```
**********************************************************
   33382 Tue Sep 25 07:16:30 2012
new/usr/src/lib/cfgadm_plugins/fp/common/cfga_utils.c
3217 cfgadm should spell "adaptors" correctly
Reviewed by: Alexander Eremin <alexander.r.eremin@gmail.com>
Reviewed by: David Hoeppner <0xffea@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
Reviewed by: Eric Schrock <Eric.Schrock@delphix.com>
**********************************************************
_____unchanged_portion_omitted_

  81 /*
  82  * The string table contains most of the strings used by the fp cfgadm plugin.
  83  * All strings which are to be internationalized must be in this table.
  84  * Some strings which are not internationalized are also included here.
  85  * Arguments to messages are NOT internationalized.
  86  */
  87 msgcvt_t str_tbl[] = {

  89 /*
  90  * The first element (ERR_UNKNOWN) MUST always be present in the array.
  91  */
  92 #define UNKNOWN_ERR_IDX         0       /* Keep the index in sync */


  95 /* msg_code     num_args, I18N  msg_string                              */

  97 /* ERRORS */
  98 {ERR_UNKNOWN,           0, 1,   "unknown error"},
  99 {ERR_OP_FAILED,         0, 1,   "operation failed"},
 100 {ERR_CMD_INVAL,         0, 1,   "invalid command"},
 101 {ERR_NOT_BUSAPID,       0, 1,   "not a FP bus apid"},
 102 {ERR_APID_INVAL,        0, 1,   "invalid FP ap_id"},
 103 {ERR_NOT_BUSOP,         0, 1,   "operation not supported for FC bus"},
 104 {ERR_NOT_DEVOP,         0, 1,   "operation not supported for FC device"},
 105 {ERR_UNAVAILABLE,       0, 1,   "unavailable"},
 106 {ERR_CTRLR_CRIT,        0, 1,   "critical partition controlled by FC HBA"},
 107 {ERR_BUS_GETSTATE,      0, 1,   "failed to get state for FC bus"},
 108 {ERR_BUS_NOTCONNECTED,  0, 1,   "FC bus not connected"},
 109 {ERR_BUS_CONNECTED,     0, 1,   "FC bus not disconnected"},
 110 {ERR_BUS_QUIESCE,       0, 1,   "FC bus quiesce failed"},
 111 {ERR_BUS_UNQUIESCE,     0, 1,   "FC bus unquiesce failed"},
 112 {ERR_BUS_CONFIGURE,     0, 1,   "failed to configure devices on FC bus"},
 113 {ERR_BUS_UNCONFIGURE,   0, 1,   "failed to unconfigure FC bus"},
 114 {ERR_DEV_CONFIGURE,     0, 1,   "failed to configure FC device"},
 115 {ERR_DEV_UNCONFIGURE,   0, 1,   "failed to unconfigure FC device"},
 116 {ERR_FCA_CONFIGURE,     0, 1,   "failed to configure ANY device on FCA port"},
 117 {ERR_FCA_UNCONFIGURE,   0, 1,   "failed to unconfigure ANY device on FCA port"},
 118 {ERR_DEV_REPLACE,       0, 1,   "replace operation failed"},
 119 {ERR_DEV_INSERT,        0, 1,   "insert operation failed"},
 120 {ERR_DEV_GETSTATE,      0, 1,   "failed to get state for FC device"},
 121 {ERR_RESET,             0, 1,   "reset failed"},
 122 {ERR_LIST,              0, 1,   "list operation failed"},
 123 {ERR_SIG_STATE,         0, 1,   "could not restore signal disposition"},
 124 {ERR_MAYBE_BUSY,        0, 1,   "device may be busy"},
 125 {ERR_BUS_DEV_MISMATCH,  0, 1,   "mismatched FC bus and device"},
 126 {ERR_MEM_ALLOC,         0, 1,   "Failed to allocated memory"},
 127 {ERR_DEVCTL_OFFLINE,    0, 1,   "failed to offline device"},
 128 {ERR_UPD_REP,           0, 1,   "Repository update failed"},
 129 {ERR_CONF_OK_UPD_REP,   0, 1,
 130                 "Configuration successful, but Repository update failed"},
 131 {ERR_UNCONF_OK_UPD_REP, 0, 1,
 132                 "Unconfiguration successful, but Repository update failed"},
 133 {ERR_PARTIAL_SUCCESS,   0, 1,
 134                 "Operation partially successful. Some failures seen"},
 135 {ERR_HBA_LOAD_LIBRARY,  0, 1,
```



```
 136                 "HBA load library failed"},
 137 {ERR_MATCHING_HBA_PORT, 0, 1,
 138                 "No match HBA port found"},
 139 {ERR_NO_ADAPTER_FOUND,  0, 1,
 140                 "No Fibre Channel adapters found"},
 140                 "No Fibre Channel adpaters found"},

 142 /* Errors with arguments */
 143 {ERRARG_OPT_INVAL,      1, 1,   "invalid option: "},
 144 {ERRARG_HWCMD_INVAL,    1, 1,   "invalid command: "},
 145 {ERRARG_DEVINFO,        1, 1,   "libdevinfo failed on path: "},
 146 {ERRARG_NOT_IN_DEVLIST, 1, 1,   "Device not found in fabric device list: "},
 147 {ERRARG_NOT_IN_DEVINFO, 1, 1,   "Could not find entry in devinfo tree: "},
 148 {ERRARG_DI_GET_PROP,    1, 1,   "Could not get libdevinfo property: "},
 149 {ERRARG_DC_DDEF_ALLOC,  1, 1,   "failed to alloc ddef space: "},
 150 {ERRARG_DC_BYTE_ARRAY,  1, 1,   "failed to add property: "},
 151 {ERRARG_DC_BUS_ACQUIRE, 1, 1,   "failed to acquire bus handle: "},
 152 {ERRARG_BUS_DEV_CREATE, 1, 1,   "failed to create device node: "},
 153 {ERRARG_BUS_DEV_CREATE_UNKNOWN, 1, 1,
 154         "failed to create device node... Device may be unconfigurable: "},
 155 {ERRARG_DEV_ACQUIRE,    1, 1,   "device acquire operation failed: "},
 156 {ERRARG_DEV_REMOVE,     1, 1,   "remove operation failed: "},

 158 /* Fibre Channel operation Errors */
 159 {ERR_FC,                0, 1,   "FC error"},
 160 {ERR_FC_GET_DEVLIST,    0, 1,   "Failed to get fabric device list"},
 161 {ERR_FC_GET_NEXT_DEV,   0, 1,   "Failed to get next device on device map"},
 162 {ERR_FC_GET_FIRST_DEV,  0, 1,   "Failed to get first device on device map"},
 163 {ERRARG_FC_DEV_MAP_INIT,        1, 1,
 164         "Failed to initialize device map for: "},
 165 {ERRARG_FC_PROP_LOOKUP_BYTES,   1, 1,   "Failed to get property of "},
 166 {ERRARG_FC_INQUIRY,     1, 1,   "inquiry failed: "},
 167 {ERRARG_FC_REP_LUNS,    1, 1,   "report LUNs failed: "},
 168 {ERRARG_FC_TOPOLOGY,    1, 1,   "failed to get port topology: "},
 169 {ERRARG_PATH_TOO_LONG,  1, 1,   "Path length exceeds max possible: "},
 170 {ERRARG_INVALID_PATH,   1, 1,   "Invalid path: "},
 171 {ERRARG_OPENDIR,        1, 1,   "failure opening directory: "},

 173 /* MPXIO Errors */
 174 {ERRARG_VHCI_GET_PATHLIST,      1, 1,   "failed to get path list from vHCI: "},
 175 {ERRARG_XPORT_NOT_IN_PHCI_LIST, 1, 1,   "Transport not in pHCI list: "},

 177 /* RCM Errors */
 178 {ERR_RCM_HANDLE,        0, 1,   "cannot get RCM handle"},
 179 {ERRARG_RCM_SUSPEND,    1, 1,   "failed to suspend: "},
 180 {ERRARG_RCM_RESUME,     1, 1,   "failed to resume: "},
 181 {ERRARG_RCM_OFFLINE,    1, 1,   "failed to offline: "},
 182 {ERRARG_RCM_ONLINE,     1, 1,   "failed to online: "},
 183 {ERRARG_RCM_REMOVE,     1, 1,   "failed to remove: "},
 184 {ERRARG_RCM_INFO,       1, 1,   "failed to query: "},

 186 /* Commands */
 187 {CMD_INSERT_DEV,        0, 0,   "insert_device"},
 188 {CMD_REMOVE_DEV,        0, 0,   "remove_device"},
 189 {CMD_REPLACE_DEV,       0, 0,   "replace_device"},
 190 {CMD_RESET_DEV,         0, 0,   "reset_device"},
 191 {CMD_RESET_BUS,         0, 0,   "reset_bus"},
 192 {CMD_RESET_ALL,         0, 0,   "reset_all"},

 194 /* help messages */
 195 {MSG_HELP_HDR,          0, 1,   "\nfp attachment point specific options:\n"},
 196 {MSG_HELP_USAGE,        0, 0,
 197                 "\t-c configure -o force_update ap_id [ap_id..]\n"
 198                 "\t-c configure -o no_update ap_id [ap_id...]\n"
 199                 "\t-c unconfigure -o force_update ap_id [ap_id... ]\n"
 200                 "\t-c unconfigure -o no_update ap_id [ap_id... ]\n"},
```

```
 202 /* hotplug messages */
 203 {MSG_INSDEV,            1, 1,   "Adding device to FC HBA: "},
 204 {MSG_RMDEV,             1, 1,   "Removing FC device: "},
 205 {MSG_REPLDEV,           1, 1,   "Replacing FC device: "},

 207 /* Hotplugging confirmation prompts */
 208 {CONF_QUIESCE_1,        1, 1,
 209         "This operation will suspend activity on FC bus: "},

 211 {CONF_QUIESCE_2,        0, 1,   "\nContinue"},

 213 {CONF_UNQUIESCE,        0, 1,
 214         "FC bus quiesced successfully.\n"
 215         "It is now safe to proceed with hotplug operation."
 216         "\nEnter y if operation is complete or n to abort"},

 218 /* Misc. */
 219 {WARN_DISCONNECT,       0, 1,
 220         "WARNING: Disconnecting critical partitions may cause system hang."
 221         "\nContinue"}
 222 };
______unchanged_portion_omitted_

1234 /*
1235  * Find the Adapter port that matches the portPath.
1236  * When the matching port is found the caller have to close handle
1237  * and free library.
