

## Advanced Platform Managment Link (APML) Library

Generated by Doxygen 1.8.14



# Contents

<b>1</b>	<b>Advanced Platform Management Link (APML) Library</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>5</b>
2.1	Modules . . . . .	5
<b>3</b>	<b>Data Structure Index</b>	<b>7</b>
3.1	Data Structures . . . . .	7
<b>4</b>	<b>File Index</b>	<b>11</b>
4.1	File List . . . . .	11
<b>5</b>	<b>Module Documentation</b>	<b>13</b>
5.1	Auxiliary functions . . . . .	13
5.1.1	Detailed Description . . . . .	13
5.1.2	Function Documentation . . . . .	13
5.1.2.1	errno_to_oob_status() . . . . .	13
5.1.2.2	esmi_get_err_msg() . . . . .	14
5.2	SB-RMI Mailbox Service . . . . .	15
5.2.1	Detailed Description . . . . .	15
5.3	Power Monitor . . . . .	16
5.3.1	Detailed Description . . . . .	16
5.3.2	Function Documentation . . . . .	16
5.3.2.1	read_socket_power() . . . . .	16
5.3.2.2	read_socket_power_limit() . . . . .	17
5.3.2.3	read_max_socket_power_limit() . . . . .	17

5.4	Platforms	18
5.4.1	Fam-19h_Mod-00h-0Fh	18
5.4.2	Fam-19h_Mod-10h-1Fh	18
5.4.3	Fam-19h_Mod-90h-9Fh	18
5.4.4	Fam-1Ah_Mod-00h-0Fh	18
5.5	Power Control	19
5.5.1	Detailed Description	19
5.5.2	Function Documentation	19
5.5.2.1	write_socket_power_limit()	19
5.6	Performance (Boost limit) Monitor	20
5.6.1	Detailed Description	20
5.6.2	Function Documentation	20
5.6.2.1	read_esb_boost_limit()	20
5.6.2.2	read_bios_boost_fmax()	20
5.7	Out-of-band Performance (Boost limit) Control	22
5.7.1	Detailed Description	22
5.7.2	Function Documentation	22
5.7.2.1	write_esb_boost_limit()	22
5.7.2.2	write_esb_boost_limit_allcores()	22
5.8	Current, Min, Max TDP	24
5.8.1	Detailed Description	24
5.8.2	Function Documentation	24
5.8.2.1	read_tdp()	24
5.8.2.2	read_max_tdp()	24
5.8.2.3	read_min_tdp()	25
5.9	Prochot	26
5.9.1	Detailed Description	26
5.9.2	Function Documentation	26
5.9.2.1	read_prochot_status()	26
5.9.2.2	read_prochot_residency()	26

5.10	Dram and other features Query	28
5.10.1	Detailed Description	28
5.10.2	Function Documentation	28
5.10.2.1	read_dram_throttle()	28
5.10.2.2	write_dram_throttle()	29
5.10.2.3	read_vddio_mem_power()	29
5.10.2.4	read_nbio_error_logging_register()	30
5.10.2.5	read_iod_bist()	30
5.10.2.6	read_ccd_bist_result()	31
5.10.2.7	read_ccx_bist_result()	31
5.10.2.8	read_cclk_freq_limit()	32
5.10.2.9	read_socket_c0_residency()	32
5.10.2.10	read_ddr_bandwidth()	33
5.11	using CPUID Register Access	34
5.11.1	Detailed Description	34
5.11.2	Function Documentation	34
5.11.2.1	esmi_get_vendor_id()	34
5.11.2.2	esmi_get_processor_info()	35
5.11.2.3	esmi_get_logical_cores_per_socket()	35
5.11.2.4	esmi_get_threads_per_core()	35
5.12	SB_RMI Read Processor Register Access	37
5.12.1	Detailed Description	37
5.12.2	Function Documentation	37
5.12.2.1	esmi_oob_read_msr()	37
5.13	SB-RMI CPUID Register Access	38
5.13.1	Detailed Description	38
5.13.2	Function Documentation	38
5.13.2.1	esmi_oob_cpuid()	38
5.13.2.2	esmi_oob_cpuid_eax()	39
5.13.2.3	esmi_oob_cpuid_ebx()	39

5.13.2.4	esmi_oob_cpuid_ecx()	40
5.13.2.5	esmi_oob_cpuid_edx()	40
5.13.2.6	read_max_threads_per_l3()	41
5.14	SB-RMI Register Read Byte Protocol	42
5.14.1	Detailed Description	43
5.14.2	Function Documentation	43
5.14.2.1	read_sbrmi_alert_status()	43
5.14.2.2	read_sbrmi_alert_mask()	44
5.14.2.3	clear_sbrmi_ras_status()	44
5.14.2.4	esmi_get_threads_per_socket()	45
5.14.3	Variable Documentation	45
5.14.3.1	thread_en_reg_v10	45
5.15	SBTSI Register Read Byte Protocol	47
5.15.1	Detailed Description	49
5.15.2	Function Documentation	49
5.15.2.1	read_sbtsi_cpuinttemp()	49
5.15.2.2	read_sbtsi_status()	50
5.15.2.3	read_sbtsi_config()	50
5.15.2.4	read_sbtsi_updaterate()	51
5.15.2.5	write_sbtsi_updaterate()	51
5.15.2.6	read_sbtsi_hitempint()	51
5.15.2.7	read_sbtsi_lotempint()	52
5.15.2.8	read_sbtsi_configwrite()	52
5.15.2.9	read_sbtsi_cputempdecimal()	53
5.15.2.10	read_sbtsi_cputempoffint()	53
5.15.2.11	read_sbtsi_cputempoffdec()	54
5.15.2.12	read_sbtsi_hitempdecimal()	54
5.15.2.13	read_sbtsi_lotempdecimal()	55
5.15.2.14	read_sbtsi_timeoutconfig()	55
5.15.2.15	read_sbtsi_alertthreshold()	55

5.15.2.16 read_sbtsi_alertconfig()	56
5.15.2.17 read_sbtsi_manufid()	56
5.15.2.18 read_sbtsi_revision()	57
5.15.2.19 sbtsi_get_cputemp()	57
5.15.2.20 sbtsi_get_temp_status()	58
5.15.2.21 sbtsi_get_config()	58
5.15.2.22 sbtsi_set_configwr()	59
5.15.2.23 sbtsi_get_timeout()	59
5.15.2.24 sbtsi_set_timeout_config()	60
5.15.2.25 sbtsi_set_hitemp_threshold()	60
5.15.2.26 sbtsi_set_lotemp_threshold()	61
5.15.2.27 sbtsi_get_hitemp_threshold()	61
5.15.2.28 sbtsi_get_lotemp_threshold()	62
5.15.2.29 read_sbtsi_cputempoffset()	62
5.15.2.30 write_sbtsi_cputempoffset()	63
5.15.2.31 sbtsi_set_alert_threshold()	63
5.15.2.32 sbtsi_set_alert_config()	63
<b>6 Data Structure Documentation</b>	<b>65</b>
6.1 action_payload Struct Reference	65
6.1.1 Detailed Description	65
6.2 apml_encodings Struct Reference	65
6.2.1 Detailed Description	66
6.3 dimm_power Struct Reference	66
6.3.1 Detailed Description	66
6.4 dimm_spd_d_in Struct Reference	66
6.4.1 Detailed Description	67
6.5 dimm_thermal Struct Reference	67
6.5.1 Detailed Description	67
6.6 dpm_level Struct Reference	68
6.6.1 Detailed Description	68

6.7	<a href="#">freq_limits Struct Reference</a>	68
6.7.1	<a href="#">Detailed Description</a>	68
6.8	<a href="#">get_ras_action_data_in Struct Reference</a>	69
6.8.1	<a href="#">Detailed Description</a>	69
6.9	<a href="#">hbm_device_info Struct Reference</a>	69
6.9.1	<a href="#">Detailed Description</a>	69
6.10	<a href="#">host_status Struct Reference</a>	70
6.10.1	<a href="#">Detailed Description</a>	70
6.11	<a href="#">lclk_dpm_level_range Struct Reference</a>	70
6.11.1	<a href="#">Detailed Description</a>	70
6.12	<a href="#">link_id_bw_type Struct Reference</a>	71
6.12.1	<a href="#">Detailed Description</a>	71
6.12.2	<a href="#">Field Documentation</a>	71
6.12.2.1	<a href="#">link_id</a>	71
6.13	<a href="#">max_ddr_bw Struct Reference</a>	71
6.13.1	<a href="#">Detailed Description</a>	72
6.14	<a href="#">max_mem_bw Struct Reference</a>	72
6.14.1	<a href="#">Detailed Description</a>	72
6.15	<a href="#">mca_bank Struct Reference</a>	72
6.15.1	<a href="#">Detailed Description</a>	73
6.16	<a href="#">mclk_fclk_pstates Struct Reference</a>	73
6.16.1	<a href="#">Detailed Description</a>	73
6.17	<a href="#">nbio_err_log Struct Reference</a>	73
6.17.1	<a href="#">Detailed Description</a>	74
6.18	<a href="#">oob_config_d_in Struct Reference</a>	74
6.18.1	<a href="#">Detailed Description</a>	74
6.18.2	<a href="#">Field Documentation</a>	74
6.18.2.1	<a href="#">mca_oob_misc0_ec_enable</a>	75
6.18.2.2	<a href="#">dram_cecc_oob_ec_mode</a>	75
6.18.2.3	<a href="#">dram_cecc_leak_rate</a>	75



6.18.2.4	pcie_err_reporting_en	75
6.18.2.5	core_mca_err_reporting_en	75
6.19	pci_address Struct Reference	75
6.19.1	Detailed Description	76
6.20	processor_info Struct Reference	76
6.20.1	Detailed Description	76
6.21	pstate_freq Struct Reference	77
6.21.1	Detailed Description	77
6.22	ras_action_status Struct Reference	77
6.22.1	Detailed Description	77
6.23	ras_df_err_chk Struct Reference	78
6.23.1	Detailed Description	78
6.24	ras_df_err_dump Union Reference	78
6.24.1	Detailed Description	78
6.24.2	Field Documentation	79
6.24.2.1	input	79
6.25	ras_override_delay Struct Reference	79
6.25.1	Detailed Description	79
6.26	ras_rt_err_req_type Struct Reference	80
6.26.1	Detailed Description	80
6.27	ras_rt_valid_err_inst Struct Reference	80
6.27.1	Detailed Description	81
6.28	run_time_err_d_in Struct Reference	81
6.28.1	Detailed Description	81
6.29	run_time_threshold Struct Reference	81
6.29.1	Detailed Description	82
6.29.2	Field Documentation	82
6.29.2.1	err_type	82
6.30	set_ras_action_data_in Struct Reference	82
6.30.1	Detailed Description	83
6.31	statistics Struct Reference	83
6.31.1	Detailed Description	83
6.32	status_query_payload Struct Reference	83
6.32.1	Detailed Description	83
6.33	svi_port_domain Struct Reference	84
6.33.1	Detailed Description	84
6.34	temp_refresh_rate Struct Reference	84
6.34.1	Detailed Description	84
6.35	xgmi_speed_rate_n_width Struct Reference	85
6.35.1	Detailed Description	85

<b>7 File Documentation</b>	<b>87</b>
7.1 apml_err.h File Reference	87
7.1.1 Detailed Description	88
7.1.2 Enumeration Type Documentation	88
7.1.2.1 oob_status_t	88
7.2 apml_recovery.h File Reference	89
7.2.1 Detailed Description	89
7.2.2 Function Documentation	89
7.2.2.1 apml_recover_dev()	89
7.3 esmi_cpuid_msr.h File Reference	90
7.3.1 Detailed Description	91
7.3.2 Enumeration Type Documentation	91
7.3.2.1 cpuid_reg	91
7.4 esmi_mailbox.h File Reference	91
7.4.1 Detailed Description	97
7.4.2 Function Documentation	97
7.4.2.1 write_bmc_report_dimm_power()	97
7.4.2.2 write_bmc_report_dimm_thermal_sensor()	98
7.4.2.3 read_bmc_ras_pcie_config_access()	98
7.4.2.4 read_bmc_ras_mca_validity_check()	99
7.4.2.5 read_bmc_ras_mca_msr_dump()	99
7.4.2.6 read_bmc_ras_fch_reset_reason()	100
7.4.2.7 read_dimm_temp_range_and_refresh_rate()	100
7.4.2.8 read_dimm_power_consumption()	101
7.4.2.9 read_dimm_thermal_sensor()	101
7.4.2.10 read_pwr_current_active_freq_limit_socket()	102
7.4.2.11 read_pwr_current_active_freq_limit_core()	102
7.4.2.12 read_pwr_svi_telemetry_all_rails()	103
7.4.2.13 read_socket_freq_range()	103
7.4.2.14 read_current_io_bandwidth()	104

7.4.2.15	<a href="#">read_current_xgmi_bandwidth()</a>	104
7.4.2.16	<a href="#">write_gmi3_link_width_range()</a>	105
7.4.2.17	<a href="#">write_xgmi_link_width_range()</a>	105
7.4.2.18	<a href="#">write_apb_disable()</a>	106
7.4.2.19	<a href="#">write_apb_enable()</a>	107
7.4.2.20	<a href="#">read_current_dfpstate_frequency()</a>	107
7.4.2.21	<a href="#">write_lclk_dpm_level_range()</a>	108
7.4.2.22	<a href="#">read_bmc_rapl_units()</a>	108
7.4.2.23	<a href="#">read_bmc_cpu_base_frequency()</a>	109
7.4.2.24	<a href="#">read_bmc_control_pcie_gen5_rate()</a>	109
7.4.2.25	<a href="#">read_rapl_core_energy_counters()</a>	110
7.4.2.26	<a href="#">read_rapl_pkg_energy_counters()</a>	110
7.4.2.27	<a href="#">write_pwr_efficiency_mode()</a>	111
7.4.2.28	<a href="#">write_df_pstate_range()</a>	111
7.4.2.29	<a href="#">read_lclk_dpm_level_range()</a>	112
7.4.2.30	<a href="#">read_ucode_revision()</a>	112
7.4.2.31	<a href="#">reset_on_sync_flood()</a>	113
7.4.2.32	<a href="#">override_delay_reset_on_sync_flood()</a>	113
7.4.2.33	<a href="#">get_post_code()</a>	114
7.4.2.34	<a href="#">get_bmc_ras_run_time_err_validity_ck()</a>	114
7.4.2.35	<a href="#">get_bmc_ras_run_time_error_info()</a>	115
7.4.2.36	<a href="#">set_bmc_ras_err_threshold()</a>	115
7.4.2.37	<a href="#">set_bmc_ras_oob_config()</a>	116
7.4.2.38	<a href="#">get_bmc_ras_oob_config()</a>	116
7.4.2.39	<a href="#">read_ppin_fuse()</a>	117
7.4.2.40	<a href="#">read_ras_df_err_validity_check()</a>	117
7.4.2.41	<a href="#">read_ras_df_err_dump()</a>	118
7.4.2.42	<a href="#">read_rtc()</a>	118
7.4.2.43	<a href="#">read_dimm_spd_register()</a>	119
7.4.2.44	<a href="#">get_dimm_serial_num()</a>	119

7.5	esmi_mailbox_nda.h File Reference	119
7.5.1	Detailed Description	120
7.5.2	Function Documentation	120
7.5.2.1	set_bmc_ras_action_status()	120
7.5.2.2	get_bmc_ras_action_status()	121
7.6	esmi_rmi.h File Reference	121
7.6.1	Detailed Description	123
7.7	esmi_tsi.h File Reference	123
7.7.1	Detailed Description	126
7.8	rmi_mailbox_mi300.h File Reference	126
7.8.1	Detailed Description	129
7.8.2	Function Documentation	129
7.8.2.1	set_gfx_core_clock()	129
7.8.2.2	set_mclk_fclk_max_pstate()	130
7.8.2.3	get_mclk_fclk_pstates()	130
7.8.2.4	set_xgmi_pstate()	131
7.8.2.5	unset_xgmi_pstate()	131
7.8.2.6	get_xgmi_pstates()	132
7.8.2.7	get_xcc_idle_residency()	132
7.8.2.8	get_energy_accum_with_timestamp()	133
7.8.2.9	get_alarms()	133
7.8.2.10	get_psn()	134
7.8.2.11	get_link_info()	134
7.8.2.12	get_max_min_gfx_freq()	135
7.8.2.13	get_act_gfx_freq_cap()	135
7.8.2.14	get_svi_rail_telemetry()	136
7.8.2.15	get_die_hotspot_info()	136
7.8.2.16	get_mem_hotspot_info()	137
7.8.2.17	get_host_status()	137
7.8.2.18	get_max_mem_bw_util()	138

7.8.2.19	<code>get_hbm_throttle()</code>	138
7.8.2.20	<code>set_hbm_throttle()</code>	138
7.8.2.21	<code>get_hbm_temperature()</code>	139
7.8.2.22	<code>get_clk_freq_limits()</code>	139
7.8.2.23	<code>get_sockets_in_system()</code>	141
7.8.2.24	<code>get_bist_results()</code>	141
7.8.2.25	<code>get_statistics()</code>	142
7.8.2.26	<code>clear_statistics()</code>	142
7.8.2.27	<code>get_die_type()</code>	143
7.8.2.28	<code>get_curr_xgmi_pstate()</code>	143
7.8.2.29	<code>get_max_operating_temp()</code>	144
7.8.2.30	<code>get_slow_down_temp()</code>	144
7.8.2.31	<code>get_hbm_dev_info()</code>	145
7.8.2.32	<code>get_pcistats()</code>	145



## Chapter 1

# Advanced Platform Management Link (APML) Library

The Advanced Platform Management Link (APML) Library (formerly known as EPYC™ System Management Interface (E-SMI) Out-of-band Library), is part of the EPYC™ System Management Out-of-band software stack. It is a C library for Linux that provides a user space interface to monitor and control the CPU's Systems Management features.

### Versioning and Backward Compatibility

The APML 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 APML 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 APML Library

#### Additional Required software for building

To build the APML library, the following components are required.

**Note:** The software versions are being used in the development, earlier versions are not guaranteed to work:

- CMake (v3.5.0)
- latex (pdfTeX 3.14159265-2.6-1.40.18)
- apml modules (apml\_sbrmi and apml\_sbtsi)
  - available on [Github](#)

## Downloading the source

The source code for APML library is available on [Github](#).

## Directory structure of the source

Once the APML 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
- `$ tool/` contains `apml_tool` based on the APML library
- `$ include/esmi_oob` contains the header files used by the APML library
- `$ src/esmi_oob` contains the APML source library

**Building the library is achieved by executing the typical CMake build sequence for native build as follows:**

- `$ mkdir -p build`
- `$ mkdir -p install`
- `$ cd build`
- `$ cmake -DCMAKE_INSTALL_PREFIX=${PWD}/install <location of root of APML library CMakeLists.txt>`
- `$ make` The built library will appear in the build folder.

## Cross compile the library for Target systems

Before installing the cross compiler, verify the target architecture:

- `$ uname -m`

Example: To cross compile for ARM32 processor:

- `$ sudo apt-get install gcc-arm-linux-gnueabi`

Example: To cross compile for AARCH64 processor:

- `$ sudo apt-get install gcc-aarch64-linux-gnu`

NOTE: For cross compilation, cross-\$ARCH.cmake file is provided for below Architectures:

- `armhf`
- `aarch64`

Compilation steps:

- `$ mkdir -p build`
- `$ cd build`
- `$ cmake -DCMAKE_TOOLCHAIN_FILE=../cross-[arch..].cmake <location of root of APML library CMakeLists.txt>`
- `$ make` The built library will appear in the build folder. Copy the required binaries and the dynamic linked library to target board(BMC).
- `$ scp libapml64.so.0 root@10.x.x.x:/usr/lib`
- `$ scp apml_tool root@10.x.x.x:/usr/bin`



## Disclaimer

Input arguments passed by the user are not validated. It might result in an unexpected system behavior.

## Building the Documentation

User can build the documentation PDF file using the following steps (continued from the steps above):

- `$ make doc` The reference manual (APML\_Library\_Manual.pdf) and release notes (APML\_Library\_Release\_Notes.pdf) will be generated upon a successful build.

## Usage

### Tool Usage

APML tool is a C program based on the APML Library. An executable "apml\_tool" will be generated in the *build/* folder. This tool provides options to monitor and control System Management functionality.

