

new/usr/src/cmd/smbios/smbios.c

1

```
*****
34314 Thu Mar 26 17:12:00 2015
new/usr/src/cmd/smbios/smbios.c
5094 Update libbios with recent items
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
Reviewed by: Garrett D'Amore <garrett@damore.org>
Reviewed by: Robert Mustacchi <rm@joyent.com>
*****

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

22 /*
23  * Copyright 2015 OmniTI Computer Consulting, Inc. All rights reserved.
24  * Copyright 2010 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26 */

28 #include <sys/sysmacros.h>
29 #include <sys/param.h>

31 #include <smbios.h>
32 #include <alloca.h>
33 #include <limits.h>
34 #include <unistd.h>
35 #include <strings.h>
36 #include <stdlib.h>
37 #include <stdarg.h>
38 #include <stdio.h>
39 #include <fcntl.h>
40 #include <errno.h>
41 #include <ctype.h>

43 #define SMBIOS_SUCCESS 0
44 #define SMBIOS_ERROR 1
45 #define SMBIOS_USAGE 2

47 static const char *g_pname;
48 static int g_hdr;

50 static int opt_e;
51 static int opt_i = -1;
52 static int opt_o;
53 static int opt_s;
54 static int opt_t = -1;
55 static int opt_x;

57 /*PRINTLIKE2*/
```

new/usr/src/cmd/smbios/smbios.c

2

```
58 static void
59 oprintf(FILE *fp, const char *format, ...)
60 {
61     va_list ap;

63     va_start(ap, format);
64     (void) vfprintf(fp, format, ap);
65     va_end(ap);
66 }

    unchanged_portion_omitted

332 static void
333 print_chassis(smbios_hdl_t *shp, id_t id, FILE *fp)
334 {
335     smbios_chassis_t c;
336     int elem_cnt;

338     (void) smbios_info_chassis(shp, id, &c);

340     oprintf(fp, " OEM Data: 0x%x\n", c.smbc_oemdata);
341     oprintf(fp, " SKU number: %s\n",
342             c.smbc_sku == NULL ? "<unknown>" : c.smbc_sku);
343     oprintf(fp, " Lock Present: %s\n", c.smbc_lock ? "Y" : "N");

345     desc_printf(smbios_chassis_type_desc(c.smbc_type),
346                fp, " Chassis Type: 0x%x", c.smbc_type);

348     desc_printf(smbios_chassis_state_desc(c.smbc_bustate),
349                fp, " Boot-Up State: 0x%x", c.smbc_bustate);

351     desc_printf(smbios_chassis_state_desc(c.smbc_psstate),
352                fp, " Power Supply State: 0x%x", c.smbc_psstate);

354     desc_printf(smbios_chassis_state_desc(c.smbc_thstate),
355                fp, " Thermal State: 0x%x", c.smbc_thstate);

357     oprintf(fp, " Chassis Height: %u\n", c.smbc_uheight);
358     oprintf(fp, " Power Cords: %u\n", c.smbc_cords);

360     elem_cnt = c.smbc_elems;
361     oprintf(fp, " Element Records: %u\n", elem_cnt);

363     if (elem_cnt > 0) {
364         id_t *elems;
365         uint8_t type;
366         int i, n, cnt;

368         elems = alloca(c.smbc_elems * sizeof(id_t));
369         cnt = smbios_info_contains(shp, id, elem_cnt, elems);
370         if (cnt > SMB_CONT_MAX) {
371             return;
372             n = MIN(elem_cnt, cnt);
373         }

374         oprintf(fp, "\n");
375         for (i = 0; i < n; i++) {
376             type = (uint8_t)elems[i];
377             if (type & 0x80) {
378                 /* SMBIOS structure Type */
379                 desc_printf(smbios_type_name(type & 0x7f), fp,
380                            " Contained SMBIOS structure Type: %u",
381                            type & 0x80);
382             } else {
383                 /* SMBIOS Base Board Type */
384                 desc_printf(smbios_bboard_type_desc(type), fp,
385                            " Contained SMBIOS Base Board Type: 0x%x",
386                            type);
387             }
388         }
389     }
390 }
```

```

387     }
388     }
389 }
390 }

392 static void
393 print_processor(smbios_hdl_t *shp, id_t id, FILE *fp)
394 {
395     smbios_processor_t p;
396     uint_t status;

398     (void) smbios_info_processor(shp, id, &p);
399     status = SMB_PRSTATUS_STATUS(p.smbp_status);

401     desc_printf(smbios_processor_family_desc(p.smbp_family),
402                fp, " Family: %u", p.smbp_family);

404     if (p.smbp_family2 != 0)
405         desc_printf(smbios_processor_family_desc(p.smbp_family2),
406                    fp, " Family Ext: %u", p.smbp_family2);

408     fprintf(fp, " CPUID: 0x%llx\n", (u_longlong_t)p.smbp_cpuid);

410     desc_printf(smbios_processor_type_desc(p.smbp_type),
411                fp, " Type: %u", p.smbp_type);

413     desc_printf(smbios_processor_upgrade_desc(p.smbp_upgrade),
414                fp, " Socket Upgrade: %u", p.smbp_upgrade);

416     fprintf(fp, " Socket Status: %s\n",
417            SMB_PRSTATUS_PRESENT(p.smbp_status) ?
418            "Populated" : "Not Populated");

420     desc_printf(smbios_processor_status_desc(status),
421                fp, " Processor Status: %u", status);

423     if (SMB_PRV_LEGACY(p.smbp_voltage)) {
424         fprintf(fp, " Supported Voltages:");
425         switch (p.smbp_voltage) {
426             case SMB_PRV_5V:
427                 fprintf(fp, " 5.0V");
428                 break;
429             case SMB_PRV_33V:
430                 fprintf(fp, " 3.3V");
431                 break;
432             case SMB_PRV_29V:
433                 fprintf(fp, " 2.9V");
434                 break;
435             }
436         fprintf(fp, "\n");
437     } else {
438         fprintf(fp, " Supported Voltages: %.1fV\n",
439                (float)SMB_PRV_VOLTAGE(p.smbp_voltage) / 10);
440     }

442     if (p.smbp_corecount != 0)
443         fprintf(fp, " Core Count: %u\n", p.smbp_corecount);
444     else
445         fprintf(fp, " Core Count: Unknown\n");

447     if (p.smbp_coresenabled != 0)
448         fprintf(fp, " Cores Enabled: %u\n", p.smbp_coresenabled);
449     else
450         fprintf(fp, " Cores Enabled: Unknown\n");

452     if (p.smbp_threadcount != 0)

```

```

453         fprintf(fp, " Thread Count: %u\n", p.smbp_threadcount);
454     else
455         fprintf(fp, " Thread Count: Unknown\n");

457     if (p.smbp_cflags) {
458         flag_printf(fp, "Processor Characteristics",
459                    p.smbp_cflags, sizeof (p.smbp_cflags) * NBBY,
460                    smbios_processor_core_flag_name,
461                    smbios_processor_core_flag_desc);
462     }

464     if (p.smbp_clkspeed != 0)
465         fprintf(fp, " External Clock Speed: %uMHz\n", p.smbp_clkspeed);
466     else
467         fprintf(fp, " External Clock Speed: Unknown\n");

469     if (p.smbp_maxspeed != 0)
470         fprintf(fp, " Maximum Speed: %uMHz\n", p.smbp_maxspeed);
471     else
472         fprintf(fp, " Maximum Speed: Unknown\n");

474     if (p.smbp_curspeed != 0)
475         fprintf(fp, " Current Speed: %uMHz\n", p.smbp_curspeed);
476     else
477         fprintf(fp, " Current Speed: Unknown\n");

479     id_printf(fp, " L1 Cache: ", p.smbp_l1cache);
480     id_printf(fp, " L2 Cache: ", p.smbp_l2cache);
481     id_printf(fp, " L3 Cache: ", p.smbp_l3cache);
482 }

    unchanged_portion_omitted

758 static void
759 print_memdevice(smbios_hdl_t *shp, id_t id, FILE *fp)
760 {
761     smbios_memdevice_t md;

763     (void) smbios_info_memdevice(shp, id, &md);

765     id_printf(fp, " Physical Memory Array: ", md.smbmd_array);
766     id_printf(fp, " Memory Error Data: ", md.smbmd_error);

768     if (md.smbmd_twidth != -1u)
769         fprintf(fp, " Total Width: %u bits\n", md.smbmd_twidth);
770     else
771         fprintf(fp, " Total Width: Unknown\n");

773     if (md.smbmd_dwidth != -1u)
774         fprintf(fp, " Data Width: %u bits\n", md.smbmd_dwidth);
775     else
776         fprintf(fp, " Data Width: Unknown\n");

778     switch (md.smbmd_size) {
779     case -1ull:
780         fprintf(fp, " Size: Unknown\n");
781         break;
782     case 0:
783         fprintf(fp, " Size: Not Populated\n");
784         break;
785     default:
786         fprintf(fp, " Size: %llu bytes\n",
787                (u_longlong_t)md.smbmd_size);
788     }

790     desc_printf(smbios_memdevice_form_desc(md.smbmd_form),
791                fp, " Form Factor: %u", md.smbmd_form);

```

```

793     if (md.smbmd_set == 0)
794         fprintf(fp, " Set: None\n");
795     else if (md.smbmd_set == (uint8_t)-1u)
796         fprintf(fp, " Set: Unknown\n");
797     else
798         fprintf(fp, " Set: %u\n", md.smbmd_set);
800     if (md.smbmd_rank != 0) {
801         desc_printf(smbios_memdevice_rank_desc(md.smbmd_rank),
802             fp, " Rank: %u", md.smbmd_rank);
803     } else {
804         fprintf(fp, " Rank: Unknown\n");
805     }
807     desc_printf(smbios_memdevice_type_desc(md.smbmd_type),
808         fp, " Memory Type: %u", md.smbmd_type);
810     flag_printf(fp, "Flags", md.smbmd_flags, sizeof (md.smbmd_flags) * NBBY,
811         smbios_memdevice_flag_name, smbios_memdevice_flag_desc);
813     if (md.smbmd_speed != 0)
814         fprintf(fp, " Speed: %u MHz\n", md.smbmd_speed);
815     else
816         fprintf(fp, " Speed: Unknown\n");
818     if (md.smbmd_clkspeed != 0)
819         fprintf(fp, " Configured Speed: %u MHz\n", md.smbmd_clkspeed);
820     else
821         fprintf(fp, " Configured Speed: Unknown\n");
823     fprintf(fp, " Device Locator: %s\n", md.smbmd_dloc);
824     fprintf(fp, " Bank Locator: %s\n", md.smbmd_bloc);
826     if (md.smbmd_minvolt != 0) {
827         fprintf(fp, " Minimum Voltage: %.2fV\n",
828             md.smbmd_minvolt / 1000.0);
829     } else {
830         fprintf(fp, " Minimum Voltage: Unknown\n");
831     }
833     if (md.smbmd_maxvolt != 0) {
834         fprintf(fp, " Maximum Voltage: %.2fV\n",
835             md.smbmd_maxvolt / 1000.0);
836     } else {
837         fprintf(fp, " Maximum Voltage: Unknown\n");
838     }
840     if (md.smbmd_confvolt != 0) {
841         fprintf(fp, " Configured Voltage: %.2fV\n",
842             md.smbmd_confvolt / 1000.0);
843     } else {
844         fprintf(fp, " Configured Voltage: Unknown\n");
845     }
846 }

```

_____unchanged_portion_omitted_____

```

1023 static int
1024 print_struct(smbios_hdl_t *shp, const smbios_struct_t *sp, void *fp)
1025 {
1026     smbios_info_t info;
1027     int hex = opt_x;
1028     const char *s;
1030     if (opt_t != -1 && opt_t != sp->smbstr_type)

```

```

1031         return (0); /* skip struct if type doesn't match -t */
1033     if (!opt_0 && (sp->smbstr_type == SMB_TYPE_MEMCTL ||
1034         sp->smbstr_type == SMB_TYPE_MEMMOD))
1035         return (0); /* skip struct if type is obsolete */
1037     if (g_hdr++ == 0 || !opt_s)
1038         fprintf(fp, "%-5s %-4s %s\n", "ID", "SIZE", "TYPE");
1040     fprintf(fp, "%-5u %-4lu",
1041         (uint_t)sp->smbstr_id, (ulong_t)sp->smbstr_size);
1043     if ((s = smbios_type_name(sp->smbstr_type)) != NULL)
1044         fprintf(fp, " (%u) %s", sp->smbstr_type, s);
1045     else if (sp->smbstr_type > SMB_TYPE_OEM_LO &&
1046         sp->smbstr_type < SMB_TYPE_OEM_HI)
1047         fprintf(fp, " (%u) %s+%u", sp->smbstr_type, "SMB_TYPE_OEM_LO",
1048             fprintf(fp, " %s+%u", "SMB_TYPE_OEM_LO",
1049                 sp->smbstr_type - SMB_TYPE_OEM_LO);
1050     else
1051         fprintf(fp, " %u", sp->smbstr_type);
1053     if ((s = smbios_type_desc(sp->smbstr_type)) != NULL)
1054         fprintf(fp, " (%s)\n", s);
1055     else
1056         fprintf(fp, "\n");
1058     if (opt_s)
1059         return (0); /* only print header line if -s specified */
1061     if (smbios_info_common(shp, sp->smbstr_id, &info) == 0) {
1062         fprintf(fp, "\n");
1063         print_common(&info, fp);
1065     switch (sp->smbstr_type) {
1066     case SMB_TYPE_BIOS:
1067         fprintf(fp, "\n");
1068         print_bios(shp, fp);
1069         break;
1070     case SMB_TYPE_SYSTEM:
1071         fprintf(fp, "\n");
1072         print_system(shp, fp);
1073         break;
1074     case SMB_TYPE_BASEBOARD:
1075         fprintf(fp, "\n");
1076         print_bboard(shp, sp->smbstr_id, fp);
1077         break;
1078     case SMB_TYPE_CHASSIS:
1079         fprintf(fp, "\n");
1080         print_chassis(shp, sp->smbstr_id, fp);
1081         break;
1082     case SMB_TYPE_PROCESSOR:
1083         fprintf(fp, "\n");
1084         print_processor(shp, sp->smbstr_id, fp);
1085         break;
1086     case SMB_TYPE_CACHE:
1087         fprintf(fp, "\n");
1088         print_cache(shp, sp->smbstr_id, fp);
1089         break;
1090     case SMB_TYPE_PORT:
1091         fprintf(fp, "\n");
1092         print_port(shp, sp->smbstr_id, fp);
1093         break;
1094     case SMB_TYPE_SLOT:

```

```

1095         oprintf(fp, "\n");
1096         print_slot(shp, sp->smbstr_id, fp);
1097         break;
1098     case SMB_TYPE_OBDEVS:
1099         oprintf(fp, "\n");
1100         print_obdevs(shp, sp->smbstr_id, fp);
1101         break;
1102     case SMB_TYPE_OEMSTR:
1103     case SMB_TYPE_SYSCONFSTR:
1104         oprintf(fp, "\n");
1105         print_strtab(shp, sp->smbstr_id, fp);
1106         break;
1107     case SMB_TYPE_LANG:
1108         oprintf(fp, "\n");
1109         print_lang(shp, sp->smbstr_id, fp);
1110         break;
1111     case SMB_TYPE_EVENTLOG:
1112         oprintf(fp, "\n");
1113         print_evlog(shp, sp->smbstr_id, fp);
1114         break;
1115     case SMB_TYPE_MEMARRAY:
1116         oprintf(fp, "\n");
1117         print_memarray(shp, sp->smbstr_id, fp);
1118         break;
1119     case SMB_TYPE_MEMDEVICE:
1120         oprintf(fp, "\n");
1121         print_memdevice(shp, sp->smbstr_id, fp);
1122         break;
1123     case SMB_TYPE_MEMARRAYMAP:
1124         oprintf(fp, "\n");
1125         print_memarrmap(shp, sp->smbstr_id, fp);
1126         break;
1127     case SMB_TYPE_MEMDEVICEMAP:
1128         oprintf(fp, "\n");
1129         print_memdevmap(shp, sp->smbstr_id, fp);
1130         break;
1131     case SMB_TYPE_SECURITY:
1132         oprintf(fp, "\n");
1133         print_hwsec(shp, fp);
1134         break;
1135     case SMB_TYPE_BOOT:
1136         oprintf(fp, "\n");
1137         print_boot(shp, fp);
1138         break;
1139     case SMB_TYPE_IPMIDEV:
1140         oprintf(fp, "\n");
1141         print_ipmi(shp, fp);
1142         break;
1143     case SMB_TYPE_OBDEVEXT:
1144         oprintf(fp, "\n");
1145         print_obdevs_ext(shp, sp->smbstr_id, fp);
1146         break;
1147     case SUN_OEM_EXT_PROCESSOR:
1148         oprintf(fp, "\n");
1149         print_extprocessor(shp, sp->smbstr_id, fp);
1150         break;
1151     case SUN_OEM_EXT_PORT:
1152         oprintf(fp, "\n");
1153         print_extport(shp, sp->smbstr_id, fp);
1154         break;
1155     case SUN_OEM_PCIEEXRC:
1156         oprintf(fp, "\n");
1157         print_pcieexrc(shp, sp->smbstr_id, fp);
1158         break;
1159     case SUN_OEM_EXT_MEMARRAY:
1160         oprintf(fp, "\n");

```

```

1161         print_extmemarray(shp, sp->smbstr_id, fp);
1162         break;
1163     case SUN_OEM_EXT_MEMDEVICE:
1164         oprintf(fp, "\n");
1165         print_extmemdevice(shp, sp->smbstr_id, fp);
1166         break;
1167     default:
1168         hex++;
1169     }
1171     if (hex)
1172         print_bytes(sp->smbstr_data, sp->smbstr_size, fp);
1173     else
1174         oprintf(fp, "\n");
1176     return (0);
1177 }