1238  */
1239 fpcfga_ret_t
1240 findMatchingAdapterPort(char *portPath, HBA_HANDLE *matchingHandle,
1241         int *matchingPortIndex, HBA_PORTATTRIBUTES *matchingPortAttrs,
1242         char **errstring)
1243 {
1244         HBA_HANDLE      handle;
1245         HBA_ADAPTERATTRIBUTES   hbaAttrs;
1246         HBA_PORTATTRIBUTES      portAttrs;
1247         HBA_STATUS status = HBA_STATUS_OK;
1248         int count, retry = 0, l_errno = 0;
1249         int adapterIndex, portIndex;
1250         char                    adapterName[256];
1251         char                    *cfg_ptr, *tmpPtr;
1252         char                    *logical_apid = NULL;

1254         status = HBA_LoadLibrary();
1255         if (status != HBA_STATUS_OK) {
1256                 cfga_err(errstring, 0, ERR_HBA_LOAD_LIBRARY, 0);
1257                 return (FPCFGA_LIB_ERR);
1258         }
1259         count = HBA_GetNumberOfAdapters();
1260         if (count == 0) {
1261                 cfga_err(errstring, 0, ERR_NO_ADAPTER_FOUND, 0);
1262                 HBA_FreeLibrary();
1263                 return (FPCFGA_LIB_ERR);
1264         }

1266         /* Loop over all HBAs */
1267         for (adapterIndex = 0; adapterIndex < count; adapterIndex ++) {
1268                 status = HBA_GetAdapterName(adapterIndex, (char *)&adapterName);
1269                 if (status != HBA_STATUS_OK) {
1270                         /* May have been DR'd */
1271                         continue;
1272                 }
1273                 handle = HBA_OpenAdapter(adapterName);
1274                 if (handle == 0) {
1275                         /* May have been DR'd */
```



```
1276                         continue;
1277                 }

1279                 do {
1280                         if (getAdapterAttrs(handle, &hbaAttrs)) {
1281                                 /* Should never happen */
1282                                 HBA_CloseAdapter(handle);
1283                                 continue;
1284                         }

1286                         /* Loop over all HBA Ports */
1287                         for (portIndex = 0;
1288                             portIndex < hbaAttrs.NumberOfPorts; portIndex++) {
1289                                 if ((status = getAdapterPortAttrs(handle,
1290                                     portIndex,
1289                             if ((status = getAdapterPortAttrs(handle, portIndex,
1291                                     &portAttrs)) != HBA_STATUS_OK) {
1292                                         /* Need to refresh adapter */
1293                                         if (status ==
1294                                             HBA_STATUS_ERROR_STALE_DATA) {
1292                                 if (status == HBA_STATUS_ERROR_STALE_DATA) {
1295                                                 HBA_RefreshInformation(handle);
1296                                                 break;
1297                                         } else {
1298                                                 continue;
1299                                         }
1300                                 }

1302                                 /*
1303                                  * check to see if OSDeviceName is a /dev/cfg
1304                                  * link or the physical path
1301                                  * check to see if OSDeviceName is a /dev/cfg link
1302                                  * or the physical path
1305                                  */
1306                                 if (strncmp(portAttrs.OSDeviceName,
1307                                     CFGA_DEV_DIR,
1304                             if (strncmp(portAttrs.OSDeviceName, CFGA_DEV_DIR,
1308                                         strlen(CFGA_DEV_DIR)) != 0) {
1309                                             tmpPtr = strstr(portAttrs.OSDeviceName,
1310                                                 MINOR_SEP);
1311                                             if ((tmpPtr != NULL) &&
1312                                                 strncmp(portPath,
1306                             tmpPtr = strstr(portAttrs.OSDeviceName, MINOR_SEP);
1307                             if (tmpPtr != NULL) {
1308                                 if (strncmp(portPath,
1313                                         portAttrs.OSDeviceName,
1314                                         strlen(portAttrs.OSDeviceName) -
1315                                         strlen(tmpPtr)) == 0) {
1316                                             if (matchingHandle)
1317                                                     *matchingHandle =
1318                                                         handle;
1313                                             *matchingHandle = handle;
1319                                             if (matchingPortIndex)
1320                                                     *matchingPortIndex =
1321                                                         portIndex;
1315                                             *matchingPortIndex = portIndex;
1322                                             if (matchingPortAttrs)
1323                                                     *matchingPortAttrs =
1324                                                         portAttrs;
1317                                             *matchingPortAttrs = portAttrs;
1325                                             return (FPCFGA_OK);
1326                                         }
1320                             }
1327                                 } else {
1328                                         /*
1329                                          * strip off the /dev/cfg/ portion of
```

```
                                        * the OSDeviceName make sure that the
                                        * OSDeviceName is at least
                        * strip off the /dev/cfg/ portion of the
                        * OSDeviceName
                        * make sure that the OSDeviceName is at least
                                        * strlen("/dev/cfg") + 1 + 1 long.
                                        * first 1 is for the / after /dev/cfg
                                        * second 1 is to make sure there is
                                        * somthing after
                        *       second 1 is to make sure there is somthing
                        *       after
                                        */
                                       if (strlen(portAttrs.OSDeviceName) <
                                           (strlen(CFGA_DEV_DIR) + 1 + 1))
                                               continue;
                                       cfg_ptr = portAttrs.OSDeviceName +
                                           strlen(CFGA_DEV_DIR) + 1;
                                       if (logical_apid == NULL) {
                                               /*
                                                * get the /dev/cfg link from
                                                * the portPath
                                                */
                                               if (make_xport_logid(portPath,
                                                    &logical_apid,
                        /* get the /dev/cfg link from the portPath */
                        if (make_xport_logid(portPath, &logical_apid,
                                                    &l_errno) != FPCFGA_OK) {
                                                    cfga_err(errstring,
                                                        l_errno,
                                       cfga_err(errstring, l_errno,
                                                        ERR_LIST, 0);
                                                    HBA_FreeLibrary();
                                                    return
                                                        (FPCFGA_LIB_ERR);
                                       return (FPCFGA_LIB_ERR);
                                               }
                                       }
                                       /* compare logical ap_id */
                                       if (strcmp(logical_apid,
                                           cfg_ptr) == 0) {
                    if (strcmp(logical_apid, cfg_ptr) == 0) {
                                               if (matchingHandle)
                                                       *matchingHandle =
                                                           handle;
                                       *matchingHandle = handle;
                                               if (matchingPortIndex)
                                                       *matchingPortIndex =
                                                           portIndex;
                                       *matchingPortIndex = portIndex;
                                               if (matchingPortAttrs)
                                                       *matchingPortAttrs =
                                                           portAttrs;
                                       *matchingPortAttrs = portAttrs;
                                               S_FREE(logical_apid);
                                               return (FPCFGA_OK);
                                       }
                               }
                       }
                       if (logical_apid != NULL)
                               S_FREE(logical_apid);
               } while ((status == HBA_STATUS_ERROR_STALE_DATA) &&
                   (retry++ < HBA_MAX_RETRIES));

               HBA_CloseAdapter(handle);
       }
       free(logical_apid);
```



```
                /* Got here. No matching adapter port found. */
                /* Got here. No mathcing adatper port found. */
                cfga_err(errstring, 0, ERR_MATCHING_HBA_PORT, 0);
                HBA_FreeLibrary();
                return (FPCFGA_LIB_ERR);
}
______unchanged_portion_omitted_
```

```
**********************************************************
  137421 Tue Sep 25 07:16:30 2012
new/usr/src/lib/smhba/common/SMHBAAPILIB.c
3217 cfgadm should spell "adaptors" correctly
Reviewed by: Alexander Eremin <alexander.r.eremin@gmail.com>
Reviewed by: David Hoeppner <0xffea@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
Reviewed by: Eric Schrock <Eric.Schrock@delphix.com>
**********************************************************
_______unchanged_portion_omitted_

1397 HBA_STATUS
1398 HBA_GetAdapterName(
1399     HBA_UINT32 adapterindex,
1400     char *adaptername)
1401 {
1402         HBA_ADAPTER_INFO        *adapt_infop;
1403         HBA_STATUS              ret = HBA_STATUS_ERROR_ILLEGAL_INDEX;

1405         if (adaptername == NULL) {
1406                 DEBUG(1, "HBA_GetAdapterName: NULL pointer adaptername",
1406                 DEBUG(1, "HBA_GetAdapterName: NULL pointer adatpername",
1407                     0, 0, 0);
1408                 return (HBA_STATUS_ERROR_ARG);
1409         }

1411         /*
1412          * The adapter index is from old code, but we have
1413          * to support it.  Go down the list looking for
1414          * the adapter
1415          */
1416         ARE_WE_INITED();
1417         GRAB_MUTEX(&_hbaapi_AL_mutex);
1418         *adaptername = '\0';
1419         for (adapt_infop = _hbaapi_adapterlist;
1420             adapt_infop != NULL;
1421             adapt_infop = adapt_infop->next) {

1423         if (adapt_infop->index == adapterindex) {
1424                 if (adapt_infop->name != NULL &&
1425                     adapt_infop->GNstatus == HBA_STATUS_OK) {
1426                 (void) strcpy(adaptername, adapt_infop->name);
1427                 } else {
1428                 *adaptername = '\0';
1429                 }
1430                 ret = adapt_infop->GNstatus;
1431                 break;
1432         }
1433         }
1434         DEBUG(2, "GetAdapterName for index:%d ->%s",
1435             adapterindex, adaptername, 0);
1436         RELEASE_MUTEX_RETURN(&_hbaapi_AL_mutex, ret);
1437 }

1439 HBA_HANDLE
1440 HBA_OpenAdapter(char *adaptername)
1441 {
1442         HBA_HANDLE              handle;
1443         HBAOpenAdapterFunc      OpenAdapterFunc;
1444         HBA_ADAPTER_INFO        *adapt_infop;
1445         HBA_LIBRARY_INFO        *lib_infop;

1447         DEBUG(2, "OpenAdapter: %s", adaptername, 0, 0);

1449         handle = HBA_HANDLE_INVALID;
1450         if (_hbaapi_librarylist == NULL) {
```



```
1451                 return (handle);
1452         }
1453         if (adaptername == NULL) {
1454                 DEBUG(1, "HBA_OpenAdapter: NULL pointer adaptername",
1454                 DEBUG(1, "HBA_OpenAdapter: NULL pointer adatpername",
1455                     0, 0, 0);
1456                 return (handle);
1457         }
1458         GRAB_MUTEX(&_hbaapi_AL_mutex);
1459         for (adapt_infop = _hbaapi_adapterlist;
1460             adapt_infop != NULL;
1461             adapt_infop = adapt_infop->next) {
1462         if (strcmp(adaptername, adapt_infop->name) != 0) {
1463                 continue;
1464         }
1465         lib_infop = adapt_infop->library;
1466         OpenAdapterFunc = FUNCCOMMON(lib_infop, OpenAdapterHandler);

1468         if (OpenAdapterFunc != NULL) {
1469             /* retrieve the vendor handle */
1470                 handle = (OpenAdapterFunc)(adaptername);
1471                 if (handle != 0) {
1472                 /* or this with the library index to get the common handle */
1473                 handle = HBA_HANDLE_FROM_LOCAL(lib_infop->index, handle);
1474                 }
1475         }
1476         break;
1477         }
1478         RELEASE_MUTEX_RETURN(&_hbaapi_AL_mutex, handle);
1479 }
_______unchanged_portion_omitted_

4328 /*
4329  * Following the similar logic of HBAAPI addaspterevents_callback.
