```
**********************************************************
    1567 Wed Aug 28 04:02:56 2019
new/usr/src/psm/promif/ieee1275/sun4u/Makefile.files
11630 remove checks for 64-bit capable hardware
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# Copyright 2019 Peter Tribble.
#
# psm/promif/ieee1275/sun4u/Makefile.files
#
#       This Makefile defines all the promif file modules for the
#       directory psm/promif/ieee1275/sun4u.
#


#
# Note that the kernel doesn't use/need prom_map.o
#

#
#                       PROM Platform-dependent routines
#
CORE_OBJS +=                            \
        prom_alloc.o                    \
        prom_cpuctl.o                   \
        prom_efcode.o                   \
        prom_fio.o                      \
        prom_getunum.o                  \
        prom_heartbeat.o                \
        prom_idprom.o                   \
        prom_init.o                     \
        prom_macaddr.o                  \
        prom_mem.o                      \
        prom_mmu.o                      \
        prom_power_off.o                \
        prom_retain.o                   \
        prom_serengeti.o                \
        prom_set_traptable.o            \
        prom_sparc.o                    \
        prom_sunfire.o                  \
        prom_tlb.o                      \
        prom_vercheck.o                 \
        prom_vername.o                  \
        prom_opl.o
```

```
**********************************************************
    1564 Wed Aug 28 04:02:56 2019
new/usr/src/psm/promif/ieee1275/sun4v/Makefile.files
11630 remove checks for 64-bit capable hardware
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# Copyright 2019 Peter Tribble.
#
#pragma ident   "%Z%%M% %I%     %E% SMI"
#
# psm/promif/ieee1275/sun4u/Makefile.files
#
#       This Makefile defines all the promif file modules for the
#       directory psm/promif/ieee1275/sun4u.
#


#
# Note that the kernel doesn't use/need prom_map.o
#


#
#                       PROM Platform-dependent routines
#
CORE_OBJS +=                            \
        prom_alloc.o                    \
        prom_cpuctl.o                   \
        prom_efcode.o                   \
        prom_fio.o                      \
        prom_getunum.o                  \
        prom_idprom.o                   \
        prom_heartbeat.o                \
        prom_init.o                     \
        prom_macaddr.o                  \
        prom_mem.o                      \
        prom_mmu.o                      \
        prom_power_off.o                \
        prom_retain.o                   \
        prom_set_mmfsa_traptable.o      \
        prom_sparc.o                    \
        prom_sun4v_api_version.o        \
        prom_sun4v_soft_state_supported.o       \
        prom_vercheck.o                 \
        prom_vername.o
```

```
**********************************************************
   14892 Wed Aug 28 04:02:56 2019
new/usr/src/psm/stand/boot/sparc/common/boot_plat.c
11630 remove checks for 64-bit capable hardware
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2019 Peter Tribbble.
 */

#include <sys/param.h>
#include <sys/fcntl.h>
#include <sys/obpdefs.h>
#include <sys/reboot.h>
#include <sys/promif.h>
#include <sys/stat.h>
#include <sys/bootvfs.h>
#include <sys/platnames.h>
#include <sys/salib.h>
#include <sys/elf.h>
#include <sys/link.h>
#include <sys/auxv.h>
#include <sys/boot_policy.h>
#include <sys/boot_redirect.h>
#include <sys/bootconf.h>
#include <sys/boot.h>
#include "boot_plat.h"

#define SUCCESS         0
#define FAILURE         -1

#define ISSPACE(c)              (c == ' ' || c == '\t')
#define SKIP_WHITESPC(cp)       while (*cp && ISSPACE(*cp)) cp++;


#ifdef DEBUG
int debug = 0;
#else
static const int debug = 0;
#endif

#define dprintf         if (debug) printf

#ifdef DEBUG_LISTS
void print_memlist(struct memlist *av);
```



```
#endif

extern  int (*readfile(int fd, int print))();
extern  void kmem_init(void);
extern  void *kmem_alloc(size_t, int);
extern  void kmem_free(void *, size_t);
extern  void get_boot_args(char *buf);
extern  void setup_bootops(void);
extern  struct  bootops bootops;
extern  void exitto(int (*entrypoint)());
extern  void exitto64(int (*entrypoint)(), void *bootvec);

int openfile(char *filename);

char *default_name;
char *default_path;

int vac;                        /* virtual address cache type (none == 0) */
int is_sun4v;                   /* sun4u vs. sun4v */
int client_isLP64 = 1;          /* SPARC clients are always LP64 */

extern bootplat_defaults_t sun4u_plat_defaults;
extern bootplat_defaults_t sun4v_plat_defaults;

/*
 * filename is the name of the standalone we're going to execute.
 */
char    filename[MAXPATHLEN];

char * const defname = "kernel/sparcv9/unix";

/*
 *  We enable the cache by default
 *  but boot -n will leave it alone...
 *  that is, we use whatever state the PROM left it in.
 */
char    *mfg_name;
int     cache_state = 1;
char    filename2[MAXPATHLEN];

int     boothowto = 0;
int     verbosemode = 0;


/*
 * Copy filename and bargs into v2args_buf, which will be exported as the
 * boot-args boot property.  We should probably warn the user if anything gets
 * cut off.
 */
void
set_client_bootargs(const char *filename, const char *bargs)
{
        int i = 0;
        const char *s;

        s = filename;
        while (*s != '\0' && i < V2ARGS_BUF_SZ - 1)
                v2args_buf[i++] = *s++;

        if (i >= V2ARGS_BUF_SZ - 2) {
                /* Not enough room for a space and any of bargs. */
                v2args_buf[i] = '\0';
                return;
        }

        v2args_buf[i++] = ' ';
```

```
129         s = bargs;
130         while (*s != '\0' && i < V2ARGS_BUF_SZ - 1)
131                 v2args_buf[i++] = *s++;
132
133         v2args_buf[i] = '\0';
134 }
```

_____unchanged_portion_omitted_

```
456 #define PROM_VERS_MAX_LEN       64
457 void
458 system_check(void)
459 {
461         char buf[PROM_VERS_MAX_LEN];
460         pnode_t n;
461         char    arch[128];
462         size_t  len;
463         bootplat_defaults_t *plat_defaults;
464
465         /*
466          * This is a sun4v machine iff the device_type property
467          * exists on the root node and has the value "sun4v".
468          * Some older sunfire proms do not have such a property.
469          */
470         is_sun4v = 0;
471         n = prom_rootnode();
472         len = prom_getproplen(n, "device_type");
473         if (len > 0 && len < sizeof (arch)) {
474                 (void) prom_getprop(n, "device_type", arch);
475                 arch[len] = '\0';
476                 dprintf("device_type=%s\n", arch);
477                 if (strcmp(arch, "sun4v") == 0) {
478                         is_sun4v = 1;
479                 }
480         } else {
481                 dprintf("device_type: no such property, len=%d\n", (int)len);
482         }
483
486         if (!is_sun4v && cpu_is_ultrasparc_1()) {
487                 printf("UltraSPARC I processors are not supported by this "
488                     "release of Solaris.\n");
489                 prom_exit_to_mon();
490         }
491
484         /*
485          * Set up defaults per platform
486          */
487         plat_defaults = (is_sun4v) ?
488             &sun4v_plat_defaults : &sun4u_plat_defaults;
489
490         default_name = plat_defaults->plat_defaults_name;
491         default_path = plat_defaults->plat_defaults_path;
492         vac = plat_defaults->plat_defaults_vac;
493
494         dprintf("default_name: %s\n", default_name);
495         dprintf("default_path: %s\n", default_path);
496         dprintf("vac: %d\n", vac);
497
506         if (prom_version_check(buf, PROM_VERS_MAX_LEN, NULL) != PROM_VER64_OK) {
507                 printf("The firmware on this system does not support the 64-bit"
508                     " OS.\n\tPlease upgrade to at least the following version:"
509                     "\n\n\t%s\n", buf);
510                 prom_exit_to_mon();
511         }
497 }
```