```

unchanged_portion_omitted

new/usr/src/common/smbios/mktables.sh

1

```
*****
5504 Thu Mar 26 17:12:00 2015
new/usr/src/common/smbios/mktables.sh
5094 Update libbios with recent items
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
Reviewed by: Garrett D'Amore <garrett@damore.org>
Reviewed by: Robert Mustacchi <rm@joyent.com>
*****

1 #!/bin/sh
2 #
3 # CDDL HEADER START
4 #
5 # The contents of this file are subject to the terms of the
6 # Common Development and Distribution License, Version 1.0 only
7 # (the "License"). You may not use this file except in compliance
8 # with the License.
9 #
10 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
11 # or http://www.opensolaris.org/os/licensing.
12 # See the License for the specific language governing permissions
13 # and limitations under the License.
14 #
15 # When distributing Covered Code, include this CDDL HEADER in each
16 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
17 # If applicable, add the following below this CDDL HEADER, with the
18 # fields enclosed by brackets "[]" replaced with your own identifying
19 # information: Portions Copyright [yyyy] [name of copyright owner]
20 #
21 # CDDL HEADER END
22 #
23 #
24 # Copyright 2015 OmniTI Computer Consulting, Inc. All rights reserved.
25 # Copyright 2005 Sun Microsystems, Inc. All rights reserved.
26 # Use is subject to license terms.
27 #
28 #ident "%Z%M% %I% %E% SMI"

29 #
30 # The SMBIOS interfaces defined in <sys/smbios.h> include a set of integer-to-
31 # string conversion routines for the various constants defined in the SMBIOS
32 # spec. These functions are used by smbios(1M) and prtdiag(1M) and can be
33 # leveraged by other clients as well. To simplify maintenance of the source
34 # base, this shell script automatically generates the source code for all of
35 # these functions from the <sys/smbios.h> header file and its comments. Each
36 # set of constants should be given a unique #define prefix, listed in the
37 # tables below. The smbios_*_name() functions return the identifier of the
38 # cpp define, and the smbios_*_desc() functions return the text of the comment.
39 #

41 name_funcs='
42 SMB_BBFL_      smbios_bboard_flag_name      uint_t
43 SMB_BIOSFL_    smbios_bios_flag_name         uint64_t
44 SMB_BIOSXB1_   smbios_bios_xb1_name          uint_t
45 SMB_BIOSXB2_   smbios_bios_xb2_name          uint_t
46 SMB_CAT_       smbios_cache_ctype_name       uint_t
47 SMB_CAF_       smbios_cache_flag_name        uint_t
48 SMB_EVFL_      smbios_evlog_flag_name        uint_t
49 SMB_IPMI_F_    smbios_ipmi_flag_name         uint_t
50 SMB_MDF_       smbios_memdevice_flag_name    uint_t
51 SMB_PRC_       smbios_processor_core_flag_name uint_t
52 SMB_TYPE_      smbios_type_name              uint_t
53 SMB_SLCH1_     smbios_slot_ch1_name          uint_t
54 SMB_SLCH2_     smbios_slot_ch2_name          uint_t
55 '
```

new/usr/src/common/smbios/mktables.sh

2

```
57 desc_funcs='
58 SMB_BBFL_      smbios_bboard_flag_desc      uint_t
59 SMB_BBT_       smbios_bboard_type_desc       uint_t
60 SMB_BIOSFL_    smbios_bios_flag_desc        uint64_t
61 SMB_BIOSXB1_   smbios_bios_xb1_desc          uint_t
62 SMB_BIOSXB2_   smbios_bios_xb2_desc          uint_t
63 SMB_BOOT_      smbios_boot_desc             uint_t
64 SMB_CAA_       smbios_cache_assoc_desc       uint_t
65 SMB_CAT_       smbios_cache_ctype_desc       uint_t
66 SMB_CAE_       smbios_cache_ecc_desc         uint_t
67 SMB_CAF_       smbios_cache_flag_desc        uint_t
68 SMB_CAL_       smbios_cache_loc_desc         uint_t
69 SMB_CAG_       smbios_cache_logical_desc     uint_t
70 SMB_CAM_       smbios_cache_mode_desc        uint_t
71 SMB_CHST_      smbios_chassis_state_desc     uint_t
72 SMB_CHT_       smbios_chassis_type_desc      uint_t
73 SMB_EVFL_      smbios_evlog_flag_desc        uint_t
74 SMB_EVHF_      smbios_evlog_format_desc      uint_t
75 SMB_EVM_       smbios_evlog_method_desc      uint_t
76 SMB_HWSEC_PS_  smbios_hwsec_desc            uint_t
77 SMB_IPMI_F_    smbios_ipmi_flag_desc        uint_t
78 SMB_IPMI_T_    smbios_ipmi_type_desc        uint_t
79 SMB_MAL_       smbios_memarray_loc_desc      uint_t
80 SMB_MAU_       smbios_memarray_use_desc      uint_t
81 SMB_MAE_       smbios_memarray_ecc_desc      uint_t
82 SMB_MDF_       smbios_memdevice_flag_desc    uint_t
83 SMB_MDF_       smbios_memdevice_form_desc    uint_t
84 SMB_MDT_       smbios_memdevice_type_desc    uint_t
85 SMB_MDR_       smbios_memdevice_rank_desc    uint_t
86 SMB_POC_       smbios_port_conn_desc         uint_t
87 SMB_POT_       smbios_port_type_desc         uint_t
88 SMB_PRC_       smbios_processor_core_flag_desc uint_t
89 SMB_PR_        smbios_processor_family_desc  uint_t
90 SMB_PR_        smbios_processor_status_desc  uint_t
91 SMB_PRT_       smbios_processor_type_desc    uint_t
92 SMB_PRU_       smbios_processor_upgrade_desc uint_t
93 SMB_SLCH1_     smbios_slot_ch1_desc          uint_t
94 SMB_SLCH2_     smbios_slot_ch2_desc          uint_t
95 SMB_SL_        smbios_slot_length_desc       uint_t
96 SMB_SLT_       smbios_slot_type_desc         uint_t
97 SMB_SLU_       smbios_slot_usage_desc        uint_t
98 SMB_SLW_       smbios_slot_width_desc        uint_t
99 SMB_TYPE_      smbios_type_desc             uint_t
100 SMB_WAKEUP_    smbios_system_wakeup_desc     uint_t
101 '

103 if [ $# -ne 1 ]; then
104     echo "Usage: $0 file.h > file.c" >&2
105     exit 2
106 fi

108 echo "\
109 /*\n\
110  * Copyright 2015 OmniTI Computer Consulting, Inc. All rights reserved.\n\
111  * Copyright 2005 Sun Microsystems, Inc. All rights reserved.\n\
112  * Use is subject to license terms.\n\
113  */\n\
114 \n\
115 #pragma ident\t\"%Z%M%\t%I%\t%E% SMI\"\n\
116 \n\
117 #include <smbios.h>"

118 echo "$name_funcs" | while read p name type; do
119     [ -z "$p" ] && continue
120     pattern="^#define[ ]*\($p[A-Za-z0-9_]*\) [A-Z0-9]*.*$"
121     replace='    case \1: return ("\"$name_funcs\"");'
```

```
122      echo "\nconst char *\n$name($type x)\n{\n\tswitch (x) {"
123      sed -n "s@$pattern@$replace@p" < $1 || exit 1
124      echo "\t}\n\treturn (NULL);\n}"
125 done

127 #
128 # Generate the description functions based on the comment next to a #define.
129 # The transformations for descriptive comments are slightly more complicated
130 # than those used for the identifier->name functions above:
131 #
132 # (1) strip any [R0] suffix from the comment (a header file convention)
133 # (2) replace any " with \" so it is escaped for the final output string
134 # (3) replace return (...); with return (...); to finish the code
135 #
136 echo "$desc_funcs" | while read p name type; do
137     [ -z "$p" ] && continue
138     pattern="^#define[ ]\($p[A-Za-z0-9_]*\)[" ]*.*\/\\* \\.*\) \\*/$"
139     replace='      case \1: return (\2);'

141     echo "\nconst char *\n$name($type x)\n{\n\tswitch (x) {"
142     sed -n "s@$pattern@$replace@p" < $1 | sed 's/ ([R0])//)' | \
143         sed 's/"/\\"/g' | sed 's/(/(/";s/);/");/' || exit 1
144     echo "\t}\n\treturn (NULL);\n}"
145 done

147 exit 0
```

new/usr/src/common/smbios/smb_info.c

1

```
*****
31885 Thu Mar 26 17:12:00 2015
new/usr/src/common/smbios/smb_info.c
5094 Update libmbios with recent items
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
Reviewed by: Garrett D'Amore <garrett@damore.org>
Reviewed by: Robert Mustacchi <rm@joyent.com>
*****

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

22 /*
23  * Copyright 2015 OmniTI Computer Consulting, Inc. All rights reserved.
24  * Copyright 2010 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26 */

28 /*
29  * SMBIOS Information Routines
30  *
31  * The routines in this file are used to convert from the SMBIOS data format to
32  * a more reasonable and stable set of structures offered as part of our ABI.
33  * These functions take the general form:
34  *
35  *     stp = smb_lookup_type(shp, foo);
36  *     smb_foo_t foo;
37  *
38  *     smb_info_bcopy(stp->smbst_hdr, &foo, sizeof (foo));
39  *     bzero(caller's struct);
40  *
41  *     copy/convert foo members into caller's struct
42  *
43  * We copy the internal structure on to an automatic variable so as to avoid
44  * checks everywhere for structures that the BIOS has improperly truncated, and
45  * also to automatically handle the case of a structure that has been extended.
46  * When necessary, this code can use smb_gteq() to determine whether the SMBIOS
47  * data is of a particular revision that is supposed to contain a new field.
48 */

50 #include <sys/smbios_impl.h>

52 #ifdef _KERNEL
53 #include <sys/sunddi.h>
54 #else
55 #include <fcntl.h>
56 #include <unistd.h>
57 #include <string.h>
```

new/usr/src/common/smbios/smb_info.c

2

```
58 #endif

60 /*
61  * A large number of SMBIOS structures contain a set of common strings used to
62  * describe a h/w component's serial number, manufacturer, etc. These fields
63  * helpfully have different names and offsets and sometimes aren't consistent.
64  * To simplify life for our clients, we factor these common things out into
65  * smbinfo_t, which can be retrieved for any structure. The following
66  * table describes the mapping from a given structure to the smbinfo_t.
67  * Multiple SMBIOS structures' contained objects are also handled here.
68 */
69 static const struct smb_infospec {
70     uint8_t is_type;           /* structure type */
71     uint8_t is_manu;          /* manufacturer offset */
72     uint8_t is_product;       /* product name offset */
73     uint8_t is_version;       /* version offset */
74     uint8_t is_serial;        /* serial number offset */
75     uint8_t is_asset;         /* asset tag offset */
76     uint8_t is_location;      /* location string offset */
77     uint8_t is_part;          /* part number offset */
78     uint8_t is_contc;         /* contained count */
79     uint8_t is_contsz;        /* contained size */
80     uint8_t is_contv;         /* contained objects */
81 } _smb_infospecs[] = {
    unchanged_portion_omitted

411 int
412 smbinfo_chassis(smbios_hdl_t *shp, id_t id, smbinfo_chassis_t *chp)
413 {
414     const smb_struct_t *stp = smb_lookup_id(shp, id);
415     /* Length is measurable by one byte, so it'll be no more than 255. */
416     uint8_t buf[256];
417     smbinfo_chassis_t *ch = (smbinfo_chassis_t *)&buf[0];
418     smbinfo_chassis_t ch;

419     if (stp == NULL)
420         return (-1); /* errno is set for us */

422     if (stp->smbst_hdr->smbh_type != SMB_TYPE_CHASSIS)
423         return (smb_set_errno(shp, ESMB_TYPE));

425     smbinfo_bcopy(stp->smbst_hdr, ch, sizeof (buf));
426     smbinfo_bcopy(stp->smbst_hdr, &ch, sizeof (ch));
427     bzero(chp, sizeof (smbinfo_chassis_t));

428     chp->smbc_oemdata = ch->smbch_oemdata;
429     chp->smbc_lock = (ch->smbch_type & SMB_CHT_LOCK) != 0;
430     chp->smbc_type = ch->smbch_type & ~SMB_CHT_LOCK;
431     chp->smbc_bustate = ch->smbch_bustate;
432     chp->smbc_pstate = ch->smbch_pstate;
433     chp->smbc_thstate = ch->smbch_thstate;
434     chp->smbc_security = ch->smbch_security;
435     chp->smbc_uheight = ch->smbch_uheight;
436     chp->smbc_cords = ch->smbch_cords;
437     chp->smbc_elems = ch->smbch_cn;
438     chp->smbc_elenlen = ch->smbch_cm;
439     chp->smbc_oemdata = ch->smbch_oemdata;
440     chp->smbc_lock = (ch->smbch_type & SMB_CHT_LOCK) != 0;
441     chp->smbc_type = ch->smbch_type & ~SMB_CHT_LOCK;
442     chp->smbc_bustate = ch->smbch_bustate;
443     chp->smbc_pstate = ch->smbch_pstate;
444     chp->smbc_thstate = ch->smbch_thstate;
445     chp->smbc_security = ch->smbch_security;
446     chp->smbc_uheight = ch->smbch_uheight;
447     chp->smbc_cords = ch->smbch_cords;
448     chp->smbc_elems = ch->smbch_cn;
```

```

435     chp->smbc_elemlen = ch.smbch_cm;

440     if (shp->sh_smbvers >= SMB_VERSION_27) {
441         (void) strcpy(chp->smbc_sku, SMB_CH_SKU(ch),
442             sizeof (chp->smbc_sku));
443     }

445     return (0);
446 }

448 int
449 smbios_info_processor(smbios_hdl_t *shp, id_t id, smbios_processor_t *pp)
450 {
451     const smb_struct_t *stp = smb_lookup_id(shp, id);
452     smb_processor_t p;

454     if (stp == NULL)
455         return (-1); /* errno is set for us */

457     if (stp->smbst_hdr->smbh_type != SMB_TYPE_PROCESSOR)
458         return (smb_set_errno(shp, ESMB_TYPE));

460     smb_info_bcopy(stp->smbst_hdr, &p, sizeof (p));
461     bzero(pp, sizeof (smbios_processor_t));

463     pp->smbp_cpuid = p.smbpr_cpuid;
464     pp->smbp_type = p.smbpr_type;
465     pp->smbp_family = p.smbpr_family;
466     pp->smbp_voltage = p.smbpr_voltage;
467     pp->smbp_maxspeed = p.smbpr_maxspeed;
468     pp->smbp_curspeed = p.smbpr_curspeed;
469     pp->smbp_status = p.smbpr_status;
470     pp->smbp_upgrade = p.smbpr_upgrade;
471     pp->smbp_l1cache = p.smbpr_l1cache;
472     pp->smbp_l2cache = p.smbpr_l2cache;
473     pp->smbp_l3cache = p.smbpr_l3cache;

475     if (shp->sh_smbvers >= SMB_VERSION_25) {
476         pp->smbp_corecount = p.smbpr_corecount;
477         pp->smbp_coresenabled = p.smbpr_coresenabled;
478         pp->smbp_threadcount = p.smbpr_threadcount;
479         pp->smbp_cflags = p.smbpr_cflags;
480     }

482     if (shp->sh_smbvers >= SMB_VERSION_26)
483         pp->smbp_family2 = p.smbpr_family2;

485     return (0);
486 }
    unchanged_portion_omitted

716 int
717 smbios_info_memarray(smbios_hdl_t *shp, id_t id, smbios_memarray_t *map)
718 {
719     const smb_struct_t *stp = smb_lookup_id(shp, id);
720     smb_memarray_t m;

722     if (stp == NULL)
723         return (-1); /* errno is set for us */

725     if (stp->smbst_hdr->smbh_type != SMB_TYPE_MEMARRAY)
726         return (smb_set_errno(shp, ESMB_TYPE));

728     smb_info_bcopy(stp->smbst_hdr, &m, sizeof (m));
729     bzero(map, sizeof (smbios_memarray_t));

```

```

731     map->smbma_location = m.smbmarr_loc;
732     map->smbma_use = m.smbmarr_use;
733     map->smbma_ecc = m.smbmarr_ecc;
734     map->smbma_ndevs = m.smbmarr_ndevs;
735     map->smbma_err = m.smbmarr_err;

737     if (m.smbmarr_cap != 0x80000000)
738         map->smbma_size = (uint64_t)m.smbmarr_cap * 1024;
739     else if (m.smbmarr_extcap != 0)
740         map->smbma_size = m.smbmarr_extcap;
741     else
742         map->smbma_size = 0; /* unknown */

744     return (0);
745 }

747 int
748 smbios_info_memarrmap(smbios_hdl_t *shp, id_t id, smbios_memarrmap_t *map)
749 {
750     const smb_struct_t *stp = smb_lookup_id(shp, id);
751     smb_memarrmap_t m;

753     if (stp == NULL)
754         return (-1); /* errno is set for us */

756     if (stp->smbst_hdr->smbh_type != SMB_TYPE_MEMARRAYMAP)
757         return (smb_set_errno(shp, ESMB_TYPE));

759     smb_info_bcopy(stp->smbst_hdr, &m, sizeof (m));
760     bzero(map, sizeof (smbios_memarrmap_t));

762     map->smbmam_array = m.smbamap_array;
763     map->smbmam_width = m.smbamap_width;

765     if (m.smbamap_start != 0xFFFFFFFF && m.smbamap_end != 0xFFFFFFFF) {
766         map->smbmam_addr = (uint64_t)m.smbamap_start * 1024;
767         map->smbmam_size = (uint64_t)
768             (m.smbamap_end - m.smbamap_start + 1) * 1024;
769     } else if (m.smbamap_extstart != 0 && m.smbamap_extend != 0) {
770         map->smbmam_addr = m.smbamap_extstart;
771         map->smbmam_size = m.smbamap_extend - m.smbamap_extstart + 1;
772     }

774     return (0);
775 }

777 int
778 smbios_info_memdevice(smbios_hdl_t *shp, id_t id, smbios_memdevice_t *mdp)
779 {
780     const smb_struct_t *stp = smb_lookup_id(shp, id);
781     smb_memdevice_t m;

783     if (stp == NULL)
784         return (-1); /* errno is set for us */

786     if (stp->smbst_hdr->smbh_type != SMB_TYPE_MEMDEVICE)
787         return (smb_set_errno(shp, ESMB_TYPE));

789     smb_info_bcopy(stp->smbst_hdr, &m, sizeof (m));
790     bzero(mdp, sizeof (smbios_memdevice_t));

792     mdp->smbmd_array = m.smbmdev_array;
793     mdp->smbmd_error = m.smbmdev_error;
794     mdp->smbmd_twidth = m.smbmdev_twidth == 0xFFFF ? -1U : m.smbmdev_twidth;
795     mdp->smbmd_dwidth = m.smbmdev_dwidth == 0xFFFF ? -1U : m.smbmdev_dwidth;

```

```

797     if (m.smbmdev_size == 0x7FFF) {
798         mdp->smbmd_size = (uint64_t)m.smbmdev_extsize;
799         mdp->smbmd_size *= 1024 * 1024; /* convert MB to bytes */
800     } else if (m.smbmdev_size != 0xFFFF) {
771         if (mdp->smbmd_size != 0xFFFF) {
801             mdp->smbmd_size = (uint64_t)(m.smbmdev_size & ~SMB_MDS_KBYTES);
802             if (m.smbmdev_size & SMB_MDS_KBYTES)
803                 mdp->smbmd_size *= 1024;
804             else
805                 mdp->smbmd_size *= 1024 * 1024;
806         } else
807             mdp->smbmd_size = -1ULL; /* size unknown */

809     mdp->smbmd_form = m.smbmdev_form;
810     mdp->smbmd_set = m.smbmdev_set;
811     mdp->smbmd_type = m.smbmdev_type;
812     mdp->smbmd_speed = m.smbmdev_speed;
813     mdp->smbmd_flags = m.smbmdev_flags;
814     mdp->smbmd_dloc = smb_strptr(stp, m.smbmdev_dloc);
815     mdp->smbmd_bloc = smb_strptr(stp, m.smbmdev_bloc);

817     if (shp->sh_smbvers >= SMB_VERSION_26)
818         mdp->smbmd_rank = m.smbmdev_attrs & 0x0F;
787     if (m.smbmdev_speed != 0)
788         mdp->smbmd_speed = 1000 / m.smbmdev_speed; /* MHz -> nsec */

820     if (shp->sh_smbvers >= SMB_VERSION_27)
821         mdp->smbmd_clkspeed = m.smbmdev_clkspeed;

823     if (shp->sh_smbvers >= SMB_VERSION_28) {
824         mdp->smbmd_minvolt = m.smbmdev_minvolt;
825         mdp->smbmd_maxvolt = m.smbmdev_maxvolt;
826         mdp->smbmd_confvolt = m.smbmdev_confvolt;
827     }

829     return (0);
830 }

832 int
833 smbios_info_memdevmap(smbios_hdl_t *shp, id_t id, smbios_memdevmap_t *mdp)
834 {
835     const smb_struct_t *stp = smb_lookup_id(shp, id);
836     smb_memdevmap_t m;

838     if (stp == NULL)
839         return (-1); /* errno is set for us */

841     if (stp->smbst_hdr->smbh_type != SMB_TYPE_MEMDEVICEMAP)
842         return (smb_set_errno(shp, ESMB_TYPE));

844     smb_info_bcopy(stp->smbst_hdr, &m, sizeof (m));
845     bzero(mdp, sizeof (smbios_memdevmap_t));

847     mdp->smbmdm_device = m.smbdmap_device;
848     mdp->smbmdm_arrmap = m.smbdmap_array;
810     mdp->smbmdm_addr = (uint64_t)m.smbdmap_start * 1024;
811     mdp->smbmdm_size = (uint64_t)
812         (m.smbdmap_end - m.smbdmap_start + 1) * 1024;
849     mdp->smbmdm_rpos = m.smbdmap_rpos;
850     mdp->smbmdm_ipos = m.smbdmap_ipos;
851     mdp->smbmdm_iddepth = m.smbdmap_iddepth;

853     if (m.smbdmap_start != 0xFFFFFFFF && m.smbdmap_end != 0xFFFFFFFF) {
854         mdp->smbmdm_addr = (uint64_t)m.smbdmap_start * 1024;
855         mdp->smbmdm_size = (uint64_t)
856             (m.smbdmap_end - m.smbdmap_start + 1) * 1024;