4330  *
4331  * Unlike other events Adapter Add Event is not limited to a specific
4332  * adapter(i.e. no adapter handle is passed for registration) so
4332  * adpater(i.e. no adatper handle is passed for registration) so
4333  * the event should be passed to all registrants.  The routine below
4334  * is passed to the VSLs as a callback and when Adapter Add event is detected
4335  * by VSL it will call smhba_adapteraddevents_callback() which in turn check
4336  * if the passed userdata ptr matches with the one stored in the callback list
4337  * and calls the stored callback.
4338  *
4339  * For the situation that multiple clients are registered for Adapter Add event
4340  * each registration is passed to VSLs so VSL may call
4341  * smhba_adapteraddevents_callback() multiple times or it may call only once
4342  * since the callback function is same.  For this implemneation, the userdata
4343  * is stored in HBA_ALLADAPTERSCALLBACK_ELEM so it is expected that VSL call
4344  * smhba_adapteraddevents_callback() only once and
4345  * smhba_adapteraddevents_callback() will call the client callback with proper
4346  * userdata.
4347  */
4348 static void
4349 smhba_adapteraddevents_callback(
4350 /* LINTED E_FUNC_ARG_UNUSED */
4351     void *data,
4352     HBA_WWN PortWWN,
4353 /* LINTED E_FUNC_ARG_UNUSED */
4354     HBA_UINT32 eventType)
4355 {
4356         HBA_ALLADAPTERSCALLBACK_ELEM    *cbp;

4358         DEBUG(3, "AddAdapterEvent, port:%s", WWN2STR1(&PortWWN), 0, 0);

4360         GRAB_MUTEX(&_smhba_AAE_mutex);
```

```
4361         for (cbp = _smhba_adapteraddevents_callback_list;
4362             cbp != NULL;
4363             cbp = cbp->next) {
4364         (*cbp->callback)(cbp->userdata, PortWWN, HBA_EVENT_ADAPTER_ADD);
4365         }
4366         RELEASE_MUTEX(&_smhba_AAE_mutex);

4368 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   12585 Tue Sep 25 07:16:31 2012
new/usr/src/lib/sun_fc/common/FCHBA.cc
3217 cfgadm should spell "adaptors" correctly
Reviewed by: Alexander Eremin <alexander.r.eremin@gmail.com>
Reviewed by: David Hoeppner <0xffea@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
Reviewed by: Eric Schrock <Eric.Schrock@delphix.com>
**********************************************************
_____unchanged_portion_omitted_

 322 void FCHBA::loadAdapters(vector<HBA*> &list) {
 323     Trace log("FCHBA::loadAdapters");
 324     fcio_t                      fcio;
 325     fc_hba_list_t               *pathList;
 326     int                 fd;
 327     int                 size = 64; // default first attempt
 328     bool                retry = false;
 329     struct stat         sb;
 330     int bufSize;

 332     /* Before we do anything, let's see if FCSM is on the system */
 333     errno = 0;
 334     if (stat(FCSM_DRIVER_PATH.c_str(), &sb) != 0) {
 335         if (errno == ENOENT) {
 336             log.genericIOError(
 337                 "The %s driver is not present. Unable to issue "
 338                 "CT commands. Please install the %s package.",
 339                 FCSM_DRIVER_PATH.c_str(), FCSM_DRIVER_PKG.c_str());
 340             throw NotSupportedException();
 341         } else {
 342             log.genericIOError(
 343                 "Can not stat the %s driver for reason \"%s\" "
 344                 "Unable to issue CT commands.",
 345                 FCSM_DRIVER_PATH.c_str(), strerror(errno));
 346             throw IOError("Unable to stat FCSM driver");
 347         }
 348     }


 351     /* construct fcio struct */
 352     memset(&fcio, 0, sizeof (fcio_t));
 353     fcio.fcio_cmd       = FCSMIO_ADAPTER_LIST;
 354     fcio.fcio_xfer      = FCIO_XFER_RW;


 357     /* open the fcsm node so we can send the ioctl to */
 358     errno = 0;
 359     if ((fd = open(FCSM_DRIVER_PATH.c_str(), O_RDONLY)) < 0) {
 360         if (errno == EBUSY) {
 361             throw BusyException();
 362         } else if (errno == EAGAIN) {
 363             throw TryAgainException();
 364         } else if (errno == ENOTSUP) {
 365             throw NotSupportedException();
 366         } else if (errno == ENOENT) {
 367             throw UnavailableException();
 368         } else {
 369             throw IOError("Unable to open FCSM driver");
 370         }
 371     }

 373     do {
 374         retry = false;
 375         errno = 0;
 376         bufSize = MAXPATHLEN * size + (int) sizeof (fc_hba_list_t) - 1;
```



```
 377         pathList = (fc_hba_list_t *)new uchar_t[bufSize];
 378         pathList->numAdapters = size;
 379         fcio.fcio_olen  = bufSize;
 380         fcio.fcio_obuf  = (char *)pathList;
 381         if (ioctl(fd, FCSMIO_CMD, &fcio) != 0) {
 382             /* Interpret the fcio error code */
 383             char fcioErrorString[MAX_FCIO_MSG_LEN] = "";

 385             log.genericIOError(
 386                 "ADAPTER_LIST failed: "
 387                 "Errno: \"%s\"",
 388                 strerror(errno));
 389             delete (pathList);
 390             close(fd);
 391             if (errno == EBUSY) {
 392                 throw BusyException();
 393             } else if (errno == EAGAIN) {
 394                 throw TryAgainException();
 395             } else if (errno == ENOTSUP) {
 396                 throw NotSupportedException();
 397             } else if (errno == ENOENT) {
 398                 throw UnavailableException();
 399             } else {
 400                 throw IOError("Unable to build HBA list");
 401             }
 402         }
 403         if (pathList->numAdapters > size) {
 404             log.debug(
 405                 "Buffer too small for number of HBAs. Retrying.");
 406             size = pathList->numAdapters;
 407             retry = true;
 408             delete (pathList);
 409         }
 410     } while (retry);

 412     close(fd);
 413     log.debug("Detected %d adapters", pathList->numAdapters);
 414     for (int i = 0, times =0; i < pathList->numAdapters;) {
 415         try {
 416             HBA *hba = new FCHBA(pathList->hbaPaths[i]);
 417             list.insert(list.begin(), hba);
 418             i++;
 419         } catch (BusyException &e) {
 420             sleep(1);
 421             if (times++ > EXCPT_RETRY_COUNT) {
 422                 i++;
 423                 times = 0;
 424             }
 425             continue;
 426         } catch (TryAgainException &e) {
 427             sleep(1);
 428             if (times++ > EXCPT_RETRY_COUNT) {
 429                 i++;
 430                 times = 0;
 431             }
 432             continue;
 433         } catch (UnavailableException &e) {
 434             sleep(1);
 435             if (times++ > EXCPT_RETRY_COUNT) {
 436                 i++;
 437                 times = 0;
 438             }
 439             continue;
 440         } catch (HBAException &e) {
 441             i++;
 442             times = 0;
```

```
443             log.debug(
444                 "Ignoring partial failure while loading an HBA");
445         }
446     }
447     if (pathList->numAdapters > HBAList::HBA_MAX_PER_LIST) {
448         delete(pathList);
449         throw InternalError(
450             "Exceeds max number of adapters that VSL supports.");
450             "Exceeds max number of adatpers that VSL supports.");
451     }
452     delete (pathList);
453 }
```

unchanged_portion_omitted_

```
**********************************************************
    9681 Tue Sep 25 07:16:32 2012
new/usr/src/lib/sun_fc/common/TgtFCHBA.cc
3217 cfgadm should spell "adaptors" correctly
Reviewed by: Alexander Eremin <alexander.r.eremin@gmail.com>
Reviewed by: David Hoeppner <0xffea@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
Reviewed by: Eric Schrock <Eric.Schrock@delphix.com>
**********************************************************
______unchanged_portion_omitted_

221 void TgtFCHBA::loadAdapters(vector<HBA*> &list)
222 {
223     Trace log("TgtFCHBA::loadAdapters");
224     fctio_t                 fctio;
225     fc_tgt_hba_list_t       *tgthbaList;
226     int                 fd;
227     int                 size = 64; // default first attempt
228     bool                retry = false;
229     struct stat         sb;
230     int bufSize;
231     char wwnStr[17];

233     /* Before we do anything, let's see if FCT is on the system */
234     errno = 0;
235     if (stat(FCT_DRIVER_PATH.c_str(), &sb) != 0) {
236         if (errno == ENOENT) {
237             log.genericIOError(
238                 "The %s driver is not present."
239                 " Please install the %s package.",
240                 FCT_DRIVER_PATH.c_str(), FCT_DRIVER_PKG.c_str());
241             throw NotSupportedException();
242         } else {
243             log.genericIOError(
244                 "Can not stat the %s driver for reason \"%s\" "
245                 "Unable to get target mode FC adapters.",
246                 FCT_DRIVER_PATH.c_str(), strerror(errno));
247             throw IOError("Unable to stat FCSM driver");
248         }
249     }


252     /* construct fcio struct */
253     memset(&fctio, 0, sizeof (fctio_t));
254     fctio.fctio_cmd     = FCTIO_ADAPTER_LIST;
255     fctio.fctio_xfer    = FCTIO_XFER_RW;

257     /* open the fcsm node so we can send the ioctl to */
258     errno = 0;
259     if ((fd = open(FCT_DRIVER_PATH.c_str(), O_RDONLY)) < 0) {
260         if (errno == EBUSY) {
261             throw BusyException();
262         } else if (errno == EAGAIN) {
263             throw TryAgainException();
264         } else if (errno == ENOTSUP) {
265             throw NotSupportedException();
266         } else if (errno == ENOENT) {
267             throw UnavailableException();
268         } else {
269             throw IOError("Unable to open FCT driver");
270         }
271     }

273     do {
274         retry = false;
275         errno = 0;
```



```
276         bufSize = 8 * (size - 1) + (int) sizeof (fc_tgt_hba_list_t);
277         tgthbaList = (fc_tgt_hba_list_t *)new uchar_t[bufSize];
278         tgthbaList->numPorts = size;
279         fctio.fctio_olen        = bufSize;
280         fctio.fctio_obuf        = (uint64_t)(uintptr_t)tgthbaList;
281         if (ioctl(fd, FCTIO_CMD, &fctio) != 0) {
282             /* Interpret the fcio error code */
283             char fcioErrorString[MAX_FCTIO_MSG_LEN] = "";

285             log.genericIOError(
286                 "TGT_ADAPTER_LIST failed: "
287                 "Errno: \"%s\"",
288                 strerror(errno));
289             delete (tgthbaList);
290             close(fd);
291             if (errno == EBUSY) {
292                 throw BusyException();
293             } else if (errno == EAGAIN) {
294                 throw TryAgainException();
295             } else if (errno == ENOTSUP) {
296                 throw NotSupportedException();
297             } else if (errno == ENOENT) {
298                 throw UnavailableException();
299             } else {
300                 throw IOError("Unable to build HBA list");
301             }
302         }
303         if (tgthbaList->numPorts > size) {
304             log.debug(
305                 "Buffer too small for number of target mode HBAs. Retrying.");
306             size = tgthbaList->numPorts;
307             retry = true;
308             delete (tgthbaList);
309         }
310     } while (retry);

312     close(fd);
313     log.debug("Detected %d target mode adapters", tgthbaList->numPorts);
314     for (int i = 0; i < tgthbaList->numPorts; i++) {
315         try {
316             std::string hbapath = FCT_ADAPTER_NAME_PREFIX.c_str();
317             hbapath += ".";
318             // move the row with two dimentional uint8 array for WWN
319             uint64_t tmp = ntohll(*((uint64_t *)&tgthbaList->port_wwn[i][0]));
320             sprintf(wwnStr, "%llx", tmp);
321             hbapath += wwnStr;

323             HBA *hba = new TgtFCHBA(hbapath);
324             list.insert(list.begin(), hba);
325         } catch (...) {
326             log.debug(
327                 "Ignoring partial failure while loading an HBA");
328         }
329     }
330     if (tgthbaList->numPorts > HBAList::HBA_MAX_PER_LIST) {
331         delete(tgthbaList);
332         throw InternalError(
333             "Exceeds max number of adapters that VSL supports.");
333             "Exceeds max number of adatpers that VSL supports.");
334     }
335     delete (tgthbaList);
336 }
______unchanged_portion_omitted_
```

```
**********************************************************
   10409 Tue Sep 25 07:16:32 2012
new/usr/src/lib/sun_sas/common/sun_sas.c
3217 cfgadm should spell "adaptors" correctly
Reviewed by: Alexander Eremin <alexander.r.eremin@gmail.com>
Reviewed by: David Hoeppner <0xffea@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
Reviewed by: Eric Schrock <Eric.Schrock@delphix.com>
**********************************************************
______unchanged_portion_omitted_
 123 /*
 124  * Given a handle, returns the open_handle structure
 125  * The routine assumes that the open_handles_lock has already
 126  * been taken.
