pluribus Networks, Inc.
32 */
33
34 #include <sys/processor.h>
35 #include <sys/time.h>
36 #include <sys/psm.h>
37 #include <sys/smp_impldefs.h>
38 #include <sys/cram.h>
39 #include <acpica/include/acpi.h>
39 #include <sys/acpi/acpi.h>
40 #include <sys/acpica.h>
41 #include <sys/psm_common.h>
42 #include <sys/pit.h>
43 #include <sys/ddi.h>
44 #include <sys/sunddi.h>
45 #include <sys/ddi_impldefs.h>
46 #include <sys/pci.h>
47 #include <sys/promif.h>
48 #include <sys/x86_archext.h>
49 #include <sys/cpc_impl.h>
50 #include <sys/uadmin.h>
51 #include <sys/panic.h>
52 #include <sys/debug.h>
53 #include <sys/archsystem.h>
54 #include <sys/trap.h>
55 #include <sys/machsystem.h>
56 #include <sys/sysmacros.h>
57 #include <sys/cpuvar.h>
58 #include <sys/rm_platter.h>
59 #include <sys/privregs.h>
60 #include <sys/note.h>
```

new/usr/src/uts/i86pc/io/apix/apix\_utils.c

2

```
61 #include <sys/pci_intr_lib.h>
62 #include <sys/spl.h>
63 #include <sys/clock.h>
64 #include <sys/dditypes.h>
65 #include <sys/sunddi.h>
66 #include <sys/x_call.h>
67 #include <sys/reboot.h>
68 #include <sys/apix.h>
69
70 static int apix_get_avail_vector_oncpu(uint32_t, int, int);
71 static apix_vector_t *apix_init_vector(processorid_t, uchar_t);
72 static void apix_cleanup_vector(apix_vector_t *);
73 static void apix_insert_av(apix_vector_t *, void *, avfunc, caddr_t, caddr_t,
74     uint64_t *, int, dev_info_t *);
75 static void apix_remove_av(apix_vector_t *, struct autovec *);
76 static void apix_clear_dev_map(dev_info_t *, int, int);
77 static boolean_t apix_is_cpu_enabled(processorid_t);
78 static void apix_wait_till_seen(processorid_t, int);
79
80 #define GET_INTR_INUM(ihdlp) \
81     (((ihdlp) != NULL) ? ((ddi_intr_handle_impl_t *) (ihdlp))->ih_inum : 0)
82
83 apix_rebind_info_t apix_rebindinfo = {0, 0, 0, NULL, 0, NULL};
84
85 /*
86  * Allocate IPI
87  *
88  * Return vector number or 0 on error
89 */
90 uchar_t
91 apix_alloc_ipi(int ipl)
92 {
93     apix_vector_t *vecp;
94     uchar_t vector;
95     int cpun;
96     int nproc;
97
98     APIX_ENTER_CPU_LOCK(0);
99
100    vector = apix_get_avail_vector_oncpu(0, APIX_IPI_MIN, APIX_IPI_MAX);
101    if (vector == 0) {
102        APIX_LEAVE_CPU_LOCK(0);
103        cmn_err(CE_WARN, "apix: no available IPI\n");
104        apic_error |= APIC_ERR_GET_IPIVECT_FAIL;
105        return (0);
106    }
107
108    nproc = max(apic_nproc, apic_max_nproc);
109    for (cpun = 0; cpun < nproc; cpun++) {
110        vecp = xv_vector(cpun, vector);
111        if (vecp == NULL) {
112            vecp = kmem_zalloc(sizeof (apix_vector_t), KM_NOSLEEP);
113            if (vecp == NULL) {
114                cmn_err(CE_WARN, "apix: No memory for ipi");
115                goto fail;
116            }
117            xv_vector(cpun, vector) = vecp;
118        }
119        vecp->v_state = APIX_STATE_ALLOCED;
120        vecp->v_type = APIX_TYPE_IPI;
121        vecp->v_cpuid = vecp->v_bound_cpuid = cpun;
122        vecp->v_vector = vector;
123        vecp->v_pri = ipl;
124    }
125    APIX_LEAVE_CPU_LOCK(0);
126    return (vector);
127 }
```



new/usr/src/uts/i86pc/io/apix/apix\_utils.c

3

```
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
```

new/usr/src/uts/i86pc/io/immu\_dmar.c

1

```
*****
27729 Fri Jan 17 07:31:31 2014
new/usr/src/uts/i86pc/io/immu_dmar.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) 2010, Oracle and/or its affiliates. All rights reserved.
23 */

25 /*
26 * Copyright (c) 2009, Intel Corporation.
27 * All rights reserved.
28 */

31 #include <sys/debug.h>
32 #include <sys/sysmacros.h>
33 #include <sys/types.h>
34 #include <sys/kmem.h>
35 #include <sys/sunddi.h>
36 #include <sys/list.h>
37 #include <sys/pci.h>
38 #include <sys/pci_cfgspace.h>
39 #include <sys/pci_impl.h>
40 #include <sys/sunndi.h>
41 #include <sys/ksynch.h>
42 #include <sys/cmn_err.h>
43 #include <sys/bootconf.h>
44 #include <sys/int_fmtio.h>
45 #include <sys/smbios.h>
46 #include <sys/apic.h>
47 #include <acpica/include/acpi.h>
47 #include <sys/acpi/acpi.h>
48 #include <sys/acpica.h>
49 #include <sys/immu.h>
50 #include <sys/smp_impldefs.h>

52 static void dmar_table_destroy(dmar_table_t *tbl);

54 /*
55  * internal global variables
56  */
57 static char *dmar_raw; /* raw DMAR ACPI table */
58 static dmar_table_t *dmar_table; /* converted form of DMAR table */

60 /*
```

new/usr/src/uts/i86pc/io/immu\_dmar.c

2

```
61 * global variables exported outside this file
62 */
63 boolean_t dmar_print = B_FALSE;
64 kmutex_t ioapic_drhd_lock;
65 list_t ioapic_drhd_list;

67 /* ##### */

69 /*
70  * helper functions to read the "raw" DMAR table
71  */

73 static uint8_t
74 get_uint8(char *cp)
75 {
76     uint8_t val = *((uint8_t *)cp);
77     return (val);
78 }

unchanged_portion_omitted
```

new/usr/src/uts/i86pc/io/immu\_dvma.c

1

```
*****
78638 Fri Jan 17 07:31:31 2014
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/archsystem.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_CONFIG_PADDR(d, p) \
```

new/usr/src/uts/i86pc/io/immu\_dvma.c

2

```
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
```

new/usr/src/uts/i86pc/io/mp\_platform\_common.c

1

```
*****
67080 Fri Jan 17 07:31:32 2014
new/usr/src/uts/i86pc/io/mp_platform_common.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 */
24 /*
25 * Copyright (c) 2010, Intel Corporation.
26 * All rights reserved.
27 */

29 /*
30 * PSMI 1.1 extensions are supported only in 2.6 and later versions.
31 * PSMI 1.2 extensions are supported only in 2.7 and later versions.
32 * PSMI 1.3 and 1.4 extensions are supported in Solaris 10.
33 * PSMI 1.5 extensions are supported in Solaris Nevada.
34 * PSMI 1.6 extensions are supported in Solaris Nevada.
35 * PSMI 1.7 extensions are supported in Solaris Nevada.
36 */
37 #define PSMI_1_7

39 #include <sys/processor.h>
40 #include <sys/time.h>
41 #include <sys/psm.h>
42 #include <sys/smp_impldefs.h>
43 #include <sys/cram.h>
44 #include <acpica/include/acpi.h>
44 #include <sys/acpi/acpi.h>
45 #include <sys/acpica.h>
46 #include <sys/psm_common.h>
47 #include <sys/apic.h>
48 #include <sys/apic_timer.h>
49 #include <sys/pit.h>
50 #include <sys/ddi.h>
51 #include <sys/sunddi.h>
52 #include <sys/ddi_impldefs.h>
53 #include <sys/pci.h>
54 #include <sys/promif.h>
55 #include <sys/x86_archext.h>
56 #include <sys/cpc_impl.h>
57 #include <sys/uadmin.h>
58 #include <sys/panic.h>
59 #include <sys/debug.h>
60 #include <sys/archsystem.h>
```

new/usr/src/uts/i86pc/io/mp\_platform\_common.c

2

```
61 #include <sys/trap.h>
62 #include <sys/machsystem.h>
63 #include <sys/cpuvar.h>
64 #include <sys/rm_platter.h>
65 #include <sys/privregs.h>
66 #include <sys/cyclic.h>
67 #include <sys/note.h>
68 #include <sys/pci_intr_lib.h>
69 #include <sys/sunddi.h>
70 #if !defined(__xpv)
71 #include <sys/hpet.h>
72 #include <sys/clock.h>
73 #endif

75 /*
76  *      Local Function Prototypes
77 */
78 static int apic_handle_defconf();
79 static int apic_parse_mpct(caddr_t mpct, int bypass);
80 static struct apic_mpfps_hdr *apic_find_fps_sig(caddr_t fptr, int size);
81 static int apic_checksum(caddr_t bptr, int len);
82 static int apic_find_bus_type(char *bus);
83 static int apic_find_bus(int busid);
84 static struct apic_io_intr *apic_find_io_intr(int irqno);
85 static int apic_find_free_irq(int start, int end);
86 struct apic_io_intr *apic_find_io_intr_w_busid(int irqno, int busid);
87 static void apic_set_pwroff_method_from_mpcnfhdr(struct apic_mp_cnf_hdr *hdrp);
88 static void apic_free_apic_cpus(void);
89 static boolean_t apic_is_ioapic_AMD_813x(uint32_t physaddr);
90 static int apic_acpi_enter_apicmode(void);

92 int apic_handle_pci_pci_bridge(dev_info_t *idip, int child_devno,
93     int child_ipin, struct apic_io_intr **intrp);
94 int apic_find_bus_id(int bustype);
95 int apic_find_intin(uchar_t ioapic, uchar_t intin);
96 void apic_record_rdt_entry(apic_irq_t *irqptr, int irq);

98 int apic_debug_mps_id = 0;      /* 1 - print MPS ID strings */

100 /* ACPI SCI interrupt configuration; -1 if SCI not used */
101 int apic_sci_vect = -1;
102 iflag_t apic_sci_flags;

104 #if !defined(__xpv)
105 /* ACPI HPET interrupt configuration; -1 if HPET not used */
106 int apic_hpet_vect = -1;
107 iflag_t apic_hpet_flags;
108 #endif

110 /*
111  * psm name pointer
112 */
113 char *psm_name;

115 /* ACPI support routines */
116 static int acpi_probe(char *);
117 static int apic_acpi_irq_configure(acpi_psm_lnk_t *acpipsm_lnk, dev_info_t *dip,
118     int *pci_irqp, iflag_t *intr_flags);

120 int apic_acpi_translate_pci_irq(dev_info_t *dip, int busid, int devid,
121     int ipin, int *pci_irqp, iflag_t *intr_flags);
122 uchar_t apic_find_ioapic(int irq);
123 static int acpi_intr_compatible(iflag_t iflag1, iflag_t iflag2);

125 /* Max wait time (in repetitions) for flags to clear in an RDT entry. */
126 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 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 millisecc */

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_intrap_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_
```

new/usr/src/uts/i86pc/io/mp\_platform\_misc.c

1

```
*****
63988 Fri Jan 17 07:31:33 2014
new/usr/src/uts/i86pc/io/mp_platform_misc.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) 2010, Oracle and/or its affiliates. All rights reserved.
23 */
24 /*
25 * Copyright (c) 2010, Intel Corporation.
26 * All rights reserved.
27 */
28
29 /*
30 * PSMI 1.1 extensions are supported only in 2.6 and later versions.
31 * PSMI 1.2 extensions are supported only in 2.7 and later versions.
32 * PSMI 1.3 and 1.4 extensions are supported in Solaris 10.
33 * PSMI 1.5 extensions are supported in Solaris Nevada.
34 * PSMI 1.6 extensions are supported in Solaris Nevada.
35 * PSMI 1.7 extensions are supported in Solaris Nevada.
36 */
37 #define PSMI_1_7
38
39 #include <sys/processor.h>
40 #include <sys/time.h>
41 #include <sys/psm.h>
42 #include <sys/smp_impldefs.h>
43 #include <sys/inttypes.h>
44 #include <sys/cram.h>
45 #include <acpica/include/acpi.h>
46 #include <sys/acpi/acpi.h>
47 #include <sys/psm_common.h>
48 #include <sys/apic.h>
49 #include <sys/apic_common.h>
50 #include <sys/pit.h>
51 #include <sys/ddi.h>
52 #include <sys/sunddi.h>
53 #include <sys/ddi_impldefs.h>
54 #include <sys/pci.h>
55 #include <sys/promif.h>
56 #include <sys/x86_archext.h>
57 #include <sys/cpc_impl.h>
58 #include <sys/uadmin.h>
59 #include <sys/panic.h>
60 #include <sys/debug.h>
```

new/usr/src/uts/i86pc/io/mp\_platform\_misc.c

2

```
61 #include <sys/archsystem.h>
62 #include <sys/trap.h>
63 #include <sys/machsystem.h>
64 #include <sys/cpuvar.h>
65 #include <sys/rm_platter.h>
66 #include <sys/privregs.h>
67 #include <sys/cyclic.h>
68 #include <sys/note.h>
69 #include <sys/pci_intr_lib.h>
70 #include <sys/sunndi.h>
71 #include <sys/hpet.h>
72 #include <sys/clock.h>
73
74 /*
75 * Part of mp_platform_common.c that's used only by pcplusmp & xpv_psm
76 * but not apix.
77 * These functions may be moved to xpv_psm later when apix and pcplusmp
78 * are merged together
79 */
80
81 /*
82 *      Local Function Prototypes
83 */
84 static void apic_mark_vector(uchar_t oldvector, uchar_t newvector);
85 static void apic_xlate_vector_free_timeout_handler(void *arg);
86 static int apic_check_stuck_interrupt(apic_irq_t *irq_ptr, int old_bind_cpu,
87      int new_bind_cpu, int apicindex, int intin_no, int which_irq,
88      struct ioapic_reprogram_data *drep);
89 static int apic_setup_irq_table(dev_info_t *dip, int irqno,
90      struct apic_io_intr *intrp, struct intrspec *ispec, iflag_t *intr_flagp,
91      int type);
92 static void apic_try_deferred_reprogram(int ipl, int vect);
93 static void delete_defer_repro_ent(int which_irq);
94 static void apic_ioapic_wait_pending_clear(int ioapicindex,
95      int intin_no);
96
97 extern int apic_acpi_translate_pci_irq(dev_info_t *dip, int busid, int devid,
98      int ipin, int *pci_irqp, iflag_t *intr_flagp);
99 extern int apic_handle_pci_pci_bridge(dev_info_t *idip, int child_devno,
100      int child_ipin, struct apic_io_intr **intrp);
101 extern uchar_t apic_find_ioapic(int irq);
102 extern struct apic_io_intr *apic_find_io_intr_w_busid(int irqno, int busid);
103 extern int apic_find_bus_id(int bustype);
104 extern int apic_find_intin(uchar_t ioapic, uchar_t intin);
105 extern void apic_record_rdt_entry(apic_irq_t *irqptr, int irq);
106
107 extern int apic_sci_vect;
108 extern iflag_t apic_sci_flags;
109 /* ACPI HPET interrupt configuration; -1 if HPET not used */
110 extern int apic_hpet_vect;
111 extern iflag_t apic_hpet_flags;
112 extern int apic_intr_policy;
113 extern char *psm_name;
114
115 /*
116 * number of bits per byte, from <sys/param.h>
117 */
118 #define UCHAR_MAX      UINT8_MAX
119
120 /* Max wait time (in repetitions) for flags to clear in an RDT entry. */
121 extern int apic_max_reps_clear_pending;
122
123 /* The irq # is implicit in the array index: */
124 struct ioapic_reprogram_data apic_reprogram_info[APIC_MAX_VECTOR+1];
125 /*
126 * 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);

```



new/usr/src/uts/i86pc/io/mp\_platform\_misc.c

5

259 }

unchanged\_portion\_omitted

new/usr/src/uts/i86pc/io/pciex/npe\_misc.c

1

```
*****
11718 Fri Jan 17 07:31:33 2014
new/usr/src/uts/i86pc/io/pciex/npe_misc.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) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
24  * Copyright 2013 PALO, Richard. All rights reserved.
25  */

27 /*
28  *      Library file that has miscellaneous support for npe(7d)
29  */

31 #include <sys/conf.h>
32 #include <sys/pci.h>
33 #include <sys/sunndi.h>
34 #include <acpica/include/acpi.h>
35 #include <sys/acpi/acpi.h>
36 #include <sys/acpi/acpi_pci.h>
37 #include <sys/acpica.h>
38 #include <sys/pci_cap.h>
39 #include <sys/pcie_impl.h>
40 #include <sys/x86_archext.h>
41 #include <io/pciex/pcie_nvidia.h>
42 #include <io/pciex/pcie_nb5000.h>
43 #include <sys/pci_cfgacc_x86.h>
44 #include <sys/cpuvar.h>

44 /*
45  * Prototype declaration
46  */
47 void    npe_query_acpi_mcfg(dev_info_t *dip);
48 void    npe_ck804_fix_aer_ptr(ddi_acc_handle_t cfg_hdl);
49 int     npe_disable_empty_bridges_workaround(dev_info_t *child);
50 void    npe_nvidia_error_workaround(ddi_acc_handle_t cfg_hdl);
51 void    npe_intel_error_workaround(ddi_acc_handle_t cfg_hdl);
52 boolean_t npe_is_child_pci(dev_info_t *dip);
53 int     npe_enable_htmsi(ddi_acc_handle_t cfg_hdl);
54 void    npe_enable_htmsi_children(dev_info_t *dip);

56 int     npe_enable_htmsi_flag = 1;

58 /*
```

new/usr/src/uts/i86pc/io/pciex/npe\_misc.c

2

```
59  * Default ecfga base address
60  */
61 int64_t npe_default_ecfga_base = 0xE0000000;

63 extern uint32_t npe_aer_uce_mask;

65 /*
66  * Query the MCFG table using ACPI. If MCFG is found, setup the
67  * 'ecfg' property accordingly. Otherwise, set the values
68  * to the default values.
69  */
70 void
71 npe_query_acpi_mcfg(dev_info_t *dip)
72 {
73     ACPI_TABLE_HEADER *mcfgp;
74     ACPI_MCFG_ALLOCATION *cfg_baap;
75     MCFG_TABLE *mcfgrp;
76     CFG_BASE_ADDR_ALLOC *cfg_baap;
77     char *cfg_baa_endp;
78     int64_t ecfginfo[4];
79     int ecfg_found = 0;

81     /* Query the MCFG table using ACPI */
82     if (AcpiGetTable(ACPI_SIG_MCFG, 1, &mcfgrp) == AE_OK) {
83         if (AcpiGetTable(ACPI_SIG_MCFG, 1,
84             (ACPI_TABLE_HEADER **)&mcfgrp) == AE_OK) {

86             cfg_baap = (ACPI_MCFG_ALLOCATION *)((ACPI_TABLE_MCFG *)mcfgrp + 1);
87             cfg_baap = (CFG_BASE_ADDR_ALLOC *)mcfgrp->CfgBaseAddrAllocList;
88             cfg_baa_endp = ((char *)mcfgrp) + mcfgrp->Length;

