E-SMI Library: EPYC™ Systems Management Interface Library

Generated by Doxygen 1.9.1

Chapter 1

EPYC™ System Management Interface (E-SMI) In-band Library

The EPYC™ System Management Interface In-band Library, or E-SMI library, is part of the EPYC™ System Management Inband software stack. It is a C library for Linux that provides a user space interface to monitor and control the CPU's power, energy, performance and other system management features.

1.1 Important note about Versioning and Backward Compatibility

The E-SMI library is currently under development, and therefore subject to change at the API level. The intention is to keep the API as stable as possible while in development, but in some cases we may need to break backwards compatibility in order to achieve future stability and usability. Following Semantic Versioning rules, while the E-SMI library is in a high state of change, the major version will remain 0, and achieving backward compatibility may not be possible.

Once new development has leveled off, the major version will become greater than 0, and backward compatibility will be enforced between major versions.

1.2 Building E-SMI

1.2.1 Dowloading the source

The source code for E-SMI library is available at `Github`.

1.2.2 Directory stucture of the source

Once the E-SMI library source has been cloned to a local Linux machine, the directory structure of source is as below:

- `$ docs/` Contains Doxygen configuration files and Library descriptions
- `$ tools/` Contains e-smi tool, based on the E-SMI library
- `$ include/` Contains the header files used by the E-SMI library
- `$ src/` Contains library E-SMI source
- `$ cmake_modules/` Contains helper utilities for determining package and library version
- `$ DEBIAN/` Contains debian pre and post installation scripts
- `$ RPM/` Contains rpm pre and post installation scripts

1.2.3 Building the library and tool

Building the library is achieved by following the typical CMake build sequence, as below

- `$ cd <location of root of E-smi library>`
- `$ mkdir -p build`
- `$ cd build`
- `$ cmake ../`

Building the library for static linking

Building the library as a static(.a) along with shared libraries(.so) is achieved by following sequence. The static library is part of RPM and DEB package when compiled with cmake as below and built with 'make package'.

- `$ cmake -DENABLE_STATIC_LIB=1 ../`
- `$ make`

Building the library and tool using clang compiler

- `$ cmake -DUSE_CLANG=1 ../`
- `$ make`
  The built library `libe_smi64_static.a`, `libe_smi64.so.X.Y` and `esmi_tool` will appear in the `build` directory
- `$ sudo make install`
  By default library file, header and tool are installed at /opt/e-sms. To change the default installation path, build with cmake -DCMAKE_INSTALL_PREFIX=xxxx. Library will be installed at xxxx/lib, tool will be installed at xxxx/bin, header will be installed at xxxx/include and doc will be installed at xxxx/e-sms/doc. Example↩ : If -DCMAKE_INSTALL_PREFIX=/usr/local then esmi lib, esmi_tool binary and headers are installed at /usr/local/lib, /usr/local/bin, /usr/local/include respectively.

`Note:` Library is dependent on amd_hsmp.h header and without this, compilation will break. Please follow the instruction in "Kernel dependencies" section

1.2.4 Building the Documentation

The documentation PDF file can be built with the following steps (continued from the steps above) `$ make doc` Upon a successful build, the `ESMI_Manual.pdf` and `ESMI_IB_Release_Notes.pdf` will be copied to the top directory of the source.

1.2.5 Building the package

The RPM and DEB packages can be created with the following steps (continued from the steps above): `$ make package`

1.3 Kernel version dependency

- Family 0x19 model 00-0fh a0-afh are supported from v5.16-rc7 onwards
- Family 0x19 model 90-9fh are supported from v6.6-rc1 onwards
- Family 0x1A model 00-1fh are supported from v6.5-rc5 onwards

1.4 Kernel driver dependencies

The E-SMI Library depends on the following device drivers from Linux to manage the system management features.

1.4.1 amd_hsmp driver

This is used to monitor and manage power metrics, boostlimits and other system management features. The power metrics, boostlimits and other features are managed by the SMU(System Management Unit of the processor) firmware and exposed via PCI config space and accessed through "Host System Management Port(HSMP)" at host/cpu side. AMD provides Linux kernel module(amd_hsmp) exposing this information to the user-space via ioctl interface.

- amd_hsmp driver is accepted in upstream kernel and is available at linux tree at drivers/platform/x86/amd/hsmp.↩ c from version 5.17.rc1 onwards either it can be compiled as part of kernel as a module or built in driver or as an out of tree module which is available at `https://github.com/amd/amd_hsmp.git`
- E-smi compilation has dependency on amd_hsmp header file from uapi header of amd_hsmp driver. It should be available at
  - /usr/include/asm/ on RHEL systems
  - /usr/include/x86_64-linux-gnu/asm/ on Ubuntu systems. If its not present, it can be copied from `amd↩ _hsmp` github repo or from the kernel source arch/x86/include/uapi/asm/amd_hsmp.h
- There is always a dependency between E-smi and amd_hsmp driver versions. The new features of E-smi work only if there is a matching HSMP driver.

1.4.2 amd_hsmp/msr_safe/amd_energy/msr

One of these drivers is needed to monitor energy counters.

- AMD family 19h, model 00-0fh and 30-3fh
  - These processors support energy monitoring through 32 bit RAPL MSR registers.
  - amd_energy driver, an out of tree kernel module, hosted at `amd_energy` can report per core and per socket counters via the HWMON sysfs entries.
  - This driver provides accumulation of energy for avoiding wrap around problem.
  - This is the only supported energy driver for 32bit RAPLS
- AMD family 19h, model 10-1fh, a0-afh and 90-9fh, AMD family 0x1A, model 00-1fh
  - These processors support energy monitoring through 64 bit RAPL MSR registers.
  - Because of 64 bit registers, there is no accumulation of energy needed.
  - For these processors either `msr_safe`, `amd_energy` or kernel's default msr driver can be used.

- AMD family 1Ah, model 0x00-0x1f support RAPL reading using HSMP mailbox.
  - For these processors either amd_hsmp driver or msr_safe driver or amd_energy driver or msr driver can be used.
  - The order of checking for the availability of drivers in e-smi is as follows.
    - If amd_hsmp driver is present and supports RAPL reading, this is used for reading energy.
    - If amd_hsmp driver is not present/not supports energy reading, and msr-safe driver is present, this is used for reading energy. Msr-safe driver needs allowlist file to be written to "/dev/cpu/msr_↩ allowlist" for allowing the read of those specific msr registers. Please follow below steps or use the tool option "writemsrallowlist" to write the allowlist file. Create "amd_allowlist" file with below contents and run the command "sudo su" and "cat amd_allowlist > /dev/cpu/msr_allowlist".
      - 0xC0010299 0x0000000000000000 # "ENERGY_PWR_UNIT_MSR"
      - 0xC001029A 0x0000000000000000 # "ENERGY_CORE_MSR"
      - 0xC001029B 0x0000000000000000 # "ENERGY_PKG_MSR"
      - Note: The first column above indicates MSR register address and 2nd column indicates write mask and the third coulmn is name of the register.
    - If msr_safe driver is not present, amd_energy driver is present, this is used for reading energy.
    - If msr_safe driver or amd_energy driver not present, msr driver will be used for reading energy.
    - Any one of msr_safe/amd_energy/msr driver is sufficient

1.5 BIOS dependency

- To get HSMP working. PCIe interface needs to be enabled in the BIOS. On the reference BIOS please follow the sequence below for enabling HSMP.

  Advanced > AMD CBS > NBIO Common Options > SMU Common Options > HSMP Support

  BIOS Default: “Auto” (Disabled) to **BIOS Default: "Enabled"**

  If the above HSMP support option is disabled, the related E-SMI APIs will return -ETIMEDOUT. The latest BIOS supports probing of HSMP driver through ACPI device. The ACPI supported `amd_hsmp` driver version is 2.2

1.6 Supported hardware

- AMD Zen3 based CPU Family `19h` Models `0h-Fh` and `30h-3Fh`.
- AMD Zen4 based CPU Family `19h` Models `10h-1Fh` and `A0-AFh`.
- AMD Zen4 based CPU Family `19h` Models `90-9Fh`.
- AMD Zen5 based CPU Family `1Ah` Models `00-1Fh`.
- AMD Zen6 based CPU Family `1Ah` Models `50-5Fh`.

1.7 Additional required software for building

In order to build the E-SMI library, the following components are required. Note that the software versions listed are what is being used in development. Earlier versions are not guaranteed to work:

- CMake (v3.5.0)

- gcc, g++, make
- build-essential

In order to build the latest documentation, the following are required:

- DOxygen (1.8.13)
- latex (pdfTeX 3.14159265-2.6-1.40.18)

1.8 Library Usage Basics

Many of the functions in the library take a "core/socket index". The core/socket index is a number greater than or equal to 0, and less than the number of cores/sockets on the system. Number of cores/sockets in a system can be obtained from esmi library APIs.

1.8.1 Hello E-SMI

The only required E-SMI call for any program that wants to use E-SMI is the `esmi_init()` call. This call initializes some internal data structures that will be used by subsequent E-SMI calls.

When E-SMI is no longer being used, `esmi_exit()` should be called. This provides a way to do any releasing of resources that E-SMI may have held. In many cases, this may have no effect, but may be necessary in future versions of the library.

Below is a simple "Hello World" type program that display the Average Power of Sockets.

```
#include <stdio.h>
#include <stdint.h>
#include <e_smi/e_smi.h>
#include <e_smi/e_smi_monitor.h>
int main()
{
    esmi_status_t ret;
    unsigned int i;
    uint32_t power;
    uint32_t total_sockets = 0;
    ret = esmi_init();
    if (ret != ESMI_SUCCESS) {
        printf("ESMI Not initialized, drivers not found.\n"
            "Err[%d]: %s\n", ret, esmi_get_err_msg(ret));
        return ret;
    }
    total_sockets = esmi_number_of_sockets_get();
    for (i = 0; i < total_sockets; i++) {
        power = 0;
        ret = esmi_socket_power_get(i, &power);
        if (ret != ESMI_SUCCESS) {
            printf("Failed to get socket[%d] avg_power, "
                "Err[%d]:%s\n", i, ret, esmi_get_err_msg(ret));
        }
        printf("socket_%d_avgpower = %.3f Watts\n",
            i, (double)power/1000);
    }
    esmi_exit();
    return ret;
}
```

1.9 Tool Usage

E-SMI tool is a C program based on the E-SMI In-band Library, the executable "e_smi_tool" will be generated in the build/ folder. This tool provides options to Monitor and Control System Management functionality.

Below is a sample usage to dump core and socket metrics

```
e_smi_library/b$ sudo ./e_smi_tool
============================= E-SMI ===================================
--------------------------------------
| CPU Family            | 0x19 (25 ) |
| CPU Model             | 0x10 (16 ) |
| NR_CPUS               | 384        |
| NR_SOCKETS            | 2          |
| THREADS PER CORE      | 2 (SMT ON) |
--------------------------------------
------------------------------------------------------------------------
| Sensor Name                    | Socket 0          | Socket 1          |
------------------------------------------------------------------------
| Energy (K Joules)              | 14437.971         | 14087.151         |
| Power (Watts)                  | 174.290           | 169.630           |
| PowerLimit (Watts)             | 400.000           | 320.000           |
| PowerLimitMax (Watts)          | 400.000           | 320.000           |
| C0 Residency (%)               | 0                 | 0                 |
| DDR Bandwidth                  |                   |                   |
|       DDR Max BW (GB/s)        | 58                | 58                |
|       DDR Utilized BW (GB/s)   | 0                 | 0                 |
|       DDR Utilized Percent(%)  | 0                 | 0                 |
| Current Active Freq limit      |                   |                   |
|        Freq limit (MHz)        | 3500              | 3500              |
|        Freq limit source       | Refer below[*0]   | Refer below[*1]   |
| Socket frequency range         |                   |                   |
|        Fmax (MHz)              | 3500              | 3500              |
|        Fmin (MHz)              | 400               | 400               |
------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------
| CPU energies in Joules:
          |
| cpu [  0] :    645.992    181.415    171.678    165.577    161.001    158.397    161.333    151.716
          |
| cpu [  8] :     88.197     79.306     73.860     73.015     72.960     69.293     67.871     78.895
          |
| cpu [ 16] :     70.376     71.231     61.756     63.061     80.656     73.360     69.566     69.969
          |
| cpu [ 24] :     67.054     65.621     64.468     66.346     64.344     64.310     71.548     65.579
          |
| cpu [ 32] :     65.731     62.931     65.526     69.765     69.050     65.782     70.630     65.282
          |
| cpu [ 40] :     69.608     67.261     63.765     69.477     68.677     63.145     62.451    159.949
          |
| cpu [ 48] :     70.810     73.084     64.584     62.966     66.581     65.620     62.381     65.602
          |
| cpu [ 56] :     72.804     70.842     69.651     64.990     63.924     66.468     63.401    296.924
          |
| cpu [ 64] :     64.693     62.723     65.057     62.515     60.091     60.422     62.217     66.552
          |
| cpu [ 72] :     81.746     70.622     68.848    301.949     78.974     68.130     68.141     65.693
          |
| cpu [ 80] :     77.475     72.441     81.296     71.441     71.988     75.237     73.986     69.467
          |
| cpu [ 88] :     73.385     69.277     61.759     61.060     62.834     60.681     62.835     62.703
          |
| cpu [ 96] :    142.718    139.519    134.449    134.097    135.045    140.307    140.553    137.153
          |
| cpu [104] :     66.016     66.736     62.224     67.137     64.881     70.592     64.701     64.056
          |
| cpu [112] :     70.791     69.107     70.638     69.998     68.199     65.263     70.638     72.557
          |
| cpu [120] :     94.391     94.151     71.881     66.493     64.653     66.141     66.132     69.593
          |
| cpu [128] :     65.800     64.742     63.130     61.771     65.416     66.205     64.663     71.349
          |
| cpu [136] :     72.183     66.754     67.090     63.343     69.450     67.979     68.285     70.478
          |
| cpu [144] :     68.281     63.809     62.717     63.348     71.164     72.289     65.516     65.513
          |
| cpu [152] :     74.588     69.074     66.711     66.011     67.896     65.933     67.031     65.474
          |
| cpu [160] :     66.668     62.996     65.945     63.734     64.060     68.597     76.405     91.436
          |
| cpu [168] :     77.658     70.085     67.025     68.951     64.678     64.821     65.031     71.694
          |
| cpu [176] :     72.782     89.196     74.777     73.703     66.247     65.419     64.748     63.978
          |
| cpu [184] :     63.887     66.080     64.042     65.151     69.661     74.616     63.834     69.824
          |
------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------
| CPU boostlimit in MHz:
          |
| cpu [  0] : 3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [ 16] : 3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [ 32] : 3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [ 48] : 3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [ 64] : 3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [ 80] : 3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [ 96] : 3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [112] : 3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [128] : 3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [144] : 3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [160] : 3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [176] : 3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500
          |
------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------
| CPU core clock current frequency limit in MHz:
          |
| cpu [  0] : 3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [ 16] : NA    NA    NA    NA    NA    NA    NA    NA    3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [ 32] : 3500  3500  3500  3500  3500  3500  3500  3500  NA    NA    NA    NA    NA    NA    NA    NA
          |
| cpu [ 48] : 3500  3500  3500  3500  3500  3500  3500  3500  NA    NA    NA    NA    NA    NA    NA    NA
          |
| cpu [ 64] : NA    NA    NA    NA    NA    NA    NA    NA    3500  3500  3500  3500  3500  3500  3500  3500
          |
| cpu [ 80] : NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
          |
| cpu [ 96] : NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
          |
| cpu [112] : NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
          |
| cpu [128] : NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
          |
| cpu [144] : NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
          |
| cpu [160] : NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
          |
| cpu [176] : NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
          |
------------------------------------------------------------------------------------------------------------------
*0 Frequency limit source names:
OPN Max
*1 Frequency limit source names:
OPN Max
Try `./e_smi_tool --help' for more information.
```