 127  */
 128 struct open_handle *
 129 RetrieveOpenHandle(HBA_HANDLE handle)
 130 {

 132         const char              ROUTINE[] = "RetrieveOpenHandle";
 133         struct open_handle      *open_handle_ptr = NULL;

 135         if (global_hba_head == NULL) {
 136                 log(LOG_DEBUG, ROUTINE, "No adapter is found.");
 136                 log(LOG_DEBUG, ROUTINE, "No adpater is found.");
 137                 return (NULL);
 138         }

 140         for (open_handle_ptr = global_hba_head->open_handles;
 141             open_handle_ptr != NULL;
 142             open_handle_ptr = open_handle_ptr->next) {
 143                 if (open_handle_ptr->handle == handle) {
 144                         break;
 145                 }
 146         }

 148         return (open_handle_ptr);
 149 }
______unchanged_portion_omitted_
```

```
**********************************************************
    3179 Tue Sep 25 07:16:33 2012
new/usr/src/uts/common/io/1394/adapters/hci1394_extern.c
3217 cfgadm should spell "adaptors" correctly
Reviewed by: Alexander Eremin <alexander.r.eremin@gmail.com>
Reviewed by: David Hoeppner <0xffea@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
Reviewed by: Eric Schrock <Eric.Schrock@delphix.com>
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 1999-2000 by Sun Microsystems, Inc.
 * All rights reserved.
 */

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

/*
 * hci1394_extern.c
 *    Central location for externs.  There are two exceptions to this,
 *    hci1394_statep (located in hci1394.c) and hci1394_evts (located in
 *    hci1394_s1394if.c).
 */

#include <sys/conf.h>
#include <sys/ddi.h>
#include <sys/modctl.h>
#include <sys/stat.h>

#include <sys/1394/h1394.h>

#include <sys/1394/adapters/hci1394.h>



/*
 * The 1394 bus ticks are in 125uS increments.  split_timeout is represented in
 * 1394 bus ticks.  800 bus ticks is 100mS.
 */
uint32_t hci1394_split_timeout = 800;


/*
 * 1394 address map for OpenHCI adapters.
 * 1394 address map for OpenHCI adpaters.
 *
```



```
 * This is what is reported to the services layer.  The hci1394 driver does not
 * modify the HW to reflect this.  This should reflect what the OpenHCI 1.0 HW
 * is set to.  The comments below give the actual address ranges where the
 * actual structure has the format of - start address, size, type.
 *
 * physical =>          0x0000000000000000 - 0x00000000FFFFFFFF
 * posted write =>      0x0000000100000000 - 0x0000FFFEFFFFFFFF
 * normal =>            0x0000FFFF00000000 - 0x0000FFFFEFFFFFFF
 * csr =>               0x0000FFFFF0000000 - 0x0000FFFFFFFFFFFF
 */
h1394_addr_map_t hci1394_addr_map[HCI1394_ADDR_MAP_SIZE] = {
        {0x0000000000000000, 0x0000000100000000, H1394_ADDR_PHYSICAL},
        {0x0000000100000000, 0x0000FFFE00000000, H1394_ADDR_POSTED_WRITE},
        {0x0000FFFF00000000, 0x00000000F0000000, H1394_ADDR_NORMAL},
        {0x0000FFFFF0000000, 0x0000000010000000, H1394_ADDR_CSR}
};
______unchanged_portion_omitted_
```

```
**********************************************************
   30071 Tue Sep 25 07:16:34 2012
new/usr/src/uts/common/io/igb/igb_regs.h
3217 cfgadm should spell "adaptors" correctly
Reviewed by: Alexander Eremin <alexander.r.eremin@gmail.com>
Reviewed by: David Hoeppner <0xffea@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
Reviewed by: Eric Schrock <Eric.Schrock@delphix.com>
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright (c) 2007-2012 Intel Corporation. All rights reserved.
 */

/*
 * Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
 */

/* IntelVersion: 1.82.2.1 v3_3_14_3_BHSW1 */

#ifndef _IGB_REGS_H
#define _IGB_REGS_H

#ifdef __cplusplus
extern "C" {
#endif

#define E1000_CTRL      0x00000  /* Device Control - RW */
#define E1000_CTRL_DUP  0x00004  /* Device Control Duplicate (Shadow) - RW */
#define E1000_STATUS    0x00008  /* Device Status - RO */
#define E1000_EECD      0x00010  /* EEPROM/Flash Control - RW */
#define E1000_EERD      0x00014  /* EEPROM Read - RW */
#define E1000_CTRL_EXT  0x00018  /* Extended Device Control - RW */
#define E1000_FLA       0x0001C  /* Flash Access - RW */
#define E1000_MDIC      0x00020  /* MDI Control - RW */
#define E1000_MDICNFG   0x00E04  /* MDI Config - RW */
#define E1000_REGISTER_SET_SIZE         0x20000 /* CSR Size */
#define E1000_EEPROM_INIT_CTRL_WORD_2   0x0F /* EEPROM Init Ctrl Word 2 */
#define E1000_BARCTRL                   0x5BBC /* BAR ctrl reg */
#define E1000_BARCTRL_FLSIZE            0x0700 /* BAR ctrl Flsize */
#define E1000_BARCTRL_CSRSIZE           0x2000 /* BAR ctrl CSR size */
#define E1000_SCTL      0x00024  /* SerDes Control - RW */
#define E1000_FCAL      0x00028  /* Flow Control Address Low - RW */
#define E1000_FCAH      0x0002C  /* Flow Control Address High -RW */
#define E1000_FEXT      0x0002C  /* Future Extended - RW */
#define E1000_FEXTNVM   0x00028  /* Future Extended NVM - RW */
```



```
#define E1000_FCT       0x00030  /* Flow Control Type - RW */
#define E1000_CONNSW    0x00034  /* Copper/Fiber switch control - RW */
#define E1000_VET       0x00038  /* VLAN Ether Type - RW */
#define E1000_ICR       0x000C0  /* Interrupt Cause Read - R/clr */
#define E1000_ITR       0x000C4  /* Interrupt Throttling Rate - RW */
#define E1000_ICS       0x000C8  /* Interrupt Cause Set - WO */
#define E1000_IMS       0x000D0  /* Interrupt Mask Set - RW */
#define E1000_IMC       0x000D8  /* Interrupt Mask Clear - WO */
#define E1000_IAM       0x000E0  /* Interrupt Acknowledge Auto Mask */
#define E1000_RCTL      0x00100  /* Rx Control - RW */
#define E1000_FCTTV     0x00170  /* Flow Control Transmit Timer Value - RW */
#define E1000_TXCW      0x00178  /* Tx Configuration Word - RW */
#define E1000_RXCW      0x00180  /* Rx Configuration Word - RO */
#define E1000_EICR      0x01580  /* Ext. Interrupt Cause Read - R/clr */
#define E1000_EITR(_n)  (0x01680 + (0x4 * (_n)))
#define E1000_EICS      0x01520  /* Ext. Interrupt Cause Set - W0 */
#define E1000_EIMS      0x01524  /* Ext. Interrupt Mask Set/Read - RW */
#define E1000_EIMC      0x01528  /* Ext. Interrupt Mask Clear - WO */
#define E1000_EIAC      0x0152C  /* Ext. Interrupt Auto Clear - RW */
#define E1000_EIAM      0x01530  /* Ext. Interrupt Ack Auto Clear Mask - RW */
#define E1000_GPIE      0x01514  /* General Purpose Interrupt Enable - RW */
#define E1000_IVAR0     0x01700  /* Interrupt Vector Allocation (array) - RW */
#define E1000_IVAR_MISC 0x01740 /* IVAR for "other" causes - RW */
#define E1000_TCTL      0x00400  /* Tx Control - RW */
#define E1000_TCTL_EXT  0x00404  /* Extended Tx Control - RW */
#define E1000_TIPG      0x00410  /* Tx Inter-packet gap -RW */
#define E1000_TBT       0x00448  /* Tx Burst Timer - RW */
#define E1000_AIT       0x00458  /* Adaptive Interframe Spacing Throttle - RW */
#define E1000_LEDCTL    0x00E00  /* LED Control - RW */
#define E1000_EXTCNF_CTRL       0x00F00  /* Extended Configuration Control */
#define E1000_EXTCNF_SIZE       0x00F08  /* Extended Configuration Size */
#define E1000_PHY_CTRL          0x00F10  /* PHY Control Register in CSR */
#define E1000_PBA       0x01000  /* Packet Buffer Allocation - RW */
#define E1000_PBS       0x01008  /* Packet Buffer Size */
#define E1000_EEMNGCTL  0x01010  /* MNG EEprom Control */
#define E1000_EEARBC    0x01024  /* EEPROM Auto Read Bus Control */
#define E1000_FLASHT    0x01028  /* FLASH Timer Register */
#define E1000_EEWR      0x0102C  /* EEPROM Write Register - RW */
#define E1000_FLSWCTL   0x01030  /* FLASH control register */
#define E1000_FLSWDATA  0x01034  /* FLASH data register */
#define E1000_FLSWCNT   0x01038  /* FLASH Access Counter */
#define E1000_FLOP      0x0103C  /* FLASH Opcode Register */
#define E1000_I2CCMD    0x01028  /* SFPI2C Command Register - RW */
#define E1000_I2CPARAMS 0x0102C  /* SFPI2C Parameters Register - RW */
#define E1000_WDSTP     0x01040  /* Watchdog Setup - RW */
#define E1000_SWDSTS    0x01044  /* SW Device Status - RW */
#define E1000_FRTIMER   0x01048  /* Free Running Timer - RW */
#define E1000_TCPTIMER  0x0104C  /* TCP Timer - RW */
#define E1000_VPDDIAG   0x01060  /* VPD Diagnostic - RO */
#define E1000_ICR_V2    0x01500  /* Interrupt Cause - new location - RC */
#define E1000_ICS_V2    0x01504  /* Interrupt Cause Set - new location - WO */
/* Interrupt Mask Set/Read - new location - RW */
#define E1000_IMS_V2    0x01508
#define E1000_IMC_V2    0x0150C  /* Interrupt Mask Clear - new location - WO */
/* Interrupt Ack Auto Mask - new location - RW */
#define E1000_IAM_V2    0x01510
#define E1000_ERT       0x02008  /* Early Rx Threshold - RW */
#define E1000_FCRTL     0x02160  /* Flow Control Receive Threshold Low - RW */
#define E1000_FCRTH     0x02168  /* Flow Control Receive Threshold High - RW */
#define E1000_PSRCTL    0x02170  /* Packet Split Receive Control - RW */
#define E1000_RDFPCQ(_n)        (0x02430 + (0x4 * (_n)))
#define E1000_PBRTH     0x02458  /* PB Rx Arbitration Threshold - RW */
#define E1000_FCRTV     0x02460  /* Flow Control Refresh Timer Value - RW */
/* Split and Replication Rx Control - RW */
#define E1000_RDPUMB    0x025CC  /* DMA Rx Descriptor uC Mailbox - RW */
#define E1000_RDPUAD    0x025D0  /* DMA Rx Descriptor uC Addr Command - RW */
```

```
#define E1000_RDPUWD    0x025D4  /* DMA Rx Descriptor uC Data Write - RW */
#define E1000_RDPURD    0x025D8  /* DMA Rx Descriptor uC Data Read - RW */
#define E1000_RDPUCTL   0x025DC  /* DMA Rx Descriptor uC Control - RW */
#define E1000_PBDIAG    0x02458  /* Packet Buffer Diagnostic - RW */
#define E1000_RXPBS     0x02404  /* Rx Packet Buffer Size - RW */
/* Same as RXPBS, renamed for newer adapters - RW */
#define E1000_IRPBS     0x02404
#define E1000_RDTR      0x02820  /* Rx Delay Timer - RW */
#define E1000_RADV      0x0282C  /* Rx Interrupt Absolute Delay Timer - RW */
/*
 * Convenience macros
 *
 * Note: "_n" is the queue number of the register to be written to.