89             while ((char *)cfg_baap < cfg_baa_endp) {
90                 if (cfg_baap->Address != (uint64_t)0 &&
91                     cfg_baap->PciSegment == 0) {
92                     if (cfg_baap->base_addr != (uint64_t)0 &&
93                         cfg_baap->segment == 0) {
94                         /*
95                          * Set up the 'ecfg' property to hold
96                          * base_addr, segment, and first/last bus.
97                          * We only do the first entry that maps
98                          * segment 0; nonzero segments are not yet
99                          * known, or handled. If they appear,
100                          * we'll need to figure out which bus node
101                          * should have which entry by examining the
102                          * ACPI_SEG method on each bus node.
103                          */
104                         ecfginfo[0] = cfg_baap->Address;
105                         ecfginfo[1] = cfg_baap->PciSegment;
106                         ecfginfo[2] = cfg_baap->StartBusNumber;
107                         ecfginfo[3] = cfg_baap->EndBusNumber;
108                         ecfginfo[0] = cfg_baap->base_addr;
109                         ecfginfo[1] = cfg_baap->segment;
110                         ecfginfo[2] = cfg_baap->start_bno;
111                         ecfginfo[3] = cfg_baap->end_bno;
112                         (void) ndi_prop_update_int64_array(
113                             DDI_DEV_T_NONE, dip, "ecfg",
114                             ecfginfo, 4);
115                         ecfg_found = 1;
116                         break;
117                     }
118                     cfg_baap++;
119                 }
120             }
121         }
122     }
123     if (ecfg_found)
124         return;
125     /*
```

new/usr/src/uts/i86pc/io/pciex/npe\_misc.c

3

```
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

new/usr/src/uts/i86pc/io/pcplusmp/apic.c

1

```
*****
36120 Fri Jan 17 07:31:34 2014
new/usr/src/uts/i86pc/io/pcplusmp/apic.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 */

22 /*
23  * Copyright (c) 1993, 2010, Oracle and/or its affiliates. All rights reserved.
24 */
25 /*
26  * Copyright (c) 2010, Intel Corporation.
27  * All rights reserved.
28 */
29 /*
30  * Copyright (c) 2013, Joyent, Inc. All rights reserved.
31 */

33 /*
34  * To understand how the pcplusmp module interacts with the interrupt subsystem
35  * read the theory statement in uts/i86pc/os/intr.c.
36 */

38 /*
39  * PSMI 1.1 extensions are supported only in 2.6 and later versions.
40  * PSMI 1.2 extensions are supported only in 2.7 and later versions.
41  * PSMI 1.3 and 1.4 extensions are supported in Solaris 10.
42  * PSMI 1.5 extensions are supported in Solaris Nevada.
43  * PSMI 1.6 extensions are supported in Solaris Nevada.
44  * PSMI 1.7 extensions are supported in Solaris Nevada.
45 */
46 #define PSMI_1_7

48 #include <sys/processor.h>
49 #include <sys/time.h>
50 #include <sys/psm.h>
51 #include <sys/smp_impldefs.h>
52 #include <sys/cram.h>
53 #include <acpica/include/acpi.h>
54 #include <sys/acpi/acpi.h>
55 #include <sys/acpica.h>
56 #include <sys/psm_common.h>
57 #include <sys/apic.h>
58 #include <sys/pit.h>
59 #include <sys/sunddi.h>
60 #include <sys/ddi_impldefs.h>
```

new/usr/src/uts/i86pc/io/pcplusmp/apic.c

2

```
61 #include <sys/pci.h>
62 #include <sys/promif.h>
63 #include <sys/x86_archext.h>
64 #include <sys/cpc_impl.h>
65 #include <sys/uadmin.h>
66 #include <sys/panic.h>
67 #include <sys/debug.h>
68 #include <sys/archsystem.h>
69 #include <sys/trap.h>
70 #include <sys/machsystem.h>
71 #include <sys/sysmacros.h>
72 #include <sys/cpuvar.h>
73 #include <sys/rm_platter.h>
74 #include <sys/privregs.h>
75 #include <sys/note.h>
76 #include <sys/pci_intr_lib.h>
77 #include <sys/spl.h>
78 #include <sys/clock.h>
79 #include <sys/cyclic.h>
80 #include <sys/dditypes.h>
81 #include <sys/sunddi.h>
82 #include <sys/x_call.h>
83 #include <sys/reboot.h>
84 #include <sys/hpet.h>
85 #include <sys/apic_common.h>
86 #include <sys/apic_timer.h>

88 /*
89  * Local Function Prototypes
90 */
91 static void apic_init_intr(void);

93 /*
94  * standard MP entries
95 */
96 static int apic_probe(void);
97 static int apic_getclkirq(int ipl);
98 static void apic_init(void);
99 static void apic_picinit(void);
100 static int apic_post_cpu_start(void);
101 static int apic_intr_enter(int ipl, int *vect);
102 static void apic_setspl(int ipl);
103 static void x2apic_setspl(int ipl);
104 static int apic_addspl(int ipl, int vector, int min_ipl, int max_ipl);
105 static int apic_delspl(int ipl, int vector, int min_ipl, int max_ipl);
106 static int apic_disable_intr(processorid_t cpun);
107 static void apic_enable_intr(processorid_t cpun);
108 static int apic_get_ipivect(int ipl, int type);
109 static void apic_post_cyclic_setup(void *arg);

111 /*
112  * The following vector assignments influence the value of iptopri and
113  * vectortoip1. Note that vectors 0 - 0xf are not used. We can program
114  * idle to 0 and IPL 0 to 0xf to differentiate idle in case
115  * we care to do so in future. Note some IPLs which are rarely used
116  * will share the vector ranges and heavily used IPLs (5 and 6) have
117  * a wide range.
118 */
119 * This array is used to initialize apic_ipls[] (in apic_init()).
120 *
121 * IPL Vector range. as passed to intr_enter
122 * 0 none.
123 * 1,2,3 0x20-0x2f 0x0-0xf
124 * 4 0x30-0x3f 0x10-0x1f
125 * 5 0x40-0x5f 0x20-0x3f
126 * 6 0x60-0x7f 0x40-0x5f
```

new/usr/src/uts/i86pc/io/pcplusmp/apic.c

3

```
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
```

new/usr/src/uts/i86pc/io/pcplusmp/apic\_common.c

1

```
*****
44511 Fri Jan 17 07:31:34 2014
new/usr/src/uts/i86pc/io/pcplusmp/apic_common.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 */

22 /*
23  * Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
24 */
25 /*
26  * Copyright (c) 2013, Joyent, Inc. All rights reserved.
27 */

29 /*
30  * PSMI 1.1 extensions are supported only in 2.6 and later versions.
31  * PSMI 1.2 extensions are supported only in 2.7 and later versions.
32  * PSMI 1.3 and 1.4 extensions are supported in Solaris 10.
33  * PSMI 1.5 extensions are supported in Solaris Nevada.
34  * PSMI 1.6 extensions are supported in Solaris Nevada.
35  * PSMI 1.7 extensions are supported in Solaris Nevada.
36 */
37 #define PSMI_1_7

39 #include <sys/processor.h>
40 #include <sys/time.h>
41 #include <sys/psm.h>
42 #include <sys/smp_impldefs.h>
43 #include <sys/cram.h>
44 #include <acpica/include/acpi.h>
44 #include <sys/acpi/acpi.h>
45 #include <sys/acpica.h>
46 #include <sys/psm_common.h>
47 #include <sys/apic.h>
48 #include <sys/pit.h>
49 #include <sys/ddi.h>
50 #include <sys/sunddi.h>
51 #include <sys/ddi_impldefs.h>
52 #include <sys/pci.h>
53 #include <sys/promif.h>
54 #include <sys/x86_archext.h>
55 #include <sys/cpc_impl.h>
56 #include <sys/uadmin.h>
57 #include <sys/panic.h>
58 #include <sys/debug.h>
59 #include <sys/archsystem.h>
60 #include <sys/trap.h>
```

new/usr/src/uts/i86pc/io/pcplusmp/apic\_common.c

2

```
61 #include <sys/machsystem.h>
62 #include <sys/symmacros.h>
63 #include <sys/cpuvar.h>
64 #include <sys/rm_platter.h>
65 #include <sys/privregs.h>
66 #include <sys/note.h>
67 #include <sys/pci_intr_lib.h>
68 #include <sys/spl.h>
69 #include <sys/clock.h>
70 #include <sys/dditypes.h>
71 #include <sys/sunddi.h>
72 #include <sys/x_call.h>
73 #include <sys/reboot.h>
74 #include <sys/hpet.h>
75 #include <sys/apic_common.h>
76 #include <sys/apic_timer.h>

78 static void    apic_record_ioapic_rdt(void *intrmap_private,
79                                         ioapic_rdt_t *irdt);
80 static void    apic_record_msi(void *intrmap_private, msi_regs_t *mregs);

82 /*
83  * Common routines between pcplusmp & apix (taken from apic.c).
84 */

86 int    apic_clkinit(int);
87 hrtime_t apic_gethrtime(void);
88 void    apic_send_ipi(int, int);
89 void    apic_set_idlecpcu(processorid_t);
90 void    apic_unset_idlecpcu(processorid_t);
91 void    apic_shutdown(int, int);
92 void    apic_preshutdown(int, int);
93 processorid_t apic_get_next_processorid(processorid_t);

95 hrtime_t apic_gettime();

97 enum apic_ioapic_method_type apix_mul_ioapic_method = APIC_MUL_IOAPIC_PCPLUSMP;

99 /* Now the ones for Dynamic Interrupt distribution */
100 int    apic_enable_dynamic_migration = 0;

102 /* maximum loop count when sending Start IPIs. */
103 int apic_sipi_max_loop_count = 0x1000;

105 /*
106  * These variables are frequently accessed in apic_intr_enter(),
107  * apic_intr_exit and apic_setspl, so group them together
108 */
109 volatile uint32_t *apicadr = NULL; /* virtual addr of local APIC */
110 int apic_setspl_delay = 1; /* apic_setspl - delay enable */
111 int apic_clkvect;

113 /* vector at which error interrupts come in */
114 int apic_errvect;
115 int apic_enable_error_intr = 1;
116 int apic_error_display_delay = 100;

118 /* vector at which performance counter overflow interrupts come in */
119 int apic_cpcovf_vect;
120 int apic_enable_cpcovf_intr = 1;

122 /* vector at which CMCI interrupts come in */
123 int apic_cmci_vect;
124 extern int cmi_enable_cmci;
125 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_hrttime = 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; /* 0x1fff */

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_hrttime_stamp = 0;
162 volatile hrttime_t apic_nsec_since_boot = 0;

164 static hrttime_t apic_last_hrttime = 0;
165 int apic_hrttime_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
```

new/usr/src/uts/i86pc/io/ppm/acpippm.c

1

```
*****
9947 Fri Jan 17 07:31:35 2014
new/usr/src/uts/i86pc/io/ppm/acpippm.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 */

22 /*
23  * Copyright 2008 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  * Copyright (c) 2011 Bayard G. Bell. All rights reserved.
26  */

28 #include <sys/types.h>
29 #include <sys/conf.h>
30 #include <sys/open.h>
31 #include <sys/modctl.h>
32 #include <sys/promif.h>
33 #include <sys/stat.h>
34 #include <sys/ddi_impldefs.h>
35 #include <sys/ddi.h>
36 #include <sys/sunddi.h>
37 #include <sys/epm.h>
38 #include <acpica/include/acpi.h>
39 #include <sys/acpi/acpi.h>
40 #include <sys/acpica.h>
41 #include <sys/psm_types.h>

42 /*
43  * ACPI Power Management Driver
44  *
45  * acpippm deals with those bits of ppm functionality that
46  * must be mediated by ACPI
47  *
48  * The routines in this driver is referenced by Platform
49  * Power Management driver of X86 workstation systems.
50  * acpippm driver is loaded because it is listed as a platform driver
51  * It is initially configured as a pseudo driver.
52  */
53 extern void pc_tod_set_rtc_offsets(ACPI_TABLE_FADT *);
54 extern int acpica_use_safe_delay;

56 /*
57  * Configuration Function prototypes and data structures
58  */
59 static int appm_attach(dev_info_t *, ddi_attach_cmd_t);
60 static int appm_detach(dev_info_t *, ddi_detach_cmd_t);
```

new/usr/src/uts/i86pc/io/ppm/acpippm.c

2

```
61 static int appm_getinfo(dev_info_t *, ddi_info_cmd_t, void *, void **);
62 static int appm_open(dev_t *dev_p, int flag, int otyp, cred_t *cred_p);
63 static int appm_close(dev_t dev, int flag, int otyp, cred_t *cred_p);
64 static int appm_ioctl(dev_t, int, intptr_t, int, cred_t *, int *);

66 /*
67  * Configuration data structures
68  */
69 static struct cb_ops appm_cbops = {
70     appm_open, /* open */
71     appm_close, /* close */
72     nodev, /* strategy */
73     nodev, /* print */
74     nodev, /* dump */
75     nodev, /* read */
76     nodev, /* write */
77     appm_ioctl, /* ioctl */
78     nodev, /* devmap */
79     nodev, /* mmap */
80     nodev, /* segmap */
81     nochpoll, /* chpoll */
82     ddi_prop_op, /* prop_op */
83     NULL, /* stream */
84     D_MP | D_NEW, /* flag */
85     CB_REV, /* rev */
86     nodev, /* aread */
87     nodev, /* awrite */
88 };
    unchanged_portion_omitted_
```



new/usr/src/uts/i86pc/io/ppm/acpisleep.c

1

```
*****
4338 Fri Jan 17 07:31:35 2014
new/usr/src/uts/i86pc/io/ppm/acpisleep.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 */

22 /*
23  * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  */

27 #include <sys/types.h>
28 #include <sys/smp_impldefs.h>
29 #include <sys/promif.h>

31 #include <sys/kmem.h>
32 #include <sys/archsystem.h>
33 #include <sys/cpuvar.h>
34 #include <sys/pte.h>
35 #include <vm/seg_kmem.h>
36 #include <sys/epm.h>
37 #include <sys/cpr.h>
38 #include <sys/machsystem.h>
39 #include <sys/clock.h>

41 #include <sys/cpr_wakecode.h>
42 #include <acpica/include/acpi.h>
42 #include <sys/acpi/acpi.h>

44 #ifdef OLDPMPCODE
45 #include "acpi.h"
46 #endif

48 #include <sys/x86_archext.h>
49 #include <sys/reboot.h>
50 #include <sys/cpu_module.h>
51 #include <sys/kdi.h>

53 /*
54  * S3 stuff
55  */

57 int acpi_rtc_wake = 0x0; /* wake in N seconds */

59 #if 0 /* debug */
60 static uint8_t branchbuf[64 * 1024]; /* for the HDT branch trace stuff */
```

new/usr/src/uts/i86pc/io/ppm/acpisleep.c

2

```
61 #endif /* debug */

63 extern int boothowto;

65 #define BOOTCPU 0 /* cpu 0 is always the boot cpu */

67 extern void kernel_wc_code(void);
68 extern tod_ops_t *tod_ops;
69 extern int flushes_require_xcalls;
70 extern int tsc_gethrtime_enable;

72 extern cpuset_t cpu_ready_set;
73 extern void (*cpu_pause_func)(void *);

77 /*
78  * This is what we've all been waiting for!
79  */
80 int
81 acpi_enter_sleepstate(s3a_t *s3ap)
82 {
83     ACPI_PHYSICAL_ADDRESS wakephys = s3ap->s3a_wakephys;
84     caddr_t wakevirt = rm_platter_va;
85     /*LINTED*/
86     wakecode_t *wp = (wakecode_t *)wakevirt;
87     uint_t Sx = s3ap->s3a_state;

89     PT(PT_SWV);
90     /* Set waking vector */
91     if (AcpiSetFirmwareWakingVector(wakephys) != AE_OK) {
92         PT(PT_SWV_FAIL);
93         PMD(PMD_SX, ("Can't SetFirmwareWakingVector(%lx)\n",
94             (long)wakephys))
95         goto insomnia;
96     }

98     PT(PT_EWE);
99     /* Enable wake events */
100     if (AcpiEnableEvent(ACPI_EVENT_POWER_BUTTON, 0) != AE_OK) {
101         PT(PT_EWE_FAIL);
102         PMD(PMD_SX, ("Can't EnableEvent(POWER_BUTTON)\n"))
103     }
104     if (acpi_rtc_wake > 0) {
105         /* clear the RTC bit first */
106         (void) AcpiWriteBitRegister(ACPI_BITREG_RT_CLOCK_STATUS, 1);
107         PT(PT_RTCW);
108         if (AcpiEnableEvent(ACPI_EVENT_RTC, 0) != AE_OK) {
109             PT(PT_RTCW_FAIL);
110             PMD(PMD_SX, ("Can't EnableEvent(RTC)\n"))
111         }
112     }

113     /*
114      * Set RTC to wake us in a wee while.
115      */
116     mutex_enter(&tod_lock);
117     PT(PT_TOD);
118     TODOP_SETWAKE(tod_ops, acpi_rtc_wake);
119     mutex_exit(&tod_lock);
120 }

122 /*
123  * Prepare for sleep ... could've done this earlier?
124  */
125 PT(PT_SXP);
126 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_
```

new/usr/src/uts/i86pc/io/psm/psm\_common.c

1

```
*****
27091 Fri Jan 17 07:31:35 2014
new/usr/src/uts/i86pc/io/psm/psm_common.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 */
21 /*
22 * Copyright (c) 2004, 2010, Oracle and/or its affiliates. All rights reserved.
23 * Copyright 2013 PALO, Richard. All rights reserved.
24 */

26 #include <sys/types.h>
27 #include <sys/param.h>
28 #include <sys/cmn_err.h>
29 #include <sys/promif.h>
30 #include <acpica/include/acpi.h>
29 #include <sys/acpi/acpi.h>
31 #include <sys/acpica.h>
32 #include <sys/sunddi.h>
33 #include <sys/ddi.h>
34 #include <sys/ddi_impldefs.h>
35 #include <sys/pci.h>
36 #include <sys/debug.h>
37 #include <sys/psm_common.h>
38 #include <sys/sunndi.h>
39 #include <sys/ksynch.h>

41 /* Global configurables */

43 char *psm_module_name; /* used to store name of psm module */

45 /*
46 * acpi_irq_check_elcr: when set elcr will also be consulted for building
47 * the reserved irq list. When 0 (false), the existing state of the ELCR
48 * is ignored when selecting a vector during IRQ translation, and the ELCR
49 * is programmed to the proper setting for the type of bus (level-triggered
50 * for PCI, edge-triggered for non-PCI). When non-zero (true), vectors
51 * set to edge-mode will not be used when in PIC-mode. The default value
52 * is 0 (false). Note that ACPI's SCI vector is always set to conform to
53 * ACPI-specification regardless of this.
54 *
55 */
56 int acpi_irq_check_elcr = 0;

58 int psm_verbose = 0;
```

new/usr/src/uts/i86pc/io/psm/psm\_common.c

2

```
60 #define PSM_VERBOSE_IRQ(fmt) \
61     if (psm_verbose & PSM_VERBOSE_IRQ_FLAG) \
62         cmn_err fmt;

64 #define PSM_VERBOSE_POWEROFF(fmt) \
65     if (psm_verbose & PSM_VERBOSE_POWEROFF_FLAG || \
66         psm_verbose & PSM_VERBOSE_POWEROFF_PAUSE_FLAG) \
67         prom_printf fmt;

69 #define PSM_VERBOSE_POWEROFF_PAUSE(fmt) \
70     if (psm_verbose & PSM_VERBOSE_POWEROFF_FLAG || \
71         psm_verbose & PSM_VERBOSE_POWEROFF_PAUSE_FLAG) {\
72         prom_printf fmt; \
73         if (psm_verbose & PSM_VERBOSE_POWEROFF_PAUSE_FLAG) \
74             (void) goany(); \
75     }

78 /* Local storage */
79 static ACPI_HANDLE acpi_sboj = NULL;
80 static kmutex_t acpi_irq_cache_mutex;

82 /*
83 * irq_cache_table is a list that serves a two-key cache. It is used
84 * as a pci busid/devid/ipin <-> irq cache and also as a acpi
85 * interrupt lnk <-> irq cache.
86 */
87 static irq_cache_t *irq_cache_table;

89 #define IRQ_CACHE_INITLEN      20
90 static int irq_cache_len = 0;
91 static int irq_cache_valid = 0;

93 static int acpi_get_gsi(dev_info_t *dip, ACPI_HANDLE pciobj, int devno,
94     int ipin, int *pci_irqp, iflag_t *iflagp, acpi_psm_lnk_t *acpipsmlnkp);

96 static int acpi_eval_lnk(dev_info_t *dip, char *lnkname,
97     int *pci_irqp, iflag_t *intr_flagp, acpi_psm_lnk_t *acpipsmlnkp);

99 static int acpi_get_irq_lnk_cache_ent(ACPI_HANDLE lnkobj, int *pci_irqp,
100     iflag_t *intr_flagp);

102 extern int goany(void);

105 #define NEXT_PRT_ITEM(p) \
106     ((void *)(((char *) (p)) + (p)->Length))

108 static int
109 acpi_get_gsi(dev_info_t *dip, ACPI_HANDLE pciobj, int devno, int ipin,
110     int *pci_irqp, iflag_t *intr_flagp, acpi_psm_lnk_t *acpipsmlnkp)
111 {
112     ACPI_BUFFER rb;
113     ACPI_PCI_ROUTING_TABLE *prtp;
114     int status;
115     int dev_adr;

117     /*
118      * Get the IRQ routing table
119      */
120     rb.Pointer = NULL;
121     rb.Length = ACPI_ALLOCATE_BUFFER;
122     if (AcpiGetIrqRoutingTable(pciobj, &rb) != AE_OK) {
123         return (ACPI_PSM_FAILURE);
124     }
```

```

126     status = ACPI_PSM_FAILURE;
127     dev_addr = (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_addr != 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  * Returns ACPI_PSM_SUCCESS on success, ACPI_PSM_FAILURE upon failure.
452  */
453 int
454 acpi_set_irq_resource(acpi_psm_lnk_t *acpipsmlnkp, int irq)
455 {
456     ACPI_BUFFER    rsb;
457     ACPI_RESOURCE  *resp;
458     ACPI_RESOURCE  *srsp;
459     ACPI_HANDLE    lnkobj;
460     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;
493         srs_len = resp->Length + sizeof (*endtag);
494         srsp = kmem_zalloc(srs_len, KM_SLEEP);
495         bcopy(resp, srsp, resp->Length);
496         endtag = ACPI_NEXT_RESOURCE(srsp);
497         endtag->Type = ACPI_RESOURCE_TYPE_END_TAG;
498         endtag->Length = ACPI_RS_SIZE_NO_DATA;
499         break; /* drop out of the loop */
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

new/usr/src/uts/i86pc/io/todpc\_subr.c

1