_____unchanged_portion_omitted_

```
**********************************************************
    2889 Wed Aug 28 04:02:56 2019
new/usr/src/psm/stand/boot/sparc/common/boot_plat.h
11630 remove checks for 64-bit capable hardware
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2019 Peter Tribble.
 */

#ifndef _BOOT_PLAT_H
#define _BOOT_PLAT_H

#ifdef __cplusplus
extern "C" {
#endif


/*
 * boot platform defaults per architecture
 */
typedef struct bootplat_defaults {
        char    *plat_defaults_name;
        char    *plat_defaults_path;
        int     plat_defaults_vac;
} bootplat_defaults_t;

/* boot_plat.c */
extern int      verbosemode;
extern char     filename[];
extern char     *const defname;
extern char     *const defname64;

extern int      bootprog(char *, char *, boolean_t);
extern char     *choose_default_filename(char *, char *);
extern char     *get_default_filename(void);
extern void     post_mountroot(char *, char *);
extern void     redirect_boot_path(char *, char *);
extern void     set_client_bootargs(const char *, const char *);
extern boolean_t is_netdev(char *devpath);


/* bootops.c */
extern struct bootops   bootops;
```



```
extern void     setup_bootops(void);
extern void     update_memlist(char *, char *, struct memlist **);


/*
 * bootprop.c.  These variables will be exported to the standalone as boot
 * properties.
 */
extern char     *v2path, *kernname, *systype, *my_own_name;
extern char     v2args_buf[];
#define V2ARGS_BUF_SZ   OBP_MAXPATHLEN
extern char     *v2args;
extern char     *mfg_name;
extern char     *impl_arch_name;
extern char     *bootp_response;
extern char     *boot_message;
extern int      cache_state;
extern uint64_t memlistextent;
extern char     *netdev_path;

extern void     set_default_filename(char *filename);


/* get.c */
extern int      cons_gets(char *, int);


/* machdep.c */
extern int  vac;

extern void     fiximp(void);
extern void     retain_nvram_page();
extern int      cpu_is_ultrasparc_1(void);


/* memlist.c */
extern void             init_memlists(void);
extern struct memlist   *fill_memlists(char *name, char *prop,
    struct memlist *old);


/* srt0.c */
extern void     _start(void *romp, ...);
extern void     exitto(int (*entrypoint)());
extern void     exitto64(int (*entrypoint)(), void *bootvec);


/* standalloc.c */
extern caddr_t  memlistpage;
extern caddr_t  scratchmemp;



#ifdef __cplusplus
}
_____unchanged_portion_omitted_
```

```
**********************************************************
    2309 Wed Aug 28 04:02:56 2019
new/usr/src/psm/stand/boot/sparc/common/machdep.c
11630 remove checks for 64-bit capable hardware
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */
/*
 * Copyright 2019 Peter Tribble.
 */

#include <sys/types.h>
#include <sys/param.h>
#include <sys/fcntl.h>
#include <sys/promif.h>
#include <sys/prom_plat.h>
#include <sys/salib.h>

extern int is_sun4v;

/*
 * Check if the CPU should default to 64-bit or not.
 * UltraSPARC-1's default to 32-bit mode.
 * Everything else defaults to 64-bit mode.
 */

/*
 * Manufacturer codes for the CPUs we're interested in
 */
#define TI_JEDEC        0x17
#define SUNW_JEDEC      0x22

/*
 * Implementation codes for the CPUs we're interested in
 */
#define IMPL_US_I       0x10

static pnode_t
visit(pnode_t node)
{
        int impl, manu;
        char name[32];
        static char ultrasparc[] = "SUNW,UltraSPARC";
        static char implementation[] = "implementation#";
```



```
        static char manufacturer[] = "manufacturer#";

        /*
         * if name isn't 'SUNW,UltraSPARC', continue.
         */
        if (prom_getproplen(node, "name") != sizeof (ultrasparc))
                return ((pnode_t)0);
        (void) prom_getprop(node, "name", name);
        if (strncmp(name, ultrasparc, sizeof (ultrasparc)) != 0)
                return ((pnode_t)0);

        if (prom_getproplen(node, manufacturer) != sizeof (int))
                return ((pnode_t)0);
        (void) prom_getprop(node, manufacturer, (caddr_t)&manu);

        if ((manu != SUNW_JEDEC) && (manu != TI_JEDEC))
                return ((pnode_t)0);

        if (prom_getproplen(node, implementation) != sizeof (int))
                return ((pnode_t)0);
        (void) prom_getprop(node, implementation, (caddr_t)&impl);

        if (impl != IMPL_US_I)
                return ((pnode_t)0);

        return (node);
}

/*
 * visit each node in the device tree, until we get a non-null answer
 */
static pnode_t
walk(pnode_t node)
{
        pnode_t id;

        if (visit(node))
                return (node);

        for (node = prom_childnode(node); node; node = prom_nextnode(node))
                if ((id = walk(node)) != (pnode_t)0)
                        return (id);

        return ((pnode_t)0);
}

/*
 * Check if the CPU is an UltraSPARC-1 or not.
 */
int
cpu_is_ultrasparc_1(void)
{
        static int cpu_checked;
        static int cpu_default;

        /*
         * If we already checked or the machine is
         * a sun4v, we already know the answer.
         */
        if (!is_sun4v || cpu_checked == 0) {
                if (walk(prom_rootnode()))
                        cpu_default = 1;
                cpu_checked = 1;
        }

        return (cpu_default);
```

```
128 }

130 /*
 39  * Retain a page or reclaim a previously retained page of physical
 40  * memory for use by the prom upgrade. If successful, leave
 41  * an indication that a page was retained by creating a boolean
 42  * property in the root node.
 43  *
 44  * XXX: SUNW,retain doesn't work as expected on server systems,
 45  * so we don't try to retain any memory on those systems.
 46  *
 47  * XXX: do a '0 to my-self' as a workaround for 4160914
 48  */

 50 int dont_retain_memory;

 52 void
 53 retain_nvram_page(void)
 54 {
 55         unsigned long long phys = 0;
 56         int len;
 57         char name[32];
 58         static char create_prop[] =
 59             "0 to my-self dev / 0 0 \" boot-retained-page\" property";
 60         static char ue[] = "SUNW,Ultra-Enterprise";
 61         extern int verbosemode;

 63         if (dont_retain_memory)
 64                 return;

 66         if (!is_sun4v) {
 67                 len = prom_getproplen(prom_rootnode(), "name");
 68                 if ((len != -1) && (len <= sizeof (name))) {
 69                         (void) prom_getprop(prom_rootnode(), "name", name);
 70                         if (strcmp(name, ue) == 0)
 71                                 return;
 72                 }
 73         }

 75         if (prom_retain("OBPnvram", PAGESIZE, PAGESIZE, &phys) != 0) {
 76                 printf("prom_retain failed\n");
 77                 return;
 78         }
 79         if (verbosemode)
 80                 printf("retained OBPnvram page at 0x%llx\n", phys);

 82         prom_interpret(create_prop, 0, 0, 0, 0, 0);
 83 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    3350 Wed Aug 28 04:02:56 2019
new/usr/src/psm/stand/lib/promif/sparcv9/ieee1275/sun4/Makefile
11630 remove checks for 64-bit capable hardware
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# Copyright 2019 Peter Tribble.
#
# psm/stand/boot/sparcv9/ieee1275/sun4u/Makefile
#
#
TOPDIR =        ../../../../../../..

include $(TOPDIR)/Makefile.master
include $(TOPDIR)/lib/Makefile.lib
include $(TOPDIR)/psm/stand/lib/Makefile.lib
include $(TOPDIR)/psm/Makefile.psm.64

CFLAGS64        += -xchip=ultra $(CCABS32)

PLATSUN4DIR =   $(TOPDIR)/psm/promif/ieee1275/sun4
SYSDIR =        $(TOPDIR)/uts

LIBPLAT =       libplat.a
LINTLIBPLAT =   llib-lplat.ln

PLAT_PFILES =

PLAT_PSUN4FILES =               \
        prom_alloc.c            \
        prom_cpuctl.c           \
        prom_fio.c              \
        prom_getunum.c          \
        prom_heartbeat.c        \
        prom_idprom.c           \
        prom_init.c             \
        prom_macaddr.c          \
        prom_map.c              \
        prom_mem.c              \
        prom_mmu.c              \
        prom_retain.c           \
        prom_sparc.c            \
        prom_vercheck.c         \
        prom_vername.c
```



```
KARCH =         sun4u
MMU =           sfmmu

OBJSDIR =       objs

PLAT_POBJ =     $(PLAT_PFILES:%.c=$(OBJSDIR)/%.o)
PLAT_PSUN4OBJ = $(PLAT_PSUN4FILES:%.c=$(OBJSDIR)/%.o)
OBJS =          $(PLAT_POBJ) $(PLAT_PSUN4OBJ)
L_OBJS =        $(OBJS:%.o=%.ln)
L_SRCS =        $(PLAT_PFILES:%=$(PLATDIR)/%)
L_SRCS +=       $(PLAT_PSUN4FILES:%=$(PLATSUN4DIR)/%)

ARCHOPTS=       -Dsun4u
ASFLAGS =       -P -D__STDC__ -D_BOOT -D_ASM
CPPDEFS =       $(ARCHOPTS) -D$(KARCH) -D_BOOT -D_KERNEL -D_MACHDEP
CPPINCS =       -I. -I$(SYSDIR)/sun4 -I$(SYSDIR)/$(KARCH) -I$(SYSDIR)/$(MMU) \
                -I$(SYSDIR)/sparc/v9 -I$(SYSDIR)/sparc \
                -I$(SYSDIR)/sun -I$(SYSDIR)/common
CPPFLAGS=       $(CPPDEFS) $(CPPINCS) $(CPPFLAGS.master)
CFLAGS +=       $(CCVERBOSE)

.KEEP_STATE:

.PARALLEL:      $(OBJS) $(L_OBJS)

all install: $(LIBPLAT)

lint: $(LINTLIBPLAT)

clean:
        $(RM) $(OBJS) $(L_OBJS)

clobber: clean
        $(RM) $(LIBPLAT) $(LINTLIBPLAT)

$(LIBPLAT): $(OBJSDIR) .WAIT $(OBJS)
        $(BUILD.AR) $(OBJS)

$(LINTLIBPLAT): $(OBJSDIR) .WAIT $(L_OBJS)
        @$(ECHO) "\nlint library construction:" $@
        @$(LINT.lib) -o plat $(L_SRCS)

$(OBJSDIR):
        -@[ -d $@ ] || mkdir $@

#
# build rules using standard library object subdirectory
#

$(OBJSDIR)/%.o: $(PLATDIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

$(OBJSDIR)/%.o: $(PLATDIR)/%.s
        $(COMPILE.s) -o $@ $<
        $(POST_PROCESS_O)

$(OBJSDIR)/%.o: $(PLATSUN4DIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

$(OBJSDIR)/%.o: $(PLATSUN4DIR)/%.s
        $(COMPILE.s) -o $@ $<
        $(POST_PROCESS_O)
```

```
127 $(OBJSDIR)/%.ln: $(PLATDIR)/%.c
128         @($(LHEAD) $(LINT.c) $< $(LTAIL))
129         @$(MV) $(@F) $@

131 $(OBJSDIR)/%.ln: $(PLATDIR)/%.s
132         @($(LHEAD) $(LINT.s) $< $(LTAIL))
133         @$(MV) $(@F) $@

135 $(OBJSDIR)/%.ln: $(PLATSUN4DIR)/%.c
136         @($(LHEAD) $(LINT.c) $< $(LTAIL))
137         @$(MV) $(@F) $@

139 $(OBJSDIR)/%.ln: $(PLATSUN4DIR)/%.s
140         @($(LHEAD) $(LINT.s) $< $(LTAIL))
141         @$(MV) $(@F) $@
```

```
**********************************************************
    3450 Wed Aug 28 04:02:56 2019
new/usr/src/psm/stand/lib/promif/sparcv9/ieee1275/sun4u/Makefile
11630 remove checks for 64-bit capable hardware
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# Copyright 2019 Peter Tribble.
#
# psm/stand/boot/sparcv9/ieee1275/sun4u/Makefile
#
#
TOPDIR =        ../../../../../../..

include $(TOPDIR)/Makefile.master
include $(TOPDIR)/lib/Makefile.lib
include $(TOPDIR)/psm/stand/lib/Makefile.lib
include $(TOPDIR)/psm/Makefile.psm.64

CFLAGS64        += -xchip=ultra $(CCABS32)

PLATDIR =       $(TOPDIR)/psm/promif/ieee1275/sun4u
PLATSUN4DIR =   $(TOPDIR)/psm/promif/ieee1275/sun4
SYSDIR =        $(TOPDIR)/uts

LIBPLAT =       libplat.a
LINTLIBPLAT =   llib-lplat.ln

PLAT_PFILES =                   \
        prom_serengeti.c        \
        prom_sunfire.c          \
        prom_tlb.c

PLAT_PSUN4FILES =               \
        prom_alloc.c            \
        prom_cpuctl.c           \
        prom_fio.c              \
        prom_getunum.c          \
        prom_heartbeat.c        \
        prom_idprom.c           \
        prom_init.c             \
        prom_macaddr.c          \
        prom_map.c              \
        prom_mem.c              \
        prom_mmu.c              \
```



```
        prom_retain.c           \
        prom_sparc.c            \
        prom_vercheck.c         \
        prom_vername.c

KARCH =         sun4u
MMU =           sfmmu

OBJSDIR =       objs

PLAT_POBJ =     $(PLAT_PFILES:%.c=$(OBJSDIR)/%.o)
PLAT_PSUN4OBJ = $(PLAT_PSUN4FILES:%.c=$(OBJSDIR)/%.o)
OBJS =          $(PLAT_POBJ) $(PLAT_PSUN4OBJ)
L_OBJS =        $(OBJS:%.o=%.ln)
L_SRCS =        $(PLAT_PFILES:%=$(PLATDIR)/%)
L_SRCS +=       $(PLAT_PSUN4FILES:%=$(PLATSUN4DIR)/%)

ARCHOPTS=       -Dsun4u
ASFLAGS =       -P -D__STDC__ -D_BOOT -D_ASM
CPPDEFS =       $(ARCHOPTS) -D$(KARCH) -D_BOOT -D_KERNEL -D_MACHDEP
CPPINCS =       -I. -I$(SYSDIR)/sun4 -I$(SYSDIR)/$(KARCH) -I$(SYSDIR)/$(MMU) \
                -I$(SYSDIR)/sparc/v9 -I$(SYSDIR)/sparc \
                -I$(SYSDIR)/sun -I$(SYSDIR)/common
CPPFLAGS=       $(CPPDEFS) $(CPPINCS) $(CPPFLAGS.master)
CFLAGS +=       $(CCVERBOSE)

.KEEP_STATE:

.PARALLEL:      $(OBJS) $(L_OBJS)

all install: $(LIBPLAT)

lint: $(LINTLIBPLAT)

clean:
        $(RM) $(OBJS) $(L_OBJS)

clobber: clean
        $(RM) $(LIBPLAT) $(LINTLIBPLAT)

$(LIBPLAT): $(OBJSDIR) .WAIT $(OBJS)
        $(BUILD.AR) $(OBJS)

$(LINTLIBPLAT): $(OBJSDIR) .WAIT $(L_OBJS)
        @$(ECHO) "\nlint library construction:" $@
        @$(LINT.lib) -o plat $(L_SRCS)