 *
 * Example usage:
 * E1000_RDBAL_REG(current_rx_queue)
 */
#define E1000_RDBAL(_n)         ((_n) < 4 ?     \
        (0x02800 + ((_n) * 0x100)) :    \
        (0x0C000 + ((_n) * 0x40)))
#define E1000_RDBAH(_n)         ((_n) < 4 ?     \
        (0x02804 + ((_n) * 0x100)) :    \
        (0x0C004 + ((_n) * 0x40)))
#define E1000_RDLEN(_n)         ((_n) < 4 ?     \
        (0x02808 + ((_n) * 0x100)) :    \
        (0x0C008 + ((_n) * 0x40)))
#define E1000_SRRCTL(_n)        ((_n) < 4 ?     \
        (0x0280C + ((_n) * 0x100)) :    \
        (0x0C00C + ((_n) * 0x40)))
#define E1000_RDH(_n)           ((_n) < 4 ?     \
        (0x02810 + ((_n) * 0x100)) :    \
        (0x0C010 + ((_n) * 0x40)))
#define E1000_RXCTL(_n)         ((_n) < 4 ?     \
        (0x02814 + ((_n) * 0x100)) :    \
        (0x0C014 + ((_n) * 0x40)))
#define E1000_DCA_RXCTRL(_n)    E1000_RXCTL(_n)
#define E1000_RDT(_n)           ((_n) < 4 ?     \
        (0x02818 + ((_n) * 0x100)) :    \
        (0x0C018 + ((_n) * 0x40)))
#define E1000_RXDCTL(_n)        ((_n) < 4 ?     \
        (0x02828 + ((_n) * 0x100)) :    \
        (0x0C028 + ((_n) * 0x40)))
#define E1000_RQDPC(_n)         ((_n) < 4 ?     \
        (0x02830 + ((_n) * 0x100)) :    \
        (0x0C030 + ((_n) * 0x40)))
#define E1000_TDBAL(_n)         ((_n) < 4 ?     \
        (0x03800 + ((_n) * 0x100)) :    \
        (0x0E000 + ((_n) * 0x40)))
#define E1000_TDBAH(_n)         ((_n) < 4 ?     \
        (0x03804 + ((_n) * 0x100)) :    \
        (0x0E004 + ((_n) * 0x40)))
#define E1000_TDLEN(_n)         ((_n) < 4 ?     \
        (0x03808 + ((_n) * 0x100)) :    \
        (0x0E008 + ((_n) * 0x40)))
#define E1000_TDH(_n)           ((_n) < 4 ?     \
        (0x03810 + ((_n) * 0x100)) :    \
        (0x0E010 + ((_n) * 0x40)))
#define E1000_TXCTL(_n)         ((_n) < 4 ?     \
        (0x03814 + ((_n) * 0x100)) :    \
        (0x0E014 + ((_n) * 0x40)))
#define E1000_DCA_TXCTRL(_n)    E1000_TXCTL(_n)
#define E1000_TDT(_n)           ((_n) < 4 ?     \
        (0x03818 + ((_n) * 0x100)) :    \
        (0x0E018 + ((_n) * 0x40)))
#define E1000_TXDCTL(_n)        ((_n) < 4 ?     \
        (0x03828 + ((_n) * 0x100)) :    \
```



```
        (0x0E028 + ((_n) * 0x40)))
#define E1000_TDWBAL(_n)        ((_n) < 4 ?     \
        (0x03838 + ((_n) * 0x100)) :    \
        (0x0E038 + ((_n) * 0x40)))
#define E1000_TDWBAH(_n)        ((_n) < 4 ?     \
        (0x0383C + ((_n) * 0x100)) :    \
        (0x0E03C + ((_n) * 0x40)))
#define E1000_TARC(_n)          (0x03840 + ((_n) * 0x100))
#define E1000_RSRPD     0x02C00  /* Rx Small Packet Detect - RW */
#define E1000_RAID      0x02C08  /* Receive Ack Interrupt Delay - RW */
#define E1000_TXDMAC    0x03000  /* Tx DMA Control - RW */
#define E1000_KABGTXD   0x03004  /* AFE Band Gap Transmit Ref Data */
#define E1000_PSRTYPE(_i)       (0x05480 + ((_i) * 4))
#define E1000_RAL(_i)           (((_i) <= 15) ? \
        (0x05400 + ((_i) * 8)) : \
        (0x054E0 + ((_i - 16) * 8)))
#define E1000_RAH(_i)           (((_i) <= 15) ? \
        (0x05404 + ((_i) * 8)) : \
        (0x054E4 + ((_i - 16) * 8)))
#define E1000_IP4AT_REG(_i)     (0x05840 + ((_i) * 8))
#define E1000_IP6AT_REG(_i)     (0x05880 + ((_i) * 4))
#define E1000_WUPM_REG(_i)      (0x05A00 + ((_i) * 4))
#define E1000_FFMT_REG(_i)      (0x09000 + ((_i) * 8))
#define E1000_FFVT_REG(_i)      (0x09800 + ((_i) * 8))
#define E1000_FFLT_REG(_i)      (0x05F00 + ((_i) * 8))
#define E1000_PBSLAC    0x03100  /* Packet Buffer Slave Access Control */
/* Packet Buffer DWORD (_n) */
#define E1000_PBSLAD(_n)        (0x03110 + (0x4 * (_n)))
#define E1000_TXPBS     0x03404  /* Tx Packet Buffer Size - RW */
/* Same as TXPBS, renamed for newer adapters - RW */
/* Same as TXPBS, renamed for newer adpaters - RW */
#define E1000_ITPBS     0x03404
#define E1000_TDFH      0x03410  /* Tx Data FIFO Head - RW */
#define E1000_TDFT      0x03418  /* Tx Data FIFO Tail - RW */
#define E1000_TDFHS     0x03420  /* Tx Data FIFO Head Saved - RW */
#define E1000_TDFTS     0x03428  /* Tx Data FIFO Tail Saved - RW */
#define E1000_TDFPC     0x03430  /* Tx Data FIFO Packet Count - RW */
#define E1000_TDPUMB    0x0357C  /* DMA Tx Descriptor uC Mail Box - RW */
#define E1000_TDPUAD    0x03580  /* DMA Tx Descriptor uC Addr Command - RW */
#define E1000_TDPUWD    0x03584  /* DMA Tx Descriptor uC Data Write - RW */
#define E1000_TDPURD    0x03588  /* DMA Tx Descriptor uC Data  Read  - RW */
#define E1000_TDPUCTL   0x0358C  /* DMA Tx Descriptor uC Control - RW */
#define E1000_DTXCTL    0x03590  /* DMA Tx Control - RW */
#define E1000_DTXTCPFLGL 0x0359C /* DMA Tx Control flag low - RW */
#define E1000_DTXTCPFLGH 0x035A0 /* DMA Tx Control flag high - RW */
#define E1000_DTXMXSZRQ  0x03540 /* DMA Tx Max Total Allow Size Requests - RW */
#define E1000_TIDV      0x03820  /* Tx Interrupt Delay Value - RW */
#define E1000_TADV      0x0382C  /* Tx Interrupt Absolute Delay Val - RW */
#define E1000_TSPMT     0x03830  /* TCP Segmentation PAD & Min Threshold - RW */
#define E1000_CRCERRS   0x04000  /* CRC Error Count - R/clr */
#define E1000_ALGNERRC  0x04004  /* Alignment Error Count - R/clr */
#define E1000_SYMERRS   0x04008  /* Symbol Error Count - R/clr */
#define E1000_RXERRC    0x0400C  /* Receive Error Count - R/clr */
#define E1000_MPC       0x04010  /* Missed Packet Count - R/clr */
#define E1000_SCC       0x04014  /* Single Collision Count - R/clr */
#define E1000_ECOL      0x04018  /* Excessive Collision Count - R/clr */
#define E1000_MCC       0x0401C  /* Multiple Collision Count - R/clr */
#define E1000_LATECOL   0x04020  /* Late Collision Count - R/clr */
#define E1000_COLC      0x04028  /* Collision Count - R/clr */
#define E1000_DC        0x04030  /* Defer Count - R/clr */
#define E1000_TNCRS     0x04034  /* Tx-No CRS - R/clr */
#define E1000_SEC       0x04038  /* Sequence Error Count - R/clr */
#define E1000_CEXTERR   0x0403C  /* Carrier Extension Error Count - R/clr */
#define E1000_RLEC      0x04040  /* Receive Length Error Count - R/clr */
#define E1000_XONRXC    0x04048  /* XON Rx Count - R/clr */
#define E1000_XONTXC    0x0404C  /* XON Tx Count - R/clr */
```

```
255 #define E1000_XOFFRXC   0x04050  /* XOFF Rx Count - R/clr */
256 #define E1000_XOFFTXC   0x04054  /* XOFF Tx Count - R/clr */
257 #define E1000_FCRUC     0x04058  /* Flow Control Rx Unsupported Count- R/clr */
258 #define E1000_PRC64     0x0405C  /* Packets Rx (64 bytes) - R/clr */
259 #define E1000_PRC127    0x04060  /* Packets Rx (65-127 bytes) - R/clr */
260 #define E1000_PRC255    0x04064  /* Packets Rx (128-255 bytes) - R/clr */
261 #define E1000_PRC511    0x04068  /* Packets Rx (255-511 bytes) - R/clr */