In execution platform, user can verify if "apml\_sbrmi" and apml\_tsi" modules are loaded. The apml modules are open-sourced at: [https://github.com/amd/apml\\_modules.git](https://github.com/amd/apml_modules.git)

For detailed usage information, use -h or --help flag:

```
bin# ./apml_tool -h

===== APML System Management Interface =====

Usage: ./apml_tool [soc_num] [Option<s>] / [--help] [module-name]
Where: soc_num : socket number 0 or 1
Description:
./apml_tool -v                - Displays tool version
./apml_tool [SOC_NUM] --showdependency - Displays module dependency
./apml_tool --help <MODULE>   - Displays help on the options for the specified module
./apml_tool <option/s>        - Runs the specified option/s.
Usage: ./apml_tool [soc_num] [Option] params

    MODULES:
    1. mailbox
    2. sbrmi
    3. sbtsi
    4. reg-access
    5. cpuid
    6. recovery

===== End of APML SMI =====

$ ./apml_tool -v

===== APML System Management Interface =====

APML lib version : 2.x.y

===== End of APML SMI =====
```

Following is a sample usage to get the individual library functionality APIs over I2C. User can pass arguments using `./apml_tool [socket_num] -p` or `./apml_tool [socket_num] --showpower`

1. \$ ./apml\_tool 0 -p

```
===== APML System Management Interface =====
-----
| Power (Watts)          | 65.029          |
| PowerLimit (Watts)     | 210.000         |
| PowerLimitMax (Watts)  | 400.000         |
-----
===== End of APML SMI =====
```

2. bin# ./apml\_tool 1 --setpowerlimit 200000

```
===== APML System Management Interface =====

Set power_limit :          200.000 Watts successfully

===== End of APML SMI =====
```

3. \$ ./apml\_tool 0 --showtsiregisters

```
*** SB-TSI REGISTER SUMMARY ***
-----
FUNCTION/Reg Name      | Reg offset | Hexa(0x) | Value [Units]
-----
_CPUTEMP               |            |           | 33.125 °C
  CPU_INT              | 0x1        | 0x21     | 33 °C
  CPU_DEC              | 0x10       | 0x1      | 0.125 °C
_STATUS               | 0x2        |           |
  CPU Temp Alert       |            |           | CPU No Temp Alert
_CONFIG               | 0x3        |           |
  ALERT_L pin          |            |           | Disabled
  Runstop              |            |           | Comparison Enabled
  Atomic Rd order      |            |           | Integer latches Decimal
  ARA response         |            |           | Enabled
  ARA response         |            |           | Enabled
_TSI_UPDATERATE        | 0x4        |           | 16.000 Hz
_HIGH_THRESHOLD_TEMP   |            |           | 255.125 °C
  HIGH_INT             | 0x7        | 0xff     | 255 °C
  HIGH_DEC             | 0x13       | 0x1      | 0.125 °C
_LOW_THRESHOLD_TEMP    |            |           | 0.000 °C
  LOW_INT              | 0x8        | 0x0      | 0 °C
  LOW_DEC              | 0x14       | 0x0      | 0.000 °C
_TEMP_OFFSET          |            |           | 0.000 °C
  OFF_INT              | 0x11       | 0x0      | 0 °C
  OFF_DEC              | 0x12       | 0x0      | 0.000 °C
_TIMEOUT_CONFIG        | 0x22       |           | Enabled
_THRESHOLD_SAMPLE      | 0x32       |           |
  CPU Alert TH         |            |           | 1
_TSI_ALERT_CONFIG      | 0xbf       |           |
  CPU Alert CFG        |            |           | Enabled
_TSI_MANUFACTURE_ID    | 0xfe       |           | 0
_TSI_REVISION          | 0xff       |           | 0x4
-----
```

## Chapter 2

# Module Index

### 2.1 Modules

Here is a list of all modules:

Auxiliary functions . . . . .	13
SB-RMI Mailbox Service . . . . .	15
Power Monitor . . . . .	16
Platforms . . . . .	18
Power Control . . . . .	19
Performance (Boost limit) Monitor . . . . .	20
Out-of-band Performance (Boost limit) Control . . . . .	22
Current, Min, Max TDP . . . . .	24
Prochot . . . . .	26
Dram and other features Query . . . . .	28
using CPUID Register Access . . . . .	34
SB_RMI Read Processor Register Access . . . . .	37
SB-RMI CPUID Register Access . . . . .	38
SB-RMI Register Read Byte Protocol . . . . .	42
SBTSI Register Read Byte Protocol . . . . .	47



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">action_payload</a>	Data for the action payload. Members contain payload, offset, repair entry number . . . . .	65
<a href="#">apml_encodings</a>	APML link ID encodings for legacy platforms and MI300A. Structure contains the following members. Link ID encoding value and its name. MI300A APML Link ID Encoding: 00000011b: P2 00000100b: P3 00001000b: G0 00001001b: G1 00001010b: G2 00001011b: G3 00001100b↔ : G4 00001101b: G5 00001110b: G6 00001111b: G7 For other platforms the APML Link ID Encoding: 00000001b: P0 00000010b: P1 00000100b: P2 00001000b: P3 00010000b: G0 00100000b: G1 01000000b: G2 10000000b: G3 . . . . .	65
<a href="#">dimm_power</a>	DIMM power(mW), update rate(ms) and dimm address . . . . .	66
<a href="#">dimm_spd_d_in</a>	Dimm_spd_d_in: structure contains following members to be passed as input, to get 4 byte DIMM spd register data . . . . .	66
<a href="#">dimm_thermal</a>	DIMM thermal sensor (degree C), update rate and dimm address . . . . .	67
<a href="#">dpm_level</a>	Max and min LCK DPM level on a given NBIO ID. Valid Max and min DPM level values are 0 - 1 . . . . .	68
<a href="#">freq_limits</a>	Struct containing max frequency and min frequency limit . . . . .	68
<a href="#">get_ras_action_data_in</a>	Data input for the get BMC RAS action status. Members contain repair entry number and ras action id . . . . .	69
<a href="#">hbm_device_info</a>	Struct containing device vendor, part number and total memory size in GBs . . . . .	69
<a href="#">host_status</a>	Struct containing power management controlling status, driving running status . . . . .	70
<a href="#">lclk_dpm_level_range</a>	Max and Min Link frequency clock (LCLK) DPM level on a socket. 8 bit NBIO ID, <a href="#">dpm_level</a> struct containing 8 bit max DPM level, 8 bit min DPM level . . . . .	70
<a href="#">link_id_bw_type</a>	APML LINK ID and Bandwidth type Information. It contains APML LINK ID Encoding. Non-M↔ I300 Platforms Valid Link ID encodings are 1(P0), 2(P1), 4(P2), 8(P3), 16(G0), 32(G1), 64(G2), 128(G3). Valid APML MI300 APML LINK ID Encoding. Valid Link ID encodings are 3(P2), 4(P3), 8(G0), 9(G1), 10(G2), 11(G3), 12(G4), 13(G5), 14(G6), 15(G7). IO Bandwidth types 1(Aggregate_BW), 2 (Read BW), 4 (Write BW) . . . . .	71

<a href="#">max_ddr_bw</a>	Structure for Max DDR bandwidth and utilization. It contains max bandwidth(12 bit data) in Gbps, current utilization bandwidth(12 bit data) in Gbps, current utilized bandwidth( 8 bit data) in percentage . . . . .	71
<a href="#">max_mem_bw</a>	Structure for Max DDR/HBM bandwidth and utilization. It contains max bandwidth(16 bit data) in Gbps, current utilization bandwidth(Read+Write)(16 bit data) in Gbps . . . . .	72
<a href="#">mca_bank</a>	MCA bank information.It contains 16 bit Index for MCA Bank and 16 bit offset . . . . .	72
<a href="#">mclk_fclk_pstates</a>	Struct containing memory clock and fabric clock pstate mappings . . . . .	73
<a href="#">nbio_err_log</a>	NBIO quadrant(8 bit data) and NBIO register offset(24 bit) data . . . . .	73
<a href="#">oob_config_d_in</a>	Configure oob state infrastructure in SoC. structure consists of mca_oob_misc0_ec_enable, dram_cecc_oob_ec_mode, dram_cecc_leak_rate, pcie_err_reporting_en and core_mca_err_reporting_en . . . . .	74
<a href="#">pci_address</a>	PCI address information .PCI address includes 4 bit segment, 12 bit aligned offset, 8 bit bus, 5 bit device info and 3 bit function . . . . .	75
<a href="#">processor_info</a>	Read Proccessor Info . . . . .	76
<a href="#">pstate_freq</a>	DF P-state frequency.It includes mem clock(16 bit data) frequency (DRAM memory clock), data fabric clock (12 bit data), UMC clock divider (UMC) (1 bit data) . . . . .	77
<a href="#">ras_action_status</a>	Output data for the get BMC RAS action status. Members contain repair result, repair entry number and ras action id . . . . .	77
<a href="#">ras_df_err_chk</a>	RAS df err validity check output status. Structure contains the following members. df_block_instances number of block instance with error log to report (0 - 256) err_log_len length of error log in bytes per instance (0 - 256) . . . . .	78
<a href="#">ras_df_err_dump</a>	RAS df error dump input. Union contains the following members. input[0] 4 byte aligned offset in error log ( 0 - 255) input[1] DF block ID (0 - 255) input[2] Zero based index of DF block instance (0 - 255) input[3] Reserved data_in 32-bit data input . . . . .	78
<a href="#">ras_override_delay</a>	BMC RAS override delay reset CPU on sync flood. The structure contains delay value override in mins [5 -120 mins], disable delay counter and stop delay counter. If disable delay counter is set ResetCpuOnSyncFlood response will NOT be delayed in the next SyncFlood regardless of the value specified in delay_val_override. If StopDelayCounter is set it stops the active delay countdown which extends the DelayResetCpuOnSyncFlood indefinitely and system will not reset . . . . .	79
<a href="#">ras_rt_err_req_type</a>	Get the Error type and request type. Supported Runtime error type[1:0]: . . . . .	80
<a href="#">ras_rt_valid_err_inst</a>	Number of valid error instance per category. It consists of number of bytes per each category and number of instances of each category . . . . .	80
<a href="#">run_time_err_d_in</a>	Get run time error information data_in. Runtime error data_in includes 4 byte aligned offset in instance, runtime error category and zero based index of valid instance number of bytes per each category and number of instances of each category . . . . .	81
<a href="#">run_time_threshold</a>	Configure threshold counters for MCA, DRAM CECC and PCIE. structure consists of error type, error count threshold and max interrupt rate . . . . .	81
<a href="#">set_ras_action_data_in</a>	Data input for set ras action status. Members contain struct <a href="#">action_payload</a> , ras action ID and eom flag . . . . .	82

<a href="#">statistics</a>	Struct containing statistics parameter of interest and output control pstate mappings . . . . .	83
<a href="#">status_query_payload</a>	Data for status query payload. Members contain repair entry number . . . . .	83
<a href="#">svi_port_domain</a>	Struct containing port and slave address . . . . .	84
<a href="#">temp_refresh_rate</a>	DIMM temperature range and refresh rate, temperature update flag . . . . .	84
<a href="#">xgmi_speed_rate_n_width</a>	Struct containing xgmi speed rate in MHZ and link width in units of Gpbs. If link_width[0] = 1 then XGMI link X2 is supported. If link_width[1] = 1 then XGMI link X4 is supported. If Link_width[2] = 1 then XGMI link X4 is supported and similarly if link_width[3] = 1 then XGMI link X8 is supported	85





## Chapter 4

# File Index

### 4.1 File List

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

<b>apml_common.h</b>	??
<a href="#">apml_err.h</a>	87
<a href="#">apml_recovery.h</a>	89
<a href="#">esmi_cpuid_msr.h</a>	90
<a href="#">esmi_mailbox.h</a>	91
<a href="#">esmi_mailbox_nda.h</a>	119
<a href="#">esmi_rmi.h</a>	121
<a href="#">esmi_tsi.h</a>	123
<a href="#">rmi_mailbox_mi300.h</a>	126



# Chapter 5

## Module Documentation

### 5.1 Auxiliary functions

#### Functions

- `oob_status_t errno_to_oob_status (int err)`  
*convert linux error to esmi error.*
- `char * esmi_get_err_msg (oob_status_t oob_err)`  
*Get the error string message for esmi oob errors.*

#### 5.1.1 Detailed Description

Below functions provide interfaces to get the total number of cores, sockets and threads per core in the system.

#### 5.1.2 Function Documentation

##### 5.1.2.1 `errno_to_oob_status()`

```
oob_status_t errno_to_oob_status (  
    int err )
```

convert linux error to esmi error.

Get the appropriate esmi error for linux error.

#### Parameters

in	<i>err</i>	a linux error number
----	------------	----------------------

**Return values**

<i>oob_↔ status_t</i>	is returned upon particular esmi error
---------------------------	--

**5.1.2.2 esmi\_get\_err\_msg()**

```
char* esmi_get_err_msg (
    oob_status_t oob_err )
```

Get the error string message for esmi oob errors.

Get the error message for the esmi oob error numbers

**Parameters**

in	<i>oob_err</i>	is a esmi oob error number
----	----------------	----------------------------

**Return values**

<i>char*</i>	value returned upon successful call.
--------------	--------------------------------------

## 5.2 SB-RMI Mailbox Service

### Modules

- [Power Monitor](#)
- [Power Control](#)
- [Performance \(Boost limit\) Monitor](#)
- [Out-of-band Performance \(Boost limit\) Control](#)
- [Current, Min, Max TDP](#)
- [Prochot](#)
- [Dram and other features Query](#)

### 5.2.1 Detailed Description

write, 'write and read' operations for a given socket.

## 5.3 Power Monitor

### Modules

- [Platforms](#)

### Functions

- [oob\\_status\\_t read\\_socket\\_power](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the power consumption of the socket.*
- [oob\\_status\\_t read\\_socket\\_power\\_limit](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the current power cap/limit value for a given socket.*
- [oob\\_status\\_t read\\_max\\_socket\\_power\\_limit](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the maximum value that can be assigned as a power cap/limit for a given socket.*

#### 5.3.1 Detailed Description

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

#### 5.3.2 Function Documentation

##### 5.3.2.1 read\_socket\_power()

```
oob_status_t read_socket_power (
    uint8_t soc_num,
    uint32_t * buffer )
```

Get the power consumption of the socket.

Given socket number and a pointer to a uint32\_t buffer, this function will get the current power consumption (in watts) to the uint32\_t pointed to by buffer. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to uint32_t value of power consumption

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 5.3.2.2 read\_socket\_power\_limit()

```
oob_status_t read_socket_power_limit (
    uint8_t soc_num,
    uint32_t * buffer )
```

Get the current power cap/limit value for a given socket.

This function will return the valid power cap `buffer` for a given socket, this value will be used for the system to limit the power. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to a <code>uint32_t</code> that indicates the valid possible power cap/limit, in watts

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 5.3.2.3 read\_max\_socket\_power\_limit()

```
oob_status_t read_max_socket_power_limit (
    uint8_t soc_num,
    uint32_t * buffer )
```

Get the maximum value that can be assigned as a power cap/limit for a given socket.

This function will return the maximum possible valid power cap/limit Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
out	<i>buffer</i>	a pointer to a <code>uint32_t</code> that indicates the maximum possible power cap/limit, in watts

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.4 Platforms

Supporting Platforms: [Fam-19h\\_Mod-00h-0Fh](#) [Fam-19h\\_Mod-10h-1Fh](#) [Fam-19h\\_Mod-90h-9Fh](#) [Fam-1Ah\\_Mod-00h-0Fh](#)

### 5.4.1 Fam-19h\_Mod-00h-0Fh

Platform with Family 19h and model in range 00h - 0Fh

### 5.4.2 Fam-19h\_Mod-10h-1Fh

Platform with Family 19h and model in range 10h - 1Fh

### 5.4.3 Fam-19h\_Mod-90h-9Fh

Platform with Family 19h and model in range 90h - 9Fh

### 5.4.4 Fam-1Ah\_Mod-00h-0Fh

Platform with Family 1Ah and model in range 00h - 0Fh



## 5.5 Power Control

### Functions

- [oob\\_status\\_t write\\_socket\\_power\\_limit](#) (uint8\_t soc\_num, uint32\_t limit)  
*Set the power cap/limit 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 write\_socket\_power\_limit()

```
oob\_status\_t write_socket_power_limit (
    uint8_t soc_num,
    uint32_t limit )
```

Set the power cap/limit value for a given socket.

This function will set the power cap/limit Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>limit</i>	uint32_t that indicates the desired power cap/limit, in milliwatts

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.6 Performance (Boost limit) Monitor

### Functions

- [oob\\_status\\_t read\\_esb\\_boost\\_limit](#) (uint8\_t soc\_num, uint32\_t value, uint32\_t \*buffer)  
*Get the Out-of-band boostlimit value for a given core.*
- [oob\\_status\\_t read\\_bios\\_boost\\_fmax](#) (uint8\_t soc\_num, uint32\_t value, uint32\_t \*buffer)  
*Get the In-band maximum boostlimit value for a given core.*

### 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 read\_esb\_boost\_limit()

```
oob_status_t read_esb_boost_limit (
    uint8_t soc_num,
    uint32_t value,
    uint32_t * buffer )
```

Get the Out-of-band boostlimit value for a given core.

This function will return the core's current Out-of-band boost limit `buffer` for a particular `value` Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>value</i>	a cpu index
in, out	<i>buffer</i>	pointer to a uint32_t that indicates the possible boost limit value

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 5.6.2.2 read\_bios\_boost\_fmax()

```
oob_status_t read_bios_boost_fmax (
    uint8_t soc_num,
```

```
uint32_t value,  
uint32_t * buffer )
```

Get the In-band maximum boostlimit value for a given core.

This function will return the core's current maximum In-band boost limit `buffer` for a particular `value` is `cpu_ind` Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>value</i>	is a cpu index
in, out	<i>buffer</i>	a pointer to a <code>uint32_t</code> that indicates the maximum boost limit value set via In-band

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.7 Out-of-band Performance (Boost limit) Control

### Functions

- [oob\\_status\\_t write\\_esb\\_boost\\_limit](#) (uint8\_t soc\_num, uint32\_t cpu\_ind, uint32\_t limit)  
*Set the Out-of-band boostlimit value for a given core.*
- [oob\\_status\\_t write\\_esb\\_boost\\_limit\\_allcores](#) (uint8\_t soc\_num, uint32\_t limit)  
*Set the boostlimit value for the whole socket (whole system).*

### 5.7.1 Detailed Description

Below functions provide ways to control the Out-of-band Boost limit values.

### 5.7.2 Function Documentation

#### 5.7.2.1 write\_esb\_boost\_limit()

```
oob_status_t write_esb_boost_limit (
    uint8_t soc_num,
    uint32_t cpu_ind,
    uint32_t limit )
```

Set the Out-of-band boostlimit value for a given core.

This function will set the boostlimit to the provided value `limit` for a given cpu. NOTE: Currently the limit is setting for all the cores instead of a particular cpu. Testing in Progress. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>cpu_ind</i>	a cpu index is a given core to set the boostlimit
in	<i>limit</i>	a uint32_t that indicates the desired Out-of-band boostlimit value of a given core

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 5.7.2.2 write\_esb\_boost\_limit\_allcores()

```
oob_status_t write_esb_boost_limit_allcores (
```

```
uint8_t soc_num,  
uint32_t limit )
```

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

This function will set the boostlimit to the provided value `boostlimit` for the socket. Supported platforms↵  
: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>limit</i>	a uint32_t that indicates the desired boostlimit value of the socket

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.8 Current, Min, Max TDP

### Functions

- [oob\\_status\\_t read\\_tdp](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the Thermal Design Power limit TDP of the socket with provided socket index.*
- [oob\\_status\\_t read\\_max\\_tdp](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the Maximum Thermal Design Power limit TDP of the socket with provided socket index.*
- [oob\\_status\\_t read\\_min\\_tdp](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the Minimum Thermal Design Power limit TDP of the socket.*

### 5.8.1 Detailed Description

Below functions provide interfaces to get the current, Min and Max TDP, Prochot and Prochot Residency for a given socket.

### 5.8.2 Function Documentation

#### 5.8.2.1 read\_tdp()

```
oob_status_t read_tdp (
    uint8_t soc_num,
    uint32_t * buffer )
```

Get the Thermal Design Power limit TDP of the socket with provided socket index.

Given a socket and a pointer to a uint32\_t buffer, this function will get the current TDP (in milliwatts) Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to uint32_t to which the Current TDP value will be copied

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 5.8.2.2 read\_max\_tdp()

```
oob_status_t read_max_tdp (
```

```
uint8_t soc_num,
uint32_t * buffer )
```

Get the Maximum Thermal Design Power limit TDP of the socket with provided socket index.

Given a socket and a pointer, this function will get the Maximum TDP (watts) Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to uint32_t to which the Maximum TDP value will be copied

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 5.8.2.3 read\_min\_tdp()

```
oob_status_t read_min_tdp (
    uint8_t soc_num,
    uint32_t * buffer )
```

Get the Minimum Thermal Design Power limit TDP of the socket.

Given a socket and a pointer to a uint32\_t, this function will get the Minimum TDP (watts) Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to uint32_t to which the Minimum TDP value will be copied

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.9 Prochot

### Functions

- [oob\\_status\\_t read\\_prochot\\_status](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the Prochot Status of the socket with provided socket index.*
- [oob\\_status\\_t read\\_prochot\\_residency](#) (uint8\_t soc\_num, float \*buffer)  
*Get the Prochot Residency (since the boot time or last read of Prochot Residency) of the socket.*

### 5.9.1 Detailed Description

Below functions provide interfaces to get Prochot and Prochot Residency for a given socket.

### 5.9.2 Function Documentation

#### 5.9.2.1 read\_prochot\_status()

```
oob_status_t read_prochot_status (
    uint8_t soc_num,
    uint32_t * buffer )
```

Get the Prochot Status of the socket with provided socket index.