```

```

857     } else if (m.smbdmap_extstart != 0 && m.smbdmap_extend != 0) {
858         mdp->smbmdm_addr = m.smbdmap_extstart;
859         mdp->smbmdm_size = m.smbdmap_extend - m.smbdmap_extstart + 1;
860     }

862     return (0);
863 }

```

unchanged_portion_omitted

```

*****
10502 Thu Mar 26 17:12:00 2015
new/usr/src/common/smbios/smb_open.c
5094 Update libbios with recent items
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
Reviewed by: Garrett D'Amore <garrett@damore.org>
Reviewed by: Robert Mustacchi <rm@joyent.com>
*****

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

22 /*
23  * Copyright 2015 OmniTI Computer Consulting, Inc. All rights reserved.
24  * Copyright 2009 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26 */

28 #include <sys/smbios_impl.h>

30 static const uint_t _smb_hashlen = 64;          /* hash length (must be Pof2) */
31 static const char _smb_emptystr[] = "";         /* empty string to return */
32 int _smb_debug = 0;                             /* default debug mode */

34 /*
35  * Strip out identification information for you privacy weenies. This is quite
36  * simple using our smbios_info_common() abstraction: we just locate any serial
37  * numbers and asset tags for each record, and then zero out those strings.
38  * Then we must handle two special cases: SMB_TYPE_SYSTEM holds a 16-byte UUID
39  * and SMB_TYPE_BATTERY stores a Smart Battery Data Spec 16-bit serial number.
40  * We use a literal '0' rather than '\0' for zeroing strings because \0\0 in
41  * the SMBIOS string table has a special meaning (denotes end-of-record).
42  */
43 static void
44 smb_strip(smbios_hdl_t *shp)
45 {
46     uint_t i;

48     for (i = 0; i < shp->sh_nstructs; i++) {
49         const smb_header_t *hp = shp->sh_structs[i].smbst_hdr;
50         smbios_info_t info;
51         char *p;

53         if (hp->smbh_type == SMB_TYPE_SYSTEM &&
54             hp->smbh_len >= offsetof(smb_system_t, smbsi_wakeup)) {
55             smb_system_t *sp = (smb_system_t *) (uintptr_t) hp;
56             bzero(sp->smbsi_uuid, sizeof(sp->smbsi_uuid));
57         }

```

```

59         if (hp->smbh_type == SMB_TYPE_BATTERY &&
60             hp->smbh_len >= offsetof(smb_battery_t, smbbat_sdate)) {
61             smbios_battery_t *bp = (smb_battery_t *) (uintptr_t) hp;
62             bp->smbbat_ssn = 0;
63         }

65         if (smbios_info_common(shp, hp->smbh_hdl, &info) != SMB_ERR) {
66             for (p = (char *) info.smbi_serial; *p != '\0'; p++)
67                 *p = '0';
68             for (p = (char *) info.smbi_asset; *p != '\0'; p++)
69                 *p = '0';
70         }
71     }
72 }

74 smbios_hdl_t *
75 smbios_bufopen(const smbios_entry_t *ep, const void *buf, size_t len,
76                int version, int flags, int *errp)
77 {
78     smbios_hdl_t *shp = smb_zalloc(sizeof(smbios_hdl_t));
79     const smb_header_t *hp, *nhp;
80     const uchar_t *p, *q, *s;
81     uint_t i, h;

83     switch (version) {
84     case SMB_VERSION_23:
85     case SMB_VERSION_24:
86     case SMB_VERSION_25:
87     case SMB_VERSION_26:
88     case SMB_VERSION_27:
89     case SMB_VERSION_28:
90         break;
91     default:
92         return (smb_open_error(shp, errp, ESMB_VERSION));
93     }

95     if (ep == NULL || buf == NULL || len == 0 || (flags & ~SMB_O_MASK))
96         return (smb_open_error(shp, errp, ESMB_INVALID));

98     if (shp == NULL)
99         return (smb_open_error(shp, errp, ESMB_NOMEM));

101     if (_smb_debug)
102         shp->sh_flags |= SMB_FL_DEBUG;

104     if (strcmp(ep->smbe_eanchor, SMB_ENTRY_EANCHOR, SMB_ENTRY_EANCHORLEN))
105         return (smb_open_error(shp, errp, ESMB_HEADER));

107     if (strcmp(ep->smbe_ianchor, SMB_ENTRY_IANCHOR, SMB_ENTRY_IANCHORLEN))
108         return (smb_open_error(shp, errp, ESMB_HEADER));

110     smb_dprintf(shp, "opening SMBIOS version %u.%u bcdrev 0x%x\n",
111                ep->smbe_major, ep->smbe_minor, ep->smbe_bcdrev);

113     if (!(flags & SMB_O_NOVERS)) {
114         if (ep->smbe_major > SMB_MAJOR(SMB_VERSION))
115             return (smb_open_error(shp, errp, ESMB_NEW));

117         if (ep->smbe_major < SMB_MAJOR(SMB_VERSION_23) || (
118             ep->smbe_major == SMB_MAJOR(SMB_VERSION_23) &&
119             ep->smbe_minor < SMB_MINOR(SMB_VERSION_23)))
120             return (smb_open_error(shp, errp, ESMB_OLD));
121     }

123     if (len < sizeof(smb_header_t) ||

```

```

124     ep->smbe_stlen < sizeof (smb_header_t) || len < ep->smbe_stlen)
125     return (smb_open_error(shp, errp, ESMB_SHORT));

127     if (!(flags & SMB_O_NOCKSUM)) {
128         uint8_t esum = 0, isum = 0;
129         q = (uchar_t *)ep;

131         for (p = q; p < q + ep->smbe_elen; p++)
132             esum += *p;

134         for (p = (uchar_t *)ep->smbe_ianchor; p < q + sizeof (*ep); p++)
135             isum += *p;

137         if (esum != 0 || isum != 0) {
138             smb_dprintf(shp, "bad cksum: e=%x i=%x\n", esum, isum);
139             return (smb_open_error(shp, errp, ESMB_CKSUM));
140         }
141     }

143     /*
144     * Copy the entry point into our handle. The underlying entry point
145     * may be larger than our structure definition, so reset smbe_elen
146     * to our internal size and recompute good checksums for our copy.
147     */
148     bcopy(ep, &shp->sh_ent, sizeof (smbios_entry_t));
149     shp->sh_ent.smbe_elen = sizeof (smbios_entry_t);
150     smbios_checksum(shp, &shp->sh_ent);

152     shp->sh_buf = buf;
153     shp->sh_buflen = len;
154     shp->sh_structs = smb_alloc(sizeof (smb_struct_t) * ep->smbe_stnum);
155     shp->sh_nstructs = 0;
156     shp->sh_hashlen = _smb_hashlen;
157     shp->sh_hash = smb_zalloc(sizeof (smb_struct_t *) * shp->sh_hashlen);
158     shp->sh_libvers = version;
159     shp->sh_smbvers = SMB_MAJMIN(ep->smbe_major, ep->smbe_minor);

161     if (shp->sh_structs == NULL || shp->sh_hash == NULL)
162         return (smb_open_error(shp, errp, ESMB_NOMEM));

164     hp = shp->sh_buf;
165     q = (const uchar_t *)buf + MIN(ep->smbe_stlen, len);

167     for (i = 0; i < ep->smbe_stnum; i++, hp = nhp) {
168         smb_struct_t *stp = &shp->sh_structs[i];
169         uint_t n = 0;

171         if ((const uchar_t *)hp + sizeof (smb_header_t) > q)
172             return (smb_open_error(shp, errp, ESMB_CORRUPT));

174         smb_dprintf(shp, "struct [%u] type %u len %u hdl %u at %p\n",
175             i, hp->smbh_type, hp->smbh_len, hp->smbh_hdl, (void *)hp);

177         if (hp->smbh_type == SMB_TYPE_EOT)
178             break; /* ignore any entries beyond end-of-table */

180         if ((const uchar_t *)hp + hp->smbh_len > q - 2)
181             return (smb_open_error(shp, errp, ESMB_CORRUPT));

183         h = hp->smbh_hdl & (shp->sh_hashlen - 1);
184         p = s = (const uchar_t *)hp + hp->smbh_len;

186         while (p <= q - 2 && (p[0] != '\0' || p[1] != '\0')) {
187             if (*p++ == '\0')
188                 n++; /* count strings until \0\0 delimiter */
189         }

```

```

191     if (p > q - 2)
192         return (smb_open_error(shp, errp, ESMB_CORRUPT));

194     if (p > s)
195         n++; /* add one for final string in string table */

197     stp->smbst_hdr = hp;
198     stp->smbst_str = s;
199     stp->smbst_end = p;
200     stp->smbst_next = shp->sh_hash[h];
201     stp->smbst_strtab = smb_alloc(sizeof (uint16_t) * n);
202     stp->smbst_strtablen = n;

204     if (n != 0 && stp->smbst_strtab == NULL)
205         return (smb_open_error(shp, errp, ESMB_NOMEM));

207     shp->sh_hash[h] = stp;
208     nhp = (void *) (p + 2);
209     shp->sh_nstructs++;

211     for (n = 0, p = s; n < stp->smbst_strtablen; p++) {
212         if (*p == '\0') {
213             stp->smbst_strtab[n++] =
214                 (uint16_t)(s - stp->smbst_str);
215             s = p + 1;
216         }
217     }
218 }

220     if (flags & SMB_O_ZIDS)
221         smb_strip(shp);

223     return (shp);
224 }

```

unchanged_portion_omitted

```

*****
3691 Thu Mar 26 17:12:00 2015
new/usr/src/lib/libsmbios/common/mapfile-vers
5094 Update libsmbios with recent items
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
Reviewed by: Garrett D'Amore <garrett@damore.org>
Reviewed by: Robert Mustacchi <rm@joyent.com>
*****
1 #
2 # CDDL HEADER START
3 #
4 # The contents of this file are subject to the terms of the
5 # Common Development and Distribution License (the "License").
6 # You may not use this file except in compliance with the License.
7 #
8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9 # or http://www.opensolaris.org/os/licensing.
10 # See the License for the specific language governing permissions
11 # and limitations under the License.
12 #
13 # When distributing Covered Code, include this CDDL HEADER in each
14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 # If applicable, add the following below this CDDL HEADER, with the
16 # fields enclosed by brackets "[]" replaced with your own identifying
17 # information: Portions Copyright [yyyy] [name of copyright owner]
18 #
19 # CDDL HEADER END
20 #
21 #
22 # Copyright 2015 OmniTI Computer Consulting, Inc. All rights reserved.
23 # Copyright (c) 2006, 2010, Oracle and/or its affiliates. All rights reserved.
24 #
25 #
26 #
27 # MAPFILE HEADER START
28 #
29 # WARNING: STOP NOW. DO NOT MODIFY THIS FILE.
30 # Object versioning must comply with the rules detailed in
31 #
32 #     usr/src/lib/README.mapfiles
33 #
34 # You should not be making modifications here until you've read the most current
35 # copy of that file. If you need help, contact a gatekeeper for guidance.
36 #
37 # MAPFILE HEADER END
38 #
39 #
40 $mapfile_version 2
41 #
42 SYMBOL_VERSION SUNWprivate_1.1 {
43     global:
44         _smb_debug;
45         smbios_bboard_flag_desc;
46         smbios_bboard_flag_name;
47         smbios_bboard_type_desc;
48         smbios_bios_flag_desc;
49         smbios_bios_flag_name;
50         smbios_bios_xb1_desc;
51         smbios_bios_xb1_name;
52         smbios_bios_xb2_desc;
53         smbios_bios_xb2_name;
54         smbios_boot_desc;
55         smbios_buf;
56         smbios_buflen;
57         smbios_bufopen;

```

```

58         smbios_cache_assoc_desc;
59         smbios_cache_ctype_desc;
60         smbios_cache_ctype_name;
61         smbios_cache_ecc_desc;
62         smbios_cache_flag_desc;
63         smbios_cache_flag_name;
64         smbios_cache_loc_desc;
65         smbios_cache_logical_desc;
66         smbios_cache_mode_desc;
67         smbios_chassis_state_desc;
68         smbios_chassis_type_desc;
69         smbios_checksum;
70         smbios_close;
71         smbios_csn;
72         smbios_errmsg;
73         smbios_errno;
74         smbios_evlog_flag_desc;
75         smbios_evlog_flag_name;
76         smbios_evlog_format_desc;
77         smbios_evlog_method_desc;
78         smbios_fdopen;
79         smbios_hwsec_desc;
80         smbios_info_bboard;
81         smbios_info_bios;
82         smbios_info_boot;
83         smbios_info_cache;
84         smbios_info_chassis;
85         smbios_info_common;
86         smbios_info_contains;
87         smbios_info_eventlog;
88         smbios_info_hwsec;
89         smbios_info_ipmi;
90         smbios_info_lang;
91         smbios_info_memarray;
92         smbios_info_extmemarray;
93         smbios_info_memarrmap;
94         smbios_info_memdevice;
95         smbios_info_extmemdevice;
96         smbios_info_memdevmap;
97         smbios_info_obdevs;
98         smbios_info_obdevs_ext;
99         smbios_info_port;
100        smbios_info_extport;
101        smbios_info_processor;
102        smbios_info_extprocessor;
103        smbios_info_slot;
104        smbios_info_smbios;
105        smbios_info_strtab;
106        smbios_info_system;
107        smbios_info_pciexrc;
108        smbios_ipmi_flag_desc;
109        smbios_ipmi_flag_name;
110        smbios_ipmi_type_desc;
111        smbios_iter;
112        smbios_lookup_id;
113        smbios_lookup_type;
114        smbios_memarray_ecc_desc;
115        smbios_memarray_loc_desc;
116        smbios_memarray_use_desc;
117        smbios_memdevice_flag_desc;
118        smbios_memdevice_flag_name;
119        smbios_memdevice_form_desc;
120        smbios_memdevice_type_desc;
121        smbios_memdevice_rank_desc;
122        smbios_open;
123        smbios_port_conn_desc;

```

```
124     smbios_port_type_desc;
125     smbios_processor_family_desc;
126     smbios_processor_status_desc;
127     smbios_processor_type_desc;
128     smbios_processor_upgrade_desc;
129     smbios_processor_core_flag_desc;
130     smbios_processor_core_flag_name;
131     smbios_psn;
132     smbios_slot_ch1_desc;
133     smbios_slot_ch1_name;
134     smbios_slot_ch2_desc;
135     smbios_slot_ch2_name;
136     smbios_slot_length_desc;
137     smbios_slot_type_desc;
138     smbios_slot_usage_desc;
139     smbios_slot_width_desc;
140     smbios_system_wakeup_desc;
141     smbios_type_desc;
142     smbios_type_name;
143     smbios_write;
144     local:
145     *;
146 };
    unchanged_portion_omitted_
```

67005 Thu Mar 26 17:12:00 2015
 new/usr/src/uts/common/sys/smbios.h
 5094 Update libsmbios with recent items
 Reviewed by: Dan McDonald <danmcd@omniti.com>
 Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
 Reviewed by: Garrett D'Amore <garrett@damore.org>
 Reviewed by: Robert Mustacchi <rm@joyent.com>

```

1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License (the "License").
6  * You may not use this file except in compliance with the License.
7  *
8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
9  * or http://www.opensolaris.org/os/licensing.
10 * See the License for the specific language governing permissions
11 * and limitations under the License.
12 *
13 * When distributing Covered Code, include this CDDL HEADER in each
14 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
15 * If applicable, add the following below this CDDL HEADER, with the
16 * fields enclosed by brackets "[]" replaced with your own identifying
17 * information: Portions Copyright [yyyy] [name of copyright owner]
18 *
19 * CDDL HEADER END
20 */
21
22 /*
23  * Copyright 2015 OmniTI Computer Consulting, Inc. All rights reserved.
24  * Copyright 2010 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26 */
27
28 /*
29  * This header file defines the interfaces available from the SMBIOS access
30  * library, libsmbios, and an equivalent kernel module. This API can be used
31  * to access DMTF SMBIOS data from a device, file, or raw memory buffer.
32  * This is NOT yet a public interface, although it may eventually become one in
33  * the fullness of time after we gain more experience with the interfaces.
34  *
35  * This is NOT a Public interface, and should be considered Unstable, as it is
36  * subject to change without notice as the DMTF SMBIOS specification evolves.
37  * Therefore, be aware that any program linked with this API in this
38  * instance of illumos is almost guaranteed to break in the next release.
39  * In the meantime, be aware that any program linked with this API in this
40  * release of Solaris is almost guaranteed to break in the next release.
41  *
42  * In short, do not use this header file or these routines for any purpose.
43  */
44
45 #ifndef _SYS_SMBIOS_H
46 #define _SYS_SMBIOS_H
47
48 #include <sys/types.h>
49
50 #ifdef __cplusplus
51 extern "C" {
52 #endif
53
54 /*
55  * SMBIOS Structure Table Entry Point. See DSP0134 5.2.1 for more information.
56  * SMBIOS Structure Table Entry Point. See DSP0134 2.1.1 for more information.
57  * The structure table entry point is located by searching for the anchor.
58  */

```

```

51 */
52 #pragma pack(1)
53
54 typedef struct smbios_entry {
55     char smbe_eanchor[4]; /* anchor tag (SMB_ENTRY_EANCHOR) */
56     uint8_t smbe_eksum; /* checksum of entry point structure */
57     uint8_t smbe_elen; /* length in bytes of entry point */
58     uint8_t smbe_major; /* major version of the SMBIOS spec */
59     uint8_t smbe_minor; /* minor version of the SMBIOS spec */
60     uint16_t smbe_maxsize; /* maximum size in bytes of a struct */
61     uint8_t smbe_revision; /* entry point structure revision */
62     uint8_t smbe_format[5]; /* entry point revision-specific data */
63     char smbe_ianchor[5]; /* intermed. tag (SMB_ENTRY_IANCHOR) */
64     uint8_t smbe_icksum; /* intermed. checksum */
65     uint16_t smbe_stlen; /* length in bytes of structure table */
66     uint32_t smbe_staddr; /* physical addr of structure table */
67     uint16_t smbe_stnum; /* number of structure table entries */
68     uint8_t smbe_bcdrev; /* BCD value representing DMI version */
69 } smbios_entry_t;
70
71 #pragma pack()
72
73 #define SMB_ENTRY_EANCHOR "_SM_" /* structure table entry point anchor */
74 #define SMB_ENTRY_EANCHORLEN 4 /* length of entry point anchor */
75 #define SMB_ENTRY_IANCHOR "_DMI_" /* intermediate anchor string */
76 #define SMB_ENTRY_IANCHORLEN 5 /* length of intermediate anchor */
77 #define SMB_ENTRY_MAXLEN 255 /* maximum length of entry point */
78
79 /*
80  * Structure type codes. The comments next to each type include an (R) note to
81  * indicate a structure that is required as of SMBIOS v2.8 and an (O) note to
82  * indicate a structure that is obsolete as of SMBIOS v2.8.
83  * indicate a structure that is required as of SMBIOS v2.3 and an (O) note to
84  * indicate a structure that is obsolete as of SMBIOS v2.3.
85  */
86 #define SMB_TYPE_BIOS 0 /* BIOS information (R) */
87 #define SMB_TYPE_SYSTEM 1 /* system information (R) */
88 #define SMB_TYPE_BASEBOARD 2 /* base board */
89 #define SMB_TYPE_CHASSIS 3 /* system enclosure or chassis (R) */
90 #define SMB_TYPE_PROCESSOR 4 /* processor (R) */
91 #define SMB_TYPE_MEMCTL 5 /* memory controller (O) */
92 #define SMB_TYPE_MEMMOD 6 /* memory module (O) */
93 #define SMB_TYPE_CACHE 7 /* processor cache (R) */
94 #define SMB_TYPE_PORT 8 /* port connector */
95 #define SMB_TYPE_SLOT 9 /* upgradeable system slot (R) */
96 #define SMB_TYPE_OBDEVS 10 /* on-board devices (O) */
97 #define SMB_TYPE_OEMSTR 11 /* OEM string table */
98 #define SMB_TYPE_SYSCONFSTR 12 /* system configuration string table */
99 #define SMB_TYPE_LANG 13 /* BIOS language information */
100 #define SMB_TYPE_GROUP 14 /* group associations */
101 #define SMB_TYPE_EVENTLOG 15 /* system event log */
102 #define SMB_TYPE_MEMARRAY 16 /* physical memory array (R) */
103 #define SMB_TYPE_MEMDEVICE 17 /* memory device (R) */
104 #define SMB_TYPE_MEMERR32 18 /* 32-bit memory error information */
105 #define SMB_TYPE_MEMARRAYMAP 19 /* memory array mapped address (R) */
106 #define SMB_TYPE_MEMDEVICEMAP 20 /* memory device mapped address (R) */
107 #define SMB_TYPE_POINTDEV 21 /* built-in pointing device */
108 #define SMB_TYPE_BATTERY 22 /* portable battery */
109 #define SMB_TYPE_RESET 23 /* system reset settings */
110 #define SMB_TYPE_SECURITY 24 /* hardware security settings */
111 #define SMB_TYPE_POWERCTL 25 /* system power controls */
112 #define SMB_TYPE_VPROBE 26 /* voltage probe */
113 #define SMB_TYPE_COOLDEV 27 /* cooling device */
114 #define SMB_TYPE_TPROBE 28 /* temperature probe */