262 #define E1000_PRC1023   0x0406C  /* Packets Rx (512-1023 bytes) - R/clr */
263 #define E1000_PRC1522   0x04070  /* Packets Rx (1024-1522 bytes) - R/clr */
264 #define E1000_GPRC      0x04074  /* Good Packets Rx Count - R/clr */
265 #define E1000_BPRC      0x04078  /* Broadcast Packets Rx Count - R/clr */
266 #define E1000_MPRC      0x0407C  /* Multicast Packets Rx Count - R/clr */
267 #define E1000_GPTC      0x04080  /* Good Packets Tx Count - R/clr */
268 #define E1000_GORCL     0x04088  /* Good Octets Rx Count Low - R/clr */
269 #define E1000_GORCH     0x0408C  /* Good Octets Rx Count High - R/clr */
270 #define E1000_GOTCL     0x04090  /* Good Octets Tx Count Low - R/clr */
271 #define E1000_GOTCH     0x04094  /* Good Octets Tx Count High - R/clr */
272 #define E1000_RNBC      0x040A0  /* Rx No Buffers Count - R/clr */
273 #define E1000_RUC       0x040A4  /* Rx Undersize Count - R/clr */
274 #define E1000_RFC       0x040A8  /* Rx Fragment Count - R/clr */
275 #define E1000_ROC       0x040AC  /* Rx Oversize Count - R/clr */
276 #define E1000_RJC       0x040B0  /* Rx Jabber Count - R/clr */
277 #define E1000_MGTPRC    0x040B4  /* Management Packets Rx Count - R/clr */
278 #define E1000_MGTPDC    0x040B8  /* Management Packets Dropped Count - R/clr */
279 #define E1000_MGTPTC    0x040BC  /* Management Packets Tx Count - R/clr */
280 #define E1000_TORL      0x040C0  /* Total Octets Rx Low - R/clr */
281 #define E1000_TORH      0x040C4  /* Total Octets Rx High - R/clr */
282 #define E1000_TOTL      0x040C8  /* Total Octets Tx Low - R/clr */
283 #define E1000_TOTH      0x040CC  /* Total Octets Tx High - R/clr */
284 #define E1000_TPR       0x040D0  /* Total Packets Rx - R/clr */
285 #define E1000_TPT       0x040D4  /* Total Packets Tx - R/clr */
286 #define E1000_PTC64     0x040D8  /* Packets Tx (64 bytes) - R/clr */
287 #define E1000_PTC127    0x040DC  /* Packets Tx (65-127 bytes) - R/clr */
288 #define E1000_PTC255    0x040E0  /* Packets Tx (128-255 bytes) - R/clr */
289 #define E1000_PTC511    0x040E4  /* Packets Tx (256-511 bytes) - R/clr */
290 #define E1000_PTC1023   0x040E8  /* Packets Tx (512-1023 bytes) - R/clr */
291 #define E1000_PTC1522   0x040EC  /* Packets Tx (1024-1522 Bytes) - R/clr */
292 #define E1000_MPTC      0x040F0  /* Multicast Packets Tx Count - R/clr */
293 #define E1000_BPTC      0x040F4  /* Broadcast Packets Tx Count - R/clr */
294 #define E1000_TSCTC     0x040F8  /* TCP Segmentation Context Tx - R/clr */
295 #define E1000_TSCTFC    0x040FC  /* TCP Segmentation Context Tx Fail - R/clr */
296 #define E1000_IAC       0x04100  /* Interrupt Assertion Count */
297 #define E1000_ICRXPTC   0x04104  /* Interrupt Cause Rx Pkt Timer Expire Count */
298 #define E1000_ICRXATC   0x04108  /* Interrupt Cause Rx Abs Timer Expire Count */
299 #define E1000_ICTXPTC   0x0410C  /* Interrupt Cause Tx Pkt Timer Expire Count */
300 #define E1000_ICTXATC   0x04110  /* Interrupt Cause Tx Abs Timer Expire Count */
301 #define E1000_ICTXQEC   0x04118  /* Interrupt Cause Tx Queue Empty Count */
302 #define E1000_ICTXQMTC  0x0411C  /* Interrupt Cause Tx Queue Min Thresh Count */
303 #define E1000_ICRXDMTC  0x04120  /* Interrupt Cause Rx Desc Min Thresh Count */
304 #define E1000_ICRXOC    0x04124  /* Interrupt Cause Receiver Overrun Count */

306 /* LinkSec Tx Untagged Packet Count - OutPktsUntagged */
307 #define E1000_LSECTXUT          0x04300
308 /* LinkSec Encrypted Tx Packets Count - OutPktsEncrypted */
309 #define E1000_LSECTXPKTE        0x04304
310 /* LinkSec Protected Tx Packet Count - OutPktsProtected */
311 #define E1000_LSECTXPKTP        0x04308
312 /* LinkSec Encrypted Tx Octets Count - OutOctetsEncrypted */
313 #define E1000_LSECTXOCTE        0x0430C
314 /* LinkSec Protected Tx Octets Count - OutOctetsProtected */
315 #define E1000_LSECTXOCTP        0x04310
316 /* LinkSec Untagged non-Strict Rx Packet Count - InPktsUntagged/InPktsNoTag */
317 #define E1000_LSECRXUT          0x04314
318 /* LinkSec Rx Octets Decrypted Count - InOctetsDecrypted */
319 #define E1000_LSECRXOCTD        0x0431C
320 /* LinkSec Rx Octets Validated - InOctetsValidated */
```



```
321 #define E1000_LSECRXOCTV        0x04320
322 /* LinkSec Rx Bad Tag - InPktsBadTag */
323 #define E1000_LSECRXBAD         0x04324
324 /* LinkSec Rx Packet No SCI Count - InPktsNoSci */
325 #define E1000_LSECRXNOSCI       0x04328
326 /* LinkSec Rx Packet Unknown SCI Count - InPktsUnknownSci */
327 #define E1000_LSECRXUNSCI       0x0432C
328 /* LinkSec Rx Unchecked Packets Count - InPktsUnchecked */
329 #define E1000_LSECRXUNCH        0x04330
330 /* LinkSec Rx Delayed Packet Count - InPktsDelayed */
331 #define E1000_LSECRXDELAY       0x04340
332 /* LinkSec Rx Late Packets Count - InPktsLate */
333 #define E1000_LSECRXLATE        0x04350
334 /* LinkSec Rx Packet OK Count - InPktsOk */
335 #define E1000_LSECRXOK(_n)      (0x04360 + (0x04 * (_n)))
336 /* LinkSec Rx Invalid Count - InPktsInvalid */
337 #define E1000_LSECRXINV(_n)     (0x04380 + (0x04 * (_n)))
338 /* LinkSec Rx Not Valid Count - InPktsNotValid */
339 #define E1000_LSECRXNV(_n)      (0x043A0 + (0x04 * (_n)))
340 /* LinkSec Rx Unused SA Count - InPktsUnusedSa */
341 #define E1000_LSECRXUNSA        0x043C0
342 /* LinkSec Rx Not Using SA Count - InPktsNotUsingSa */
343 #define E1000_LSECRXNUSA        0x043D0
344 /* LinkSec Tx Capabilities Register - RO */
345 #define E1000_LSECTXCAP         0x0B000
346 /* LinkSec Rx Capabilities Register - RO */
347 #define E1000_LSECRXCAP         0x0B300
348 #define E1000_LSECTXCTRL        0x0B004  /* LinkSec Tx Control - RW */
349 #define E1000_LSECRXCTRL        0x0B304  /* LinkSec Rx Control - RW */
350 #define E1000_LSECTXSCL         0x0B008  /* LinkSec Tx SCI Low - RW */
351 #define E1000_LSECTXSCH         0x0B00C  /* LinkSec Tx SCI High - RW */
352 #define E1000_LSECTXSA          0x0B010  /* LinkSec Tx SA0 - RW */
353 #define E1000_LSECTXPN0         0x0B018  /* LinkSec Tx SA PN 0 - RW */
354 #define E1000_LSECTXPN1         0x0B01C  /* LinkSec Tx SA PN 1 - RW */
355 #define E1000_LSECRXSCL         0x0B3D0  /* LinkSec Rx SCI Low - RW */
356 #define E1000_LSECRXSCH         0x0B3E0  /* LinkSec Rx SCI High - RW */
357 /* LinkSec Tx 128-bit Key 0 - WO */
358 #define E1000_LSECTXKEY0(_n)    (0x0B020 + (0x04 * (_n)))
359 /* LinkSec Tx 128-bit Key 1 - WO */
360 #define E1000_LSECTXKEY1(_n)    (0x0B030 + (0x04 * (_n)))
361 /* LinkSec Rx SAs - RW */
362 #define E1000_LSECRXSA(_n)      (0x0B310 + (0x04 * (_n)))
363 /* LinkSec Rx SAs - RW */
364 #define E1000_LSECRXPN(_n)      (0x0B330 + (0x04 * (_n)))
365 /*
366  * LinkSec Rx Keys  - where _n is the SA no. and _m the 4 dwords of the 128 bit
367  * key - RW.