Given a socket and a pointer to a uint32\_t, this function will get the Prochot status as active/1 or inactive/0 Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to uint32_t to which the Prochot status will be copied

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 5.9.2.2 read\_prochot\_residency()

```
oob_status_t read_prochot_residency (
    uint8_t soc_num,
    float * buffer )
```



Get the Prochot Residency (since the boot time or last read of Prochot Residency) of the socket.

Given a socket and a pointer to a `uint32_t`, this function will get the Prochot residency as a percentage Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to float to which the Prochot residency will be copied

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.10 Dram and other features Query

### Functions

- [oob\\_status\\_t read\\_dram\\_throttle](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Read Dram Throttle will always read the highest percentage value.*
- [oob\\_status\\_t write\\_dram\\_throttle](#) (uint8\_t soc\_num, uint32\_t limit)  
*Set Dram Throttle value in terms of percentage.*
- [oob\\_status\\_t read\\_vddio\\_mem\\_power](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Read VDDIOMem Power returns the estimated VDDIOMem power consumed within the socket.*
- [oob\\_status\\_t read\\_nbio\\_error\\_logging\\_register](#) (uint8\_t soc\_num, struct [nbio\\_err\\_log](#) nbio, uint32\_t \*buffer)  
*Read NBIO Error Logging Register.*
- [oob\\_status\\_t read\\_iod\\_bist](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Read IOD Bist status.*
- [oob\\_status\\_t read\\_ccd\\_bist\\_result](#) (uint8\_t soc\_num, uint32\_t input, uint32\_t \*buffer)  
*Read CCD Bist status. Results are read for each CCD present in the system.*
- [oob\\_status\\_t read\\_ccx\\_bist\\_result](#) (uint8\_t soc\_num, uint32\_t value, uint32\_t \*ccx\_bist)  
*Read CPU Core Complex Bist result. results are read for each Logical CCX instance number and returns a value which is the concatenation of L3 pass status and all cores in the complex(n:0).*
- [oob\\_status\\_t read\\_cclk\\_freq\\_limit](#) (uint8\_t soc\_num, uint32\_t \*cclk\_freq)  
*Read CCLK frequency limit for the given socket.*
- [oob\\_status\\_t read\\_socket\\_c0\\_residency](#) (uint8\_t soc\_num, uint32\_t \*c0\_res)  
*Read socket C0 residency.*
- [oob\\_status\\_t read\\_ddr\\_bandwidth](#) (uint8\_t soc\_num, struct [max\\_ddr\\_bw](#) \*max\_ddr)  
*Get the Theoretical maximum DDR Bandwidth of the system in GB/s, Current utilized DDR Bandwidth (Read + Write) in GB/s and Current utilized DDR Bandwidth as a percentage of theoretical maximum. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#).*

### 5.10.1 Detailed Description

### 5.10.2 Function Documentation

#### 5.10.2.1 read\_dram\_throttle()

```
oob_status_t read_dram_throttle (
    uint8_t soc_num,
    uint32_t * buffer )
```

Read Dram Throttle will always read the highest percentage value.

This function will always read the highest percentage value as represented by PROCHOT throttle or write dram throttle. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
out	<i>buffer</i>	is to read the dram throttle in % (0 - 100).

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.10.2.2 write\_dram\_throttle()

```
oob_status_t write_dram_throttle (
    uint8_t soc_num,
    uint32_t limit )
```

Set Dram Throttle value in terms of percentage.

This function will set the dram throttle of the provided value limit for the given socket. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>limit</i>	that indicates the desired limit as per SSP PPR write can be between 0 to 80% to for a given socket

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.10.2.3 read\_vddio\_mem\_power()

```
oob_status_t read_vddio_mem_power (
    uint8_t soc_num,
    uint32_t * buffer )
```

Read VDDIOMem Power returns the estimated VDDIOMem power consumed within the socket.

This function will read VDDIOMem Power for the given socket Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>buffer</i>	is to read VDDIOMem Power.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
------------------------------------	-----------------------------------

## Return values

<i>Non-zero</i>	is returned upon failure.
-----------------	---------------------------

## 5.10.2.4 read\_nbio\_error\_logging\_register()

```
oob_status_t read_nbio_error_logging_register (
    uint8_t soc_num,
    struct nbio_err_log nbio,
    uint32_t * buffer )
```

Read NBIO Error Logging Register.

Given a socket, quadrant and register offset as input, this function will read NBIOErrorLogging Register. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>nbio</i>	<a href="#">nbio_err_log</a> Struct containing nbio quadrant and offset.
out	<i>buffer</i>	is to read NBIOErrorLoggingRegister(register value).

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.10.2.5 read\_iod\_bist()

```
oob_status_t read_iod_bist (
    uint8_t soc_num,
    uint32_t * buffer )
```

Read IOD Bist status.

This function will read IOD Bist result for the given socket. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>buffer</i>	is to read IOdBistResult 0 = Bist pass, 1 = Bist fail

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.10.2.6 read\_ccd\_bist\_result()

```
oob_status_t read_ccd_bist_result (
    uint8_t soc_num,
    uint32_t input,
    uint32_t * buffer )
```

Read CCD Bist status. Results are read for each CCD present in the system.

Given a socket bus number and address, Logical CCD instance number as *input*, this function will read CC↔DBistResult. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>input</i>	is a Logical CCD instance number.
out	<i>buffer</i>	is to read CCDBistResult 0 = Bist pass, 1 = Bist fail

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.10.2.7 read\_ccx\_bist\_result()

```
oob_status_t read_ccx_bist_result (
    uint8_t soc_num,
    uint32_t value,
    uint32_t * ccx_bist )
```

Read CPU Core Complex Bist result. results are read for each Logical CCX instance number and returns a value which is the concatenation of L3 pass status and all cores in the complex(n:0).

Given a socket bus number, address, Logical CCX instance number as *input*, this function will read CCX↔BistResult. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
----	----------------	---------------

## Parameters

in	<i>value</i>	is a Logical CCX instance number. Valid values [0, (k -1)] where k is the number of logical CCX instances.
out	<i>ccx_bist</i>	result is concatenation of bist results for all cores[31:16] in the complex(n:0) L3 bist[15:0], where n num of cores in CCX.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.10.2.8 read\_cclk\_freq\_limit()

```
oob_status_t read_cclk_freq_limit (
    uint8_t soc_num,
    uint32_t * cclk_freq )
```

Read CCLK frequency limit for the given socket.

This function will read CPU core clock frequency limit for the given socket. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>cclk_freq</i>	CPU core clock frequency limit [MHz]

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.10.2.9 read\_socket\_c0\_residency()

```
oob_status_t read_socket_c0_residency (
    uint8_t soc_num,
    uint32_t * c0_res )
```

Read socket C0 residency.

This function will provides average C0 residency across all cores in the socket. 100% specifies that all enabled cores in the socket are running in C0. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>c0_res</i>	is to read Socket C0 residency[%].

## Return values

<i>OOB_SUCCESS</i>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.10.2.10 read\_ddr\_bandwidth()

```
oob_status_t read_ddr_bandwidth (
    uint8_t soc_num,
    struct max_ddr_bw * max_ddr )
```

Get the Theoretical maximum DDR Bandwidth of the system in GB/s, Current utilized DDR Bandwidth (Read + Write) in GB/s and Current utilized DDR Bandwidth as a percentage of theoretical maximum. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>max_ddr</i>	<a href="#">max_ddr_bw</a> struct containing max bandwidth, utilized bandwidth and utilized bandwidth percentage.

## Return values

<i>OOB_SUCCESS</i>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.11 using CPUID Register Access

### Functions

- [oob\\_status\\_t esmi\\_get\\_vendor\\_id](#) (uint8\_t soc\_num, char \*vendor\_id)  
*Get the number of logical cores per socket.*
- [oob\\_status\\_t esmi\\_get\\_processor\\_info](#) (uint8\_t soc\_num, struct [processor\\_info](#) \*proc\_info)  
*Get the number of logical cores per socket.*
- [oob\\_status\\_t esmi\\_get\\_logical\\_cores\\_per\\_socket](#) (uint8\_t soc\_num, uint32\_t \*logical\_cores\_per\_socket)  
*Get the number of logical cores per socket.*
- [oob\\_status\\_t esmi\\_get\\_threads\\_per\\_core](#) (uint8\_t soc\_num, uint32\_t \*threads\_per\_core)  
*Get number of threads per core.*

### 5.11.1 Detailed Description

Below function provide interface to read the processor info using CPUID register. output from commmand will be written into the buffer.

### 5.11.2 Function Documentation

#### 5.11.2.1 esmi\_get\_vendor\_id()

```
oob_status_t esmi_get_vendor_id (
    uint8_t soc_num,
    char * vendor_id )
```

Get the number of logical cores per socket.

Get the processor vendor Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
out	<i>vendor↵_id</i>	to get the processor vendor, 12 byte RO value

#### Return values

<i>uint32↵_t</i>	is returned upon successful call.
------------------	-----------------------------------



## 5.11.2.2 esmi\_get\_processor\_info()

```
oob_status_t esmi_get_processor_info (
    uint8_t soc_num,
    struct processor_info * proc_info )
```

Get the number of logical cores per socket.

Get the effective family, model and step\_id of the processor. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>proc_info</i>	to get family, model & stepping identifier

## Return values

<i>uint32_t</i>	is returned upon successful call.
-----------------	-----------------------------------

## 5.11.2.3 esmi\_get\_logical\_cores\_per\_socket()

```
oob_status_t esmi_get_logical_cores_per_socket (
    uint8_t soc_num,
    uint32_t * logical_cores_per_socket )
```

Get the number of logical cores per socket.

Get the total number of logical cores in a socket. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>logical_cores_per_socket</i>	is returned

## Return values

<i>logical_cores_per_socket</i>	is returned upon successful call.
---------------------------------	-----------------------------------

## 5.11.2.4 esmi\_get\_threads\_per\_core()

```
oob_status_t esmi_get_threads_per_core (
    uint8_t soc_num,
    uint32_t * threads_per_core )
```

Get number of threads per core.

Get the number of threads per core. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>threads_per_core</i>	is returned

#### Return values

<i>threads_per_core</i>	is returned upon successful call.
-------------------------	-----------------------------------

## 5.12 SB\_RMI Read Processor Register Access

### Functions

- `oob_status_t esmi_oob_read_msr` (`uint8_t soc_num`, `uint32_t thread`, `uint32_t msraddr`, `uint64_t *buffer`)  
*Read the MCA MSR register for a given thread.*

### 5.12.1 Detailed Description

Below function provide interface to read the SB-RMI MCA MSR register. output from MCA MSR command will be written into the buffer.

### 5.12.2 Function Documentation

#### 5.12.2.1 `esmi_oob_read_msr()`

```
oob_status_t esmi_oob_read_msr (
    uint8_t soc_num,
    uint32_t thread,
    uint32_t msraddr,
    uint64_t * buffer )
```

Read the MCA MSR register for a given thread.

Given a `thread` and SB-RMI register command, this function reads msr value.

#### Parameters

in	<code>soc_num</code>	Socket index.
in	<code>thread</code>	is a particular thread in the system.
in	<code>msraddr</code>	MCA MSR register to read
out	<code>buffer</code>	is to hold the return output of msr value.

#### Return values

<code>OOB_SUCCESS</code>	is returned upon successful call.
<code>None-zero</code>	is returned upon failure.

## 5.13 SB-RMI CPUID Register Access

### Functions

- `oob_status_t esmi_oob_cpuid` (uint8\_t soc\_num, uint32\_t thread, uint32\_t \*eax, uint32\_t \*ebx, uint32\_t \*ecx, uint32\_t \*edx)  
*Read CPUID functionality for a particular thread in a system.*
- `oob_status_t esmi_oob_cpuid_eax` (uint8\_t soc\_num, uint32\_t thread, uint32\_t fn\_eax, uint32\_t fn\_ecx, uint32\_t \*eax)  
*Read eax register on CPUID functionality.*
- `oob_status_t esmi_oob_cpuid_ebx` (uint8\_t soc\_num, uint32\_t thread, uint32\_t fn\_eax, uint32\_t fn\_ecx, uint32\_t \*ebx)  
*Read ebx register on CPUID functionality.*
- `oob_status_t esmi_oob_cpuid_ecx` (uint8\_t soc\_num, uint32\_t thread, uint32\_t fn\_eax, uint32\_t fn\_ecx, uint32\_t \*ecx)  
*Read ecx register on CPUID functionality.*
- `oob_status_t esmi_oob_cpuid_edx` (uint8\_t soc\_num, uint32\_t thread, uint32\_t fn\_eax, uint32\_t fn\_ecx, uint32\_t \*edx)  
*Read edx register on CPUID functionality.*
- `oob_status_t read_max_threads_per_l3` (uint8\_t soc\_num, uint32\_t \*threads\_l3)  
*Read max threads per L3 cache.*

### 5.13.1 Detailed Description

Below function provide interface to get the CPUID access via the SBRMI.

Output from CPUID command will be written into registers eax, ebx, ecx and edx.

### 5.13.2 Function Documentation

#### 5.13.2.1 esmi\_oob\_cpuid()

```
oob_status_t esmi_oob_cpuid (
    uint8_t soc_num,
    uint32_t thread,
    uint32_t * eax,
    uint32_t * ebx,
    uint32_t * ecx,
    uint32_t * edx )
```

Read CPUID functionality for a particular thread in a system.

Given a `thread`, `eax` as function input and `ecx` as extended function input. this function will get the cpuid details for a particular thread in a pointer to `eax`, `ebx`, `ecx`, `edx`

#### Parameters

in	<code>soc_num</code>	Socket index.
in	<code>thread</code>	is a particular thread in the system.
in, out	<code>eax</code>	a pointer uint32_t to get eax value
out	<code>ebx</code>	a pointer uint32_t to get ebx value
in, out	<code>ecx</code>	a pointer uint32_t to get ecx value

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>None-zero</i>	is returned upon failure.

## 5.13.2.2 esmi\_oob\_cpuid\_eax()

```
oob_status_t esmi_oob_cpuid_eax (
    uint8_t soc_num,
    uint32_t thread,
    uint32_t fn_eax,
    uint32_t fn_ecx,
    uint32_t * eax )
```

Read eax register on CPUID functionality.

Given a `thread`, `fn_eax` as function and `fn_ecx` as extended function input, this function will get the cpuid details for a particular thread at `eax`.

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>thread</i>	is a particular thread in the system.
in	<i>fn_eax</i>	cpuid function
in	<i>fn_ecx</i>	cpuid extended function
out	<i>eax</i>	is to read eax from cpuid functionality.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>None-zero</i>	is returned upon failure.

## 5.13.2.3 esmi\_oob\_cpuid\_ebx()

```
oob_status_t esmi_oob_cpuid_ebx (
    uint8_t soc_num,
    uint32_t thread,
    uint32_t fn_eax,
    uint32_t fn_ecx,
    uint32_t * ebx )
```

Read ebx register on CPUID functionality.

Given a `thread`, `fn_eax` as function and `fn_ecx` as extended function input, this function will get the cpuid details for a particular thread at `ebx`.

**Parameters**

in	<i>soc_num</i>	Socket index.
in	<i>thread</i>	is a particular thread in the system.
in	<i>fn_eax</i>	cpuid function
in	<i>fn_ecx</i>	cpuid extended function
out	<i>ebx</i>	is to read ebx from cpuid functionality.

**Return values**

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>None-zero</i>	is returned upon failure.

**5.13.2.4 esmi\_oob\_cpuid\_ecx()**

```
oob_status_t esmi_oob_cpuid_ecx (
    uint8_t soc_num,
    uint32_t thread,
    uint32_t fn_eax,
    uint32_t fn_ecx,
    uint32_t * ecx )
```

Read ecx register on CPUID functionality.

Given a *thread*, *fn\_eax* as function and *fn\_ecx* as extended function input, this function will get the cpuid details for a particular thread at *ecx*.

**Parameters**

in	<i>soc_num</i>	Socket index.
in	<i>thread</i>	is a particular thread in the system.
in	<i>fn_eax</i>	cpuid function
in	<i>fn_ecx</i>	cpuid extended function
out	<i>ecx</i>	is to read ecx from cpuid functionality.

**Return values**

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>None-zero</i>	is returned upon failure.

**5.13.2.5 esmi\_oob\_cpuid\_edx()**

```
oob_status_t esmi_oob_cpuid_edx (
    uint8_t soc_num,
```

```
uint32_t thread,
uint32_t fn_eax,
uint32_t fn_ecx,
uint32_t * edx )
```

Read edx register on CPUID functionality.

Given a `thread`, `fn_eax` as function and `fn_ecx` as extended function input, this function will get the cpuid details for a particular thread at `edx`.

#### Parameters

in	<code>soc_num</code>	Socket index.
in	<code>thread</code>	is a particular thread in the system.
in	<code>fn_eax</code>	cpuid function
in	<code>fn_ecx</code>	cpuid extended function
out	<code>edx</code>	is to read edx from cpuid functionality.

#### Return values

<a href="#"><code>OOB_SUCCESS</code></a>	is returned upon successful call.
<code>None-zero</code>	is returned upon failure.

#### 5.13.2.6 read\_max\_threads\_per\_l3()

```
oob_status_t read_max_threads_per_l3 (
    uint8_t soc_num,
    uint32_t * threads_l3 )
```

Read max threads per L3 cache.

Reads max threads per L3 cache.

#### Parameters

in	<code>soc_num</code>	Socket index.
out	<code>threads_l3</code>	threads per L3 cache.

#### Return values

<a href="#"><code>OOB_SUCCESS</code></a>	is returned upon successful call.
<code>None-zero</code>	is returned upon failure.

## 5.14 SB-RMI Register Read Byte Protocol

### Functions

- [oob\\_status\\_t read\\_sbrmi\\_revision](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
Read revision from SB\_RMI register command. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_control](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
Read Control byte from SB\_RMI register command. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_status](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
Read one byte of Status value from SB\_RMI register command. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_readsize](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
This register specifies the number of bytes to return when using the block read protocol to read SBRMI←\_x[4F:10]. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_threadenablestatus](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
Read one byte of Thread Status from SB\_RMI register command. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_multithreadenablestatus](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
Read one byte of Thread Status from SB\_RMI register command. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_swinterrupt](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
This register is used by the SMBus master to generate an interrupt to the processor to indicate that a message is available.. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_threadnumber](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
This register indicates the maximum number of threads present. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_mp0\\_msg](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
This register will read the message running on the MP0. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_alert\\_status](#) (uint8\_t soc\_num, uint8\_t num\_of\_alert\_status\_reg, uint8\_t \*\*buffer)  
This function will read bit vector for all the threads. Value of 1 indicates MCE occurred for the thread and is set by hardware. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_alert\\_mask](#) (uint8\_t soc\_num, uint8\_t num\_of\_alert\_mask\_reg, uint8\_t \*\*buffer)  
This function will read bit vector for all the threads. Value of 1 indicates alert signaling disabled for corresponding SBRMI::AlertStatus[MceStat] for the thread. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_inbound\\_msg](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
This register will read the inbound message. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_outbound\\_msg](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
This register will read the outbound message. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_threadnumberlow](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
This register indicates the low part of maximum number of threads. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_threadnumberghi](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
This register indicates the upper part of maximum number of threads. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_thread\\_cs](#) (uint8\_t soc\_num, uint8\_t \*buffer)



This register is used to read the thread cs. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

- `oob_status_t read_sbrmi_ras_status` (uint8\_t soc\_num, uint8\_t \*buffer)

This register will read the ras status. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

- `oob_status_t clear_sbrmi_ras_status` (uint8\_t soc\_num, uint8\_t buffer)

This API will clear ras status register. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

- `oob_status_t esmi_get_threads_per_socket` (uint8\_t soc\_num, uint32\_t \*threads\_per\_socket)

Get the number of threads per socket.

## Variables

- `const uint8_t thread_en_reg_v10` [MAX\_THREAD\_REG\_V10]  
thread enable register revision 0x10
- `const uint8_t thread_en_reg_v20` [MAX\_THREAD\_REG\_V20]  
thread enable register revision 0x20
- `const uint8_t alert_status` [MAX\_ALERT\_REG]  
alert status register
- `const uint8_t alert_mask` [MAX\_ALERT\_REG]  
alert mask

### 5.14.1 Detailed Description

The SB-RMI registers can be read or written from the SMBus interface using the SMBus defined PEC-optional Read Byte and Write Byte protocols with the SB-RMI register number in the command byte.

### 5.14.2 Function Documentation

#### 5.14.2.1 read\_sbrmi\_alert\_status()

```
oob_status_t read_sbrmi_alert_status (
    uint8_t soc_num,
    uint8_t num_of_alert_status_reg,
    uint8_t ** buffer )
```

This function will read bit vector for all the threads. Value of 1 indicates MCE occurred for the thread and is set by hardware. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<code>soc_num</code>	Socket index.
in	<code>num_of_alert_status_reg</code>	number of alert status registers.
in, out	<code>buffer</code>	a pointer to read all "num_of_alert_status_reg" of alert status registers. Buffer length should be equal to "num_of_alert_status_reg" value.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>None-zero</i>	is returned upon failure.