```
*****
11730 Fri Jan 17 07:31:36 2014
new/usr/src/uts/i86pc/io/todpc_subr.c
PANKOVs restrcture
*****

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 2012 Nexenta Systems, Inc. All rights reserved.
23 */
24 /*
25 * Copyright 2010 Sun Microsystems, Inc. All rights reserved.
26 * Use is subject to license terms.
27 */

29 /*      Copyright (c) 1990, 1991 UNIX System Laboratories, Inc. */
30 /*      Copyright (c) 1984, 1986, 1987, 1988, 1989, 1990 AT&T */
31 /*      All Rights Reserved */

33 /*      Copyright (c) 1987, 1988 Microsoft Corporation */
34 /*      All Rights Reserved */

36 #include <sys/param.h>
37 #include <sys/time.h>
38 #include <sys/systm.h>

40 #include <sys/cpuvar.h>
41 #include <sys/clock.h>
42 #include <sys/debug.h>
43 #include <sys/rtc.h>
44 #include <sys/archsystm.h>
45 #include <sys/sysmacros.h>
46 #include <sys/lockstat.h>
47 #include <sys/stat.h>
48 #include <sys/sunddi.h>

50 #include <acpica/include/acpi.h>
50 #include <sys/acpi/acpi.h>
51 #include <sys/acpica.h>

53 static int todpc_rtcget(unsigned char *buf);
54 static void todpc_rtcput(unsigned char *buf);

56 #define CLOCK_RES          1000          /* 1 microsec in nanosecs */

58 int clock_res = CLOCK_RES;

60 /*
```

new/usr/src/uts/i86pc/io/todpc\_subr.c

2

```
61 * The minimum sleep time till an alarm can be fired.
62 * This can be tuned in /etc/system, but if the value is too small,
63 * there is a danger that it will be missed if it takes too long to
64 * get from the set point to sleep. Or that it can fire quickly, and
65 * generate a power spike on the hardware. And small values are
66 * probably only usefull for test setups.
67 */
68 int clock_min_alarm = 4;

70 /*
71 * Machine-dependent clock routines.
72 */

74 extern long gmt_lag;

76 struct rtc_offset {
77     int8_t loaded;
78     uint8_t day_alarm;
79     uint8_t mon_alarm;
80     uint8_t century;
81 };
_____unchanged_portion_omitted_
```

new/usr/src/uts/i86pc/io/tzmon/tzmon.c

1

```
*****
17867 Fri Jan 17 07:31:36 2014
new/usr/src/uts/i86pc/io/tzmon/tzmon.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 */

22 /*
23  * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  */

27 /*
28  * Solaris x86 ACPI ThermalZone Monitor
29  */

32 #include <sys/errno.h>
33 #include <sys/conf.h>
34 #include <sys/modctl.h>
35 #include <sys/open.h>
36 #include <sys/stat.h>
37 #include <sys/ddi.h>
38 #include <sys/sunddi.h>
39 #include <sys/ksynch.h>
40 #include <sys/uadmin.h>
41 #include <acpica/include/acpi.h>
42 #include <sys/acpi/acpi.h>
43 #include <sys/acpica.h>
44 #include <sys/sdt.h>

45 #include "tzmon.h"

48 #define TZMON_ENUM_TRIP_POINTS 1
49 #define TZMON_ENUM_DEV_LISTS 2
50 #define TZMON_ENUM_ALL (TZMON_ENUM_TRIP_POINTS | TZMON_ENUM_DEV_LISTS)

52 /*
53  * TZ_TASKQ_NAME_LEN is precisely the length of the string "AcpiThermalMonitor"
54  * plus a two-digit instance number plus a NULL. If the taskq name is changed
55  * (particularly if it is lengthened), then this value needs to change.
56  */
57 #define TZ_TASKQ_NAME_LEN 21

59 /*
60  * Kelvin to Celsius conversion
```

new/usr/src/uts/i86pc/io/tzmon/tzmon.c

2

```
61 * The formula for converting degrees Kelvin to degrees Celsius is
62 * C = K - 273.15 (we round to 273.2). The unit for thermal zone
63 * temperatures is tenths of a degree Kelvin. Use tenth of a degree
64 * to convert, then make a whole number out of it.
65 */
66 #define K_TO_C(temp) (((temp) - 2732) / 10)

69 /* cb_ops or dev_ops forward declarations */
70 static int tzmon_getinfo(dev_info_t *dip, ddi_info_cmd_t infocmd,
71 void *arg, void **result);
72 static int tzmon_attach(dev_info_t *dip, ddi_attach_cmd_t cmd);
73 static int tzmon_detach(dev_info_t *dip, ddi_detach_cmd_t cmd);

75 /* other forward declarations */
76 static void tzmon_notify_zone(ACPI_HANDLE obj, UINT32 val, void *ctx);
77 static void tzmon_eval_int(ACPI_HANDLE obj, char *method, int *rv);
78 static thermal_zone_t *tzmon_alloc_zone();
79 static void tzmon_free_zone_list();
80 static void tzmon_discard_buffers(thermal_zone_t *tzp);
81 static void tzmon_enumerate_zone(ACPI_HANDLE obj, thermal_zone_t *tzp,
82 int enum_flag);
83 static ACPI_STATUS tzmon_zone_callback(ACPI_HANDLE obj, UINT32 nest,
84 void *ctx, void **rv);
85 static void tzmon_find_zones(void);
86 static void tzmon_monitor(void *ctx);
87 static void tzmon_set_power_device(ACPI_HANDLE dev, int on_off, char *tz_name);
88 static void tzmon_set_power(ACPI_BUFFER devlist, int on_off, char *tz_name);
89 static void tzmon_eval_zone(thermal_zone_t *tzp);
90 static void tzmon_do_shutdown(void);

92 extern void halt(char *);

94 static struct cb_ops tzmon_cb_ops = {
95 nodev, /* no open routine */
96 nodev, /* no close routine */
97 nodev, /* not a block driver */
98 nodev, /* no print routine */
99 nodev, /* no dump routine */
100 nodev, /* no read routine */
101 nodev, /* no write routine */
102 nodev, /* no ioctl routine */
103 nodev, /* no devmap routine */
104 nodev, /* no mmap routine */
105 nodev, /* no segmap routine */
106 nochpoll, /* no chpoll routine */
107 ddi_prop_op,
108 0, /* not a STREAMS driver */
109 D_NEW | D_MP, /* safe for multi-thread/multi-processor */
110 };

_____unchanged_portion_omitted_____
```

new/usr/src/uts/i86pc/os/acpi\_stubs.c

1

\*\*\*\*\*

1456 Fri Jan 17 07:31:37 2014

new/usr/src/uts/i86pc/os/acpi\_stubs.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 2007 Sun Microsystems, Inc. All rights reserved.
23 * Use is subject to license terms.
24 */
```

```
26 #pragma ident      "%Z%M% %I%      %E% SMI"
```

```
26 #include <sys/types.h>
```

```
28 #include <acpica/include/acpi.h>
```

```
30 #include <sys/acpi/acpi.h>
```

```
29 #include <sys/acpica.h>
```

```
31 /*
32 * This file contains ACPI functions that are needed by the kernel before
33 * the ACPI module is loaded. Any functions or definitions need to be
34 * able to deal with the possibility that ACPI doesn't get loaded, or
35 * doesn't contain the required method.
36 */
```

```
38 int (*acpi_fp_setwake)();
```

```
40 /*
41 *
42 */
43 int
44 acpi_ddi_setwake(dev_info_t *dip, int level)
45 {
46     if (acpi_fp_setwake == NULL)
47         return (AE_ERROR);
48
49     return ((*acpi_fp_setwake)(dip, level));
50 }
```

unchanged\_portion\_omitted

new/usr/src/uts/i86pc/os/cpr\_impl.c

1

```
*****
27105 Fri Jan 17 07:31:37 2014
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/machsystem.h>
50 #include <sys/vmsystem.h>
51 #include <sys/systm.h>
52 #include <sys/archsystem.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>
```

new/usr/src/uts/i86pc/os/cpr\_impl.c

2

```
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>
67 #include <sys/acpi/acpi.h>
68 #include <sys/acpica.h>
69 #include <sys/fp.h>
70 #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_descbr_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
```



new/usr/src/uts/i86pc/os/cpupm/cpu\_idle.c

1

```
*****
26724 Fri Jan 17 07:31:38 2014
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/machsystem.h>
32 #include <sys/x_call.h>
33 #include <sys/stat.h>
34 #include <acpica/include/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/archsystem.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);
```

new/usr/src/uts/i86pc/os/cpupm/cpu\_idle.c

2

```
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_____
```

new/usr/src/uts/i86pc/os/cpum/p/cpum\_mach.c

1

```
*****
26818 Fri Jan 17 07:31:38 2014
new/usr/src/uts/i86pc/os/cpum/p/cpum_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 */
29
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/machsystem.h>
35 #include <sys/archsystem.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/cpum.h>
40 #include <sys/cpu_idle.h>
41 #include <sys/cpu_acpi.h>
42 #include <sys/cpum_throttle.h>
43 #include <sys/dtrace.h>
44 #include <sys/note.h>
45
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 (*cpum_ppm_alloc_pstate_domains)(cpu_t *);
53
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 (*cpum_ppm_free_pstate_domains)(cpu_t *);
```

new/usr/src/uts/i86pc/os/cpum/p/cpum\_mach.c

2

```
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 (*cpum_redefine_topspeed)(void *);
71
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 (*cpum_set_topspeed_callb)(void *, int);
77
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 (*cpum_get_topspeed_callb)(void *);
83
84 static void cpum_event_notify_handler(ACPI_HANDLE, UINT32, void *);
85 static void cpum_free_notify_handlers(cpu_t *);
86 static void cpum_power_manage_notifications(void *);
87
88 /*
89 * Until proven otherwise, all power states are manageable.
90 */
91 static uint32_t cpum_enabled = CPUM_ALL_STATES;
92
93 cpum_state_domains_t *cpum_pstate_domains = NULL;
94 cpum_state_domains_t *cpum_tstate_domains = NULL;
95 cpum_state_domains_t *cpum_cstate_domains = NULL;
96
97 /*
98 * c-state tunables
99 */
100 * cpum_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 cpum_cs_sample_interval since it last
103 * calculated which C-state to go to.
104 *
105 * cpum_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 * cpum_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 cpum_cs_sample_interval = 100*1000*1000; /* 100 milliseconds */
115 uint32_t cpum_cs_idle_cost_tunable = 10; /* work time / latency cost */
116 uint32_t cpum_cs_idle_save_tunable = 2; /* idle power savings */
117 uint16_t cpum_C2_idle_pct_tunable = 70;
118 uint16_t cpum_C3_idle_pct_tunable = 80;
119
120 #ifndef __xpv
121 extern boolean_t cpum_intel_init(cpu_t *);
122 extern boolean_t cpum_amd_init(cpu_t *);
123
124 typedef struct cpum_vendor {
125     boolean_t (*cpuv_init)(cpu_t *);
126 } cpum_vendor_t;
127
128 unchanged_portion_omitted
```

new/usr/src/uts/i86pc/os/cpum/pwrnow.c

1

\*\*\*\*\*

7735 Fri Jan 17 07:31:38 2014

new/usr/src/uts/i86pc/os/cpum/pwrnow.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 #include <sys/x86_archext.h>
26 #include <sys/machsysm.h>
27 #include <sys/x_call.h>
28 #include <acpica/include/acpi.h>
29 #include <sys/acpi/acpi.h>
30 #include <sys/acpica.h>
31 #include <sys/pwrnow.h>
32 #include <sys/cpu_acpi.h>
33 #include <sys/cpum.h>
34 #include <sys/dtrace.h>
35 #include <sys/sdt.h>

36 static int pwrnow_init(cpu_t *);
37 static void pwrnow_fini(cpu_t *);
38 static void pwrnow_power(cpuset_t, uint32_t);
39 static void pwrnow_stop(cpu_t *);

41 static boolean_t pwrnow_cpb_supported(void);

43 /*
44 * Interfaces for modules implementing AMD's PowerNow!.
45 */
46 cpum_state_ops_t pwrnow_ops = {
47     "PowerNow! Technology",
48     pwrnow_init,
49     pwrnow_fini,
50     pwrnow_power,
51     pwrnow_stop
52 };
unchanged portion omitted
```

new/usr/src/uts/i86pc/os/cpum/p/speedstep.c

1

\*\*\*\*\*

7884 Fri Jan 17 07:31:39 2014

new/usr/src/uts/i86pc/os/cpum/p/speedstep.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 */
24 /*
25 * Copyright (c) 2009, Intel Corporation.
26 * All Rights Reserved.
27 */

29 #include <sys/x86_archext.h>
30 #include <sys/machsystem.h>
31 #include <sys/archsystem.h>
32 #include <sys/x_call.h>
33 #include <acpica/include/acpi.h>
34 #include <sys/acpi/acpi.h>
35 #include <sys/speedstep.h>
36 #include <sys/cpu_acpi.h>
37 #include <sys/cpum.h>
38 #include <sys/dtrace.h>
39 #include <sys/sdt.h>

41 static int speedstep_init(cpu_t *);
42 static void speedstep_fini(cpu_t *);
43 static void speedstep_power(cpuset_t, uint32_t);
44 static void speedstep_stop(cpu_t *);
45 static boolean_t speedstep_turbo_supported(void);

47 /*
48 * Interfaces for modules implementing Intel's Enhanced SpeedStep.
49 */
50 cpum_state_ops_t speedstep_ops = {
51     "Enhanced SpeedStep Technology",
52     speedstep_init,
53     speedstep_fini,
54     speedstep_power,
55     speedstep_stop
56 };
57 _____unchanged_portion_omitted_____
```

new/usr/src/uts/i86pc/os/cpum/turbo.c

1

\*\*\*\*\*

5163 Fri Jan 17 07:31:39 2014

new/usr/src/uts/i86pc/os/cpum/turbo.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 */
24 /*
25 * Copyright (c) 2009, Intel Corporation.
26 * All Rights Reserved.
27 */

28 #include <sys/x86_archext.h>
29 #include <sys/machsystem.h>
30 #include <sys/archsystem.h>
31 #include <sys/x_call.h>
32 #include <sys/acpi.h>
33 #include <acpica/include/acpi.h>
34 #include <sys/acpi.h>
35 #include <sys/speedstep.h>
36 #include <sys/cpu_acpi.h>
37 #include <sys/cpum.h>
38 #include <sys/dtrace.h>
39 #include <sys/sdt.h>

40 typedef struct turbo_kstat_s {
41     struct kstat_named    turbo_supported;    /* turbo flag */
42     struct kstat_named    t_mcmt;            /* IA32_MPERF_MSR */
43     struct kstat_named    t_acnt;            /* IA32_APERF_MSR */
44 } turbo_kstat_t;
45 }
46
47 _____
48 unchanged_portion_omitted_
49 _____
```

new/usr/src/uts/i86pc/sys/acpidev.h

1

```
*****
19844 Fri Jan 17 07:31:40 2014
new/usr/src/uts/i86pc/sys/acpidev.h
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 */
21 /*
22  * Copyright (c) 2009-2010, Intel Corporation.
23  * All rights reserved.
24  */

26 #ifndef _SYS_ACPIDEV_H
27 #define _SYS_ACPIDEV_H
28 #include <sys/types.h>
29 #include <sys/obpdefs.h>
30 #include <sys/sunddi.h>
31 #ifdef _KERNEL
32 #include <acpica/include/acpi.h>
32 #include <sys/acpi/acpi.h>
33 #include <sys/acpica.h>
34 #endif

36 #ifdef __cplusplus
37 extern "C" {
38 #endif

40 /* Maximum recursion levels when enumerating objects in ACPI namespace. */
41 #define ACPIDEV_MAX_ENUM_LEVELS 32

43 /* Maximum length of device name for ACPI object. */
44 #define ACPIDEV_MAX_NAMELEN OBP_MAXDRVNAME

46 /* Pseudo ACPI device HID for ACPI root object. */
47 #define ACPIDEV_HID_ROOTNEX "SOLA0001"
48 /* Pseudo ACPI device HID for ACPI virtual bus. */
49 #define ACPIDEV_HID_VIRTNEX "SOLA0002"
50 #define ACPIDEV_HID_SCOPE "SOLA0003"
51 #define ACPIDEV_HID_PROCESSOR "SOLA0004"

53 /* ACPI device HIDs/CIDs defined by ACPI specification. */
54 #define ACPIDEV_HID_CONTAINER1 "PNP0A05"
55 #define ACPIDEV_HID_CONTAINER2 "PNP0A06"
56 #define ACPIDEV_HID_MODULE "ACPI0004"
57 #define ACPIDEV_HID_CPU "ACPI0007"
58 #define ACPIDEV_HID_PCI_HOSTBRIDGE "PNP0A03"
59 #define ACPIDEV_HID_PCIE_HOSTBRIDGE "PNP0A08"
```

new/usr/src/uts/i86pc/sys/acpidev.h

2

```
60 #define ACPIDEV_HID_PCIE_HOSTBRIDGE "PNP0A08"
61 #define ACPIDEV_HID_MEMORY "PNP0C80"

