E-SMI Library: EPYC™ Systems Management Interface Library

Generated by Doxygen 1.8.17

1 EPYC™ System Management Interface (E-SMI) In-band Library 1

1.1 Important note about Versioning and Backward Compatibility 1

1.2 Building E-SMI 1

1.2.1 Dowloading the source 1

1.2.2 Directory stucture of the source 1

1.2.3 Building the library and tool 2

1.2.4 Building the library for static linking 2

1.2.5 Building the Documentation 2

1.2.6 Building the package 2

1.3 Kernel dependencies 3

1.3.1 Monitoring energy counters 3

1.3.2 Monitoring and managing power metrics, boostlimits and other system management features 3

1.3.3 Supported hardware 3

1.3.4 Additional required software for building 3

1.4 Usage Basics 4

1.4.1 Device Indices 4

1.4.1.1 Hello E-SMI 4

1.5 Usage 4

1.5.1 Tool Usage 4

2 Module Index 7

2.1 Modules 7

3 Data Structure Index 9

3.1 Data Structures 9

4 File Index 11

4.1 File List 11

5 Module Documentation 13

5.1 Initialization and Shutdown 13

5.1.1 Detailed Description 13

5.1.2 Function Documentation 13

5.1.2.1 esmi_init() 13

5.2 Energy Monitor (RAPL MSR) 14

5.2.1 Detailed Description 14

5.2.2 Function Documentation 14

5.2.2.1 esmi_core_energy_get() 14

5.2.2.2 esmi_socket_energy_get() 15

5.2.2.3 esmi_all_energies_get() 15

5.3 HSMP System Statistics 16

5.3.1 Detailed Description 16

5.3.2 Function Documentation 16

5.3.2.1 esmi_smu_fw_version_get() 16

5.3.2.2 esmi_prochot_status_get() 17
5.3.2.3 esmi_fclk_mclk_get() 17
5.3.2.4 esmi_cclk_limit_get() 17
5.3.2.5 esmi_hsmp_proto_ver_get() 18
5.4 Power Monitor 19
5.4.1 Detailed Description 19
5.4.2 Function Documentation 19
5.4.2.1 esmi_socket_power_get() 19
5.4.2.2 esmi_socket_power_cap_get() 20
5.4.2.3 esmi_socket_power_cap_max_get() 20
5.5 Power Control 21
5.5.1 Detailed Description 21
5.5.2 Function Documentation 21
5.5.2.1 esmi_socket_power_cap_set() 21
5.6 Performance (Boost limit) Monitor 22
5.6.1 Detailed Description 22
5.6.2 Function Documentation 22
5.6.2.1 esmi_core_boostlimit_get() 22
5.6.2.2 esmi_socket_c0_residency_get() 22
5.7 Performance (Boost limit) Control 24
5.7.1 Detailed Description 24
5.7.2 Function Documentation 24
5.7.2.1 esmi_core_boostlimit_set() 24
5.7.2.2 esmi_socket_boostlimit_set() 24
5.7.2.3 esmi_package_boostlimit_set() 25
5.8 ddr_bandwidth Monitor 26
5.8.1 Detailed Description 26
5.8.2 Function Documentation 26
5.8.2.1 esmi_ddr_bw_get() 26
5.9 Temperature Query 27
5.9.1 Detailed Description 27
5.9.2 Function Documentation 27
5.9.2.1 esmi_socket_temperature_get() 27
5.10 xGMI bandwidth control 28
5.10.1 Detailed Description 28
5.10.2 Function Documentation 28
5.10.2.1 esmi_xgmi_width_set() 28
5.11 APB and LCLK level control 29
5.11.1 Detailed Description 29
5.11.2 Function Documentation 29
5.11.2.1 esmi_apb_enable() 29
5.11.2.2 esmi_apb_disable() 30

5.11.2.3 esmi_socket_lclk_dpm_level_set() 30
5.12 Auxiliary functions 31
5.12.1 Detailed Description 31
5.12.2 Function Documentation 31
5.12.2.1 esmi_cpu_family_get() 31
5.12.2.2 esmi_cpu_model_get() 32
5.12.2.3 esmi_threads_per_core_get() 32
5.12.2.4 esmi_number_of_cpus_get() 32
5.12.2.5 esmi_number_of_sockets_get() 33
5.12.2.6 esmi_first_online_core_on_socket() 33
5.12.2.7 esmi_get_err_msg() 34