## 5.14.2.2 read\_sbrmi\_alert\_mask()

```
oob_status_t read_sbrmi_alert_mask (
    uint8_t soc_num,
    uint8_t num_of_alert_mask_reg,
    uint8_t ** buffer )
```

This function will read bit vector for all the threads. Value of 1 indicates alert signaling disabled for corresponding S↔BRMI::AlertStatus[MceStat] for the thread. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>num_of_alert_mask_reg</i>	number of alert mask registers.
in, out	<i>buffer</i>	a pointer to read all "num_of_alert_mask_reg" of alert mask registers. Buffer length should be equal to "num_of_alert_mask_reg" value.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>None-zero</i>	is returned upon failure.

## 5.14.2.3 clear\_sbrmi\_ras\_status()

```
oob_status_t clear_sbrmi_ras_status (
    uint8_t soc_num,
    uint8_t buffer )
```

This API will clear ras status register. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>buffer</i>	bit mask to clear ras status bits

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
------------------------------------	-----------------------------------

## Return values

<i>Non-zero</i>	is returned upon failure.
-----------------	---------------------------

## 5.14.2.4 esmi\_get\_threads\_per\_socket()

```
oob_status_t esmi_get_threads_per_socket (
    uint8_t soc_num,
    uint32_t * threads_per_socket )
```

Get the number of threads per socket.

Get the total number of threads in a socket.

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>threads_per_socket</i>	is returned

## Return values

<i>threads_per_socket</i>	is returned upon successful call.
---------------------------	-----------------------------------

## 5.14.3 Variable Documentation

## 5.14.3.1 thread\_en\_reg\_v10

```
const uint8_t thread_en_reg_v10
```

thread enable register revision 0x10

Read one byte from a given SB\_RMI register number provided socket index and buffer to get the read data for a particular SB-RMI command register.

Given a socket index *socket\_ind* and a pointer to hold the output at *uint8\_t buffer*, this function will get the value from a particular command of SB\_RMI register.

## Parameters

in	<i>soc_num</i>	Socket uindex.
in, out	<i>buffer</i>	a pointer to a <i>uint8_t</i> that indicates value to hold

## Return values

<i>OOB_SUCCESS</i>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure. This value specifies the APML specification revision that the product is compliant to. 0x10 = 1.0x Revision. thread enable register revision 0x10

## 5.15 SBTSI Register Read Byte Protocol

### Functions

- **`oob_status_t read_sbtsi_cpuinttemp`** (uint8\_t soc\_num, uint8\_t \*buffer)  
*integer CPU temperature value The CPU temperature is calculated by adding the CPU temperature offset(SBTSI::CpuTempOffInt, SBTSI::CpuTempOffDec) to the processor control temperature (Tctl). SBTSI::CpuTempInt and SBTSI::CpuTempDec combine to return the CPU temperature. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- **`oob_status_t read_sbtsi_status`** (uint8\_t soc\_num, uint8\_t \*buffer)  
*Status register is Read-only, volatile field If SBTSI::AlertConfig[AlertCompEn] == 0 , the temperature alert is latched high until the alert is read. If SBTSI::AlertConfig[AlertCompEn] == 1, the alert is cleared when the temperature does not meet the threshold conditions for temperature and number of samples. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- **`oob_status_t read_sbtsi_config`** (uint8\_t soc\_num, uint8\_t \*buffer)  
*The bits in this register are Read-only and can be written by Writing to the corresponding bits in SBTSI::ConfigWr Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- **`oob_status_t read_sbtsi_updaterate`** (uint8\_t soc\_num, float \*buffer)  
*This register value specifies the rate at which CPU temperature is compared against the temperature thresholds to determine if an alert event has occurred. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- **`oob_status_t write_sbtsi_updaterate`** (uint8\_t soc\_num, float uprate)  
*This register value specifies the rate at which CPU temperature is compared against the temperature thresholds to determine if an alert event has occurred. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- **`oob_status_t read_sbtsi_hitempint`** (uint8\_t soc\_num, uint8\_t \*buffer)  
*This value specifies the integer portion of the high temperature threshold. The high temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is greater than or equal to the threshold. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- **`oob_status_t read_sbtsi_lotempint`** (uint8\_t soc\_num, uint8\_t \*buffer)  
*This value specifies the integer portion of the low temperature threshold. The low temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is less than or equal to the threshold Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- **`oob_status_t read_sbtsi_configwrite`** (uint8\_t soc\_num, uint8\_t \*buffer)  
*This register provides write access to SBTSI::Config Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- **`oob_status_t read_sbtsi_cputempdecimal`** (uint8\_t soc\_num, float \*buffer)  
*The value returns the decimal portion of the CPU temperature Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- **`oob_status_t read_sbtsi_cputempoffint`** (uint8\_t soc\_num, uint8\_t \*temp\_int)  
*SBTSI::CpuTempOffInt and SBTSI::CpuTempOffDec combine to specify the CPU temperature offset Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- **`oob_status_t read_sbtsi_cputempoffdec`** (uint8\_t soc\_num, float \*temp\_dec)  
*This value specifies the decimal/fractional portion of the CPU temperature offset added to Tctl to calculate the CPU temperature. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- **`oob_status_t read_sbtsi_hitempdecimal`** (uint8\_t soc\_num, float \*temp\_dec)  
*This value specifies the decimal portion of the high temperature threshold. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- **`oob_status_t read_sbtsi_lotempdecimal`** (uint8\_t soc\_num, float \*temp\_dec)  
*value specifies the decimal portion of the low temperature threshold. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*

- [oob\\_status\\_t read\\_sbtsi\\_timeoutconfig](#) (uint8\_t soc\_num, uint8\_t \*timeout)  
value specifies 0=SMBus defined timeout support disabled. 1=SMBus defined timeout support enabled. SMBus timeout enable. If SB-RMI is in use, SMBus timeouts should be enabled or disabled in a consistent manner on both interfaces. SMBus defined timeouts are not disabled for SB-RMI when this bit is set to 0. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_alertthreshold](#) (uint8\_t soc\_num, uint8\_t \*samples)  
Specifies the number of consecutive CPU temperature samples for which a temperature alert condition needs to remain valid before the corresponding alert bit is set. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_alertconfig](#) (uint8\_t soc\_num, uint8\_t \*mode)  
Status register is Read-only, volatile field If SBTSL::AlertConfig[AlertCompEn] == 0 , the temperature alert is latched high until the alert is read. If SBTSL::AlertConfig[AlertCompEn] == 1, the alert is cleared when the temperature does not meet the threshold conditions for temperature and number of samples. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_manufid](#) (uint8\_t soc\_num, uint8\_t \*man\_id)  
Returns the AMD manufacture ID Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_revision](#) (uint8\_t soc\_num, uint8\_t \*revision)  
Specifies the SBI temperature sensor interface revision Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t sbtsi\\_get\\_cputemp](#) (uint8\_t soc\_num, float \*cpu\_temp)  
CPU temperature value The CPU temperature is calculated by adding SBTSL::CpuTempInt and SBTSL::CpuTempDec combine to return the CPU temperature. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t sbtsi\\_get\\_temp\\_status](#) (uint8\_t soc\_num, uint8\_t \*loalert, uint8\_t \*hialert)  
Status register is Read-only, volatile field If SBTSL::AlertConfig[AlertCompEn] == 0 , the temperature alert is latched high until the alert is read. If SBTSL::AlertConfig[AlertCompEn] == 1, the alert is cleared when the temperature does not meet the threshold conditions for temperature and number of samples. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t sbtsi\\_get\\_config](#) (uint8\_t soc\_num, uint8\_t \*al\_mask, uint8\_t \*run\_stop, uint8\_t \*read\_ord, uint8\_t \*ara)  
The bits in this register are Read-only and can be written by Writing to the corresponding bits in SBTSL::ConfigWr Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t sbtsi\\_set\\_configwr](#) (uint8\_t soc\_num, uint8\_t mode, uint8\_t config\_mask)  
The bits in this register are defined sbtsi\_config\_write and can be written by writing to the corresponding bits in SBTSL::ConfigWr Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t sbtsi\\_get\\_timeout](#) (uint8\_t soc\_num, uint8\_t \*timeout\_en)  
To verify if timeout support enabled or disabled Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t sbtsi\\_set\\_timeout\\_config](#) (uint8\_t soc\_num, uint8\_t mode)  
To enable/disable timeout support Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t sbtsi\\_set\\_hitemp\\_threshold](#) (uint8\_t soc\_num, float hitemp\_thr)  
This value set the high temperature threshold. The high temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is greater than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t sbtsi\\_set\\_lotemp\\_threshold](#) (uint8\_t soc\_num, float lotemp\_thr)  
This value set the low temperature threshold. The low temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is less than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t sbtsi\\_get\\_hitemp\\_threshold](#) (uint8\_t soc\_num, float \*hitemp\_thr)  
This value specifies the high temperature threshold. The high temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is greater than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

- `oob_status_t sbtsi_get_lotemp_threshold` (uint8\_t soc\_num, float \*lotemp\_thr)  
This value specifies the low temperature threshold. The low temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is less than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- `oob_status_t read_sbtsi_cputempoffset` (uint8\_t soc\_num, float \*temp\_offset)  
SBTSI::CpuTempOffInt and SBTSI::CpuTempOffDec combine to specify the CPU temperature offset Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- `oob_status_t write_sbtsi_cputempoffset` (uint8\_t soc\_num, float temp\_offset)  
SBTSI::CpuTempOffInt and SBTSI::CpuTempOffDec combine to set the CPU temperature offset Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- `oob_status_t sbtsi_set_alert_threshold` (uint8\_t soc\_num, uint8\_t samples)  
Specifies the number of consecutive CPU temperature samples for which a temperature alert condition needs to remain valid before the corresponding alert bit is set. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- `oob_status_t sbtsi_set_alert_config` (uint8\_t soc\_num, uint8\_t mode)  
Alert comparator mode enable Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

### 5.15.1 Detailed Description

Below functions provide interface to read one byte from the SB-TSI register and output is from a given SB\_TSI register command.

### 5.15.2 Function Documentation

#### 5.15.2.1 read\_sbtsi\_cpuinttemp()

```
oob_status_t read_sbtsi_cpuinttemp (
    uint8_t soc_num,
    uint8_t * buffer )
```

integer CPU temperature value The CPU temperature is calculated by adding the CPU temperature offset(SBTSI::CpuTempOffInt, SBTSI::CpuTempOffDec) to the processor control temperature (Tctl). SBTSI::CpuTempInt and SBTSI::CpuTempDec combine to return the CPU temperature. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

This field returns the integer portion of the CPU temperature

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to hold the cpu temperature

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 5.15.2.2 read\_sbtsi\_status()

```
oob_status_t read_sbtsi_status (
    uint8_t soc_num,
    uint8_t * buffer )
```

Status register is Read-only, volatile field. If SBTSI::AlertConfig[AlertCompEn] == 0, the temperature alert is latched high until the alert is read. If SBTSI::AlertConfig[AlertCompEn] == 1, the alert is cleared when the temperature does not meet the threshold conditions for temperature and number of samples. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to hold the cpu temperature

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 5.15.2.3 read\_sbtsi\_config()

```
oob_status_t read_sbtsi_config (
    uint8_t soc_num,
    uint8_t * buffer )
```

The bits in this register are Read-only and can be written by Writing to the corresponding bits in SBTSI←::ConfigWr. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to hold the cpu temperature

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.



## 5.15.2.4 read\_sbtsi\_updaterate()

```
oob_status_t read_sbtsi_updaterate (
    uint8_t soc_num,
    float * buffer )
```

This register value specifies the rate at which CPU temperature is compared against the temperature thresholds to determine if an alert event has occurred. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to hold the cpu temperature

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.5 write\_sbtsi\_updaterate()

```
oob_status_t write_sbtsi_updaterate (
    uint8_t soc_num,
    float uprate )
```

This register value specifies the rate at which CPU temperature is compared against the temperature thresholds to determine if an alert event has occurred. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>uprate</i>	value to write in raw format

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.6 read\_sbtsi\_hitempint()

```
oob_status_t read_sbtsi_hitempint (
    uint8_t soc_num,
    uint8_t * buffer )
```

This value specifies the integer portion of the high temperature threshold. The high temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is greater than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to hold the integer part of high cpu temp

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 5.15.2.7 read\_sbtsi\_lotempint()

```
oob_status_t read_sbtsi_lotempint (
    uint8_t soc_num,
    uint8_t * buffer )
```

This value specifies the integer portion of the low temperature threshold. The low temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is less than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to hold the integer part of low cpu temp

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 5.15.2.8 read\_sbtsi\_configwrite()

```
oob_status_t read_sbtsi_configwrite (
    uint8_t soc_num,
    uint8_t * buffer )
```

This register provides write access to SBTSI::Config Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to hold the configuraion

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.9 read\_sbtsi\_cputempdecimal()

```
oob_status_t read_sbtsi_cputempdecimal (
    uint8_t soc_num,
    float * buffer )
```

The value returns the decimal portion of the CPU temperature Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>buffer</i>	a pointer to hold the cpu temperature decimal

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.10 read\_sbtsi\_cputempoffint()

```
oob_status_t read_sbtsi_cputempoffint (
    uint8_t soc_num,
    uint8_t * temp_int )
```

SBTSI::CpuTempOffInt and SBTSI::CpuTempOffDec combine to specify the CPU temperature offset Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>temp_int</i>	a pointer to hold the cpu offset interger

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.11 read\_sbtsi\_cputempoffdec()

```
oob_status_t read_sbtsi_cputempoffdec (
    uint8_t soc_num,
    float * temp_dec )
```

This value specifies the decimal/fractional portion of the CPU temperature offset added to Tctl to calculate the CPU temperature. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>temp_dec</i>	a pointer to hold the cpu offset decimal

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.12 read\_sbtsi\_hitempdecimal()

```
oob_status_t read_sbtsi_hitempdecimal (
    uint8_t soc_num,
    float * temp_dec )
```

This value specifies the decimal portion of the high temperature threshold. Supported platforms↔: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>temp_dec</i>	a pointer to hold the decimal part of cpu high temp

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 5.15.2.13 read\_sbtsi\_lotempdecimal()

```
oob_status_t read_sbtsi_lotempdecimal (
    uint8_t soc_num,
    float * temp_dec )
```

value specifies the decimal portion of the low temperature threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>temp_dec</i>	a pointer to hold the decimal part of cpu low temperature

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 5.15.2.14 read\_sbtsi\_timeoutconfig()

```
oob_status_t read_sbtsi_timeoutconfig (
    uint8_t soc_num,
    uint8_t * timeout )
```

value specifies 0=SMBus defined timeout support disabled. 1=SMBus defined timeout support enabled. SMBus timeout enable. If SB-RMI is in use, SMBus timeouts should be enabled or disabled in a consistent manner on both interfaces. SMBus defined timeouts are not disabled for SB-RMI when this bit is set to 0. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>timeout</i>	a pointer to hold the cpu timeout configuration

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 5.15.2.15 read\_sbtsi\_alertthreshold()

```
oob_status_t read_sbtsi_alertthreshold (
```

```
uint8_t soc_num,
uint8_t * samples )
```

Specifies the number of consecutive CPU temperature samples for which a temperature alert condition needs to remain valid before the corresponding alert bit is set. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>samples</i>	a pointer to hold the cpu temperature alert threshold

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 5.15.2.16 read\_sbtsi\_alertconfig()

```
oob_status_t read_sbtsi_alertconfig (
    uint8_t soc_num,
    uint8_t * mode )
```

Status register is Read-only, volatile field If SBTSI::AlertConfig[AlertCompEn] == 0 , the temperature alert is latched high until the alert is read. If SBTSI::AlertConfig[AlertCompEn] == 1, the alert is cleared when the temperature does not meet the threshold conditions for temperature and number of samples. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>mode</i>	a pointer to hold the cpu temperature alert configuration

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 5.15.2.17 read\_sbtsi\_manufid()

```
oob_status_t read_sbtsi_manufid (
    uint8_t soc_num,
    uint8_t * man_id )
```

Returns the AMD manufacture ID Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>man_id</i>	a pointer to hold the manufacture id

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.18 read\_sbtsi\_revision()

```
oob_status_t read_sbtsi_revision (
    uint8_t soc_num,
    uint8_t * revision )
```

Specifies the SBI temperature sensor interface revision Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>revision</i>	a pointer to hold the cpu temperature revision

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.19 sbtsi\_get\_cputemp()

```
oob_status_t sbtsi_get_cputemp (
    uint8_t soc_num,
    float * cpu_temp )
```

CPU temperature value The CPU temperature is calculated by adding SBTSI::CpuTempInt and SBTSI::CpuTempDec combine to return the CPU temperature. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>cpu_temp</i>	a pointer to get temperature of the CPU

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.20 sbtsi\_get\_temp\_status()

```
oob_status_t sbtsi_get_temp_status (
    uint8_t soc_num,
    uint8_t * loalert,
    uint8_t * hialert )
```

Status register is Read-only, volatile field. If SBTSL::AlertConfig[AlertCompEn] == 0, the temperature alert is latched high until the alert is read. If SBTSL::AlertConfig[AlertCompEn] == 1, the alert is cleared when the temperature does not meet the threshold conditions for temperature and number of samples. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>loalert</i>	1=> CPU temp is less than or equal to low temperature threshold for consecutive samples
in, out	<i>hialert</i>	1=> CPU temp is greater than or equal to high temperature threshold for consecutive samples

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.21 sbtsi\_get\_config()

```
oob_status_t sbtsi_get_config (
    uint8_t soc_num,
    uint8_t * al_mask,
    uint8_t * run_stop,
    uint8_t * read_ord,
    uint8_t * ara )
```

The bits in this register are Read-only and can be written by Writing to the corresponding bits in SBTSL::ConfigWr. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
----	----------------	---------------



## Parameters

in, out	<i>al_mask</i>	0=> ALERT_L pin enabled. 1=> ALERT_L pin disabled and does not assert.
in, out	<i>run_stop</i>	0=> Updates to CpuTempInt and CpuTempDec and alert comparisons are enabled. 1=> Updates are disabled and alert comparisons are disabled.
in, out	<i>read_ord</i>	0=> Reading CpuTempInt causes the satate of CpuTempDec to be latched. 1=> Reading CpuTempInt causes the satate of CpuTempDec to be latched.
in, out	<i>ara</i>	1=> ARA response disabled.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.22 sbtsi\_set\_configwr()

```
oob_status_t sbtsi_set_configwr (
    uint8_t soc_num,
    uint8_t mode,
    uint8_t config_mask )
```

The bits in this register are defined sbtsi\_config\_write and can be written by writing to the corresponding bits in SBTSI::ConfigWr Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

NOTE: Currently testing is not done for this API.

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>mode</i>	value to update 0 or 1
in	<i>config_mask</i>	which bit need to update

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.23 sbtsi\_get\_timeout()

```
oob_status_t sbtsi_get_timeout (
    uint8_t soc_num,
    uint8_t * timeout_en )
```

To verify if timeout support enabled or disabled Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>timeout_en</i>	0=>SMBus defined timeout support disabled.

1=SMBus defined timeout support enabled. SMBus timeout enable. If SB-RMI is in use, SMBus timeouts should be enabled or disabled in a consistent manner on both interfaces. SMBus defined timeouts are not disabled for SB-RMI when this bit is set to 0.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.24 sbtsi\_set\_timeout\_config()

```
oob_status_t sbtsi_set_timeout_config (
    uint8_t soc_num,
    uint8_t mode )
```

To enable/disable timeout support Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>mode</i>	0=>SMBus defined timeout support disabled.

1=>SMBus defined timeout support enabled. SMBus timeout enable. If SB-RMI is in use, SMBus timeouts should be enabled or disabled in a consistent manner on both interfaces. SMBus defined timeouts are not disabled for SB-RMI when this bit is set to 0.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.25 sbtsi\_set\_hitemp\_threshold()

```
oob_status_t sbtsi_set_hitemp_threshold (
    uint8_t soc_num,
    float hitemp_thr )
```

This value set the high temperature threshold. The high temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is greater than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>hitemp_thr</i>	Specifies the high temperature threshold

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.26 sbtsi\_set\_lotemp\_threshold()

```
oob_status_t sbtsi_set_lotemp_threshold (
    uint8_t soc_num,
    float lotemp_thr )
```

This value set the low temperature threshold. The low temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is less than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>lotemp_thr</i>	Specifies the low temperature threshold

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.27 sbtsi\_get\_hitemp\_threshold()

```
oob_status_t sbtsi_get_hitemp_threshold (
    uint8_t soc_num,
    float * hitemp_thr )
```

This value specifies the high temperature threshold. The high temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is greater than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>hitemp_thr</i>	Specifies the high temperature threshold

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.28 sbtsi\_get\_lotemp\_threshold()

```
oob_status_t sbtsi_get_lotemp_threshold (
    uint8_t soc_num,
    float * lotemp_thr )
```

This value specifies the low temperature threshold. The low temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is less than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>lotemp_thr</i>	Get the low temperature threshold

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.29 read\_sbtsi\_cputempoffset()

```
oob_status_t read_sbtsi_cputempoffset (
    uint8_t soc_num,
    float * temp_offset )
```

SBTSI::CpuTempOffInt and SBTSI::CpuTempOffDec combine to specify the CPU temperature offset Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in, out	<i>temp_offset</i>	to get the offset value for temperature

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.30 write\_sbtsi\_cputempoffset()

```
oob_status_t write_sbtsi_cputempoffset (
    uint8_t soc_num,
    float temp_offset )
```

SBTSI::CpuTempOffInt and SBTSI::CpuTempOffDec combine to set the CPU temperature offset Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>temp_offset</i>	to set the offset value for temperature

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.31 sbtsi\_set\_alert\_threshold()

```
oob_status_t sbtsi_set_alert_threshold (
    uint8_t soc_num,
    uint8_t samples )
```

Specifies the number of consecutive CPU temperature samples for which a temperature alert condition needs to remain valid before the corresponding alert bit is set. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>samples</i>	Number of samples 0h: 1 sample 6h-1h: (value + 1) sample 7h: 8 sample

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 5.15.2.32 sbtsi\_set\_alert\_config()

```
oob_status_t sbtsi_set_alert_config (
```

```
uint8_t soc_num,  
uint8_t mode )
```

Alert comparator mode enable Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>mode</i>	0=> SBTSI::Status[TempHighAlert] & SBTSI::Status[TempLowAlert] are read-clear. 1=> SBTSI::Status[TempHighAlert] & SBTSI::Status[TempLowAlert] are read-only. ARA response disabled.