```

```

113 #define SMB_TYPE_IPROBE 29 /* current probe */
114 #define SMB_TYPE_OOBRA 30 /* out-of-band remote access facility */
115 #define SMB_TYPE_BIS 31 /* boot integrity services */
116 #define SMB_TYPE_BOOT 32 /* system boot status (R) */
117 #define SMB_TYPE_MEMERR64 33 /* 64-bit memory error information */
118 #define SMB_TYPE_MGMTDEV 34 /* management device */
119 #define SMB_TYPE_MGMTDEVCP 35 /* management device component */
120 #define SMB_TYPE_MGMTDEVDATA 36 /* management device threshold data */
121 #define SMB_TYPE_MEMCHAN 37 /* memory channel */
122 #define SMB_TYPE_IPMIDEV 38 /* IPMI device information */
123 #define SMB_TYPE_POWERUP 39 /* system power supply */
124 #define SMB_TYPE_ADDINFO 40 /* additional information */
125 #define SMB_TYPE_OBDEEXT 41 /* on-board device extended info */
126 #define SMB_TYPE_MCHI 42 /* mgmt controller host interface */
127 #define SMB_TYPE_INACTIVE 126 /* inactive table entry */
128 #define SMB_TYPE_EOT 127 /* end of table */

130 #define SMB_TYPE_OEM_LO 128 /* start of OEM-specific type range */
131 #define SUN_OEM_EXT_PROCESSOR 132 /* processor extended info */
132 #define SUN_OEM_EXT_PORT 136 /* port extended info */
133 #define SUN_OEM_PCIEXRC 138 /* PCIE RootComplex/RootPort info */
134 #define SUN_OEM_EXT_MEMARRAY 144 /* phys memory array extended info */
135 #define SUN_OEM_EXT_MEMDEVICE 145 /* memory device extended info */
136 #define SMB_TYPE_OEM_HI 256 /* end of OEM-specific type range */

138 /*
139 * OEM string indicating "Platform Resource Management Specification"
140 * compliance.
141 */
142 #define SMB_PRMS1 "SUNW-PRMS-1"

144 /*
145 * Some default values set by BIOS vendor
146 */
147 #define SMB_DEFAULT1 "To Be Filled By O.E.M."
148 #define SMB_DEFAULT2 "Not Available"

150 /*
151 * SMBIOS Common Information. These structures do not correspond to anything
152 * in the SMBIOS specification, but allow library clients to more easily read
153 * information that is frequently encoded into the various SMBIOS structures.
154 */
155 typedef struct smbios_info {
156     const char *smbi_manufacturer; /* manufacturer */
157     const char *smbi_product; /* product name */
158     const char *smbi_version; /* version */
159     const char *smbi_serial; /* serial number */
160     const char *smbi_asset; /* asset tag */
161     const char *smbi_location; /* location tag */
162     const char *smbi_part; /* part number */
163 } smbios_info_t;
164 #define smbios_info_t smbios_info_t /* unchanged portion omitted */

170 #define SMB_CONT_BYTE 1 /* contained elements are byte size */
171 #define SMB_CONT_WORD 2 /* contained elements are word size */
172 #define SMB_CONT_MAX 255 /* maximum contained objects */

174 /*
175 * SMBIOS Bios Information. See DSP0134 Section 7.1 for more information.
176 * SMBIOS Bios Information. See DSP0134 Section 3.3.1 for more information.
177 */
178 typedef struct smbios_bios {
179     const char *smbb_vendor; /* bios vendor string */
180     const char *smbb_version; /* bios version string */
181     const char *smbb_reldate; /* bios release date */

```

```

182     uint32_t smbb_segment; /* bios address segment location */
183     uint32_t smbb_romsize; /* bios rom size in bytes */
184     uint32_t smbb_runsize; /* bios image size in bytes */
185     uint64_t smbb_cflags; /* bios characteristics */
186     const uint8_t *smbb_xcflags; /* bios characteristics extensions */
187     size_t smbb_nxcflags; /* number of smbb_xcflags[] bytes */
188     smbios_version_t smbb_biosv; /* bios version */
189     smbios_version_t smbb_ecfwv; /* bios embedded ctrl f/w version */
190 } smbios_bios_t;

192 #define SMB_BIOSFL_RSVD0 0x00000001 /* reserved bit zero */
193 #define SMB_BIOSFL_RSVD1 0x00000002 /* reserved bit one */
194 #define SMB_BIOSFL_UNKNOWN 0x00000004 /* unknown */
195 #define SMB_BIOSFL_BCNOTSUP 0x00000008 /* BIOS chars not supported */
196 #define SMB_BIOSFL_ISA 0x00000010 /* ISA is supported */
197 #define SMB_BIOSFL_MCA 0x00000020 /* MCA is supported */
198 #define SMB_BIOSFL_EISA 0x00000040 /* EISA is supported */
199 #define SMB_BIOSFL_PCI 0x00000080 /* PCI is supported */
200 #define SMB_BIOSFL_PCMCIA 0x00000100 /* PCMCIA is supported */
201 #define SMB_BIOSFL_PLUGNPLAY 0x00000200 /* Plug and Play is supported */
202 #define SMB_BIOSFL_APM 0x00000400 /* APM is supported */
203 #define SMB_BIOSFL_FLASH 0x00000800 /* BIOS is Flash Upgradeable */
204 #define SMB_BIOSFL_SHADOW 0x00001000 /* BIOS shadowing is allowed */
205 #define SMB_BIOSFL_VLVESA 0x00002000 /* VL-VESA is supported */
206 #define SMB_BIOSFL_ESCD 0x00004000 /* ESCD support is available */
207 #define SMB_BIOSFL_CDBOOT 0x00008000 /* Boot from CD is supported */
208 #define SMB_BIOSFL_SELBOOT 0x00010000 /* Selectable Boot supported */
209 #define SMB_BIOSFL_ROMSOCK 0x00020000 /* BIOS ROM is socketed */
210 #define SMB_BIOSFL_PCBMBOOT 0x00040000 /* Boot from PCMCIA supported */
211 #define SMB_BIOSFL_EDD 0x00080000 /* EDD Spec is supported */
212 #define SMB_BIOSFL_NEC9800 0x00100000 /* int 0x13 NEC 9800 floppy */
213 #define SMB_BIOSFL_TOSHIBA 0x00200000 /* int 0x13 Toshiba floppy */
214 #define SMB_BIOSFL_525_360K 0x00400000 /* int 0x13 5.25" 360K floppy */
215 #define SMB_BIOSFL_525_12M 0x00800000 /* int 0x13 5.25" 1.2M floppy */
216 #define SMB_BIOSFL_35_720K 0x01000000 /* int 0x13 3.5" 720K floppy */
217 #define SMB_BIOSFL_35_288M 0x02000000 /* int 0x13 3.5" 2.88M floppy */
218 #define SMB_BIOSFL_I5_PRINT 0x04000000 /* int 0x5 print screen svcs */
219 #define SMB_BIOSFL_I9_KBD 0x08000000 /* int 0x9 8042 keyboard svcs */
220 #define SMB_BIOSFL_I14_SER 0x10000000 /* int 0x14 serial svcs */
221 #define SMB_BIOSFL_I17_PRINTER 0x20000000 /* int 0x17 printer svcs */
222 #define SMB_BIOSFL_I10_CGA 0x40000000 /* int 0x10 CGA svcs */
223 #define SMB_BIOSFL_NEC_PC98 0x80000000 /* NEC PC-98 */

225 #define SMB_BIOSXB_1 0 /* bios extension byte 1 (7.1.2.1) */
226 #define SMB_BIOSXB_2 1 /* bios extension byte 2 (7.1.2.2) */
227 #define SMB_BIOSXB_1_AGP 0 /* bios extension byte 1 (3.3.1.2.1) */
228 #define SMB_BIOSXB_2 1 /* bios extension byte 2 (3.3.1.2.2) */
229 #define SMB_BIOSXB_BIOS_MAJ 2 /* bios major version */
230 #define SMB_BIOSXB_BIOS_MIN 3 /* bios minor version */
231 #define SMB_BIOSXB_ECFW_MAJ 4 /* extended ctrl f/w major version */
232 #define SMB_BIOSXB_ECFW_MIN 5 /* extended ctrl f/w minor version */

232 #define SMB_BIOSXB1_ACPI 0x01 /* ACPI is supported */
233 #define SMB_BIOSXB1_USBL 0x02 /* USB legacy is supported */
234 #define SMB_BIOSXB1_AGP 0x04 /* AGP is supported */
235 #define SMB_BIOSXB1_I20 0x08 /* I20 boot is supported */
236 #define SMB_BIOSXB1_LS120 0x10 /* LS-120 boot is supported */
237 #define SMB_BIOSXB1_ATZIP 0x20 /* ATAPI ZIP drive boot is supported */
238 #define SMB_BIOSXB1_1394 0x40 /* 1394 boot is supported */
239 #define SMB_BIOSXB1_SMBAT 0x80 /* Smart Battery is supported */

241 #define SMB_BIOSXB2_BB00T 0x01 /* BIOS Boot Specification supported */
242 #define SMB_BIOSXB2_FKNETSVC 0x02 /* F-key Network Svc boot supported */
243 #define SMB_BIOSXB2_ETCDIST 0x04 /* Enable Targeted Content Distrib. */
244 #define SMB_BIOSXB2_UEFI 0x08 /* UEFI Specification supported */
245 #define SMB_BIOSXB2_VM 0x10 /* SMBIOS table describes a VM */

```

```

247 /*
248  * SMBIOS System Information. See DSP0134 Section 7.2 for more information.
249  * SMBIOS Bios Information. See DSP0134 Section 3.3.2 for more information.
250  * The current set of smbs_wakeup values is defined after the structure.
251 */
252 typedef struct smbios_system {
253     const uint8_t *smbs_uuid;        /* UUID byte array */
254     uint8_t smbs_uuidlen;            /* UUID byte array length */
255     uint8_t smbs_wakeup;             /* wake-up event */
256     const char *smbs_sku;            /* SKU number */
257     const char *smbs_family;         /* family */
258 } smbios_system_t;
259
260 #define SMB_WAKEUP_RSVD 0x00 /* reserved */
261 #define SMB_WAKEUP_OTHER 0x01 /* other */
262 #define SMB_WAKEUP_UNKNOWN 0x02 /* unknown */
263 #define SMB_WAKEUP_APM 0x03 /* APM timer */
264 #define SMB_WAKEUP_MODEM 0x04 /* modem ring */
265 #define SMB_WAKEUP_LAN 0x05 /* LAN remote */
266 #define SMB_WAKEUP_SWITCH 0x06 /* power switch */
267 #define SMB_WAKEUP_PCIPME 0x07 /* PCI PME# */
268 #define SMB_WAKEUP_AC 0x08 /* AC power restored */
269
270 /*
271  * SMBIOS Base Board description. See DSP0134 Section 7.3 for more
272  * SMBIOS Base Board description. See DSP0134 Section 3.3.3 for more
273  * information. smbb_flags and smbb_type definitions are below.
274 */
275 typedef struct smbios_bboard {
276     id_t smbb_chassis; /* chassis containing this board */
277     uint8_t smbb_flags; /* flags (see below) */
278     uint8_t smbb_type; /* board type (see below) */
279     uint8_t smbb_contn; /* number of contained object hlds */
280 } smbios_bboard_t;
281
282 #define SMB_BBFL_MOTHERBOARD 0x01 /* board is a motherboard */
283 #define SMB_BBFL_NEEDAUX 0x02 /* auxiliary card or daughter req'd */
284 #define SMB_BBFL_REMOVABLE 0x04 /* board is removable */
285 #define SMB_BBFL_REPLACABLE 0x08 /* board is field-replacable */
286 #define SMB_BBFL_HOTSWAP 0x10 /* board is hot-swappable */
287
288 #define SMB_BBT_UNKNOWN 0x1 /* unknown */
289 #define SMB_BBT_OTHER 0x2 /* other */
290 #define SMB_BBT_SBLADE 0x3 /* server blade */
291 #define SMB_BBT_CSWITCH 0x4 /* connectivity switch */
292 #define SMB_BBT_SMM 0x5 /* system management module */
293 #define SMB_BBT_PROC 0x6 /* processor module */
294 #define SMB_BBT_IO 0x7 /* i/o module */
295 #define SMB_BBT_MEM 0x8 /* memory module */
296 #define SMB_BBT_DAUGHTER 0x9 /* daughterboard */
297 #define SMB_BBT_MOTHER 0xA /* motherboard */
298 #define SMB_BBT_PROCMEM 0xB /* processor/memory module */
299 #define SMB_BBT_PROCIO 0xC /* processor/i/o module */
300 #define SMB_BBT_INTER 0xD /* interconnect board */
301
302 /*
303  * SMBIOS Chassis description. See DSP0134 Section 7.4 for more information.
304  * SBIOS Chassis description. See DSP0134 Section 3.3.4 for more information.
305  * We move the lock bit of the type field into smbc_lock for easier processing.
306 */
307 typedef struct smbios_chassis {
308     uint32_t smbc_oemdata; /* OEM-specific data */
309     uint8_t smbc_lock; /* lock present? */
310     uint8_t smbc_type; /* type */
311     uint8_t smbc_bustate; /* boot-up state */

```

```

309     uint8_t smbc_psstate; /* power supply state */
310     uint8_t smbc_thstate; /* thermal state */
311     uint8_t smbc_security; /* security status */
312     uint8_t smbc_uheight; /* enclosure height in U's */
313     uint8_t smbc_cords; /* number of power cords */
314     uint8_t smbc_elems; /* number of element records (n) */
315     uint8_t smbc_elemlen; /* length of contained element (m) */
316     char smbc_sku[256]; /* SKU number (as a string) */
317 } smbios_chassis_t;
318
319 #define SMB_CHT_OTHER 0x01 /* other */
320 #define SMB_CHT_UNKNOWN 0x02 /* unknown */
321 #define SMB_CHT_DESKTOP 0x03 /* desktop */
322 #define SMB_CHT_LPDESKTOP 0x04 /* low-profile desktop */
323 #define SMB_CHT_PIZZA 0x05 /* pizza box */
324 #define SMB_CHT_MINITOWER 0x06 /* mini-tower */
325 #define SMB_CHT_TOWER 0x07 /* tower */
326 #define SMB_CHT_PORTABLE 0x08 /* portable */
327 #define SMB_CHT_LAPTOP 0x09 /* laptop */
328 #define SMB_CHT_NOTEBOOK 0x0A /* notebook */
329 #define SMB_CHT_HANDHELD 0x0B /* hand-held */
330 #define SMB_CHT_DOCK 0x0C /* docking station */
331 #define SMB_CHT_ALLIN1 0x0D /* all-in-one */
332 #define SMB_CHT_SUBNOTE 0x0E /* sub-notebook */
333 #define SMB_CHT_SPACESAVE 0x0F /* space-saving */
334 #define SMB_CHT_LUNCHBOX 0x10 /* lunchbox */
335 #define SMB_CHT_MAIN 0x11 /* main server chassis */
336 #define SMB_CHT_EXPANSION 0x12 /* expansion chassis */
337 #define SMB_CHT_SUB 0x13 /* sub-chassis */
338 #define SMB_CHT_BUS 0x14 /* bus expansion chassis */
339 #define SMB_CHT_PERIPHERAL 0x15 /* peripheral chassis */
340 #define SMB_CHT_RAID 0x16 /* raid chassis */
341 #define SMB_CHT_RACK 0x17 /* rack mount chassis */
342 #define SMB_CHT_SEALED 0x18 /* sealed case pc */
343 #define SMB_CHT_MULTI 0x19 /* multi-system chassis */
344 #define SMB_CHT_CPCI 0x1A /* compact PCI */
345 #define SMB_CHT_ATCA 0x1B /* advanced TCA */
346 #define SMB_CHT_BLADE 0x1C /* blade */
347 #define SMB_CHT_BLADEENC 0x1D /* blade enclosure */
348
349 #define SMB_CHST_OTHER 0x01 /* other */
350 #define SMB_CHST_UNKNOWN 0x02 /* unknown */
351 #define SMB_CHST_SAFE 0x03 /* safe */
352 #define SMB_CHST_WARNING 0x04 /* warning */
353 #define SMB_CHST_CRITICAL 0x05 /* critical */
354 #define SMB_CHST_NONREC 0x06 /* non-recoverable */
355
356 #define SMB_CHSC_OTHER 0x01 /* other */
357 #define SMB_CHSC_UNKNOWN 0x02 /* unknown */
358 #define SMB_CHSC_NONE 0x03 /* none */
359 #define SMB_CHSC_EILOCK 0x04 /* external interface locked out */
360 #define SMB_CHSC_EIENAB 0x05 /* external interface enabled */
361
362 /*
363  * SMBIOS Processor description. See DSP0134 Section 7.5 for more details.
364  * SBIOS Processor description. See DSP0134 Section 3.3.5 for more details.
365  * If the L1, L2, or L3 cache handle is -1, the cache information is unknown.
366  * If the handle refers to something of size 0, that type of cache is absent.
367  * NOTE: Although SMBIOS exports a 64-bit CPUID result, this value should not
368  * be used for any purpose other than BIOS debugging. illumos itself computes
369  * its own CPUID value and applies knowledge of additional errata and processor
370  * specific CPUID variations, so this value should not be used for anything.
371 */
372 typedef struct smbios_processor {