For detailed and up to date usage information, we recommend consulting the help:

For convenience purposes, following is the output from the -h flag on hsmp protocol version 7 based system:

```
============================= E-SMI ===================================
Output Option<s>:
  -h, --help                                                 Show this help message
  -A, --showall                                              Show all esmi parameter values
  -V  --version                                              Show e-smi library version
  --testmailbox [SOCKET] [VALUE<0-0xFFFFFFFF>]               Test HSMP mailbox interface
  --writemsrallowlist                                        Write msr-safe allowlist file
  --json                                                     Print output on console as json
       format[applicable only for get commands]
  --csv                                                      Print output on console as csv
       format[applicable only for get commands]
  --initialdelay [INITIAL_DELAY] <TIME_RANGE<ms,s,m,h,d»    Initial delay before start of execution
  --loopdelay    [LOOP_DELAY]    <TIME_RANGE<ms,s,m,h,d»    Loop delay before executing each loop
  --loopcount    [LOOP_COUNT]                                Set the loop count to the specified
       value[pass "-1" for infinite loop]
  --stoploop     [STOPLOOP_FILE_NAME]                        Set the StopLoop file name, loop will stop
       once the stoploop file is available
  --printonconsole [ENABLE_PRINT<0-1>]                          Print output on console if set to 1, or 0 to
       suppress the console output
  --log [LOG_FILE_NAME]                                         Set the Log file name, in which the data
       collected need to be logged
Get Option<s>:
  --showcoreenergy [CORE]                                       Show energy for a given CPU (Joules)
  --showsockenergy                                              Show energy for all sockets (KJoules)
  --showsockpower                                               Show power metrics for all sockets (Watts)
  --showcorebl [CORE]                                           Show BoostLimit for a given CPU (MHz)
  --showsockc0res [SOCKET]                                      Show C0Residency for a given socket (%%)
  --showsmufwver                                                Show SMU FW Version
  --showhsmpdriverver                                           Show HSMP Driver Version
  --showhsmpprotover                                            Show HSMP Protocol Version
  --showprochotstatus                                           Show HSMP PROCHOT status for all sockets
  --showclocks                                                  Show Clock Metrics (MHz) for all sockets
  --showddrbw                                                   Show DDR bandwidth details (Gbps)
  --showdimmtemprange [SOCKET] [DIMM_ADDR]                      Show dimm temperature range and refresh rate
       for a given socket and dimm address
  --showdimmthermal [SOCKET] [DIMM_ADDR]                        Show dimm thermal values for a given socket
       and dimm address
  --showdimmpower [SOCKET] [DIMM_ADDR]                          Show dimm power consumption for a given
       socket and dimm address
  --showcclkfreqlimit [CORE]                                    Show current clock frequency limit(MHz) for
       a given core
  --showsvipower                                                Show svi based power telemetry of all rails
       for all sockets
  --showiobw [SOCKET] [LINK<P0-P2,P4-P5,G0-G2>]                 Show IO aggregate bandwidth for a given
       socket and linkname
  --showlclkdpmlevel [SOCKET] [NBIOID<0-3>]                     Show lclk dpm level for a given nbio in a
       given socket
  --showsockclkfreqlimit [SOCKET]                               Show current clock frequency limit(MHz) for
       a given socket
  --showxgmibw [SOCKET] [LINK<P0-P2,G0-G2>] [BW<AGG_BW,RD_BW,WR_BW>]
                                                                Show xGMI bandwidth for a given socket,
       linkname and bwtype
  --showcurrpwrefficiencymode [SOCKET]                          Show current power effciency mode
  --showcpurailisofreqpolicy [SOCKET]                           Show current CPU ISO frequency policy
  --showdfcstatectrl [SOCKET]                                   Show current DF C-state status
  --getapbstatus [SOCKET]                                       Get APB status and Data Fabric pstate(if
       APBDisabled)
  --getxgmiwidth [SOCKET]                                       Get xgmi link width
  --getdfpstaterange [SOCKET]                                   Get df pstate range for a given socket
  --getxgmipstaterange [SOCKET]                                 Get xgmi pstate range for a given socket
  --getccdpower [CORE]                                          Get CCD power for a given core
  --gettdelta [SOCKET]                                          Get thermal solution behaviour for a given
       socket
  --getspdregdata [SOCKET] [DIMM_ADDR] [LID] [OFFSET] [REGSPACE]
                                                                Get SPD SB register
       data(REGSPACE:0->Volatile,1->NVM)
  --getsvi3vrtemp [SOCKET] [TYPE] [RAIL_INDEX(if TYPE=1)]       Get svi3 vr controller
       temperature(TYPE:0->HottestRail,1->IndividualRail)
  --getpc6enable [SOCKET]                                       Get the PC6 Enable Control
  --getcc6enable [SOCKET]                                       Get the CC6 Enable Control
Set Option<s>:
  --setpowerlimit [SOCKET] [POWER]                              Set power limit for a given socket (mWatts)
  --setcorebl [CORE] [BOOSTLIMIT]                               Set BoostLimit for a given core (MHz)
  --setsockbl [SOCKET] [BOOSTLIMIT]                             Set BoostLimit for a given Socket (MHz)
  --apbdisable [SOCKET] [PSTATE<0-2>]                           Set Data Fabric Pstate for a given socket
  --apbenable [SOCKET]                                          Enable the Data Fabric performance boost
       algorithm for a given socket
  --setxgmiwidth [SOCKET] [MIN<0-2>] [MAX<0-2>]                 Set xgmi link width in a multi socket system
       (MAX >= MIN)
  --setlclkdpmlevel [SOCKET] [NBIOID<0-3>] [MIN<0-3>] [MAX<0-3>]Set lclk dpm level for a given nbio in a
       given socket (MAX >= MIN)
  --setdfpstaterange [SOCKET] [MIN<0-2>] [MAX<0-2>]             Set df pstate range for a given socket (MAX
       <= MIN)
  --setpowerefficiencymode [SOCKET] [MODE<0-5>]                 Set power efficiency mode for a given socket
  --setxgmipstaterange [SOCKET] [MIN<0,1>] [MAX<0,1>]           Set xgmi pstate range
  --setcpurailisofreqpolicy [SOCKET] [VAL<0,1>]                 Set CPU ISO frequency policy
  --setdfcctrl [SOCKET] [VAL<0,1>]                              Enable or disable DF c-state
  --setpc6enable [SOCKET] [val<0,1>]                            Set the PC6 Enable Control
  --setcc6enable [SOCKET] [val<0,1>]                            Set the CC6 Enable Control
============================ End of E-SMI ============================
```

Following are the value ranges and other information needed for passing it to tool

```
1.  ----showxgmibw [SOCKET] [LINKNAME] [BWTYPE]
      LINKNAME :
      Rolling Stones:P0/P1/P2/P3/G0/G1/G2/G3
      Mi300:G0/G1/G2/G3/G4/G5/G6/G7
      Family 0x1A, model 0x00-0x1F:P1/P3/G0/G1/G2/G3
      Family 0x1A, model 0x50-0x5F:P0/P1/P2/G0/G1/G2
      BWTYPE : AGG_BW/RD_BW/WR_BW
2.  --setxgmiwidth [MIN] [MAX]
      MIN : MAX :  0 - 2 with MIN <= MAX
3.  --showlclkdpmlevel [SOCKET] [NBIOID]
      NBIOID : 0 - 3
```

```
4.  --apbdisable [SOCKET] [PSTATE]
      PSTATE : 0 - 2
5.  --setlclkdpmlevel [SOCKET] [NBIOID] [MIN] [MAX]
      NBIOID : 0 - 3
      MI300A:  MIN : MAX : 0 - 2 with MIN <= MAX
      Other platforms: MIN : MAX : 0 - 3 with MIN <= MAX
6.  --setpcielinkratecontrol [SOCKET] [CTL]
      CTL : 0 - 2
7.  --setpowerefficiencymode [SOCKET] [MODE]
      Rolling Stones: MODE : 0 - 3
      Family 0x1A model 0x00-0x1F: MODE : 0 - 5
8.  --setdfpstaterange [SOCKET] [MAX] [MIN]
      MIN : MAX : 0 - 2 with MAX <= MIN
9.  --setgmi3linkwidth [SOCKET] [MIN] [MAX]
      MIN : MAX : 0 - 2 with MIN <= MAX
10. --testmailbox [SOCKET] [VALUE]
      VALUE : Any 32 bit value
```