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## Chapter 6

# Data Structure Documentation

### 6.1 action\_payload Struct Reference

data for the action payload. Members contain payload, offset, repair entry number.

```
#include <esmi_mailbox_nda.h>
```

#### Data Fields

- uint16\_t [pay\\_load](#)  
*2 byte data payload*
- uint8\_t [offset](#): 5  
*2 byte offset*
- uint8\_t [repair\\_entry\\_num](#): 3  
*repair entry number*

#### 6.1.1 Detailed Description

data for the action payload. Members contain payload, offset, repair entry number.

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

- [esmi\\_mailbox\\_nda.h](#)

### 6.2 apml\_encodings Struct Reference

APML link ID encodings for legacy platforms and MI300A. Structure contains the following members. Link ID encoding value and its name. MI300A APML Link ID Encoding: 00000011b: P2 00000100b: P3 00001000b: G0 00001001b: G1 00001010b: G2 00001011b: G3 00001100b: G4 00001101b: G5 00001110b: G6 00001111b: G7 For other platforms the APML Link ID Encoding: 00000001b: P0 00000010b: P1 00000100b: P2 00001000b: P3 00010000b: G0 00100000b: G1 01000000b: G2 10000000b: G3.

```
#include <apml_common.h>
```

## Data Fields

- `uint8_t val`  
*Link ID encoding value.*
- `char name [3]`  
*Link ID encoding name.*

### 6.2.1 Detailed Description

APML link ID encodings for legacy platforms and MI300A. Structure contains the following members. Link ID encoding value and its name. MI300A APML Link ID Encoding: 00000011b: P2 00000100b: P3 00001000b: G0 00001001b: G1 00001010b: G2 00001011b: G3 00001100b: G4 00001101b: G5 00001110b: G6 00001111b: G7 For other platforms the APML Link ID Encoding: 00000001b: P0 00000010b: P1 00000100b: P2 00001000b: P3 00010000b: G0 00100000b: G1 01000000b: G2 10000000b: G3.

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

- `apml_common.h`

## 6.3 dimm\_power Struct Reference

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

```
#include <esmi_mailbox.h>
```

## Data Fields

- `uint16_t power: 15`  
*Dimm power consumption.*
- `uint16_t update_rate: 9`  
*update rate in ms*
- `uint8_t dimm_addr`  
*Dimm address.*

### 6.3.1 Detailed Description

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

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

- `esmi_mailbox.h`

## 6.4 dimm\_spd\_d\_in Struct Reference

`dimm_spd_d_in`: strucutre contains following memebbers to be passed as input, to get 4 byte DIMM spd register data

```
#include <esmi_mailbox.h>
```



## Data Fields

- `uint8_t dimm_addr`  
*DIMM address.*
- `uint8_t lid: 4`  
*local identifier of device*
- `uint16_t reg_offset: 11`  
*Register offset in given register space.*
- `uint8_t reg_space: 1`  
*Register space, Volatile:0, NVM:1.*
- `uint8_t rsvd`  
*reserved*

### 6.4.1 Detailed Description

`dimm_spd_d_in`: structre contains following memebbers to be passed as input, to get 4 byte DIMM spd register data

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

- [esmi\\_mailbox.h](#)

## 6.5 dimm\_thermal Struct Reference

DIMM thermal sensor (degree C), update rate and dimm address.

```
#include <esmi_mailbox.h>
```

## Data Fields

- `uint16_t sensor: 11`  
*Dimm thermal sensor.*
- `uint16_t update_rate: 9`  
*update rate in ms*
- `uint8_t dimm_addr`  
*Dimm address.*

### 6.5.1 Detailed Description

DIMM thermal sensor (degree C), update rate and dimm address.

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

- [esmi\\_mailbox.h](#)

## 6.6 dpm\_level Struct Reference

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

```
#include <esmi_mailbox.h>
```

### Data Fields

- `uint8_t max_dpm_level`  
*Max LCLK DPM level [0 - 1].*
- `uint8_t min_dpm_level`  
*Min LCLK DPM level [0 - 1].*

### 6.6.1 Detailed Description

Max and min LCK 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:

- [esmi\\_mailbox.h](#)

## 6.7 freq\_limits Struct Reference

struct containing max frequency and min frequency limit

```
#include <rmi_mailbox_mi300.h>
```

### Data Fields

- `uint16_t max`  
*Max clock frequency.*
- `uint16_t min`  
*Min clock frequency.*

### 6.7.1 Detailed Description

struct containing max frequency and min frequency limit

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

- [rmi\\_mailbox\\_mi300.h](#)

## 6.8 get\_ras\_action\_data\_in Struct Reference

data input for the get BMC RAS action status. Members contain repair entry number and ras action id.

```
#include <esmi_mailbox_nda.h>
```

### Data Fields

- struct [status\\_query\\_payload pay\\_load](#)  
*It contains repair entry number.*
- uint8\_t [ras\\_action\\_id](#): 4  
*RAS action id.*

#### 6.8.1 Detailed Description

data input for the get BMC RAS action status. Members contain repair entry number and ras action id.

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

- [esmi\\_mailbox\\_nda.h](#)

## 6.9 hbm\_device\_info Struct Reference

struct containing device vendor, part number and total memory size in GBs.

```
#include <rmi_mailbox_mi300.h>
```

### Data Fields

- uint8\_t [dev\\_vendor](#)  
*device vendor*
- uint8\_t [part\\_num](#)  
*part number*
- uint16\_t [total\\_mem](#)  
*total memory size(GBs)*

#### 6.9.1 Detailed Description

struct containing device vendor, part number and total memory size in GBs.

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

- [rmi\\_mailbox\\_mi300.h](#)

## 6.10 host\_status Struct Reference

struct containing power management controlling status, driving running status.

```
#include <rmi_mailbox_mi300.h>
```

### Data Fields

- `uint8_t controller_status`: 1  
*power mangagement controller status*
- `uint8_t driver_status`: 1  
*driver running status*

### 6.10.1 Detailed Description

struct containing power management controlling status, driving running status.

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

- [rmi\\_mailbox\\_mi300.h](#)

## 6.11 lclk\_dpm\_level\_range Struct Reference

Max and Min Link frequency clock (LCLK) DPM level on a socket. 8 bit NBIO ID, [dpm\\_level](#) struct containing 8 bit max DPM level, 8 bit min DPM level.

```
#include <esmi_mailbox.h>
```

### Data Fields

- `uint8_t nbio_id`  
*NBIOD id (8 bit data [0 - 3])*
- `struct dpm_level dpm`  
*struct with max dpm, min dpm levels*

### 6.11.1 Detailed Description

Max and Min Link frequency clock (LCLK) DPM level on a socket. 8 bit NBIO ID, [dpm\\_level](#) struct containing 8 bit max DPM level, 8 bit min DPM level.

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

- [esmi\\_mailbox.h](#)

## 6.12 link\_id\_bw\_type Struct Reference

APML LINK ID and Bandwidth type Information. It contains APML LINK ID Encoding. Non-MI300 Platforms Valid Link ID encodings are 1(P0), 2(P1), 4(P2), 8(P3), 16(G0), 32(G1), 64(G2), 128(G3). Valid APML MI300 APML LINK ID Encoding. Valid Link ID encodings are 3(P2), 4(P3), 8(G0), 9(G1), 10(G2), 11(G3), 12(G4), 13(G5), 14(G6), 15(G7). IO Bandwidth types 1(Aggregate\_BW), 2 (Read BW), 4 (Write BW).

```
#include <esmi_mailbox.h>
```

### Data Fields

- [apml\\_io\\_bw\\_encoding bw\\_type](#)  
*Bandwidth Type Information [1, 2, 4].*
- [uint8\\_t link\\_id](#)  
*Link ID [3,4,8,9,10,11,12,13,14,15] for MI300 platforms.*

### 6.12.1 Detailed Description

APML LINK ID and Bandwidth type Information. It contains APML LINK ID Encoding. Non-MI300 Platforms Valid Link ID encodings are 1(P0), 2(P1), 4(P2), 8(P3), 16(G0), 32(G1), 64(G2), 128(G3). Valid APML MI300 APML LINK ID Encoding. Valid Link ID encodings are 3(P2), 4(P3), 8(G0), 9(G1), 10(G2), 11(G3), 12(G4), 13(G5), 14(G6), 15(G7). IO Bandwidth types 1(Aggregate\_BW), 2 (Read BW), 4 (Write BW).

### 6.12.2 Field Documentation

#### 6.12.2.1 link\_id

```
uint8_t link_id_bw_type::link_id
```

Link ID [3,4,8,9,10,11,12,13,14,15] for MI300 platforms.

Link ID [1,2,4,8,16,32,64,128] for Non-MI300 platforms

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

- [esmi\\_mailbox.h](#)

## 6.13 max\_ddr\_bw Struct Reference

Structure for Max DDR bandwidth and utilization. It contains max bandwidth(12 bit data) in GBps, current utilization bandwidth(12 bit data) in GBps, current utilized bandwidth( 8 bit data) in percentage.

```
#include <esmi_mailbox.h>
```

## Data Fields

- uint16\_t [max\\_bw](#): 12  
*Max Bandwidth (12 bit data)*
- uint16\_t [utilized\\_bw](#): 12  
*Utilized Bandwidth (12 bit data)*
- uint8\_t [utilized\\_pct](#)  
*Utilized Bandwidth percentage.*

### 6.13.1 Detailed Description

Structure for Max DDR bandwidth and utilization. It contains max bandwidth(12 bit data) in GBps, current utilization bandwidth(12 bit data) in GBps, current utilized bandwidth( 8 bit data) in percentage.

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

- [esmi\\_mailbox.h](#)

## 6.14 max\_mem\_bw Struct Reference

Structure for Max DDR/HBM bandwidth and utilization. It contains max bandwidth(16 bit data) in GBps, current utilization bandwidth(Read+Write)(16 bit data) in GBps.

```
#include <rmi_mailbox_mi300.h>
```

## Data Fields

- uint16\_t [max\\_bw](#)  
*Max Bandwidth (16 bit data)*
- uint16\_t [utilized\\_bw](#)  
*Utilized Bandwidth (16 bit data)*

### 6.14.1 Detailed Description

Structure for Max DDR/HBM bandwidth and utilization. It contains max bandwidth(16 bit data) in GBps, current utilization bandwidth(Read+Write)(16 bit data) in GBps.

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

- [rmi\\_mailbox\\_mi300.h](#)

## 6.15 mca\_bank Struct Reference

MCA bank information.It contains 16 bit Index for MCA Bank and 16 bit offset.

```
#include <esmi_mailbox.h>
```

## Data Fields

- `uint16_t offset`  
*Offset with in MCA Bank.*
- `uint16_t index`  
*Index of MCA Bank.*

### 6.15.1 Detailed Description

MCA bank information.It contains 16 bit Index for MCA Bank and 16 bit offset.

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

- [esmi\\_mailbox.h](#)

## 6.16 mclk\_fclk\_pstates Struct Reference

struct containing memory clock and fabric clock pstate mappings.

```
#include <rmi_mailbox_mi300.h>
```

## Data Fields

- `uint16_t mem_clk`  
*memory clock frequency in MHz*
- `uint16_t f_clk`  
*fabric clock frequenecy in MHz*

### 6.16.1 Detailed Description

struct containing memory clock and fabric clock pstate mappings.

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

- [rmi\\_mailbox\\_mi300.h](#)

## 6.17 nbio\_err\_log Struct Reference

NBIO quadrant(8 bit data) and NBIO register offset(24 bit) data.

```
#include <esmi_mailbox.h>
```

## Data Fields

- `uint8_t quadrant`  
*< NBIO quadrant data*
- `uint32_t offset: 24`  
*< NBIO register offset (24 bit data)*

### 6.17.1 Detailed Description

NBIO quadrant(8 bit data) and NBIO register offset(24 bit) data.

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

- [esmi\\_mailbox.h](#)

## 6.18 oob\_config\_d\_in Struct Reference

Configure oob state infrastructure in SoC. structure consists of `mca_oob_misc0_ec_enable`, `dram_cecc_oob_ec_mode`, `dram_cecc_leak_rate`, `pcie_err_reporting_en` and `core_mca_err_reporting_en`.

```
#include <esmi_mailbox.h>
```

## Data Fields

- `uint8_t mca_oob_misc0_ec_enable: 1`  
*Enable: 0 = disable, 1 = enable.*
- `uint8_t dram_cecc_oob_ec_mode: 2`
- `uint8_t dram_cecc_leak_rate: 5`  
*Valid values are 00 - 1Fh.*
- `uint8_t pcie_err_reporting_en: 1`  
*0 disable and 1 for enable*
- `uint8_t core_mca_err_reporting_en: 1`  
*0 disable and 1 for enable*

### 6.18.1 Detailed Description

Configure oob state infrastructure in SoC. structure consists of `mca_oob_misc0_ec_enable`, `dram_cecc_oob_ec_mode`, `dram_cecc_leak_rate`, `pcie_err_reporting_en` and `core_mca_err_reporting_en`.

### 6.18.2 Field Documentation



#### 6.18.2.1 mca\_oob\_misc0\_ec\_enable

```
uint8_t oob_config_d_in::mca_oob_misc0_ec_enable
```

Enable: 0 = disable, 1 = enable.

MCA OOB MISC0 Error Counter

#### 6.18.2.2 dram\_cecc\_oob\_ec\_mode

```
uint8_t oob_config_d_in::dram_cecc_oob_ec_mode
```

DRAM CECC OOB error counter mode 00 = disable, 01 enable in noleak mode, 10 = enable in leak mode and 11 is reserved

#### 6.18.2.3 dram\_cecc\_leak\_rate

```
uint8_t oob_config_d_in::dram_cecc_leak_rate
```

Valid values are 00 - 1Fh.

DRAM CECC OOB leak rate.

#### 6.18.2.4 pcie\_err\_reporting\_en

```
uint8_t oob_config_d_in::pcie_err_reporting_en
```

0 disable and 1 for enable

PCIe OOB error reporting enable

#### 6.18.2.5 core\_mca\_err\_reporting\_en

```
uint8_t oob_config_d_in::core_mca_err_reporting_en
```

0 disable and 1 for enable

Core MCA OOB error reporting enable

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

- [esmi\\_mailbox.h](#)

## 6.19 pci\_address Struct Reference

PCI address information .PCI address includes 4 bit segment, 12 bit aligned offset, 8 bit bus, 5 bit device info and 3 bit function.

```
#include <esmi_mailbox.h>
```

## Data Fields

- `uint8_t func`: 3  
*function (3 bit data)*
- `uint8_t device`: 5  
*device info (5 bit data)*
- `uint8_t bus`  
*bus (8 bit data)*
- `uint16_t offset`: 12  
*offset address (12 bit data)*
- `uint8_t segment`: 4  
*segment (4 bit data)*

### 6.19.1 Detailed Description

PCI address information .PCI address includes 4 bit segment, 12 bit aligned offset, 8 bit bus, 5 bit device info and 3 bit function.

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

- [esmi\\_mailbox.h](#)

## 6.20 processor\_info Struct Reference

Read Proccessor Info.

```
#include <esmi_cpuid_msr.h>
```

## Data Fields

- `uint32_t family`  
*Processor Family in hexa.*
- `uint32_t model`  
*Processor Model in hexa.*
- `uint32_t step_id`  
*Stepping Identifier in hexa.*

### 6.20.1 Detailed Description

Read Proccessor Info.

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

- [esmi\\_cpuid\\_msr.h](#)

## 6.21 pstate\_freq Struct Reference

DF P-state frequency. It includes mem clock (16 bit data) frequency (DRAM memory clock), data fabric clock (12 bit data), UMC clock divider (UMC) (1 bit data).

```
#include <esmi_mailbox.h>
```

### Data Fields

- `uint16_t mem_clk`  
*DRAM Memory clock Frequency (MHz) (12 bit)*
- `uint16_t fclk: 12`  
*Data fabric clock (MHz) (12 bit data)*
- `uint8_t uclk: 1`  
*UMC clock divider (1 bit data)*

### 6.21.1 Detailed Description

DF P-state frequency. It includes mem clock (16 bit data) frequency (DRAM memory clock), data fabric clock (12 bit data), UMC clock divider (UMC) (1 bit data).

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

- [esmi\\_mailbox.h](#)

## 6.22 ras\_action\_status Struct Reference

Output data for the get BMC RAS action status. Members contain repair result, repair entry number and ras action id.

```
#include <esmi_mailbox_nda.h>
```

### Data Fields

- `uint8_t repair_result`  
*Repair result for the requested entry number.*
- `uint8_t repair_entry_num: 3`  
*Repair entry number.*
- `uint8_t ras_action_id: 4`  
*RAS action ID.*

### 6.22.1 Detailed Description

Output data for the get BMC RAS action status. Members contain repair result, repair entry number and ras action id.

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

- [esmi\\_mailbox\\_nda.h](#)

## 6.23 ras\_df\_err\_chk Struct Reference

RAS df err validity check output status. Structure contains the following members. df\_block\_instances number of block instance with error log to report (0 - 256) err\_log\_len length of error log in bytes per instance (0 - 256).

```
#include <esmi_mailbox.h>
```

### Data Fields

- uint16\_t [df\\_block\\_instances](#): 9  
*Number of DF block instances.*
- uint16\_t [err\\_log\\_len](#): 9  
*len of er log in bytes per inst.*

### 6.23.1 Detailed Description

RAS df err validity check output status. Structure contains the following members. df\_block\_instances number of block instance with error log to report (0 - 256) err\_log\_len length of error log in bytes per instance (0 - 256).

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

- [esmi\\_mailbox.h](#)

## 6.24 ras\_df\_err\_dump Union Reference

RAS df error dump input. Union contains the following members. input[0] 4 byte aligned offset in error log ( 0 - 255) input[1] DF block ID (0 - 255) input[2] Zero based index of DF block instance (0 - 255) input[3] Reserved data\_in 32-bit data input.

```
#include <esmi_mailbox.h>
```

### Data Fields

- uint8\_t [input](#) [4]  
*[2] block ID inst, [3] RESERVED*
- uint32\_t [data\\_in](#)  
*32 bit data in for the DF err dump*

### 6.24.1 Detailed Description

RAS df error dump input. Union contains the following members. input[0] 4 byte aligned offset in error log ( 0 - 255) input[1] DF block ID (0 - 255) input[2] Zero based index of DF block instance (0 - 255) input[3] Reserved data\_in 32-bit data input.

## 6.24.2 Field Documentation

### 6.24.2.1 input

```
uint8_t ras_df_err_dump::input[4]
```

[2] block ID inst, [3] RESERVED

[0] offset, [1] DF block ID

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

- [esmi\\_mailbox.h](#)

## 6.25 ras\_override\_delay Struct Reference

BMC RAS override delay reset CPU on sync flood. The structure contains delay value override in mins [5 -120 mins], disable delay counter and stop delay counter. If disable delay counter is set ResetCpuOnSyncFlood response will NOT be delayed in the next SyncFlood regardless of the value specified in delay\_val\_override. If StopDelayCounter is set it stops the active delay countdown which extends the DelayResetCpuOnSyncFlood indefinitely and system will not reset.

```
#include <esmi_mailbox.h>
```

### Data Fields

- `uint8_t delay_val_override`  
*Delay value override [5 -120 mins].*
- `uint8_t disable_delay_counter: 1`  
*Disable delay counter.*
- `uint8_t stop_delay_counter: 1`  
*stop delay counter*

### 6.25.1 Detailed Description

BMC RAS override delay reset CPU on sync flood. The structure contains delay value override in mins [5 -120 mins], disable delay counter and stop delay counter. If disable delay counter is set ResetCpuOnSyncFlood response will NOT be delayed in the next SyncFlood regardless of the value specified in delay\_val\_override. If StopDelayCounter is set it stops the active delay countdown which extends the DelayResetCpuOnSyncFlood indefinitely and system will not reset.