$(OBJSDIR):
        -@[ -d $@ ] || mkdir $@

#
# build rules using standard library object subdirectory
#
$(OBJSDIR)/%.o: $(PLATDIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

$(OBJSDIR)/%.o: $(PLATDIR)/%.s
        $(COMPILE.s) -o $@ $<
        $(POST_PROCESS_O)

$(OBJSDIR)/%.o: $(PLATSUN4DIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

$(OBJSDIR)/%.o: $(PLATSUN4DIR)/%.s
```

```
127         $(COMPILE.s) -o $@ $<
128         $(POST_PROCESS_O)

130 $(OBJSDIR)/%.ln: $(PLATDIR)/%.c
131         @($(LHEAD) $(LINT.c) $< $(LTAIL))
132         @$(MV) $(@F) $@

134 $(OBJSDIR)/%.ln: $(PLATDIR)/%.s
135         @($(LHEAD) $(LINT.s) $< $(LTAIL))
136         @$(MV) $(@F) $@

138 $(OBJSDIR)/%.ln: $(PLATSUN4DIR)/%.c
139         @($(LHEAD) $(LINT.c) $< $(LTAIL))
140         @$(MV) $(@F) $@

142 $(OBJSDIR)/%.ln: $(PLATSUN4DIR)/%.s
143         @($(LHEAD) $(LINT.s) $< $(LTAIL))
144         @$(MV) $(@F) $@
```

```
**********************************************************
    3417 Wed Aug 28 04:02:56 2019
new/usr/src/psm/stand/lib/promif/sparcv9/ieee1275/sun4v/Makefile
11630 remove checks for 64-bit capable hardware
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
# Copyright 2019 Peter tribble.
#
# psm/stand/boot/sparcv9/ieee1275/sun4v/Makefile
#
#
TOPDIR =        ../../../../../../..

include $(TOPDIR)/Makefile.master
include $(TOPDIR)/lib/Makefile.lib
include $(TOPDIR)/psm/stand/lib/Makefile.lib
include $(TOPDIR)/psm/Makefile.psm.64

CFLAGS64        += -xchip=ultra $(CCABS32)

PLATDIR =       $(TOPDIR)/psm/promif/ieee1275/sun4u
PLATSUN4DIR =   $(TOPDIR)/psm/promif/ieee1275/sun4
SYSDIR =        $(TOPDIR)/uts

LIBPLAT =       libplat.a
LINTLIBPLAT =   llib-lplat.ln

PLAT_PFILES =

PLAT_PSUN4FILES =               \
        prom_alloc.c            \
        prom_cpuctl.c           \
        prom_fio.c              \
        prom_getunum.c          \
        prom_heartbeat.c        \
        prom_idprom.c           \
        prom_init.c             \
        prom_macaddr.c          \
        prom_map.c              \
        prom_mem.c              \
        prom_mmu.c              \
        prom_retain.c           \
        prom_sparc.c            \
        prom_vername.c
```



```
        prom_vername.c          \
        prom_vercheck.c

KARCH =         sun4v
MMU =           sfmmu

OBJSDIR =       objs

PLAT_POBJ =     $(PLAT_PFILES:%.c=$(OBJSDIR)/%.o)
PLAT_PSUN4OBJ = $(PLAT_PSUN4FILES:%.c=$(OBJSDIR)/%.o)
OBJS =          $(PLAT_POBJ) $(PLAT_PSUN4OBJ)
L_OBJS =        $(OBJS:%.o=%.ln)
L_SRCS =        $(PLAT_PFILES:%=$(PLATDIR)/%)
L_SRCS +=       $(PLAT_PSUN4FILES:%=$(PLATSUN4DIR)/%)

ARCHOPTS=       -Dsun4v
ASFLAGS =       -P -D__STDC__ -D_BOOT -D_ASM
CPPDEFS =       $(ARCHOPTS) -D$(KARCH) -D_BOOT -D_KERNEL -D_MACHDEP
CPPINCS =       -I. -I$(SYSDIR)/sun4 -I$(SYSDIR)/$(KARCH) -I$(SYSDIR)/$(MMU) \
                -I$(SYSDIR)/sun4u \
                -I$(SYSDIR)/sparc/v9 -I$(SYSDIR)/sparc \
                -I$(SYSDIR)/sun -I$(SYSDIR)/common
CPPFLAGS=       $(CPPDEFS) $(CPPINCS) $(CPPFLAGS.master)
CFLAGS +=       $(CCVERBOSE)

.KEEP_STATE:

.PARALLEL:      $(OBJS) $(L_OBJS)

all install: $(LIBPLAT)

lint: $(LINTLIBPLAT)

clean:
        $(RM) $(OBJS) $(L_OBJS)

clobber: clean
        $(RM) $(LIBPLAT) $(LINTLIBPLAT)

$(LIBPLAT): $(OBJSDIR) .WAIT $(OBJS)
        $(BUILD.AR) $(OBJS)

$(LINTLIBPLAT): $(OBJSDIR) .WAIT $(L_OBJS)
        @$(ECHO) "\nlint library construction:" $@
        @$(LINT.lib) -o plat $(L_SRCS)

$(OBJSDIR):
        -@[ -d $@ ] || mkdir $@

#
# build rules using standard library object subdirectory
#
$(OBJSDIR)/%.o: $(PLATDIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

$(OBJSDIR)/%.o: $(PLATDIR)/%.s
        $(COMPILE.s) -o $@ $<
        $(POST_PROCESS_O)

$(OBJSDIR)/%.o: $(PLATSUN4DIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

$(OBJSDIR)/%.o: $(PLATSUN4DIR)/%.s
        $(COMPILE.s) -o $@ $<
```

```
126         $(POST_PROCESS_O)

128 $(OBJSDIR)/%.ln: $(PLATDIR)/%.c
129         @($(LHEAD) $(LINT.c) $< $(LTAIL))
130         @$(MV) $(@F) $@

132 $(OBJSDIR)/%.ln: $(PLATDIR)/%.s
133         @($(LHEAD) $(LINT.s) $< $(LTAIL))
134         @$(MV) $(@F) $@

136 $(OBJSDIR)/%.ln: $(PLATSUN4DIR)/%.c
137         @($(LHEAD) $(LINT.c) $< $(LTAIL))
138         @$(MV) $(@F) $@

140 $(OBJSDIR)/%.ln: $(PLATSUN4DIR)/%.s
141         @($(LHEAD) $(LINT.s) $< $(LTAIL))
142         @$(MV) $(@F) $@
```

```
**********************************************************
   30878 Wed Aug 28 04:02:56 2019
new/usr/src/uts/common/io/openprom.c
11630 remove checks for 64-bit capable hardware
**********************************************************
```

```
   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  * Copyright 2019 Peter Tribble.
  25  */

  27 /*
  28  * Ported from 4.1.1_PSRA: "@(#)openprom.c 1.19 91/02/19 SMI";
  29  *
  30  * Porting notes:
  31  *
  32  * OPROMU2P unsupported after SunOS 4.x.
  33  *
  34  * Only one of these devices per system is allowed.
  35  */

  37 /*
  38  * Openprom eeprom options/devinfo driver.
  39  */

  41 #include <sys/types.h>
  42 #include <sys/errno.h>
  43 #include <sys/file.h>
  44 #include <sys/cmn_err.h>
  45 #include <sys/kmem.h>
  46 #include <sys/openpromio.h>
  47 #include <sys/conf.h>
  48 #include <sys/stat.h>
  49 #include <sys/modctl.h>
  50 #include <sys/debug.h>
  51 #include <sys/autoconf.h>
  52 #include <sys/ddi.h>
  53 #include <sys/sunddi.h>
  54 #include <sys/promif.h>
  55 #include <sys/sysmacros.h>      /* offsetof */
  56 #include <sys/nvpair.h>
  57 #include <sys/zone.h>
  58 #include <sys/consplat.h>
  59 #include <sys/bootconf.h>
  60 #include <sys/systm.h>
  61 #include <sys/bootprops.h>
```



```
  63 #define MAX_OPENS       32      /* Up to this many simultaneous opens */

  65 #define IOC_IDLE        0       /* snapshot ioctl states */
  66 #define IOC_SNAP        1       /* snapshot in progress */
  67 #define IOC_DONE        2       /* snapshot done, but not copied out */
  68 #define IOC_COPY        3       /* copyout in progress */

  70 /*
  71  * XXX  Make this dynamic.. or (better still) make the interface stateless
  72  */
  73 static struct oprom_state {
  74         pnode_t current_id;     /* node we're fetching props from */
  75         int16_t already_open;   /* if true, this instance is 'active' */
  76         int16_t ioc_state;      /* snapshot ioctl state */
  77         char    *snapshot;      /* snapshot of all prom nodes */
  78         size_t  size;           /* size of snapshot */
  79         prom_generation_cookie_t tree_gen;
  80 } oprom_state[MAX_OPENS];
_____unchanged_portion_omitted_

 352 /*ARGSUSED*/
 353 static int
 354 opromioctl_cb(void *avp, int has_changed)
 355 {
 356         struct opromioctl_args *argp = avp;
 357         int cmd;
 358         intptr_t arg;
 359         int mode;
 360         struct oprom_state *st;
 361         struct openpromio *opp;
 362         int valsize;
 363         char *valbuf;
 364         int error = 0;
 365         uint_t userbufsize;
 366         pnode_t node_id;
 367         char propname[OBP_MAXPROPNAME];

 369         st = argp->st;
 370         cmd = argp->cmd;
 371         arg = argp->arg;
 372         mode = argp->mode;

 374         if (has_changed) {
 375                 /*
 376                  * The prom tree has changed since we last used current_id,
 377                  * so we need to check it.
 378                  */
 379                 if ((st->current_id != OBP_NONODE) &&
 380                     (st->current_id != OBP_BADNODE)) {
 381                         if (oprom_checknodeid(st->current_id, OBP_NONODE) == 0)
 382                                 st->current_id = OBP_BADNODE;
 383                 }
 384         }

 386         /*
 387          * Check permissions
 388          * and weed out unsupported commands on x86 platform
 389          */
 390         switch (cmd) {
 391 #if !defined(__i386) && !defined(__amd64)
 392         case OPROMLISTKEYSLEN:
 393                 valsize = prom_asr_list_keys_len();
 394                 opp = (struct openpromio *)kmem_zalloc(
 395                     sizeof (uint_t) + 1, KM_SLEEP);
 396                 opp->oprom_size = valsize;
```

```
397                 if (copyout(opp, (void *)arg, (sizeof (uint_t))) != 0)
398                         error = EFAULT;
399                 kmem_free(opp, sizeof (uint_t) + 1);
400                 break;
401         case OPROMLISTKEYS:
402                 valsize = prom_asr_list_keys_len();
403                 if (copyin((void *)arg, &userbufsize, sizeof (uint_t)) != 0)
404                         return (EFAULT);
405                 if (valsize > userbufsize)
406                         return (EINVAL);
407                 valbuf = (char *)kmem_zalloc(valsize + 1, KM_SLEEP);
408                 if (prom_asr_list_keys((caddr_t)valbuf) == -1) {
409                         kmem_free(valbuf, valsize + 1);
410                         return (EFAULT);
411                 }
412                 opp = (struct openpromio *)kmem_zalloc(
413                     valsize + sizeof (uint_t) + 1, KM_SLEEP);
414                 opp->oprom_size = valsize;
415                 bcopy(valbuf, opp->oprom_array, valsize);
416                 if (copyout(opp, (void *)arg, (valsize + sizeof (uint_t))) != 0)
417                         error = EFAULT;
418                 kmem_free(valbuf, valsize + 1);
419                 kmem_free(opp, valsize + sizeof (uint_t) + 1);
420                 break;
421         case OPROMEXPORT:
422                 valsize = prom_asr_export_len();
423                 if (copyin((void *)arg, &userbufsize, sizeof (uint_t)) != 0)
424                         return (EFAULT);
425                 if (valsize > userbufsize)
426                         return (EINVAL);
427                 valbuf = (char *)kmem_zalloc(valsize + 1, KM_SLEEP);
428                 if (prom_asr_export((caddr_t)valbuf) == -1) {
429                         kmem_free(valbuf, valsize + 1);
430                         return (EFAULT);
431                 }
432                 opp = (struct openpromio *)kmem_zalloc(
433                     valsize + sizeof (uint_t) + 1, KM_SLEEP);
434                 opp->oprom_size = valsize;
435                 bcopy(valbuf, opp->oprom_array, valsize);
436                 if (copyout(opp, (void *)arg, (valsize + sizeof (uint_t))) != 0)
437                         error = EFAULT;
438                 kmem_free(valbuf, valsize + 1);
439                 kmem_free(opp, valsize + sizeof (uint_t) + 1);
440                 break;
441         case OPROMEXPORTLEN:
442                 valsize = prom_asr_export_len();
443                 opp = (struct openpromio *)kmem_zalloc(
444                     sizeof (uint_t) + 1, KM_SLEEP);
445                 opp->oprom_size = valsize;
446                 if (copyout(opp, (void *)arg, (sizeof (uint_t))) != 0)
447                         error = EFAULT;
448                 kmem_free(opp, sizeof (uint_t) + 1);
449                 break;
450 #endif
451         case OPROMGETOPT:
452         case OPROMNXTOPT:
453                 if ((mode & FREAD) == 0) {
454                         return (EPERM);
455                 }
456                 node_id = options_nodeid;
457                 break;

459         case OPROMSETOPT:
460         case OPROMSETOPT2:
461 #if !defined(__i386) && !defined(__amd64)
462                 if (mode & FWRITE) {
```



```
463                         node_id = options_nodeid;
464                         break;
465                 }
466 #endif /* !__i386 && !__amd64 */
467                 return (EPERM);

469         case OPROMNEXT:
470         case OPROMCHILD:
471         case OPROMGETPROP:
472         case OPROMGETPROPLEN:
473         case OPROMNXTPROP:
474         case OPROMSETNODEID:
475                 if ((mode & FREAD) == 0) {
476                         return (EPERM);
477                 }
478                 node_id = st->current_id;
479                 break;
480         case OPROMCOPYOUT:
481                 if (st->snapshot == NULL)
482                         return (EINVAL);
483                 /*FALLTHROUGH*/
484         case OPROMSNAPSHOT:
485         case OPROMGETCONS:
486         case OPROMGETBOOTARGS:
487         case OPROMGETBOOTPATH:
488         case OPROMGETVERSION:
489         case OPROMPATH2DRV:
490         case OPROMPROM2DEVNAME:
491 #if !defined(__i386) && !defined(__amd64)
492         case OPROMGETFBNAME:
493         case OPROMDEV2PROMNAME:
493         case OPROMREADY64:
494 #endif  /* !__i386 && !__amd64 */
495                 if ((mode & FREAD) == 0) {
496                         return (EPERM);
497                 }
498                 break;

500         default:
501                 return (EINVAL);
502         }

504         /*
