E-SMI Library: EPYC™ Systems Management Interface Library

Generated by Doxygen 1.8.14

Contents

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.

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.

Building E-SMI

Dowloading the source

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

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

Building the library and tool

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

```
$ mkdir -p build
```

```
$ cd build
```

```
$ cmake <location of root of E-SMI library CMakeLists.txt>
```

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

```
$ cmake -DENABLE_STATIC_LIB=1 <location of root of E-SMI library CMake↩
Lists.txt>
```

```
$ make
```

The built library `libe_smi64.so.X.Y` will appear in the `build` folder.

```
# Install library file and header; default location is /opt/e-sms
```

```
$ sudo make install
```

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.

Building the package

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

```
$ make package
```

Kernel dependencies

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

Monitoring energy counters

The Energy counters reported by the RAPL MSRs, the AMD Energy driver can report per core and per socket counters via the HWMON sys entries. The AMD Energy driver is an out of kernel module hosted `https://github.com/amd/amd_energy`. The kernel config symbol SENSORS_AMD_ENERGY needs to be selected, can be built and inserted as a module.

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 ioctl interface.

- amd_hsmp driver is accepted in upstream kernel under pd/x86
  - Please build the library against uapi header asm/amd_hsmp.h
- PCIe interface needs to be enabled in the BIOS. On the reference BIOS, the CBS option may be found in the following path

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

`BIOS Default: "Auto" (Disabled)`

If the option is disabled, the related E-SMI APIs will return -ETIMEDOUT.

Supported hardware

AMD Zen3 based CPU Family `19h` Models `0h-Fh` and `30h-3Fh`. AMD Zen4 based CPU Family `19h` Models `10h-1Fh`.

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)

Usage Basics

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.

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;
}
```

Usage

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:

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, 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)

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

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.
- 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.

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.

5.3.2.6 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

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.

5.3.2.7 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.

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.

5.3.2.8 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.

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.

5.4 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.

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.

5.4.2.4 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.

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.

5.5 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.

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.

5.5.2.2 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.

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.

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.

5.6 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 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

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`.

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.

5.8 ddr_bandwidth Monitor

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

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 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.

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.

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

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

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.

5.11 xGMI bandwidth control

Functions

- esmi_status_t esmi_xgmi_width_set (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.

5.11.1 Detailed Description

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

5.11.2 Function Documentation

5.11.2.1 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 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.

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.

5.12 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 bandwidth.

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

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_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 max_pstate, uint8_t min_pstate)

  Set data fabric pstate range.

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.

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) - 3 (lowest).

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.
None-zero	is returned upon failure.

5.13.2.3 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.

5.13.2.4 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 lelvel is an encoding to represent PCIe link frequency

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.

5.13.2.5 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. Value are: 0 = auto detect bandwidth utilisation and set link rate 1 = Limit at gen4 rate 2 = Limit at gen5 rate

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.

5.13.2.6 esmi_df_pstate_range_set()

```
esmi_status_t esmi_df_pstate_range_set (
            uint8_t sock_ind,
            uint8_t max_pstate,
            uint8_t min_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) - 3 (lowest) with max <= min.

Parameters

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

Return values

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

5.14 Bandwidth Query

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 (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.

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 (
            struct link_id_bw_type link,
            uint32_t ∗ xgmi_bw )
```

Get xGMI bandwidth.

This function will get the xGMI Aggregate bandwidth for the specified link in a multi socket system.

Parameters

`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 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.

5.15.1 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.

5.15.2 Function Documentation

5.15.2.1 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.

5.15.2.2 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.

5.15.2.3 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.

5.15.2.4 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.

5.15.2.5 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.

5.15.2.6 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.

5.15.2.7 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 link_id_bw_type Struct Reference

LINK ID and Bandwidth type Information.It contains LINK ID Encoding. Valid Link ID encodings are 1(P0), 2(P1), 4(P2), 8(P3), 16(G0), 32(G1), 64(G2), 128(G3). Valid xGMI 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].
- link_id_encoding link_id

  Link ID [1,2,4,8,16,32,64,128].

6.5.1 Detailed Description

LINK ID and Bandwidth type Information.It contains LINK ID Encoding. Valid Link ID encodings are 1(P0), 2(P1), 4(P2), 8(P3), 16(G0), 32(G1), 64(G2), 128(G3). Valid xGMI 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

6.6 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

6.6.1 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

6.7 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)

6.7.1 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>
```

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.
- 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 ID and Bandwidth type Information.It contains LINK ID Encoding. Valid Link ID encodings are 1(P0), 2(P1), 4(P2), 8(P3), 16(G0), 32(G1), 64(G2), 128(G3). Valid xGMI 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.

Macros

- #define ENERGY_DEV_NAME "amd_energy"

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

  HSMP device 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 link_id_encoding {
  P0 = BIT(0), **P1** = BIT(1), **P2** = BIT(2), **P3** = BIT(3),
  G0 = BIT(4), **G1** = BIT(5), **G2** = BIT(6), **G3** = BIT(7) }

    IO LINK and xGMI link Encoding values.
- 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 }

    Error codes retured by E-SMI functions.
- enum hsmp_proto_versions { **HSMP_PROTO_VER2** = 2, **HSMP_PROTO_VER3**, **HSMP_PROTO_VER4**, **HSMP_PROTO_VER5** }

    HSMP protocol version names.

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_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_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_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_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 (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_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_xgmi_width_set (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.
- 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_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 max_pstate, uint8_t min_pstate)

  Set data fabric pstate range.
- 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 (struct link_id_bw_type link, uint32_t ∗xgmi_bw)

  Get xGMI bandwidth.
- 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.

Index