```

```

373     uint64_t smbp_cpuid;           /* processor cpuid information */
374     uint32_t smbp_family;         /* processor family */
375     uint8_t smbp_type;           /* processor type (SMB_PRT_*) */
376     uint8_t smbp_voltage;        /* voltage (SMB_PRV_*) */
377     uint8_t smbp_status;         /* status (SMB_PRS_*) */
378     uint8_t smbp_upgrade;        /* upgrade (SMB_PRU_*) */
379     uint32_t smbp_clkspeed;       /* external clock speed in MHz */
380     uint32_t smbp_maxspeed;       /* maximum speed in MHz */
381     uint32_t smbp_curspeed;       /* current speed in MHz */
382     id_t smbp_l1cache;           /* L1 cache handle */
383     id_t smbp_l2cache;           /* L2 cache handle */
384     id_t smbp_l3cache;           /* L3 cache handle */
385     uint8_t smbp_corecount;       /* number of cores per processor socket */
386     uint8_t smbp_coresenabled;    /* number of enabled cores per processor socket */
387     /* number of enabled cores per processor socket */
388     uint8_t smbp_threadcount;     /* number of threads per processor socket */
389     /* number of threads per processor socket */
390     uint16_t smbp_cflags;         /* processor characteristics (SMB_PRC_*) */
391     /* processor characteristics (SMB_PRC_*) */
392     uint16_t smbp_family2;        /* processor family 2 */
393 } smbios_processor_t;

395 #define SMB_PRT_OTHER           0x01 /* other */
396 #define SMB_PRT_UNKNOWN        0x02 /* unknown */
397 #define SMB_PRT_CENTRAL        0x03 /* central processor */
398 #define SMB_PRT_MATH           0x04 /* math processor */
399 #define SMB_PRT_DSP            0x05 /* DSP processor */
400 #define SMB_PRT_VIDEO          0x06 /* video processor */

402 #define SMB_PRV_LEGACY(v)      (!((v) & 0x80)) /* legacy voltage mode */
403 #define SMB_PRV_FIXED(v)      ((v) & 0x80) /* fixed voltage mode */

405 #define SMB_PRV_5V             0x01 /* 5V is supported */
406 #define SMB_PRV_33V           0x02 /* 3.3V is supported */
407 #define SMB_PRV_29V           0x04 /* 2.9V is supported */

409 #define SMB_PRV_VOLTAGE(v)     ((v) & 0x7f)

411 #define SMB_PRSTATUS_PRESENT(s) ((s) & 0x40) /* socket is populated */
412 #define SMB_PRSTATUS_STATUS(s) ((s) & 0x07) /* status (see below) */

414 #define SMB_PRS_UNKNOWN        0x0 /* unknown */
415 #define SMB_PRS_ENABLED        0x1 /* enabled */
416 #define SMB_PRS_BDISABLED      0x2 /* disabled in bios user setup */
417 #define SMB_PRS_PDISABLED      0x3 /* disabled in bios from post error */
418 #define SMB_PRS_IDLE           0x4 /* waiting to be enabled */
419 #define SMB_PRS_OTHER          0x7 /* other */

421 #define SMB_PRU_OTHER           0x01 /* other */
422 #define SMB_PRU_UNKNOWN        0x02 /* unknown */
423 #define SMB_PRU_DAUGHTER       0x03 /* daughter board */
424 #define SMB_PRU_ZIF            0x04 /* ZIF socket */
425 #define SMB_PRU_PIGGY          0x05 /* replaceable piggy back */
426 #define SMB_PRU_NONE           0x06 /* none */
427 #define SMB_PRU_LIF            0x07 /* LIF socket */
428 #define SMB_PRU_SLOT1          0x08 /* slot 1 */
429 #define SMB_PRU_SLOT2          0x09 /* slot 2 */
430 #define SMB_PRU_370PIN         0x0A /* 370-pin socket */
431 #define SMB_PRU_SLOTA          0x0B /* slot A */
432 #define SMB_PRU_SLOTM          0x0C /* slot M */
433 #define SMB_PRU_423            0x0D /* socket 423 */
434 #define SMB_PRU_A              0x0E /* socket A (socket 462) */
435 #define SMB_PRU_478            0x0F /* socket 478 */
436 #define SMB_PRU_754            0x10 /* socket 754 */
437 #define SMB_PRU_940            0x11 /* socket 940 */
438 #define SMB_PRU_939            0x12 /* socket 939 */

```

```

439 #define SMB_PRU_MPGA604        0x13 /* mPGA604 */
440 #define SMB_PRU_LGA771        0x14 /* LGA771 */
441 #define SMB_PRU_LGA775        0x15 /* LGA775 */
442 #define SMB_PRU_S1            0x16 /* socket S1 */
443 #define SMB_PRU_AM2           0x17 /* socket AM2 */
444 #define SMB_PRU_F             0x18 /* socket F */
445 #define SMB_PRU_LGA1366       0x19 /* LGA1366 */
446 #define SMB_PRU_G34           0x1A /* socket G34 */
447 #define SMB_PRU_AM3           0x1B /* socket AM3 */
448 #define SMB_PRU_C32           0x1C /* socket C32 */
449 #define SMB_PRU_LGA1156       0x1D /* LGA1156 */
450 #define SMB_PRU_LGA1567       0x1E /* LGA1567 */
451 #define SMB_PRU_PGA988A       0x1F /* PGA988A */
452 #define SMB_PRU_BGA1288       0x20 /* BGA1288 */
453 #define SMB_PRU_RPGA988B      0x21 /* rPGA988B */
454 #define SMB_PRU_BGA1023       0x22 /* BGA1023 */
455 #define SMB_PRU_BGA1224       0x23 /* BGA1224 */
456 #define SMB_PRU_LGA1155       0x24 /* LGA1155 */
457 #define SMB_PRU_LGA1356       0x25 /* LGA1356 */
458 #define SMB_PRU_LGA2011       0x26 /* LGA2011 */
459 #define SMB_PRU_FS1           0x27 /* socket FS1 */
460 #define SMB_PRU_FS2           0x28 /* socket FS2 */
461 #define SMB_PRU_FM1           0x29 /* socket FM1 */
462 #define SMB_PRU_FM2           0x2A /* socket FM2 */
463 #define SMB_PRU_LGA2011-3     0x2B /* LGA2011-3 */
464 #define SMB_PRU_LGA1356-3     0x2C /* LGA1356-3 */

466 #define SMB_PRC_RESERVED      0x0001 /* reserved */
467 #define SMB_PRC_UNKNOWN        0x0002 /* unknown */
468 #define SMB_PRC_64BIT         0x0004 /* 64-bit capable */
469 #define SMB_PRC_MC            0x0008 /* multi-core */
470 #define SMB_PRC_HT            0x0010 /* hardware thread */
471 #define SMB_PRC_NX            0x0020 /* execution protection */
472 #define SMB_PRC_VT            0x0040 /* enhanced virtualization */
473 #define SMB_PRC_PM            0x0080 /* power/performance control */

475 #define SMB_PRF_OTHER          0x01 /* other */
476 #define SMB_PRF_UNKNOWN        0x02 /* unknown */
477 #define SMB_PRF_8086          0x03 /* 8086 */
478 #define SMB_PRF_80286         0x04 /* 80286 */
479 #define SMB_PRF_I386          0x05 /* Intel 386 */
480 #define SMB_PRF_I486          0x06 /* Intel 486 */
481 #define SMB_PRF_8087          0x07 /* 8087 */
482 #define SMB_PRF_80287         0x08 /* 80287 */
483 #define SMB_PRF_80387         0x09 /* 80387 */
484 #define SMB_PRF_80487         0x0A /* 80487 */
485 #define SMB_PRF_PENTIUM        0x0B /* Pentium Family */
486 #define SMB_PRF_PENTIUMPRO     0x0C /* Pentium Pro */
487 #define SMB_PRF_PENTIUMII      0x0D /* Pentium II */
488 #define SMB_PRF_PENTIUM_MMX    0x0E /* Pentium w/ MMX */
489 #define SMB_PRF_CELERON        0x0F /* Celeron */
490 #define SMB_PRF_PENTIUMII_XEON 0x10 /* Pentium II Xeon */
491 #define SMB_PRF_PENTIUMIII     0x11 /* Pentium III */
492 #define SMB_PRF_M1            0x12 /* M1 */
493 #define SMB_PRF_M2            0x13 /* M2 */
494 #define SMB_PRF_CELERON_M      0x14 /* Celeron M */
495 #define SMB_PRF_PENTIUMIV_HT   0x15 /* Pentium 4 HT */
496 #define SMB_PRF_DURON          0x18 /* AMD Duron */
497 #define SMB_PRF_K5            0x19 /* K5 */
498 #define SMB_PRF_K6            0x1A /* K6 */
499 #define SMB_PRF_K6_2          0x1B /* K6-2 */
500 #define SMB_PRF_K6_3          0x1C /* K6-3 */
501 #define SMB_PRF_ATHLON         0x1D /* Athlon */
502 #define SMB_PRF_2900          0x1E /* AMD 2900 */
503 #define SMB_PRF_K6_2PLUS       0x1F /* K6-2+ */
504 #define SMB_PRF_PPC            0x20 /* PowerPC */

```

```

505 #define SMB_PRF_PPC_601      0x21    /* PowerPC 601 */
506 #define SMB_PRF_PPC_603      0x22    /* PowerPC 603 */
507 #define SMB_PRF_PPC_603PLUS  0x23    /* PowerPC 603+ */
508 #define SMB_PRF_PPC_604      0x24    /* PowerPC 604 */
509 #define SMB_PRF_PPC_620      0x25    /* PowerPC 620 */
510 #define SMB_PRF_PPC_704      0x26    /* PowerPC x704 */
511 #define SMB_PRF_PPC_750      0x27    /* PowerPC 750 */
512 #define SMB_PRF_CORE_DUO     0x28    /* Core Duo */
513 #define SMB_PRF_CORE_DUO_M   0x29    /* Core Duo mobile */
514 #define SMB_PRF_CORE_SOLO_M  0x2A    /* Core Solo mobile */
515 #define SMB_PRF_ATOM          0x2B    /* Intel Atom */
516 #define SMB_PRF_ALPHA         0x30    /* Alpha */
517 #define SMB_PRF_ALPHA_21064  0x31    /* Alpha 21064 */
518 #define SMB_PRF_ALPHA_21066  0x32    /* Alpha 21066 */
519 #define SMB_PRF_ALPHA_21164  0x33    /* Alpha 21164 */
520 #define SMB_PRF_ALPHA_21164PC 0x34    /* Alpha 21164PC */
521 #define SMB_PRF_ALPHA_21164A 0x35    /* Alpha 21164a */
522 #define SMB_PRF_ALPHA_21264  0x36    /* Alpha 21264 */
523 #define SMB_PRF_ALPHA_21364  0x37    /* Alpha 21364 */
524 #define SMB_PRF_TURION2U_2C_MM 0x38 /* AMD Turion II Ultra Dual-Core Mobile M */
525 /* AMD Turion II Ultra Dual-Core Mobile M */
526 #define SMB_PRF_TURION2_2C_MM 0x39 /* AMD Turion II Dual-Core Mobile M */
527 #define SMB_PRF_ATHLON2_2C_M  0x3A /* AMD Athlon II Dual-Core M */
528 #define SMB_PRF_OPTERON_6100   0x3B /* AMD Opteron 6100 series */
529 #define SMB_PRF_OPTERON_4100   0x3C /* AMD Opteron 4100 series */
530 #define SMB_PRF_OPTERON_6200   0x3D /* AMD Opteron 6200 series */
531 #define SMB_PRF_OPTERON_4200   0x3E /* AMD Opteron 4200 series */
532 #define SMB_PRF_AMD_FX         0x3F /* AMD FX series */
533 #define SMB_PRF_MIPS           0x40 /* MIPS */
534 #define SMB_PRF_MIPS_R4000     0x41 /* MIPS R4000 */
535 #define SMB_PRF_MIPS_R4200     0x42 /* MIPS R4200 */
536 #define SMB_PRF_MIPS_R4400     0x43 /* MIPS R4400 */
537 #define SMB_PRF_MIPS_R4600     0x44 /* MIPS R4600 */
538 #define SMB_PRF_MIPS_R10000    0x45 /* MIPS R10000 */
539 #define SMB_PRF_AMD_C          0x46 /* AMD C-series */
540 #define SMB_PRF_AMD_E          0x47 /* AMD E-series */
541 #define SMB_PRF_AMD_A          0x48 /* AMD A-series */
542 #define SMB_PRF_AMD_G          0x49 /* AMD G-series */
543 #define SMB_PRF_AMD_Z          0x4A /* AMD Z-series */
544 #define SMB_PRF_AMD_R          0x4B /* AMD R-series */
545 #define SMB_PRF_OPTERON_4300   0x4C /* AMD Opteron 4300 series */
546 #define SMB_PRF_OPTERON_6300   0x4D /* AMD Opteron 6300 series */
547 #define SMB_PRF_OPTERON_3300   0x4E /* AMD Opteron 3300 series */
548 #define SMB_PRF_AMD_FIREPRO    0x4F /* AMD FirePro series */
549 #define SMB_PRF_SPARC          0x50 /* SPARC */
550 #define SMB_PRF_SUPERSPARC     0x51 /* SuperSPARC */
551 #define SMB_PRF_MICROSPARCI    0x52 /* microSPARC II */
552 #define SMB_PRF_MICROSPARCIiep 0x53 /* microSPARC IIep */
553 #define SMB_PRF_ULTRASPARC     0x54 /* UltraSPARC */
554 #define SMB_PRF_USII           0x55 /* UltraSPARC II */
555 #define SMB_PRF_USIII          0x56 /* UltraSPARC III */
556 #define SMB_PRF_USIIIi         0x57 /* UltraSPARC IIIi */
557 #define SMB_PRF_USIIII         0x58 /* UltraSPARC IIIi */
558 #define SMB_PRF_68040          0x60 /* 68040 */
559 #define SMB_PRF_68XXX          0x61 /* 68XXX */
560 #define SMB_PRF_68000          0x62 /* 68000 */
561 #define SMB_PRF_68010          0x63 /* 68010 */
562 #define SMB_PRF_68020          0x64 /* 68020 */
563 #define SMB_PRF_68030          0x65 /* 68030 */
564 #define SMB_PRF_HOBBIT         0x70 /* Hobbit */
565 #define SMB_PRF_TM5000         0x78 /* Crusoe TM5000 */
566 #define SMB_PRF_TM3000         0x79 /* Crusoe TM3000 */
567 #define SMB_PRF_TM8000         0x7A /* Efficeon TM8000 */
568 #define SMB_PRF_WEITEK         0x80 /* Weitek */
569 #define SMB_PRF_ITANIC         0x82 /* Itanium */
570 #define SMB_PRF_ATHLON64       0x83 /* Athlon64 */

```

```

571 #define SMB_PRF_OPTERON      0x84    /* Opteron */
572 #define SMB_PRF_SEMPRON      0x85    /* Sempron */
573 #define SMB_PRF_TURION64_M    0x86    /* Turion 64 Mobile */
574 #define SMB_PRF_OPTERON_2C    0x87    /* AMD Opteron Dual-Core */
575 #define SMB_PRF_ATHLON64_X2_2C 0x88    /* AMD Athlon 64 X2 Dual-Core */
576 #define SMB_PRF_TURION64_X2_M 0x89    /* AMD Turion 64 X2 Mobile */
577 #define SMB_PRF_OPTERON_4C    0x8A    /* AMD Opteron Quad-Core */
578 #define SMB_PRF_OPTERON_3G    0x8B    /* AMD Opteron 3rd Generation */
579 #define SMB_PRF_PHENOM_FX_4C  0x8C    /* AMD Phenom FX Quad-Core */
580 #define SMB_PRF_PHENOM_X4_4C  0x8D    /* AMD Phenom X4 Quad-Core */
581 #define SMB_PRF_PHENOM_X2_2C  0x8E    /* AMD Phenom X2 Dual-Core */
582 #define SMB_PRF_ATHLON_X2_2C  0x8F    /* AMD Athlon X2 Dual-Core */
583 #define SMB_PRF_PA            0x90    /* PA-RISC */
584 #define SMB_PRF_PA8500        0x91    /* PA-RISC 8500 */
585 #define SMB_PRF_PA8000        0x92    /* PA-RISC 8000 */
586 #define SMB_PRF_PA7300LC      0x93    /* PA-RISC 7300LC */
587 #define SMB_PRF_PA7200        0x94    /* PA-RISC 7200 */
588 #define SMB_PRF_PA7100LC      0x95    /* PA-RISC 7100LC */
589 #define SMB_PRF_PA7100        0x96    /* PA-RISC 7100 */
590 #define SMB_PRF_V30           0xA0    /* V30 */
591 #define SMB_PRF_XEON_4C_3200   0xA1    /* Xeon Quad Core 3200 */
592 #define SMB_PRF_XEON_2C_3000   0xA2    /* Xeon Dual Core 3000 */
593 #define SMB_PRF_XEON_4C_5300   0xA3    /* Xeon Quad Core 5300 */
594 #define SMB_PRF_XEON_2C_5100   0xA4    /* Xeon Dual Core 5100 */
595 #define SMB_PRF_XEON_2C_5000   0xA5    /* Xeon Dual Core 5000 */
596 #define SMB_PRF_XEON_2C_LV     0xA6    /* Xeon Dual Core LV */
597 #define SMB_PRF_XEON_2C_ULV    0xA7    /* Xeon Dual Core ULV */
598 #define SMB_PRF_XEON_2C_7100   0xA8    /* Xeon Dual Core 7100 */
599 #define SMB_PRF_XEON_4C_5400   0xA9    /* Xeon Quad Core 5400 */
600 #define SMB_PRF_XEON_4C        0xAA    /* Xeon Quad Core */
601 #define SMB_PRF_XEON_2C_5200   0xAB    /* Xeon Dual Core 5200 */
602 #define SMB_PRF_XEON_2C_7200   0xAC    /* Xeon Dual Core 7200 */
603 #define SMB_PRF_XEON_4C_7300   0xAD    /* Xeon Quad Core 7300 */
604 #define SMB_PRF_XEON_4C_7400   0xAE    /* Xeon Quad Core 7400 */
605 #define SMB_PRF_XEON_XC_7400   0xAF    /* Xeon Multi Core 7400 */
606 #define SMB_PRF_PENTIUMIII_XEON 0xB0    /* Pentium III Xeon */
607 #define SMB_PRF_PENTIUMIII_SS  0xB1    /* Pentium III with SpeedStep */
608 #define SMB_PRF_P4             0xB2    /* Pentium 4 */
609 #define SMB_PRF_XEON           0xB3    /* Intel Xeon */
610 #define SMB_PRF_AS400          0xB4    /* AS400 */
611 #define SMB_PRF_XEON_MP        0xB5    /* Intel Xeon MP */
612 #define SMB_PRF_ATHLON_XP      0xB6    /* AMD Athlon XP */
613 #define SMB_PRF_ATHLON_MP      0xB7    /* AMD Athlon MP */
614 #define SMB_PRF_ITANIC2        0xB8    /* Itanium 2 */
615 #define SMB_PRF_PENTIUM_M      0xB9    /* Pentium M */
616 #define SMB_PRF_CELERON_D      0xBA    /* Celeron D */
617 #define SMB_PRF_PENTIUM_D      0xBB    /* Pentium D */
618 #define SMB_PRF_PENTIUM_EE     0xBC    /* Pentium Extreme Edition */
619 #define SMB_PRF_CORE_SOLO      0xBD    /* Intel Core Solo */
620 #define SMB_PRF_CORE2_DUO      0xBF    /* Intel Core 2 Duo */
621 #define SMB_PRF_CORE2_SOLO     0xC0    /* Intel Core 2 Solo */
622 #define SMB_PRF_CORE2_EX       0xC1    /* Intel Core 2 Extreme */
623 #define SMB_PRF_CORE2_QUAD     0xC2    /* Intel Core 2 Quad */
624 #define SMB_PRF_CORE2_EX_M     0xC3    /* Intel Core 2 Extreme mobile */
625 #define SMB_PRF_CORE2_DUO_M    0xC4    /* Intel Core 2 Duo mobile */
626 #define SMB_PRF_CORE2_SOLO_M   0xC5    /* Intel Core 2 Solo mobile */
627 #define SMB_PRF_CORE_I7        0xC6    /* Intel Core i7 */
628 #define SMB_PRF_CELERON_2C     0xC7    /* Celeron Dual-Core */
629 #define SMB_PRF_CORE           0xBD    /* Intel Core */
630 #define SMB_PRF_CORE2          0xBF    /* Intel Core 2 */
631 #define SMB_PRF_IBM390         0xC8    /* IBM 390 */
632 #define SMB_PRF_G4             0xC9    /* G4 */
633 #define SMB_PRF_G5             0xCA    /* G5 */
634 #define SMB_PRF_ESA390         0xCB    /* ESA390 */
635 #define SMB_PRF_ZARCH          0xCC    /* z/Architecture */
636 #define SMB_PRF_CORE_I5        0xCD    /* Intel Core i5 */