Below is a sample usage to get different system metrics information

```
1.  e_smi_library/b$ sudo ./e_smi_tool --showcoreenergy 0
    ============================ E-SMI ==================================
    -------------------------------------------------
    | core[000] energy  |           646.549 Joules  |
    -------------------------------------------------
    ============================= End of E-SMI ============================
2.  e_smi_library/b$ sudo ./e_smi_tool --showcoreenergy 12 --showsockpower --setpowerlimit 1 220000
       --showsockpower
    ============================ E-SMI ==================================
    -------------------------------------------------
    | core[012] energy  |            73.467 Joules  |
    -------------------------------------------------
    ---------------------------------------------------------------------
    | Sensor Name                    | Socket 0         | Socket 1         |
    ---------------------------------------------------------------------
    | Power (Watts)                  | 174.051          | 169.451          |
    | PowerLimit (Watts)             | 400.000          | 220.000          |
    | PowerLimitMax (Watts)          | 400.000          | 320.000          |
    ---------------------------------------------------------------------
    Socket[1] power_limit set to 220.000 Watts successfully
    ---------------------------------------------------------------------
    | Sensor Name                    | Socket 0         | Socket 1         |
    ---------------------------------------------------------------------
    | Power (Watts)                  | 174.085          | 169.431          |
    | PowerLimit (Watts)             | 400.000          | 220.000          |
    | PowerLimitMax (Watts)          | 400.000          | 320.000          |
    ---------------------------------------------------------------------
3.  e_smi_library/b$$ ./e_smi_tool --showxgmibandwidth G2 AGG_BW
    ============================ E-SMI ==================================
    -------------------------------------------------------------
    | Current Aggregate bandwidth of xGMI link G2 |     40 Mbps |
    -------------------------------------------------------------
    ============================= End of E-SMI ============================
4.  e_smi_library/b$sudo ./e_smi_tool --setdfpstaterange 0 1 2
    [sudo] password for user:
    ============================ E-SMI ==================================
    Data Fabric PState range(max:1 min:2) set successfully
    ============================= End of E-SMI ============================
5.  e_smi_library/b$./e_smi_tool --showsockpower --showprochotstatus --loopdelay 1 s --loopcount 10 --log
       power.csv --printonconsole 0
    e_smi_library/b$cat power.csv
    2025-06-10,12:39:37:88,45.951,43.225,400.000,400.000,500.000,500.000,inactive,inactive,
    2025-06-10,12:39:38:98,39.572,39.175,400.000,400.000,500.000,500.000,inactive,inactive,
    2025-06-10,12:39:39:105,39.539,38.884,400.000,400.000,500.000,500.000,inactive,inactive,
    2025-06-10,12:39:40:117,41.892,42.220,400.000,400.000,500.000,500.000,inactive,inactive,
    2025-06-10,12:39:41:123,40.466,39.659,400.000,400.000,500.000,500.000,inactive,inactive,
    2025-06-10,12:39:42:134,39.681,39.218,400.000,400.000,500.000,500.000,inactive,inactive,
    2025-06-10,12:39:43:138,39.349,38.562,400.000,400.000,500.000,500.000,inactive,inactive,
    2025-06-10,12:39:44:145,39.517,38.807,400.000,400.000,500.000,500.000,inactive,inactive,
    2025-06-10,12:39:45:148,39.726,39.457,400.000,400.000,500.000,500.000,inactive,inactive,
    2025-06-10,12:39:46:160,39.393,38.699,400.000,400.000,500.000,500.000,inactive,inactive,
6.  e_smi_library/b$./e_smi_tool --showsockc0res 0 --showcorebl 0 --showsockc0res 1 --json
    {
            {
                "Socket":0,
                "C0Residency(%)":0
            },
            {
                "Core":0,
                "BoostLimit(MHz)":4100
            },
            {
                "Socket":1,
                "C0Residency(%)":0
            },
            {
                "JSONFormatVersion":1
            }
    }
7.  e_smi_library/b$./e_smi_tool --showsockc0res 0 --showcorebl 0 --showsockc0res 1 --csv
    Socket,C0Residency(%)
    0,0
    Core,BoostLimit(MHz)
    0,4100
    Socket,C0Residency(%)
    1,0
8.  //To display the data in the console for a specified number of user-defined loops and loop delay.
    e_smi_library/b$./e_smi_tool --showsockpower --initialdelay 2 s --loopdelay 1 s --loopcount 2
    ============================ E-SMI ==================================
    * InitialDelay(in secs):2.000000, ...
    * LoopCount:0, LoopDelay(in secs):1.000000, ...
    * CurrentTime:2025-06-13,11:40:18:367
    ----------------------------------------------------------------------
    | Sensor Name                     | Socket 0         | Socket 1         |
    ----------------------------------------------------------------------
    | Power (Watts)                   | 48.148           | 42.990           |
    | PowerLimit (Watts)              | 400.000          | 400.000          |
    | PowerLimitMax (Watts)           | 500.000          | 500.000          |
    ----------------------------------------------------------------------
    * LoopCount:1, LoopDelay(in secs):1.000000, ...
    * CurrentTime:2025-06-13,11:40:19:370
    ----------------------------------------------------------------------
    | Sensor Name                     | Socket 0         | Socket 1         |
    ----------------------------------------------------------------------
    | Power (Watts)                   | 103.711          | 87.128           |
    | PowerLimit (Watts)              | 400.000          | 400.000          |
    | PowerLimitMax (Watts)           | 500.000          | 500.000          |
    ----------------------------------------------------------------------
9.  //To output the data to the console and simultaneously record it in log(CSV format) for user-defined
       loops and loop delay.
    e_smi_library/b$./e_smi_tool --showsockpower --initialdelay 2 s --loopdelay 1 s --loopcount 2 --log
       power.csv
    ============================ E-SMI ==================================
    * InitialDelay(in secs):2.000000, ...
    * LoopCount:0, LoopDelay(in secs):1.000000, ...
    * CurrentTime:2025-06-13,11:40:18:367
    ----------------------------------------------------------------------
    | Sensor Name                     | Socket 0         | Socket 1         |
    ----------------------------------------------------------------------
    | Power (Watts)                   | 48.148           | 42.990           |
    | PowerLimit (Watts)              | 400.000          | 400.000          |
    | PowerLimitMax (Watts)           | 500.000          | 500.000          |
    ----------------------------------------------------------------------
    * LoopCount:1, LoopDelay(in secs):1.000000, ...
    * CurrentTime:2025-06-13,11:40:19:370
    ----------------------------------------------------------------------
    | Sensor Name                     | Socket 0         | Socket 1         |
    ----------------------------------------------------------------------
    | Power (Watts)                   | 103.711          | 87.128           |
    | PowerLimit (Watts)              | 400.000          | 400.000          |
    | PowerLimitMax (Watts)           | 500.000          | 500.000          |
    ----------------------------------------------------------------------
    e_smi_library/b$cat power.csv

       Date,Timestamp,Socket0:Power(Watts),Socket1:Power(Watts),Socket0:PowerLimit(Watts),Socket1:PowerLimit(Watts),Socket0:Po
    2025-06-13,11:43:22:587,41.007,39.949,400.000,400.000,500.000,500.000,
    2025-06-13,11:43:23:590,47.329,46.269,400.000,400.000,500.000,500.000,
10. //To continuously collect data in the log(CSV format) without interruption until the exit condition is
       met(such as detecting a stoploop file).
    [Terminal 1]:
      e_smi_library/b$./e_smi_tool --showsockpower --loopdelay 1 s --loopcount -1 --log power.csv
       --printonconsole 0 --stoploop stresslog.txt
    [Terminal 2]:
      //Consider the user initiates a stress test for a random duration, which generates a stresslog.txt
       file upon completion of stress test.
      //For experimental purposes, we manually generate the stresslog.txt file after a random duration.
      e_smi_library/b$touch stresslog.txt
    [Terminal 1]:
      //The execution of e_smi_tool should have concluded by now, allowing the user to examine power.csv
       file generated during the stress test.
      e_smi_library/b$cat power.csv

       Date,Timestamp,Socket0:Power(Watts),Socket1:Power(Watts),Socket0:PowerLimit(Watts),Socket1:PowerLimit(Watts),Socket0:Po
      2025-06-13,11:53:53:245,40.082,39.281,400.000,400.000,500.000,500.000,
      2025-06-13,11:53:54:247,40.521,39.533,400.000,400.000,500.000,500.000,
      2025-06-13,11:53:55:249,42.126,40.270,400.000,400.000,500.000,500.000,
      2025-06-13,11:53:56:253,42.416,40.861,400.000,400.000,500.000,500.000,
      2025-06-13,11:53:57:254,41.132,40.421,400.000,400.000,500.000,500.000,
      2025-06-13,11:53:58:258,40.363,39.443,400.000,400.000,500.000,500.000,
      2025-06-13,11:53:59:265,41.472,40.382,400.000,400.000,500.000,500.000,
11. //To continuously display the data in the console for an indefinite duration (press CTRL+C to stop).
    e_smi_library/b$./e_smi_tool --showsockpower --loopdelay 1 s --loopcount -1
    ============================ E-SMI ==================================
    * LoopCount:0, LoopDelay(in secs):1.000000, ...
* CurrentTime:2025-06-13,12:10:08:350
--------------------------------------------------------------------------
| Sensor Name                      | Socket 0          | Socket 1          |
--------------------------------------------------------------------------
| Power (Watts)                    | 43.373            | 40.041            |
| PowerLimit (Watts)               | 400.000           | 400.000           |
| PowerLimitMax (Watts)            | 500.000           | 500.000           |
--------------------------------------------------------------------------
* LoopCount:1, LoopDelay(in secs):1.000000, ...
* CurrentTime:2025-06-13,12:10:09:353
--------------------------------------------------------------------------
| Sensor Name                      | Socket 0          | Socket 1          |
--------------------------------------------------------------------------
| Power (Watts)                    | 40.273            | 39.875            |
| PowerLimit (Watts)               | 400.000           | 400.000           |
| PowerLimitMax (Watts)            | 500.000           | 500.000           |
--------------------------------------------------------------------------
* LoopCount:2, LoopDelay(in secs):1.000000, ...
^C
```

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Module Documentation

5.1 Initialization and Shutdown

Functions

- esmi_status_t esmi_init (void)

  Initialize the library, validates the dependencies.
- void esmi_exit (void)

  Clean up any allocation done during init.

5.1.1 Detailed Description

This function validates the dependencies that exist and initializes the library.

5.1.2 Function Documentation

5.1.2.1 esmi_init()

```
esmi_status_t esmi_init (
            void  )
```

Initialize the library, validates the dependencies.

Search the available dependency entries and initialize the library accordingly.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.2 Energy Monitor (RAPL MSR)

Functions

- esmi_status_t esmi_core_energy_get (uint32_t core_ind, uint64_t ∗penergy)

  Get the core energy for a given core.
- esmi_status_t esmi_socket_energy_get (uint32_t socket_idx, uint64_t ∗penergy)

  Get the socket energy for a given socket.
- esmi_status_t esmi_all_energies_get (uint64_t ∗penergy)

  Get energies of all cores in the system.
- esmi_status_t esmi_rapl_units_hsmp_mailbox_get (uint32_t sock_ind, uint8_t ∗tu, uint8_t ∗esu)

  Get the RAPL units through HSMP mailbox.
- esmi_status_t esmi_rapl_package_counter_hsmp_mailbox_get (uint32_t sock_ind, uint32_t ∗counter1, uint32_t ∗counter0)

  Get the socket energy counter values for a given socket through mailbox.
- esmi_status_t esmi_rapl_core_counter_hsmp_mailbox_get (uint32_t core_ind, uint32_t ∗counter1, uint32_t ∗counter0)

  Get the core energy counter values for a given socket through mailbox.
- esmi_status_t esmi_core_energy_hsmp_mailbox_get (uint32_t core_ind, uint64_t ∗penergy)

  Get the core energy for a given core through HSMP mailbox.
- esmi_status_t esmi_package_energy_hsmp_mailbox_get (uint32_t sock_ind, uint64_t ∗penergy)

  Get the socket energy for a given socket through mailbox.

5.2.1 Detailed Description

Below functions provide interfaces to get the core energy value for a given core and to get the socket energy value for a given socket.

5.2.2 Function Documentation

5.2.2.1 esmi_core_energy_get()

```
esmi_status_t esmi_core_energy_get (
            uint32_t core_ind,
            uint64_t * penergy )
```

Get the core energy for a given core.

Given a core index `core_ind`, and a `penergy` argument for 64bit energy counter of that particular cpu, this function will read the energy counter of the given core and update the `penergy` in micro Joules.

Note: The energy status registers are accessed at core level. In a system with SMT enabled in BIOS, the sibling threads would report duplicate values. Aggregating the energy counters of the sibling threads is incorrect.

Parameters

in	*core_ind*	is a core index
in,out	*penergy*	Input buffer to return the core energy.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.2.2.2 esmi_socket_energy_get()

```
esmi_status_t esmi_socket_energy_get (
            uint32_t socket_idx,
            uint64_t * penergy )
```

Get the socket energy for a given socket.

Given a socket index `socket_idx`, and a `penergy` argument for 64bit energy counter of a particular socket.

Updates the `penergy` with socket energy in micro Joules.

Parameters

`in`	*socket_idx*	a socket index
`in,out`	*penergy*	Input buffer to return the socket energy.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.2.2.3 esmi_all_energies_get()

```
esmi_status_t esmi_all_energies_get (
            uint64_t * penergy )
```

Get energies of all cores in the system.

Given an argument for energy profile `penergy`, This function will read all core energies in an array `penergy` in micro Joules.

Parameters

`in,out`	*penergy*	Input buffer to return the energies of all cores. penergy should be allocated by user as below (esmi_number_of_cpus_get()/esmi_threads_per_core_get()) $*$ sizeof (uint64_t)

Return values

ESMI_SUCCESS	is returned upon successful call.

Return values

None-zero	is returned upon failure.

5.2.2.4 esmi_rapl_units_hsmp_mailbox_get()

```
esmi_status_t esmi_rapl_units_hsmp_mailbox_get (
            uint32_t sock_ind,
            uint8_t * tu,
            uint8_t * esu )
```

Get the RAPL units through HSMP mailbox.

Parameters

`in`	*sock_ind*	a socket index
`in,out`	*tu*	Input buffer to return the time units.
`in,out`	*esu*	Input buffer to return the energy units. actual energy will be calculated by multiplying the energy counter value with (1/2)^ESU

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.2.2.5 esmi_rapl_package_counter_hsmp_mailbox_get()

```
esmi_status_t esmi_rapl_package_counter_hsmp_mailbox_get (
            uint32_t sock_ind,
            uint32_t * counter1,
            uint32_t * counter0 )
```

Get the socket energy counter values for a given socket through mailbox.

Updates the `counter0` and `counter1` with lower 32 bit and upper 32 bit of socket energy counter respectively. Please note these units need to be multiplied with energy units to get actual energy consumption.

Parameters

`in`	*sock_ind*	a socket index
`in,out`	*counter0*	Input buffer to return the lower 32 bit of socket energy counter.
`in,out`	*counter1*	Input buffer to return the upper 32 bit of socket energy counter.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.2.2.6 esmi_rapl_core_counter_hsmp_mailbox_get()

```
esmi_status_t esmi_rapl_core_counter_hsmp_mailbox_get (
            uint32_t core_ind,
            uint32_t * counter1,
            uint32_t * counter0 )
```

Get the core energy counter values for a given socket through mailbox.

Updates the `counter0` and `counter1` with lower 32 bit and upper 32 bit of core energy counter respectively. Please note these units need to be multiplied with energy units to get actual energy consumption.

Parameters

`in`	*core_ind*	a core index
`in,out`	*counter0*	Input buffer to return the lower 32 bit of core energy.
`in,out`	*counter1*	Input buffer to return the upper 32 bit of core energy.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.2.2.7 esmi_core_energy_hsmp_mailbox_get()

```
esmi_status_t esmi_core_energy_hsmp_mailbox_get (
            uint32_t core_ind,
            uint64_t * penergy )
```

Get the core energy for a given core through HSMP mailbox.

Given a core index `core_ind`, this function will calculate the energy of that particular cpu by multiplying counter values obtained from esmi_rapl_core_counter_hsmp_mailbox_get() with ESU values from esmi_rapl_units_hsmp_mailbox_get()(counter value ∗ 1/2^ESU) and updates the `penergy` in micro Joules.

Parameters

`in`	*core_ind*	is a core index
`in,out`	*penergy*	Input buffer to return the core energy.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.2.2.8 esmi_package_energy_hsmp_mailbox_get()

```
esmi_status_t esmi_package_energy_hsmp_mailbox_get (
            uint32_t sock_ind,
            uint64_t * penergy )
```

Get the socket energy for a given socket through mailbox.

Given a socket index `sock_ind`, this function will calculate the energy of that particular socket by multiplying counter values obtained from esmi_rapl_package_counter_hsmp_mailbox_get() with ESU values from esmi_rapl_units_hsmp_mailbox_get()(counter value ∗ 1/2^ESU) and returns it in `penergy` in micro joules.

Parameters

`in`	*sock_ind*	a socket index
`in,out`	*penergy*	Input buffer to return the socket energy.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.3 HSMP System Statistics

Functions

- esmi_status_t esmi_hsmp_driver_version_get (struct hsmp_driver_version ∗hsmp_driver_ver)

    Get the HSMP Driver version.
- esmi_status_t esmi_smu_fw_version_get (struct smu_fw_version ∗smu_fw)

    Get the SMU Firmware Version.
- esmi_status_t esmi_prochot_status_get (uint32_t socket_idx, uint32_t ∗prochot)

    Get normalized status of the processor's PROCHOT status. 1 - PROCHOT active, 0 - PROCHOT inactive.
- esmi_status_t esmi_fclk_mclk_get (uint32_t socket_idx, uint32_t ∗fclk, uint32_t ∗mclk)

    Get the Data Fabric clock and Memory clock in MHz, for a given socket index.
- esmi_status_t esmi_cclk_limit_get (uint32_t socket_idx, uint32_t ∗cclk)

    Get the core clock (MHz) allowed by the most restrictive infrastructure limit at the time of the message.
- esmi_status_t esmi_hsmp_proto_ver_get (uint32_t ∗proto_ver)

    Get the HSMP interface (protocol) version.
- esmi_status_t esmi_socket_current_active_freq_limit_get (uint32_t sock_ind, uint16_t ∗freq, char ∗∗src_type)

Get the current active frequency limit of the socket.
- esmi_status_t esmi_socket_freq_range_get (uint8_t sock_ind, uint16_t ∗fmax, uint16_t ∗fmin)

  Get the Socket frequency range.
- esmi_status_t esmi_current_freq_limit_core_get (uint32_t core_id, uint32_t ∗freq)

  Get the current active frequency limit of the core.
- esmi_status_t esmi_cpurail_isofreq_policy_get (uint8_t sock_ind, bool ∗val)

  Get the CpuRailIsoFreqPolicy.
- esmi_status_t esmi_dfc_ctrl_setting_get (uint8_t sock_ind, bool ∗val)

  get the DfcEnable.