505          * Deal with SNAPSHOT and COPYOUT ioctls first
506          */
507         switch (cmd) {
508         case OPROMCOPYOUT:
509                 return (oprom_copyout(st, arg));

511         case OPROMSNAPSHOT:
512                 return (oprom_snapshot(st, arg));
513         }

515         /*
516          * Copy in user argument length and allocation memory
517          *
518          * NB do not copyin the entire buffer we may not need
519          *      to. userbufsize can be as big as 32 K.
520          */
521         if (copyin((void *)arg, &userbufsize, sizeof (uint_t)) != 0)
522                 return (EFAULT);

524         if (userbufsize == 0 || userbufsize > OPROMMAXPARAM)
525                 return (EINVAL);

527         opp = (struct openpromio *)kmem_zalloc(
```

```
528             userbufsize + sizeof (uint_t) + 1, KM_SLEEP);

530         /*
531          * Execute command
532          */
533         switch (cmd) {

535         case OPROMGETOPT:
536         case OPROMGETPROP:
537         case OPROMGETPROPLEN:

539                 if ((prom_is_openprom() == 0) ||
540                     (node_id == OBP_NONODE) || (node_id == OBP_BADNODE)) {
541                         error = EINVAL;
542                         break;
543                 }

545                 /*
546                  * The argument, a NULL terminated string, is a prop name.
547                  */
548                 if ((error = oprom_copyinstr(arg, opp->oprom_array,
549                     (size_t)userbufsize, OBP_MAXPROPNAME)) != 0) {
550                         break;
551                 }
552                 (void) strcpy(propname, opp->oprom_array);
553                 valsize = prom_getproplen(node_id, propname);

555                 /*
556                  * 4010173: 'name' is a property, but not an option.
557                  */
558                 if ((cmd == OPROMGETOPT) && (strcmp("name", propname) == 0))
559                         valsize = -1;

561                 if (cmd == OPROMGETPROPLEN)  {
562                         int proplen = valsize;

564                         if (userbufsize < sizeof (int)) {
565                                 error = EINVAL;
566                                 break;
567                         }
568                         opp->oprom_size = valsize = sizeof (int);
569                         bcopy(&proplen, opp->oprom_array, valsize);
570                 } else if (valsize > 0 && valsize <= userbufsize) {
571                         bzero(opp->oprom_array, valsize + 1);
572                         (void) prom_getprop(node_id, propname,
573                             opp->oprom_array);
574                         opp->oprom_size = valsize;
575                         if (valsize < userbufsize)
576                                 ++valsize;      /* Forces NULL termination */
577                                                 /* If space permits */
578                 } else {
579                         /*
580                          * XXX: There is no error code if the buf is too small.
581                          * which is consistent with the current behavior.
582                          *
583                          * NB: This clause also handles the non-error
584                          * zero length (boolean) property value case.
585                          */
586                         opp->oprom_size = 0;
587                         (void) strcpy(opp->oprom_array, "");
588                         valsize = 1;
589                 }
590                 if (copyout(opp, (void *)arg, (valsize + sizeof (uint_t))) != 0)
591                         error = EFAULT;
592                 break;
```



```
594         case OPROMNXTOPT:
595         case OPROMNXTPROP:
596                 if ((prom_is_openprom() == 0) ||
597                     (node_id == OBP_NONODE) || (node_id == OBP_BADNODE)) {
598                         error = EINVAL;
599                         break;
600                 }

602                 /*
603                  * The argument, a NULL terminated string, is a prop name.
604                  */
605                 if ((error = oprom_copyinstr(arg, opp->oprom_array,
606                     (size_t)userbufsize, OBP_MAXPROPNAME)) != 0) {
607                         break;
608                 }
609                 valbuf = (char *)prom_nextprop(node_id, opp->oprom_array,
610                     propname);
611                 valsize = strlen(valbuf);

613                 /*
614                  * 4010173: 'name' is a property, but it's not an option.
615                  */
616                 if ((cmd == OPROMNXTOPT) && valsize &&
617                     (strcmp(valbuf, "name") == 0)) {
618                         valbuf = (char *)prom_nextprop(node_id, "name",
619                             propname);
620                         valsize = strlen(valbuf);
621                 }

623                 if (valsize == 0) {
624                         opp->oprom_size = 0;
625                 } else if (++valsize <= userbufsize) {
626                         opp->oprom_size = valsize;
627                         bzero((caddr_t)opp->oprom_array, (size_t)valsize);
628                         bcopy((caddr_t)valbuf, (caddr_t)opp->oprom_array,
629                             (size_t)valsize);
630                 }

632                 if (copyout(opp, (void *)arg, valsize + sizeof (uint_t)) != 0)
633                         error = EFAULT;
634                 break;

636         case OPROMNEXT:
637         case OPROMCHILD:
638         case OPROMSETNODEID:

640                 if (prom_is_openprom() == 0 ||
641                     userbufsize < sizeof (pnode_t)) {
642                         error = EINVAL;
643                         break;
644                 }

646                 /*
647                  * The argument is a phandle. (aka pnode_t)
648                  */
649                 if (copyin(((caddr_t)arg + sizeof (uint_t)),
650                     opp->oprom_array, sizeof (pnode_t)) != 0) {
651                         error = EFAULT;
652                         break;
653                 }

655                 /*
656                  * If pnode_t from userland is garbage, we
657                  * could confuse the PROM.
658                  */
659                 node_id = *(pnode_t *)opp->oprom_array;
```

```
660                 if (oprom_checknodeid(node_id, st->current_id) == 0) {
661                         cmn_err(CE_NOTE, "!nodeid 0x%x not found",
662                             (int)node_id);
663                         error = EINVAL;
664                         break;
665                 }

667                 if (cmd == OPROMNEXT)
668                         st->current_id = prom_nextnode(node_id);
669                 else if (cmd == OPROMCHILD)
670                         st->current_id = prom_childnode(node_id);
671                 else {
672                         /* OPROMSETNODEID */
673                         st->current_id = node_id;
674                         break;
675                 }

677                 opp->oprom_size = sizeof (pnode_t);
678                 *(pnode_t *)opp->oprom_array = st->current_id;

680                 if (copyout(opp, (void *)arg,
681                     sizeof (pnode_t) + sizeof (uint_t)) != 0)
682                         error = EFAULT;
683                 break;

685         case OPROMGETCONS:
686                 /*
687                  * Is openboot supported on this machine?
688                  * This ioctl used to return the console device,
689                  * information; this is now done via modctl()
690                  * in libdevinfo.
691                  */
692                 opp->oprom_size = sizeof (char);

694                 opp->oprom_array[0] |= prom_is_openprom() ?
695                     OPROMCONS_OPENPROM : 0;

697                 /*
698                  * The rest of the info is needed by Install to
699                  * decide if graphics should be started.
700                  */
701                 if ((getzoneid() == GLOBAL_ZONEID) &&
702                     plat_stdin_is_keyboard()) {
703                         opp->oprom_array[0] |= OPROMCONS_STDIN_IS_KBD;
704                 }

706                 if ((getzoneid() == GLOBAL_ZONEID) &&
707                     plat_stdout_is_framebuffer()) {
708                         opp->oprom_array[0] |= OPROMCONS_STDOUT_IS_FB;
709                 }

711                 if (copyout(opp, (void *)arg,
712                     sizeof (char) + sizeof (uint_t)) != 0)
713                         error = EFAULT;
714                 break;

716         case OPROMGETBOOTARGS: {
717                 extern char kern_bootargs[];

719                 valsize = strlen(kern_bootargs) + 1;
720                 if (valsize > userbufsize) {
721                         error = EINVAL;
722                         break;
723                 }
724                 (void) strcpy(opp->oprom_array, kern_bootargs);
725                 opp->oprom_size = valsize - 1;
```



```
727                 if (copyout(opp, (void *)arg, valsize + sizeof (uint_t)) != 0)
728                         error = EFAULT;
729                 break;
730         }

732         case OPROMGETBOOTPATH: {
733 #if defined(__sparc) && defined(_OBP)

735                 char bpath[OBP_MAXPATHLEN];
736                 if (get_bootpath_prop(bpath) != 0) {
737                         error = EINVAL;
738                         break;
739                 }
740                 valsize = strlen(bpath) + 1;
741                 if (valsize > userbufsize) {
742                         error = EINVAL;
743                         break;
744                 }
745                 (void) strcpy(opp->oprom_array, bpath);

747 #elif defined(__i386) || defined(__amd64)

749                 extern char saved_cmdline[];
750                 valsize = strlen(saved_cmdline) + 1;
751                 if (valsize > userbufsize) {
752                         error = EINVAL;
753                         break;
754                 }
755                 (void) strcpy(opp->oprom_array, saved_cmdline);
756 #endif
757                 opp->oprom_size = valsize - 1;
758                 if (copyout(opp, (void *)arg, valsize + sizeof (uint_t)) != 0)
759                         error = EFAULT;
760                 break;
761         }

763         /*
764          * convert a prom device path to an equivalent devfs path
765          */
766         case OPROMPROM2DEVNAME: {
767                 char *dev_name;

769                 /*
770                  * The input argument, a pathname, is a NULL terminated string.
771                  */
772                 if ((error = oprom_copyinstr(arg, opp->oprom_array,
773                     (size_t)userbufsize, MAXPATHLEN)) != 0) {
774                         break;
775                 }

777                 dev_name = kmem_alloc(MAXPATHLEN, KM_SLEEP);

779                 error = i_promname_to_devname(opp->oprom_array, dev_name);
780                 if (error != 0) {
781                         kmem_free(dev_name, MAXPATHLEN);
782                         break;
783                 }
784                 valsize = opp->oprom_size = strlen(dev_name);
785                 if (++valsize > userbufsize) {
786                         kmem_free(dev_name, MAXPATHLEN);
787                         error = EINVAL;
788                         break;
789                 }
790                 (void) strcpy(opp->oprom_array, dev_name);
791                 if (copyout(opp, (void *)arg, sizeof (uint_t) + valsize) != 0)
```

```
792                         error = EFAULT;

794                 kmem_free(dev_name, MAXPATHLEN);
795                 break;
796         }

798         /*
799          * Convert a prom device path name to a driver name
800          */
801         case OPROMPATH2DRV: {
802                 char *drv_name;
803                 major_t maj;

805                 /*
806                  * The input argument, a pathname, is a NULL terminated string.
807                  */
808                 if ((error = oprom_copyinstr(arg, opp->oprom_array,
809                     (size_t)userbufsize, MAXPATHLEN)) != 0) {
810                         break;
811                 }

813                 /*
814                  * convert path to a driver binding name
815                  */
816                 maj = path_to_major((char *)opp->oprom_array);
817                 if (maj == DDI_MAJOR_T_NONE) {
818                         error = EINVAL;
819                         break;
820                 }

822                 /*
823                  * resolve any aliases
824                  */
825                 if ((drv_name = ddi_major_to_name(maj)) == NULL) {