63 /* Device names for ACPI objects. */
64 #define ACPIDEV_NODE_NAME_ROOT "fw"
65 #define ACPIDEV_NODE_NAME_ACPIDR "acpidr"
66 #define ACPIDEV_NODE_NAME_CONTAINER "container"
67 #define ACPIDEV_NODE_NAME_MODULE_SBD "sb"
68 #define ACPIDEV_NODE_NAME_MODULE_CPU "socket"
69 #define ACPIDEV_NODE_NAME_CPU "cpu"
70 #define ACPIDEV_NODE_NAME_PROCESSOR "cpus"
71 #define ACPIDEV_NODE_NAME_MEMORY "mem"
72 #define ACPIDEV_NODE_NAME_PCI "pci"

74 /* Device types for ACPI objects. */
75 #define ACPIDEV_TYPE_ROOTNEX "acpirootnex"
76 #define ACPIDEV_TYPE_VIRTNEX "acpivirtnex"
77 #define ACPIDEV_TYPE_SCOPE "acpiscope"
78 #define ACPIDEV_TYPE_DEVICE "acpidevice"
79 #define ACPIDEV_TYPE_CONTAINER "acpicontainer"
80 #define ACPIDEV_TYPE_CPU "acpicpu"
81 #define ACPIDEV_TYPE_MEMORY "acpimemory"
82 #define ACPIDEV_TYPE_PCI "pci"
83 #define ACPIDEV_TYPE_PCIE "pcie"

85 /* Device property names for ACPI objects. */
86 #define ACPIDEV_PROP_NAME_UNIT_ADDR "unit-address"
87 #define ACPIDEV_PROP_NAME_ACPI_UID "acpi-uid"
88 #define ACPIDEV_PROP_NAME_PROCESSOR_ID "acpi-processor-id"
89 #define ACPIDEV_PROP_NAME_LOCALAPIC_ID "apic-id"
90 #define ACPIDEV_PROP_NAME_PROXIMITY_ID "proximity-id"

92 #define ACPIDEV_PROP_NAME_UID_FORMAT "acpidev-uid-format"

94 /* Miscellaneous strings. */
95 #define ACPIDEV_CMD_OST_PREFIX "acpi-update-status"
96 #define ACPIDEV_CMD_OST_INPROGRESS "acpi-update-status=inprogress"
97 #define ACPIDEV_CMD_OST_SUCCESS "acpi-update-status=success"
98 #define ACPIDEV_CMD_OST_FAILURE "acpi-update-status=failure"
99 #define ACPIDEV_CMD_OST_NOOP "acpi-update-status=noop"

101 #define ACPIDEV_EVENT_TYPE_ATTR_NAME "acpi-event-type"
102 #define ACPIDEV_EVENT_TYPE_BUS_CHECK "bus_check"
103 #define ACPIDEV_EVENT_TYPE_DEVICE_CHECK "device_check"
104 #define ACPIDEV_EVENT_TYPE_DEVICE_CHECK_LIGHT "device_check_light"
105 #define ACPIDEV_EVENT_TYPE_EJECT_REQUEST "eject_request"

107 /* ACPI device class Id. */
108 typedef enum acpidev_class_id {
109     ACPIDEV_CLASS_ID_INVALID = 0,
110     ACPIDEV_CLASS_ID_ROOTNEX = 1,
111     ACPIDEV_CLASS_ID_SCOPE = 2,
112     ACPIDEV_CLASS_ID_DEVICE = 3,
113     ACPIDEV_CLASS_ID_CONTAINER = 4,
114     ACPIDEV_CLASS_ID_CPU = 5,
115     ACPIDEV_CLASS_ID_MEMORY = 6,
116     ACPIDEV_CLASS_ID_PCI = 7,
117     ACPIDEV_CLASS_ID_PCIE = 8,
118     ACPIDEV_CLASS_ID_MAX
119 } acpidev_class_id_t;

121 /* Flags for acpidev_options boot options. */
122 #define ACPIDEV_OUSER_NO_CPU 0x1
123 #define ACPIDEV_OUSER_NO_MEM 0x2
124 #define ACPIDEV_OUSER_NO_CONTAINER 0x4
125 #define ACPIDEV_OUSER_NO_PCI 0x8
```

new/usr/src/uts/i86pc/sys/acpidev.h

3

```
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
```

new/usr/src/uts/i86pc/sys/acpidev.h

4

```
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;
220 unchanged_portion_omitted
```

new/usr/src/uts/i86pc/sys/acpidev\_dr.h

1

```
*****
8327 Fri Jan 17 07:31:40 2014
new/usr/src/uts/i86pc/sys/acpidev_dr.h
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) 2010, Intel Corporation.
24  * All rights reserved.
25  */
26 /*
27  * Interfaces to support System Board Dynamic Reconfiguration.
28  */

30 #ifndef _SYS_ACPIDEV_DR_H
31 #define _SYS_ACPIDEV_DR_H
32 #include <sys/types.h>
33 #include <sys/obpdefs.h>
34 #include <sys/cpuvar.h>
35 #include <sys/memlist.h>
36 #include <sys/sunddi.h>
37 #include <acpica/include/acpi.h>
37 #include <sys/acpi/acpi.h>
38 #include <sys/acpica.h>
39 #include <sys/acpidev.h>
40 #include <sys/acpidev_rsc.h>

42 #ifdef __cplusplus
43 extern "C" {
44 #endif

46 #ifdef _KERNEL

48 /* Maximum number of DR capable system boards supported. */
49 #define ACPIDEV_DR_MAX_BOARDS 0x40
50 #define ACPIDEV_DR_SEGS_PER_MEM_DEV 0x10
51 #define ACPIDEV_DR_MEMLISTS_PER_SEG 0x10
52 #define ACPIDEV_DR_MAX_MEMLIST_ENTRIES 0x10000

54 #define ACPIDEV_DR_PROP_PORTID "portid"
55 #define ACPIDEV_DR_PROP_BOARDNUM OBP_BOARDNUM
56 #define ACPIDEV_DR_PROP_DEVNAME OBP_NAME

58 /*
59  * Format strings for DR capable system boards.
60  * They will be used as attachment point names.
```

new/usr/src/uts/i86pc/sys/acpidev\_dr.h

2

```
61 */
62 #define ACPIDEV_DR_CPU_BD_FMT "CPU%u"
63 #define ACPIDEV_DR_MEMORY_BD_FMT "MEM%u"
64 #define ACPIDEV_DR_IO_BD_FMT "IO%u"
65 #define ACPIDEV_DR_SYSTEM_BD_FMT "SB%u"

67 typedef enum {
68     ACPIDEV_INVALID_BOARD = 0,
69     ACPIDEV_CPU_BOARD,
70     ACPIDEV_MEMORY_BOARD,
71     ACPIDEV_IO_BOARD,
72     ACPIDEV_SYSTEM_BOARD
73 } acpidev_board_type_t;
unchanged_portion_omitted
```



new/usr/src/uts/i86pc/sys/acpidev\_impl.h

1

```
*****
4564 Fri Jan 17 07:31:40 2014
new/usr/src/uts/i86pc/sys/acpidev_impl.h
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) 2009-2010, Intel Corporation.
23 * All rights reserved.
24 */

26 #ifndef _SYS_ACPIDEV_IMPL_H
27 #define _SYS_ACPIDEV_IMPL_H
28 #include <sys/types.h>
29 #include <sys/cmn_err.h>
30 #include <sys/bitmap.h>
31 #include <sys/synch.h>
32 #include <sys/sunddi.h>
33 #include <acpica/include/acpi.h>
34 #include <sys/acpi/acpi.h>
35 #include <sys/acpica.h>
36 #include <sys/acpidev.h>
37 #include <sys/acpidev_dr.h>

38 #ifdef __cplusplus
39 extern "C" {
40 #endif

42 #ifdef _KERNEL

44 #define ACPIDEV_ARRAY_PARAM(a)      (a), (sizeof (a) / sizeof ((a)[0]))

46 /* Debug support facilities. */
47 extern int acpidev_debug;
48 #define ACPIDEV_DEBUG(lvl, ...) if (acpidev_debug) cmn_err((lvl), __VA_ARGS__)

50 /* Data attached to an ACPI object to maintain device status information. */
51 struct acpidev_data_impl {
52     uint32_t          aod_eflag;      /* External flags */
53     uint32_t          aod_iflag;      /* Internal flags */
54     uint32_t          aod_level;
55     int               aod_status;      /* Cached _STA value */
56     ACPI_HANDLE        *aod_hdl;
57     dev_info_t         *aod_dip;
58     acpidev_class_t    *aod_class;
59     acpidev_class_list_t **aod_class_list;
60     acpidev_board_type_t aod_bdtype; /* Type of board. */

```

new/usr/src/uts/i86pc/sys/acpidev\_impl.h

2

```
61     uint32_t          aod_bdnm;      /* Board # for DR. */
62     uint32_t          aod_portid;     /* Port id for DR. */
63     uint32_t          aod_bdid;       /* Index # of AP */
64     volatile uint32_t aod_chidx;      /* Index # of child */
65     uint32_t          aod_memidx;     /* Index # of memory */
66     acpidev_class_id_t aod_class_id; /* Dev type for DR. */
67 };
    unchanged_portion_omitted

```

new/usr/src/uts/i86pc/sys/acpidev\_rsc.h

1

\*\*\*\*\*

7693 Fri Jan 17 07:31:41 2014

new/usr/src/uts/i86pc/sys/acpidev\_rsc.h

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 2009 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25 */
26 /*
27  * Copyright (c) 2009, Intel Corporation.
28  * All rights reserved.
29 */

31 #ifndef _SYS_ACPIDEV_RSC_H
32 #define _SYS_ACPIDEV_RSC_H
33 #include <sys/types.h>
34 #include <sys/obpdefs.h>
35 #include <sys/sunddi.h>
36 #include <acpica/include/acpi.h>
36 #include <sys/acpi/acpi.h>
37 #include <sys/acpica.h>

39 #ifdef __cplusplus
40 extern "C" {
41 #endif

43 /* ACPI bus range structure. */
44 typedef struct acpidev_bus_range {
45     uint_t bus_start;
46     uint_t bus_end;
47 } acpidev_bus_range_t;
48 unchanged portion omitted
```

new/usr/src/uts/i86pc/sys/cpu\_acpi.h

1

```
*****
7156 Fri Jan 17 07:31:41 2014
new/usr/src/uts/i86pc/sys/cpu_acpi.h
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 */

26 #ifndef _CPU ACPI_H
27 #define _CPU ACPI_H

29 #include <sys/cpuvar.h>

31 #include <acpica/include/acpi.h>
32 #include <acpica/include/accommon.h>
33 #include <acpica/include/acresrc.h>
34 #include <sys/acpi/acpi.h>
35 #include <sys/acpi/accommon.h>
36 #include <sys/acpi/acresrc.h>
37 #include <sys/acpica.h>

38 #ifdef __cplusplus
39 extern "C" {
40 #endif

41 /*
42  * P-state related macros
43  */
44 #define CPU ACPI_PPC(sp) sp->cs_ppc
45 #define CPU ACPI_PSD(sp) sp->cs_psd
46 #define CPU ACPI_PCT(sp) sp->cs_pct
47 #define CPU ACPI_PCT_CTRL(sp) &sp->cs_pct[0]
48 #define CPU ACPI_PCT_STATUS(sp) &sp->cs_pct[1]
49 #define CPU ACPI_PSTATES(sp) sp->cs_pstates.ss_states
50 #define CPU ACPI_PSTATES_COUNT(sp) sp->cs_pstates.ss_count

51 #define CPU ACPI_FREQ(pstate) pstate->ps_freq
52 #define CPU ACPI_PSTATE_TRANSLAT(pstate) pstate->ps_translat
53 #define CPU ACPI_PSTATE_CTRL(pstate) pstate->ps_ctrl

54 /*
55  * T-state related macros
56  */
57 #define CPU ACPI_TPC(sp) sp->cs_tpc
```