5.3.1 Detailed Description

Below functions to get HSMP System Statistics.

5.3.2 Function Documentation

5.3.2.1 esmi_hsmp_driver_version_get()

```
esmi_status_t esmi_hsmp_driver_version_get (
            struct hsmp_driver_version ∗ hsmp_driver_ver )
```

Get the HSMP Driver version.

This function will return the HSMP Driver version at `hsmp_driver_ver` Supported on all hsmp protocol versions

Return values

ESMI_SUCCESS	is returned upon successful call.

5.3.2.2 esmi_smu_fw_version_get()

```
esmi_status_t esmi_smu_fw_version_get (
            struct smu_fw_version ∗ smu_fw )
```

Get the SMU Firmware Version.

This function will return the SMU FW version at `smu_fw` Supported on all hsmp protocol versions

Parameters

`in,out`	*smu_fw*	Input buffer to return the smu firmware version.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.3.2.3 esmi_prochot_status_get()

```
esmi_status_t esmi_prochot_status_get (
            uint32_t socket_idx,
            uint32_t * prochot )
```

Get normalized status of the processor's PROCHOT status. 1 - PROCHOT active, 0 - PROCHOT inactive.

Given a socket index `socket_idx` and this function will get PROCHOT at `prochot`. Supported on all hsmp protocol versions

Parameters

`in`	*socket_idx*	a socket index
`in,out`	*prochot*	Input buffer to return the PROCHOT status.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.3.2.4 esmi_fclk_mclk_get()

```
esmi_status_t esmi_fclk_mclk_get (
            uint32_t socket_idx,
            uint32_t * fclk,
            uint32_t * mclk )
```

Get the Data Fabric clock and Memory clock in MHz, for a given socket index.

Given a socket index `socket_idx` and a pointer to a uint32_t `fclk` and `mclk`, this function will get the data fabric clock and memory clock. Supported on all hsmp protocol versions

Parameters

`in`	*socket_idx*	a socket index
`in,out`	*fclk*	Input buffer to return the data fabric clock.
`in,out`	*mclk*	Input buffer to return the memory clock.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.3.2.5 esmi_cclk_limit_get()

```
esmi_status_t esmi_cclk_limit_get (
             uint32_t socket_idx,
             uint32_t * cclk )
```

Get the core clock (MHz) allowed by the most restrictive infrastructure limit at the time of the message.

Given a socket index `socket_idx` and a pointer to a uint32_t `cclk`, this function will get the core clock throttle limit. Supported on all hsmp protocol versions

Parameters

`in`	*socket_idx*	a socket index
`in,out`	*cclk*	Input buffer to return the core clock throttle limit.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.3.2.6 esmi_hsmp_proto_ver_get()

```
esmi_status_t esmi_hsmp_proto_ver_get (
             uint32_t * proto_ver )
```

Get the HSMP interface (protocol) version.

This function will get the HSMP interface version at `proto_ver` Supported on all hsmp protocol versions

Parameters

`in,out`	*proto_ver*	Input buffer to return the hsmp protocol version.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.3.2.7 esmi_socket_current_active_freq_limit_get()

```
esmi_status_t esmi_socket_current_active_freq_limit_get (
             uint32_t sock_ind,
             uint16_t * freq,
             char ** src_type )
```

Get the current active frequency limit of the socket.

This function will get the socket frequency and source of this limit Supported on all hsmp protocol versions

Parameters

`in`	*sock_ind*	A socket index.
`in,out`	*freq*	Input buffer to return the frequency(MHz).
`in,out`	*src_type*	Input buffer to return the source of this limit

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.3.2.8 esmi_socket_freq_range_get()

```
esmi_status_t esmi_socket_freq_range_get (
             uint8_t sock_ind,
             uint16_t * fmax,
             uint16_t * fmin )
```

Get the Socket frequency range.

This function returns the socket frequency range, fmax and fmin. Supported only on hsmp protocol version 5 and 7.

Parameters

`in`	*sock_ind*	Socket index.
`in,out`	*fmax*	Input buffer to return the maximum frequency(MHz).
`in,out`	*fmin*	Input buffer to return the minimum frequency(MHz).

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.3.2.9 esmi_current_freq_limit_core_get()

```
esmi_status_t esmi_current_freq_limit_core_get (
            uint32_t core_id,
            uint32_t * freq )
```

Get the current active frequency limit of the core.

This function returns the core frequency limit for the specified core. Supported only on hsmp protocol version 5 and 7.

Parameters

in	*core←↩ _id*	Core index.
in,out	*freq*	Input buffer to return the core frequency limit(MHz)

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.3.2.10 esmi_cpurail_isofreq_policy_get()

```
esmi_status_t esmi_cpurail_isofreq_policy_get (
            uint8_t sock_ind,
            bool * val )
```

Get the CpuRailIsoFreqPolicy.

This function gets the CpuRailIsoFreqPolicy.

Parameters

in	*sock_ind*	a socket index
in	*val*	Input buffer containing boolean value which indicates whether all cores on both rails have same frequency limit or different frequency limit. All cores on both rails have same freq limit - 1 Each rail has different independent frequency limit - 0

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.3.2.11 esmi_dfc_ctrl_setting_get()

```
esmi_status_t esmi_dfc_ctrl_setting_get (
            uint8_t sock_ind,
            bool * val )
```

get the DfcEnable.

This function gets DF C-state enabling control. DF C-state is a low power state for IOD.

Parameters

in	*sock_ind*	a socket index
in	*val*	Input buffer holds a boolean which indicates whether DFC is enabled or disabled. Enable DFC - 1 Disable DFC - 0

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.4 Power Monitor

Functions

- esmi_status_t esmi_socket_power_get (uint32_t socket_idx, uint32_t ∗ppower)

  Get the instantaneous power consumption of the provided socket.
- esmi_status_t esmi_socket_power_cap_get (uint32_t socket_idx, uint32_t ∗pcap)

  Get the current power cap value for a given socket.
- esmi_status_t esmi_socket_power_cap_max_get (uint32_t socket_idx, uint32_t ∗pmax)

  Get the maximum power cap value for a given socket.
- esmi_status_t esmi_pwr_svi_telemetry_all_rails_get (uint32_t sock_ind, uint32_t ∗power)

  Get the SVI based power telemetry for all rails.
- esmi_status_t esmi_pwr_efficiency_mode_get (uint8_t sock_ind, uint8_t ∗mode)

  Get the current power efficiency mode.
- esmi_status_t esmi_read_ccd_power (uint32_t coreid, uint32_t ∗power)

  Get CCD power.

5.4.1 Detailed Description

Below functions provide interfaces to get the current power usage and Power Limits for a given socket.

5.4.2 Function Documentation

5.4.2.1 esmi_socket_power_get()

```
esmi_status_t esmi_socket_power_get (
             uint32_t socket_idx,
             uint32_t * ppower )
```

Get the instantaneous power consumption of the provided socket.

Given a socket index `socket_idx` and a pointer to a uint32_t `ppower`, this function will get the current power consumption (in milliwatts) to the uint32_t pointed to by `ppower`. Supported on all hsmp protocol versions

Parameters

`in`	*socket_idx*	a socket index
`in,out`	*ppower*	Input buffer to return power consumption in the socket.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.4.2.2 esmi_socket_power_cap_get()

```
esmi_status_t esmi_socket_power_cap_get (
             uint32_t socket_idx,
             uint32_t * pcap )
```

Get the current power cap value for a given socket.

This function will return the valid power cap `pcap` for a given socket `socket_idx`, this value will be used by the system to limit the power usage. Supported on all hsmp protocol versions

Parameters

`in`	*socket_idx*	a socket index
`in,out`	*pcap*	Input buffer to return power limit on the socket, in milliwatts.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.4.2.3 esmi_socket_power_cap_max_get()

```
esmi_status_t esmi_socket_power_cap_max_get (
            uint32_t socket_idx,
            uint32_t * pmax )
```

Get the maximum power cap value for a given socket.

This function will return the maximum possible valid power cap `pmax` from a `socket_idx`. Supported on all hsmp protocol versions

Parameters

`in`	*socket_idx*	a socket index
`in,out`	*pmax*	Input buffer to return maximum power limit on socket, in milliwatts.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.4.2.4 esmi_pwr_svi_telemetry_all_rails_get()

```
esmi_status_t esmi_pwr_svi_telemetry_all_rails_get (
            uint32_t sock_ind,
            uint32_t * power )
```

Get the SVI based power telemetry for all rails.

This function returns the SVI based power telemetry for all rails. Supported only on hsmp protocol version 5 and 7.

Parameters

`in`	*sock_ind*	Socket index.
`in,out`	*power*	Input buffer to return the power(mW).

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.4.2.5 esmi_pwr_efficiency_mode_get()

```
esmi_status_t esmi_pwr_efficiency_mode_get (
            uint8_t sock_ind,
            uint8_t * mode )
```

Get the current power efficiency mode.

This function returns the current power efficiency mode.

Parameters

`in`	*sock_ind*	Socket index.
`in,out`	*mode*	Input buffer to return the mode. Refer esmi_pwr_efficiency_mode_set for details on the modes

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.4.2.6 esmi_read_ccd_power()

```
esmi_status_t esmi_read_ccd_power (
            uint32_t coreid,
            uint32_t * power )
```

Get CCD power.

This function returns the average CCD power

Parameters

`in`	*coreid*	core index.
`in,out`	*power*	Input buffer to return the power

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.5 Power Control

Functions

- esmi_status_t esmi_socket_power_cap_set (uint32_t socket_idx, uint32_t pcap)

  Set the power cap value for a given socket.
- esmi_status_t esmi_pwr_efficiency_mode_set (uint8_t sock_ind, uint8_t mode)

  Set the power efficiency profile policy.
- esmi_status_t esmi_cpurail_isofreq_policy_set (uint8_t sock_ind, bool ∗val)

  Set the CpuRailIsoFreqPolicy.
- esmi_status_t esmi_dfc_enable_set (uint8_t sock_ind, bool ∗val)

  Set the DfcEnable.

5.5.1 Detailed Description

This function provides a way to control Power Limit.

5.5.2 Function Documentation

5.5.2.1 esmi_socket_power_cap_set()

```
esmi_status_t esmi_socket_power_cap_set (
            uint32_t socket_idx,
            uint32_t pcap )
```

Set the power cap value for a given socket.

This function will set the power cap to the provided value `pcap`. This cannot be more than the value returned by esmi_socket_power_cap_max_get().

Note: The power limit specified will be clipped to the maximum cTDP range for the processor. There is a limit on the minimum power that the processor can operate at, no further socket power reduction occurs if the limit is set below that minimum and also there are independent registers through HSMP and APML whichever is the most constraining between the two is enforced. Supported on all hsmp protocol versions.

Parameters

in	*socket_idx*	a socket index
in	*pcap*	a uint32_t that indicates the desired power cap, in milliwatts

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.5.2.2 esmi_pwr_efficiency_mode_set()

```
esmi_status_t esmi_pwr_efficiency_mode_set (
            uint8_t sock_ind,
            uint8_t mode )
```

Set the power efficiency profile policy.

This function will set the power efficiency mode. Supported only on hsmp protocol version 5 and 7.

Power efficiency modes are:

0 = High performance mode: This mode favours core performance. In this mode all df pstates are available and default df pstate and DLWM algorithms are active.

1 = Power efficient mode: This mode limits the boost frequency available to the cores and restricts the DF P-States. This mode also monitors the system load to dynamically adjust performance for maximum power efficiency.

2 = IO performance mode: This mode sets up data fabric to maximize IO performance. This can result in lower core performance to increase the IO throughput.

3 = Balanced Memory Performance Mode: This mode biases the memory subsystem and Infinity Fabric™ performance towards efficiency, by lowering the frequency of the fabric and the width of the xGMI links under light traffic conditions. Core behavior is unaffected. There may be a performance impact under lightly loaded conditions for memory-bound applications compared to the default high performance mode. With higher memory and fabric load, the system becomes similar in performance to the default high performance mode.

4 = Balanced Core Performance Mode: This mode biases toward consistent core performance across varying core utilization levels, by preventing active cores from using the power budget of inactive cores. This mode allows core "boosting" as in the default high performance mode, but does not allow core boost to take advantage of the power budget of inactive cores, resulting in a more efficient operating point for the active cores. The memory subsystem and Infinity Fabric behavior is unaffected. There may be a performance impact under light core utilization conditions compared to the default high performance mode. With high core utilization levels, the performance is similar to the default high performance mode.

5 = Balanced Core and Memory Performance Mode. This mode combines the Balanced Memory Performance and the Balanced Core Performance mode and may result in lower performance under light loads compared to the default high performance mode, but with significant increase in efficiency under light loads. Performance in this mode will be similar to the default high performance mode as the system load increases.

Parameters

in	*sock_ind*	A socket index.
in	*mode*	Power efficiency mode to be set.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.5.2.3 esmi_cpurail_isofreq_policy_set()

```
esmi_status_t esmi_cpurail_isofreq_policy_set (
             uint8_t sock_ind,
             bool * val )
```

Set the CpuRailIsoFreqPolicy.

This function sets the CpuRailIsoFreqPolicy.

If a socket wide limit (e.g. PPT)is setting the core clock frequency, then this setting has no effect. For other limiters specific to CPU power rails (e.g. TDC), this policy allows or disables independent core clocks per rail(VDDCR_CPU0 or VDDCR_CPU1).