368  */
369 #define E1000_LSECRXKEY(_n, _m) (0x0B350 + (0x10 * (_n)) + (0x04 * (_m)))

371 #define E1000_SSVPC     0x041A0  /* Switch Security Violation Packet Count */
372 #define E1000_IPSCTRL   0xB430   /* IpSec Control Register */
373 #define E1000_IPSRXCMD  0x0B408  /* IPSec Rx Command Register - RW */
374 #define E1000_IPSRXIDX  0x0B400  /* IPSec Rx Index - RW */
375 /* IPSec Rx IPv4/v6 Address - RW */
376 #define E1000_IPSRXIPADDR(_n)   (0x0B420+ (0x04 * (_n)))
377 /* IPSec Rx 128-bit Key - RW */
378 #define E1000_IPSRXKEY(_n)      (0x0B410 + (0x04 * (_n)))
379 #define E1000_IPSRXSALT 0x0B404  /* IPSec Rx Salt - RW */
380 #define E1000_IPSRXSPI  0x0B40C  /* IPSec Rx SPI - RW */
381 /* IPSec Tx 128-bit Key - RW */
382 #define E1000_IPSTXKEY(_n)      (0x0B460 + (0x04 * (_n)))
383 #define E1000_IPSTXSALT 0x0B454  /* IPSec Tx Salt - RW */
384 #define E1000_IPSTXIDX  0x0B450  /* IPSec Tx SA IDX - RW */
385 #define E1000_PCS_CFG0  0x04200  /* PCS Configuration 0 - RW */
386 #define E1000_PCS_LCTL  0x04208  /* PCS Link Control - RW */
```

```
387 #define E1000_PCS_LSTAT 0x0420C  /* PCS Link Status - RO */
388 #define E1000_CBTMPC    0x0402C  /* Circuit Breaker Tx Packet Count */
389 #define E1000_HTDPMC    0x0403C  /* Host Transmit Discarded Packets */
390 #define E1000_CBRDPC    0x04044  /* Circuit Breaker Rx Dropped Count */
391 #define E1000_CBRMPC    0x040FC  /* Circuit Breaker Rx Packet Count */
392 #define E1000_RPTHC     0x04104  /* Rx Packets To Host */
393 #define E1000_HGPTC     0x04118  /* Host Good Packets Tx Count */
394 #define E1000_HTCBDPC   0x04124  /* Host Tx Circuit Breaker Dropped Count */
395 #define E1000_HGORCL    0x04128  /* Host Good Octets Received Count Low */
396 #define E1000_HGORCH    0x0412C  /* Host Good Octets Received Count High */
397 #define E1000_HGOTCL    0x04130  /* Host Good Octets Transmit Count Low */
398 #define E1000_HGOTCH    0x04134  /* Host Good Octets Transmit Count High */
399 #define E1000_LENERRS   0x04138  /* Length Errors Count */
400 #define E1000_SCVPC     0x04228  /* SerDes/SGMII Code Violation Pkt Count */
401 #define E1000_HRMPC     0x0A018  /* Header Redirection Missed Packet Count */
402 #define E1000_PCS_ANADV 0x04218  /* AN advertisement - RW */
403 #define E1000_PCS_LPAB  0x0421C  /* Link Partner Ability - RW */
404 #define E1000_PCS_NPTX  0x04220  /* AN Next Page Transmit - RW */
405 #define E1000_PCS_LPABNP 0x04224 /* Link Partner Ability Next Page - RW */
406 #define E1000_1GSTAT_RCV 0x04228 /* 1GSTAT Code Violation Packet Count - RW */
407 #define E1000_RXCSUM    0x05000  /* Rx Checksum Control - RW */
408 #define E1000_RLPML     0x05004  /* Rx Long Packet Max Length */
409 #define E1000_RFCTL     0x05008  /* Receive Filter Control */
410 #define E1000_MTA       0x05200  /* Multicast Table Array - RW Array */
411 #define E1000_RA        0x05400  /* Receive Address - RW Array */
412 /* 2nd half of receive address array - RW Array */
413 #define E1000_RA2       0x054E0
414 #define E1000_VFTA      0x05600  /* VLAN Filter Table Array - RW Array */
415 #define E1000_VT_CTL    0x0581C  /* VMDq Control - RW */
416 #define E1000_VFQA0     0x0B000  /* VLAN Filter Queue Array 0 - RW Array */
417 #define E1000_VFQA1     0x0B200  /* VLAN Filter Queue Array 1 - RW Array */
418 #define E1000_WUC       0x05800  /* Wakeup Control - RW */
419 #define E1000_WUFC      0x05808  /* Wakeup Filter Control - RW */
420 #define E1000_WUS       0x05810  /* Wakeup Status - RO */
421 #define E1000_MANC      0x05820  /* Management Control - RW */
422 #define E1000_IPAV      0x05838  /* IP Address Valid - RW */
423 #define E1000_IP4AT     0x05840  /* IPv4 Address Table - RW Array */
424 #define E1000_IP6AT     0x05880  /* IPv6 Address Table - RW Array */
425 #define E1000_WUPL      0x05900  /* Wakeup Packet Length - RW */
426 #define E1000_WUPM      0x05A00  /* Wakeup Packet Memory - RO A */
427 #define E1000_PBACL     0x05B68  /* MSIx PBA Clear - Read/Write 1's to clear */
428 #define E1000_FFLT      0x05F00  /* Flexible Filter Length Table - RW Array */
429 #define E1000_HOST_IF   0x08800  /* Host Interface */
430 #define E1000_FFMT      0x09000  /* Flexible Filter Mask Table - RW Array */
431 #define E1000_FFVT      0x09800  /* Flexible Filter Value Table - RW Array */
432 /* Flexible Host Filter Table */
433 #define E1000_FHFT(_n)  (0x09000 + (_n * 0x100))
434 /* Ext Flexible Host Filter Table */
435 #define E1000_FHFT_EXT(_n) (0x09A00 + (_n * 0x100))

437 #define E1000_KMRNCTRLSTA       0x00034 /* MAC-PHY interface - RW */
438 #define E1000_MDPHYA            0x0003C /* PHY address - RW */
439 #define E1000_MANC2H            0x05860 /* Management Control To Host - RW */
440 /* Software-Firmware Synchronization - RW */
441 #define E1000_SW_FW_SYNC        0x05B5C
442 #define E1000_CCMCTL            0x05B48 /* CCM Control Register */
443 #define E1000_GIOCTL            0x05B44 /* GIO Analog Control Register */
444 #define E1000_SCCTL             0x05B4C /* PCIc PLL Configuration Register */
445 #define E1000_GCR               0x05B00 /* PCI-Ex Control */
446 #define E1000_GCR2              0x05B64 /* PCI-Ex Control #2 */
447 #define E1000_GSCL_1            0x05B10 /* PCI-Ex Statistic Control #1 */
448 #define E1000_GSCL_2            0x05B14 /* PCI-Ex Statistic Control #2 */
449 #define E1000_GSCL_3            0x05B18 /* PCI-Ex Statistic Control #3 */
450 #define E1000_GSCL_4            0x05B1C /* PCI-Ex Statistic Control #4 */
451 /* Function Active and Power State to MNG */
452 #define E1000_FACTPS            0x05B30
```



```
453 #define E1000_SWSM              0x05B50 /* SW Semaphore */
454 #define E1000_FWSM              0x05B54 /* FW Semaphore */
455 /* Driver-only SW semaphore (not used by BOOT agents) */
456 #define E1000_SWSM2             0x05B58
457 #define E1000_DCA_ID            0x05B70 /* DCA Requester ID Information - RO */
458 #define E1000_DCA_CTRL          0x05B74 /* DCA Control - RW */
459 #define E1000_UFUSE             0x05B78 /* UFUSE - RO */
460 #define E1000_FFLT_DBG          0x05F04 /* Debug Register */
461 #define E1000_HICR              0x08F00 /* Host Interface Control */

463 /* RSS registers */
464 #define E1000_CPUVEC            0x02C10 /* CPU Vector Register - RW */
465 #define E1000_MRQC              0x05818 /* Multiple Receive Control - RW */
466 #define E1000_IMIR(_i)  (0x05A80 + ((_i) * 4))  /* Immediate Interrupt */
467 /* Immediate Interrupt Ext */
468 #define E1000_IMIREXT(_i)       (0x05AA0 + ((_i) * 4))
469 #define E1000_IMIRVP    0x05AC0 /* Immediate Interrupt Rx VLAN Priority - RW */
470 /* MSI-X Allocation Register (_i) - RW */
471 #define E1000_MSIXBM(_i)        (0x01600 + ((_i) * 4))
472 /* MSI-X Table entry addr low reg 0 - RW */
473 #define E1000_MSIXTADD(_i)      (0x0C000 + ((_i) * 0x10))
474 /* MSI-X Table entry addr upper reg 0 - RW */
475 #define E1000_MSIXTUADD(_i)     (0x0C004 + ((_i) * 0x10))
476 /* MSI-X Table entry message reg 0 - RW */
477 #define E1000_MSIXTMSG(_i)      (0x0C008 + ((_i) * 0x10))
478 /* MSI-X Table entry vector ctrl reg 0 - RW */
479 #define E1000_MSIXVCTRL(_i)     (0x0C00C + ((_i) * 0x10))
480 #define E1000_MSIXPBA           0x0E000 /* MSI-X Pending bit array */
481 /* Redirection Table - RW Array */
482 #define E1000_RETA(_i)          (0x05C00 + ((_i) * 4))
483 /* RSS Random Key - RW Array */
484 #define E1000_RSSRK(_i)         (0x05C80 + ((_i) * 4))
485 #define E1000_RSSIM     0x05864 /* RSS Interrupt Mask */
486 #define E1000_RSSIR     0x05868 /* RSS Interrupt Request */
487 /* VT Registers */
488 #define E1000_SWPBS     0x03004 /* Switch Packet Buffer Size - RW */
489 #define E1000_MBVFICR   0x00C80 /* Mailbox VF Cause - RWC */
490 #define E1000_MBVFIMR   0x00C84 /* Mailbox VF int Mask - RW */
491 #define E1000_VFLRE     0x00C88 /* VF Register Events - RWC */
492 #define E1000_VFRE      0x00C8C /* VF Receive Enables */
493 #define E1000_VFTE      0x00C90 /* VF Transmit Enables */
494 #define E1000_QDE       0x02408 /* Queue Drop Enable - RW */
495 #define E1000_DTXSWC    0x03500 /* DMA Tx Switch Control - RW */
496 #define E1000_RPLOLR    0x05AF0 /* Replication Offload - RW */
497 #define E1000_UTA       0x0A000 /* Unicast Table Array - RW */
498 #define E1000_IOVTCL    0x05BBC /* IOV Control Register */
499 #define E1000_VMRCTL    0X05D80 /* Virtual Mirror Rule Control */
500 /* These act per VF so an array friendly macro is used */
501 #define E1000_V2PMAILBOX(_n)    (0x00C40 + (4 * (_n)))
502 #define E1000_P2VMAILBOX(_n)    (0x00C00 + (4 * (_n)))
503 #define E1000_VMBMEM(_n)        (0x00800 + (64 * (_n)))
504 #define E1000_VFVMBMEM(_n)      (0x00800 + (_n))
505 #define E1000_VMOLR(_n)         (0x05AD0 + (4 * (_n)))
506 /* VLAN Virtual Machine Filter - RW */
507 #define E1000_VLVF(_n)          (0x05D00 + (4 * (_n)))
508 #define E1000_VMVIR(_n)         (0x03700 + (4 * (_n)))

510 /* Filtering Registers */
