```
**********************************************************
   96501 Thu Jan 24 09:58:58 2019
new/usr/src/cmd/picl/plugins/common/devtree/picldevtree.c
10135 picl plugins need smatch fixes
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Copyright (c) 2018, Joyent, Inc.
 */

/*
 * PICL plug-in that creates device tree nodes for all platforms
 */

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <limits.h>
#include <stdlib.h>
#include <assert.h>
#include <alloca.h>
#include <unistd.h>
#include <stropts.h>
#include <syslog.h>
#include <libdevinfo.h>
#include <sys/dkio.h>
#include <sys/vtoc.h>
#include <sys/time.h>
#include <fcntl.h>
#include <picl.h>
#include <picltree.h>
#include <sys/types.h>
#include <sys/processor.h>
#include <kstat.h>
#include <sys/sysinfo.h>
#include <dirent.h>
#include <libintl.h>
#include <pthread.h>
#include <libnvpair.h>
#include <sys/utsname.h>
#include <sys/systeminfo.h>
#include <sys/obpdefs.h>
```



```
#include <sys/openpromio.h>
#include "picldevtree.h"

/*
 * Plugin registration entry points
 */
static void     picldevtree_register(void);
static void     picldevtree_init(void);
static void     picldevtree_fini(void);

static void     picldevtree_evhandler(const char *ename, const void *earg,
                    size_t size, void *cookie);

#pragma init(picldevtree_register)

/*
 * Log message texts
 */
#define DEVINFO_PLUGIN_INIT_FAILED      gettext("SUNW_picldevtree failed!\n")
#define PICL_EVENT_DROPPED      \
        gettext("SUNW_picldevtree '%s' event dropped.\n")

/*
 * Macro to get PCI device id (from IEEE 1275 spec)
 */
#define PCI_DEVICE_ID(x)                        (((x) >> 11) & 0x1f)
/*
 * Local variables
 */
static picld_plugin_reg_t  my_reg_info = {
        PICLD_PLUGIN_VERSION_1,
        PICLD_PLUGIN_CRITICAL,
        "SUNW_picldevtree",
        picldevtree_init,
        picldevtree_fini
};
______unchanged_portion_omitted_

/*
 * Walk the snapshot and check the OBP properties of each node.
 */
static int
is_snapshot_stale(di_node_t root)
{
        snapshot_stale = 0;
        (void) di_walk_node(root, DI_WALK_CLDFIRST, NULL, check_stale_node);
        di_walk_node(root, DI_WALK_CLDFIRST, NULL, check_stale_node);
        return (snapshot_stale);
}
______unchanged_portion_omitted_

/*
 * This function reads the export file list from ASR
 */
static int
get_asr_export_list(char **exportlist, int *exportlistlen)
{
        struct openpromio oppbuf;
        struct openpromio *opp = &oppbuf;
        int d;
        int listsize;

        d = open("/dev/openprom", O_RDWR);
        if (d < 0)
                return (0);
```

```
2528         if (ioctl(d, OPROMEXPORTLEN, opp) == -1) {
2529                 (void) close(d);
2530                 return (0);
2531         }
2532         listsize = opp->oprom_size;
2533         opp = (struct openpromio *)malloc(sizeof (struct openpromio) +
2534             listsize);
2535         if (opp == NULL) {
2536                 (void) close(d);
2537                 return (0);
2538         }
2539         (void) memset(opp, '\0', sizeof (struct openpromio) + listsize);
2540         opp->oprom_size = listsize;
2541         if (ioctl(d, OPROMEXPORT, opp) == -1) {
2542                 free(opp);
2543                 (void) close(d);
2544                 return (0);
2545         }
2546         *exportlist = malloc(listsize);
2547         if (*exportlist == NULL) {
2548                 free(opp);
2549                 (void) close(d);
2550                 return (0);
2551         }
2552         (void) memcpy(*exportlist, opp->oprom_array, opp->oprom_size);
2553         *exportlistlen = opp->oprom_size;
2554         free(opp);
2550         *exportlistlen = opp->oprom_size;
2555         (void) close(d);
2556         return (1);
2557 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   23005 Thu Jan 24 09:58:59 2019
new/usr/src/cmd/picl/plugins/common/memcfg/piclmemcfg_comm.c
10135 picl plugins need smatch fixes
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright (c) 2018, Joyent, Inc.
 */

/*
 * This plugin creates memory configuration nodes and properties in the
 * PICL tree for Cheetah platforms.
 *
 * Subtree of memory-controller in the physical aspect.
 * memory-controller --- memory-module-group --- memory-module
 *
 * Subtree of memory in the logical aspect.
 * memory --- memory-segment --- memory-bank
 * Add property _memory-module-group_ at memory-segment referring to the
 * memory-module-group if InterleaveFactor is one, or at memory-bank
 * if InterleaveFactor is greater than one.
 *
 * Undo strategy:
 * Create all nodes and properties, or none if it fails in physical and
 * logical memory tree respectively. It keeps on creating logic
 * memory tree although it falis on physical logic tree, but no link to
 * memory module group.
 *
 * NOTE:
 * It depends on PICL devtree plugin and currently
 * there is no refresh routine for DR.