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

- [esmi\\_mailbox.h](#)

## 6.26 ras\_rt\_err\_req\_type Struct Reference

Get the Error type and request type. Supported Runtime error type[1:0]:

```
#include <esmi_mailbox.h>
```

### Data Fields

- uint8\_t [err\\_type](#): 2  
*Error type, [00, 01, 10].*
- uint8\_t [req\\_type](#): 1  
*Request type, [0, 1].*

### 6.26.1 Detailed Description

Get the Error type and request type. Supported Runtime error type[1:0]:

- 00 = MCA
- 01 = DRAM CECC
- 10 = PCIe Supported Request type[31]:
- 0: polling mode
- 1: interrupt mode

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

- [esmi\\_mailbox.h](#)

## 6.27 ras\_rt\_valid\_err\_inst Struct Reference

Number of valid error instance per category. It consists of number of bytes per each category and number of instances of each category.

```
#include <esmi_mailbox.h>
```

### Data Fields

- uint16\_t [number\\_bytes](#)  
*Number of bytes of given category.*
- uint16\_t [number\\_of\\_inst](#)  
*Number of instances of given catg.*

### 6.27.1 Detailed Description

Number of valid error instance per category. It consists of number of bytes per each category and number of instances of each category.

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

- [esmi\\_mailbox.h](#)

## 6.28 run\_time\_err\_d\_in Struct Reference

Get run time error information data\_in. Runtime error data\_in includes 4 byte aligned offset in instance, runtime error category and zero based index of valid instance number of bytes per each category and number of instances of each category.

```
#include <esmi_mailbox.h>
```

### Data Fields

- [uint8\\_t offset](#)  
*4 byte aligned offset in instance*
- [uint8\\_t category](#)  
*Runtime error category.*
- [uint8\\_t valid\\_inst\\_index](#)  
*Valid inst index returned by 61h.*

### 6.28.1 Detailed Description

Get run time error information data\_in. Runtime error data\_in includes 4 byte aligned offset in instance, runtime error category and zero based index of valid instance number of bytes per each category and number of instances of each category.

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

- [esmi\\_mailbox.h](#)

## 6.29 run\_time\_threshold Struct Reference

Configure threshold counters for MCA, DRAM CECC and PCIE. structure consists of error type, error count threshold and max interrupt rate.

```
#include <esmi_mailbox.h>
```

## Data Fields

- `uint8_t err_type`: 2  
*10 (PCIE\_UE) and 11(RSVD)*
- `uint16_t err_count_th`  
*error count threshold*
- `uint8_t max_intrupt_rate`: 4  
*Max interrupt rate.*

### 6.29.1 Detailed Description

Configure threshold counters for MCA, DRAM CECC and PCIE. structure consists of error type, error count threshold and max interrupt rate.

### 6.29.2 Field Documentation

#### 6.29.2.1 err\_type

```
uint8_t run_time_threshold::err_type
```

10 (PCIE\_UE) and 11(RSVD)

error type [00(MCA), 01(DRAM CECC)

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

- [esmi\\_mailbox.h](#)

## 6.30 set\_ras\_action\_data\_in Struct Reference

data input for set ras action status. Members contain struct [action\\_payload](#), ras action ID and eom flag.

```
#include <esmi_mailbox_nda.h>
```

## Data Fields

- struct [action\\_payload](#) `payload`  
*It contains payload, offset and repair entry number.*
- `uint8_t ras_act_id`: 4  
*RAS action ID.*
- bool [eom\\_flag](#)  
*End of msg flag.*



### 6.30.1 Detailed Description

data input for set ras action status. Members contain struct [action\\_payload](#), ras action ID and eom flag.

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

- [esmi\\_mailbox\\_nda.h](#)

## 6.31 statistics Struct Reference

struct containing statistics parameter of interest and output control pstate mappings.

```
#include <rmi_mailbox_mi300.h>
```

### Data Fields

- `uint16_t` [stat\\_param](#)  
*statistics parameter of interest*
- `uint16_t` [output\\_control](#)  
*Output control.*

### 6.31.1 Detailed Description

struct containing statistics parameter of interest and output control pstate mappings.

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

- [rmi\\_mailbox\\_mi300.h](#)

## 6.32 status\_query\_payload Struct Reference

data for status query payload. Members contain repair entry number.

```
#include <esmi_mailbox_nda.h>
```

### Data Fields

- `uint8_t` [repair\\_entry\\_num](#): 3  
*Repair entry number.*

### 6.32.1 Detailed Description

data for status query payload. Members contain repair entry number.

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

- [esmi\\_mailbox\\_nda.h](#)

## 6.33 svi\_port\_domain Struct Reference

struct containing port and slave address.

```
#include <rmi_mailbox_mi300.h>
```

### Data Fields

- `uint8_t port`: 2  
*SVI port.*
- `uint8_t slave_addr`: 3  
*slave address*

#### 6.33.1 Detailed Description

struct containing port and slave address.

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

- [rmi\\_mailbox\\_mi300.h](#)

## 6.34 temp\_refresh\_rate Struct Reference

DIMM temperature range and refresh rate, temperature update flag.

```
#include <esmi_mailbox.h>
```

### Data Fields

- `uint8_t range`: 3  
*temp refresh rate (3 bit data)*
- `uint8_t ref_rate`: 1  
*temp update flag (1 bit data)*

#### 6.34.1 Detailed Description

DIMM temperature range and refresh rate, temperature update flag.

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

- [esmi\\_mailbox.h](#)

## 6.35 xgmi\_speed\_rate\_n\_width Struct Reference

struct containing xgmi speed rate in MHZ and link width in units of Gpbs. If link\_width[0] = 1 then XGMI link X2 is supported. If link\_width[1] = 1 then XGMI link X4 is supported. If Link\_width[2] = 1 then XGMI link X4 is supported and similarly if link\_width[3] = 1 then XGMI link X8 is supported.

```
#include <rmi_mailbox_mi300.h>
```

### Data Fields

- uint16\_t [speed\\_rate](#)  
*Speed rate.*
- uint8\_t [link\\_width](#): 4  
*XGMI link width.*

#### 6.35.1 Detailed Description

struct containing xgmi speed rate in MHZ and link width in units of Gpbs. If link\_width[0] = 1 then XGMI link X2 is supported. If link\_width[1] = 1 then XGMI link X4 is supported. If Link\_width[2] = 1 then XGMI link X4 is supported and similarly if link\_width[3] = 1 then XGMI link X8 is supported.

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

- [rmi\\_mailbox\\_mi300.h](#)



# Chapter 7

## File Documentation

### 7.1 apml\_err.h File Reference

#### Macros

- #define [OOB\\_CPUID\\_MSR\\_ERR\\_BASE](#) 0x800  
*CPUID MSR FW error code.*
- #define [OOB\\_MAILBOX\\_ERR\\_BASE](#) 0x900  
*MAILBOX FW error code.*

#### Enumerations

- enum [oob\\_status\\_t](#) {  
    [OOB\\_SUCCESS](#) = 0, [OOB\\_NOT\\_FOUND](#), [OOB\\_PERMISSION](#), [OOB\\_NOT\\_SUPPORTED](#),  
    [OOB\\_FILE\\_ERROR](#), [OOB\\_INTERRUPTED](#), [OOB\\_UNEXPECTED\\_SIZE](#), [OOB\\_UNKNOWN\\_ERROR](#),  
    [OOB\\_ARG\\_PTR\\_NULL](#), [OOB\\_NO\\_MEMORY](#), [OOB\\_NOT\\_INITIALIZED](#), [OOB\\_TRY\\_AGAIN](#),  
    [OOB\\_INVALID\\_INPUT](#), [OOB\\_CMD\\_TIMEOUT](#), [OOB\\_INVALID\\_MSGSIZE](#), [OOB\\_CPUID\\_MSR\\_ERR\\_START](#),  
    [OOB\\_CPUID\\_MSR\\_CMD\\_TIMEOUT](#), [OOB\\_CPUID\\_MSR\\_CMD\\_WARM\\_RESET](#), [OOB\\_CPUID\\_MSR\\_CMD\\_UNKNOWN\\_FIRMWARE](#),  
    [OOB\\_CPUID\\_MSR\\_CMD\\_INVALID\\_RD\\_LEN](#),  
    [OOB\\_CPUID\\_MSR\\_CMD\\_EXCESS\\_DATA\\_LEN](#), [OOB\\_CPUID\\_MSR\\_CMD\\_INVALID\\_THREAD](#), [OOB\\_CPUID\\_MSR\\_CMD\\_INVALID\\_THREAD\\_ID](#),  
    [OOB\\_CPUID\\_MSR\\_CMD\\_ABORTED](#),  
    [OOB\\_CPUID\\_MSR\\_ERR\\_END](#), [OOB\\_MAILBOX\\_ERR\\_START](#), [OOB\\_MAILBOX\\_CMD\\_ABORTED](#),  
    [OOB\\_MAILBOX\\_CMD\\_UNKNOWN](#),  
    [OOB\\_MAILBOX\\_CMD\\_INVALID\\_CORE](#), [OOB\\_MAILBOX\\_INVALID\\_INPUT\\_ARGS](#), [OOB\\_MAILBOX\\_INVALID\\_OOBRAS\\_CONFIG](#),  
    [OOB\\_MAILBOX\\_ERR\\_END](#) }  
*Error codes returned by APML\_ERR functions.*

#### Functions

- [oob\\_status\\_t](#) [errno\\_to\\_oob\\_status](#) (int err)  
*convert linux error to esmi error.*
- char \* [esmi\\_get\\_err\\_msg](#) ([oob\\_status\\_t](#) oob\_err)  
*Get the error string message for esmi oob errors.*

### 7.1.1 Detailed Description

Header file for the APML library error/return codes.

This header file has error/return codes for the API.

### 7.1.2 Enumeration Type Documentation

#### 7.1.2.1 oob\_status\_t

```
enum oob_status_t
```

Error codes returned by APML\_ERR functions.

#### Enumerator

OOB_SUCCESS	Operation was successful.
OOB_NOT_FOUND	An item was searched for but not found.
OOB_PERMISSION	many functions require root access to run. Permission denied/EACCESS file error.
OOB_NOT_SUPPORTED	The requested information or action is not available for the given input, on the given system
OOB_FILE_ERROR	Problem accessing a file. This may because the operation is not supported by the Linux kernel version running on the executing machine
OOB_INTERRUPTED	execution of function An interrupt occurred during
OOB_UNEXPECTED_SIZE	was read An unexpected amount of data
OOB_UNKNOWN_ERROR	An unknown error occurred.
OOB_ARG_PTR_NULL	Parsed argument ptr null.
OOB_NO_MEMORY	Not enough memory to allocate.
OOB_NOT_INITIALIZED	APML object not initialized.
OOB_TRY_AGAIN	No match Try again.
OOB_INVALID_INPUT	Input value is invalid.
OOB_CMD_TIMEOUT	Command timed out.
OOB_INVALID_MSGSIZE	Mesg size too long.
OOB_CPUID_MSR_CMD_TIMEOUT	RMI cmd timeout.
OOB_CPUID_MSR_CMD_WARM_RESET	Warm reset during RMI cmd.
OOB_CPUID_MSR_CMD_UNKNOWN_FMT	Cmd fmt field not recognised.
OOB_CPUID_MSR_CMD_INVALID_RD_LEN	RMI cmd invalid read len.
OOB_CPUID_MSR_CMD_EXCESS_DATA_LEN	excess data
OOB_CPUID_MSR_CMD_INVALID_THREAD	Invalid thread selected.
OOB_CPUID_MSR_CMD_UNSUPP	Cmd not supported.
OOB_CPUID_MSR_CMD_ABORTED	Cmd aborted.
OOB_MAILBOX_CMD_ABORTED	Mailbox cmd aborted.
OOB_MAILBOX_CMD_UNKNOWN	Unknown mailbox cmd.
OOB_MAILBOX_CMD_INVALID_CORE	Invalid core.
OOB_MAILBOX_INVALID_INPUT_ARGS	Invalid Input Arguments.
OOB_MAILBOX_INVALID_OOBRAS_CONFIG	Invalid OOB RAS config.

## 7.2 apml\_recovery.h File Reference

### Enumerations

- enum [apml\\_client](#) { **DEV\_SBRMI** = 0x0, **DEV\_SBTSI** }  
*APML Client Devices.*

### Functions

- [oob\\_status\\_t](#) [apml\\_recover\\_dev](#) (uint8\_t soc\_num, uint8\_t client)  
*Recover the APML client device for the given socket.*

#### 7.2.1 Detailed Description

Header file for the APML recovery flow

#### 7.2.2 Function Documentation

##### 7.2.2.1 apml\_recover\_dev()

```
oob_status_t apml_recover_dev (  
    uint8_t soc_num,  
    uint8_t client )
```

Recover the APML client device for the given socket.

This function will recover the APML client device and returns successful on recovery or error if recovery is unsuccessful

NOTE: This fix is a software workaround for the erratum, Erratum:1444, "Advanced Platform Management Link (APML) May Cease to Function After Incomplete Read Transaction" Further details can be found in mentioned tech doc [https://www.amd.com/system/files/TechDocs/57095-PUB\\_1.00.pdf](https://www.amd.com/system/files/TechDocs/57095-PUB_1.00.pdf)

##### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>client</i>	DEV_SBRMI[0]/DEV_SBTSI[1] enum: apml_client

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful recovery
<i>Non-zero</i>	is returned upon failure.

## 7.3 esmi\_cpuid\_msr.h File Reference

```
#include "apml_err.h"
```

### Data Structures

- struct [processor\\_info](#)  
*Read Proccessor Info.*

### Enumerations

- enum [cpuid\\_reg](#) { **EAX** = 0, **EBX**, **ECX**, **EDX** }

### Functions

- [oob\\_status\\_t esmi\\_get\\_vendor\\_id](#) (uint8\_t soc\_num, char \*vendor\_id)  
*Get the number of logical cores per socket.*
- [oob\\_status\\_t esmi\\_get\\_processor\\_info](#) (uint8\_t soc\_num, struct [processor\\_info](#) \*proc\_info)  
*Get the number of logical cores per socket.*
- [oob\\_status\\_t esmi\\_get\\_logical\\_cores\\_per\\_socket](#) (uint8\_t soc\_num, uint32\_t \*logical\_cores\_per\_socket)  
*Get the number of logical cores per socket.*
- [oob\\_status\\_t esmi\\_get\\_threads\\_per\\_core](#) (uint8\_t soc\_num, uint32\_t \*threads\_per\_core)  
*Get number of threads per core.*
- [oob\\_status\\_t esmi\\_oob\\_read\\_msr](#) (uint8\_t soc\_num, uint32\_t thread, uint32\_t msraddr, uint64\_t \*buffer)  
*Read the MCA MSR register for a given thread.*
- [oob\\_status\\_t esmi\\_oob\\_cpuid](#) (uint8\_t soc\_num, uint32\_t thread, uint32\_t \*eax, uint32\_t \*ebx, uint32\_t \*ecx, uint32\_t \*edx)  
*Read CPUID functionality for a particular thread in a system.*
- [oob\\_status\\_t esmi\\_oob\\_cpuid\\_eax](#) (uint8\_t soc\_num, uint32\_t thread, uint32\_t fn\_eax, uint32\_t fn\_ecx, uint32\_t \*eax)  
*Read eax register on CPUID functionality.*
- [oob\\_status\\_t esmi\\_oob\\_cpuid\\_ebx](#) (uint8\_t soc\_num, uint32\_t thread, uint32\_t fn\_eax, uint32\_t fn\_ecx, uint32\_t \*ebx)  
*Read ebx register on CPUID functionality.*
- [oob\\_status\\_t esmi\\_oob\\_cpuid\\_ecx](#) (uint8\_t soc\_num, uint32\_t thread, uint32\_t fn\_eax, uint32\_t fn\_ecx, uint32\_t \*ecx)  
*Read ecx register on CPUID functionality.*
- [oob\\_status\\_t esmi\\_oob\\_cpuid\\_edx](#) (uint8\_t soc\_num, uint32\_t thread, uint32\_t fn\_eax, uint32\_t fn\_ecx, uint32\_t \*edx)  
*Read edx register on CPUID functionality.*
- [oob\\_status\\_t read\\_max\\_threads\\_per\\_l3](#) (uint8\_t soc\_num, uint32\_t \*threads\_l3)  
*Read max threads per L3 cache.*

### Variables

- struct [processor\\_info](#) [plat\\_info](#) [1]  
*Platform Info instance.*



### 7.3.1 Detailed Description

Header file for the APML library cpuid and msr read functions. All required function, structure, enum and protocol specific data etc. definitions should be defined in this header.

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

### 7.3.2 Enumeration Type Documentation

#### 7.3.2.1 cpuid\_reg

enum [cpuid\\_reg](#)

CPUID register indexes 0 for EAX, 1 for EBX, 2 ECX and 3 for EDX

## 7.4 esmi\_mailbox.h File Reference

```
#include "apml_common.h"
#include "apml_err.h"
#include <stdbool.h>
```

### Data Structures

- struct [dimm\\_power](#)  
*DIMM power(mW), update rate(ms) and dimm address.*
- struct [dimm\\_thermal](#)  
*DIMM thermal sensor (degree C), update rate and dimm address.*
- struct [temp\\_refresh\\_rate](#)  
*DIMM temperature range and refresh rate, temperature update flag.*
- struct [pci\\_address](#)  
*PCI address information .PCI address includes 4 bit segment, 12 bit aligned offset, 8 bit bus, 5 bit device info and 3 bit function.*
- struct [dpm\\_level](#)  
*Max and min LCK DPM level on a given NBIO ID. Valid Max and min DPM level values are 0 - 1.*
- struct [lclk\\_dpm\\_level\\_range](#)  
*Max and Min Link frequency clock (LCLK) DPM level on a socket. 8 bit NBIO ID, [dpm\\_level](#) struct containing 8 bit max DPM level, 8 bit min DPM level.*
- struct [nbio\\_err\\_log](#)  
*NBIO quadrant(8 bit data) and NBIO register offset(24 bit) data.*
- struct [max\\_ddr\\_bw](#)  
*Structure for Max DDR bandwidth and utilization. It contains max bandwidth(12 bit data) in GBps, current utilization bandwidth(12 bit data) in GBps, current utilized bandwidth( 8 bit data) in percentage.*
- struct [mca\\_bank](#)  
*MCA bank information.It contains 16 bit Index for MCA Bank and 16 bit offset.*

- struct [link\\_id\\_bw\\_type](#)  
*APML LINK ID and Bandwidth type Information. It contains APML LINK ID Encoding. Non-MI300 Platforms Valid Link ID encodings are 1(P0), 2(P1), 4(P2), 8(P3), 16(G0), 32(G1), 64(G2), 128(G3). Valid APML MI300 APML LINK ID Encoding. Valid Link ID encodings are 3(P2), 4(P3), 8(G0), 9(G1), 10(G2), 11(G3), 12(G4), 13(G5), 14(G6), 15(G7). IO Bandwidth types 1(Aggregate\_BW), 2 (Read BW), 4 (Write BW).*
- struct [pstate\\_freq](#)  
*DF P-state frequency. It includes mem clock(16 bit data) frequency (DRAM memory clock), data fabric clock (12 bit data), UMC clock divider (UMC) (1 bit data).*
- struct [ras\\_override\\_delay](#)  
*BMC RAS override delay reset CPU on sync flood. The structure contains delay value override in mins [5 -120 mins], disable delay counter and stop delay counter. If disable delay counter is set ResetCpuOnSyncFlood response will NOT be delayed in the next SyncFlood regardless of the value specified in delay\_val\_override. If StopDelayCounter is set it stops the active delay countdown which extends the DelayResetCpuOnSyncFlood indefinitely and system will not reset.*
- struct [ras\\_rt\\_err\\_req\\_type](#)  
*Get the Error type and request type. Supported Runtime error type[1:0]:*
- struct [ras\\_rt\\_valid\\_err\\_inst](#)  
*Number of valid error instance per category. It consists of number of bytes per each category and number of instances of each category.*
- struct [run\\_time\\_err\\_d\\_in](#)  
*Get run time error information data\_in. Runtime error data\_in includes 4 byte aligned offset in instance, runtime error category and zero based index of valid instance number of bytes per each category and number of instances of each category.*
- struct [run\\_time\\_threshold](#)  
*Configure threshold counters for MCA, DRAM CECC and PCIE. structure consists of error type, error count threshold and max interrupt rate.*
- struct [oob\\_config\\_d\\_in](#)  
*Configure oob state infrastructure in SoC. structure consists of mca\_oob\_misc0\_ec\_enable, dram\_cecc\_oob\_ec\_mode, dram\_cecc\_leak\_rate, pcie\_err\_reporting\_en and core\_mca\_err\_reporting\_en.*
- struct [ras\\_df\\_err\\_chk](#)  
*RAS df err validity check output status. Structure contains the following members. df\_block\_instances number of block instance with error log to report (0 - 256) err\_log\_len length of error log in bytes per instance (0 - 256).*
- union [ras\\_df\\_err\\_dump](#)  
*RAS df error dump input. Union contains the following members. input[0] 4 byte aligned offset in error log ( 0 - 255) input[1] DF block ID (0 - 255) input[2] Zero based index of DF block instance (0 - 255) input[3] Reserved data\_in 32-bit data input.*
- struct [dimm\\_spd\\_d\\_in](#)  
*dimm\_spd\_d\_in: structure contains following members to be passed as input, to get 4 byte DIMM spd register data*