511 #define E1000_SAQF(_n)  (0x05980 + (4 * (_n))) /* Source Address Queue Fltr */
512 #define E1000_DAQF(_n)  (0x059A0 + (4 * (_n))) /* Dest Address Queue Fltr */
513 #define E1000_SPQF(_n)  (0x059C0 + (4 * (_n))) /* Source Port Queue Fltr */
514 #define E1000_FTQF(_n)  (0x059E0 + (4 * (_n))) /* 5-tuple Queue Fltr */
515 #define E1000_TTQF(_n)  (0x059E0 + (4 * (_n))) /* 2-tuple Queue Fltr */
516 #define E1000_SYNQF(_n) (0x055FC + (4 * (_n))) /* SYN Packet Queue Fltr */
517 #define E1000_ETQF(_n)  (0x05CB0 + (4 * (_n))) /* EType Queue Fltr */
```

```
519 #define E1000_RTTDCS    0x3600 /* Reedtown Tx Desc plane control and status */
520 #define E1000_RTTPCS    0x3474 /* Reedtown Tx Packet Plane control and status */
521 #define E1000_RTRPCS    0x2474 /* Rx packet plane control and status */
522 #define E1000_RTRUP2TC  0x05AC4 /* Rx User Priority to Traffic Class */
523 #define E1000_RTTUP2TC  0x0418  /* Transmit User Priority to Traffic Class */
524 /* Tx Desc plane TC Rate-scheduler config */
525 #define E1000_RTTDTCRC(_n)      (0x3610 + ((_n) * 4))
526 /* Tx Packet plane TC Rate-Scheduler Config */
527 #define E1000_RTTPTCRC(_n)      (0x3480 + ((_n) * 4))
528 /* Rx Packet plane TC Rate-Scheduler Config */
529 #define E1000_RTRPTCRC(_n)      (0x2480 + ((_n) * 4))
530 /* Tx Desc Plane TC Rate-Scheduler Status */
531 #define E1000_RTTDTCRS(_n)      (0x3630 + ((_n) * 4))
532 /* Tx Desc Plane TC Rate-Scheduler MMW */
533 #define E1000_RTTDTCRM(_n)      (0x3650 + ((_n) * 4))
534 /* Tx Packet plane TC Rate-Scheduler Status */
535 #define E1000_RTTPTCRS(_n)      (0x34A0 + ((_n) * 4))
536 /* Tx Packet plane TC Rate-scheduler MMW */
537 #define E1000_RTTPTCRM(_n)      (0x34C0 + ((_n) * 4))
538 /* Rx Packet plane TC Rate-Scheduler Status */
539 #define E1000_RTRPTCRS(_n)      (0x24A0 + ((_n) * 4))
540 /* Rx Packet plane TC Rate-Scheduler MMW */
541 #define E1000_RTRPTCRM(_n)      (0x24C0 + ((_n) * 4))
542 /* Tx Desc plane VM Rate-Scheduler MMW */
543 #define E1000_RTTDVMRM(_n)      (0x3670 + ((_n) * 4))
544 /* Tx BCN Rate-Scheduler MMW */
545 #define E1000_RTTBCNRM(_n)      (0x3690 + ((_n) * 4))
546 #define E1000_RTTDQSEL  0x3604  /* Tx Desc Plane Queue Select */
547 #define E1000_RTTDVMRC  0x3608  /* Tx Desc Plane VM Rate-Scheduler Config */
548 #define E1000_RTTDVMRS  0x360C  /* Tx Desc Plane VM Rate-Scheduler Status */
549 #define E1000_RTTBCNRC  0x36B0  /* Tx BCN Rate-Scheduler Config */
550 #define E1000_RTTBCNRS  0x36B4  /* Tx BCN Rate-Scheduler Status */
551 #define E1000_RTTBCNCR  0xB200  /* Tx BCN Control Register */
552 #define E1000_RTTBCNTG  0x35A4  /* Tx BCN Tagging */
553 #define E1000_RTTBCNCP  0xB208  /* Tx BCN Congestion point */
554 #define E1000_RTRBCNCR  0xB20C  /* Rx BCN Control Register */
555 #define E1000_RTTBCNRD  0x36B8  /* Tx BCN Rate Drift */
556 #define E1000_PFCTOP    0x1080  /* Priority Flow Control Type and Opcode */
557 #define E1000_RTTBCNIDX 0xB204  /* Tx BCN Congestion Point */
558 #define E1000_RTTBCNACH 0x0B214 /* Tx BCN Control High */
559 #define E1000_RTTBCNACL 0x0B210 /* Tx BCN Control Low */

561 /* DMA Coalescing registers */
562 #define E1000_DMACR     0x02508 /* Control Register */
563 #define E1000_DMCTXTH   0x03550 /* Transmit Threshold */
564 #define E1000_DMCTLX    0x02514 /* Time to Lx Request */
565 #define E1000_DMCRTRH   0x05DD0 /* Receive Packet Rate Threshold */
566 #define E1000_DMCCNT    0x05DD4 /* Current RX Count */
567 #define E1000_FCRTC     0x02170 /* Flow Control Rx high watermark */
568 #define E1000_PCIEMISC  0x05BB8 /* PCIE misc config register */

570 /* PCIe Parity Status Register */
571 #define E1000_PCIEERRSTS        0x05BA8

573 /* Energy Efficient Ethernet "EEE" registers */
574 #define E1000_IPCNFG    0x0E38 /* Internal PHY Configuration */
575 #define E1000_LTRC      0x01A0 /* Latency Tolerance Reporting Control */
576 #define E1000_EEER      0x0E30 /* Energy Efficient Ethernet "EEE" */
577 #define E1000_EEE_SU    0x0E34 /* EEE Setup */
578 #define E1000_TLPIC     0x4148 /* EEE Tx LPI Count - TLPIC */
579 #define E1000_RLPIC     0x414C /* EEE Rx LPI Count - RLPIC */

581 #ifdef __cplusplus
582 }
583 #endif
584 #endif  /* _IGB_REGS_H */
```

```
**********************************************************
   72161 Tue Sep 25 07:16:34 2012
new/usr/src/uts/common/io/ipw/ipw2100.c
3217 cfgadm should spell "adaptors" correctly
Reviewed by: Alexander Eremin <alexander.r.eremin@gmail.com>
Reviewed by: David Hoeppner <0xffea@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
Reviewed by: Eric Schrock <Eric.Schrock@delphix.com>
**********************************************************
______unchanged_portion_omitted_

1264 /*
1265  * all ipw2100 interrupts will be masked by this routine
1266  */
1267 static int
1268 ipw2100_chip_reset(struct ipw2100_softc *sc)
1269 {
1270         int             ntries;
1271         uint32_t        tmp;

1273         ipw2100_master_stop(sc);

1275         /*
1276          * move adapter to DO state
1276          * move adatper to DO state
1277          */
1278         tmp = ipw2100_csr_get32(sc, IPW2100_CSR_CTL);
1279         ipw2100_csr_put32(sc, IPW2100_CSR_CTL, tmp | IPW2100_CTL_INIT);

1281         /*
1282          * wait for clock stabilization
1283          */
1284         for (ntries = 0; ntries < 1000; ntries++) {
1285                 if (ipw2100_csr_get32(sc, IPW2100_CSR_CTL)
1286                     & IPW2100_CTL_CLOCK_READY)
1287                         break;
1288                 drv_usecwait(200);
1289         }
1290         if (ntries == 1000)
1291                 return (DDI_FAILURE);

1293         tmp = ipw2100_csr_get32(sc, IPW2100_CSR_RST);
1294         ipw2100_csr_put32(sc, IPW2100_CSR_RST, tmp | IPW2100_RST_SW_RESET);

1296         drv_usecwait(10);

1298         tmp = ipw2100_csr_get32(sc, IPW2100_CSR_CTL);
1299         ipw2100_csr_put32(sc, IPW2100_CSR_CTL, tmp | IPW2100_CTL_INIT);

1301         return (DDI_SUCCESS);
1302 }
______unchanged_portion_omitted_
```

```
**********************************************************
    3478 Tue Sep 25 07:16:35 2012
new/usr/src/uts/common/io/iwi/ipw2200.h
3217 cfgadm should spell "adaptors" correctly
Reviewed by: Alexander Eremin <alexander.r.eremin@gmail.com>
Reviewed by: David Hoeppner <0xffea@gmail.com>
Reviewed by: Gary Mills <gary_mills@fastmail.fm>
Reviewed by: Eric Schrock <Eric.Schrock@delphix.com>
**********************************************************
```

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

/*
 * Copyright(c) 2004
 *      Damien Bergamini <damien.bergamini@free.fr>. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice unmodified, this list of conditions, and the following
 *    disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#ifndef _SYS_IPW2200_H
#define _SYS_IPW2200_H

#ifdef __cplusplus
extern "C" {
#endif

/*
 * Intel Wireless PRO/2200 mini-pci adapter drier
 * Intel Wireless PRO/2200 mini-pci adpater drier
 * ipw2200.h: common definitions and interface to user land application
 */
#include <sys/types.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>

#define IPW2200_DRV_NAME  "iwi"
#define IPW2200_DRV_DESC  "Intel Wireless 2200/2915"

/*
 * Debug functionalities
 */
#define IPW2200_DBG_INIT        (0x00000001)  /* initialization */
#define IPW2200_DBG_GLD         (0x00000002)  /* GLD */
#define IPW2200_DBG_WIFI        (0x00000004)  /* WiFi */
```



```
#define IPW2200_DBG_DMA         (0x00000008)  /* DMA */
#define IPW2200_DBG_CSR         (0x00000010)  /* low-level CSR access */
#define IPW2200_DBG_FW          (0x00000020)  /* uc & fw */
#define IPW2200_DBG_RING        (0x00000040)  /* ring operations */
#define IPW2200_DBG_IOCTL       (0x00000080)  /* ioctl */
#define IPW2200_DBG_INT         (0x00000100)  /* interrupt */
#define IPW2200_DBG_TABLE       (0x00000200)  /* ipw2200 tables */
#define IPW2200_DBG_RX          (0x00000400)  /* rx */
#define IPW2200_DBG_TX          (0x00000800)  /* tx */
#define IPW2200_DBG_HWCAP       (0x00001000)  /* hardware capabilities */
#define IPW2200_DBG_NOTIF       (0x00002000)  /* ipw2200 notification */
#define IPW2200_DBG_SCAN        (0x00004000)  /* scan results */
#define IPW2200_DBG_SOFTINT     (0x00008000)  /* soft interrupt */
#define IPW2200_DBG_FATAL       (0x00010000)  /* interrupt report error */
#define IPW2200_DBG_SUSPEND     (0x00020000)  /* suspend resume */
#define IPW2200_DBG_BRUSSELS    (0x00040000)  /* brussels support */

extern uint32_t ipw2200_debug;
extern void     ipw2200_dbg(dev_info_t *dip, int level, const char *fmt, ...);

#ifdef  DEBUG
#define IPW2200_DBG(l, x)       do {                            \
        _NOTE(CONSTANTCONDITION)                                \
        if ((l) & ipw2200_debug)                                \
                ipw2200_dbg x;                                  \
        _NOTE(CONSTANTCONDITION)                                \
} while (0)
```

*unchanged_portion_omitted_*