6 Data Structure Documentation **35**
6.1 ddr_bw_metrics Struct Reference 35
6.1.1 Detailed Description 35
6.2 smu_fw_version Struct Reference 35
6.2.1 Detailed Description 36

7 File Documentation **37**
7.1 e_smi.h File Reference 37
7.1.1 Detailed Description 39
7.1.2 Enumeration Type Documentation 39
7.1.2.1 esmi_status_t 39

Index **41**

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 on `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
- `$ example/` 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

1.2.3 Building the library and tool

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

1.2.3.0.1 <tt>$ mkdir -p build</tt>

1.2.3.0.2 <tt>$ cd build</tt>

1.2.3.0.3 <tt>$ cmake <location of root of E-SMI library CMakeLists.txt></tt>

1.2.4 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'. The next step can be skipped if static lib support is not required

1.2.4.0.1 <tt>$ cmake -DENABLE_STATIC_LIB=1 <location of root of E-SMI library CMakeLists.txt></tt>

1.2.4.0.2 <tt>$ make</tt> The built library `libe_smi64.so.X.Y` will appear in the `build` folder.

1.2.4.0.3 <tt># Install library file and header; default location is /opt/esmi</tt>

1.2.4.0.4 <tt>$ sudo make install</tt>

1.2.5 Building the Documentation

The documentation PDF file can be built with the following steps (continued from the steps above):

1.2.5.0.1 <tt>$ make doc</tt> Upon a successful build, the `ESMI_Manual.pdf` and `ESMI_IB_Release_Notes.pdf` will be copied to the top directory of the source.

1.2.6 Building the package

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

1.2.6.0.1 <tt>$ make package</tt>

1.3 Kernel dependencies

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

1.3.1 Monitoring energy counters

The Energy counters reported by the RAPL MSRs, the AMD Energy driver exposes the per core and per socket counters via the HWMON sys entries. The AMD Energy driver is upstreamed and is available as part of Linux v5.8+. The kernel config symbol SENSORS_AMD_ENERGY needs to be selected, can be built and inserted as a module.

1.3.2 Monitoring and managing power metrics, boostlimits and other system management features

The power metrics, boostlimits and other features are managed by the SMU firmware and exposed via PCI config space. AMD provides Linux kernel module exposing this information to the user-space via sys entries.

- amd_hsmp module will be made available at `https://github.com/amd/amd_hsmp.git`
- PCIe interface needs to be enabled in the BIOS. On the reference BIOS, the CBS option may be found in the following path

`BIOS Default: "Auto" (Disabled)` If the option is disabled, the related E-SMI APIs will return -ET↩
IMEDOUT.

1.3.3 Supported hardware

AMD Zen3 based CPU Family `19h` Models `0h-Fh` and `30h-3Fh`.

1.3.4 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)

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

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

1.4 Usage Basics

1.4.1 Device Indices

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.

1.4.1.1 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_get_number_of_sockets();
    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;
}
```

1.5 Usage

1.5.1 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 the functionality, with default core/socket/package as 0.

```
e_smi_library/build> sudo ./e_smi_tool
=============== EPYC System Management Interface ===============
--------------------------------------
| CPU Family            | 0x19 (25 ) |
| CPU Model             | 0x0  (0  ) |
| NR_CPUS               | 128        |
| NR_SOCKETS            | 2          |
| THREADS PER CORE      | 1 (SMT OFF)|
--------------------------------------
```