Parameters

in	*sock_ind*	a socket index
in	*val*	Input buffer to contian a boolean which indicates whether all cores on both rails have same frequency limit or different frequency limit. All cores on both rails have same freq limit - 1 Each rail has different independent frequency limit - 0

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.5.2.4 esmi_dfc_enable_set()

```
esmi_status_t esmi_dfc_enable_set (
            uint8_t sock_ind,
            bool * val )
```

Set the DfcEnable.

This function sets DF C-state enabling control. DF C-state is a low power state for IOD.

Parameters

in	*sock_ind*	a socket index
in	*val*	Input buffer holds a boolean which indicates whether to disable DFC or to enable DFC. Enable DFC - 1 Disable DFC - 0

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.6 Performance (Boost limit) Monitor

Functions

- esmi_status_t esmi_core_boostlimit_get (uint32_t cpu_ind, uint32_t ∗pboostlimit)

  Get the boostlimit value for a given core.
- esmi_status_t esmi_socket_c0_residency_get (uint32_t socket_idx, uint32_t ∗pc0_residency)

  Get the c0_residency value for a given socket.

5.6.1 Detailed Description

This function provides the current boostlimit value for a given core.

5.6.2 Function Documentation

5.6.2.1 esmi_core_boostlimit_get()

```
esmi_status_t esmi_core_boostlimit_get (
             uint32_t cpu_ind,
             uint32_t * pboostlimit )
```

Get the boostlimit value for a given core.

This function provides the frequency currently enforced through esmi_core_boostlimit_set() and esmi_socket_boostlimit_set() APIs for a particular `cpu_ind`. Supported on all hsmp protocol versions. Please note: there are independent registers through HSMP and APML. This message provides boost limit associated with HSMP only.

Parameters

`in`	*cpu_ind*	a cpu index
`in,out`	*pboostlimit*	Input buffer to return the boostlimit.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.6.2.2 esmi_socket_c0_residency_get()

```
esmi_status_t esmi_socket_c0_residency_get (
             uint32_t socket_idx,
             uint32_t * pc0_residency )
```

Get the c0_residency value for a given socket.

This function will return the socket's current c0_residency `pc0_residency` for a particular `socket_idx` Supported on all hsmp protocol versions

Parameters

`in`	*socket_idx*	a socket index provided.
`in,out`	*pc0_residency*	Input buffer to return the c0_residency.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.7 Performance (Boost limit) Control

Functions

- esmi_status_t esmi_core_boostlimit_set (uint32_t cpu_ind, uint32_t boostlimit)

  Set the boostlimit value for a given core.
- esmi_status_t esmi_socket_boostlimit_set (uint32_t socket_idx, uint32_t boostlimit)

  Set the boostlimit value for a given socket.

5.7.1 Detailed Description

Below functions provide ways to control Boost limit values.

5.7.2 Function Documentation

5.7.2.1 esmi_core_boostlimit_set()

```
esmi_status_t esmi_core_boostlimit_set (
            uint32_t cpu_ind,
            uint32_t boostlimit )
```

Set the boostlimit value for a given core.

This function will set the boostlimit to the provided value `boostlimit` for a given cpu `cpu_ind`.

Note: Even though set boost limit provides ability to limit frequency on a core basis, if all the cores of a CCX are not programmed for the same boost limit frequency, then the lower-frequency cores are limited to a frequency resolution that can be as low as 20% of the requested frequency. If the specified boost limit frequency of a core is not supported, then the processor selects the next lower supported frequency. For processor with SMT enabled, writes to different APIC ids that map to the same physical core overwrite the previous write to that core. There are independent registers through HSMP and APML whichever is the most constraining between the two is enforced. Supported on all hsmp protocol versions

Parameters

in	*cpu_ind*	a cpu index is a given core to set the boostlimit
in	*boostlimit*	a uint32_t that indicates the desired boostlimit value of a given core. The maximum accepted value is 65535MHz(UINT16_MAX).

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.7.2.2 esmi_socket_boostlimit_set()

```
esmi_status_t esmi_socket_boostlimit_set (
            uint32_t socket_idx,
            uint32_t boostlimit )
```

Set the boostlimit value for a given socket.

This function will set the boostlimit to the provided value `boostlimit` for a given socket `socket_idx`. There are independent registers through HSMP and APML whichever is the most constraining between the two is enforced. Supported on all hsmp protocol versions

Parameters

`in`	*socket_idx*	a socket index to set boostlimit.
`in`	*boostlimit*	a uint32_t that indicates the desired boostlimit value of a particular socket. The maximum accepted value is 65535MHz(UINT16_MAX).

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.8 ddr_bandwidth Monitor

Functions

- esmi_status_t esmi_ddr_bw_get (uint8_t sock_ind, struct ddr_bw_metrics ∗ddr_bw)

  Get the Theoretical maximum DDR Bandwidth in GB/s, Current utilized DDR Bandwidth in GB/s and Current utilized DDR Bandwidth as a percentage of theoretical maximum in a system. Supported only on hsmp protocol version >= 3.

5.8.1 Detailed Description

This function provides the DDR Bandwidth for a system

5.8.2 Function Documentation

5.8.2.1 esmi_ddr_bw_get()

```
esmi_status_t esmi_ddr_bw_get (
            uint8_t sock_ind,
            struct ddr_bw_metrics * ddr_bw )
```

Get the Theoretical maximum DDR Bandwidth in GB/s, Current utilized DDR Bandwidth in GB/s and Current utilized DDR Bandwidth as a percentage of theoretical maximum in a system. Supported only on hsmp protocol version >= 3.

This function will return the DDR Bandwidth metrics `ddr_bw` for DDR connected to a socket.

Parameters

in	*sock_ind*	a socket index provided.
in,out	*ddr_bw*	Input buffer to return the DDR bandwidth metrics, contains max_bw, utilized_bw and utilized_pct.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.9 Temperature Query

Functions

- esmi_status_t esmi_socket_temperature_get (uint32_t sock_ind, uint32_t ∗ptmon)

  Get temperature monitor for a given socket.
- esmi_status_t esmi_read_tdelta (uint8_t sock_ind, uint8_t ∗status)

  Get thermal solution behaviour for a given socket.
- esmi_status_t esmi_get_svi3_vr_controller_temp (uint8_t sock_ind, struct svi3_info ∗inout)

  Get temperature of svi3 VR controller rail for a given socket.

5.9.1 Detailed Description

This function provides the current tempearature value in degree C.

5.9.2 Function Documentation

5.9.2.1 esmi_socket_temperature_get()

```
esmi_status_t esmi_socket_temperature_get (
            uint32_t sock_ind,
            uint32_t * ptmon )
```

Get temperature monitor for a given socket.

This function will return the socket's current temperature in milli degree celsius `ptmon` for a particular `sock_ind`. Supported only on hsmp protocol version-4

Parameters

in	*sock_ind*	a socket index provided.
in,out	*ptmon*	pointer to a uint32_t that indicates the possible tmon value.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.9.2.2 esmi_read_tdelta()

```
esmi_status_t esmi_read_tdelta (
            uint8_t sock_ind,
            uint8_t * status )
```

Get thermal solution behaviour for a given socket.

This is a mechanism for thermal solution health. Supported only on hsmp protocol version-7

Parameters

`in`	*sock_ind*	a socket index provided.
`in,out`	*status*	indicates whether thermal solution is normal(0) or out of range(1).

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.9.2.3 esmi_get_svi3_vr_controller_temp()

```
esmi_status_t esmi_get_svi3_vr_controller_temp (
            uint8_t sock_ind,
            struct svi3_info * inout )
```

Get temperature of svi3 VR controller rail for a given socket.

Temperature of the hottest rail or the temperature of given rail is provided. Supported only on hsmp protocol version-7

Parameters

`in`	*sock_ind*	a socket index provided.
`in,out`	*inout*	has input data and contains output data.

Return values

ESMI_SUCCESS	is returned upon successful call.

Return values

None-zero	is returned upon failure.

5.10 Dimm statistics

Functions

- esmi_status_t esmi_dimm_temp_range_and_refresh_rate_get (uint8_t sock_ind, uint8_t dimm_addr, struct temp_range_refresh_rate ∗rate)

  Get dimm temperature range and refresh rate.
- esmi_status_t esmi_dimm_power_consumption_get (uint8_t sock_ind, uint8_t dimm_addr, struct dimm_power ∗dimm_pow)

  Get dimm power consumption and update rate.
- esmi_status_t esmi_dimm_thermal_sensor_get (uint8_t sock_ind, uint8_t dimm_addr, struct dimm_thermal ∗dimm_temp)

  Get dimm thermal sensor.
- esmi_status_t esmi_spd_sb_reg_read (uint8_t sock_ind, struct spd_info ∗inout)

  Execute a four byte read transaction at a given register offset in a specified device on the target DIMM. Supported only on hsmp protocol version 7.

5.10.1 Detailed Description

This function provides the dimm temperature, power and update rates.

5.10.2 Function Documentation

5.10.2.1 esmi_dimm_temp_range_and_refresh_rate_get()

```
esmi_status_t esmi_dimm_temp_range_and_refresh_rate_get (
            uint8_t sock_ind,
            uint8_t dimm_addr,
            struct temp_range_refresh_rate * rate )
```

Get dimm temperature range and refresh rate.

This function returns the per DIMM temperature range and refresh rate from the MR4 register. Supported only on hsmp protocol version 5 and 7.

Parameters

in	*sock_ind*	Socket index through which the DIMM can be accessed
in	*dimm_addr*	DIMM identifier, follow "HSMP DIMM Addres encoding".
in,out	*rate*	Input buffer of type struct temp_range_refresh_rate with refresh rate and temp range.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.10.2.2 esmi_dimm_power_consumption_get()

```
esmi_status_t esmi_dimm_power_consumption_get (
            uint8_t sock_ind,
            uint8_t dimm_addr,
            struct dimm_power * dimm_pow )
```

Get dimm power consumption and update rate.

This function returns the DIMM power and update rate Supported only on hsmp protocol version 5 and 7.

Parameters

`in`	*sock_ind*	Socket index through which the DIMM can be accessed.
`in`	*dimm_addr*	DIMM identifier, follow "HSMP DIMM Addres encoding".
`in,out`	*dimm_pow*	Input buffer of type struct dimm_power containing power(mW), update rate(ms) and dimm address.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.10.2.3 esmi_dimm_thermal_sensor_get()

```
esmi_status_t esmi_dimm_thermal_sensor_get (
            uint8_t sock_ind,
            uint8_t dimm_addr,
            struct dimm_thermal * dimm_temp )
```

Get dimm thermal sensor.

This function will return the DIMM thermal sensor(2 sensors per DIMM) and update rate Supported only on hsmp protocol version 5 and 7.

Parameters

`in`	*sock_ind*	Socket index through which the DIMM can be accessed.
`in`	*dimm_addr*	DIMM identifier, follow "HSMP DIMM Addres encoding".
`in,out`	*dimm_temp*	Input buffer of type struct dimm_thermal which contains temperature(℃), update rate(ms) and dimm address Update rate value can vary from 0 to 511ms. Update rate of "0" means last update was $<$ 1ms and 511ms means update was $>=$ 511ms.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.10.2.4 esmi_spd_sb_reg_read()

```
esmi_status_t esmi_spd_sb_reg_read (
            uint8_t sock_ind,
            struct spd_info * inout )
```

Execute a four byte read transaction at a given register offset in a specified device on the target DIMM. Supported only on hsmp protocol version 7.

Parameters

in	*sock_ind*	Socket index through which the DIMM can be accessed.
in,out	*contains*	dimm_addr Adddress of the DIMM. register offset in given register space. type of register offset space, volatile/non volatile. output register data.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.11 xGMI bandwidth control

Functions

- esmi_status_t esmi_xgmi_width_set (uint8_t sock_ind, uint8_t min, uint8_t max)

  Set xgmi width for a multi socket system. values range from 0 to 2.
- esmi_status_t esmi_xgmi_width_get (uint32_t sock_ind, uint8_t ∗min, uint8_t ∗max)

  Get xgmi width for a multi socket system. values range from 0 to 2.

5.11.1 Detailed Description

This function provides a way to control width of the xgmi links connected in multisocket systems.

5.11.2 Function Documentation

5.11.2.1 esmi_xgmi_width_set()

```
esmi_status_t esmi_xgmi_width_set (
             uint8_t sock_ind,
             uint8_t min,
             uint8_t max )
```

Set xgmi width for a multi socket system. values range from 0 to 2.

0 => 4 lanes on family 19h model 10h and 2 lanes on other models.

1 => 8 lanes.

2 => 16 lanes.

Supported on all hsmp protocol versions.

This function will set the xgmi width `min` and `max` for all the sockets in the system

Parameters

in	*sock_ind*	Socket index.
in	*min*	minimum xgmi link width, varies from 0 to 2 with min <= max.
in	*max*	maximum xgmi link width, varies from 0 to 2.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.11.2.2 esmi_xgmi_width_get()

```
esmi_status_t esmi_xgmi_width_get (
             uint32_t sock_ind,
             uint8_t * min,
             uint8_t * max )
```

Get xgmi width for a multi socket system. values range from 0 to 2.

0 => 4 lanes on family 19h model 10h and 2 lanes on other models.

1 => 8 lanes.

2 => 16 lanes.

Supported on all protocol version >= 7

This function will get the xgmi width `min` and `max` for all the sockets in the system

Parameters

in	*sock_ind*	Socket index.
in,out	*min*	minimum xgmi link width, varies from 0 to 2.
in,out	*max*	maximum xgmi link width, varies from 0 to 2.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.12 GMI3 width control

Functions

- esmi_status_t esmi_gmi3_link_width_range_set (uint8_t sock_ind, uint8_t min_link_width, uint8_t max_link_width)

  Set gmi3 width.

5.12.1 Detailed Description

This function provides a way to control global memory interconnect link width.

5.12.2 Function Documentation

5.12.2.1 esmi_gmi3_link_width_range_set()

```
esmi_status_t esmi_gmi3_link_width_range_set (
            uint8_t sock_ind,
            uint8_t min_link_width,
            uint8_t max_link_width )
```

Set gmi3 width.

This function will set the global memory interconnect width. Values can be 0, 1 or 2.