new/usr/src/uts/i86pc/sys/cpu\_acpi.h

2

```
59 #define CPU ACPI_TSD(sp) sp->cs_tsd
60 #define CPU ACPI_PTC(sp) sp->cs_ptc
61 #define CPU ACPI_PTC_CTRL(sp) &sp->cs_ptc[0]
62 #define CPU ACPI_PTC_STATUS(sp) &sp->cs_ptc[1]
63 #define CPU ACPI_TSTATES(sp) sp->cs_tstates.ss_states
64 #define CPU ACPI_TSTATES_COUNT(sp) sp->cs_tstates.ss_count

66 #define CPU ACPI_FREQPER(tstate) tstate->ts_freqper
67 #define CPU ACPI_TSTATE_TRANSLAT(tstate) tstate->ts_translat
68 #define CPU ACPI_TSTATE_CTRL(tstate) tstate->ts_ctrl
69 #define CPU ACPI_TSTATE_STAT(tstate) tstate->ts_state

71 /*
72  * C-state related macros
73  */
74 #define CPU ACPI_CSD(sp) sp->cs_csd
75 #define CPU ACPI_CSTATES(sp) sp->cs_cstates.ss_states
76 #define CPU ACPI_CSTATES_COUNT(sp) sp->cs_cstates.ss_count

78 #define CPU ACPI_NONE_CACHED 0x0000
79 #define CPU ACPI_PCT_CACHED 0x0001
80 #define CPU ACPI_PSS_CACHED 0x0002
81 #define CPU ACPI_PSD_CACHED 0x0004
82 #define CPU ACPI_PPC_CACHED 0x0008
83 #define CPU ACPI_PTC_CACHED 0x0010
84 #define CPU ACPI_TSS_CACHED 0x0020
85 #define CPU ACPI_TSD_CACHED 0x0040
86 #define CPU ACPI_TPC_CACHED 0x0080
87 #define CPU ACPI_CST_CACHED 0x0100
88 #define CPU ACPI_CSD_CACHED 0x0200

90 #define CPU ACPI_IS_OBJ_CACHED(sp, obj) (sp->cpu_acpi_cached & obj)
91 #define CPU ACPI_OBJ_IS_CACHED(sp, obj) (sp->cpu_acpi_cached != obj)
92 #define CPU ACPI_OBJ_IS_NOT_CACHED(sp, obj) (sp->cpu_acpi_cached &= ~obj)

94 #define CPU ACPI_PSTATES_SIZE(cnt) (cnt * sizeof (cpu_acpi_pstate_t))
95 #define CPU ACPI_PSS_CNT (sizeof (cpu_acpi_pstate_t) / sizeof (uint32_t))
96 #define CPU ACPI_TSTATES_SIZE(cnt) (cnt * sizeof (cpu_acpi_tstate_t))
97 #define CPU ACPI_TSS_CNT (sizeof (cpu_acpi_tstate_t) / sizeof (uint32_t))
98 #define CPU ACPI_CSTATES_SIZE(cnt) (cnt * sizeof (cpu_acpi_cstate_t))
99 #define CPU ACPI_CST_CNT (sizeof (cpu_acpi_cstate_t) / sizeof (uint32_t))
100 /*
101  * CPU Domain Coordination Types
102  */
103 #define CPU ACPI_SW_ALL 0xfc
104 #define CPU ACPI_SW_ANY 0xfd
105 #define CPU ACPI_HW_ALL 0xfe

107 /*
108  * Container for ACPI processor state dependency information
109  */
110 typedef struct cpu_acpi_state_dependency
111 {
112     uint8_t sd_entries;
113     uint8_t sd_revision;
114     uint32_t sd_domain;
115     uint32_t sd_type;
116     uint32_t sd_num;
117     uint32_t sd_index;
118 } cpu_acpi_state_dependency_t;
119 #endif
120 #endif
121 #endif
122 #endif
123 #endif
124 #endif
125 #endif
126 #endif
127 #endif
128 #endif
129 #endif
130 #endif
131 #endif
132 #endif
133 #endif
134 #endif
135 #endif
136 #endif
137 #endif
138 #endif
139 #endif
140 #endif
141 #endif
142 #endif
143 #endif
144 #endif
145 #endif
146 #endif
147 #endif
148 #endif
149 #endif
150 #endif
151 #endif
152 #endif
153 #endif
154 #endif
155 #endif
156 #endif
157 #endif
158 #endif
159 #endif
160 #endif
161 #endif
162 #endif
163 #endif
164 #endif
165 #endif
166 #endif
167 #endif
168 #endif
169 #endif
170 #endif
171 #endif
172 #endif
173 #endif
174 #endif
175 #endif
176 #endif
177 #endif
178 #endif
179 #endif
180 #endif
181 #endif
182 #endif
183 #endif
184 #endif
185 #endif
186 #endif
187 #endif
188 #endif
189 #endif
190 #endif
191 #endif
192 #endif
193 #endif
194 #endif
195 #endif
196 #endif
197 #endif
198 #endif
199 #endif
200 #endif
201 #endif
202 #endif
203 #endif
204 #endif
205 #endif
206 #endif
207 #endif
208 #endif
209 #endif
210 #endif
211 #endif
212 #endif
213 #endif
214 #endif
215 #endif
216 #endif
217 #endif
218 #endif
219 #endif
220 #endif
221 #endif
222 #endif
223 #endif
224 #endif
225 #endif
226 #endif
227 #endif
228 #endif
229 #endif
230 #endif
231 #endif
232 #endif
233 #endif
234 #endif
235 #endif
236 #endif
237 #endif
238 #endif
239 #endif
240 #endif
241 #endif
242 #endif
243 #endif
244 #endif
245 #endif
246 #endif
247 #endif
248 #endif
249 #endif
250 #endif
251 #endif
252 #endif
253 #endif
254 #endif
255 #endif
256 #endif
257 #endif
258 #endif
259 #endif
260 #endif
261 #endif
262 #endif
263 #endif
264 #endif
265 #endif
266 #endif
267 #endif
268 #endif
269 #endif
270 #endif
271 #endif
272 #endif
273 #endif
274 #endif
275 #endif
276 #endif
277 #endif
278 #endif
279 #endif
280 #endif
281 #endif
282 #endif
283 #endif
284 #endif
285 #endif
286 #endif
287 #endif
288 #endif
289 #endif
290 #endif
291 #endif
292 #endif
293 #endif
294 #endif
295 #endif
296 #endif
297 #endif
298 #endif
299 #endif
300 #endif
301 #endif
302 #endif
303 #endif
304 #endif
305 #endif
306 #endif
307 #endif
308 #endif
309 #endif
310 #endif
311 #endif
312 #endif
313 #endif
314 #endif
315 #endif
316 #endif
317 #endif
318 #endif
319 #endif
320 #endif
321 #endif
322 #endif
323 #endif
324 #endif
325 #endif
326 #endif
327 #endif
328 #endif
329 #endif
330 #endif
331 #endif
332 #endif
333 #endif
334 #endif
335 #endif
336 #endif
337 #endif
338 #endif
339 #endif
340 #endif
341 #endif
342 #endif
343 #endif
344 #endif
345 #endif
346 #endif
347 #endif
348 #endif
349 #endif
350 #endif
351 #endif
352 #endif
353 #endif
354 #endif
355 #endif
356 #endif
357 #endif
358 #endif
359 #endif
360 #endif
361 #endif
362 #endif
363 #endif
364 #endif
365 #endif
366 #endif
367 #endif
368 #endif
369 #endif
370 #endif
371 #endif
372 #endif
373 #endif
374 #endif
375 #endif
376 #endif
377 #endif
378 #endif
379 #endif
380 #endif
381 #endif
382 #endif
383 #endif
384 #endif
385 #endif
386 #endif
387 #endif
388 #endif
389 #endif
390 #endif
391 #endif
392 #endif
393 #endif
394 #endif
395 #endif
396 #endif
397 #endif
398 #endif
399 #endif
400 #endif
401 #endif
402 #endif
403 #endif
404 #endif
405 #endif
406 #endif
407 #endif
408 #endif
409 #endif
410 #endif
411 #endif
412 #endif
413 #endif
414 #endif
415 #endif
416 #endif
417 #endif
418 #endif
419 #endif
420 #endif
421 #endif
422 #endif
423 #endif
424 #endif
425 #endif
426 #endif
427 #endif
428 #endif
429 #endif
430 #endif
431 #endif
432 #endif
433 #endif
434 #endif
435 #endif
436 #endif
437 #endif
438 #endif
439 #endif
440 #endif
441 #endif
442 #endif
443 #endif
444 #endif
445 #endif
446 #endif
447 #endif
448 #endif
449 #endif
450 #endif
451 #endif
452 #endif
453 #endif
454 #endif
455 #endif
456 #endif
457 #endif
458 #endif
459 #endif
460 #endif
461 #endif
462 #endif
463 #endif
464 #endif
465 #endif
466 #endif
467 #endif
468 #endif
469 #endif
470 #endif
471 #endif
472 #endif
473 #endif
474 #endif
475 #endif
476 #endif
477 #endif
478 #endif
479 #endif
480 #endif
481 #endif
482 #endif
483 #endif
484 #endif
485 #endif
486 #endif
487 #endif
488 #endif
489 #endif
490 #endif
491 #endif
492 #endif
493 #endif
494 #endif
495 #endif
496 #endif
497 #endif
498 #endif
499 #endif
500 #endif
501 #endif
502 #endif
503 #endif
504 #endif
505 #endif
506 #endif
507 #endif
508 #endif
509 #endif
510 #endif
511 #endif
512 #endif
513 #endif
514 #endif
515 #endif
516 #endif
517 #endif
518 #endif
519 #endif
520 #endif
521 #endif
522 #endif
523 #endif
524 #endif
525 #endif
526 #endif
527 #endif
528 #endif
529 #endif
530 #endif
531 #endif
532 #endif
533 #endif
534 #endif
535 #endif
536 #endif
537 #endif
538 #endif
539 #endif
540 #endif
541 #endif
542 #endif
543 #endif
544 #endif
545 #endif
546 #endif
547 #endif
548 #endif
549 #endif
550 #endif
551 #endif
552 #endif
553 #endif
554 #endif
555 #endif
556 #endif
557 #endif
558 #endif
559 #endif
560 #endif
561 #endif
562 #endif
563 #endif
564 #endif
565 #endif
566 #endif
567 #endif
568 #endif
569 #endif
570 #endif
571 #endif
572 #endif
573 #endif
574 #endif
575 #endif
576 #endif
577 #endif
578 #endif
579 #endif
580 #endif
581 #endif
582 #endif
583 #endif
584 #endif
585 #endif
586 #endif
587 #endif
588 #endif
589 #endif
590 #endif
591 #endif
592 #endif
593 #endif
594 #endif
595 #endif
596 #endif
597 #endif
598 #endif
599 #endif
600 #endif
601 #endif
602 #endif
603 #endif
604 #endif
605 #endif
606 #endif
607 #endif
608 #endif
609 #endif
610 #endif
611 #endif
612 #endif
613 #endif
614 #endif
615 #endif
616 #endif
617 #endif
618 #endif
619 #endif
620 #endif
621 #endif
622 #endif
623 #endif
624 #endif
625 #endif
626 #endif
627 #endif
628 #endif
629 #endif
630 #endif
631 #endif
632 #endif
633 #endif
634 #endif
635 #endif
636 #endif
637 #endif
638 #endif
639 #endif
640 #endif
641 #endif
642 #endif
643 #endif
644 #endif
645 #endif
646 #endif
647 #endif
648 #endif
649 #endif
650 #endif
651 #endif
652 #endif
653 #endif
654 #endif
655 #endif
656 #endif
657 #endif
658 #endif
659 #endif
660 #endif
661 #endif
662 #endif
663 #endif
664 #endif
665 #endif
666 #endif
667 #endif
668 #endif
669 #endif
670 #endif
671 #endif
672 #endif
673 #endif
674 #endif
675 #endif
676 #endif
677 #endif
678 #endif
679 #endif
680 #endif
681 #endif
682 #endif
683 #endif
684 #endif
685 #endif
686 #endif
687 #endif
688 #endif
689 #endif
690 #endif
691 #endif
692 #endif
693 #endif
694 #endif
695 #endif
696 #endif
697 #endif
698 #endif
699 #endif
700 #endif
701 #endif
702 #endif
703 #endif
704 #endif
705 #endif
706 #endif
707 #endif
708 #endif
709 #endif
710 #endif
711 #endif
712 #endif
713 #endif
714 #endif
715 #endif
716 #endif
717 #endif
718 #endif
719 #endif
720 #endif
721 #endif
722 #endif
723 #endif
724 #endif
725 #endif
726 #endif
727 #endif
728 #endif
729 #endif
730 #endif
731 #endif
732 #endif
733 #endif
734 #endif
735 #endif
736 #endif
737 #endif
738 #endif
739 #endif
740 #endif
741 #endif
742 #endif
743 #endif
744 #endif
745 #endif
746 #endif
747 #endif
748 #endif
749 #endif
750 #endif
751 #endif
752 #endif
753 #endif
754 #endif
755 #endif
756 #endif
757 #endif
758 #endif
759 #endif
760 #endif
761 #endif
762 #endif
763 #endif
764 #endif
765 #endif
766 #endif
767 #endif
768 #endif
769 #endif
770 #endif
771 #endif
772 #endif
773 #endif
774 #endif
775 #endif
776 #endif
777 #endif
778 #endif
779 #endif
780 #endif
781 #endif
782 #endif
783 #endif
784 #endif
785 #endif
786 #endif
787 #endif
788 #endif
789 #endif
790 #endif
791 #endif
792 #endif
793 #endif
794 #endif
795 #endif
796 #endif
797 #endif
798 #endif
799 #endif
800 #endif
801 #endif
802 #endif
803 #endif
804 #endif
805 #endif
806 #endif
807 #endif
808 #endif
809 #endif
810 #endif
811 #endif
812 #endif
813 #endif
814 #endif
815 #endif
816 #endif
817 #endif
818 #endif
819 #endif
820 #endif
821 #endif
822 #endif
823 #endif
824 #endif
825 #endif
826 #endif
827 #endif
828 #endif
829 #endif
830 #endif
831 #endif
832 #endif
833 #endif
834 #endif
835 #endif
836 #endif
837 #endif
838 #endif
839 #endif
840 #endif
841 #endif
842 #endif
843 #endif
844 #endif
845 #endif
846 #endif
847 #endif
848 #endif
849 #endif
850 #endif
851 #endif
852 #endif
853 #endif
854 #endif
855 #endif
856 #endif
857 #endif
858 #endif
859 #endif
860 #endif
861 #endif
862 #endif
863 #endif
864 #endif
865 #endif
866 #endif
867 #endif
868 #endif
869 #endif
870 #endif
871 #endif
872 #endif
873 #endif
874 #endif
875 #endif
876 #endif
877 #endif
878 #endif
879 #endif
880 #endif
881 #endif
882 #endif
883 #endif
884 #endif
885 #endif
886 #endif
887 #endif
888 #endif
889 #endif
890 #endif
891 #endif
892 #endif
893 #endif
894 #endif
895 #endif
896 #endif
897 #endif
898 #endif
899 #endif
900 #endif
901 #endif
902 #endif
903 #endif
904 #endif
905 #endif
906 #endif
907 #endif
908 #endif
909 #endif
910 #endif
911 #endif
912 #endif
913 #endif
914 #endif
915 #endif
916 #endif
917 #endif
918 #endif
919 #endif
920 #endif
921 #endif
922 #endif
923 #endif
924 #endif
925 #endif
926 #endif
927 #endif
928 #endif
929 #endif
930 #endif
931 #endif
932 #endif
933 #endif
934 #endif
935 #endif
936 #endif
937 #endif
938 #endif
939 #endif
940 #endif
941 #endif
942 #endif
943 #endif
944 #endif
945 #endif
946 #endif
947 #endif
948 #endif
949 #endif
950 #endif
951 #endif
952 #endif
953 #endif
954 #endif
955 #endif
956 #endif
957 #endif
958 #endif
959 #endif
960 #endif
961 #endif
962 #endif
963 #endif
964 #endif
965 #endif
966 #endif
967 #endif
968 #endif
969 #endif
970 #endif
971 #endif
972 #endif
973 #endif
974 #endif
975 #endif
976 #endif
977 #endif
978 #endif
979 #endif
980 #endif
981 #endif
982 #endif
983 #endif
984 #endif
985 #endif
986 #endif
987 #endif
988 #endif
989 #endif
990 #endif
991 #endif
992 #endif
993 #endif
994 #endif
995 #endif
996 #endif
997 #endif
998 #endif
999 #endif
1000 #endif
```

new/usr/src/uts/i86pc/sys/hpet\_acpi.h

1

```
*****
9810 Fri Jan 17 07:31:42 2014
new/usr/src/uts/i86pc/sys/hpet_acpi.h
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) 2009, 2010, Oracle and/or its affiliates. All rights reserved.
23 */

25 #ifndef _HPET_ACPI_H
26 #define _HPET_ACPI_H

28 #if defined(_KERNEL)
29 #include <acpica/include/acpi.h>
30 #include <acpica/include/actbl1.h>
29 #include <sys/acpi/acpi.h>
30 #include <sys/acpi/actbl1.h>
31 #include <sys/acpica.h>
32 #endif /* defined(_KERNEL) */

34 #ifdef __cplusplus
35 extern "C" {
36 #endif

38 /*
39 * Solaris uses an HPET Timer to generate interrupts for CPUs in Deep C-state
40 * with stalled LAPIC Timers. All CPUs use one HPET timer. The timer's
41 * interrupt targets one CPU (via the I/O APIC). The one CPU that receives
42 * the HPET's interrupt wakes up other CPUs as needed during the HPET Interrupt
43 * Service Routing. The HPET ISR uses poke_cpus to wake up other CPUs with an
44 * Inter Processor Interrupt.
45 *
46 * Please see the Intel Programmer's guides. Interrupts are disabled before
47 * a CPU Halts into Deep C-state. (This allows CPU-hardware-specific cleanup
48 * before servicing interrupts.) When a Deep C-state CPU wakes up (due to
49 * an externally generated interrupt), it resume execution where it halted.
50 * The CPU returning from Deep C-state must enable interrupts before it will
51 * handle the pending interrupt that woke it from Deep C-state.
52 *
53 *
54 * HPET bits as defined in the Intel IA-PC HPET Specification Rev 1.0a.
55 *
56 * The physical address space layout of the memory mapped HPET looks like this:
57 *
58 * struct hpet {
59 *     uint64_t         gen_cap;
```

new/usr/src/uts/i86pc/sys/hpet\_acpi.h

2

```
60 *     uint64_t         res1;
61 *     uint64_t         gen_config;
62 *     uint64_t         res2;
63 *     uint64_t         gen_inter_stat;
64 *     uint64_t         res3;
65 *     uint64_t         main_counter_value;
66 *     uint64_t         res4;
67 *     struct hpet_timer {
68 *         uint64_t         config_and_capability;
69 *         uint64_t         comparator_value;
70 *         uint64_t         FSB_interrupt_route;
71 *         uint64_t         reserved;
72 *     } timers[32];
73 * }
74 *
75 * There are 32 possible timers in an hpet. Only the first 3 timers are
76 * required. The other 29 timers are optional.
77 *
78 * HPETs can have 64-bit or 32-bit timers. Timers/compare registers can
79 * be 64-bit or 32-bit and can be a mixture of both.
80 * The first two timers are not used. The HPET spec intends the first two
81 * timers to be used as "legacy replacement" for the PIT and RTC timers.
82 *
83 * Solaris uses the first available non-legacy replacement timer as a proxy
84 * timer for processor Local APIC Timers that stop in deep idle C-states.
85 */

87 /*
88 * We only use HPET table 1 on x86. Typical x86 systems only have 1 HPET.
89 * ACPI allows for multiple HPET tables to describe multiple HPETs.
90 */
91 #define HPET_TABLE_1                (1)

93 /*
94 * HPET Specification 1.0a defines the HPET to occupy 1024 bytes regardless of
95 * the number of counters (3 to 32) in this implementation.
96 */
97 #define HPET_SIZE                    (1024)

99 /*
100 * Offsets of hpet registers and macros to access them from HPET base address.
101 */
102 #define HPET_GEN_CAP_OFFSET          (0)
103 #define HPET_GEN_CONFIG_OFFSET       (0x10)
104 #define HPET_GEN_INTR_STAT_OFFSET    (0x20)
105 #define HPET_MAIN_COUNTER_OFFSET     (0xF0)
106 #define HPET_TIMER_N_CONF_OFFSET(n)  (0x100 + (n * 0x20))
107 #define HPET_TIMER_N_COMP_OFFSET(n)  (0x108 + (n * 0x20))

109 #define OFFSET_ADDR(a, o)             (((uintptr_t)(a)) + (o))
110 #define HPET_GEN_CAP_ADDRESS(la)      \
111     OFFSET_ADDR(la, HPET_GEN_CAP_OFFSET)
112 #define HPET_GEN_CONFIG_ADDRESS(la)   \
113     OFFSET_ADDR(la, HPET_GEN_CONFIG_OFFSET)
114 #define HPET_GEN_INTR_STAT_ADDRESS(la) \
115     OFFSET_ADDR(la, HPET_GEN_INTR_STAT_OFFSET)
116 #define HPET_MAIN_COUNTER_ADDRESS(la) \
117     OFFSET_ADDR(la, HPET_MAIN_COUNTER_OFFSET)
118 #define HPET_TIMER_N_CONF_ADDRESS(la, n) \
119     OFFSET_ADDR(la, HPET_TIMER_N_CONF_OFFSET(n))
120 #define HPET_TIMER_N_COMP_ADDRESS(la, n) \
121     OFFSET_ADDR(la, HPET_TIMER_N_COMP_OFFSET(n))

123 /*
124 * HPET General Capabilities and ID Register
125 */
```

```
126 typedef struct hpet_gen_cap {
127     uint32_t      counter_clk_period; /* period in femtoseconds */
128     uint32_t      vendor_id           /* vendor */
129     uint32_t      leg_route_cap       /* 1=LegacyReplacemnt support */
130     uint32_t      res1                /* reserved */
131     uint32_t      count_size_cap      /* 0=32bit, 1=64bit wide */
132     uint32_t      num_tim_cap         /* number of timers -1 */
133     uint32_t      rev_id              /* revision number */
134 } hpet_gen_cap_t;
_____unchanged_portion_omitted_____
```

new/usr/src/uts/i86pc/sys/psm\_common.h

1

\*\*\*\*\*

4042 Fri Jan 17 07:31:42 2014

new/usr/src/uts/i86pc/sys/psm\_common.h

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 2007 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  * Copyright 2013 Nexenta Systems, Inc. All rights reserved.
26 */
```

```
28 #ifndef _SYS_PSM_COMMON_H
29 #define _SYS_PSM_COMMON_H
```

```
29 #pragma ident      "%Z%M% %I%      %E% SMI"
```

```
31 #include <sys/types.h>
32 #include <sys/param.h>
33 #include <sys/cmn_err.h>
```

```
35 #include <acpica/include/acpi.h>
34 #include <sys/acpi/acpi.h>
36 #include <sys/acpica.h>
```

```
38 #include <sys/sunddi.h>
39 #include <sys/ddi.h>
40 #include <sys/ddi_impldefs.h>
41 #include <sys/pci.h>
42 #include <sys/debug.h>
```

```
44 #ifdef __cplusplus
45 extern "C" {
46 #endif
```

```
49 /* private data used in psm_common */
50 typedef struct acpi_prs_private {
51     uchar_t      prs_irqflags;
52     uchar_t      prs_type;
53 } acpi_prs_private_t;
54 _____unchanged_portion_omitted_____
```

new/usr/src/uts/i86xpv/io/psm/mp\_platform\_xpv.c

1

```
*****
62027 Fri Jan 17 07:31:42 2014
new/usr/src/uts/i86xpv/io/psm/mp_platform_xpv.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) 2010, Oracle and/or its affiliates. All rights reserved.
23 */
24 /*
25 * Copyright (c) 2010, Intel Corporation.
26 * All rights reserved.
27 */

29 /*
30 * PSMI 1.1 extensions are supported only in 2.6 and later versions.
31 * PSMI 1.2 extensions are supported only in 2.7 and later versions.
32 * PSMI 1.3 and 1.4 extensions are supported in Solaris 10.
33 * PSMI 1.5 extensions are supported in Solaris Nevada.
34 * PSMI 1.6 extensions are supported in Solaris Nevada.
35 * PSMI 1.7 extensions are supported in Solaris Nevada.
36 */
37 #define PSMI_1_7

39 #include <sys/processor.h>
40 #include <sys/time.h>
41 #include <sys/psm.h>
42 #include <sys/smp_impldefs.h>
43 #include <sys/cram.h>
44 #include <acpica/include/acpi.h>
44 #include <sys/acpi/acpi.h>
45 #include <sys/acpica.h>
46 #include <sys/psm_common.h>
47 #include <sys/apic.h>
48 #include <sys/apic_common.h>
49 #include <sys/pit.h>
50 #include <sys/ddi.h>
51 #include <sys/sunddi.h>
52 #include <sys/ddi_impldefs.h>
53 #include <sys/pci.h>
54 #include <sys/promif.h>
55 #include <sys/x86_archext.h>
56 #include <sys/cpc_impl.h>
57 #include <sys/uadmin.h>
58 #include <sys/panic.h>
59 #include <sys/debug.h>
60 #include <sys/archsystem.h>
```

new/usr/src/uts/i86xpv/io/psm/mp\_platform\_xpv.c

2

```
61 #include <sys/trap.h>
62 #include <sys/machsystem.h>
63 #include <sys/cpuvar.h>
64 #include <sys/rm_platter.h>
65 #include <sys/privregs.h>
66 #include <sys/cyclic.h>
67 #include <sys/note.h>
68 #include <sys/pci_intr_lib.h>
69 #include <sys/sunndi.h>

72 /*
73  * Local Function Prototypes
74 */
75 static void apic_mark_vector(uchar_t oldvector, uchar_t newvector);
76 static void apic_xlate_vector_free_timeout_handler(void *arg);
77 static int apic_check_stuck_interrupt(apic_irq_t *irq_ptr, int old_bind_cpu,
78 int new_bind_cpu, int apicindex, int intin_no, int which_irq,
79 struct ioapic_reprogram_data *drep);
80 static int apic_setup_irq_table(dev_info_t *dip, int irqno,
81 struct apic_io_intr *intrp, struct intrspec *ispec, iflag_t *intr_flagp,
82 int type);
83 static void apic_try_deferred_reprogram(int ipl, int vect);
84 static void delete_defer_repro_ent(int which_irq);
85 static void apic_ioapic_wait_pending_clear(int ioapicindex,
86 int intin_no);

88 extern int apic_acpi_translate_pci_irq(dev_info_t *dip, int busid, int devid,
89 int ipin, int *pci_irqp, iflag_t *intr_flagp);
90 extern int apic_handle_pci_pci_bridge(dev_info_t *idip, int child_devno,
91 int child_ipin, struct apic_io_intr **intrp);
92 extern uchar_t acpi_find_ioapic(int irq);
93 extern struct apic_io_intr *apic_find_io_intr_w_busid(int irqno, int busid);
94 extern int apic_find_bus_id(int bustype);
95 extern int apic_find_intin(uchar_t ioapic, uchar_t intin);
96 extern void apic_record_rdt_entry(apic_irq_t *irqptr, int irq);

98 extern int apic_sci_vect;
99 extern iflag_t apic_sci_flags;
100 extern int apic_intr_policy;
101 extern char *psm_name;

103 /*
104  * number of bits per byte, from <sys/param.h>
105 */
106 #define UCHAR_MAX ((1 << NBBY) - 1)

108 /* Max wait time (in repetitions) for flags to clear in an RDT entry. */
109 extern int apic_max_reps_clear_pending;