826                         error = EINVAL;
827                         break;
828                 }

830                 (void) strcpy(opp->oprom_array, drv_name);
831                 opp->oprom_size = strlen(drv_name);
832                 if (copyout(opp, (void *)arg,
833                     sizeof (uint_t) + opp->oprom_size + 1) != 0)
834                         error = EFAULT;
835                 break;
836         }

838         case OPROMGETVERSION:
839                 /*
840                  * Get a string representing the running version of the
841                  * prom. How to create such a string is platform dependent,
842                  * so we just defer to a promif function. If no such
843                  * association exists, the promif implementation
844                  * may copy the string "unknown" into the given buffer,
845                  * and return its length (incl. NULL terminator).
846                  *
847                  * We expect prom_version_name to return the actual
848                  * length of the string, but copy at most userbufsize
849                  * bytes into the given buffer, including NULL termination.
850                  */

852                 valsize = prom_version_name(opp->oprom_array, userbufsize);
853                 if (valsize < 0) {
854                         error = EINVAL;
855                         break;
856                 }
```



```
858                 /*
859                  * copyout only the part of the user buffer we need to.
860                  */
861                 if (copyout(opp, (void *)arg,
862                     (size_t)(min((uint_t)valsize, userbufsize) +
863                     sizeof (uint_t))) != 0)
864                         error = EFAULT;
865                 break;

867 #if !defined(__i386) && !defined(__amd64)
868         case OPROMGETFBNAME:
869                 /*
870                  * Return stdoutpath, if it's a frame buffer.
871                  * Yes, we are comparing a possibly longer string against
872                  * the size we're really going to copy, but so what?
873                  */
874                 if ((getzoneid() == GLOBAL_ZONEID) &&
875                     (prom_stdout_is_framebuffer() != 0) &&
876                     (userbufsize > strlen(prom_stdoutpath()))) {
877                         prom_strip_options(prom_stdoutpath(),
878                             opp->oprom_array);  /* strip options and copy */
879                         valsize = opp->oprom_size = strlen(opp->oprom_array);
880                         if (copyout(opp, (void *)arg,
881                             valsize + 1 + sizeof (uint_t)) != 0)
882                                 error = EFAULT;
883                 } else
884                         error = EINVAL;
885                 break;

887         /*
888          * Convert a logical or physical device path to prom device path
889          */
890         case OPROMDEV2PROMNAME: {
891                 char *prom_name;

893                 /*
894                  * The input argument, a pathname, is a NULL terminated string.
895                  */
896                 if ((error = oprom_copyinstr(arg, opp->oprom_array,
897                     (size_t)userbufsize, MAXPATHLEN)) != 0) {
898                         break;
899                 }

901                 prom_name = kmem_alloc(userbufsize, KM_SLEEP);

903                 /*
904                  * convert the devfs path to an equivalent prom path
905                  */
906                 error = i_devname_to_promname(opp->oprom_array, prom_name,
907                     userbufsize);

909                 if (error != 0) {
910                         kmem_free(prom_name, userbufsize);
911                         break;
912                 }

914                 for (valsize = 0; valsize < userbufsize; valsize++) {
915                         opp->oprom_array[valsize] = prom_name[valsize];

917                         if ((valsize > 0) && (prom_name[valsize] == '\0') &&
918                             (prom_name[valsize-1] == '\0')) {
919                                 break;
920                         }
921                 }
922                 opp->oprom_size = valsize;
```

```
924                 kmem_free(prom_name, userbufsize);
925                 if (copyout(opp, (void *)arg, sizeof (uint_t) + valsize) != 0)
926                         error = EFAULT;
927
928                 break;
929         }
930
931         case OPROMSETOPT:
932         case OPROMSETOPT2: {
933                 int namebuflen;
934                 int valbuflen;
935
936                 if ((prom_is_openprom() == 0) ||
937                     (node_id == OBP_NONODE) || (node_id == OBP_BADNODE)) {
938                         error = EINVAL;
939                         break;
940                 }
941
942                 /*
943                  * The arguments are a property name and a value.
944                  * Copy in the entire user buffer.
945                  */
946                 if (copyin(((caddr_t)arg + sizeof (uint_t)),
947                     opp->oprom_array, userbufsize) != 0) {
948                         error = EFAULT;
949                         break;
950                 }
951
952                 /*
953                  * The property name is the first string, value second
954                  */
955                 namebuflen = strlen(opp->oprom_array);
956                 valbuf = opp->oprom_array + namebuflen + 1;
957                 valbuflen = strlen(valbuf);
958
959                 if (cmd == OPROMSETOPT) {
960                         valsize = valbuflen + 1;  /* +1 for the '\0' */
961                 } else {
962                         if ((namebuflen + 1 + valbuflen + 1) > userbufsize) {
963                                 error = EINVAL;
964                                 break;
965                         }
966                         valsize = (opp->oprom_array + userbufsize) - valbuf;
967                 }
968
969                 /*
970                  * 4010173: 'name' is not an option, but it is a property.
971                  */
972                 if (strcmp(opp->oprom_array, "name") == 0)
973                         error = EINVAL;
974                 else if (prom_setprop(node_id, opp->oprom_array,
975                     valbuf, valsize) < 0)
976                         error = EINVAL;
977
978                 break;
979         }
980
981         case OPROMREADY64: {
982                 struct openprom_opr64 *opr =
983                     (struct openprom_opr64 *)opp->oprom_array;
984                 int i;
985                 pnode_t id;
986
987                 if (userbufsize < sizeof (*opr)) {
988                         error = EINVAL;
989                         break;
```



```
990                 }
991
992                 valsize = userbufsize -
993                     offsetof(struct openprom_opr64, message);
994
995                 i = prom_version_check(opr->message, valsize, &id);
996                 opr->return_code = i;
997                 opr->nodeid = (int)id;
998
999                 valsize = offsetof(struct openprom_opr64, message);
1000                valsize += strlen(opr->message) + 1;
1001
1002                /*
1003                 * copyout only the part of the user buffer we need to.
1004                 */
1005                if (copyout(opp, (void *)arg,
1006                    (size_t)(min((uint_t)valsize, userbufsize) +
1007                    sizeof (uint_t))) != 0)
1008                        error = EFAULT;
1009                break;
1010
1011        }       /* case OPROMREADY64 */
 980 #endif  /* !__i386 && !__amd64 */
 981         }       /* switch (cmd) */
 982
 983         kmem_free(opp, userbufsize + sizeof (uint_t) + 1);
 984         return (error);
 985 }
______unchanged_portion_omitted_
```

```
**********************************************************
    4139 Wed Aug 28 04:02:56 2019
new/usr/src/uts/common/sys/openpromio.h
11630 remove checks for 64-bit capable hardware
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2010 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2019 Peter Tribble.
 */

#ifndef _SYS_OPENPROMIO_H
#define _SYS_OPENPROMIO_H

/* From SunOS 4.1.1 <sundev/openpromio.h> */

#ifdef  __cplusplus
extern "C" {
#endif

/*
 * XXX HACK ALERT
 *
 * You might think that this interface could support setting non-ASCII
 * property values.  Unfortunately the 4.0.3c openprom driver SETOPT
 * code ignores oprom_size and uses strlen() to compute the length of
 * the value.  The 4.0.3c openprom eeprom command makes its contribution
 * by not setting oprom_size to anything meaningful.  So, if we want the
 * driver to trust oprom_size we have to use SETOPT2.  XXX.
 */
struct openpromio {
        uint_t  oprom_size;             /* real size of following array */
        union {
                char    b[1];           /* For property names and values */
                                        /* NB: Adjacent, Null terminated */
                int     i;
        } opio_u;
};

#define oprom_array     opio_u.b
#define oprom_node      opio_u.i
#define oprom_len       opio_u.i

/*
 * OPROMMAXPARAM is used as a limit by the driver, and it has been
 * increased to be 4 times the largest possible size of a property,
```



```
 * which is 8K (nvramrc property).
 */
#define OPROMMAXPARAM   32768           /* max size of array */

/*
 * Note that all OPROM ioctl codes are type void. Since the amount
 * of data copied in/out may (and does) vary, the openprom driver
 * handles the copyin/copyout itself.
 */
#define OIOC    ('O'<<8)
#define OPROMGETOPT     (OIOC | 1)
#define OPROMSETOPT     (OIOC | 2)
#define OPROMNXTOPT     (OIOC | 3)
#define OPROMSETOPT2    (OIOC | 4)      /* working OPROMSETOPT */
#define OPROMNEXT       (OIOC | 5)      /* interface to raw config_ops */
#define OPROMCHILD      (OIOC | 6)      /* interface to raw config_ops */
#define OPROMGETPROP    (OIOC | 7)      /* interface to raw config_ops */
#define OPROMNXTPROP    (OIOC | 8)      /* interface to raw config_ops */
#define OPROMU2P        (OIOC | 9)      /* NOT SUPPORTED after 4.x */
#define OPROMGETCONS    (OIOC | 10)     /* enquire which console device */
#define OPROMGETFBNAME  (OIOC | 11)     /* Frame buffer OBP pathname */
#define OPROMGETBOOTARGS (OIOC | 12)    /* Get boot arguments */
#define OPROMGETVERSION (OIOC | 13)     /* Get OpenProm Version string */
#define OPROMPATH2DRV   (OIOC | 14)     /* Convert prom path to driver name */
#define OPROMDEV2PROMNAME (OIOC | 15)   /* Convert devfs path to prom path */
#define OPROMPROM2DEVNAME (OIOC | 16)   /* Convert devfs path to prom path */
#define OPROMGETPROPLEN (OIOC | 17)     /* interface to raw config_ops */
#define OPROMREADY64    (OIOC | 18)     /* DEPRECATED is prom 64-bit ready? */
#define OPROMREADY64    (OIOC | 18)     /* is prom 64-bit ready? */
#define OPROMSETNODEID  (OIOC | 19)     /* set current node_id */
#define OPROMSNAPSHOT   (OIOC | 20)     /* create a snapshot */
#define OPROMCOPYOUT    (OIOC | 21)     /* copyout and free snapshot */
#define OPROMLISTKEYS   (OIOC | 22)     /* asr-list-keys */
#define OPROMLISTKEYSLEN (OIOC | 23)    /* asr-list-keys-len */
#define OPROMEXPORT     (OIOC | 24)     /* asr-export */
#define OPROMEXPORTLEN  (OIOC | 25)     /* asr-export-len */
#define OPROMGETBOOTPATH        (OIOC | 26)     /* Get bootpath */

/*
 * Return values from OPROMGETCONS:
 */

#define OPROMCONS_NOT_WSCONS    0
#define OPROMCONS_STDIN_IS_KBD  0x1     /* stdin device is kbd */
#define OPROMCONS_STDOUT_IS_FB  0x2     /* stdout is a framebuffer */
#define OPROMCONS_OPENPROM      0x4     /* supports openboot */

#if defined(__sparc)

/*
 * Data structure returned in oprom_array, from OPROMREADY64:
 *
 * With return codes 1 and 2, also returns nodeid, a nodeid
 * of a flashprom node, and a message string with the minimum version
 * requirement for this platform.
 */
struct openprom_opr64 {
        int return_code;                /* See below */
        int nodeid;                     /* Valid with positive return codes */
        char message[1];                /* NULL terminated message string */
};

/*
 * return_code values from OPROMREADY64:
 */
#define OP64R_READY                     0       /* ready or not applicable */
```

```
126 #define OP64R_UPGRADE_REQUIRED          1       /* Upgrade required */
127 #define OP64R_UPGRADE_RECOMMENDED       2       /* Upgrade recommended */

129 #endif

108 #ifdef __cplusplus
109 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   10195 Wed Aug 28 04:02:56 2019
new/usr/src/uts/sun/sys/promif.h
11630 remove checks for 64-bit capable hardware
**********************************************************
```