```

```

635 #define SMB_PRF_CORE_I3      0xCE /* Intel Core i3 */
636 #define SMB_PRF_C7M          0xD2 /* VIA C7-M */
637 #define SMB_PRF_C7D          0xD3 /* VIA C7-D */
638 #define SMB_PRF_C7           0xD4 /* VIA C7 */
639 #define SMB_PRF_EDEN         0xD5 /* VIA Eden */
640 #define SMB_PRF_XEON_XC      0xD6 /* Intel Xeon Multi-Core */
641 #define SMB_PRF_XEON_2C_3XXX 0xD7 /* Intel Xeon Dual-Core 3xxx */
642 #define SMB_PRF_XEON_4C_3XXX 0xD8 /* Intel Xeon Quad-Core 3xxx */
643 #define SMB_PRF_VIA_NANO     0xD9 /* VIA Nano */
644 #define SMB_PRF_XEON_2C_5XXX 0xDA /* Intel Xeon Dual-Core 5xxx */
645 #define SMB_PRF_XEON_4C_5XXX 0xDB /* Intel Xeon Quad-Core 5xxx */
646 #define SMB_PRF_XEON_2C_7XXX 0xDD /* Intel Xeon Dual-Core 7xxx */
647 #define SMB_PRF_XEON_4C_7XXX 0xDE /* Intel Xeon Quad-Core 7xxx */
648 #define SMB_PRF_XEON_XC_7XXX 0xDF /* Intel Xeon Multi-Core 7xxx */
649 #define SMB_PRF_XEON_XC_3400 0xE0 /* Intel Xeon Multi-Core 3400 */
650 #define SMB_PRF_OPTERON_3000 0xE4 /* AMD Opteron 3000 */
651 #define SMB_PRF_SEMPRON_II    0xE5 /* AMD Sempron II */
652 #define SMB_PRF_OPTERON_4C_EM 0xE6 /* AMD Opteron Quad-Core embedded */
653 #define SMB_PRF_PHENOM_3C     0xE7 /* AMD Phenom Triple-Core */
654 #define SMB_PRF_TURIONU_2C_M  0xE8 /* AMD Turion Ultra Dual-Core mobile */
655 #define SMB_PRF_TURION_2C_M   0xE9 /* AMD Turion Dual-Core mobile */
656 #define SMB_PRF_ATHLON_2C     0xEA /* AMD Athlon Dual-Core */
657 #define SMB_PRF_SEMPRON_SI    0xEB /* AMD Sempron SI */
658 #define SMB_PRF_PHENOM_II     0xEC /* AMD Phenom II */
659 #define SMB_PRF_ATHLON_II     0xED /* AMD Athlon II */
660 #define SMB_PRF_OPTERON_6C     0xEE /* AMD Opteron Six-Core */
661 #define SMB_PRF_SEMPRON_M     0xEF /* AMD Sempron M */
662 #define SMB_PRF_I860          0xFA /* 1860 */
663 #define SMB_PRF_I960          0xFB /* 1960 */
664 #define SMB_PRF_SH3           0x104 /* SH-3 */
665 #define SMB_PRF_SH4           0x105 /* SH-4 */
666 #define SMB_PRF_ARM           0x118 /* ARM */
667 #define SMB_PRF_SARM          0x119 /* StrongARM */
668 #define SMB_PRF_6X86          0x12C /* 6x86 */
669 #define SMB_PRF_MEDIAGX       0x12D /* MediaGX */
670 #define SMB_PRF_MII           0x12E /* MII */
671 #define SMB_PRF_WINCHIP       0x140 /* WinChip */
672 #define SMB_PRF_DSP           0x15E /* DSP */
673 #define SMB_PRF_VIDEO         0x1F4 /* Video Processor */

675 /*
676  * SMBIOS Cache Information. See DSP0134 Section 7.8 for more information.
677  * SBIOS Cache Information. See DSP0134 Section 3.3.8 for more information.
678  * If smba_size is zero, this indicates the specified cache is not present.
679  */
680 typedef struct smbios_cache {
681     uint32_t smba_maxsize; /* maximum installed size in bytes */
682     uint32_t smba_size; /* installed size in bytes */
683     uint16_t smba_stype; /* supported SRAM types (SMB_CAT_*) */
684     uint16_t smba_ctype; /* current SRAM type (SMB_CAT_*) */
685     uint8_t smba_speed; /* speed in nanoseconds */
686     uint8_t smba_etype; /* error correction type (SMB_CAE_*) */
687     uint8_t smba_ltype; /* logical cache type (SMB_CAG_*) */
688     uint8_t smba_assoc; /* associativity (SMB_CAA_*) */
689     uint8_t smba_level; /* cache level */
690     uint8_t smba_mode; /* cache mode (SMB_CAM_*) */
691     uint8_t smba_location; /* cache location (SMB_CAL_*) */
692     uint8_t smba_flags; /* cache flags (SMB_CAF_*) */
693 } smbios_cache_t;

694 #define SMB_CAT_OTHER      0x0001 /* other */
695 #define SMB_CAT_UNKNOWN   0x0002 /* unknown */
696 #define SMB_CAT_NONBURST  0x0004 /* non-burst */
697 #define SMB_CAT_BURST     0x0008 /* burst */
698 #define SMB_CAT_PBURST    0x0010 /* pipeline burst */
699 #define SMB_CAT_SYNC      0x0020 /* synchronous */

```

```

700 #define SMB_CAT_ASYNC      0x0040 /* asynchronous */

702 #define SMB_CAE_OTHER      0x01 /* other */
703 #define SMB_CAE_UNKNOWN   0x02 /* unknown */
704 #define SMB_CAE_NONE      0x03 /* none */
705 #define SMB_CAE_PARITY     0x04 /* parity */
706 #define SMB_CAE_SBECC     0x05 /* single-bit ECC */
707 #define SMB_CAE_MBECC     0x06 /* multi-bit ECC */

709 #define SMB_CAG_OTHER      0x01 /* other */
710 #define SMB_CAG_UNKNOWN   0x02 /* unknown */
711 #define SMB_CAG_INSTR      0x03 /* instruction */
712 #define SMB_CAG_DATA       0x04 /* data */
713 #define SMB_CAG_UNIFIED   0x05 /* unified */

715 #define SMB_CAA_OTHER      0x01 /* other */
716 #define SMB_CAA_UNKNOWN   0x02 /* unknown */
717 #define SMB_CAA_DIRECT     0x03 /* direct mapped */
718 #define SMB_CAA_2WAY       0x04 /* 2-way set associative */
719 #define SMB_CAA_4WAY       0x05 /* 4-way set associative */
720 #define SMB_CAA_FULL       0x06 /* fully associative */
721 #define SMB_CAA_8WAY       0x07 /* 8-way set associative */
722 #define SMB_CAA_16WAY      0x08 /* 16-way set associative */
723 #define SMB_CAA_12WAY      0x09 /* 12-way set associative */
724 #define SMB_CAA_24WAY      0x0A /* 24-way set associative */
725 #define SMB_CAA_32WAY      0x0B /* 32-way set associative */
726 #define SMB_CAA_48WAY      0x0C /* 48-way set associative */
727 #define SMB_CAA_64WAY      0x0D /* 64-way set associative */
728 #define SMB_CAA_20WAY      0x0E /* 20-way set associative */

730 #define SMB_CAM_WT         0x00 /* write-through */
731 #define SMB_CAM_WB         0x01 /* write-back */
732 #define SMB_CAM_VARY       0x02 /* varies by address */
733 #define SMB_CAM_UNKNOWN   0x03 /* unknown */

735 #define SMB_CAL_INTERNAL   0x00 /* internal */
736 #define SMB_CAL_EXTERNAL   0x01 /* external */
737 #define SMB_CAL_RESERVED   0x02 /* reserved */
738 #define SMB_CAL_UNKNOWN   0x03 /* unknown */

740 #define SMB_CAF_ENABLED    0x01 /* enabled at boot time */
741 #define SMB_CAF_SOCKETED   0x02 /* cache is socketed */

743 /*
744  * SBIOS Port Information. See DSP0134 Section 7.9 for more information.
745  * SBIOS Port Information. See DSP0134 Section 3.3.9 for more information.
746  * The internal reference designator string is also mapped to the location.
747  */
748 typedef struct smbios_port {
749     const char *smbo_iref; /* internal reference designator */
750     const char *smbo_eref; /* external reference designator */
751     uint8_t smbo_itype; /* internal connector type (SMB_POC_*) */
752     uint8_t smbo_etype; /* external connector type (SMB_POC_*) */
753     uint8_t smbo_ptype; /* port type (SMB_POT_*) */
754     uint8_t smbo_pad; /* padding */
755 } smbios_port_t;

756 #define SMB_POC_NONE      0x00 /* none */
757 #define SMB_POC_CENT      0x01 /* Centronics */
758 #define SMB_POC_MINICENT  0x02 /* Mini-Centronics */
759 #define SMB_POC_PROPRIETARY 0x03 /* proprietary */
760 #define SMB_POC_DB25M     0x04 /* DB-25 pin male */
761 #define SMB_POC_DB25F     0x05 /* DB-25 pin female */
762 #define SMB_POC_DB15M     0x06 /* DB-15 pin male */
763 #define SMB_POC_DB15F     0x07 /* DB-15 pin female */
764 #define SMB_POC_DB9M      0x08 /* DB-9 pin male */

```

```

765 #define SMB_POC_DB9F      0x09      /* DB-9 pin female */
766 #define SMB_POC_RJ11      0x0A      /* RJ-11 */
767 #define SMB_POC_RJ45      0x0B      /* RJ-45 */
768 #define SMB_POC_MINISCSI  0x0C      /* 50-pin MiniSCSI */
769 #define SMB_POC_MINIDIN   0x0D      /* Mini-DIN */
770 #define SMB_POC_MICRODIN  0x0E      /* Micro-DIN */
771 #define SMB_POC_PS2       0x0F      /* PS/2 */
772 #define SMB_POC_IR        0x10      /* Infrared */
773 #define SMB_POC_HPHIL     0x11      /* HP-HIL */
774 #define SMB_POC_USB       0x12      /* USB */
775 #define SMB_POC_SSA       0x13      /* SSA SCSI */
776 #define SMB_POC_DIN8M     0x14      /* Circular DIN-8 male */
777 #define SMB_POC_DIN8F     0x15      /* Circular DIN-8 female */
778 #define SMB_POC_OBIDE     0x16      /* on-board IDE */
779 #define SMB_POC_OBFLOPPY  0x17      /* on-board floppy */
780 #define SMB_POC_DI9       0x18      /* 9p dual inline (p10 cut) */
781 #define SMB_POC_DI25      0x19      /* 25p dual inline (p26 cut) */
782 #define SMB_POC_DI50      0x1A      /* 50p dual inline */
783 #define SMB_POC_DI68      0x1B      /* 68p dual inline */
784 #define SMB_POC_CDRUM     0x1C      /* on-board sound from CDRUM */
785 #define SMB_POC_MINI14    0x1D      /* Mini-Centronics Type 14 */
786 #define SMB_POC_MINI26    0x1E      /* Mini-Centronics Type 26 */
787 #define SMB_POC_MINIJACK  0x1F      /* Mini-jack (headphones) */
788 #define SMB_POC_BNC       0x20      /* BNC */
789 #define SMB_POC_1394      0x21      /* 1394 */
790 #define SMB_POC_SATA       0x22      /* SAS/SATA plug receptacle */
791 #define SMB_POC_PC98      0xA0      /* PC-98 */
792 #define SMB_POC_PC98HR    0xA1      /* PC-98Hireso */
793 #define SMB_POC_PCH98     0xA2      /* PC-H98 */
794 #define SMB_POC_PC98NOTE  0xA3      /* PC-98Note */
795 #define SMB_POC_PC98FULL  0xA4      /* PC-98Full */
796 #define SMB_POC_OTHER     0xFF      /* other */

798 #define SMB_POT_NONE      0x00      /* none */
799 #define SMB_POT_PP_XTAT   0x01      /* Parallel Port XT/AT compat */
800 #define SMB_POT_PP_PS2    0x02      /* Parallel Port PS/2 */
801 #define SMB_POT_PP_ECP    0x03      /* Parallel Port ECP */
802 #define SMB_POT_PP_EPP    0x04      /* Parallel Port EPP */
803 #define SMB_POT_PP_ECPEPP 0x05      /* Parallel Port ECP/EPP */
804 #define SMB_POT_SP_XTAT   0x06      /* Serial Port XT/AT compat */
805 #define SMB_POT_SP_16450  0x07      /* Serial Port 16450 compat */
806 #define SMB_POT_SP_16550  0x08      /* Serial Port 16550 compat */
807 #define SMB_POT_SP_16550A 0x09      /* Serial Port 16550A compat */
808 #define SMB_POT_SCSI      0x0A      /* SCSI port */
809 #define SMB_POT_MIDI      0x0B      /* MIDI port */
810 #define SMB_POT_JOYSTICK  0x0C      /* Joystick port */
811 #define SMB_POT_KEYBOARD  0x0D      /* Keyboard port */
812 #define SMB_POT_MOUSE     0x0E      /* Mouse port */
813 #define SMB_POT_SSA       0x0F      /* SSA SCSI */
814 #define SMB_POT_USB       0x10      /* USB */
815 #define SMB_POT_FIREWIRE  0x11      /* Firewire (IEEE P1394) */
816 #define SMB_POT_PCMII     0x12      /* PCMCIA Type II */
817 #define SMB_POT_PCMIIa    0x13      /* PCMCIA Type II (alternate) */
818 #define SMB_POT_PCMIII    0x14      /* PCMCIA Type III */
819 #define SMB_POT_CARDBUS   0x15      /* Cardbus */
820 #define SMB_POT_ACCESS    0x16      /* Access Bus Port */
821 #define SMB_POT_SCSI2     0x17      /* SCSI II */
822 #define SMB_POT_SCSIW     0x18      /* SCSI Wide */
823 #define SMB_POT_PC98      0x19      /* PC-98 */
824 #define SMB_POT_PC98HR    0x1A      /* PC-98Hireso */
825 #define SMB_POT_PCH98     0x1B      /* PC-H98 */
826 #define SMB_POT_VIDEO     0x1C      /* Video port */
827 #define SMB_POT_AUDIO     0x1D      /* Audio port */
828 #define SMB_POT_MODEM     0x1E      /* Modem port */
829 #define SMB_POT_NETWORK   0x1F      /* Network port */
830 #define SMB_POT_SATA      0x20      /* SATA */

```

```

831 #define SMB_POT_SAS      0x21      /* SAS */
832 #define SMB_POT_8251     0xA0      /* 8251 compatible */
833 #define SMB_POT_8251F    0xA1      /* 8251 FIFO compatible */
834 #define SMB_POT_OTHER    0xFF      /* other */

836 /*
837  * SMBIOS Slot Information. See DSP0134 Section 7.10 for more information.
838  * See DSP0134 7.10.5 for how to interpret the value of smbl_id.
839  * SMBIOS Slot Information. See DSP0134 Section 3.3.10 for more information.
840  * See DSP0134 3.3.10.5 for how to interpret the value of smbl_id.
841 */
842 typedef struct smbios_slot {
843     const char *smbl_name;          /* reference designation */
844     uint8_t smbl_type;              /* slot type */
845     uint8_t smbl_width;             /* slot data bus width */
846     uint8_t smbl_usage;             /* current usage */
847     uint8_t smbl_length;            /* slot length */
848     uint16_t smbl_id;               /* slot ID */
849     uint8_t smbl_ch1;               /* slot characteristics 1 */
850     uint8_t smbl_ch2;               /* slot characteristics 2 */
851     uint16_t smbl_sg;               /* segment group number */
852     uint8_t smbl_bus;               /* bus number */
853     uint8_t smbl_df;               /* device/function number */
854 } smbios_slot_t;

854 #define SMB_SLT_OTHER      0x01      /* other */
855 #define SMB_SLT_UNKNOWN    0x02      /* unknown */
856 #define SMB_SLT_ISA        0x03      /* ISA */
857 #define SMB_SLT_MCA        0x04      /* MCA */
858 #define SMB_SLT_EISA       0x05      /* EISA */
859 #define SMB_SLT_PCI        0x06      /* PCI */
860 #define SMB_SLT_PCMCIA     0x07      /* PCMCIA */
861 #define SMB_SLT_VLVESA     0x08      /* VL-VESA */
862 #define SMB_SLT_PROPRIETARY 0x09      /* proprietary */
863 #define SMB_SLT_PROCC      0x0A      /* processor card slot */
864 #define SMB_SLT_MEM        0x0B      /* proprietary memory card slot */
865 #define SMB_SLT_IOR        0x0C      /* I/O riser card slot */
866 #define SMB_SLT_NUBUS      0x0D      /* NuBus */
867 #define SMB_SLT_PCI66      0x0E      /* PCI (66MHz capable) */
868 #define SMB_SLT_AGP        0x0F      /* AGP */
869 #define SMB_SLT_AGP2X      0x10      /* AGP 2X */
870 #define SMB_SLT_AGP4X      0x11      /* AGP 4X */
871 #define SMB_SLT_PCIX       0x12      /* PCI-X */
872 #define SMB_SLT_AGP8X      0x13      /* AGP 8X */
873 #define SMB_SLT_PC98_C20   0xA0      /* PC-98/C20 */
874 #define SMB_SLT_PC98_C24   0xA1      /* PC-98/C24 */
875 #define SMB_SLT_PC98_E     0xA2      /* PC-98/E */
876 #define SMB_SLT_PC98_LB    0xA3      /* PC-98/Local Bus */
877 #define SMB_SLT_PC98_C     0xA4      /* PC-98/Card */
878 #define SMB_SLT_PCIE       0xA5      /* PCI Express */
879 #define SMB_SLT_PCIE1      0xA6      /* PCI Express x1 */
880 #define SMB_SLT_PCIE2      0xA7      /* PCI Express x2 */
881 #define SMB_SLT_PCIE4      0xA8      /* PCI Express x4 */
882 #define SMB_SLT_PCIE8      0xA9      /* PCI Express x8 */
883 #define SMB_SLT_PCIE16     0xAA      /* PCI Express x16 */
884 #define SMB_SLT_PCIE2G     0xAB      /* PCI Exp. Gen 2 */
885 #define SMB_SLT_PCIE2G1    0xAC      /* PCI Exp. Gen 2 x1 */
886 #define SMB_SLT_PCIE2G2    0xAD      /* PCI Exp. Gen 2 x2 */
887 #define SMB_SLT_PCIE2G4    0xAE      /* PCI Exp. Gen 2 x4 */
888 #define SMB_SLT_PCIE2G8    0xAF      /* PCI Exp. Gen 2 x8 */
889 #define SMB_SLT_PCIE2G16   0xB0      /* PCI Exp. Gen 2 x16 */
890 #define SMB_SLT_PCIE3G     0xB1      /* PCI Exp. Gen 3 */
891 #define SMB_SLT_PCIE3G1    0xB2      /* PCI Exp. Gen 3 x1 */
892 #define SMB_SLT_PCIE3G2    0xB3      /* PCI Exp. Gen 3 x2 */
893 #define SMB_SLT_PCIE3G4    0xB4      /* PCI Exp. Gen 3 x4 */
894 #define SMB_SLT_PCIE3G8    0xB5      /* PCI Exp. Gen 3 x8 */