111 /* The irq # is implicit in the array index: */
112 struct ioapic_reprogram_data apic_reprogram_info[APIC_MAX_VECTOR+1];
113 /*
114  * APIC_MAX_VECTOR + 1 is the maximum # of IRQs as well. ioapic_reprogram_info
115  * is indexed by IRQ number, NOT by vector number.
116 */

118 extern int apic_int_busy_mark;
119 extern int apic_int_free_mark;
120 extern int apic_diff_for_redistribution;
121 extern int apic_sample_factor_redistribution;
122 extern int apic_redist_cpu_skip;
123 extern int apic_num_imbalance;
124 extern int apic_num_rebind;

126 /* 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



new/usr/src/uts/intel/Makefile.files

1

```
*****
7388 Fri Jan 17 07:31:43 2014
new/usr/src/uts/intel/Makefile.files
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 #
21 #
22 #
23 # Copyright (c) 1999, 2010, Oracle and/or its affiliates. All rights reserved.
24 # Copyright (c) 2012, Joyent, Inc. All rights reserved.
25 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
26 #
27 #
28 #
29 # This Makefile defines all file modules and build rules for the
30 # directory uts/intel and its children. These are the source files which
31 # are specific to x86 processor architectures.
32 #
33 #
34 #
35 # Core (unix) objects
36 #
37 CORE_OBJS += \
38     arch_kdi.o \
39     copy.o \
40     copy_subr.o \
41     cpc_subr.o \
42     ddi_arch.o \
43     ddi_i86.o \
44     ddi_i86_asm.o \
45     desctbls.o \
46     desctbls_asm.o \
47     exception.o \
48     float.o \
49     fmsmb.o \
50     fpu.o \
51     i86_subr.o \
52     lock_prim.o \
53     ovbcopy.o \
54     polled_io.o \
55     sseblk.o \
56     sundep.o \
57     swtch.o \
58     sysi86.o
59 #
60 #
```

new/usr/src/uts/intel/Makefile.files

2

```
61 # 64-bit multiply/divide compiler helper routines
62 # used only for ia32
63 #
64 #
65 SPECIAL_OBJS += \
66     muldiv.o
67 #
68 #
69 # Generic-unix Module
70 #
71 GENUNIX_OBJS += \
72     archdep.o \
73     getcontext.o \
74     install_utrap.o \
75     lwp_private.o \
76     prom_enter.o \
77     prom_exit.o \
78     prom_panic.o \
79     sendsig.o \
80     syscall.o
81 #
82 #
83 #
84 # PROM Routines
85 #
86 GENUNIX_OBJS += \
87     prom_env.o \
88     prom_emul.o \
89     prom_getchar.o \
90     prom_init.o \
91     prom_node.o \
92     prom_printf.o \
93     prom_prop.o \
94     prom_putchar.o \
95     prom_reboot.o \
96     prom_version.o
97 #
98 #
99 # file system modules
100 #
101 CORE_OBJS += \
102     prmachdep.o
103 #
104 #
105 # ZFS file system module
106 #
107 ZFS_OBJS += \
108     spa_boot.o
109 #
110 #
111 # Decompression code
112 #
113 CORE_OBJS += decompress.o
114 #
115 #
116 # Microcode utilities
117 #
118 CORE_OBJS += ucode_utils.o
119 #
120 #
121 # Driver modules
122 #
123 AGPGART_OBJS += agpgart.o agp_kstat.o
124 AGPTARGET_OBJS += agptarget.o
125 AMD64GART_OBJS += amd64_gart.o
126 ARCMSR_OBJS += arcmsr.o
```

```

127 ATA_OBJS += $(GHD_OBJS) ata_blacklist.o ata_common.o ata_disk.o \
128     ata_dma.o atapi.o atapi_fsm.o ata_debug.o \
129     sil3xxx.o
130 BSCBUS_OBJS += bscbus.o
131 BSCV_OBJS += bscv.o
132 CMDK_OBJS += cmdk.o
133 CMLB_OBJS += cmlb.o
134 CPUNEX_OBJS += cpunex.o
135 DADK_OBJS += dadk.o
136 DCOPY_OBJS += dcopy.o
137 DNET_OBJS += dnet.o dnet_mii.o
138 FD_OBJS += fd.o
139 GDA_OBJS += gda.o
140 GHD_OBJS += ghd.o ghd_debug.o ghd_dma.o ghd_queue.o ghd_scscs.o \
141     ghd_scsci.o ghd_timer.o ghd_waitq.o ghd_gcmd.o
142 I915_OBJS += i915_dma.o i915_drv.o i915_irq.o i915_mem.o \
143     i915_gem.o i915_gem_debug.o i915_gem_tiling.o
144 NSKERN_OBJS += nsc_asm.o
145 PCICFG_OBJS += pcicfg.o
146 PCI_PCINEXUS_OBJS += pci_pci.o
147 PCIEB_OBJS += pcieb_x86.o
148 PIT_BEEP_OBJS += pit_bEEP.o
149 POWER_OBJS += power.o
150 PCI_AUTOCONFIG_OBJS += pci_autoconfig.o pci_boot.o pcie_nvidia.o \
151     pci_memlist.o pci_resource.o
152 RADEON_OBJS += r300_cmdbuf.o radeon_cp.o radeon_drv.o \
153     radeon_state.o radeon_irq.o radeon_mem.o
154 SD_OBJS += sd.o sd_xbuf.o

156 HECI_OBJS += \
157     heci_init.o \
158     heci_intr.o \
159     heci_interface.o \
160     io_heci.o \
161     heci_main.o

163 STRATEGY_OBJS += strategy.o
164 UCODE_OBJS += ucode_drv.o
165 VGATEXT_OBJS += vgatext.o vgasubr.o

167 #
168 #       Kernel linker
169 #
170 KRTLD_OBJS += \
171     bootrd.o \
172     ufsops.o \
173     hsfs.o \
174     doreloc.o \
175     kobj_boot.o \
176     kobj_convrelstr.o \
177     kobj_crt.o \
178     kobj_isa.o \
179     kobj_reloc.o

181 #
182 #       ACPI CA module
181 #       misc. modules
183 #

185 # native
186 ACPICA_OBJS += acpica.o acpica_ec.o acpi_enum.o master_ops.o osl.o \
187     osl_ml.o
188 # debugger
189 ACPICA_OBJS += dbcmds.o dbdisply.o dbexec.o dbfileio.o dbhistory.o \
190     dbinput.o dbmethod.o dbnames.o dbstats.o dbutils.o \
191     dbxface.o dbconvert.o

```

```

192 # disassembler
193 ACPICA_OBJS += dmbuffer.o dmnames.o dmbobject.o dmopcode.o dmresrc.o \
194     dmresrc1.o dmresrcs.o dmutils.o dmwalk.o dmdeferred.o \
195     dmresrc12.o
196 # dispatcher
197 ACPICA_OBJS += dsargs.o dscontrol.o dsfield.o dsinit.o dsmethod.o \
198     ACPICA_OBJS += dbcmds.o dbdisply.o \
199     dbexec.o dbfileio.o dbhistory.o dbinput.o dbstats.o \
200     dbutils.o dbxface.o evevent.o evgpe.o evgpeblk.o \
201     evmisc.o evregion.o evrgnini.o evsci.o evxface.o \
202     evxfevnt.o evxfregn.o hwacpi.o hwgpe.o hwregs.o \
203     hwsleep.o hwtimer.o dsfield.o dsinit.o dsmethod.o \
204     dsmthdat.o dsobject.o dsopcode.o dsutils.o dswexec.o \
205     dswload.o dswload2.o dswscope.o dswstate.o
206 # events
207 ACPICA_OBJS += evevent.o evglock.o evgpe.o evgpeblk.o evgpeinit.o \
208     evgpeutil.o evmisc.o evregion.o evrgnini.o evsci.o \
209     evxface.o evxfevnt.o evxfgpe.o evxfregn.o evhandler.o
210 # executer
211 ACPICA_OBJS += exconfig.o exconvrt.o excreate.o exdebug.o exdump.o \
212     exfield.o exfldio.o exmisc.o exmutex.o exnames.o \
213     exoparg1.o exoparg2.o exoparg3.o exoparg6.o exprep.o \
214     exregion.o exresnte.o exresolv.o exresop.o exstore.o \
215     exstoren.o exstorob.o exsystem.o exutils.o
216 # hardware
217 ACPICA_OBJS += hwacpi.o hwgpe.o hwpci.o hwregs.o hwsleep.o hwtimer.o \
218     hwvalid.o hwxface.o hwesleep.o hwxfsleep.o
219 # namespace
220 ACPICA_OBJS += nsaccess.o nsalloc.o nsdump.o nsdumpdv.o nseval.o \
221     nsinit.o nsload.o nsnames.o nsobject.o nsparse.o \
222     nspredef.o nsrepair.o nsrepair2.o nssearch.o nsutils.o \
223     nswalk.o nsxfeval.o nsxfname.o nsxfobj.o nsarguments.o \
224     nsconvert.o nsprepkg.o
225 # parser
226 ACPICA_OBJS += psargs.o psloop.o psopcode.o psparse.o psscope.o \
227     pstree.o psutils.o pswalk.o psxface.o psobject.o psopinfo.o
228 # resources
229 ACPICA_OBJS += rsaddr.o rscalc.o rscreeate.o rsdump.o rsinfo.o rsio.o \
230     rsirq.o rslst.o rsmemory.o rsmisc.o rsutils.o \
231     rsxface.o rsdumpinfo.o rserial.o
232 # tables
233 ACPICA_OBJS += tbfadt.o tbfnd.o tbinstal.o tbutils.o tbxface.o \
234     tbxfroot.o tbprint.o tbxfload.o
235 # utilities
236 ACPICA_OBJS += utalloc.o utcache.o utclib.o utcopy.o utdebug.o \
237     utdecode.o utdelete.o uteval.o utglobal.o utids.o \
238     utinit.o utlock.o utmath.o utmisc.o utmutex.o \
239     utobject.o utosi.o utresrc.o utstate.o uttrack.o \
240     utxface.o utxferror.o utaddress.o utbuffer.o uterror.o \
241     utexcep.o utownerid.o utpredef.o utstring.o utxfinit.o \
242     utxfmutex.o
243     dswload.o dswscope.o dswstate.o exconfig.o exconvrt.o \
244     excreate.o exdump.o exfield.o exfldio.o exmisc.o \
245     exmutex.o exnames.o exoparg1.o exoparg2.o exoparg3.o \
246     exoparg6.o exprep.o exregion.o exresnte.o exresolv.o \
247     exresop.o exstore.o exstoren.o exstorob.o exsystem.o \
248     exutils.o psargs.o psopcode.o psparse.o psscope.o \
249     pstree.o psutils.o pswalk.o psxface.o nsaccess.o \
250     nsalloc.o nsdump.o nsdumpdv.o nseval.o nsinit.o \
251     nsload.o nsnames.o nsobject.o nsparse.o nssearch.o \
252     nsutils.o nswalk.o nsxfeval.o nsxfname.o nsxfobj.o \
253     rsaddr.o rscalc.o rscreeate.o rsdump.o \
254     rsinfo.o rsio.o rsirq.o rslst.o rsmemory.o rsmisc.o \
255     rsutils.o rsxface.o tbfadt.o tbfnd.o tbinstal.o \
256     tbutils.o tbxface.o tbxfroot.o \
257     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 dmbobject.o \
209         dmopcode.o dmresrc.o dmresrc1.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_OBJS += \
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

```

new/usr/src/uts/intel/Makefile.rules

1

```
*****
13941 Fri Jan 17 07:31:43 2014
new/usr/src/uts/intel/Makefile.rules
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 #
21 #
22 #
23 # Copyright 2009 Sun Microsystems, Inc. All rights reserved.
24 # Use is subject to license terms.
25 # Copyright 2012 Joyent, Inc. All rights reserved.
26 # Copyright 2013 Nexenta Systems, Inc. All rights reserved.
27 #
28 #
29 #
30 # This Makefile defines all file modules and build rules for the
31 # directory uts/intel and its children. These are the source files which
32 # are specific to the intel processor.
33 #
34 # The following two-level ordering must be maintained in this file.
35 # Lines are sorted first in order of decreasing specificity based on
36 # the first directory component. That is, sun4u rules come before
37 # sparc rules come before common rules.
38 #
39 # Lines whose initial directory components are equal are sorted
40 # alphabetically by the remaining components.
41 #
42 #
43 # Need a way to distinguish between the ia32 and amd64 subdirs.
44 #
45 SUBARCH_DIR_32 = ia32
46 SUBARCH_DIR_64 = amd64
47 SUBARCH_DIR = $(SUBARCH_DIR_$(CLASS))
48 #
49 #
50 # Section 1a: C object build rules
51 #
52 $(OBJS_DIR)/%.o: $(SRC)/common/fs/%.c
53 $(COMPILE.c) -o $@ $<
54 $(CTFCONVERT_O)
55 #
56 $(OBJS_DIR)/%.o: $(UTSBASE)/common/io/power/%.c
57 $(COMPILE.c) -o $@ $<
58 $(CTFCONVERT_O)
59 #
60 $(OBJS_DIR)/%.o: $(SRC)/common/util/i386/%.s
```

new/usr/src/uts/intel/Makefile.rules

2

```
61 $(COMPILE.s) -o $@ $<
62 #
63 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/brand/snl/%.s
64 $(COMPILE.s) -o $@ $<
65 #
66 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/brand/solaris10/%.s
67 $(COMPILE.s) -o $@ $<
68 #
69 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/dtrace/%.c
70 $(COMPILE.c) -o $@ $<
71 $(CTFCONVERT_O)
72 #
73 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/dtrace/%.s
74 $(COMPILE.s) -o $@ $<
75 #
76 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/fs/proc/%.c
77 $(COMPILE.c) -o $@ $<
78 $(CTFCONVERT_O)
79 #
80 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/ia32/ml/%.s
81 $(COMPILE.s) -o $@ $<
82 #
83 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/ia32/os/%.c
84 $(COMPILE.c) -o $@ $<
85 $(CTFCONVERT_O)
86 #
87 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/ia32/promif/%.c
88 $(COMPILE.c) -o $@ $<
89 $(CTFCONVERT_O)
90 #
91 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/ia32/syscall/%.c
92 $(COMPILE.c) -o $@ $<
93 $(CTFCONVERT_O)
94 #
95 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/io/%.c
96 $(COMPILE.c) -o $@ $<
97 $(CTFCONVERT_O)
98 #
99 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/io/acpica/%.c
100 $(COMPILE.c) -o $@ $<
101 $(CTFCONVERT_O)
102 #
103 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/io/acpica/%.s
104 $(COMPILE.s) -o $@ $<
105 #
106 $(OBJS_DIR)/%.o: $(SRC)/common/acpica/components/debugger/%.c
107 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/io/acpica/debugger/%.c
108 $(COMPILE.c) -o $@ $<
109 $(CTFCONVERT_O)
110 #
111 $(OBJS_DIR)/%.o: $(SRC)/common/acpica/components/disassembler/%.c
112 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/io/acpica/events/%.c
113 $(COMPILE.c) -o $@ $<
114 $(CTFCONVERT_O)
115 #
116 $(OBJS_DIR)/%.o: $(SRC)/common/acpica/components/dispatcher/%.c
117 $(OBJS_DIR)/%.o: $(UTSBASE)/intel/io/acpica/hardware/%.c
118 $(COMPILE.c) -o $@ $<
119 $(CTFCONVERT_O)
120 #
121 #
122 $(OBJS_DIR)/%.o: $(SRC)/common/acpica/components/executer/%.c
```

## new/usr/src/uts/intel/Makefile.rules

3

```
120 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/acpica/executer/%.c
123 $(COMPILE.c) -o $@ $<
124 $(CTFCONVERT_O)

126 $(OBJSDIR)/%.o: $(SRC)/common/acpica/components/hardware/%.c
124 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/acpica/parser/%.c
127 $(COMPILE.c) -o $@ $<
128 $(CTFCONVERT_O)

130 $(OBJSDIR)/%.o: $(SRC)/common/acpica/components/namespace/%.c
128 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/acpica/namespace/%.c
131 $(COMPILE.c) -o $@ $<
132 $(CTFCONVERT_O)

134 $(OBJSDIR)/%.o: $(SRC)/common/acpica/components/parser/%.c
132 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/acpica/resources/%.c
135 $(COMPILE.c) -o $@ $<
136 $(CTFCONVERT_O)

138 $(OBJSDIR)/%.o: $(SRC)/common/acpica/components/resources/%.c
136 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/acpica/tables/%.c
139 $(COMPILE.c) -o $@ $<
140 $(CTFCONVERT_O)

142 $(OBJSDIR)/%.o: $(SRC)/common/acpica/components/tables/%.c
140 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/acpica/utilities/%.c
143 $(COMPILE.c) -o $@ $<
144 $(CTFCONVERT_O)

146 $(OBJSDIR)/%.o: $(SRC)/common/acpica/components/utilities/%.c
144 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/acpica/disassembler/%.c
147 $(COMPILE.c) -o $@ $<
148 $(CTFCONVERT_O)

150 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/agpgart/%.c
151 $(COMPILE.c) -o $@ $<
152 $(CTFCONVERT_O)

154 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/agpmaster/%.c
155 $(COMPILE.c) -o $@ $<
156 $(CTFCONVERT_O)

158 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/amd8111s/%.c
159 $(COMPILE.c) -o $@ $<
160 $(CTFCONVERT_O)

162 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/amr/%.c
163 $(COMPILE.c) -o $@ $<
164 $(CTFCONVERT_O)

166 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/drm/%.c
167 $(COMPILE.c) -o $@ $<
168 $(CTFCONVERT_O)

170 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/hotplug/pcicfg/%.c
171 $(COMPILE.c) -o $@ $<
172 $(CTFCONVERT_O)

174 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/ipmi/%.c
175 $(COMPILE.c) -o $@ $<
176 $(CTFCONVERT_O)

178 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/intel_nb5000/%.c
179 $(COMPILE.c) -o $@ $<
180 $(CTFCONVERT_O)
```