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

/*
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2019 Peter Tribble.
 */

#ifndef _SYS_PROMIF_H
#define _SYS_PROMIF_H

#pragma ident   "%Z%%M% %I%     %E% SMI"

#include <sys/types.h>
#include <sys/obpdefs.h>

#if defined(_KERNEL) || defined(_KMDB)
#include <sys/va_list.h>
#endif

#ifdef __cplusplus
extern "C" {
#endif

/*
 *  These are for V0 ops only.  We sometimes have to specify
 *  to promif which type of operation we need to perform
 *  and since we can't get such a property from a V0 prom, we
 *  sometimes just assume it.  V2 and later proms do the right thing.
 */
#define BLOCK   0
#define NETWORK 1
#define BYTE    2


#if defined(_KERNEL) || defined(_KMDB)

#if !defined(_BOOT)
/*
 * Due to FCode on sun4u machines running in a pseudo-32-bit environment
 * we need to enable code in several of the promif routines to ensure
 * that 64-bit pointers from the kernel are not passed through the CIF
```



```
 * to OpenBoot.
 *
 * Client programs defining this token need to provide two callbacks to
 * allow the promif routines to allocate and free memory allocated from
 * the bottom 32-bits of the 64-bit address space:
 *
 *      void *promplat_alloc(size_t);
 *      void promplat_free(void *, size_t);
 *
 * The alloc function should guarantee that it will never return an
 * invalid pointer.
 */
#define PROM_32BIT_ADDRS
#endif /* _BOOT */

typedef void promif_preprom_f(void);
typedef void promif_postprom_f(void);

/*
 * resource allocation group: OBP and IEEE 1275-1994.
 * prom_alloc is platform dependent on SPARC.
 */
extern  caddr_t         prom_alloc(caddr_t virthint, size_t size, uint_t align);
extern  void            prom_free(caddr_t virt, size_t size);

/*
 * Device tree and property group: OBP and IEEE 1275-1994.
 */
extern  pnode_t         prom_childnode(pnode_t nodeid);
extern  pnode_t         prom_nextnode(pnode_t nodeid);
extern  pnode_t         prom_parentnode(pnode_t nodeid);
extern  pnode_t         prom_rootnode(void);
extern  pnode_t         prom_chosennode(void);
extern  pnode_t         prom_alias_node(void);
extern  pnode_t         prom_optionsnode(void);

extern  int             prom_asr_list_keys_len();
extern  int             prom_asr_list_keys(caddr_t value);
extern  int             prom_asr_export_len();
extern  int             prom_asr_export(caddr_t value);
extern  int             prom_asr_disable(char *keystr, int keystr_len,
                            char *reason, int reason_len);
extern  int             prom_asr_enable(char *keystr, int keystr_len);

extern  int             prom_getproplen(pnode_t nodeid, caddr_t name);
extern  int             prom_getprop(pnode_t nodeid, caddr_t name,
                            caddr_t value);
extern  caddr_t         prom_nextprop(pnode_t nodeid, caddr_t previous,
                            caddr_t next);
extern  int             prom_setprop(pnode_t nodeid, caddr_t name,
                            caddr_t value, int len);

extern  int             prom_getnode_byname(pnode_t id, char *name);
extern  int             prom_devicetype(pnode_t id, char *type);

extern  char            *prom_decode_composite_string(void *buf,
                            size_t buflen, char *prev);

/*
 * Device tree and property group: IEEE 1275-1994 Only.
 */
extern  pnode_t         prom_finddevice(char *path);    /* Also on obp2.x */

extern  int             prom_bounded_getprop(pnode_t nodeid,
                            caddr_t name, caddr_t buffer, int buflen);
```

```
126 extern  phandle_t       prom_getphandle(ihandle_t i);

128 /*
129  * Device pathnames and pathname conversion: OBP and IEEE 1275-1994.
130  */
131 extern  int             prom_devname_from_pathname(char *path, char *buffer);
132 extern  char            *prom_path_options(char *pathname);
133 extern  char            *prom_path_gettoken(char *from, char *to);
134 extern  void            prom_pathname(char *pathname);
135 extern  void            prom_strip_options(char *from, char *to);

137 /*
138  * Device pathnames and pathname conversion: IEEE 1275-1994 only.
139  */
140 extern  int             prom_ihandle_to_path(ihandle_t, char *buf,
141                             uint_t buflen);
142 extern  int             prom_phandle_to_path(phandle_t, char *buf,
143                             uint_t buflen);

145 /*
146  * Special device nodes: OBP and IEEE 1275-1994.
147  */
148 extern  ihandle_t       prom_stdin_ihandle(void);
149 extern  ihandle_t       prom_stdout_ihandle(void);
150 extern  pnode_t         prom_stdin_node(void);
151 extern  pnode_t         prom_stdout_node(void);
152 extern  char            *prom_stdinpath(void);
153 extern  char            *prom_stdoutpath(void);
154 extern  int             prom_stdin_devname(char *buffer);
155 extern  int             prom_stdout_devname(char *buffer);
156 extern  int             prom_stdin_is_keyboard(void);
157 extern  int             prom_stdout_is_framebuffer(void);
158 extern  int             prom_stdin_stdout_equivalence(void);

160 extern void             prom_get_tem_inverses(int *, int *);
161 extern void             prom_get_tem_size(size_t *, size_t *);
162 extern void             prom_get_tem_pos(uint32_t *, uint32_t *);
163 extern void             prom_get_term_font_size(int *, int *);
164 extern void             prom_hide_cursor(void);

166 /*
167  * Special device nodes: IEEE 1275-1994 only.
168  */
169 extern  ihandle_t       prom_memory_ihandle(void);
170 extern  ihandle_t       prom_mmu_ihandle(void);

172 /*
173  * Administrative group: OBP and IEEE 1275-1994.
174  */
175 extern  void            prom_enter_mon(void);
176 extern  void            prom_exit_to_mon(void)
177         __NORETURN;
178 extern  void            prom_reboot(char *bootstr);

180 extern  void            prom_panic(char *string)
181         __NORETURN;

183 extern  int             prom_getversion(void);
184 extern  int             prom_is_openprom(void);
185 extern  int             prom_is_p1275(void);
186 extern  int             prom_version_name(char *buf, int buflen);
188 extern  int             prom_version_check(char *buf, size_t len, pnode_t *n);

188 extern  void            *prom_mon_id(void);     /* SMCC/OBP platform centric */

190 extern  uint_t          prom_gettime(void);
```



```
192 extern  char            *prom_bootpath(void);
193 extern  char            *prom_bootargs(void);

195 extern  void            prom_interpret(char *str, uintptr_t arg1,
196                             uintptr_t arg2, uintptr_t arg3, uintptr_t arg4,
197                             uintptr_t arg5);

199 /*
202  * Return code values from prom_version_check:
203  *
204  * This routine uses past-prediction mode to determine if the firmware
205  * on the current system is 64-bit ready.
206  *
207  * return code 2 could happen on a board-based server with a slave CPU board
208  * running down-rev firmware and the current master running adequate fw.
209  */
210 #define PROM_VER64_OK           0       /* Prom is 64-bit ready (or n/a) */
211 #define PROM_VER64_UPGRADE      1       /* Down-rev firmware is running */
212 #define PROM_VER64_SUGGEST      2       /* Down-rev firmware detected .. */
213                                         /* .. but not currently active */

215 /*
200  * Administrative group: OBP only.
201  */
202 extern  int             prom_sethandler(void (*v0_func)(), void (*v2_func)());

204 extern  struct bootparam *prom_bootparam(void);

206 /*
207  * Administrative group: IEEE 1275-1994 only.
208  */
209 extern void             *prom_set_callback(void *handler);
210 extern void             prom_set_symbol_lookup(void *sym2val, void *val2sym);

212 /*
213  * Administrative group: IEEE 1275 only.
214  */
215 extern  int             prom_test(char *service);
216 extern  int             prom_test_method(char *method, pnode_t node);

218 /*
219  * Promif support group: Generic.
220  */
221 extern  void            prom_init(char *progname, void *prom_cookie);

223 extern  void            prom_set_preprom(promif_preprom_f *);
224 extern  void            prom_set_postprom(promif_postprom_f *);

226 extern  void            prom_get_tem_pos(uint32_t *, uint32_t *);
227 extern  void            prom_get_tem_size(size_t *, size_t *);

229 typedef struct          __promif_redir_arg *promif_redir_arg_t;
230 typedef ssize_t         (*promif_redir_t)(promif_redir_arg_t,
231                                 uchar_t *, size_t);
232 extern  void            prom_set_stdout_redirect(promif_redir_t,
233                                 promif_redir_arg_t);

235 extern  void            prom_suspend_prepost(void);
236 extern  void            prom_resume_prepost(void);

238 extern  void            (*prom_set_nextprop_preprom(void (*)(void)))(void);
239 extern  void            (*prom_set_nextprop_postprom(void (*)(void)))(void);

241 extern  void            prom_montrap(void (*funcptr)());
```

```
243 typedef uint_t          prom_generation_cookie_t;

245 extern  int             prom_tree_access(int (*callback)(void *arg,
246                                 int has_changed), void *arg,
247                                 prom_generation_cookie_t *);
248 extern  int             prom_tree_update(int (*callback)(void *arg), void *arg);

250 /*
251  * I/O Group: OBP and IEEE 1275.
252  */
253 extern  uchar_t         prom_getchar(void);
254 extern  void            prom_putchar(char c);
255 extern  int             prom_mayget(void);
256 extern  int             prom_mayput(char c);