0 => Quarter width

1 => Half width

2 => Full width

Supported only on hsmp protocol version 5 and 7.

Parameters

in	*sock_ind*	Socket index.
in	*min_link_width*	Minimum link width to be set.
in	*max_link_width*	Maximum link width to be set.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.13 APB and LCLK level control

Functions

- esmi_status_t esmi_apb_enable (uint32_t sock_ind)
  Enable automatic P-state selection.
- esmi_status_t esmi_apb_disable (uint32_t sock_ind, uint8_t pstate)
  Set data fabric P-state to user specified value.
- esmi_status_t esmi_apb_status_get (uint32_t sock_ind, uint8_t ∗apb_disabled, uint8_t ∗pstate)
  Get APBDisabled status and gets data fabric P-state if APBDisabled.
- esmi_status_t esmi_socket_lclk_dpm_level_set (uint32_t sock_ind, uint8_t nbio_id, uint8_t min, uint8_t max)
  Set lclk dpm level.
- esmi_status_t esmi_socket_lclk_dpm_level_get (uint8_t sock_ind, uint8_t nbio_id, struct dpm_level ∗nbio)
  Get lclk dpm level.
- esmi_status_t esmi_pcie_link_rate_set (uint8_t sock_ind, uint8_t rate_ctrl, uint8_t ∗prev_mode)
  Set pcie link rate.
- esmi_status_t esmi_df_pstate_range_set (uint8_t sock_ind, uint8_t min_pstate, uint8_t max_pstate)
  Set data fabric pstate range.
- esmi_status_t esmi_df_pstate_range_get (uint8_t sock_ind, uint8_t ∗min_pstate, uint8_t ∗max_pstate)
  Get data fabric pstate range.
- esmi_status_t esmi_xgmi_pstate_range_set (uint8_t sock_ind, uint8_t min_state, uint8_t max_state)
  Set xgmi pstate range.
- esmi_status_t esmi_xgmi_pstate_range_get (uint8_t sock_ind, uint8_t ∗min_state, uint8_t ∗max_state)
  Get xgmi pstate range.
- esmi_status_t esmi_pc6_enable_set (uint8_t sock_ind, uint8_t pc6_enable)
  Set PC6 enable.
- esmi_status_t esmi_pc6_enable_get (uint8_t sock_ind, uint8_t ∗current_pc6_enable)
  Get PC6 enable.
- esmi_status_t esmi_cc6_enable_set (uint8_t sock_ind, uint8_t cc6_enable)
  Set CC6 enable.
- esmi_status_t esmi_cc6_enable_get (uint8_t sock_ind, uint8_t ∗current_cc6_enable)
  Get CC6 enable.

5.13.1 Detailed Description

This functions provides a way to control APB and lclk values.

5.13.2 Function Documentation

5.13.2.1 esmi_apb_enable()

```
esmi_status_t esmi_apb_enable (
            uint32_t sock_ind )
```

Enable automatic P-state selection.

Given a socket index `sock_ind`, this function will enable performance boost algorithm By default, an algorithm adjusts DF P-States automatically in order to optimize performance. However, this default may be changed to a fixed DF P-State through a CBS option at boottime. APBDisable may also be used to disable this algorithm and force a fixed DF P-State. Supported on all hsmp protocol versions

NOTE: While the socket is in PC6 or if PROCHOT_L is asserted, the lowest DF P-State (highest value) is enforced regardless of the APBEnable/APBDisable state.

Parameters

in	*sock_ind*	a socket index

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.13.2.2 esmi_apb_disable()

```
esmi_status_t esmi_apb_disable (
            uint32_t sock_ind,
            uint8_t pstate )
```

Set data fabric P-state to user specified value.

This function will set the desired P-state at `pstate`. Acceptable values for the P-state are 0(highest) - 2 (lowest) If the PC6 or PROCHOT_L is asserted, then the lowest DF pstate is enforced regardless of the APBenable/APBdiable states. Supported on all hsmp protocol versions.

Parameters

in	*sock_ind*	a socket index
in	*pstate*	a uint8_t that indicates the desired P-state to set.

Return values

ESMI_SUCCESS	is returned upon successful call.

Return values

None-zero	is returned upon failure.

5.13.2.3 esmi_apb_status_get()

```
esmi_status_t esmi_apb_status_get (
            uint32_t sock_ind,
            uint8_t * apb_disabled,
            uint8_t * pstate )
```

Get APBDisabled status and gets data fabric P-state if APBDisabled.

This function will get the current P-state at `pstate` if APBDisabled. Supported on all protocol version >= 7

Parameters

`in`	*sock_ind*	a socket index
`in,out`	*apb_disabled*	Input buffer for apb disabled status.
`in,out`	*pstate*	Input buffer for df pstate, during apbdisabled condition.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.13.2.4 esmi_socket_lclk_dpm_level_set()

```
esmi_status_t esmi_socket_lclk_dpm_level_set (
            uint32_t sock_ind,
            uint8_t nbio_id,
            uint8_t min,
            uint8_t max )
```

Set lclk dpm level.

This function will set the lclk dpm level / nbio pstate for the specified `nbio_id` in a specified socket `sock_ind` with provided values `min` and `max`. Supported on hsmp protocol version >= 2

Parameters

`in`	*sock_ind*	socket index.
`in`	*nbio_id*	northbridge number varies from 0 to 3.
`in`	*min*	pstate minimum value, varies from 0(lowest) to 3(highest) with min <= max
`in`	*max*	pstate maximum value, varies from 0 to 3.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.13.2.5 esmi_socket_lclk_dpm_level_get()

```
esmi_status_t esmi_socket_lclk_dpm_level_get (
              uint8_t sock_ind,
              uint8_t nbio_id,
              struct dpm_level * nbio )
```

Get lclk dpm level.

This function will get the lclk dpm level. DPM level is an encoding to represent PCIe link frequency. DPM levels can be set from APML also. This API gives current levels which may have been set from either APML or HSMP.

Supported in hsmp protocol version 5 and 7.

Parameters

in	*sock_ind*	Socket index
in	*nbio_id*	NBIO id(0-3)
in,out	*nbio*	Input buffer of struct dpm_level type to hold min and max dpm levels

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.13.2.6 esmi_pcie_link_rate_set()

```
esmi_status_t esmi_pcie_link_rate_set (
              uint8_t sock_ind,
              uint8_t rate_ctrl,
              uint8_t * prev_mode )
```

Set pcie link rate.

This function will set the pcie link rate to gen4/5 or auto detection based on bandwidth utilisation. Values are: 0 => auto detect bandwidth utilisation and set link rate

1 => Limit at gen4 rate

2 => Limit at gen5 rate

Supported only on hsmp protocol version 5 and 7.

Parameters

`in`	*sock_ind*	Socket index.
`in`	*rate_ctrl*	Control value to be set.
`in,out`	*prev_mode*	Input buffer to hold the previous mode.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.13.2.7 esmi_df_pstate_range_set()

```
esmi_status_t esmi_df_pstate_range_set (
            uint8_t sock_ind,
            uint8_t min_pstate,
            uint8_t max_pstate )
```

Set data fabric pstate range.

This function will set the max and min pstates for the data fabric. Acceptable values for the P-state are 0(highest) - 2 (lowest) with max <= min. DF pstate range can be set from both HSMP and APML, the most recent of the two is enforced. Supported only on hsmp protocol version 5 and 7.

Parameters

`in`	*sock_ind*	a socket index.
`in`	*min_pstate*	Minimum pstate value to be set.
`in`	*max_pstate*	Maximum pstate value to be set.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.13.2.8 esmi_df_pstate_range_get()

```
esmi_status_t esmi_df_pstate_range_get (
            uint8_t sock_ind,
            uint8_t * min_pstate,
            uint8_t * max_pstate )
```

Get data fabric pstate range.

This function will get the max and min pstates for the data fabric. This function is supported only on hsmp protocol version >= 7.

Parameters

in	*sock_ind*	a socket index.
in,out	*min_pstate*	Minimum pstate value to be get.
in,out	*max_pstate*	Maximum pstate value to be get.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.13.2.9 esmi_xgmi_pstate_range_set()

```
esmi_status_t esmi_xgmi_pstate_range_set (
             uint8_t sock_ind,
             uint8_t min_state,
             uint8_t max_state )
```

Set xgmi pstate range.

This function will set the max and min xgmi pstate. Acceptable values for the P-state are 0(high performance) and 1(low performance) with max_state <= min_state. XGMI pstate range can be set from both HSMP and APML, the most recent of the two is enforced.

Parameters

in	*sock_ind*	a socket index.
in	*min_pstate*	Minimum pstate value to be set.
in	*max_pstate*	Maximum pstate value to be set.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.13.2.10 esmi_xgmi_pstate_range_get()

```
esmi_status_t esmi_xgmi_pstate_range_get (
             uint8_t sock_ind,
             uint8_t * min_state,
             uint8_t * max_state )
```

Get xgmi pstate range.

This function will get the max and min xgmi pstate. Supported only on hsmp protocol version >= 7.

Parameters

`in`	*sock_ind*	a socket index.
`in,out`	*min_pstate*	Minimum pstate value to be get.
`in,out`	*max_pstate*	Maximum pstate value to be get.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.13.2.11 esmi_pc6_enable_set()

```
esmi_status_t esmi_pc6_enable_set (
             uint8_t sock_ind,
             uint8_t pc6_enable )
```

Set PC6 enable.

This function will set the pc6 enable control. PC6 is low power state for CCDs, also known as package C6 state. Acceptable values are 0(disable) and 1(enable) Supported on hsmp protocol version >= 7.

Parameters

`in`	*sock_ind*	a socket index.
`in`	*pc6_enable*	0(disable pc6) and 1(enable pc6).

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.13.2.12 esmi_pc6_enable_get()

```
esmi_status_t esmi_pc6_enable_get (
             uint8_t sock_ind,
             uint8_t * current_pc6_enable )
```

Get PC6 enable.

This function will get the pc6 enable control. Supported only on hsmp protocol version >= 7.

Parameters

`in`	*sock_ind*	a socket index.
`in,out`	*current_pc6_enable*	will get the pc6 current enable status.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.13.2.13 esmi_cc6_enable_set()

```
esmi_status_t esmi_cc6_enable_set (
             uint8_t sock_ind,
             uint8_t cc6_enable )
```

Set CC6 enable.

This function will set the cc6 enable control. Acceptable values are 0(disable) and 1(enable) Supported only on hsmp protocol version >= 7.

Parameters

in	*sock_ind*	a socket index.
in	*cc6_enable*	0(disable cc6) and 1(enable cc6).

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.13.2.14 esmi_cc6_enable_get()

```
esmi_status_t esmi_cc6_enable_get (
             uint8_t sock_ind,
             uint8_t * current_cc6_enable )
```

Get CC6 enable.

This function will get the cc6 enable control. Supported only on hsmp protocol version >= 7.

Parameters

in	*sock_ind*	a socket index.
in,out	*current_cc6_enable*	will get the cc6 current enable status.

Return values

ESMI_SUCCESS	is returned upon successful call.

Return values

None-zero	is returned upon failure.

5.14 Bandwidth Monitor

Functions

- esmi_status_t esmi_current_io_bandwidth_get (uint8_t sock_ind, struct link_id_bw_type link, uint32_t ∗io_↩ bw)

  Get IO bandwidth on IO link.
- esmi_status_t esmi_current_xgmi_bw_get (uint8_t sock_ind, struct link_id_bw_type link, uint32_t ∗xgmi_bw)

  Get xGMI bandwidth.

5.14.1 Detailed Description

This function provides the IO and xGMI bandiwtdh.

5.14.2 Function Documentation

5.14.2.1 esmi_current_io_bandwidth_get()

```
esmi_status_t esmi_current_io_bandwidth_get (
            uint8_t sock_ind,
            struct link_id_bw_type link,
            uint32_t * io_bw )
```

Get IO bandwidth on IO link.

This function returns the IO Aggregate bandwidth for the given link id. Supported only on hsmp protocol version 5 and 7.

Parameters

in	*sock_ind*	Socket index.
in	*link*	structure containing link_id(Link encoding values of given link) and bwtype info.
in,out	*io_bw*	Input buffer for bandwidth data in Mbps.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.14.2.2 esmi_current_xgmi_bw_get()

```
esmi_status_t esmi_current_xgmi_bw_get (
            uint8_t sock_ind,
            struct link_id_bw_type link,
            uint32_t * xgmi_bw )
```

Get xGMI bandwidth.

This function will read xGMI bandwidth in Mbps for the specified link and bandwidth type in a multi socket system. Supported only on hsmp protocol version 5 and 7.

Parameters

in	*sock_ind*	Socket index.
in	*link*	structure containing link_id(Link encoding values of given link) and bwtype info.
in,out	*xgmi_bw*	Input buffer for bandwidth data in Mbps.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.15 Metrics Table

Functions

- esmi_status_t esmi_metrics_table_version_get (uint32_t ∗metrics_version)

    Get metrics table version.
- esmi_status_t esmi_metrics_table_get (uint8_t sock_ind, struct hsmp_metric_table ∗metrics_table)

    Get metrics table.
- esmi_status_t esmi_dram_address_metrics_table_get (uint8_t sock_ind, uint64_t ∗dram_addr)

    Get the DRAM address for the metrics table.

5.15.1 Detailed Description

The following functions are assigned for CPU relative functionality, which is expected to be compatible with Epyc products.

5.15.2 Function Documentation

5.15.2.1 esmi_metrics_table_version_get()

```
esmi_status_t esmi_metrics_table_version_get (
            uint32_t * metrics_version )
```

Get metrics table version.

Get the version number[31:0] of metrics table

Parameters

in,out	*metrics_version*	input buffer to return the metrics table version.

Return values

ESMI_SUCCESS	is returned upon successful call.
Non-zero	is returned upon failure.

5.15.2.2 esmi_metrics_table_get()

```
esmi_status_t esmi_metrics_table_get (
            uint8_t sock_ind,
            struct hsmp_metric_table * metrics_table )
```

Get metrics table.

Read the metrics table

Parameters

in	*sock_ind*	Socket index.
in,out	*metrics_table*	input buffer to return the metrics table.

Return values

ESMI_SUCCESS	is returned upon successful call.
Non-zero	is returned upon failure.