## new/usr/src/uts/intel/Makefile.rules

4

```
182 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/intel_nhm/%.c
183 $(COMPILE.c) -o $@ $<
184 $(CTFCONVERT_O)

186 $(OBJSDIR)/%.o: $(SRC)/common/mc/mc-amd/%.c
187 $(COMPILE.c) -o $@ $<
188 $(CTFCONVERT_O)

190 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/mc-amd/%.c
191 $(COMPILE.c) -o $@ $<
192 $(CTFCONVERT_O)

194 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/pci/%.c
195 $(COMPILE.c) -o $@ $<
196 $(CTFCONVERT_O)

198 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/pciex/%.c
199 $(COMPILE.c) -o $@ $<
200 $(CTFCONVERT_O)

202 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/dktp/controller/ata/%.c
203 $(COMPILE.c) -o $@ $<
204 $(CTFCONVERT_O)

206 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/dktp/dcdev/%.c
207 $(COMPILE.c) -o $@ $<
208 $(CTFCONVERT_O)

210 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/dktp/disk/%.c
211 $(COMPILE.c) -o $@ $<
212 $(CTFCONVERT_O)

214 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/dktp/drvobj/%.c
215 $(COMPILE.c) -o $@ $<
216 $(CTFCONVERT_O)

218 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/dktp/hba/ghd/%.c
219 $(COMPILE.c) -o $@ $<
220 $(CTFCONVERT_O)

222 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/dnet/%.c
223 $(COMPILE.c) -o $@ $<
224 $(CTFCONVERT_O)

226 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/scsi/adapters/arcmsr/%.c
227 $(COMPILE.c) -o $@ $<
228 $(CTFCONVERT_O)

230 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/scsi/targets/%.c
231 $(COMPILE.c) -o $@ $<
232 $(CTFCONVERT_O)

234 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/vgatext/%.c
235 $(COMPILE.c) -o $@ $<
236 $(CTFCONVERT_O)

238 $(OBJSDIR)/%.o: $(UTSBASE)/intel/io/heci/%.c
239 $(COMPILE.c) -o $@ $<
240 $(CTFCONVERT_O)

242 $(OBJSDIR)/%.o: $(UTSBASE)/intel/nskern/%.s
243 $(COMPILE.s) -o $@ $<

245 $(OBJSDIR)/%.o: $(UTSBASE)/intel/os/%.c
246 $(COMPILE.c) -o $@ $<
247 $(CTFCONVERT_O)
```

```

249 $(OBJSDIR)/%.o: $(UTSBASE)/intel/pcbe/%.c
250     $(COMPILE.c) -o $@ $<
251     $(CTFCONVERT_O)

253 $(OBJSDIR)/%.o: $(UTSBASE)/intel/promif/%.c
254     $(COMPILE.c) -o $@ $<
255     $(CTFCONVERT_O)

257 $(OBJSDIR)/%.o: $(UTSBASE)/intel/syscall/%.c
258     $(COMPILE.c) -o $@ $<
259     $(CTFCONVERT_O)

261 $(OBJSDIR)/%.o: $(UTSBASE)/common/os/%.c
262     $(COMPILE.c) -o $@ $<
263     $(CTFCONVERT_O)

265 $(OBJSDIR)/%.o: $(UTSBASE)/intel/kdi/%.c
266     $(COMPILE.c) -o $@ $<
267     $(CTFCONVERT_O)

269 $(OBJSDIR)/%.o: $(UTSBASE)/intel/kdi/%.s
270     $(COMPILE.s) -o $@ $<

272 $(OBJSDIR)/%.o: $(UTSBASE)/intel/kdi/$(SUBARCH_DIR)/%.s
273     $(COMPILE.s) -o $@ $<

275 $(OBJSDIR)/%.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 = -DELFTARGET_386
287 KRTLD_CPPFLAGS_64 = -DELFTARGET_AMD64 -DMODDIR_SUFFIX=\"amd64\"
288 KRTLD_CPPFLAGS     = $(KRTLD_CPPFLAGS_$(CLASS)) -D_KRTLD

290 $(OBJSDIR)/%.o: $(UTSBASE)/common/krtld/%.c
291     $(COMPILE.c) $(KRTLD_INC_PATH) $(KRTLD_CPPFLAGS) -o $@ $<
292     $(CTFCONVERT_O)

294 $(OBJSDIR)/%.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 $(OBJSDIR)/kobj.o := CPPFLAGS += -D_DBOOT

303 $(OBJSDIR)/%.o: $(UTSBASE)/intel/$(SUBARCH_DIR)/krtld/%.s
304     $(COMPILE.s) $(KRTLD_INC_PATH) $(KRTLD_CPPFLAGS) -o $@ $<
305     $(CTFCONVERT_O)

307 $(OBJSDIR)/%.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 $(LINTSDIR)/%.ln: $(SRC)/common/fs/%.c
316     @$(LHEAD) $(LINT.c) $< $(LTAIL))

318 $(LINTSDIR)/%.ln: $(SRC)/common/util/i386/%.s
319     @$(LHEAD) $(LINT.s) $< $(LTAIL))

321 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/brand/snl/%.s
322     @$(LHEAD) $(LINT.s) $< $(LTAIL))

324 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/brand/solaris10/%.s
325     @$(LHEAD) $(LINT.s) $< $(LTAIL))

327 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/dtrace/%.c
328     @$(LHEAD) $(LINT.c) $< $(LTAIL))

330 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/dtrace/%.s
331     @$(LHEAD) $(LINT.s) $< $(LTAIL))

333 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/zfs/%.c
334     @$(LHEAD) $(LINT.c) $< $(LTAIL))

336 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/fs/proc/%.c
337     @$(LHEAD) $(LINT.c) $< $(LTAIL))

339 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/ia32/ml/%.s
340     @$(LHEAD) $(LINT.s) $< $(LTAIL))

342 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/ia32/os/%.c
343     @$(LHEAD) $(LINT.c) $< $(LTAIL))

345 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/ia32/promif/%.c
346     @$(LHEAD) $(LINT.c) $< $(LTAIL))

348 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/ia32/syscall/%.c
349     @$(LHEAD) $(LINT.c) $< $(LTAIL))

351 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/io/%.c
352     @$(LHEAD) $(LINT.c) $< $(LTAIL))

354 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/io/acpica/%.s
355     @$(LHEAD) $(LINT.s) $< $(LTAIL))

357 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/io/acpica/%.c
358     @$(LHEAD) $(LINT.c) $< $(LTAIL))

360 $(LINTSDIR)/%.ln: $(SRC)/common/acpica/components/debugger/%.c
358 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/io/acpica/debugger/%.c
361     @$(LHEAD) $(LINT.c) $< $(LTAIL))

363 $(LINTSDIR)/%.ln: $(SRC)/common/acpica/components/events/%.c
361 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/io/acpica/events/%.c
364     @$(LHEAD) $(LINT.c) $< $(LTAIL))

366 $(LINTSDIR)/%.ln: $(SRC)/common/acpica/components/hardware/%.c
364 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/io/acpica/hardware/%.c
367     @$(LHEAD) $(LINT.c) $< $(LTAIL))

369 $(LINTSDIR)/%.ln: $(SRC)/common/acpica/components/dispatcher/%.c
367 $(LINTSDIR)/%.ln: $(UTSBASE)/intel/io/acpica/dispatcher/%.c
370     @$(LHEAD) $(LINT.c) $< $(LTAIL))

372 $(LINTSDIR)/%.ln: $(SRC)/common/acpica/components/executer/%.c
370 $(LINTSDIR)/%.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: $(UTSBASE)/intel/io/dktp/drvojb/%.c
448 @$(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 $(LINTSDIR)/%.ln:      $(UTSBASE)/intel/$(SUBARCH_DIR)/krtld/%.s
502     @$(LHEAD) $(LINT.s) $(KRTLD_INC_PATH) $(KRTLD_CPPFLAGS) $< $(LTAIL))

504 $(LINTSDIR)/%.ln:      $(SRC)/common/util/$(SUBARCH_DIR)/%.c
505     @$(LHEAD) $(LINT.c) $(KRTLD_INC_PATH) $(KRTLD_CPPFLAGS) $< $(LTAIL))

507 $(OBSJSDIR)/kobj.ln    := CPPFLAGS += -D_DBOOT
```



new/usr/src/uts/intel/acpica/Makefile

1

```
*****
1595 Fri Jan 17 07:31:44 2014
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 = $(ACPICA_OBJS:%=$(OBJSDIR)/%)
22 LINTS = $(ACPICA_OBJS:%.o=$(LINTSDIR)/%.ln)
23 ROOTMODULE = $(ROOTMISC_DIR)/$(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
```

new/usr/src/uts/intel/acpica/Makefile

2

```
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
```

new/usr/src/uts/intel/io/acpica/acpi\_enum.c

1

\*\*\*\*\*

28791 Fri Jan 17 07:31:44 2014

new/usr/src/uts/intel/io/acpica/acpi\_enum.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 */
```

```
22 /*
22 * Copyright (c) 2012 Gary Mills
23 *
23 * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
24 * Use is subject to license terms.
25 * Copyright (c) 2012 Gary Mills
26 * Copyright 2013 Nexenta Systems, Inc. All rights reserved.
27 */
```

```
29 /*
30 * ACPI enumerator
31 */
```

```
33 #include <sys/ddi.h>
34 #include <sys/sunddi.h>
35 #include <sys/sunndi.h>
36 #include <sys/note.h>
36 #include <sys/acpi/acpi.h>
37 #include <sys/acpica.h>
37 #include <util/sscanf.h>
```

```
39 #include <acpica/include/acpi.h>
40 #include <sys/acpica.h>
```

```
42 static char keyboard_alias[] = "keyboard";
43 static char mouse_alias[] = "mouse";
44 #define ACPI_ENUM_DEBUG "acpi_enum_debug"
45 #define PARSE_RESOURCES_DEBUG 0x0001
46 #define MASTER_LOOKUP_DEBUG 0x0002
47 #define DEVICES_NOT_ENUMED 0x0004
48 #define PARSE_RES_IRQ 0x0008
49 #define PARSE_RES_DMA 0x0010
50 #define PARSE_RES_MEMORY 0x0020
51 #define PARSE_RES_IO 0x0040
52 #define PARSE_RES_ADDRESS 0x0080
53 #define ISA_DEVICE_ENUM 0x1000
54 #define PROCESS_CIDS 0x2000
55 static unsigned long acpi_enum_debug = 0x00;
```

```
57 static char USED_RESOURCES[] = "used-resources";
```

new/usr/src/uts/intel/io/acpica/acpi\_enum.c

2

```
58 static dev_info_t *usedrdip = NULL;
59 static unsigned short used_interrupts = 0;
60 static unsigned short used_dmas = 0;
61 typedef struct used_io_mem {
62     unsigned int start_addr;
63     unsigned int length;
64     struct used_io_mem *next;
65 } used_io_mem_t;
```

unchanged\_portion\_omitted

new/usr/src/uts/intel/io/acpica/acpica.c

1

\*\*\*\*\*

22335 Fri Jan 17 07:31:45 2014

new/usr/src/uts/intel/io/acpica/acpica.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 */

22 /*
23  * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
24  * Copyright (c) 2011, Joyent, Inc. All rights reserved.
25  * Copyright 2013 Nexenta Systems, Inc. All rights reserved.
26  * Copyright 2011 Nexenta Systems, Inc. All rights reserved.
27 */

28 /*
29  * Copyright (c) 2009, Intel Corporation.
30  * All rights reserved.
31 */

33 /*
34  * x86 ACPI CA services
35  */

37 #include <sys/file.h>
38 #include <sys/errno.h>
39 #include <sys/conf.h>
40 #include <sys/modctl.h>
41 #include <sys/open.h>
42 #include <sys/stat.h>
43 #include <sys/spl.h>
44 #include <sys/ddi.h>
45 #include <sys/sunddi.h>
46 #include <sys/esunddi.h>
47 #include <sys/kstat.h>
48 #include <sys/x86_archext.h>

50 #include <acpica/include/acpi.h>
51 #include <sys/acpi/acpi.h>
52 #include <sys/acpica.h>
53 #include <sys/archsystem.h>

54 /*
55  *
56  */
57 static struct modlmisc modlmisc = {
58     &mod_miscops,
```

new/usr/src/uts/intel/io/acpica/acpica.c

2

```
59     "ACPI interpreter",
60 };
61
62 _____unchanged_portion_omitted_____

327 /*
328  * Process acpi-user-options property if present
329  */
330 static void
331 acpica_process_user_options()
332 {
333     static int processed = 0;
334     int acpi_user_options;
335     char *acpi_prop;

337     /*
338      * return if acpi-user-options has already been processed
339      */
340     if (processed)
341         return;
342     else
343         processed = 1;

345     /* converts acpi-user-options from type string to int, if any */
346     if (ddi_prop_lookup_string(DDI_DEV_T_ANY, ddi_root_node(),
347         DDI_PROP_DONTPASS, "acpi-user-options", &acpi_prop) ==
348         DDI_PROP_SUCCESS) {
349         long data;
350         int ret;
351         ret = ddi_strtol(acpi_prop, NULL, 0, &data);
352         if (ret == 0) {
353             e_ddi_prop_remove(DDI_DEV_T_NONE, ddi_root_node(),
354                 "acpi-user-options");
355             e_ddi_prop_update_int(DDI_DEV_T_NONE, ddi_root_node(),
356                 "acpi-user-options", data);
357         }
358         ddi_prop_free(acpi_prop);
359     }

361     /*
362      * fetch the optional options property
363      */
364     acpi_user_options = ddi_prop_get_int(DDI_DEV_T_ANY, ddi_root_node(),
365         DDI_PROP_DONTPASS, "acpi-user-options", 0);

367     /*
368      * Note that 'off' has precedence over 'on'
369      * Also note - all cases of ACPI_OUSER_MASK
370      * provided here, no default: case is present
371      */
372     switch (acpi_user_options & ACPI_OUSER_MASK) {
373     case ACPI_OUSER_DFLT:
374         acpica_enable = acpica_check_bios_date(1999, 1, 1);
375         break;
376     case ACPI_OUSER_ON:
377         acpica_enable = TRUE;
378         break;
379     case ACPI_OUSER_OFF:
380     case ACPI_OUSER_OFF | ACPI_OUSER_ON:
381         acpica_enable = FALSE;
382         break;
383     }

385     acpi_init_level = ACPI_FULL_INITIALIZATION;

387     /*
```

```
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;
395
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);
403
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;
410         if (acpica_muzzle_debug_output)
411             AcpiDbgLevel = 0;
412     }
413
414     _____unchanged_portion_omitted_____
```

new/usr/src/uts/intel/io/acpica/acpica\_ec.c

1

```
*****
16042 Fri Jan 17 07:31:45 2014
new/usr/src/uts/intel/io/acpica/acpica_ec.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 */

22 /*
23  * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  * Copyright 2011 Joyent, Inc. All rights reserved.
26  * Copyright 2013 Nexenta Systems, Inc. All rights reserved.
27  * Copyright 2011 Nexenta Systems, Inc. All rights reserved.
28 */

29 /*
30  * x86 ACPI CA Embedded Controller operation region handler
31  * Solaris x86 ACPI CA Embedded Controller operation region handler
32 */

33 #include <sys/file.h>
34 #include <sys/errno.h>
35 #include <sys/conf.h>
36 #include <sys/modctl.h>
37 #include <sys/open.h>
38 #include <sys/stat.h>
39 #include <sys/ddi.h>
40 #include <sys/sunddi.h>
41 #include <sys/note.h>
42 #include <sys/atomic.h>

44 #include <acpica/include/acpi.h>
42 #include <sys/acpi/acpi.h>
45 #include <sys/acpica.h>

47 /*
48  * EC status bits
49  * Low to high
50  *   Output buffer full?
51  *   Input buffer full?
52  *   <reserved>
53  *   Data register is command byte?
54  *   Burst mode enabled?
55  *   SCI event?
56  *   SMI event?
57  *   <reserved>
58 */
```

new/usr/src/uts/intel/io/acpica/acpica\_ec.c

2

```
59 #define EC_OBF (0x01)
60 #define EC_IBF (0x02)
61 #define EC_DRC (0x08)
62 #define EC_BME (0x10)
63 #define EC_SCI (0x20)
64 #define EC_SMI (0x40)

66 /*
67  * EC commands
68 */
69 #define EC_RD (0x80)
70 #define EC_WR (0x81)
71 #define EC_BE (0x82)
72 #define EC_BD (0x83)
73 #define EC_QR (0x84)

75 #define IO_PORT_DES (0x47)

77 /*
78  * EC softstate
79 */
80 static struct ec_softstate {
81     uint8_t ec_ok; /* != 0 if we have ec_base, ec_sc */
82     uint16_t ec_base; /* base of EC I/O port - data */
83     uint16_t ec_sc; /* EC status/command */
84     ACPI_HANDLE ec_dev_hdl; /* EC device handle */
85     ACPI_HANDLE ec_gpe_hdl; /* GPE info */
86     ACPI_INTEGER ec_gpe_bit;
87     kmutex_t ec_mutex; /* serialize access to EC */
88 } ec;

    unchanged_portion_omitted
```

new/usr/src/uts/intel/io/acpica/master\_ops.c

1

\*\*\*\*\*

8653 Fri Jan 17 07:31:47 2014

new/usr/src/uts/intel/io/acpica/master\_ops.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 */

22 /*
23  * Copyright (c) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
24  * Copyright 2013 Nexenta Systems, Inc. All rights reserved.
25  */

27 #include <sys/kobj.h>
28 #include <sys/kobj_lex.h>
29 #include <sys/ddi.h>
30 #include <sys/sunddi.h>
31 #include <sys/sunndi.h>

33 #include <acpica/include/acpi.h>
30 #include <sys/acpi/acpi.h>
34 #include <sys/acpica.h>

36 #define masterfile "/boot/solaris/devicedb/master"

38 /*
39  * Internal definitions
40  */

42 typedef enum {
43     MF_UNEXPECTED = -1,
44     MF_IDENT,
45     MF_STRING,
46     MF_EOF,
47     MF_NEWLINE,
48     MF_EQUALS,
49     MF_BIT_OR
50 } mftoken_t;
_____unchanged_portion_omitted_
```

new/usr/src/uts/intel/io/acpica/osl.c

1

```
*****
51515 Fri Jan 17 07:31:48 2014
new/usr/src/uts/intel/io/acpica/osl.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 2009 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  * Copyright 2012 Joyent, Inc. All rights reserved.
26  * Copyright 2013 Nexenta Systems, Inc. All rights reserved.
27  * Copyright 2013 PALO, Richard. All rights reserved.
28  * Copyright 2011 Joyent, Inc. All rights reserved.
29  */

30 /*
31  * Copyright (c) 2009-2010, Intel Corporation.
32  * All rights reserved.
33  */

35 /*
36  * x86 ACPI CA OSL
37  * ACPI CA OSL for Solaris x86
38  */

39 #include <sys/types.h>
40 #include <sys/kmem.h>
41 #include <sys/psm.h>
42 #include <sys/pci_cfgspace.h>
43 #include <sys/apic.h>
44 #include <sys/ddi.h>
45 #include <sys/sunddi.h>
46 #include <sys/sunndi.h>
47 #include <sys/pci.h>
48 #include <sys/kobj.h>
49 #include <sys/taskq.h>
50 #include <sys/strlog.h>
51 #include <sys/x86_archext.h>
52 #include <sys/note.h>
53 #include <sys/promif.h>

55 #include <acpica/include/accommon.h>
51 #include <sys/acpi/accommon.h>
56 #include <sys/acpica.h>
```

new/usr/src/uts/intel/io/acpica/osl.c

2

```
58 #define MAX_DAT_FILE_SIZE (64*1024)

60 /* local functions */
61 static int CompressEisaID(char *np);

63 static void scan_d2a_subtree(dev_info_t *dip, ACPI_HANDLE acpiobj, int bus);
64 static int acpica_query_bbn_problem(void);
65 static int acpica_find_pcibus(int busno, ACPI_HANDLE *rh);
66 static int acpica_eval_hid(ACPI_HANDLE dev, char *method, int *rint);
67 static ACPI_STATUS acpica_set_devinf(ACPI_HANDLE, dev_info_t *);
68 static ACPI_STATUS acpica_unset_devinf(ACPI_HANDLE);
69 static void acpica_devinf_handler(ACPI_HANDLE, void *);

71 /*
72  * Event queue vars
73  */
74 int acpica_eventq_init = 0;
75 ddi_taskq_t *osl_eventq[OSL_EC_BURST_HANDLER+1];

77 /*
78  * Priorities relative to minclsyspri that each taskq
79  * run at; OSL_NOTIFY_HANDLER needs to run at a higher
80  * priority than OSL_GPE_HANDLER. There's an implicit
81  * assumption that no priority here results in exceeding
82  * maxclsyspri.
83  * Note: these initializations need to match the order of
84  * ACPI_EXECUTE_TYPE.
85  */
86 int osl_eventq_pri_delta[OSL_EC_BURST_HANDLER+1] = {
87     0, /* OSL_GLOBAL_LOCK_HANDLER */
88     2, /* OSL_NOTIFY_HANDLER */
89     0, /* OSL_GPE_HANDLER */
90     0, /* OSL_DEBUGGER_THREAD */
91     0, /* OSL_EC_POLL_HANDLER */
92     0, /* OSL_EC_BURST_HANDLER */
93 };
    unchanged_portion_omitted

322 ACPI_STATUS
323 AcpiOsPhysicalTableOverride(ACPI_TABLE_HEADER *ExistingTable,
324                             ACPI_PHYSICAL_ADDRESS *NewAddress, UINT32 *NewTableLength)
325 {
326     return (AE_SUPPORT);
327 }