## Macros

- #define [BYTE\\_BITS](#) 8  
*Byte bits //.*
- #define [WORD\\_BITS](#) 16  
*word bits //*
- #define [DRAM\\_CECC\\_OOB\\_EC\\_MODE](#) 1  
*DRAM cecc oob ec mode //.*
- #define [ERR\\_COUNT\\_TH](#) 2  
*Error count threshold position //.*
- #define [DRAM\\_CECC\\_LEAK\\_RATE](#) 3  
*DRAM CECC Leak rate //.*
- #define [PCIE\\_ERR\\_REPORT\\_EN](#) 8  
*PCIE oob counter enable //.*

- #define [MCA\\_TH\\_INTR](#) 11  
*MCA threshold interrupt //.*
- #define [CECC\\_TH\\_INTR](#) 12  
*CECC threshold interrupt //.*
- #define [PCIE\\_TH\\_INTR](#) 13  
*PCIE threshold interrupt //.*
- #define [MCA\\_MAX\\_INTR\\_RATE](#) 15  
*MCA max interrupt rate //.*
- #define [MAX\\_INTR\\_RATE\\_POS](#) 18  
*Max interrupt rate position //.*
- #define [DRAM\\_CECC\\_MAX\\_INTR\\_RATE](#) 19  
*DRAM CECC Max interrupt rate //.*
- #define [PCIE\\_MAX\\_INTR\\_RATE](#) 23  
*PCIE Max interrupt rate //.*
- #define [CORE\\_MCA\\_ERR\\_REPORT\\_EN](#) 31  
*CORE MCA error report enable //.*
- #define [MAX\\_ERR\\_LOG\\_LEN](#) 256  
*Max error log length //.*
- #define [MAX\\_DF\\_BLOCK\\_IDS](#) 256  
*Max DF block IDs //.*
- #define [MAX\\_DF\\_BLOCK\\_INSTS](#) 256  
*Max DF block instances //.*
- #define [LEGACY\\_ENCODING\\_SIZE](#) 8  
*Legacy encoding size //.*

## Enumerations

- enum [esb\\_mailbox\\_commands](#) {  
[READ\\_PACKAGE\\_POWER\\_CONSUMPTION](#) = 0x1, [WRITE\\_PACKAGE\\_POWER\\_LIMIT](#), [READ\\_PACKAGE\\_POWER\\_LIMIT](#), [READ\\_MAX\\_PACKAGE\\_POWER\\_LIMIT](#),  
[READ\\_TDP](#), [READ\\_MAX\\_cTDP](#), [READ\\_MIN\\_cTDP](#), [READ\\_BIOS\\_BOOST\\_Fmax](#),  
[READ\\_APML\\_BOOST\\_LIMIT](#), [WRITE\\_APML\\_BOOST\\_LIMIT](#), [WRITE\\_APML\\_BOOST\\_LIMIT\\_ALLCORES](#), [READ\\_DRAM\\_THROTTLE](#),  
[WRITE\\_DRAM\\_THROTTLE](#), [READ\\_PROCHOT\\_STATUS](#), [READ\\_PROCHOT\\_RESIDENCY](#), [READ\\_NBIO\\_ERROR\\_LOGGING\\_REGISTER](#) = 0x11,  
[READ\\_IOD\\_BIST](#) = 0x13, [READ\\_CCD\\_BIST\\_RESULT](#), [READ\\_CCX\\_BIST\\_RESULT](#), [READ\\_PACKAGE\\_CCLK\\_FREQ\\_LIMIT](#),  
[READ\\_PACKAGE\\_C0\\_RESIDENCY](#), [READ\\_DDR\\_BANDWIDTH](#) = 0x18, [READ\\_PPIN\\_FUSE](#) = 0x1F, [GET\\_POST\\_CODE](#) = 0x20,  
[GET\\_RTC](#), [WRITE\\_BMC\\_REPORT\\_DIMM\\_POWER](#) = 0x40, [WRITE\\_BMC\\_REPORT\\_DIMM\\_THERMAL\\_SENSOR](#), [READ\\_BMC\\_RAS\\_PCIE\\_CONFIG\\_ACCESS](#),  
[READ\\_BMC\\_RAS\\_MCA\\_VALIDITY\\_CHECK](#), [READ\\_BMC\\_RAS\\_MCA\\_MSR\\_DUMP](#), [READ\\_BMC\\_RAS\\_FCH\\_RESET\\_REASON](#), [READ\\_DIMM\\_TEMP\\_RANGE\\_AND\\_REFRESH\\_RATE](#),  
[READ\\_DIMM\\_POWER\\_CONSUMPTION](#), [READ\\_DIMM\\_THERMAL\\_SENSOR](#), [READ\\_PWR\\_CURRENT\\_ACTIVE\\_FREQ\\_LIMIT\\_SOCKET](#), [READ\\_PWR\\_CURRENT\\_ACTIVE\\_FREQ\\_LIMIT\\_CORE](#),  
[READ\\_PWR\\_SVI\\_TELEMETRY\\_ALL\\_RAILS](#), [READ\\_SOCKET\\_FREQ\\_RANGE](#), [READ\\_CURRENT\\_IO\\_BANDWIDTH](#), [READ\\_CURRENT\\_XGMI\\_BANDWIDTH](#),  
[WRITE\\_GMI3\\_LINK\\_WIDTH\\_RANGE](#), [WRITE\\_XGMI\\_LINK\\_WIDTH\\_RANGE](#), [WRITE\\_APB\\_DISABLE](#), [WRITE\\_APB\\_ENABLE](#),  
[READ\\_CURRENT\\_DFPSTATE\\_FREQUENCY](#), [WRITE\\_LCLK\\_DPM\\_LEVEL\\_RANGE](#), [READ\\_BMC\\_RAPL\\_UNITS](#), [READ\\_BMC\\_RAPL\\_CORE\\_LO\\_COUNTER](#),  
[READ\\_BMC\\_RAPL\\_CORE\\_HI\\_COUNTER](#), [READ\\_BMC\\_RAPL\\_PKG\\_COUNTER](#), [READ\\_BMC\\_CPU\\_BASE\\_FREQUENCY](#), [READ\\_BMC\\_CONTROL\\_PCIE\\_GEN5\\_RATE](#),

```

READ_RAS_LAST_TRANS_ADDR_CHK, READ_RAS_LAST_TRANS_ADDR_DUMP, WRITE_PWR_↵
EFFICIENCY_MODE, WRITE_DF_PSTATE_RANGE,
READ_LCLK_DPM_LEVEL_RANGE, READ_UCODE_REVISION, GET_BMC_RAS_RUNTIME_ERR_V↵
ALIDITY_CHECK, GET_BMC_RAS_RUNTIME_ERR_INFO,
SET_BMC_RAS_ERR_THRESHOLD, SET_BM_RAS_OOB_CONFIG, GET_BMC_RAS_OOB_CONFIG,
BMC_RAS_DELAY_RESET_ON_SYNCFLOOD_OVERRIDE = 0x6A,
READ_BMC_RAS_RESET_ON_SYNC_FLOOD, GET_DIMM_SPD = 0x70 }

```

*Mailbox message types defined in the APML library.*

- enum [apml\\_io\\_bw\\_encoding](#) { **AGG\_BW** = BIT(0), **RD\_BW** = BIT(1), **WR\_BW** = BIT(2) }

*APML IO Bandwidth Encoding defined in the APML library.*

- enum [apml\\_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) }

*APML IO LINK ID Encoding defined in the APML library.*

## Functions

- [oob\\_status\\_t read\\_socket\\_power](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the power consumption of the socket.*
- [oob\\_status\\_t read\\_socket\\_power\\_limit](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the current power cap/limit value for a given socket.*
- [oob\\_status\\_t read\\_max\\_socket\\_power\\_limit](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the maximum value that can be assigned as a power cap/limit for a given socket.*
- [oob\\_status\\_t write\\_socket\\_power\\_limit](#) (uint8\_t soc\_num, uint32\_t limit)  
*Set the power cap/limit value for a given socket.*
- [oob\\_status\\_t read\\_esb\\_boost\\_limit](#) (uint8\_t soc\_num, uint32\_t value, uint32\_t \*buffer)  
*Get the Out-of-band boostlimit value for a given core.*
- [oob\\_status\\_t read\\_bios\\_boost\\_fmax](#) (uint8\_t soc\_num, uint32\_t value, uint32\_t \*buffer)  
*Get the In-band maximum boostlimit value for a given core.*
- [oob\\_status\\_t write\\_esb\\_boost\\_limit](#) (uint8\_t soc\_num, uint32\_t cpu\_ind, uint32\_t limit)  
*Set the Out-of-band boostlimit value for a given core.*
- [oob\\_status\\_t write\\_esb\\_boost\\_limit\\_allcores](#) (uint8\_t soc\_num, uint32\_t limit)  
*Set the boostlimit value for the whole socket (whole system).*
- [oob\\_status\\_t read\\_tdp](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the Thermal Design Power limit TDP of the socket with provided socket index.*
- [oob\\_status\\_t read\\_max\\_tdp](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the Maximum Thermal Design Power limit TDP of the socket with provided socket index.*
- [oob\\_status\\_t read\\_min\\_tdp](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the Minimum Thermal Design Power limit TDP of the socket.*
- [oob\\_status\\_t read\\_prochot\\_status](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Get the Prochot Status of the socket with provided socket index.*
- [oob\\_status\\_t read\\_prochot\\_residency](#) (uint8\_t soc\_num, float \*buffer)  
*Get the Prochot Residency (since the boot time or last read of Prochot Residency) of the socket.*
- [oob\\_status\\_t read\\_dram\\_throttle](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Read Dram Throttle will always read the highest percentage value.*
- [oob\\_status\\_t write\\_dram\\_throttle](#) (uint8\_t soc\_num, uint32\_t limit)  
*Set Dram Throttle value in terms of percentage.*
- [oob\\_status\\_t read\\_vddio\\_mem\\_power](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Read VDDIOMem Power returns the estimated VDDIOMem power consumed within the socket.*
- [oob\\_status\\_t read\\_nbio\\_error\\_logging\\_register](#) (uint8\_t soc\_num, struct [nbio\\_err\\_log](#) nbio, uint32\_t \*buffer)  
*Read NBIO Error Logging Register.*

- [oob\\_status\\_t read\\_iod\\_bist](#) (uint8\_t soc\_num, uint32\_t \*buffer)  
*Read IOD Bist status.*
- [oob\\_status\\_t read\\_ccd\\_bist\\_result](#) (uint8\_t soc\_num, uint32\_t input, uint32\_t \*buffer)  
*Read CCD Bist status. Results are read for each CCD present in the system.*
- [oob\\_status\\_t read\\_ccx\\_bist\\_result](#) (uint8\_t soc\_num, uint32\_t value, uint32\_t \*ccx\_bist)  
*Read CPU Core Complex Bist result. results are read for each Logical CCX instance number and returns a value which is the concatenation of L3 pass status and all cores in the complex(n:0).*
- [oob\\_status\\_t read\\_cclk\\_freq\\_limit](#) (uint8\_t soc\_num, uint32\_t \*cclk\_freq)  
*Read CCLK frequency limit for the given socket.*
- [oob\\_status\\_t read\\_socket\\_c0\\_residency](#) (uint8\_t soc\_num, uint32\_t \*c0\_res)  
*Read socket C0 residency.*
- [oob\\_status\\_t read\\_ddr\\_bandwidth](#) (uint8\_t soc\_num, struct [max\\_ddr\\_bw](#) \*max\_ddr)  
*Get the Theoretical maximum DDR Bandwidth of the system in GB/s, Current utilized DDR Bandwidth (Read + Write) in GB/s and Current utilized DDR Bandwidth as a percentage of theoretical maximum. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-1Ah\_Mod-00h-0Fh.*
- [oob\\_status\\_t write\\_bmc\\_report\\_dimm\\_power](#) (uint8\_t soc\_num, struct [dimm\\_power](#) dp\_info)  
*Set DIMM Power consumption in mwatts.*
- [oob\\_status\\_t write\\_bmc\\_report\\_dimm\\_thermal\\_sensor](#) (uint8\_t soc\_num, struct [dimm\\_thermal](#) dt\_info)  
*Set DIMM thermal Sensor in degree Celcius.*
- [oob\\_status\\_t read\\_bmc\\_ras\\_pcie\\_config\\_access](#) (uint8\_t soc\_num, struct [pci\\_address](#) pci\_addr, uint32\_t \*out\_buf)  
*Read BMC RAS PCIE config access.*
- [oob\\_status\\_t read\\_bmc\\_ras\\_mca\\_validity\\_check](#) (uint8\_t soc\_num, uint16\_t \*bytes\_per\_mca, uint16\_t \*mca\_banks)  
*Read number of MCA banks with valid status after a fatal error.*
- [oob\\_status\\_t read\\_bmc\\_ras\\_mca\\_msr\\_dump](#) (uint8\_t soc\_num, struct [mca\\_bank](#) mca\_dump, uint32\_t \*out\_buf)  
*Read data from mca bank reported by bmc ras mca validity check.*
- [oob\\_status\\_t read\\_bmc\\_ras\\_fch\\_reset\\_reason](#) (uint8\_t soc\_num, uint32\_t input, uint32\_t \*out\_buf)  
*Read FCH reason code from the previous reset.*
- [oob\\_status\\_t read\\_dimm\\_temp\\_range\\_and\\_refresh\\_rate](#) (uint8\_t soc\_num, uint32\_t dimm\_addr, struct [temp\\_refresh\\_rate](#) \*rate)  
*Read DIMM temperature range and refresh rate.*
- [oob\\_status\\_t read\\_dimm\\_power\\_consumption](#) (uint8\_t soc\_num, uint32\_t dimm\_addr, struct [dimm\\_power](#) \*dimm\_pow)  
*Read DIMM power consumption.*
- [oob\\_status\\_t read\\_dimm\\_thermal\\_sensor](#) (uint8\_t soc\_num, uint32\_t dimm\_addr, struct [dimm\\_thermal](#) \*dimm\_temp)  
*Read DIMM thermal sensor.*
- [oob\\_status\\_t read\\_pwr\\_current\\_active\\_freq\\_limit\\_socket](#) (uint8\_t soc\_num, uint16\_t \*freq, char \*\*source\_type)  
*Read current active frequency limit per socket.*
- [oob\\_status\\_t read\\_pwr\\_current\\_active\\_freq\\_limit\\_core](#) (uint8\_t soc\_num, uint32\_t core\_id, uint16\_t \*base\_freq)  
*Read current active frequency limit set per core.*
- [oob\\_status\\_t read\\_pwr\\_svi\\_telemetry\\_all\\_rails](#) (uint8\_t soc\_num, uint32\_t \*power)  
*Read SVR based telemetry for all rails.*
- [oob\\_status\\_t read\\_socket\\_freq\\_range](#) (uint8\_t soc\_num, uint16\_t \*fmax, uint16\_t \*fmin)  
*Read socket frequency range.*
- [oob\\_status\\_t read\\_current\\_io\\_bandwidth](#) (uint8\_t soc\_num, struct [link\\_id\\_bw\\_type](#) link, uint32\_t \*io\_bw)  
*Read current bandwidth on IO Link.*
- [oob\\_status\\_t read\\_current\\_xgmi\\_bandwidth](#) (uint8\_t soc\_num, struct [link\\_id\\_bw\\_type](#) link, uint32\_t \*xgmi\_bw)

- Read current bandwidth on xGMI Link.*

  - [oob\\_status\\_t write\\_gmi3\\_link\\_width\\_range](#) (uint8\_t soc\_num, uint8\_t min\_link\_width, uint8\_t max\_link\_width)

*Set the max and min width of GMI3 link.*
- [oob\\_status\\_t write\\_xgmi\\_link\\_width\\_range](#) (uint8\_t soc\_num, uint8\_t min\_link\_width, uint8\_t max\_link\_width)

*Set the max and min width of xGMI link.*
- [oob\\_status\\_t write\\_apb\\_disable](#) (uint8\_t soc\_num, uint8\_t df\_pstate, bool \*prochot\_asserted)

*Set the APBDisabled.*
- [oob\\_status\\_t write\\_apb\\_enable](#) (uint8\_t soc\_num, bool \*prochot\_asserted)

*Enable the DF p-state performance boost algorithm.*
- [oob\\_status\\_t read\\_current\\_dfpstate\\_frequency](#) (uint8\_t soc\_num, struct [pstate\\_freq](#) \*df\_pstate)

*Read current DF p-state frequency .*
- [oob\\_status\\_t write\\_lclk\\_dpm\\_level\\_range](#) (uint8\_t soc\_num, struct [lclk\\_dpm\\_level\\_range](#) lclk)

*Set the max and min LCK DPM Level on a given NBIO per socket.*
- [oob\\_status\\_t read\\_bmc\\_rapl\\_units](#) (uint8\_t soc\_num, uint8\_t \*tu\_value, uint8\_t \*esu\_value)

*Read RAPL (Running Average Power Limit) Units.*
- [oob\\_status\\_t read\\_bmc\\_cpu\\_base\\_frequency](#) (uint8\_t soc\_num, uint16\_t \*base\_freq)

*Read RAPL base frequency per CPU socket.*
- [oob\\_status\\_t read\\_bmc\\_control\\_pcie\\_gen5\\_rate](#) (uint8\_t soc\_num, uint8\_t rate, uint8\_t \*mode)

*Control PCIe Rate on Gen5-Capable devices..*
- [oob\\_status\\_t read\\_rapl\\_core\\_energy\\_counters](#) (uint8\_t soc\_num, uint32\_t core\_id, double \*energy\_↵ counters)

*Read RAPL core energy counters.*
- [oob\\_status\\_t read\\_rapl\\_pkg\\_energy\\_counters](#) (uint8\_t soc\_num, double \*energy\_counters)

*Read RAPL package energy counters.*
- [oob\\_status\\_t write\\_pwr\\_efficiency\\_mode](#) (uint8\_t soc\_num, uint8\_t mode)

*Write power efficiency profile policy.*
- [oob\\_status\\_t write\\_df\\_pstate\\_range](#) (uint8\_t soc\_num, uint8\_t max\_pstate, uint8\_t min\_pstate)

*Write df pstate range.*
- [oob\\_status\\_t read\\_lclk\\_dpm\\_level\\_range](#) (uint8\_t soc\_num, uint8\_t nbio\_id, struct [dpm\\_level](#) \*dpm)

*Read LCLK Max and Min DPM level range.*
- [oob\\_status\\_t read\\_ucode\\_revision](#) (uint8\_t soc\_num, uint32\_t \*ucode\_rev)

*Read ucode revision.*
- [oob\\_status\\_t reset\\_on\\_sync\\_flood](#) (uint8\_t soc\_num, uint32\_t \*ack\_resp)

*Read BMC RAS reset on sync flood.*
- [oob\\_status\\_t override\\_delay\\_reset\\_on\\_sync\\_flood](#) (uint8\_t soc\_num, struct [ras\\_override\\_delay](#) data\_in, bool \*ack\_resp)

*Overrides delay reset cpu on sync flood value.*
- [oob\\_status\\_t get\\_post\\_code](#) (uint8\_t soc\_num, uint32\_t offset, uint32\_t \*post\_code)

*Read post code.*
- [oob\\_status\\_t get\\_bmc\\_ras\\_run\\_time\\_err\\_validity\\_ck](#) (uint8\_t soc\_num, struct [ras\\_rt\\_err\\_req\\_type](#) err\_↵ category, struct [ras\\_rt\\_valid\\_err\\_inst](#) \*inst)

*Reads number of valid error instances per category.*
- [oob\\_status\\_t get\\_bmc\\_ras\\_run\\_time\\_error\\_info](#) (uint8\_t soc\_num, struct [run\\_time\\_err\\_d\\_in](#) d\_in, uint32\_t \*err\_info)

*Reads BMC RAS runtime error information.*
- [oob\\_status\\_t set\\_bmc\\_ras\\_err\\_threshold](#) (uint8\_t soc\_num, struct [run\\_time\\_threshold](#) th)

*Sets BMC RAS error threshold.*
- [oob\\_status\\_t set\\_bmc\\_ras\\_oob\\_config](#) (uint8\_t soc\_num, struct [oob\\_config\\_d\\_in](#) d\_in)

*Configures OOB state infrastructure in SoC.*
- [oob\\_status\\_t get\\_bmc\\_ras\\_oob\\_config](#) (uint8\_t soc\_num, uint32\_t \*oob\_config)

*Reads the current status of OOB state infrastructure in SoC.*

- [oob\\_status\\_t read\\_ppin\\_fuse](#) (uint8\_t soc\_num, uint64\_t \*data)  
*Get the 64 bit PPIN fuse.*
- [oob\\_status\\_t read\\_ras\\_df\\_err\\_validity\\_check](#) (uint8\_t soc\_num, uint8\_t df\_block\_id, struct [ras\\_df\\_err\\_chk](#) \*err\_chk)  
*Read number of instances of DF blocks of type DF\_BLOCK\_ID with errors.*
- [oob\\_status\\_t read\\_ras\\_df\\_err\\_dump](#) (uint8\_t soc\_num, union [ras\\_df\\_err\\_dump](#) ras\_err, uint32\_t \*data)  
*Read RAS DF error dump.*
- [oob\\_status\\_t read\\_rtc](#) (uint8\_t soc\_num, uint64\_t \*rtc)  
*Read RTC.*
- [oob\\_status\\_t read\\_dimm\\_spd\\_register](#) (uint8\_t soc\_num, struct [dimm\\_spd\\_d\\_in](#) spd\_d\_in, uint32\_t \*spd\_data)  
*Get SPD SB data.*
- [oob\\_status\\_t get\\_dimm\\_serial\\_num](#) (uint8\_t soc\_num, uint8\_t dimm\_addr, uint32\_t \*serial\_num)  
*Get DIMM serial number.*

## Variables

- float [esu\\_multiplier](#)  
*energy status multiplier value is  $1/2^{\wedge}ESU$  where ESU is [12:8] bit of the mailbox command 0x55h.*

### 7.4.1 Detailed Description

Header file for the Mailbox messages supported by APML library. All required function, structure, enum, etc. definitions should be defined in this file.