5.15.2.3 esmi_dram_address_metrics_table_get()

```
esmi_status_t esmi_dram_address_metrics_table_get (
            uint8_t sock_ind,
            uint64_t * dram_addr )
```

Get the DRAM address for the metrics table.

Get DRAM address for Metric table transfer

Parameters

`in`	*sock_ind*	Socket index.
`in,out`	*dram_addr*	64-bit DRAM address

Return values

ESMI_SUCCESS	is returned upon successful call.
Non-zero	is returned upon failure.

5.16 Test HSMP mailbox

Functions

- esmi_status_t esmi_test_hsmp_mailbox (uint8_t sock_ind, uint32_t ∗data)

  Test HSMP mailbox interface.

5.16.1 Detailed Description

This is used to check if the HSMP interface is functioning correctly. Increments the input argument value by 1.

5.16.2 Function Documentation

5.16.2.1 esmi_test_hsmp_mailbox()

```
esmi_status_t esmi_test_hsmp_mailbox (
            uint8_t sock_ind,
            uint32_t * data )
```

Test HSMP mailbox interface.

[31:0] = input value

Parameters

`in`	*sock_ind*	Socket index.
`in,out`	*data*	input buffer to send input value and to get the output value

Return values

ESMI_SUCCESS	is returned upon successful call.
Non-zero	is returned upon failure.

5.17 Auxiliary functions

Functions

- esmi_status_t esmi_cpu_family_get (uint32_t ∗family)

  Get the CPU family.
- esmi_status_t esmi_cpu_model_get (uint32_t ∗model)

  Get the CPU model.
- esmi_status_t esmi_threads_per_core_get (uint32_t ∗threads)

  Get the number of threads per core in the system.
- esmi_status_t esmi_number_of_cpus_get (uint32_t ∗cpus)

  Get the number of cpus available in the system.
- esmi_status_t esmi_number_of_sockets_get (uint32_t ∗sockets)

  Get the total number of sockets available in the system.
- esmi_status_t esmi_first_online_core_on_socket (uint32_t socket_idx, uint32_t ∗pcore_ind)

  Get the first online core on a given socket.
- char ∗ esmi_get_err_msg (esmi_status_t esmi_err)

  Get the error string message for esmi errors.

5.17.1 Detailed Description

Below functions provide interfaces to get the total number of cores and sockets available and also to get the first online core on a given socket in the system.

5.17.2 Function Documentation

5.17.2.1 esmi_cpu_family_get()

```
esmi_status_t esmi_cpu_family_get (
            uint32_t * family )
```

Get the CPU family.

Parameters

in,out	*family*	Input buffer to return the cpu family.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.17.2.2 esmi_cpu_model_get()

```
esmi_status_t esmi_cpu_model_get (
            uint32_t * model )
```

Get the CPU model.

Parameters

in,out	*model*	Input buffer to reurn the cpu model.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.17.2.3 esmi_threads_per_core_get()

```
esmi_status_t esmi_threads_per_core_get (
            uint32_t * threads )
```

Get the number of threads per core in the system.

Parameters

in,out	*threads*	input buffer to return number of SMT threads.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.17.2.4 esmi_number_of_cpus_get()

```
esmi_status_t esmi_number_of_cpus_get (
            uint32_t * cpus )
```

Get the number of cpus available in the system.

Parameters

in,out	*cpus*	input buffer to return number of cpus, reported by nproc (including threads in case of SMT enable).

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.17.2.5 esmi_number_of_sockets_get()

```
esmi_status_t esmi_number_of_sockets_get (
            uint32_t * sockets )
```

Get the total number of sockets available in the system.

Parameters

in,out	*sockets*	input buffer to return number of sockets.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.17.2.6 esmi_first_online_core_on_socket()

```
esmi_status_t esmi_first_online_core_on_socket (
            uint32_t socket_idx,
            uint32_t * pcore_ind )
```

Get the first online core on a given socket.

Parameters

in	*socket_idx*	a socket index provided.
in,out	*pcore_ind*	input buffer to return the index of first online core in the socket.

Return values

ESMI_SUCCESS	is returned upon successful call.
None-zero	is returned upon failure.

5.17.2.7 esmi_get_err_msg()

```
char* esmi_get_err_msg (
            esmi_status_t esmi_err )
```

Get the error string message for esmi errors.

Get the error message for the esmi error numbers

Parameters

in	*esmi_err*	is a esmi error number

Return values

*char**	value returned upon successful call.

Chapter 6

Data Structure Documentation

6.1 ddr_bw_metrics Struct Reference

DDR bandwidth metrics.

```
#include <e_smi.h>
```

Data Fields

- uint32_t max_bw

    DDR Maximum theoritical bandwidth in GB/s.
- uint32_t utilized_bw

    DDR bandwidth utilization in GB/s.
- uint32_t utilized_pct

    DDR bandwidth utilization in % of theoritical max.

6.1.1 Detailed Description

DDR bandwidth metrics.

The documentation for this struct was generated from the following file:

- e_smi.h

6.2 dimm_power Struct Reference

DIMM Power(mW), power update rate(ms) and dimm address.

```
#include <e_smi.h>
```

Data Fields

- uint16_t power: 15

  Dimm power consumption[31:17](15 bits data)
- uint16_t update_rate: 9

  Time since last update[16:8](9 bit data)
- uint8_t dimm_addr

  Dimm address[7:0](8 bit data)

6.2.1 Detailed Description

DIMM Power(mW), power update rate(ms) and dimm address.

The documentation for this struct was generated from the following file:

- e_smi.h

6.3 dimm_thermal Struct Reference

DIMM temperature(℃) and update rate(ms) and dimm address.

```
#include <e_smi.h>
```

Data Fields

- uint16_t sensor: 11

  Dimm thermal sensor[31:21](11 bit data)
- uint16_t update_rate: 9

  Time since last update[16:8](9 bit data)
- uint8_t dimm_addr

  Dimm address[7:0](8 bit data)
- float temp

  temperature in degree celcius

6.3.1 Detailed Description

DIMM temperature(℃) and update rate(ms) and dimm address.

The documentation for this struct was generated from the following file:

- e_smi.h

6.4 dpm_level Struct Reference

max and min LCLK DPM level on a given NBIO ID. Valid max and min DPM level values are 0 - 1.

```
#include <e_smi.h>
```

Data Fields

- uint8_t max_dpm_level

  Max LCLK DPM level[15:8](8 bit data)
- uint8_t min_dpm_level

  Min LCLK DPM level[7:0](8 bit data)

6.4.1 Detailed Description

max and min LCLK DPM level on a given NBIO ID. Valid max and min DPM level values are 0 - 1.

The documentation for this struct was generated from the following file:

- e_smi.h

6.5 hsmp_driver_version Struct Reference

HSMP Driver major and minor version numbers.

```
#include <e_smi.h>
```

Data Fields

- uint32_t major

  Major version number.
- uint32_t minor

  Minor version number.

6.5.1 Detailed Description

HSMP Driver major and minor version numbers.

The documentation for this struct was generated from the following file:

- e_smi.h

6.6 link_id_bw_type Struct Reference

LINK name and Bandwidth type Information.It contains link names i.e valid link names are "P0", "P1", "P2", "P3", "P4", "G0", "G1", "G2", "G3", "G4" "G5", "G6", "G7" Valid bandwidth types 1(Aggregate_BW), 2 (Read BW), 4 (Write BW).

```
#include <e_smi.h>
```

Data Fields

- io_bw_encoding bw_type

  Bandwidth Type Information [1, 2, 4].
- char ∗ link_name

  Link name [P0, P1, G0, G1 etc].

6.6.1 Detailed Description

LINK name and Bandwidth type Information.It contains link names i.e valid link names are "P0", "P1", "P2", "P3", "P4", "G0", "G1", "G2", "G3", "G4" "G5", "G6", "G7" Valid bandwidth types 1(Aggregate_BW), 2 (Read BW), 4 (Write BW).

The documentation for this struct was generated from the following file:

- e_smi.h

6.7 smu_fw_version Struct Reference

Deconstruct raw uint32_t into SMU firmware major and minor version numbers.

```
#include <e_smi.h>
```

Data Fields

- uint8_t debug

  SMU fw Debug version number.
- uint8_t minor

  SMU fw Minor version number.
- uint8_t major

  SMU fw Major version number.
- uint8_t unused

  reserved fields

6.7.1 Detailed Description

Deconstruct raw uint32_t into SMU firmware major and minor version numbers.

The documentation for this struct was generated from the following file:

- e_smi.h

6.8 spd_info Struct Reference

Data Fields

- spd_info_inarg **m_spd_info_inarg**
- uint32_t **data**

The documentation for this struct was generated from the following file:

- e_smi.h

6.9 spd_info_inarg::spd_info_ Struct Reference

Data Fields

- uint32_t **dimm_addr**:8
- uint32_t **lid**:4
- uint32_t **reg_offset**:11
- uint32_t **reg_space**:1
- uint32_t **reserved**:8

The documentation for this struct was generated from the following file:

- e_smi.h

6.10 spd_info_inarg Union Reference

Data Structures

- struct spd_info_

Data Fields

- struct spd_info_inarg::spd_info_ **info**
- uint32_t **reg_value**

The documentation for this union was generated from the following file:

- e_smi.h

6.11 svi3_getinfo_outarg::svi3_getinfo_ Struct Reference

Data Fields

- uint32_t **svi3_temperature**:28
- uint32_t **svi3_rail_index**:3
- uint32_t **reserved**:1

The documentation for this struct was generated from the following file:

- e_smi.h

6.12 svi3_getinfo_outarg Union Reference

Data Structures

- struct svi3_getinfo_

Data Fields

- struct svi3_getinfo_outarg::svi3_getinfo_ **info**
- uint32_t **reg_value**

The documentation for this union was generated from the following file:

- e_smi.h

6.13 svi3_info Struct Reference

Data Fields

- svi3_info_inarg **m_svi3_info_inarg**
- uint32_t **temperature**

The documentation for this struct was generated from the following file:

- e_smi.h

6.14 svi3_info_inarg::svi3_info_ Struct Reference

Data Fields

- uint32_t **svi3_rail_selection**:1
- uint32_t **svi3_rail_index**:3
- uint32_t **svi3_temperature**:28

The documentation for this struct was generated from the following file:

- e_smi.h

6.15 svi3_info_inarg Union Reference

Data Structures

- struct svi3_info_

Data Fields

- struct svi3_info_inarg::svi3_info_ **info**
- uint32_t **reg_value**

The documentation for this union was generated from the following file:

- e_smi.h

6.16 temp_range_refresh_rate Struct Reference

temperature range and refresh rate metrics of a DIMM

```
#include <e_smi.h>
```

Data Fields

- uint8_t range: 3

  temp range[2:0](3 bit data)
- uint8_t ref_rate: 1

  DDR refresh rate mode[3](1 bit data)

6.16.1 Detailed Description

temperature range and refresh rate metrics of a DIMM

The documentation for this struct was generated from the following file:

- e_smi.h

Chapter 7

File Documentation

7.1 e_smi.h File Reference

```
#include <stdbool.h>
#include <asm/amd_hsmp.h>
```

Data Structures

- struct hsmp_driver_version

  HSMP Driver major and minor version numbers.
- struct smu_fw_version

  Deconstruct raw uint32_t into SMU firmware major and minor version numbers.
- struct ddr_bw_metrics

  DDR bandwidth metrics.
- struct temp_range_refresh_rate

  temperature range and refresh rate metrics of a DIMM
- struct dimm_power

  DIMM Power(mW), power update rate(ms) and dimm address.
- struct dimm_thermal

  DIMM temperature(℃) and update rate(ms) and dimm address.
- struct link_id_bw_type

  LINK name and Bandwidth type Information.It contains link names i.e valid link names are "P0", "P1", "P2", "P3", "P4", "G0", "G1", "G2", "G3", "G4" "G5", "G6", "G7" Valid bandwidth types 1(Aggregate_BW), 2 (Read BW), 4 (Write BW).
- struct dpm_level

  max and min LCLK DPM level on a given NBIO ID. Valid max and min DPM level values are 0 - 1.
- union svi3_info_inarg
- struct svi3_info_inarg::svi3_info_
- union svi3_getinfo_outarg
- struct svi3_getinfo_outarg::svi3_getinfo_
- struct svi3_info
- union spd_info_inarg
- struct spd_info_inarg::spd_info_
- struct spd_info

Macros

- #define ENERGY_DEV_NAME "amd_energy"

  Supported Energy driver name.
- #define HSMP_CHAR_DEVFILE_NAME "/dev/hsmp"

  HSMP device path.
- #define HSMP_METRICTABLE_PATH "/sys/devices/platform/amd_hsmp"

  HSMP MetricTable sysfs path.
- #define ARRAY_SIZE(arr) (sizeof(arr) / sizeof((arr)[0]))

  macro to calculate size
- #define BIT(N) (1 << N)

  macro for mask

Enumerations

- enum io_bw_encoding { AGG_BW = BIT(0) , RD_BW = BIT(1) , WR_BW = BIT(2) }

  xGMI Bandwidth Encoding types
- enum esmi_status_t {
  ESMI_SUCCESS = 0 , ESMI_INITIALIZED = 0 , ESMI_NO_ENERGY_DRV , ESMI_NO_MSR_DRV ,
  ESMI_NO_HSMP_DRV , ESMI_NO_HSMP_SUP , ESMI_NO_DRV , ESMI_FILE_NOT_FOUND ,
  ESMI_DEV_BUSY , ESMI_PERMISSION , ESMI_NOT_SUPPORTED , ESMI_FILE_ERROR ,
  ESMI_INTERRUPTED , ESMI_IO_ERROR , ESMI_UNEXPECTED_SIZE , ESMI_UNKNOWN_ERROR ,
  ESMI_ARG_PTR_NULL , ESMI_NO_MEMORY , ESMI_NOT_INITIALIZED , ESMI_INVALID_INPUT ,
  ESMI_HSMP_TIMEOUT , ESMI_NO_HSMP_MSG_SUP , ESMI_PRE_REQ_NOT_SAT , ESMI_SMU_BUSY
  }

  Error codes retured by E-SMI functions.