```
-----------------------------------------------------------------
| Sensor Name            | Socket 0         | Socket 1         |
-----------------------------------------------------------------
| Energy (K Joules)      | 206.088          | 212.171          |
| Power (Watts)          | 42.224           | 42.634           |
| PowerLimit (Watts)     | 200.000          | 120.000          |
| PowerLimitMax (Watts)  | 225.000          | 225.000          |
| C0 Residency (%)       | 0                | 0                |
-----------------------------------------------------------------
| Core[0] Energy (Joules)| 6.123            | 5.520            |
| Core[0] boostlimit(MHz)| 2500             | 2000             |
-----------------------------------------------------------------
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:

```
e_smi_library/build> ./e_smi_tool --help
=============== EPYC System Management Interface ===============
Usage: ./e_smi_tool [Option]... <INPUT>...
Output Option<s>:
  -h, --help                         Show this help message
  -A, --showall                      Get all esmi parameter Values
Get Option<s>:
  -e, --showcoreenergy [CORE]                Get energy for a given CPU (Joules)
  -s, --showsockenergy                   Get energy for all sockets (KJoules)
  -p, --showsockpower                    Get power metrics for all sockets (mWatts)
  -L, --showcorebl [CORE]                Get Boostlimit for a given CPU (MHz)
  -r, --showsockc0res [SOCKET]               Get c0_residency for a given socket (%)
  -d, --showddrbw                    Show DDR bandwidth details (Gbps)
  -t, --showsockettemp                   Show Temperature monitor of socket (°C)
  --showsmufwver                     Show SMU FW Version
  --showhsmpprotover                     Show HSMP Protocol Version
  --showprochotstatus                    Show HSMP PROCHOT status (in/active)
  --showclocks                       Show (CPU, Mem & Fabric) clock frequencies (MHz)
Set Option<s>:
  -C, --setpowerlimit [SOCKET] [POWER]           Set power limit for a given socket (mWatts)
  -a, --setcorebl [CORE] [BOOSTLIMIT]            Set boost limit for a given core (MHz)
  --setsockbl [SOCKET] [BOOSTLIMIT]          Set Boost limit for a given Socket (MHz)
  --setpkgbl [BOOSTLIMIT]                Set Boost limit for all sockets in a package (MHz)
  --apbdisable [SOCKET] [PSTATE]             Set Data Fabric Pstate for a given socket, PSTATE = 0 to 3
  --apbenable [SOCKET]                   Enable the Data Fabric performance boost algorithm for a given
       socket
  --setxgmiwidth [MIN] [MAX]                 Set xgmi link width in a multi socket system, MIN = MAX = 0 to 2
  --setlclkdpmlevel [SOCKET] [NBIOID] [MIN] [MAX]    Set lclk dpm level for a given nbio, given socket, MIN =
       MAX = NBIOID = 0 to 3
====================== End of EPYC SMI Log ======================
```

Below is a sample usage to get the individual library functionality API's. We can pass arguments in short or long options ex: "./e_smi_tool -e 0" or "./e_smi_tool --showcoreenergy 0"

```
1.  e_smi_library/build> ./e_smi_tool -e 0
    =============== EPYC System Management Interface ===============
    core[0] energy  :         17211.219 Joules
    ====================== End of EPYC SMI Log ======================
2.  e_smi_library/build> ./e_smi_tool --showcoreenergy 0
    =============== EPYC System Management Interface ===============
    core[0] energy  :         17216.800 Joules
    ====================== End of EPYC SMI Log ======================
3.      e_smi_library/build>./e_smi_tool -e 12 --showsockpower --setpowerlimit 1 220000 -p
    =============== EPYC System Management Interface ===============
    core[12] energy :           246.251 Joules
    -----------------------------------------------------------------
    | Sensor Name            | Socket 0         | Socket 1         |
    -----------------------------------------------------------------
    | Power (Watts)          | 66.508           | 67.548           |
    | PowerLimit (Watts)     | 22.000           | 220.000          |
    | PowerLimitMax (Watts)  | 240.000          | 240.000          |
    -----------------------------------------------------------------
    Set socket[1] power_limit :         220.000 Watts successfully
    -----------------------------------------------------------------
    | Sensor Name            | Socket 0         | Socket 1         |
    -----------------------------------------------------------------
    | Power (Watts)          | 66.520           | 67.556           |
    | PowerLimit (Watts)     | 22.000           | 220.000          |
    | PowerLimitMax (Watts)  | 240.000          | 240.000          |
    -----------------------------------------------------------------
    ====================== End of EPYC SMI Log ======================
```

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Initialization and Shutdown 13
Energy Monitor (RAPL MSR) 14
HSMP System Statistics 16
Power Monitor 19
Power Control 21
Performance (Boost limit) Monitor 22
Performance (Boost limit) Control 24
ddr_bandwidth Monitor 26
Temperature Query 27
xGMI bandwidth control 28
APB and LCLK level control 29
Auxiliary functions 31

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

ddr_bw_metrics
DDR bandwidth metrics 35
smu_fw_version
Deconstruct raw uint32_t into SMU firmware major and minor version numbers 35

Chapter 4

File Index

4.1 File List

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

e_smi.h 37

Chapter 5

Module Documentation

5.1 Initialization and Shutdown

This function validates the dependencies exists and initializes the library.