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <alloca.h>
#include <syslog.h>
#include <string.h>
#include <libintl.h>
#include <picl.h>
#include <picltree.h>
#include <fcntl.h>
```



```
#include <errno.h>
#include <sys/types.h>
#include <dirent.h>
#include <sys/stat.h>
#include <mc.h>
#include <libnvpair.h>
#include <limits.h>
#include "piclmemcfg.h"

/*
 * Plugin registration entry points
 */
static  void    piclmemcfg_register(void);
static  void    piclmemcfg_init(void);
static  void    piclmemcfg_fini(void);

/*
 * PICL event handler
 */
static void  piclmemcfg_evhandler(const char *ename, const void *earg,
                size_t size, void *cookie);

#pragma init(piclmemcfg_register)

static picld_plugin_reg_t  my_reg_info = {
        PICLD_PLUGIN_VERSION_1,
        PICLD_PLUGIN_NON_CRITICAL,
        "SUNW_piclmemcfg",
        piclmemcfg_init,
        piclmemcfg_fini
};
```

*unchanged_portion_omitted_*

```
/*
 * Create logical memory tree
 * memory --- memory-segment --- memory-bank
 * Get information via ioctl of memory control driver
 */
static int
create_logical_tree(picl_nodehdl_t memh, int fd)
{
        int                     i;
        int                     err = PICL_SUCCESS;
        picl_nodehdl_t          msegh;
        ptree_propinfo_t        propinfo;
        struct mc_memory        *mcmem;
        struct mc_segment       *mcseg;
        picl_prophdl_t          proph;
        uint64_t                memsize = 0;

        /*
         * allocate memory for mc_memory where nsegmentids are various
         */
        if ((mcmem = alloca((nsegments - 1) * sizeof (mcmem->segmentids[0]) +
            sizeof (*mcmem))) == NULL)
                return (PICL_FAILURE);

        mcmem->nsegments = nsegments;

        /*
         * Get logical memory information
         */
        if (ioctl(fd, MCIOC_MEM, mcmem) == -1)
                return (PICL_FAILURE);

        /*
```

```
397          * allocate memory for mc_segment where nbanks are various
398          */
399         if ((mcseg = alloca((nbanks - 1) * sizeof (mcseg->bankids[0]) +
400             sizeof (*mcseg))) == NULL)
401                 return (PICL_FAILURE);

403         /*
404          * Get all segments to create memory-segment nodes and
405          * add properties.
406          */
407         for (i = 0; i < nsegments; i++) {
408                 mcseg->id = mcmem->segmentids[i].globalid;
409                 mcseg->nbanks = nbanks;

411                 if (ioctl(fd, MCIOC_SEG, mcseg) == -1)
412                         break;

414                 /*
415                  * Create memory-segment node under memory node
416                  */
417                 err = ptree_create_and_add_node(memh, PICL_NAME_MEMORY_SEGMENT,
418                     PICL_CLASS_MEMORY_SEGMENT, &msegh);
419                 if (err != PICL_SUCCESS)
420                         break;

422                 msegh_info[i] = msegh;

424                 /*
425                  * Add property, Size to memory-segment node
426                  */
427                 err = ptree_init_propinfo(&propinfo, PTREE_PROPINFO_VERSION,
425                 if ((ptree_init_propinfo(&propinfo, PTREE_PROPINFO_VERSION,
428                     PICL_PTYPE_UNSIGNED_INT, PICL_READ, sizeof (mcseg->size),
429                     PICL_PROP_SIZE, NULL, NULL);
427                     PICL_PROP_SIZE, NULL, NULL)) != PICL_SUCCESS)
430                 if (err != PICL_SUCCESS)
431                         break;

433                 memsize += mcseg->size;
434                 err = ptree_create_and_add_prop(msegh, &propinfo, &mcseg->size,
435                     NULL);
436                 if (err != PICL_SUCCESS)
437                         break;

439                 /*
440                  * Add property, BaseAddress to memory-segment node
441                  */
442                 err = ptree_init_propinfo(&propinfo, PTREE_PROPINFO_VERSION,
443                     PICL_PTYPE_UNSIGNED_INT, PICL_READ, sizeof (mcseg->base),
444                     PICL_PROP_BASEADDRESS, NULL, NULL);
445                 if (err != PICL_SUCCESS)
446                         break;

448                 err = ptree_create_and_add_prop(msegh, &propinfo, &mcseg->base,
449                     NULL);
450                 if (err != PICL_SUCCESS)
451                         break;

453                 err = ptree_init_propinfo(&propinfo, PTREE_PROPINFO_VERSION,
454                     PICL_PTYPE_UNSIGNED_INT, PICL_READ, sizeof (mcseg->ifactor),
455                     PICL_PROP_INTERLEAVE_FACTOR, NULL, NULL);
456                 if (err != PICL_SUCCESS)
457                         break;

459                 err = ptree_create_and_add_prop(msegh, &propinfo,
460                     &mcseg->ifactor, NULL);
```



```
461                 if (err != PICL_SUCCESS)
462                         break;

464                 err = add_mem_banks(msegh, fd, mcseg);
465                 if (err != PICL_SUCCESS)
466                         break;
467         }

469         if (err != PICL_SUCCESS) {
470                 undo_logical_tree(nsegments);
471                 return (err);
472         }

474         err = ptree_get_prop_by_name(memh, PICL_PROP_SIZE, &proph);
475         if (err == PICL_SUCCESS) {      /* update the value */
476                 err = ptree_update_propval(proph, &memsize, sizeof (memsize));
477                 return (err);
478         }

480         /*
481          * Add the size property
482          */
483         (void) ptree_init_propinfo(&propinfo, PTREE_PROPINFO_VERSION,
484             PICL_PTYPE_UNSIGNED_INT, PICL_READ, sizeof (memsize),
485             PICL_PROP_SIZE, NULL, NULL);
486         err = ptree_create_and_add_prop(memh, &propinfo, &memsize, NULL);

488         return (err);
489 }
```

_____***unchanged_portion_omitted_***