```

```

895 #define SMB_SLT_PCIE3G16      0xB6    /* PCI Exp. Gen 3 x16 */

897 #define SMB_SLW_OTHER          0x01    /* other */
898 #define SMB_SLW_UNKNOWN        0x02    /* unknown */
899 #define SMB_SLW_8               0x03    /* 8 bit */
900 #define SMB_SLW_16             0x04    /* 16 bit */
901 #define SMB_SLW_32             0x05    /* 32 bit */
902 #define SMB_SLW_64             0x06    /* 64 bit */
903 #define SMB_SLW_128            0x07    /* 128 bit */
904 #define SMB_SLW_1X             0x08    /* 1x or x1 */
905 #define SMB_SLW_2X             0x09    /* 2x or x2 */
906 #define SMB_SLW_4X             0x0A    /* 4x or x4 */
907 #define SMB_SLW_8X             0x0B    /* 8x or x8 */
908 #define SMB_SLW_12X            0x0C    /* 12x or x12 */
909 #define SMB_SLW_16X            0x0D    /* 16x or x16 */
910 #define SMB_SLW_32X            0x0E    /* 32x or x32 */

912 #define SMB_SLU_OTHER          0x01    /* other */
913 #define SMB_SLU_UNKNOWN        0x02    /* unknown */
914 #define SMB_SLU_AVAIL          0x03    /* available */
915 #define SMB_SLU_INUSE          0x04    /* in use */

917 #define SMB_SLL_OTHER          0x01    /* other */
918 #define SMB_SLL_UNKNOWN        0x02    /* unknown */
919 #define SMB_SLL_SHORT          0x03    /* short length */
920 #define SMB_SLL_LONG           0x04    /* long length */

922 #define SMB_SLCH1_UNKNOWN      0x01    /* characteristics unknown */
923 #define SMB_SLCH1_5V           0x02    /* provides 5.0V */
924 #define SMB_SLCH1_33V          0x04    /* provides 3.3V */
925 #define SMB_SLCH1_SHARED       0x08    /* opening shared with other slot */
926 #define SMB_SLCH1_PC16         0x10    /* slot supports PC Card-16 */
927 #define SMB_SLCH1_PCCB         0x20    /* slot supports CardBus */
928 #define SMB_SLCH1_PCZV         0x40    /* slot supports Zoom Video */
929 #define SMB_SLCH1_PCMRR        0x80    /* slot supports Modem Ring Resume */

931 #define SMB_SLCH2_PME          0x01    /* slot supports PME# signal */
932 #define SMB_SLCH2_HOTPLUG      0x02    /* slot supports hot-plug devices */
933 #define SMB_SLCH2_SMBUS        0x04    /* slot supports SMBus signal */

935 /*
936  * SMBIOS On-Board Device Information. See DSP0134 Section 7.11 for more
937  * information. Any number of on-board device sections may be present, each
938  * containing one or more records. The smbios_info_obdevs() function permits
939  * the caller to retrieve one or more of the records from a given section.
940  */
941 typedef struct smbios_obdev {
942     const char *smbd_name;    /* description string for this device */
943     uint8_t smbd_type;        /* type code (SMB_OBT_*) */
944     uint8_t smbd_enabled;     /* boolean (device is enabled) */
945 } smbios_obdev_t;

947 #define SMB_OBT_OTHER          0x01    /* other */
948 #define SMB_OBT_UNKNOWN        0x02    /* unknown */
949 #define SMB_OBT_VIDEO          0x03    /* video */
950 #define SMB_OBT_SCSI           0x04    /* scsi */
951 #define SMB_OBT_ETHERNET       0x05    /* ethernet */
952 #define SMB_OBT_TOKEN          0x06    /* token ring */
953 #define SMB_OBT_SOUND          0x07    /* sound */
954 #define SMB_OBT_PATA           0x08    /* pata */
955 #define SMB_OBT_SATA           0x09    /* sata */
956 #define SMB_OBT_SAS            0x0A    /* sas */

958 /*
959  * SMBIOS BIOS Language Information. See DSP0134 Section 7.14 for more

```

```

816  * SMBIOS BIOS Language Information. See DSP0134 Section 3.3.14 for more
860  * information. The smbios_info_strtab() function can be applied using a
861  * count of smbla_num to retrieve the other possible language settings.
862  */
863 typedef struct smbios_lang {
864     const char *smbla_cur;    /* current language setting */
865     uint_t smbla_fmt;         /* language name format (see below) */
866     uint_t smbla_num;         /* number of installed languages */
867 } smbios_lang_t;

869 #define SMB_LFMT_LONG          0        /* <ISO639>|<ISO3166>|Encoding Method */
870 #define SMB_LFMT_SHORT        1        /* <ISO930>|<ISO3166> */

872 /*
873  * SMBIOS System Event Log Information. See DSP0134 Section 7.16 for more
874  * information. Accessing the event log itself requires additional interfaces.
875  */
876 typedef struct smbios_evtype {
877     uint8_t smbevt_ltype;     /* log type */
878     uint8_t smbevt_dtype;     /* variable data format type */
879 } smbios_evtype_t;

880 #define SMB_EVFL_VALID         0x1      /* log area valid */
881 #define SMB_EVFL_FULL          0x2      /* log area full */

883 #define SMB_EVHF_NONE          0        /* no log headers used */
884 #define SMB_EVHF_F1            1        /* DMTF log header type 1 */

886 /*
887  * SMBIOS Physical Memory Array Information. See DSP0134 Section 7.17 for
888  * information. This describes a collection of physical memory devices.
889  */
890 typedef struct smbios_memarray {
891     uint8_t smbma_location;    /* physical device location */
892     uint8_t smbma_use;         /* physical device functional purpose */
893     uint8_t smbma_ecc;         /* error detect/correct mechanism */
894     uint8_t smbma_pad0;        /* padding */
895     uint32_t smbma_padi;       /* padding */
896     uint32_t smbma_ndevs;      /* number of slots or sockets */
897     id_t smbma_err;            /* handle of error (if any) */
898     uint64_t smbma_size;       /* maximum capacity in bytes */
899 } smbios_memarray_t;

901 #define SMB_MAL_OTHER          0x01    /* other */
902 #define SMB_MAL_UNKNOWN        0x02    /* unknown */
903 #define SMB_MAL_SYSMB          0x03    /* system board or motherboard */
904 #define SMB_MAL_ISA            0x04    /* ISA add-on card */
905 #define SMB_MAL_EISA           0x05    /* EISA add-on card */
906 #define SMB_MAL_PCI            0x06    /* PCI add-on card */
907 #define SMB_MAL_MCA            0x07    /* MCA add-on card */
908 #define SMB_MAL_PCMCIA         0x08    /* PCMCIA add-on card */
909 #define SMB_MAL_PROP           0x09    /* proprietary add-on card */
910 #define SMB_MAL_NUBUS          0x0A    /* NuBus */
911 #define SMB_MAL_PC98C20        0xA0    /* PC-98/C20 add-on card */
912 #define SMB_MAL_PC98C24        0xA1    /* PC-98/C24 add-on card */
913 #define SMB_MAL_PC98E          0xA2    /* PC-98/E add-on card */
914 #define SMB_MAL_PC98LB         0xA3    /* PC-98/Local bus add-on card */

```

```

1044 #define SMB_MAU_OTHER          0x01    /* other */
1045 #define SMB_MAU_UNKNOWN         0x02    /* unknown */
1046 #define SMB_MAU_SYSTEM          0x03    /* system memory */
1047 #define SMB_MAU_VIDEO           0x04    /* video memory */
1048 #define SMB_MAU_FLASH           0x05    /* flash memory */
1049 #define SMB_MAU_NVRAM           0x06    /* non-volatile RAM */
1050 #define SMB_MAU_CACHE           0x07    /* cache memory */

1052 #define SMB_MAE_OTHER           0x01    /* other */
1053 #define SMB_MAE_UNKNOWN         0x02    /* unknown */
1054 #define SMB_MAE_NONE            0x03    /* none */
1055 #define SMB_MAE_PARITY          0x04    /* parity */
1056 #define SMB_MAE_SECC            0x05    /* single-bit ECC */
1057 #define SMB_MAE_MECC            0x06    /* multi-bit ECC */
1058 #define SMB_MAE_CRC             0x07    /* CRC */

1060 /*
1061  * SMBIOS Memory Device Information. See DSP0134 Section 7.18 for more
1062  * SMBIOS Memory Device Information. See DSP0134 Section 3.3.18 for more
1063  * information. One or more of these structures are associated with each
1064  * smbios_memarray_t. A structure is present even for unpopulated sockets.
1065  * Unknown values are set to -1. A smbmdev_size of 0 indicates unpopulated.
1066  * WARNING: Some BIOSes appear to export the *maximum* size of the device
1067  * that can appear in the corresponding socket as opposed to the current one.
1068 */
1069 typedef struct smbios_memdevice {
1070     id_t smbmdev_array;          /* handle of physical memory array */
1071     id_t smbmdev_error;          /* handle of memory error data */
1072     uint32_t smbmdev_twidth;     /* total width in bits including ecc */
1073     uint32_t smbmdev_dwidth;     /* data width in bits */
1074     uint64_t smbmdev_size;        /* size in bytes (see note above) */
1075     uint8_t smbmdev_form;        /* form factor */
1076     uint8_t smbmdev_set;         /* set (0x00=none, 0xFF=unknown) */
1077     uint8_t smbmdev_type;        /* memory type */
1078     uint8_t smbmdev_pad;         /* padding */
1079     uint32_t smbmdev_flags;       /* flags (see below) */
1080     uint32_t smbmdev_speed;       /* speed in MHz */
1081     uint32_t smbmdev_speed;       /* speed in nanoseconds */
1082     const char *smbmdev_dloc;     /* physical device locator string */
1083     const char *smbmdev_bloc;     /* physical bank locator string */
1084     uint8_t smbmdev_rank;         /* rank */
1085     uint16_t smbmdev_clkspeed;    /* configured clock speed */
1086     uint16_t smbmdev_minvolt;     /* minimum voltage */
1087     uint16_t smbmdev_maxvolt;     /* maximum voltage */
1088     uint16_t smbmdev_confvolt;    /* configured voltage */
1089 } smbios_memdevice_t;

1089 #define SMB_MDF_OTHER           0x01    /* other */
1090 #define SMB_MDF_UNKNOWN         0x02    /* unknown */
1091 #define SMB_MDF_SIMM            0x03    /* SIMM */
1092 #define SMB_MDF_SIP             0x04    /* SIP */
1093 #define SMB_MDF_CHIP            0x05    /* chip */
1094 #define SMB_MDF_DIP             0x06    /* DIP */
1095 #define SMB_MDF_ZIP             0x07    /* ZIP */
1096 #define SMB_MDF_PROP            0x08    /* proprietary card */
1097 #define SMB_MDF_DIMM            0x09    /* DIMM */
1098 #define SMB_MDF_TSOP            0x0A    /* TSOP */
1099 #define SMB_MDF_CHIPROW         0x0B    /* row of chips */
1100 #define SMB_MDF_RIMM            0x0C    /* RIMM */
1101 #define SMB_MDF_SODIMM          0x0D    /* SODIMM */
1102 #define SMB_MDF_SRRIMM          0x0E    /* SRRIMM */
1103 #define SMB_MDF_FBDIMM          0x0F    /* FBDIMM */

1105 #define SMB_MDT_OTHER           0x01    /* other */
1106 #define SMB_MDT_UNKNOWN         0x02    /* unknown */

```

```

1107 #define SMB_MDT_DRAM            0x03    /* DRAM */
1108 #define SMB_MDT_EDRAM           0x04    /* EDRAM */
1109 #define SMB_MDT_VRAM            0x05    /* VRAM */
1110 #define SMB_MDT_SRAM            0x06    /* SRAM */
1111 #define SMB_MDT_RAM             0x07    /* RAM */
1112 #define SMB_MDT_ROM             0x08    /* ROM */
1113 #define SMB_MDT_FLASH           0x09    /* FLASH */
1114 #define SMB_MDT_EEPROM          0x0A    /* EEPROM */
1115 #define SMB_MDT_FEPROM          0x0B    /* FEPROM */
1116 #define SMB_MDT_EPROM           0x0C    /* EPROM */
1117 #define SMB_MDT_CDRAM           0x0D    /* CDRAM */
1118 #define SMB_MDT_3DRAM           0x0E    /* 3DRAM */
1119 #define SMB_MDT_SDRAM           0x0F    /* SDRAM */
1120 #define SMB_MDT_SGRAM           0x10    /* SGRAM */
1121 #define SMB_MDT_RDRAM           0x11    /* RDRAM */
1122 #define SMB_MDT_DDR             0x12    /* DDR */
1123 #define SMB_MDT_DDR2            0x13    /* DDR2 */
1124 #define SMB_MDT_DDR2FBDIMM      0x14    /* DDR2 FBDIMM */
1125 #define SMB_MDT_DDR3            0x18    /* DDR3 */
1126 #define SMB_MDT_FBD2            0x19    /* FBD2 */

1128 #define SMB_MDF_OTHER           0x0002  /* other */
1129 #define SMB_MDF_UNKNOWN         0x0004  /* unknown */
1130 #define SMB_MDF_FASTPG          0x0008  /* fast-paged */
1131 #define SMB_MDF_STATIC          0x0010  /* static column */
1132 #define SMB_MDF_PSTATIC         0x0020  /* pseudo-static */
1133 #define SMB_MDF_RAMBUS          0x0040  /* RAMBUS */
1134 #define SMB_MDF_SYNC            0x0080  /* synchronous */
1135 #define SMB_MDF_CMOS            0x0100  /* CMOS */
1136 #define SMB_MDF_EDO             0x0200  /* EDO */
1137 #define SMB_MDF_WDRAM           0x0400  /* Window DRAM */
1138 #define SMB_MDF_CDRAM           0x0800  /* Cache DRAM */
1139 #define SMB_MDF_NV              0x1000  /* non-volatile */
1140 #define SMB_MDF_REG              0x2000  /* Registered (Buffered) */
1141 #define SMB_MDF_UNREG           0x4000  /* Unregistered (Unbuffered) */
1142 #define SMB_MDF_LRDIMM          0x8000  /* LRDIMM */

1144 #define SMB_MDR_SINGLE          0x01    /* single */
1145 #define SMB_MDR_DUAL            0x02    /* dual */
1146 #define SMB_MDR_QUAD            0x04    /* quad */
1147 #define SMB_MDR_OCTAL           0x08    /* octal */

1149 /*
1150  * SMBIOS Memory Array Mapped Address. See DSP0134 Section 7.20 for more
1151  * SMBIOS Memory Array Mapped Address. See DSP0134 Section 3.3.20 for more
1152  * information. We convert start/end addresses into addr/size for convenience.
1153 */
1154 typedef struct smbios_memarrmap {
1155     id_t smbmam_array;          /* physical memory array handle */
1156     uint32_t smbmam_width;       /* number of devices that form a row */
1157     uint64_t smbmam_addr;        /* physical address of mapping */
1158     uint64_t smbmam_size;        /* size in bytes of address range */
1159 } smbios_memarrmap_t;

1160 /*
1161  * SMBIOS Memory Device Mapped Address. See DSP0134 Section 7.21 for more
1162  * SMBIOS Memory Device Mapped Address. See DSP0134 Section 3.3.21 for more
1163  * information. We convert start/end addresses into addr/size for convenience.
1164 */
1165 typedef struct smbios_memdevmap {
1166     id_t smbmdev_device;         /* memory device handle */
1167     id_t smbmdev_arrmap;         /* memory array mapped address handle */
1168     uint64_t smbmdev_addr;       /* physical address of mapping */
1169     uint64_t smbmdev_size;       /* size in bytes of address range */
1170     uint8_t smbmdev_rpos;        /* partition row position */
1171     uint8_t smbmdev_ipos;        /* interleave position */

```

```

1171     uint8_t smbmdm_idepth;          /* interleave data depth */
1172 } smbios_memdevmap_t;

1174 /*
1175 * SMBIOS Hardware Security Settings. See DSP0134 Section 7.25 for more
1176 * SMBIOS Hardware Security Settings. See DSP0134 Section 3.3.25 for more
1177 * information. Only one such record will be present in the SMBIOS.
1178 */
1178 typedef struct smbios_hwsec {
1179     uint8_t smbh_pwr_ps;           /* power-on password status */
1180     uint8_t smbh_kbd_ps;           /* keyboard password status */
1181     uint8_t smbh_adm_ps;           /* administrator password status */
1182     uint8_t smbh_pan_ps;           /* front panel reset status */
1183 } smbios_hwsec_t;

1185 #define SMB_HWSEC_PS_DISABLED    0x00    /* password disabled */
1186 #define SMB_HWSEC_PS_ENABLED    0x01    /* password enabled */
1187 #define SMB_HWSEC_PS_NOTIMPL    0x02    /* password not implemented */
1188 #define SMB_HWSEC_PS_UNKNOWN    0x03    /* password status unknown */

1190 /*
1191 * SMBIOS System Boot Information. See DSP0134 Section 7.33 for more
1192 * SMBIOS System Boot Information. See DSP0134 Section 3.3.33 for more
1193 * information. The contents of the data varies by type and is undocumented
1194 * from the perspective of DSP0134 -- it seems to be left as vendor-specific.
1195 * The (D) annotation next to SMB_BOOT_* below indicates possible data payload.
1196 */
1196 typedef struct smbios_boot {
1197     uint8_t smbt_status;           /* boot status code (see below) */
1198     const void *smbt_data;         /* data buffer specific to status */
1199     size_t smbt_size;              /* size of smbt_data buffer in bytes */
1200 } smbios_boot_t;

1202 #define SMB_BOOT_NORMAL          0        /* no errors detected */
1203 #define SMB_BOOT_NOMEDIA         1        /* no bootable media */
1204 #define SMB_BOOT_OSFALL          2        /* normal o/s failed to load */
1205 #define SMB_BOOT_FWHWFAIL        3        /* firmware-detected hardware failure */
1206 #define SMB_BOOT_OSHWFAIL        4        /* o/s-detected hardware failure */
1207 #define SMB_BOOT_USERREQ         5        /* user-requested boot (keystroke) */
1208 #define SMB_BOOT_SECURITY        6        /* system security violation */
1209 #define SMB_BOOT_PREVREQ         7        /* previously requested image (D) */
1210 #define SMB_BOOT_WATCHDOG        8        /* watchdog initiated reboot */
1211 #define SMB_BOOT_RESV_LO         9        /* low end of reserved range */
1212 #define SMB_BOOT_RESV_HI        127       /* high end of reserved range */
1213 #define SMB_BOOT_OEM_LO         128       /* low end of OEM-specific range */
1214 #define SMB_BOOT_OEM_HI         191       /* high end of OEM-specific range */
1215 #define SMB_BOOT_PROD_LO        192       /* low end of product-specific range */
1216 #define SMB_BOOT_PROD_HI        255       /* high end of product-specific range */

1218 /*
1219 * SMBIOS IPMI Device Information. See DSP0134 Section 7.39 and also
1220 * SMBIOS IPMI Device Information. See DSP0134 Section 3.3.39 and also
1221 * Appendix C1 of the IPMI specification for more information on this record.
1222 */
1222 typedef struct smbios_ipmi {
1223     uint_t smbip_type;             /* BMC interface type */
1224     smbios_version_t smbip_vers;   /* BMC's IPMI specification version */
1225     uint32_t smbip_i2c;            /* BMC I2C bus slave address */
1226     uint32_t smbip_bus;            /* bus ID of NV storage device, or -1 */
1227     uint64_t smbip_addr;           /* BMC base address */
1228     uint32_t smbip_flags;          /* flags (see below) */
1229     uint16_t smbip_intr;           /* interrupt number (or zero if none) */
1230     uint16_t smbip_regspacing;     /* i/o space register spacing (bytes) */
1231 } smbios_ipmi_t;

1233 #define SMB_IPMI_T_UNKNOWN      0x00    /* unknown */