Functions

- esmi_status_t esmi_init (void)

  Initialize the library, validates the dependencies.
- void esmi_exit (void)

  Clean up any allocation done during init.
- esmi_status_t esmi_core_energy_get (uint32_t core_ind, uint64_t ∗penergy)

  Get the core energy for a given core.
- esmi_status_t esmi_socket_energy_get (uint32_t socket_idx, uint64_t ∗penergy)

  Get the socket energy for a given socket.
- esmi_status_t esmi_all_energies_get (uint64_t ∗penergy)

  Get energies of all cores in the system.
- esmi_status_t esmi_rapl_units_hsmp_mailbox_get (uint32_t sock_ind, uint8_t ∗tu, uint8_t ∗esu)

  Get the RAPL units through HSMP mailbox.
- esmi_status_t esmi_rapl_package_counter_hsmp_mailbox_get (uint32_t sock_ind, uint32_t ∗counter1, uint32_t ∗counter0)

  Get the socket energy counter values for a given socket through mailbox.
- esmi_status_t esmi_rapl_core_counter_hsmp_mailbox_get (uint32_t core_ind, uint32_t ∗counter1, uint32_t ∗counter0)

  Get the core energy counter values for a given socket through mailbox.
- esmi_status_t esmi_core_energy_hsmp_mailbox_get (uint32_t core_ind, uint64_t ∗penergy)

  Get the core energy for a given core through HSMP mailbox.
- esmi_status_t esmi_package_energy_hsmp_mailbox_get (uint32_t sock_ind, uint64_t ∗penergy)

  Get the socket energy for a given socket through mailbox.

- esmi_status_t esmi_hsmp_driver_version_get (struct hsmp_driver_version ∗hsmp_driver_ver)

  Get the HSMP Driver version.
- esmi_status_t esmi_smu_fw_version_get (struct smu_fw_version ∗smu_fw)

  Get the SMU Firmware Version.
- esmi_status_t esmi_prochot_status_get (uint32_t socket_idx, uint32_t ∗prochot)

  Get normalized status of the processor's PROCHOT status. 1 - PROCHOT active, 0 - PROCHOT inactive.
- esmi_status_t esmi_fclk_mclk_get (uint32_t socket_idx, uint32_t ∗fclk, uint32_t ∗mclk)

  Get the Data Fabric clock and Memory clock in MHz, for a given socket index.
- esmi_status_t esmi_cclk_limit_get (uint32_t socket_idx, uint32_t ∗cclk)

  Get the core clock (MHz) allowed by the most restrictive infrastructure limit at the time of the message.
- esmi_status_t esmi_hsmp_proto_ver_get (uint32_t ∗proto_ver)

  Get the HSMP interface (protocol) version.
- esmi_status_t esmi_socket_current_active_freq_limit_get (uint32_t sock_ind, uint16_t ∗freq, char ∗∗src_←
type)

  Get the current active frequency limit of the socket.
- esmi_status_t esmi_socket_freq_range_get (uint8_t sock_ind, uint16_t ∗fmax, uint16_t ∗fmin)

  Get the Socket frequency range.
- esmi_status_t esmi_current_freq_limit_core_get (uint32_t core_id, uint32_t ∗freq)

  Get the current active frequency limit of the core.
- esmi_status_t esmi_cpurail_isofreq_policy_get (uint8_t sock_ind, bool ∗val)

  Get the CpuRailIsoFreqPolicy.
- esmi_status_t esmi_dfc_ctrl_setting_get (uint8_t sock_ind, bool ∗val)

  get the DfcEnable.
- esmi_status_t esmi_socket_power_get (uint32_t socket_idx, uint32_t ∗ppower)

  Get the instantaneous power consumption of the provided socket.
- esmi_status_t esmi_socket_power_cap_get (uint32_t socket_idx, uint32_t ∗pcap)

  Get the current power cap value for a given socket.
- esmi_status_t esmi_socket_power_cap_max_get (uint32_t socket_idx, uint32_t ∗pmax)

  Get the maximum power cap value for a given socket.
- esmi_status_t esmi_pwr_svi_telemetry_all_rails_get (uint32_t sock_ind, uint32_t ∗power)

  Get the SVI based power telemetry for all rails.
- esmi_status_t esmi_pwr_efficiency_mode_get (uint8_t sock_ind, uint8_t ∗mode)

  Get the current power efficiency mode.
- esmi_status_t esmi_read_ccd_power (uint32_t coreid, uint32_t ∗power)

  Get CCD power.
- esmi_status_t esmi_socket_power_cap_set (uint32_t socket_idx, uint32_t pcap)

  Set the power cap value for a given socket.
- esmi_status_t esmi_pwr_efficiency_mode_set (uint8_t sock_ind, uint8_t mode)

  Set the power efficiency profile policy.
- esmi_status_t esmi_cpurail_isofreq_policy_set (uint8_t sock_ind, bool ∗val)

  Set the CpuRailIsoFreqPolicy.
- esmi_status_t esmi_dfc_enable_set (uint8_t sock_ind, bool ∗val)

  Set the DfcEnable.
- esmi_status_t esmi_core_boostlimit_get (uint32_t cpu_ind, uint32_t ∗pboostlimit)

  Get the boostlimit value for a given core.
- esmi_status_t esmi_socket_c0_residency_get (uint32_t socket_idx, uint32_t ∗pc0_residency)

  Get the c0_residency value for a given socket.
- esmi_status_t esmi_core_boostlimit_set (uint32_t cpu_ind, uint32_t boostlimit)

  Set the boostlimit value for a given core.
- esmi_status_t esmi_socket_boostlimit_set (uint32_t socket_idx, uint32_t boostlimit)

  Set the boostlimit value for a given socket.

- esmi_status_t esmi_ddr_bw_get (uint8_t sock_ind, struct ddr_bw_metrics ∗ddr_bw)

  Get the Theoretical maximum DDR Bandwidth in GB/s, Current utilized DDR Bandwidth in GB/s and Current utilized DDR Bandwidth as a percentage of theoretical maximum in a system. Supported only on hsmp protocol version >= 3.
- esmi_status_t esmi_socket_temperature_get (uint32_t sock_ind, uint32_t ∗ptmon)

  Get temperature monitor for a given socket.
- esmi_status_t esmi_read_tdelta (uint8_t sock_ind, uint8_t ∗status)

  Get thermal solution behaviour for a given socket.
- esmi_status_t esmi_get_svi3_vr_controller_temp (uint8_t sock_ind, struct svi3_info ∗inout)

  Get temperature of svi3 VR controller rail for a given socket.
- esmi_status_t esmi_dimm_temp_range_and_refresh_rate_get (uint8_t sock_ind, uint8_t dimm_addr, struct temp_range_refresh_rate ∗rate)

  Get dimm temperature range and refresh rate.
- esmi_status_t esmi_dimm_power_consumption_get (uint8_t sock_ind, uint8_t dimm_addr, struct dimm_power ∗dimm_pow)

  Get dimm power consumption and update rate.
- esmi_status_t esmi_dimm_thermal_sensor_get (uint8_t sock_ind, uint8_t dimm_addr, struct dimm_thermal ∗dimm_temp)

  Get dimm thermal sensor.
- esmi_status_t esmi_spd_sb_reg_read (uint8_t sock_ind, struct spd_info ∗inout)

  Execute a four byte read transaction at a given register offset in a specified device on the target DIMM. Supported only on hsmp protocol version 7.
- esmi_status_t esmi_xgmi_width_set (uint8_t sock_ind, uint8_t min, uint8_t max)

  Set xgmi width for a multi socket system. values range from 0 to 2.
- esmi_status_t esmi_xgmi_width_get (uint32_t sock_ind, uint8_t ∗min, uint8_t ∗max)

  Get xgmi width for a multi socket system. values range from 0 to 2.
- esmi_status_t esmi_gmi3_link_width_range_set (uint8_t sock_ind, uint8_t min_link_width, uint8_t max_↩ link_width)

  Set gmi3 width.
- esmi_status_t esmi_apb_enable (uint32_t sock_ind)

  Enable automatic P-state selection.
- esmi_status_t esmi_apb_disable (uint32_t sock_ind, uint8_t pstate)

  Set data fabric P-state to user specified value.
- esmi_status_t esmi_apb_status_get (uint32_t sock_ind, uint8_t ∗apb_disabled, uint8_t ∗pstate)

  Get APBDisabled status and gets data fabric P-state if APBDisabled.
- esmi_status_t esmi_socket_lclk_dpm_level_set (uint32_t sock_ind, uint8_t nbio_id, uint8_t min, uint8_t max)

  Set lclk dpm level.
- esmi_status_t esmi_socket_lclk_dpm_level_get (uint8_t sock_ind, uint8_t nbio_id, struct dpm_level ∗nbio)

  Get lclk dpm level.
- esmi_status_t esmi_pcie_link_rate_set (uint8_t sock_ind, uint8_t rate_ctrl, uint8_t ∗prev_mode)

  Set pcie link rate.
- esmi_status_t esmi_df_pstate_range_set (uint8_t sock_ind, uint8_t min_pstate, uint8_t max_pstate)

  Set data fabric pstate range.
- esmi_status_t esmi_df_pstate_range_get (uint8_t sock_ind, uint8_t ∗min_pstate, uint8_t ∗max_pstate)

  Get data fabric pstate range.
- esmi_status_t esmi_xgmi_pstate_range_set (uint8_t sock_ind, uint8_t min_state, uint8_t max_state)

  Set xgmi pstate range.
- esmi_status_t esmi_xgmi_pstate_range_get (uint8_t sock_ind, uint8_t ∗min_state, uint8_t ∗max_state)

  Get xgmi pstate range.
- esmi_status_t esmi_pc6_enable_set (uint8_t sock_ind, uint8_t pc6_enable)

  Set PC6 enable.
- esmi_status_t esmi_pc6_enable_get (uint8_t sock_ind, uint8_t ∗current_pc6_enable)

Get PC6 enable.
- esmi_status_t esmi_cc6_enable_set (uint8_t sock_ind, uint8_t cc6_enable)

  Set CC6 enable.
- esmi_status_t esmi_cc6_enable_get (uint8_t sock_ind, uint8_t ∗current_cc6_enable)

  Get CC6 enable.
- esmi_status_t esmi_current_io_bandwidth_get (uint8_t sock_ind, struct link_id_bw_type link, uint32_t ∗io_↩ bw)

  Get IO bandwidth on IO link.
- esmi_status_t esmi_current_xgmi_bw_get (uint8_t sock_ind, struct link_id_bw_type link, uint32_t ∗xgmi_bw)

  Get xGMI bandwidth.
- esmi_status_t esmi_metrics_table_version_get (uint32_t ∗metrics_version)

  Get metrics table version.
- esmi_status_t esmi_metrics_table_get (uint8_t sock_ind, struct hsmp_metric_table ∗metrics_table)

  Get metrics table.
- esmi_status_t esmi_dram_address_metrics_table_get (uint8_t sock_ind, uint64_t ∗dram_addr)

  Get the DRAM address for the metrics table.
- esmi_status_t esmi_test_hsmp_mailbox (uint8_t sock_ind, uint32_t ∗data)

  Test HSMP mailbox interface.
- esmi_status_t esmi_cpu_family_get (uint32_t ∗family)

  Get the CPU family.
- esmi_status_t esmi_cpu_model_get (uint32_t ∗model)

  Get the CPU model.
- esmi_status_t esmi_threads_per_core_get (uint32_t ∗threads)

  Get the number of threads per core in the system.
- esmi_status_t esmi_number_of_cpus_get (uint32_t ∗cpus)

  Get the number of cpus available in the system.
- esmi_status_t esmi_number_of_sockets_get (uint32_t ∗sockets)

  Get the total number of sockets available in the system.
- esmi_status_t esmi_first_online_core_on_socket (uint32_t socket_idx, uint32_t ∗pcore_ind)

  Get the first online core on a given socket.
- char ∗ esmi_get_err_msg (esmi_status_t esmi_err)

  Get the error string message for esmi errors.

7.1.1 Detailed Description

Main header file for the E-SMI library. All required function, structure, enum, etc. definitions should be defined in this file.

This header file contains the following: APIs prototype of the APIs exported by the E-SMI library. Description of the API, arguments and return values. The Error codes returned by the API.

7.1.2 Enumeration Type Documentation

7.1.2.1 io_bw_encoding

```
enum io_bw_encoding
```

xGMI Bandwidth Encoding types

Enumerator

AGG_BW	Aggregate Bandwidth.
RD_BW	Read Bandwidth.
WR_BW	Write Bandwdith.

7.1.2.2 esmi_status_t

```
enum esmi_status_t
```

Error codes retured by E-SMI functions.

Enumerator

ESMI_SUCCESS	Operation was successful.
ESMI_INITIALIZED	ESMI initialized successfully.
ESMI_NO_ENERGY_DRV	Energy driver not found.
ESMI_NO_MSR_DRV	MSR driver not found.
ESMI_NO_HSMP_DRV	HSMP driver not found.
ESMI_NO_HSMP_SUP	HSMP not supported.
ESMI_NO_DRV	No Energy and HSMP driver present.
ESMI_FILE_NOT_FOUND	file or directory not found
ESMI_DEV_BUSY	Device or resource busy.
ESMI_PERMISSION	Many functions require root access to run. Permission denied/EACCESS file error.
ESMI_NOT_SUPPORTED	The requested information or action is not available for the given input, on the given system
ESMI_FILE_ERROR	Problem accessing a file. This may because the operation is not supported by the Linux kernel version running on the executing machine
ESMI_INTERRUPTED	execution of function An interrupt occurred during
ESMI_IO_ERROR	An input or output error.
ESMI_UNEXPECTED_SIZE	was read An unexpected amount of data
ESMI_UNKNOWN_ERROR	An unknown error occurred.
ESMI_ARG_PTR_NULL	Parsed argument is invalid.
ESMI_NO_MEMORY	Not enough memory to allocate.
ESMI_NOT_INITIALIZED	ESMI path not initialized.
ESMI_INVALID_INPUT	Input value is invalid.
ESMI_HSMP_TIMEOUT	HSMP message is timedout.
ESMI_NO_HSMP_MSG_SUP	HSMP message/feature not supported.
ESMI_PRE_REQ_NOT_SAT	Prerequisite to execute the command not satisfied.
ESMI_SMU_BUSY	SMU is busy.

Index