329 /*
330  * ACPI semaphore implementation
331  */
332 typedef struct {
333     kmutex_t      mutex;
334     kcondvar_t    cv;
335     uint32_t      available;
336     uint32_t      initial;
337     uint32_t      maximum;
338 } acpi_sema_t;
    unchanged_portion_omitted

756 void
757 AcpiOsWaitEventsComplete (void)
758 {
759     if (acpica_eventq_init) {
760         int i;
761         /*
762          * blocks until all events initiated by AcpiOsExecute have compl
763          */
764         for (i = OSL_GLOBAL_LOCK_HANDLER; i <= OSL_EC_BURST_HANDLER; i++)
```

```

765         if (osl_eventq[i])
766             ddi_taskq_wait(osl_eventq[i]);
767     }
768 }
769 return;
770 }

772 void
773 AcpiOsSleep(ACPI_INTEGER Milliseconds)
774 {
775     /*
776      * During kernel startup, before the first tick interrupt
777      * has taken place, we can't call delay; very late in
778      * kernel shutdown or suspend/resume, clock interrupts
779      * are blocked, so delay doesn't work then either.
780      * So we busy wait if lbolt == 0 (kernel startup)
781      * or if acpica_use_safe_delay has been set to a
782      * non-zero value.
783      */
784     if ((ddi_get_lbolt() == 0) || acpica_use_safe_delay)
785         drv_usecwait(Milliseconds * 1000);
786     else
787         delay(drv_usectohz(Milliseconds * 1000));
788 }

```

unchanged\_portion\_omitted

```

902 /*
903 *
904 */

```

```

906 #define OSL_RW(ptr, val, type, rw) \
907     { if (rw) *((type *) (ptr)) = *((type *) val); \
908       else *((type *) val) = *((type *) (ptr)); }

```

```

911 static void
912 osl_rw_memory(ACPI_PHYSICAL_ADDRESS Address, UINT64 *Value,
886 osl_rw_memory(ACPI_PHYSICAL_ADDRESS Address, UINT32 *Value,
913               UINT32 Width, int write)
914 {
915     size_t maplen = Width / 8;
916     caddr_t ptr;

918     ptr = psm_map_new((paddr_t)Address, maplen,
919                       PSM_PROT_WRITE | PSM_PROT_READ);

921     switch (maplen) {
922     case 1:
923         OSL_RW(ptr, Value, uint8_t, write);
924         break;
925     case 2:
926         OSL_RW(ptr, Value, uint16_t, write);
927         break;
928     case 4:
929         OSL_RW(ptr, Value, uint32_t, write);
930         break;
931     case 8:
932         OSL_RW(ptr, Value, uint64_t, write);
933         break;
934     default:
935         cmn_err(CE_WARN, "!osl_rw_memory: invalid size %d",
936                Width);
937         break;
938     }

```

```

940     psm_unmap(ptr, maplen);
941 }

943 ACPI_STATUS
944 AcpiOsReadMemory(ACPI_PHYSICAL_ADDRESS Address,
945                  UINT64 *Value, UINT32 Width)
916                  UINT32 *Value, UINT32 Width)
946 {
947     osl_rw_memory(Address, Value, Width, 0);
948     return (AE_OK);
949 }

951 ACPI_STATUS
952 AcpiOsWriteMemory(ACPI_PHYSICAL_ADDRESS Address,
953                   UINT64 Value, UINT32 Width)
924                   UINT32 Value, UINT32 Width)
954 {
955     osl_rw_memory(Address, &Value, Width, 1);
956     return (AE_OK);
957 }

```

unchanged\_portion\_omitted



new/usr/src/uts/intel/io/pci/pci\_boot.c

1

```
*****
96795 Fri Jan 17 07:31:50 2014
new/usr/src/uts/intel/io/pci/pci_boot.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) 2005, 2010, Oracle and/or its affiliates. All rights reserved.
23 */

25 #include <sys/types.h>
26 #include <sys/stat.h>
27 #include <sys/sysmacros.h>
28 #include <sys/sunndi.h>
29 #include <sys/pci.h>
30 #include <sys/pci_impl.h>
31 #include <sys/pcie_impl.h>
32 #include <sys/memlist.h>
33 #include <sys/bootconf.h>
34 #include <io/pci/mps_table.h>
35 #include <sys/pci_cfgacc.h>
36 #include <sys/pci_cfgspace.h>
37 #include <sys/pci_cfgspace_impl.h>
38 #include <sys/psw.h>
39 #include "../common/pci/pci_strings.h"
40 #include <sys/apic.h>
41 #include <io/pciex/pcie_nvidia.h>
42 #include <sys/hotplug/pci/pciehpc_acpi.h>
43 #include <acpica/include/acpi.h>
44 #include <sys/acpi/acpi.h>
45 #include <sys/acpica.h>
46 #include <sys/iommu/lib.h>
47 #include <sys/devcache.h>
48 #include <sys/pci_cfgacc_x86.h>

49 #define pci_getb      (*pci_getb_func)
50 #define pci_getw      (*pci_getw_func)
51 #define pci_getl      (*pci_getl_func)
52 #define pci_putb      (*pci_putb_func)
53 #define pci_putw      (*pci_putw_func)
54 #define pci_putl      (*pci_putl_func)
55 #define dcmn_err      if (pci_boot_debug) cmn_err

57 #define CONFIG_INFO      0
58 #define CONFIG_UPDATE    1
59 #define CONFIG_NEW       2
60 #define CONFIG_FIX       3
```

new/usr/src/uts/intel/io/pci/pci\_boot.c

2

```
61 #define COMPAT_BUFSIZE  512

63 #define PPB_IO_ALIGNMENT      0x1000      /* 4K aligned */
64 #define PPB_MEM_ALIGNMENT    0x100000    /* 1M aligned */
65 /* round down to nearest power of two */
66 #define P2LE(align)
67 {
68     int i = 0;
69     while (align >= 1)
70         i++;
71     align = 1 << i;
72 }

74 /* for is_vga and list_is_vga_only */

76 enum io_mem {
77     IO,
78     MEM
79 };
unchanged_portion_omitted
```

new/usr/src/uts/intel/io/pci/pci\_resource.c

1

```
*****
16007 Fri Jan 17 07:31:51 2014
new/usr/src/uts/intel/io/pci/pci_resource.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 2010 Sun Microsystems, Inc. All rights reserved.
23 * Use is subject to license terms.
24 *
25 * pci_resource.c -- routines to retrieve available bus resources from
26 * the MP Spec. Table and Hotplug Resource Table
27 */

29 #include <sys/types.h>
30 #include <sys/memlist.h>
31 #include <sys/pci_impl.h>
32 #include <sys/system.h>
33 #include <sys/cmn_err.h>
34 #include <acpica/include/acpi.h>
34 #include <sys/acpi/acpi.h>
35 #include <sys/acpica.h>
36 #include "mps_table.h"
37 #include "pcihrt.h"

39 extern int pci_boot_debug;
40 extern int pci_bios_maxbus;
41 #define dprintf if (pci_boot_debug) printf

43 static int tbl_init = 0;
44 static uchar_t *mps_extp = NULL;
45 static uchar_t *mps_ext_endp = NULL;
46 static struct php_entry *hrt_hpep;
47 static int hrt_entry_cnt = 0;
48 static int acpi_cb_cnt = 0;

50 static void mps_probe(void);
51 static void acpi_pci_probe(void);
52 static int mps_find_bus_res(int, int, struct memlist **);
53 static void hrt_probe(void);
54 static int hrt_find_bus_res(int, int, struct memlist **);
55 static int acpi_find_bus_res(int, int, struct memlist **);
56 static uchar_t *find_sig(uchar_t *cp, int len, char *sig);
57 static int checksum(unsigned char *cp, int len);
58 static ACPI_STATUS acpi_wr_cb(ACPI_RESOURCE *rp, void *context);
59 void bus_res_fini(void);
60 static void acpi_trim_bus_ranges(void);
```

new/usr/src/uts/intel/io/pci/pci\_resource.c

2

```
62 struct memlist *acpi_io_res[256];
63 struct memlist *acpi_mem_res[256];
64 struct memlist *acpi_pmem_res[256];
65 struct memlist *acpi_bus_res[256];

67 /*
68 * -1 = attempt ACPI resource discovery
69 * 0 = don't attempt ACPI resource discovery
70 * 1 = ACPI resource discovery successful
71 */
72 volatile int acpi_resource_discovery = -1;

74 struct memlist *
75 find_bus_res(int bus, int type)
76 {
77     struct memlist *res = NULL;

79     if (tbl_init == 0) {
80         tbl_init = 1;
81         acpi_pci_probe();
82         hrt_probe();
83         mps_probe();
84     }

86     if (acpi_find_bus_res(bus, type, &res) > 0)
87         return (res);

89     if (hrt_find_bus_res(bus, type, &res) > 0)
90         return (res);

92     (void) mps_find_bus_res(bus, type, &res);
93     return (res);
94 }
_____unchanged_portion_omitted_____
```

new/usr/src/uts/intel/io/pciex/pcie\_acpi.c

1

```
*****
7920 Fri Jan 17 07:31:52 2014
new/usr/src/uts/intel/io/pciex/pcie_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 2009 Sun Microsystems, Inc. All rights reserved.
23 * Use is subject to license terms.
24 */
25 #include <sys/sysmacros.h>
26 #include <sys/types.h>
27 #include <sys/kmem.h>
28 #include <sys/ddi.h>
29 #include <sys/sunddi.h>
30 #include <sys/sunndi.h>
31 #include <sys/promif.h>
32 #include <sys/pcie.h>
33 #include <sys/pci_cap.h>
34 #include <sys/pcie_impl.h>
35 #include <sys/pcie_acpi.h>
36 #include <acpica/include/acpi.h>
37 #include <sys/acpi/acpi.h>
38 #include <sys/acpica.h>
39
40 ACPI_STATUS pcie_acpi_eval_osc(dev_info_t *dip, ACPI_HANDLE osc_hdl,
41                               uint32_t *osc_flags);
42 static ACPI_STATUS pcie_acpi_find_osc(ACPI_HANDLE busobj,
43                                       ACPI_HANDLE *osc_hdlp);
44
45 #ifdef DEBUG
46 static void pcie_dump_acpi_obj(ACPI_HANDLE pcibus_obj);
47 static ACPI_STATUS pcie_walk_obj_namespace(ACPI_HANDLE hdl, uint32_t nl,
48 void *context, void **ret);
49 static ACPI_STATUS pcie_print_acpi_name(ACPI_HANDLE hdl, uint32_t nl,
50 void *context, void **ret);
51 #endif /* DEBUG */
52
53 int
54 pcie_acpi_osc(dev_info_t *dip, uint32_t *osc_flags)
55 {
56     ACPI_HANDLE pcibus_obj;
57     int status = AE_ERROR;
58     ACPI_HANDLE osc_hdl;
59     pcie_bus_t *bus_p = PCIE_DIP2BUS(dip);
60     pcie_x86_priv_t *osc_p = (pcie_x86_priv_t *)bus_p->bus_plat_private;
```

new/usr/src/uts/intel/io/pciex/pcie\_acpi.c

2

```
61     /* Mark this so we know _OSC has been called for this device */
62     osc_p->bus_osc = B_TRUE;
63
64     /*
65      * (1) Find the ACPI device node for this bus node.
66      */
67     status = acpica_get_handle(dip, &pcibus_obj);
68     if (status != AE_OK) {
69         PCIE_DBG("No ACPI device found (dip %p)\n", (void *)dip);
70         return (DDI_FAILURE);
71     }
72
73     /*
74      * (2) Check if _OSC method is present.
75      */
76     if (pcie_acpi_find_osc(pcibus_obj, &osc_hdl) != AE_OK) {
77         /* no _OSC method present */
78         PCIE_DBG("no _OSC method present for dip %p\n",
79                 (void *)dip);
80         return (DDI_FAILURE);
81     }
82
83     /*
84      * (3) _OSC method exists; evaluate _OSC.
85      */
86     if (pcie_acpi_eval_osc(dip, osc_hdl, osc_flags) != AE_OK) {
87         PCIE_DBG("Failed to evaluate _OSC method for dip 0x%p\n",
88                 (void *)dip);
89         return (DDI_FAILURE);
90     }
91
92     osc_p->bus_osc_hp = (*osc_flags & OSC_CONTROL_PCIE_NAT_HP) ?
93         B_TRUE : B_FALSE;
94     osc_p->bus_osc_aer = (*osc_flags & OSC_CONTROL_PCIE_ADV_ERR) ?
95         B_TRUE : B_FALSE;
96
97 #ifdef DEBUG
98     if (pcie_debug_flags > 1)
99         pcie_dump_acpi_obj(pcibus_obj);
100 #endif /* DEBUG */
101
102     return (DDI_SUCCESS);
103 }
104
105 _____
106 unchanged portion omitted
```

new/usr/src/uts/intel/pci\_autoconfig/Makefile

1

```
*****
2322 Fri Jan 17 07:31:52 2014
new/usr/src/uts/intel/pci_autoconfig/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 # uts/intel/pci_autoconfig/Makefile
23 # Copyright 2009 Sun Microsystems, Inc. All rights reserved.
24 # Use is subject to license terms.
25 #
26 # This makefile drives the production of the PCI autoconfiguration
27 # kernel module.
28 #
29 # intel platform dependent
30 #

32 #
33 # Path to the base of the uts directory tree (usually /usr/src/uts).
34 #
35 UTSBASE = ../../

37 #
38 # Define the module and object file sets.
39 #
40 MODULE      = pci_autoconfig
41 OBJECTS      = $(PCI_AUTOCONFIG_OBJS:%=$(OBJS_DIR)/%)
42 LINTS        = $(PCI_AUTOCONFIG_OBJS:%.o=$(LINTS_DIR)/%.ln)
43 ROOTMODULE   = $(ROOT_MISC_DIR)/$(MODULE)
44 INC_PATH     += -I$(UTSBASE)/i86pc
45 INC_PATH     += -I$(SRC)/common

47 #
48 # Include common rules.
49 #
50 include $(UTSBASE)/intel/Makefile.intel

52 #
53 # Define targets
54 #
55 ALL_TARGET   = $(BINARY)
56 LINT_TARGET   = $(MODULE).lint
57 INSTALL_TARGET = $(BINARY) $(ROOTMODULE)

59 #
60 # Depends on acpica ACPI CA interpreter and PCI-E framework
61 #
```

new/usr/src/uts/intel/pci\_autoconfig/Makefile

2

```
62 LDFLAGS      += -dy -Nmisc/acpica -Nmisc/pcie

64 #
65 # For now, disable these lint checks; maintainers should endeavor
66 # to investigate and remove these for maximum lint coverage.
67 # Please do not carry these forward to new Makefiles.
68 #
69 LINTTAGS      += -erroff=E_BAD_PTR_CAST_ALIGN
70 LINTTAGS      += -erroff=E_ASSIGN_NARROW_CONV

72 CERRWARN      += -_gcc=-Wno-parentheses

74 #
75 # Default build targets.
76 #
77 .KEEP_STATE:

79 def:          $(DEF_DEPS)

81 all:          $(ALL_DEPS)

83 clean:        $(CLEAN_DEPS)

85 clobber:      $(CLOBBER_DEPS)

87 lint:         $(LINT_DEPS)

89 modlintlib:   $(MODLINTLIB_DEPS)

91 clean.lint:   $(CLEAN_LINT_DEPS)

93 install:      $(INSTALL_DEPS)

95 #
96 # Include common targets.
97 #
98 include $(UTSBASE)/intel/Makefile.targ
```

new/usr/src/uts/intel/power/Makefile

1

\*\*\*\*\*

2365 Fri Jan 17 07:31:52 2014

new/usr/src/uts/intel/power/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 # uts/intel/power/Makefile
23 # Copyright 2007 Sun Microsystems, Inc. All rights reserved.
24 # Use is subject to license terms.
25 #
26 #
27 #
28 # This makefile drives the production of the power driver
29 #
30 # intel architecture dependent
31 #
32 #
33 #
34 # Path to the base of the uts directory tree (usually /usr/src/uts).
35 #
36 UTBASE = ../../
37 #
38 #
39 # Define the module and object file sets.
40 #
41 MODULE      = power
42 OBJECTS      = $(POWER_OBJS:%=$(OBJS_DIR)/%)
43 LINTS        = $(POWER_OBJS:%.o=$(LINTS_DIR)/%.ln)
44 ROOTMODULE   = $(ROOT_DRV_DIR)/$(MODULE)
45 CONF_SRCDIR  = $(UTBASE)/common/io
46 INC_PATH     += -I$(SRC)/common
47 #
48 #
49 # Include common rules.
50 #
51 include $(UTBASE)/intel/Makefile.intel
52 #
53 #
54 # Define targets
55 #
56 ALL_TARGET   = $(BINARY) $(CONFMOD)
57 LINT_TARGET   = $(MODULE).lint
58 INSTALL_TARGET = $(BINARY) $(ROOTMODULE) $(ROOT_CONFFILE)
59 #
60 #
61 # lint pass one enforcement
```

new/usr/src/uts/intel/power/Makefile

2

```
62 #
63 CFLAGS += $(CCVERBOSE)
64 #
65 #
66 # ACPI power button
67 #
68 CFLAGS += -DACPI_POWER_BUTTON
69 LINTFLAGS += -DACPI_POWER_BUTTON
70 #
71 LDFLAGS += -dy -N misc/acpica
72 #
73 #
74 # For now, disable these lint checks; maintainers should endeavor
75 # to investigate and remove these for maximum lint coverage.
76 # Please do not carry these forward to new Makefiles.
77 #
78 LINTTAGS      += -erroff=E_BAD_PTR_CAST_ALIGN
79 LINTTAGS      += -erroff=E_STATIC_UNUSED
80 #
81 CERRWARN      += -_gcc=-Wno-unused-variable
82 #
83 #
84 # Default build targets.
85 #
86 .KEEP_STATE:
87 #
88 def:           $(DEF_DEPS)
89 #
90 all:           $(ALL_DEPS)
91 #
92 clean:         $(CLEAN_DEPS)
93 #
94 clobber:       $(CLOBBER_DEPS)
95 #
96 lint:         $(LINT_DEPS)
97 #
98 modlintlib:    $(MODLINTLIB_DEPS)
99 #
100 clean.lint:    $(CLEAN_LINT_DEPS)
101 #
102 install:       $(INSTALL_DEPS)
103 #
104 #
105 # Include common targets.
106 #
107 include $(UTBASE)/intel/Makefile.targ
```

new/usr/src/uts/intel/sys/hotplug/pci/pciehpc\_acpi.h

1

\*\*\*\*\*

2492 Fri Jan 17 07:31:54 2014

new/usr/src/uts/intel/sys/hotplug/pci/pciehpc\_acpi.h

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 2009 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  */

27 #ifndef _PCIEHPC_ACPI_H
28 #define _PCIEHPC_ACPI_H

30 #ifdef __cplusplus
31 extern "C" {
32 #endif

34 #include <acpica/include/acpi.h>
34 #include <sys/acpi/acpi.h>
35 #include <sys/acpica.h>
36 #include <sys/hotplug/pci/pcie_hp.h>
37 #include <sys/hotplug/pci/pciehpc.h>

39 /* soft state data structure for ACPI hot plug mode */
40 typedef struct pciehpc_acpi {
41     /* handle for the ACPI device for the bus node with HPC */
42     ACPI_HANDLE    bus_obj;

44     /* handle for the ACPI device for the slot (dev#0,func#0) */
45     ACPI_HANDLE    slot_dev_obj;

47     /* ACPI control methods present on the bus node */
48     uint16_t        bus_methods;

50     /* ACPI control methods on the slot device functions */
51     uint16_t        slot_methods;
52 } pciehpc_acpi_t;
    unchanged_portion_omitted_
```