258 extern  int             prom_open(char *name);
259 extern  int             prom_close(int fd);
260 extern  ssize_t         prom_read(ihandle_t fd, caddr_t buf, size_t len,
261                             uint_t startblk, char type);
262 extern  ssize_t         prom_write(ihandle_t fd, caddr_t buf, size_t len,
263                             uint_t startblk, char type);
264 extern  int             prom_seek(int fd, u_longlong_t offset);

266 extern  void            prom_writestr(const char *buf, size_t bufsize);
267 extern  void            prom_pnode_to_pathname(pnode_t, char *);

269 /*PRINTFLIKE1*/
270 extern  void            prom_printf(const char *fmt, ...)
271         __PRINTFLIKE(1);
272 #pragma rarely_called(prom_printf)

274 extern  void            prom_vprintf(const char *fmt, __va_list adx)
275         __VPRINTFLIKE(1);
276 #pragma rarely_called(prom_vprintf)

278 /*PRINTFLIKE2*/
279 extern  char            *prom_sprintf(char *s, const char *fmt, ...)
280         __PRINTFLIKE(2);
281 extern  char            *prom_vsprintf(char *s, const char *fmt, __va_list adx)
282         __VPRINTFLIKE(2);

284 #define PROM_WALK_CONTINUE      0       /* keep walking to next node */
285 #define PROM_WALK_TERMINATE     1       /* abort walk now */

287 extern  void            prom_walk_devs(pnode_t node,
288                             int (*f)(pnode_t, void *, void *),
289                             void *arg, void *result);

291 extern  pnode_t         prom_findnode_byname(pnode_t id, char *name);
292 extern  pnode_t         prom_findnode_bydevtype(pnode_t id, char *devtype);

294 #define PROM_STOP       {       \
295         prom_printf("File %s line %d\n", __FILE__, __LINE__); \
296         prom_enter_mon();       \
297 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   95898 Wed Aug 28 04:02:56 2019
new/usr/src/uts/sun4/os/startup.c
11630 remove checks for 64-bit capable hardware
**********************************************************
```

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

/*
 * Copyright (c) 2003, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2016 by Delphix. All rights reserved.
 * Copyright 2019 Peter Tribble.
 */

#include <sys/machsystm.h>
#include <sys/archsystm.h>
#include <sys/vm.h>
#include <sys/cpu.h>
#include <sys/atomic.h>
#include <sys/reboot.h>
#include <sys/kdi.h>
#include <sys/bootconf.h>
#include <sys/memlist_plat.h>
#include <sys/memlist_impl.h>
#include <sys/prom_plat.h>
#include <sys/prom_isa.h>
#include <sys/autoconf.h>
#include <sys/ivintr.h>
#include <sys/fpu/fpusystm.h>
#include <sys/iommutsb.h>
#include <vm/vm_dep.h>
#include <vm/seg_dev.h>
#include <vm/seg_kmem.h>
#include <vm/seg_kpm.h>
#include <vm/seg_map.h>
#include <vm/seg_kp.h>
#include <sys/sysconf.h>
#include <vm/hat_sfmmu.h>
#include <sys/kobj.h>
#include <sys/sun4asi.h>
#include <sys/clconf.h>
#include <sys/platform_module.h>
#include <sys/panic.h>
#include <sys/cpu_sgnblk_defs.h>
#include <sys/clock.h>
#include <sys/cmn_err.h>
#include <sys/dumphdr.h>
#include <sys/promif.h>
```



```
#include <sys/prom_debug.h>
#include <sys/traptrace.h>
#include <sys/memnode.h>
#include <sys/mem_cage.h>
#include <sys/mmu.h>
#include <sys/swap.h>

extern void setup_trap_table(void);
extern int cpu_intrq_setup(struct cpu *);
extern void cpu_intrq_register(struct cpu *);
extern void contig_mem_init(void);
extern caddr_t contig_mem_prealloc(caddr_t, pgcnt_t);
extern void mach_dump_buffer_init(void);
extern void mach_descrip_init(void);
extern void mach_descrip_startup_fini(void);
extern void mach_memscrub(void);
extern void mach_fpras(void);
extern void mach_cpu_halt_idle(void);
extern void mach_hw_copy_limit(void);
extern void load_mach_drivers(void);
extern void load_tod_module(void);
#pragma weak load_tod_module

extern int ndata_alloc_mmfsa(struct memlist *ndata);
#pragma weak ndata_alloc_mmfsa

extern void cif_init(void);
#pragma weak cif_init

extern void parse_idprom(void);
extern void add_vx_handler(char *, int, void (*)(cell_t *));
extern void mem_config_init(void);
extern void memseg_remap_init(void);

extern void mach_kpm_init(void);
extern void pcf_init();
extern int size_pse_array(pgcnt_t, int);
extern void pg_init();

/*
 * External Data:
 */
extern int vac_size;    /* cache size in bytes */
extern uint_t vac_mask; /* VAC alignment consistency mask */
extern uint_t vac_colors;

/*
 * Global Data Definitions:
 */

/*
 * XXX - Don't port this to new architectures
 * A 3rd party volume manager driver (vxdm) depends on the symbol romp.
 * 'romp' has no use with a prom with an IEEE 1275 client interface.
 * The driver doesn't use the value, but it depends on the symbol.
 */
void *romp;             /* veritas driver won't load without romp 4154976 */
/*
 * Declare these as initialized data so we can patch them.
 */
pgcnt_t physmem = 0;    /* memory size in pages, patch if you want less */
pgcnt_t segkpsize =
    btop(SEGKPDEFSIZE); /* size of segkp segment in pages */
uint_t segmap_percent = 6; /* Size of segmap segment */

int use_cache = 1;              /* cache not reliable (605 bugs) with MP */
```

```
128 int vac_copyback = 1;
129 char *cache_mode = NULL;
130 int use_mix = 1;
131 int prom_debug = 0;

133 caddr_t boot_tba;               /* %tba at boot - used by kmdb */
134 uint_t  tba_taken_over = 0;

136 caddr_t s_text;                 /* start of kernel text segment */
137 caddr_t e_text;                 /* end of kernel text segment */
138 caddr_t s_data;                 /* start of kernel data segment */
139 caddr_t e_data;                 /* end of kernel data segment */

141 caddr_t modtext;                /* beginning of module text */
142 size_t  modtext_sz;             /* size of module text */
143 caddr_t moddata;                /* beginning of module data reserve */
144 caddr_t e_moddata;              /* end of module data reserve */

146 /*
147  * End of first block of contiguous kernel in 32-bit virtual address space
148  */
149 caddr_t         econtig32;      /* end of first blk of contiguous kernel */

151 caddr_t         ncbase;         /* beginning of non-cached segment */
152 caddr_t         ncend;          /* end of non-cached segment */

154 size_t  ndata_remain_sz;        /* bytes from end of data to 4MB boundary */
155 caddr_t nalloc_base;            /* beginning of nucleus allocation */
156 caddr_t nalloc_end;             /* end of nucleus allocatable memory */
157 caddr_t valloc_base;            /* beginning of kvalloc segment */

159 caddr_t kmem64_base;            /* base of kernel mem segment in 64-bit space */
160 caddr_t kmem64_end;             /* end of kernel mem segment in 64-bit space */
161 size_t  kmem64_sz;              /* bytes in kernel mem segment, 64-bit space */
162 caddr_t kmem64_aligned_end;     /* end of large page, overmaps 64-bit space */
163 int     kmem64_szc;             /* page size code */
164 uint64_t kmem64_pabase = (uint64_t)-1;  /* physical address of kmem64_base */

166 uintptr_t shm_alignment;        /* VAC address consistency modulus */
167 struct memlist *phys_install;   /* Total installed physical memory */
168 struct memlist *phys_avail;     /* Available (unreserved) physical memory */
169 struct memlist *virt_avail;     /* Available (unmapped?) virtual memory */
170 struct memlist *nopp_list;      /* pages with no backing page structs */
171 struct memlist ndata;           /* memlist of nucleus allocatable memory */
172 int memexp_flag;                /* memory expansion card flag */
173 uint64_t ecache_flushaddr;      /* physical address used for flushing E$ */
174 pgcnt_t obp_pages;              /* Physical pages used by OBP */

176 /*
177  * VM data structures
178  */
179 long page_hashsz;               /* Size of page hash table (power of two) */
180 unsigned int page_hashsz_shift; /* log2(page_hashsz) */
181 struct page *pp_base;           /* Base of system page struct array */
182 size_t pp_sz;                   /* Size in bytes of page struct array */
183 struct page **page_hash;        /* Page hash table */
184 pad_mutex_t *pse_mutex;         /* Locks protecting pp->p_selock */
185 size_t pse_table_size;          /* Number of mutexes in pse_mutex[] */
186 int pse_shift;                  /* log2(pse_table_size) */
187 struct seg ktextseg;            /* Segment used for kernel executable image */
188 struct seg kvalloc;             /* Segment used for "valloc" mapping */
189 struct seg kpseg;               /* Segment used for pageable kernel virt mem */
190 struct seg ktexthole;           /* Segment used for nucleus text hole */
191 struct seg kmapseg;             /* Segment used for generic kernel mappings */
192 struct seg kpmseg;              /* Segment used for physical mapping */
193 struct seg kdebugseg;           /* Segment used for the kernel debugger */
```



```
195 void *kpm_pp_base;              /* Base of system kpm_page array */
196 size_t  kpm_pp_sz;              /* Size of system kpm_page array */
197 pgcnt_t kpm_npages;             /* How many kpm pages are managed */

199 struct seg *segkp = &kpseg;     /* Pageable kernel virtual memory segment */
200 struct seg *segkmap = &kmapseg; /* Kernel generic mapping segment */
201 struct seg *segkpm = &kpmseg;   /* 64bit kernel physical mapping segment */

203 int segzio_fromheap = 0;        /* zio allocations occur from heap */
204 caddr_t segzio_base;            /* Base address of segzio */
205 pgcnt_t segziosize = 0;         /* size of zio segment in pages */

207 /*
208  * A static DR page_t VA map is reserved that can map the page structures
209  * for a domain's entire RA space. The pages that backs this space are
210  * dynamically allocated and need not be physically contiguous.  The DR
211  * map size is derived from KPM size.
212  */
213 int ppvm_enable = 0;            /* Static virtual map for page structs */
214 page_t *ppvm_base;              /* Base of page struct map */
215 pgcnt_t ppvm_size = 0;          /* Size of page struct map */

217 /*
218  * debugger pages (if allocated)
219  */
220 struct vnode kdebugvp;

222 /*
223  * VA range available to the debugger
224  */
225 const caddr_t kdi_segdebugbase = (const caddr_t)SEGDEBUGBASE;
226 const size_t kdi_segdebugsize = SEGDEBUGSIZE;

228 /*
229  * Segment for relocated kernel structures in 64-bit large RAM kernels
230  */
231 struct seg kmem64;

233 struct memseg *memseg_free;

235 struct vnode unused_pages_vp;

237 /*
238  * VM data structures allocated early during boot.
239  */
240 size_t pagehash_sz;
241 uint64_t memlist_sz;

243 char tbr_wr_addr_inited = 0;

245 caddr_t mpo_heap32_buf = NULL;
246 size_t  mpo_heap32_bufsz = 0;

248 /*
249  * Static Routines:
250  */
251 static int ndata_alloc_memseg(struct memlist *, size_t);
252 static void memlist_new(uint64_t, uint64_t, struct memlist **);
253 static void memlist_add(uint64_t, uint64_t,
254         struct memlist **, struct memlist **);
255 static void kphysm_init(void);
256 static void kvm_init(void);
257 static void install_kmem64_tte(void);

259 static void startup_init(void);
```

```
 260 static void startup_memlist(void);
 261 static void startup_modules(void);
 262 static void startup_bop_gone(void);
 263 static void startup_vm(void);
 264 static void startup_end(void);
 265 static void setup_cage_params(void);
 266 static void startup_create_io_node(void);

 268 static pgcnt_t npages;
 269 static struct memlist *memlist;
 270 void *memlist_end;

 272 static pgcnt_t bop_alloc_pages;
 273 static caddr_t hblk_base;
 274 uint_t hblk_alloc_dynamic = 0;
 275 uint_t hblk1_min = H1MIN;


 278 /*
 279  * Hook for down-rev firmware
 280  */
 281 static void do_prom_version_check(void);

 283 /*
 279  * After receiving a thermal interrupt, this is the number of seconds
 280  * to delay before shutting off the system, assuming
 281  * shutdown fails.  Use /etc/system to change the delay if this isn't
 282  * large enough.
 283  */
 284 int thermal_powerdown_delay = 1200;

 286 /*
 287  * Used to hold off page relocations into the cage until OBP has completed
 288  * its boot-time handoff of its resources to the kernel.
 289  */
 290 int page_relocate_ready = 0;

 292 /*
 293  * Indicate if kmem64 allocation was done in small chunks
 294  */
 295 int kmem64_smchunks = 0;

 297 /*
 298  * Enable some debugging messages concerning memory usage...
 299  */
 300 #ifdef  DEBUGGING_MEM
 301 static int debugging_mem;
 302 static void
 303 printmemlist(char *title, struct memlist *list)
 304 {
 305         if (!debugging_mem)
 306                 return;

 308         printf("%s\n", title);

 310         while (list) {
 311                 prom_printf("\taddr = 0x%x %8x, size = 0x%x %8x\n",
 312                     (uint32_t)(list->ml_address >> 32),
 313                     (uint32_t)list->ml_address,
 314                     (uint32_t)(list->ml_size >> 32),
 315                     (uint32_t)(list->ml_size));
 316                 list = list->ml_next;
 317         }
 318 }
_____unchanged_portion_omitted_
```



```
1492 static void
1493 startup_modules(void)
1494 {
1495         int nhblk1, nhblk8;
1496         size_t  nhblksz;
1497         pgcnt_t pages_per_hblk;
1498         size_t hme8blk_sz, hme1blk_sz;

1500         /*
1501          * The system file /etc/system was read already under startup_memlist.
1502          */
1503         if (&set_platform_defaults)
1504                 set_platform_defaults();

1506         /*
1507          * Calculate default settings of system parameters based upon
1508          * maxusers, yet allow to be overridden via the /etc/system file.
1509          */
1510         param_calc(0);

1512         mod_setup();

1514         /*
1520          * If we are running firmware that isn't 64-bit ready
1521          * then complain and halt.
1522          */
1523         do_prom_version_check();

1525         /*
1515          * Initialize system parameters
1516          */
1517         param_init();

1519         /*
1520          * maxmem is the amount of physical memory we're playing with.
1521          */
1522         maxmem = physmem;

1524         /* Set segkp limits. */
1525         ncbase = kdi_segdebugbase;
1526         ncend = kdi_segdebugbase;

1528         /*
1529          * Initialize the hat layer.
1530          */
1531         hat_init();

1533         /*
1534          * Initialize segment management stuff.
1535          */
1536         seg_init();

1538         /*
1539          * Create the va>tte handler, so the prom can understand
1540          * kernel translations.  The handler is installed later, just
1541          * as we are about to take over the trap table from the prom.
1542          */
1543         create_va_to_tte();

1545         /*
1546          * Load the forthdebugger (optional)
1547          */
1548         forthdebug_init();

1550         /*
1551          * Create OBP node for console input callbacks
```

```
1552          * if it is needed.
1553          */
1554         startup_create_io_node();

1556         if (modloadonly("fs", "specfs") == -1)
1557                 halt("Can't load specfs");

1559         if (modloadonly("fs", "devfs") == -1)
1560                 halt("Can't load devfs");

1562         if (modloadonly("fs", "procfs") == -1)
1563                 halt("Can't load procfs");

1565         if (modloadonly("misc", "swapgeneric") == -1)
1566                 halt("Can't load swapgeneric");

1568         (void) modloadonly("sys", "lbl_edition");

1570         dispinit();

1572         /*
1573          * Infer meanings to the members of the idprom buffer.
1574          */
1575         parse_idprom();

1577         /* Read cluster configuration data. */
1578         clconf_init();

1580         setup_ddi();

1582         /*
1583          * Lets take this opportunity to load the root device.
1584          */
1585         if (loadrootmodules() != 0)
1586                 debug_enter("Can't load the root filesystem");

1588         /*
1589          * Load tod driver module for the tod part found on this system.
1590          * Recompute the cpu frequency/delays based on tod as tod part
1591          * tends to keep time more accurately.
1592          */
1593         if (&load_tod_module)
1594                 load_tod_module();

1596         /*
1597          * Allow platforms to load modules which might
1598          * be needed after bootops are gone.
1599          */
1600         if (&load_platform_modules)
1601                 load_platform_modules();

1603         setcpudelay();

1605         copy_boot_memlists(&boot_physinstalled, &boot_physinstalled_len,
1606             &boot_physavail, &boot_physavail_len,
1607             &boot_virtavail, &boot_virtavail_len);

1609         /*
1610          * Calculation and allocation of hmeblks needed to remap
1611          * the memory allocated by PROM till now.
1612          * Overestimate the number of hblk1 elements by assuming
1613          * worst case of TTE64K mappings.
1614          * sfmmu_hblk_alloc will panic if this calculation is wrong.
1615          */
1616         bop_alloc_pages = btopr(kmem64_end - kmem64_base);
1617         pages_per_hblk = btop(HMEBLK_SPAN(TTE64K));
```



```
1618         bop_alloc_pages = roundup(bop_alloc_pages, pages_per_hblk);
1619         nhblk1 = bop_alloc_pages / pages_per_hblk + hblk1_min;

1621         bop_alloc_pages = size_virtalloc(boot_virtavail, boot_virtavail_len);

1623         /* sfmmu_init_nucleus_hblks expects properly aligned data structures */
1624         hme8blk_sz = roundup(HME8BLK_SZ, sizeof (int64_t));
1625         hme1blk_sz = roundup(HME1BLK_SZ, sizeof (int64_t));

1627         bop_alloc_pages += btopr(nhblk1 * hme1blk_sz);

1629         pages_per_hblk = btop(HMEBLK_SPAN(TTE8K));
1630         nhblk8 = 0;
1631         while (bop_alloc_pages > 1) {
1632                 bop_alloc_pages = roundup(bop_alloc_pages, pages_per_hblk);
1633                 nhblk8 += bop_alloc_pages /= pages_per_hblk;
1634                 bop_alloc_pages *= hme8blk_sz;
1635                 bop_alloc_pages = btopr(bop_alloc_pages);
1636         }
1637         nhblk8 += 2;

1639         /*
1640          * Since hblk8's can hold up to 64k of mappings aligned on a 64k
1641          * boundary, the number of hblk8's needed to map the entries in the
1642          * boot_virtavail list needs to be adjusted to take this into
1643          * consideration.  Thus, we need to add additional hblk8's since it
1644          * is possible that an hblk8 will not have all 8 slots used due to
1645          * alignment constraints.  Since there were boot_virtavail_len entries
1646          * in that list, we need to add that many hblk8's to the number
1647          * already calculated to make sure we don't underestimate.
1648          */
1649         nhblk8 += boot_virtavail_len;
1650         nhblksz = nhblk8 * hme8blk_sz + nhblk1 * hme1blk_sz;

1652         /* Allocate in pagesize chunks */
1653         nhblksz = roundup(nhblksz, MMU_PAGESIZE);
1654         hblk_base = kmem_zalloc(nhblksz, KM_SLEEP);
1655         sfmmu_init_nucleus_hblks(hblk_base, nhblksz, nhblk8, nhblk1);
1656 }
```