Functions

- esmi_status_t esmi_init (void)

  Initialize the library, validate the dependencies exists.
- void esmi_exit (void)

  Clean up allocation during init.

5.1.1 Detailed Description

This function validates the dependencies exists and initializes the library.

5.1.2 Function Documentation

5.1.2.1 esmi_init()

```
esmi_status_t esmi_init (
             void  )
```

Initialize the library, validate the dependencies exists.

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)

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.

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.

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.
None-zero	is returned upon failure.

5.3 HSMP System Statistics

Below functions to get HSMP System Statistics.

Functions

- 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.

5.3.1 Detailed Description

Below functions to get HSMP System Statistics.

5.3.2 Function Documentation

5.3.2.1 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`

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.

5.3.2.2 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`.

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.

5.3.2.3 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.

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.

5.3.2.4 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.

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.

5.3.2.5 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`

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.

5.4 Power Monitor

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

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.

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`.

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.

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`.

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.

5.5 Power Control

This function provides a way to control Power Limit.

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.

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 power socket reduction occurs if the limit is set below that minimum.

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.

5.6 Performance (Boost limit) Monitor

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

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 will return the core's current boost limit `pboostlimit` for a particular `cpu_ind`

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`

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

Below functions provide ways to control Boost limit values.

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.
- esmi_status_t esmi_package_boostlimit_set (uint32_t boostlimit)

  Set the boostlimit value for the package (whole system).

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`.

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

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`.

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.

Return values

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

5.7.2.3 esmi_package_boostlimit_set()

```
esmi_status_t esmi_package_boostlimit_set (
            uint32_t boostlimit )
```

Set the boostlimit value for the package (whole system).

This function will set the boostlimit to the provided value `boostlimit` for the whole package.

Parameters

in	*boostlimit*	a uint32_t that indicates the desired boostlimit value of the package.

Return values

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

5.8 ddr_bandwidth Monitor

This function provides the DDR Bandwidth for a system.

Functions

- esmi_status_t esmi_ddr_bw_get (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.

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 (
            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.

This function will return the DDR Bandwidth metrics `ddr_bw`

Parameters

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

This function provides the current tempearature value in degree C.

Functions

- esmi_status_t esmi_socket_temperature_get (uint32_t sock_ind, uint32_t *ptmon)

    Get temperature monitor 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`.

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.

5.10 xGMI bandwidth control

This function provides a way to control xgmi bandwidth connected in 2P systems.

Functions

- esmi_status_t esmi_xgmi_width_set (uint8_t min, uint8_t max)

  Set xgmi width for a multi socket system.

5.10.1 Detailed Description

This function provides a way to control xgmi bandwidth connected in 2P systems.

5.10.2 Function Documentation

5.10.2.1 esmi_xgmi_width_set()

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

Set xgmi width for a multi socket system.

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

Parameters

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 APB and LCLK level control

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

Functions

- esmi_status_t esmi_apb_enable (uint32_t sock_ind, bool ∗prochot_asserted)

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

  Set data fabric P-state to user specified value.
- 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.

5.11.1 Detailed Description

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

5.11.2 Function Documentation

5.11.2.1 esmi_apb_enable()

```
esmi_status_t esmi_apb_enable (
            uint32_t sock_ind,
            bool * prochot_asserted )
```

Enable automatic P-state selection.

Given a socket index `sock_ind`, this function will enable performance boost algorithm provided `prochot_asserted` is not asserted

Parameters

in	*sock_ind*	a socket index
in,out	*prochot_asserted*	input buffer to fill the prochot status

Return values

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

5.11.2.2 esmi_apb_disable()

```
esmi_status_t esmi_apb_disable (
             uint32_t sock_ind,
             uint8_t pstate,
             bool * prochot_asserted )
```

Set data fabric P-state to user specified value.