```

```

1234 #define SMB_IPMI_T_KCS          0x01    /* KCS: Keyboard Controller Style */
1235 #define SMB_IPMI_T_SMIC         0x02    /* SMIC: Server Mgmt Interface Chip */
1236 #define SMB_IPMI_T_BT           0x03    /* BT: Block Transfer */
1237 #define SMB_IPMI_T_SSIF         0x04    /* SSIF: SMBus System Interface */

1239 #define SMB_IPMI_F_IOADDR        0x01    /* base address is in i/o space */
1240 #define SMB_IPMI_F_INTRSPEC      0x02    /* intr information is specified */
1241 #define SMB_IPMI_F_INTRHIGH     0x04    /* intr active high (else low) */
1242 #define SMB_IPMI_F_INTREDGE     0x08    /* intr is edge triggered (else lvl) */

1244 /*
1245 * SMBIOS Onboard Devices Extended Information. See DSP0134 Section 7.42
1246 * SMBIOS Onboard Devices Extended Information. See DSP0134 Section 3.3.42
1247 * for more information.
1248 */
1248 typedef struct smbios_obdev_ext {
1249     const char *smboe_name;        /* reference designation */
1250     uint8_t smboe_dtype;          /* device type */
1251     uint8_t smboe_dti;            /* device type instance */
1252     uint16_t smboe_sg;            /* segment group number */
1253     uint8_t smboe_bus;            /* bus number */
1254     uint8_t smboe_df;             /* device/function number */
1255 } smbios_obdev_ext_t;
1256 unchanged_portion_omitted

1306 /*
1307 * SMBIOS Interfaces. An SMBIOS image can be opened by either providing a file
1308 * pathname, device pathname, file descriptor, or raw memory buffer. Once an
1309 * image is opened the functions below can be used to iterate over the various
1310 * structures and convert the underlying data representation into the simpler
1311 * data structures described earlier in this header file. The SMB_VERSION
1312 * constant specified when opening an image indicates the version of the ABI
1313 * the caller expects and the DMTF SMBIOS version the client can understand.
1314 * The library will then map older or newer data structures to that as needed.
1315 */

1317 #define SMB_VERSION_23           0x0203    /* SMBIOS encoding for DMTF spec 2.3 */
1318 #define SMB_VERSION_24           0x0204    /* SMBIOS encoding for DMTF spec 2.4 */
1319 #define SMB_VERSION_25           0x0205    /* SMBIOS encoding for DMTF spec 2.5 */
1320 #define SMB_VERSION_26           0x0206    /* SMBIOS encoding for DMTF spec 2.6 */
1321 #define SMB_VERSION_27           0x0207    /* SMBIOS encoding for DMTF spec 2.7 */
1322 #define SMB_VERSION_28           0x0208    /* SMBIOS encoding for DMTF spec 2.8 */
1323 #define SMB_VERSION              SMB_VERSION_28 /* SMBIOS latest version definitions */
1324 #define SMB_VERSION              SMB_VERSION_24 /* SMBIOS latest version definitions */

1325 #define SMB_O_NOCKSUM            0x1        /* do not verify header checksums */
1326 #define SMB_O_NOVERS             0x2        /* do not verify header versions */
1327 #define SMB_O_ZIDS               0x4        /* strip out identification numbers */
1328 #define SMB_O_MASK               0x7        /* mask of valid smbios_*open flags */

1330 #define SMB_ID_NOTSUP            0xFFFF     /* structure is not supported by BIOS */
1331 #define SMB_ID_NONE              0xFFFF     /* structure is a null reference */

1333 #define SMB_ERR                  (-1)        /* id_t value indicating error */

1335 typedef struct smbios_hdl smbios_hdl_t;

1337 typedef struct smbios_struct {
1338     id_t smbstr_id;               /* structure ID handle */
1339     uint_t smbstr_type;           /* structure type */
1340     const void *smbstr_data;      /* structure data */
1341     size_t smbstr_size;           /* structure size */
1342 } smbios_struct_t;

1344 typedef int smbios_struct_f(smbios_hdl_t *,
1345     const smbios_struct_t *, void *);

```

```

1347 extern smbios_hdl_t *smbios_open(const char *, int, int, int *);
1348 extern smbios_hdl_t *smbios_fdopen(int, int, int, int *);
1349 extern smbios_hdl_t *smbios_bufopen(const smbios_entry_t *,
1350     const void *, size_t, int, int, int *);

1352 extern const void *smbios_buf(smbios_hdl_t *);
1353 extern size_t smbios_buflen(smbios_hdl_t *);

1355 extern void smbios_checksum(smbios_hdl_t *, smbios_entry_t *);
1356 extern int smbios_write(smbios_hdl_t *, int);
1357 extern void smbios_close(smbios_hdl_t *);

1359 extern int smbios_errno(smbios_hdl_t *);
1360 extern const char *smbios_errmsg(int);

1362 extern int smbios_lookup_id(smbios_hdl_t *, id_t, smbios_struct_t *);
1363 extern int smbios_lookup_type(smbios_hdl_t *, uint_t, smbios_struct_t *);
1364 extern int smbios_iter(smbios_hdl_t *, smbios_struct_t *, void *);

1366 extern void smbios_info_smbios(smbios_hdl_t *, smbios_entry_t *);
1367 extern int smbios_info_common(smbios_hdl_t *, id_t, smbios_info_t *);
1368 extern int smbios_info_contains(smbios_hdl_t *, id_t, uint_t, id_t *);
1369 extern id_t smbios_info_bios(smbios_hdl_t *, smbios_bios_t *);
1370 extern id_t smbios_info_system(smbios_hdl_t *, smbios_system_t *);
1371 extern int smbios_info_bboard(smbios_hdl_t *, id_t, smbios_bboard_t *);
1372 extern int smbios_info_chassis(smbios_hdl_t *, id_t, smbios_chassis_t *);
1373 extern int smbios_info_processor(smbios_hdl_t *, id_t, smbios_processor_t *);
1374 extern int smbios_info_extprocessor(smbios_hdl_t *, id_t,
1375     smbios_processor_ext_t *);
1376 extern int smbios_info_cache(smbios_hdl_t *, id_t, smbios_cache_t *);
1377 extern int smbios_info_port(smbios_hdl_t *, id_t, smbios_port_t *);
1378 extern int smbios_info_extport(smbios_hdl_t *, id_t, smbios_port_ext_t *);
1379 extern int smbios_info_slot(smbios_hdl_t *, id_t, smbios_slot_t *);
1380 extern int smbios_info_obdevs(smbios_hdl_t *, id_t, int, smbios_obdev_t *);
1381 extern int smbios_info_obdevs_ext(smbios_hdl_t *, id_t, smbios_obdev_ext_t *);
1382 extern int smbios_info_strtab(smbios_hdl_t *, id_t, int, const char *[]);
1383 extern id_t smbios_info_lang(smbios_hdl_t *, smbios_lang_t *);
1384 extern id_t smbios_info_eventlog(smbios_hdl_t *, smbios_evlog_t *);
1385 extern int smbios_info_memarray(smbios_hdl_t *, id_t, smbios_memarray_t *);
1386 extern int smbios_info_extmemarray(smbios_hdl_t *, id_t,
1387     smbios_memarray_ext_t *);
1388 extern int smbios_info_memarrmap(smbios_hdl_t *, id_t, smbios_memarrmap_t *);
1389 extern int smbios_info_memdevice(smbios_hdl_t *, id_t, smbios_memdevice_t *);
1390 extern int smbios_info_extmemdevice(smbios_hdl_t *, id_t,
1391     smbios_memdevice_ext_t *);
1392 extern int smbios_info_memdevmap(smbios_hdl_t *, id_t, smbios_memdevmap_t *);
1393 extern id_t smbios_info_hwsec(smbios_hdl_t *, smbios_hwsec_t *);
1394 extern id_t smbios_info_boot(smbios_hdl_t *, smbios_boot_t *);
1395 extern id_t smbios_info_ipmi(smbios_hdl_t *, smbios_ipmi_t *);
1396 extern int smbios_info_pciexrc(smbios_hdl_t *, id_t, smbios_pciexrc_t *);

1398 extern const char *smbios_psn(smbios_hdl_t *);
1399 extern const char *smbios_csn(smbios_hdl_t *);

1401 #ifndef _KERNEL
1402 /*
1403  * The smbios_*_desc() and smbios_*_name() interfaces can be used for utilities
1404  * such as smbios(1M) that wish to decode SMBIOS fields for humans. The _desc
1405  * functions return the comment string next to the #defines listed above, and
1406  * the _name functions return the appropriate #define identifier itself.
1407  */
1408 extern const char *smbios_bboard_flag_desc(uint_t);
1409 extern const char *smbios_bboard_flag_name(uint_t);
1410 extern const char *smbios_bboard_type_desc(uint_t);

```

```

1412 extern const char *smbios_bios_flag_desc(uint64_t);
1413 extern const char *smbios_bios_flag_name(uint64_t);

1415 extern const char *smbios_bios_xb1_desc(uint_t);
1416 extern const char *smbios_bios_xb1_name(uint_t);
1417 extern const char *smbios_bios_xb2_desc(uint_t);
1418 extern const char *smbios_bios_xb2_name(uint_t);

1420 extern const char *smbios_boot_desc(uint_t);

1422 extern const char *smbios_cache_assoc_desc(uint_t);
1423 extern const char *smbios_cache_ctype_desc(uint_t);
1424 extern const char *smbios_cache_ctype_name(uint_t);
1425 extern const char *smbios_cache_ecc_desc(uint_t);
1426 extern const char *smbios_cache_flag_desc(uint_t);
1427 extern const char *smbios_cache_flag_name(uint_t);
1428 extern const char *smbios_cache_loc_desc(uint_t);
1429 extern const char *smbios_cache_logical_desc(uint_t);
1430 extern const char *smbios_cache_mode_desc(uint_t);

1432 extern const char *smbios_chassis_state_desc(uint_t);
1433 extern const char *smbios_chassis_type_desc(uint_t);

1435 extern const char *smbios_evlog_flag_desc(uint_t);
1436 extern const char *smbios_evlog_flag_name(uint_t);
1437 extern const char *smbios_evlog_format_desc(uint_t);
1438 extern const char *smbios_evlog_method_desc(uint_t);

1440 extern const char *smbios_ipmi_flag_name(uint_t);
1441 extern const char *smbios_ipmi_flag_desc(uint_t);
1442 extern const char *smbios_ipmi_type_desc(uint_t);

1444 extern const char *smbios_hwsec_desc(uint_t);

1446 extern const char *smbios_memarray_loc_desc(uint_t);
1447 extern const char *smbios_memarray_use_desc(uint_t);
1448 extern const char *smbios_memarray_ecc_desc(uint_t);

1450 extern const char *smbios_memdevice_form_desc(uint_t);
1451 extern const char *smbios_memdevice_type_desc(uint_t);
1452 extern const char *smbios_memdevice_flag_name(uint_t);
1453 extern const char *smbios_memdevice_flag_desc(uint_t);
1454 extern const char *smbios_memdevice_rank_desc(uint_t);

1456 extern const char *smbios_port_conn_desc(uint_t);
1457 extern const char *smbios_port_type_desc(uint_t);

1459 extern const char *smbios_processor_family_desc(uint_t);
1460 extern const char *smbios_processor_status_desc(uint_t);
1461 extern const char *smbios_processor_type_desc(uint_t);
1462 extern const char *smbios_processor_upgrade_desc(uint_t);
1463 extern const char *smbios_processor_core_flag_name(uint_t);
1464 extern const char *smbios_processor_core_flag_desc(uint_t);

1466 extern const char *smbios_slot_type_desc(uint_t);
1467 extern const char *smbios_slot_width_desc(uint_t);
1468 extern const char *smbios_slot_usage_desc(uint_t);
1469 extern const char *smbios_slot_length_desc(uint_t);
1470 extern const char *smbios_slot_ch1_desc(uint_t);
1471 extern const char *smbios_slot_ch1_name(uint_t);
1472 extern const char *smbios_slot_ch2_desc(uint_t);
1473 extern const char *smbios_slot_ch2_name(uint_t);

1475 extern const char *smbios_type_desc(uint_t);
1476 extern const char *smbios_type_name(uint_t);

```

```
1478 extern const char *smbios_system_wakeup_desc(uint_t);
1479 #endif /* !_KERNEL */

1481 #ifdef _KERNEL
1482 /*
1483  * For SMBIOS clients within the kernel itself, ksmbios is used to refer to
1484  * the kernel's current snapshot of the SMBIOS, if one exists, and the
1485  * ksmbios_flags tunable is the set of flags for use with smbios_open().
1486  */
1487 extern smbios_hdl_t *ksmbios;
1488 extern int ksmbios_flags;
1489 #endif /* _KERNEL */

1491 #ifdef __cplusplus
1492 }
_____unchanged_portion_omitted_
```

new/usr/src/uts/common/sys/smbios_impl.h

1

```
*****
19937 Thu Mar 26 17:12:01 2015
new/usr/src/uts/common/sys/smbios_impl.h
5094 Update libsbios with recent items
Reviewed by: Dan McDonald <danmcd@omniti.com>
Reviewed by: Josef 'Jeff' Sipek <jeffpc@josefsipek.net>
Reviewed by: Garrett D'Amore <garrett@damore.org>
Reviewed by: Robert Mustacchi <rm@joyent.com>
*****

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

22 /*
23  * Copyright 2015 OmniTI Computer Consulting, Inc. All rights reserved.
24  * Copyright 2010 Sun Microsystems, Inc. All rights reserved.
25  * Use is subject to license terms.
26 */

28 /*
29  * This header file defines the implementation structures for the SMBIOS access
30  * library, libsbios, and an equivalent kernel module. Clients should use
31  * the <smbios.h> or <sys/smbios.h> header files to access DMTF SMBIOS
32  * information, NOT these underlying implementation structures from the spec.
33  * In short, do not user this header file or these routines for any purpose.
34 */

36 #ifndef _SYS_SMBIOS_IMPL_H
37 #define _SYS_SMBIOS_IMPL_H

39 #include <sys/smbios.h>
40 #include <sys/sysmacros.h>

42 #ifdef _KERNEL
43 #include <sys/system.h>
44 #else
45 #include <strings.h>
46 #include <stddef.h>
47 #endif

49 #ifdef __cplusplus
50 extern "C" {
51 #endif

53 #pragma pack(1)

55 typedef struct smb_header {
56     uint8_t smbh_type;          /* structure type (SMB_TYPE_* value) */
57     uint8_t smbh_len;           /* length in bytes of formatted area */
```

new/usr/src/uts/common/sys/smbios_impl.h

2

```
58     uint16_t smbh_hdl;         /* structure handle */
59 } smb_header_t;
    unchanged_portion_omitted

118 /* WARNING: the argument is evaluated three times! */
119 #define SMB_CH_SKU(smbcp) ((char *) \
120     (smbcp)->smbch_cv + ((smbcp)->smbch_cn * (smbcp)->smbch_cm))
121 #define SMB_CHT_LOCK        0x80    /* lock bit within smbch_type */

123 typedef struct smb_processor {
124     smb_header_t smbpr_hdr;        /* structure header */
125     uint8_t smbpr_socket;          /* socket designation */
126     uint8_t smbpr_type;            /* processor type (see <smbios.h>) */
127     uint8_t smbpr_family;          /* processor family (see <smbios.h>) */
128     uint8_t smbpr_manufacturer;    /* manufacturer */
129     uint64_t smbpr_cpuid;          /* processor cpuid information */
130     uint8_t smbpr_version;         /* version */
131     uint8_t smbpr_voltage;         /* voltage */
132     uint16_t smbpr_clkspeed;       /* external clock speed in MHz */
133     uint16_t smbpr_maxspeed;       /* maximum speed in MHz */
134     uint16_t smbpr_curspeed;       /* current speed in MHz */
135     uint8_t smbpr_status;          /* status (see <smbios.h>) */
136     uint8_t smbpr_upgrade;         /* upgrade */
137     uint16_t smbpr_l1cache;        /* L1 cache handle (if any) */
138     uint16_t smbpr_l2cache;        /* L2 cache handle (if any) */
139     uint16_t smbpr_l3cache;        /* L3 cache handle (if any) */
140     uint8_t smbpr_serial;          /* serial number */
141     uint8_t smbpr_asset;           /* asset tag */
142     uint8_t smbpr_part;            /* part number */
143     uint8_t smbpr_corecount;       /* number of cores per socket */
144     uint8_t smbpr_coresenabled;    /* number of enabled cores per socket */
145     uint8_t smbpr_threadcount;     /* number of threads per socket */
146     uint16_t smbpr_cflags;         /* cpu characteristics (see <smbios.h>) */
147     uint16_t smbpr_family2;        /* processor family2 (see <smbios.h>) */
148 } smb_processor_t;
    unchanged_portion_omitted

164 /*
165  * Convert encoded cache size to bytes: DSP0134 Section 7.8 explains the
166  * Convert encoded cache size to bytes: DSP0134 Section 3.3.8 explains the
167  * encoding. The highest bit is 0 for 1k units, 1 for 64k units, and this
168  * macro decodes the value into bytes for exporting to our clients.
169 */
169 #define SMB_CACHE_SIZE(s) (((s) & 0x8000) ? \
170     ((uint32_t)((s) & 0x7FFF) * 64 * 1024) : ((uint32_t)(s) * 1024))

172 #define SMB_CACHE_CFG_MODE(c)      (((c) >> 8) & 3)
173 #define SMB_CACHE_CFG_ENABLED(c)   (((c) >> 7) & 1)
174 #define SMB_CACHE_CFG_LOCATION(c)  (((c) >> 5) & 3)
175 #define SMB_CACHE_CFG_SOCKETED(c)  (((c) >> 3) & 1)
176 #define SMB_CACHE_CFG_LEVEL(c)     (((c) & 7) + 1)

178 typedef struct smb_port {
179     smb_header_t smbpo_hdr;        /* structure header */
180     uint8_t smbpo_iref;            /* internal reference designator */
181     uint8_t smbpo_itype;           /* internal connector type */
182     uint8_t smbpo_eref;            /* external reference designator */
183     uint8_t smbpo_etype;           /* external connector type */
184     uint8_t smbpo_ptype;           /* port type */
185 } smb_port_t;
    unchanged_portion_omitted

202 typedef struct smb_obdev {
203     uint8_t smbob_type;            /* encoded type and enable bit */
204     uint8_t smbob_name;            /* description string */
205     uint8_t smbob_name;            /* description string */
```

```

205 } smb_obdev_t;
    unchanged_portion_omitted

237 typedef struct smb_memarray {
238     smb_header_t smbarr_hdr;        /* structure header */
239     uint8_t smbarr_loc;             /* location */
240     uint8_t smbarr_use;             /* use */
241     uint8_t smbarr_ecc;             /* error detect/correct mechanism */
242     uint32_t smbarr_cap;             /* maximum capacity */
243     uint16_t smbarr_err;            /* error handle */
244     uint16_t smbarr_ndevs;          /* number of slots or sockets */
245     uint64_t smbarr_extcap;        /* extended maximum capacity */
246 } smb_memarray_t;

248 typedef struct smb_memarrmap {
249     smb_header_t smbamap_hdr;       /* structure header */
250     uint32_t smbamap_start;          /* starting address in kilobytes */
251     uint32_t smbamap_end;            /* ending address in kilobytes */
252     uint16_t smbamap_array;          /* physical memory array handle */
253     uint8_t smbamap_width;           /* partition width */
254     uint64_t smbamap_extstart;      /* extended starting address in bytes */
255     uint64_t smbamap_extend;       /* extended ending address in bytes */
256 } smb_memarrmap_t;

258 typedef struct smb_memdevice {
259     smb_header_t smbmddev_hdr;       /* structure header */
260     uint16_t smbmddev_array;          /* array handle */
261     uint16_t smbmddev_error;          /* error handle */
262     uint16_t smbmddev_twidth;         /* total width */
263     uint16_t smbmddev_dwidth;         /* data width */
264     uint16_t smbmddev_size;           /* size in either K or MB */
265     uint8_t smbmddev_form;            /* form factor */
266     uint8_t smbmddev_set;             /* device set */
267     uint8_t smbmddev_dloc;            /* device locator */
268     uint8_t smbmddev_bloc;            /* bank locator */
269     uint8_t smbmddev_type;            /* memory type */
270     uint16_t smbmddev_flags;          /* detail flags */
271     uint16_t smbmddev_speed;          /* speed in MHz */
272     uint8_t smbmddev_manufacturer;    /* manufacturer */
273     uint8_t smbmddev_serial;          /* serial number */
274     uint8_t smbmddev_asset;           /* asset tag */
275     uint8_t smbmddev_part;            /* part number */
276     uint8_t smbmddev_attr;          /* attributes */
277     uint32_t smbmddev_extsize;      /* extended size */
278     uint16_t smbmddev_clksped;      /* configured clock speed */
279     uint16_t smbmddev_minvolt;      /* minimum voltage */
280     uint16_t smbmddev_maxvolt;      /* maximum voltage */
281     uint16_t smbmddev_confvolt;    /* configured voltage */
282 } smb_memdevice_t;

284 #define SMB_MDS_KBYTES      0x8000 /* size in specified in kilobytes */

286 typedef struct smb_memdevmap {
287     smb_header_t smbmdmap_hdr;       /* structure header */
288     uint32_t smbmdmap_start;          /* starting address in kilobytes */
289     uint32_t smbmdmap_end;            /* ending address in kilobytes */
290     uint16_t smbmdmap_device;         /* memory device handle */
291     uint16_t smbmdmap_array;          /* memory array mapped address handle */
292     uint8_t smbmdmap_rpos;            /* row position */
293     uint8_t smbmdmap_ipos;            /* interleave position */
294     uint8_t smbmdmap_idpth;           /* interleave depth */
295     uint64_t smbmdmap_extstart;      /* extended starting address */
296     uint64_t smbmdmap_extend;       /* extended ending address */
297 } smb_memdevmap_t;
    unchanged_portion_omitted

```