_____unchanged_portion_omitted_

```
3092 static void
3093 do_prom_version_check(void)
3094 {
3095         int i;
3096         pnode_t node;
3097         char buf[64];
3098         static char drev[] = "Down-rev firmware detected%s\n"
3099             "\tPlease upgrade to the following minimum version:\n"
3100             "\t\t%s\n";

3102         i = prom_version_check(buf, sizeof (buf), &node);

3104         if (i == PROM_VER64_OK)
3105                 return;

3107         if (i == PROM_VER64_UPGRADE) {
3108                 cmn_err(CE_WARN, drev, "", buf);

3110 #ifdef  DEBUG
3111                 prom_enter_mon();       /* Type 'go' to continue */
3112                 cmn_err(CE_WARN, "Booting with down-rev firmware\n");
3113                 return;
3114 #else
3115                 halt(0);
```

```
#endif
        }

        /*
         * The other possibility is that this is a server running
         * good firmware, but down-rev firmware was detected on at
         * least one other cpu board. We just complain if we see
         * that.
         */
        cmn_err(CE_WARN, drev, " on one or more CPU boards", buf);
}


/*
 * Must be defined in platform dependent code.
 */
extern caddr_t modtext;
extern size_t modtext_sz;
extern caddr_t moddata;

#define HEAPTEXT_ARENA(addr)    \
        ((uintptr_t)(addr) < KERNELBASE + 2 * MMU_PAGESIZE4M ? 0 : \
        (((uintptr_t)(addr) - HEAPTEXT_BASE) / \
        (HEAPTEXT_MAPPED + HEAPTEXT_UNMAPPED) + 1))

#define HEAPTEXT_OVERSIZED(addr)        \
        ((uintptr_t)(addr) >= HEAPTEXT_BASE + HEAPTEXT_SIZE - HEAPTEXT_OVERSIZE)

#define HEAPTEXT_IN_NUCLEUSDATA(addr) \
        (((uintptr_t)(addr) >= KERNELBASE + 2 * MMU_PAGESIZE4M) && \
        ((uintptr_t)(addr) < KERNELBASE + 3 * MMU_PAGESIZE4M))

vmem_t *texthole_source[HEAPTEXT_NARENAS];
vmem_t *texthole_arena[HEAPTEXT_NARENAS];
kmutex_t texthole_lock;

char kern_bootargs[OBP_MAXPATHLEN];
char kern_bootfile[OBP_MAXPATHLEN];

void
kobj_vmem_init(vmem_t **text_arena, vmem_t **data_arena)
{
        uintptr_t addr, limit;

        addr = HEAPTEXT_BASE;
        limit = addr + HEAPTEXT_SIZE - HEAPTEXT_OVERSIZE;

        /*
         * Before we initialize the text_arena, we want to punch holes in the
         * underlying heaptext_arena.  This guarantees that for any text
         * address we can find a text hole less than HEAPTEXT_MAPPED away.
         */
        for (; addr + HEAPTEXT_UNMAPPED <= limit;
            addr += HEAPTEXT_MAPPED + HEAPTEXT_UNMAPPED) {
                (void) vmem_xalloc(heaptext_arena, HEAPTEXT_UNMAPPED, PAGESIZE,
                    0, 0, (void *)addr, (void *)(addr + HEAPTEXT_UNMAPPED),
                    VM_NOSLEEP | VM_BESTFIT | VM_PANIC);
        }

        /*
         * Allocate one page at the oversize to break up the text region
         * from the oversized region.
         */
        (void) vmem_xalloc(heaptext_arena, PAGESIZE, PAGESIZE, 0, 0,
            (void *)limit, (void *)(limit + PAGESIZE),
            VM_NOSLEEP | VM_BESTFIT | VM_PANIC);
```



```
        *text_arena = vmem_create("module_text", modtext_sz ? modtext : NULL,
            modtext_sz, sizeof (uintptr_t), segkmem_alloc, segkmem_free,
            heaptext_arena, 0, VM_SLEEP);
        *data_arena = vmem_create("module_data", moddata, MODDATA, 1,
            segkmem_alloc, segkmem_free, heap32_arena, 0, VM_SLEEP);
}
```

_____***unchanged_portion_omitted_***