This function will set the desired P-state at `pstate`. provided the `prochot_asserted` is not asserted for `sock_ind`. Acceptable values for the P-state are 0(highest) - 3 (lowest).

Parameters

in	*sock_ind*	a socket index
in	*pstate*	a uint8_t that indicates the desired P-state to set.
in,out	*prochot_asserted*	input buffer to fill the proc hot status.

Return values

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

5.11.2.3 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`.

Parameters

in	*sock_ind*	socket index.
in	*nbio_id*	northbridge number varies from 0 to 3.
in	*min*	pstate minimum value, varies from 0 to 3 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.

5.12 Auxiliary functions

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.

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.

5.12.1 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.

5.12.2 Function Documentation

5.12.2.1 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.

5.12.2.2 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.

5.12.2.3 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.

5.12.2.4 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.

5.12.2.5 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.

5.12.2.6 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.

5.12.2.7 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 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

6.2.1 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

Chapter 7

File Documentation

7.1 e_smi.h File Reference

```
#include <stdbool.h>
```

Data Structures

- struct smu_fw_version

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

  DDR bandwidth metrics.

Macros

- #define ENERGY_DEV_NAME "amd_energy"

  Supported Energy driver name.
- #define HSMP_DEV_NAME "amd_hsmp"

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

  HSMP device path.

Enumerations

- enum esmi_status_t {
  ESMI_SUCCESS = 0, ESMI_INITIALIZED = 0, ESMI_NO_ENERGY_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 }

  Error codes retured by E-SMI functions.

Functions

- esmi_status_t esmi_init (void)

  Initialize the library, validate the dependencies exists.
- void esmi_exit (void)

  Clean up allocation 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_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_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_socket_power_cap_set (uint32_t socket_idx, uint32_t pcap)

  Set the power cap value for a given socket.
- 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_package_boostlimit_set (uint32_t boostlimit)

  Set the boostlimit value for the package (whole system).
- esmi_status_t esmi_ddr_bw_get (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.
- 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_xgmi_width_set (uint8_t min, uint8_t max)

  Set xgmi width for a multi socket system.
- esmi_status_t esmi_apb_enable (uint32_t sock_ind, bool ∗prochot_asserted)

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

  Set data fabric P-state to user specified value.

- 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_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 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_HSMP_DRV	HSMP driver not found.
ESMI_NO_HSMP_SUP	HSMP feature 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

Enumerator

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.

Index

APB and LCLK level control, 29
  esmi_apb_disable, 29
  esmi_apb_enable, 29
  esmi_socket_lclk_dpm_level_set, 30
Auxiliary functions, 31
  esmi_cpu_family_get, 31
  esmi_cpu_model_get, 32
  esmi_first_online_core_on_socket, 33
  esmi_get_err_msg, 34
  esmi_number_of_cpus_get, 32
  esmi_number_of_sockets_get, 33
  esmi_threads_per_core_get, 32

ddr_bandwidth Monitor, 26
  esmi_ddr_bw_get, 26
ddr_bw_metrics, 35

e_smi.h, 37
  ESMI_ARG_PTR_NULL, 40
  ESMI_DEV_BUSY, 39
  ESMI_FILE_ERROR, 40
  ESMI_FILE_NOT_FOUND, 39
  ESMI_INITIALIZED, 39
  ESMI_INTERRUPTED, 40
  ESMI_INVALID_INPUT, 40
  ESMI_IO_ERROR, 40
  ESMI_NO_DRV, 39
  ESMI_NO_ENERGY_DRV, 39
  ESMI_NO_HSMP_DRV, 39
  ESMI_NO_HSMP_SUP, 39
  ESMI_NO_MEMORY, 40
  ESMI_NOT_INITIALIZED, 40
  ESMI_NOT_SUPPORTED, 39
  ESMI_PERMISSION, 39
  esmi_status_t, 39
  ESMI_SUCCESS, 39
  ESMI_UNEXPECTED_SIZE, 40
  ESMI_UNKNOWN_ERROR, 40
Energy Monitor (RAPL MSR), 14
  esmi_all_energies_get, 15
  esmi_core_energy_get, 14
  esmi_socket_energy_get, 15
esmi_all_energies_get
  Energy Monitor (RAPL MSR), 15
esmi_apb_disable
  APB and LCLK level control, 29
esmi_apb_enable
  APB and LCLK level control, 29
ESMI_ARG_PTR_NULL
  e_smi.h, 40
esmi_cclk_limit_get
  HSMP System Statistics, 17
esmi_core_boostlimit_get
  Performance (Boost limit) Monitor, 22
esmi_core_boostlimit_set
  Performance (Boost limit) Control, 24
esmi_core_energy_get
  Energy Monitor (RAPL MSR), 14
esmi_cpu_family_get
  Auxiliary functions, 31
esmi_cpu_model_get
  Auxiliary functions, 32
esmi_ddr_bw_get
  ddr_bandwidth Monitor, 26
ESMI_DEV_BUSY
  e_smi.h, 39
esmi_fclk_mclk_get
  HSMP System Statistics, 17
ESMI_FILE_ERROR
  e_smi.h, 40
ESMI_FILE_NOT_FOUND
  e_smi.h, 39
esmi_first_online_core_on_socket
  Auxiliary functions, 33
esmi_get_err_msg
  Auxiliary functions, 34
esmi_hsmp_proto_ver_get
  HSMP System Statistics, 18
esmi_init
  Initialization and Shutdown, 13
ESMI_INITIALIZED
  e_smi.h, 39
ESMI_INTERRUPTED
  e_smi.h, 40
ESMI_INVALID_INPUT
  e_smi.h, 40
ESMI_IO_ERROR
  e_smi.h, 40
ESMI_NO_DRV
  e_smi.h, 39
ESMI_NO_ENERGY_DRV
  e_smi.h, 39
ESMI_NO_HSMP_DRV
  e_smi.h, 39
ESMI_NO_HSMP_SUP
  e_smi.h, 39
ESMI_NO_MEMORY
  e_smi.h, 40
ESMI_NOT_INITIALIZED

e_smi.h, 40
ESMI_NOT_SUPPORTED
e_smi.h, 39
esmi_number_of_cpus_get
Auxiliary functions, 32
esmi_number_of_sockets_get
Auxiliary functions, 33
esmi_package_boostlimit_set
Performance (Boost limit) Control, 25
ESMI_PERMISSION
e_smi.h, 39
esmi_prochot_status_get
HSMP System Statistics, 16
esmi_smu_fw_version_get
HSMP System Statistics, 16
esmi_socket_boostlimit_set
Performance (Boost limit) Control, 24
esmi_socket_c0_residency_get
Performance (Boost limit) Monitor, 22
esmi_socket_energy_get
Energy Monitor (RAPL MSR), 15
esmi_socket_lclk_dpm_level_set
APB and LCLK level control, 30
esmi_socket_power_cap_get
Power Monitor, 19
esmi_socket_power_cap_max_get
Power Monitor, 20
esmi_socket_power_cap_set
Power Control, 21
esmi_socket_power_get
Power Monitor, 19
esmi_socket_temperature_get
Temperature Query, 27
esmi_status_t
e_smi.h, 39
ESMI_SUCCESS
e_smi.h, 39
esmi_threads_per_core_get
Auxiliary functions, 32
ESMI_UNEXPECTED_SIZE
e_smi.h, 40
ESMI_UNKNOWN_ERROR
e_smi.h, 40
esmi_xgmi_width_set
xGMI bandwidth control, 28

HSMP System Statistics, 16
esmi_cclk_limit_get, 17
esmi_fclk_mclk_get, 17
esmi_hsmp_proto_ver_get, 18
esmi_prochot_status_get, 16
esmi_smu_fw_version_get, 16

Initialization and Shutdown, 13
esmi_init, 13

Performance (Boost limit) Control, 24
esmi_core_boostlimit_set, 24
esmi_package_boostlimit_set, 25
esmi_socket_boostlimit_set, 24
Performance (Boost limit) Monitor, 22
esmi_core_boostlimit_get, 22
esmi_socket_c0_residency_get, 22
Power Control, 21
esmi_socket_power_cap_set, 21
Power Monitor, 19
esmi_socket_power_cap_get, 19
esmi_socket_power_cap_max_get, 20
esmi_socket_power_get, 19

smu_fw_version, 35

Temperature Query, 27
esmi_socket_temperature_get, 27

xGMI bandwidth control, 28
esmi_xgmi_width_set, 28