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

### 7.4.2 Function Documentation

#### 7.4.2.1 write\_bmc\_report\_dimm\_power()

```
oob_status_t write_bmc_report_dimm_power (
    uint8_t soc_num,
    struct dimm_power dp_info )
```

Set DIMM Power consumption in mwatts.

This function will set DIMM Power consumption periodically by BMC at specified update rate (10 ms or less) when bmc owns the SPD side-band bus. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>dp_info</i>	<a href="#">dimm_power</a> Struct with power(mw), updatarate(ms) & dimm address

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.2 write\_bmc\_report\_dimm\_thermal\_sensor()

```
oob_status_t write_bmc_report_dimm_thermal_sensor (
    uint8_t soc_num,
    struct dimm_thermal dt_info )
```

Set DIMM thermal Sensor in degree Celcius.

This function will set DIMM thermal sensor (in degree celcius) periodically by BMC at specified update rate (10 ms or less) when bmc owns the SPD side-band bus. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>dt_info</i>	struct with temp( <sup>o</sup> C), updatarate(ms) & dimm address

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.3 read\_bmc\_ras\_pcie\_config\_access()

```
oob_status_t read_bmc_ras_pcie_config_access (
    uint8_t soc_num,
    struct pci_address pci_addr,
    uint32_t * out_buf )
```

Read BMC RAS PCIE config access.

This function will read the 32 bit BMC RAS extended PCI config space.

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>pci_addr</i>	<a href="#">pci_address</a> structure with function(3 bit), device(4 bit) bus(8 bit), offset(12 bit), segment(4 bit). SEGMENT:0 BUS 0:DEVICE 18 and SEGMENT:0 BUS 0:DEVICE 19 are inaccessible. Supported platforms: <a href="#">Fam-19h_Mod-10h-1Fh</a> , <a href="#">Fam-1Ah_Mod-00h-0Fh</a> and <a href="#">Fam-19h_Mod-90h-9Fh</a> .
out	<i>out_buf</i>	32 bit data from offset in PCI config space.



## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.4 read\_bmc\_ras\_mca\_validity\_check()

```
oob_status_t read_bmc_ras_mca_validity_check (
    uint8_t soc_num,
    uint16_t * bytes_per_mca,
    uint16_t * mca_banks )
```

Read number of MCA banks with valid status after a fatal error.

This function returns the number of MCA banks with valid status after a fatal error. Supported platforms↵: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>bytes_per_mca</i>	returns bytes per mca.
out	<i>mca_banks</i>	number of mca banks.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.5 read\_bmc\_ras\_mca\_msr\_dump()

```
oob_status_t read_bmc_ras_mca_msr_dump (
    uint8_t soc_num,
    struct mca_bank mca_dump,
    uint32_t * out_buf )
```

Read data from mca bank reported by bmc ras mca validity check.

This function returns the data from mca bank reported by bmc ras mca validity check. Supported platforms↵: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>mca_dump</i>	<a href="#">mca_bank</a> Struct containing offset, index of MCA bank.
out	<i>out_buf</i>	32 bit data from offset in mca bank.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.6 read\_bmc\_ras\_fch\_reset\_reason()

```
oob_status_t read_bmc_ras_fch_reset_reason (
    uint8_t soc_num,
    uint32_t input,
    uint32_t * out_buf )
```

Read FCH reason code from the previous reset.

This function reads the FCH reason code from the previous reset. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>input</i>	integer for id of FCH register.
out	<i>out_buf</i>	Data from FCH register.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.7 read\_dimm\_temp\_range\_and\_refresh\_rate()

```
oob_status_t read_dimm_temp_range_and_refresh_rate (
    uint8_t soc_num,
    uint32_t dimm_addr,
    struct temp_refresh_rate * rate )
```

Read DIMM temperature range and refresh rate.

This function returns the per DIMM temperature range and refresh rate from the MR4 register, per JEDEC spec. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>dimm_addr</i>	Encoded address of the dimm.
out	<i>rate</i>	<a href="#">temp_refresh_rate</a> structure with refresh rate(1 bit) and range(3 bit). refresh rate: 0 = 1X, 1 = 2X. Temperature range: 001b = 1X, 101b = 2X.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.8 read\_dimm\_power\_consumption()

```
oob_status_t read_dimm_power_consumption (
    uint8_t soc_num,
    uint32_t dimm_addr,
    struct dimm_power * dimm_pow )
```

Read DIMM power consumption.

This function returns the DIMM power consumption when bmc does not own the SPD side band bus. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>dimm_addr</i>	Encoded address of the dimm.
out	<i>dimm_pow</i>	struct <a href="#">dimm_power</a> contains updatarate(ms): Time since last update (0-511ms). 0 means last update was < 1ms, and 511 means update was >= 511ms power consumption(mw): power consumed (0 - 32767 mW)

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.9 read\_dimm\_thermal\_sensor()

```
oob_status_t read_dimm_thermal_sensor (
    uint8_t soc_num,
    uint32_t dimm_addr,
    struct dimm_thermal * dimm_temp )
```

Read DIMM thermal sensor.

This function returns the DIMM thermal sensor (2 sensors per DIMM) when bmc does not own the SPD side band bus. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>dimm_addr</i>	Encoded address of the dimm.
out	<i>dimm_temp</i>	struct <a href="#">dimm_thermal</a> struct contains updatarate(ms): Time since last update (0-511ms). 0 means last update was < 1ms, and 511 means update was >= 511ms temperature (Degrees C): Temperature (-256 - 255.75 degree C)
Generated by Doxygen		

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.10 read\_pwr\_current\_active\_freq\_limit\_socket()

```
oob_status_t read_pwr_current_active_freq_limit_socket (
    uint8_t soc_num,
    uint16_t * freq,
    char ** source_type )
```

Read current active frequency limit per socket.

This function returns the current active frequency limit per socket. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>freq</i>	Frequency (MHz).
out	<i>source_type</i>	Source of limit.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.11 read\_pwr\_current\_active\_freq\_limit\_core()

```
oob_status_t read_pwr_current_active_freq_limit_core (
    uint8_t soc_num,
    uint32_t core_id,
    uint16_t * base_freq )
```

Read current active frequency limit set per core.

This function returns the current active frequency limit per core. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>core_id</i>	index.
out	<i>base_freq</i>	Frequency (MHz).

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.12 read\_pwr\_svi\_telemetry\_all\_rails()

```
oob_status_t read_pwr_svi_telemetry_all_rails (
    uint8_t soc_num,
    uint32_t * power )
```

Read SVR based telemtry for all rails.

This function returns the SVR based telemetry (power and update rate) for all rails. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>power</i>	SVI-based Telemetry for all rails(mW)

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.13 read\_socket\_freq\_range()

```
oob_status_t read_socket_freq_range (
    uint8_t soc_num,
    uint16_t * fmax,
    uint16_t * fmin )
```

Read socket frequency range.

This function returns the fmax and fmin frequency per socket. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>fmax</i>	maximum frequency (MHz).
out	<i>fmin</i>	minimum frequency (MHz).

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.14 read\_current\_io\_bandwidth()

```
oob_status_t read_current_io_bandwidth (
    uint8_t soc_num,
    struct link_id_bw_type link,
    uint32_t * io_bw )
```

Read current bandwidth on IO Link.

This function returns the current IO bandwidth. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>link</i>	<a href="#">link_id_bw_type</a> struct containing bandwidth type and Link ID encoding bandwidth type: 001b Aggregate BW Other Reserved MI300A APLM Link ID Encoding: 00000011b: P2 00000100b: P3 00001000b: G0 00001001b: G1 00001010b: G2 00001011b: G3 00001100b: G4 00001101b: G5 00001110b: G6 00001111b: G7 For other platforms the APLM Link ID Encoding: 00000001b: P0 00000010b: P1 00000100b: P2 00001000b: P3 00010000b: G0 00100000b: G1 01000000b: G2 10000000b: G3
out	<i>io_bw</i>	io bandwidth (Mbps).

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.15 read\_current\_xgmi\_bandwidth()

```
oob_status_t read_current_xgmi_bandwidth (
    uint8_t soc_num,
    struct link_id_bw_type link,
    uint32_t * xgmi_bw )
```

Read current bandwidth on xGMI Link.

This function returns the current xGMI bandwidth. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>link</i>	<a href="#">link_id_bw_type</a> struct containing link id and bandwidth type info. Valid BW type are 001b Aggregate BW 010b Read BW 100b Write BW Other Reserved MI300A APML Link ID Encoding: 00000011b: P2 00000100b: P3 00001000b: G0 00001001b: G1 00001010b: G2 00001011b: G3 00001100b: G4 00001101b: G5 00001110b: G6 00001111b: G7 For other platforms the APML Link ID Encoding: 00000001b: P0 00000010b: P1 00000100b: P2 00001000b: P3 00010000b: G0 00100000b: G1 01000000b: G2 10000000b: G3
out	<i>xgmi_bw</i>	io bandwidth (Mbps).

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.16 write\_gmi3\_link\_width\_range()

```
oob_status_t write_gmi3_link_width_range (
    uint8_t soc_num,
    uint8_t min_link_width,
    uint8_t max_link_width )
```

Set the max and min width of GMI3 link.

This function will set the max and min width of GMI3 Link. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>min_link_width</i>	minimum link width. 0 = Quarter width 1 = Half width 2 = full width
in	<i>max_link_width</i>	maximum link width. 0 = Quarter width 1 = Half width 2 = full width NOTE: max value must be greater than or equal to min value.

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.17 write\_xgmi\_link\_width\_range()

```
oob_status_t write_xgmi_link_width_range (
    uint8_t soc_num,
```

```
uint8_t min_link_width,
uint8_t max_link_width )
```

Set the max and min width of xGMI link.

This function will set the max and min width of xGMI Link. If this API is called from both the master and the slave sockets, then the largest width values from either calls are used. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>min_link_width</i>	minimum link width. 0 = X4 1 = X8 2 = X16
in	<i>max_link_width</i>	maximum link width. 0 = X4 1 = X8 2 = X16 NOTE: Max value must be greater than or equal to min value.

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.4.2.18 write\_apb\_disable()

```
oob_status_t write_apb_disable (
    uint8_t soc_num,
    uint8_t df_pstate,
    bool * prochot_asserted )
```

Set the APBDisabled.

This function will set the APBDisabled by specifying the Data Fabric(DF) P-state. Messages APBEnable and APBDisable specify DF(Data Fabric) P-state behavior. DF P-states specify the frequency of clock domains from the CPU core boundary through to and including system memory, where 0 is the highest DF P-state and 2 is the lowest. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>df_pstate</i>	data fabric p-state.
out	<i>prochot_asserted</i>	prochot asserted status. True indicates asserted False indicates not-asserted.

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.



#### 7.4.2.19 write\_apb\_enable()

```
oob_status_t write_apb_enable (
    uint8_t soc_num,
    bool * proshot_asserted )
```

Enable the DF p-state performance boost algorithm.

This function will enable the DF p-state performance boost algorithm. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index.
out	<i>proshot_asserted</i>	proshot asserted status. True indicates asserted and false indicates not-asserted.

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.4.2.20 read\_current\_dfpstate\_frequency()

```
oob_status_t read_current_dfpstate_frequency (
    uint8_t soc_num,
    struct pstate_freq * df_pstate )
```

Read current DF p-state frequency .

This function returns the current DF p-state frequency. Returns the Fclk, DRAM memory clock(memclk),umc clock divider for the current socket DF P-state. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index.
out	<i>df_pstate</i>	struct <a href="#">pstate_freq</a> contains DRAM memory clock(mem clk) data fabric clock (Fclk) UMC clock divider Uclk = 0 means divide by 1 else divide by 2.

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.4.2.21 write\_lclk\_dpm\_level\_range()

```
oob_status_t write_lclk_dpm_level_range (
    uint8_t soc_num,
    struct lclk_dpm_level_range lclk )
```

Set the max and min LCK DPM Level on a given NBIO per socket.

This function will set the LCK DPM Level on a given NBIO per socket. The DPM Level is an encoding to represent the PCIE Link Frequency (LCLK) under a root complex (NBIO). Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>lclk</i>	<a href="#">lclk_dpm_level_range</a> struct containing NBIOID (8 bit) Min dpm level (8 bit) and Max dpm level(8 bit). Valid NBIOID, min dpm level and max dpm level values are between 0 ~ 3.

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.4.2.22 read\_bmc\_rapl\_units()

```
oob_status_t read_bmc_rapl_units (
    uint8_t soc_num,
    uint8_t * tu_value,
    uint8_t * esu_value )
```

Read RAPL (Running Average Power Limit) Units.

This function returns the RAPL (Running Average Power Limit) Units. Energy information (in Joules) is based on the multiplier:  $1/(2^{\wedge}ESU)$ . Time information (in Seconds) is based on the multiplier:  $1/(2^{\wedge}TU)$ . Supported platforms↵: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index.
out	<i>tu_value</i>	TU value.
out	<i>esu_value</i>	esu value.

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.23 read\_bmc\_cpu\_base\_frequency()

```
oob_status_t read_bmc_cpu_base_frequency (
    uint8_t soc_num,
    uint16_t * base_freq )
```

Read RAPL base frequency per CPU socket.

This function returns the base frequency per CPU socket. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>base_freq</i>	base frequency.

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.24 read\_bmc\_control\_pcie\_gen5\_rate()

```
oob_status_t read_bmc_control_pcie_gen5_rate (
    uint8_t soc_num,
    uint8_t rate,
    uint8_t * mode )
```

Control PCIe Rate on Gen5-Capable devices..

This function returns the PCIe rate. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>rate</i>	PCIe gen rate. 0 indicates Auto-Detect BW and set link rate accordingly. 1 is for Limit at Gen4 Rate. 2 is for Limit at Gen5 rate.
out	<i>mode</i>	previous mode. 0 for Auto-Detect, 1 for Limit at Gen4 rate, 2 for limit at Gen5 rate.

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.4.2.25 read\_rapl\_core\_energy\_counters()

```
oob_status_t read_rapl_core_energy_counters (
    uint8_t soc_num,
    uint32_t core_id,
    double * energy_counters )
```

Read RAPL core energy counters.

This function returns the RAPL core energy counters. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

##### Parameters

in	<i>soc_num</i>	Soskcet index.
in	<i>core_id</i>	core id.
out	<i>energy_counters</i>	core energy.

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.4.2.26 read\_rapl\_pckg\_energy\_counters()

```
oob_status_t read_rapl_pckg_energy_counters (
    uint8_t soc_num,
    double * energy_counters )
```

Read RAPL package energy counters.

This function returns the RAPL package energy counters. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index.
out	<i>energy_counters</i>	core energy.

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.4.2.27 write\_pwr\_efficiency\_mode()

```
oob_status_t write_pwr_efficiency_mode (
    uint8_t soc_num,
    uint8_t mode )
```

Write power efficiency profile policy.

This function writes power efficiency mode Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>mode</i>	power efficiency mode. 0 indicates High Performance Mode 1 indicates Power Efficiency Mode. 2 indicates I/O Performance Mode. 3 Balanced Memory Performance Mode. 4 Balanced Core Performance Mode. 5 Balanced Core and Memory Performance Mode.

Modes supported as per platform, 0 to 2: Fam-19h\_Mod-10h-1Fh Fam-19h\_Mod-90h-9Fh 0 to 5: Fam-1Ah\_Mod-00h-0Fh

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.4.2.28 write\_df\_pstate\_range()

```
oob_status_t write_df_pstate_range (
    uint8_t soc_num,
    uint8_t max_pstate,
    uint8_t min_pstate )
```

Write df pstate range.

This function writes df pstate range Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#)

##### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>max_pstate</i>	value.Max value must be less than or equal to min value. Valid values are 0 - 2.
in	<i>min_pstate</i>	value. Valid values are from 0 - 2.

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.4.2.29 read\_lclk\_dpm\_level\_range()

```
oob_status_t read_lclk_dpm_level_range (
    uint8_t soc_num,
    uint8_t nbio_id,
    struct dpm_level * dpm )
```

Read LCLK Max and Min DPM level range.

This function returns the LCLK Max and Min DPM level range. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>nbio_id</i>	nbio for a socket.
out	<i>dpm</i>	struct dpm level containing max and min dpm levels. Valid max and min dpm levels are from 0 - 1.

##### Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.4.2.30 read\_ucode\_revision()

```
oob_status_t read_ucode_revision (
    uint8_t soc_num,
    uint32_t * ucode_rev )
```

Read ucode revision.

This function reads the micro code revision number. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index.
out	<i>ucode_rev</i>	micro code revision.

##### Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.4.2.31 reset\_on\_sync\_flood()

```
oob_status_t reset_on_sync_flood (
    uint8_t soc_num,
    uint32_t * ack_resp )
```

Read BMC RAS reset on sync flood.

This function requests reset after sync flood. Reset only works during sync flood condition Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index. At present, only P0 handles this request.
out	<i>ack_resp</i>	acknowledgement response.

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.4.2.32 override\_delay\_reset\_on\_sync\_flood()

```
oob_status_t override_delay_reset_on_sync_flood (
    uint8_t soc_num,
    struct ras_override_delay data_in,
    bool * ack_resp )
```

Overrides delay reset cpu on sync flood value.

This function will override delay reset cpu on sync flood value for the current boot instance. Delay value reverts to BIOS config selection after reboot. Number of override requests is limited to 5 per boot instance. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#).

##### Parameters

in	<i>soc_num</i>	Socket index. At present, only P0 handles this request.
in	<i>data_in</i>	struct ras_override_delay_d_in containing delay value override [5 - 120 mins], disable delay counter bit and stop delay counter bit.
out	<i>ack_resp</i>	acknowledgment response.

##### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 7.4.2.33 get\_post\_code()

```
oob_status_t get_post_code (
    uint8_t soc_num,
    uint32_t offset,
    uint32_t * post_code )
```

Read post code.

This function will read most recent post code at the specified offset. SMU caches 8 most recent post codes.↵ When the input is 0 the SMU will refresh the cache before running the latest post code. Input as 0 refers to most recent post code and higher inputs refers to the older post code. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>offset</i>	post code offset
out	<i>post_code</i>	recent post code of error category and number of instances of error category.

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 7.4.2.34 get\_bmc\_ras\_run\_time\_err\_validity\_ck()

```
oob_status_t get_bmc_ras_run_time_err_validity_ck (
    uint8_t soc_num,
    struct ras_rt_err_req_type err_category,
    struct ras_rt_valid_err_inst * inst )
```

Reads number of valid error instances per category.

This function will read number of valid error instances per category. Valid categories are MCA[00], DRAM CECC[01], PCIE [10], RSVD[11]. Supported platforms: [Fam-1Ah\\_Mod-00h-0Fh](#)

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>err_category</i>	Runtime error category MCA[00], DRAM CECC[01], PCIe[10] and RSVD [11].
out	<i>inst</i>	struct <a href="#">ras_rt_valid_err_inst</a> containing number of bytes of error category and number of instances of error category.



## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.35 get\_bmc\_ras\_run\_time\_error\_info()

```
oob_status_t get_bmc_ras_run_time_error_info (
    uint8_t soc_num,
    struct run_time_err_d_in d_in,
    uint32_t * err_info )
```

Reads BMC RAS runtime error information.

This function will read BMC RAS runtime error information based on category. If category is MCA then it will read 32 bit of data from MCA MSR Bank. If category is DRAM CECC then will read error count, counter info and corresponding from the valid DRAM ECC correctable error counter instance. If category is PCIE then it returns 32 bit PCIE error data from given offset of given instance. Supported platforms: [Fam-1Ah\\_Mod-00h-0Fh](#)

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>d_in</i>	struct <a href="#">run_time_err_d_in</a> containing 4 byte aligned offset, runtime error category and 0 based index of valid instance returned by BMC_RAS_RUNTIME_ERR_VALIDITY_CHECK.
out	<i>err_info</i>	error information for a given category with valid instance and offset.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.36 set\_bmc\_ras\_err\_threshold()

```
oob_status_t set_bmc_ras_err_threshold (
    uint8_t soc_num,
    struct run_time_threshold th )
```

Sets BMC RAS error threshold.

This function will configure thresholding counters for MCA, DRAM CECC or PCIE. Supported platforms: [Fam-1Ah\\_Mod-00h-0Fh](#)

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>th</i>	struct <a href="#">run_time_threshold</a> containing error type [00(MCA), 01(DRAM CECC), 10(PCIE), error count threshold and max interrupt rate.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.37 set\_bmc\_ras\_oob\_config()

```
oob_status_t set_bmc_ras_oob_config (
    uint8_t soc_num,
    struct oob_config_d_in d_in )
```

Configures OOB state infrastructure in SoC.

This function will configure OOB state infrastructure in the SoC. Supported platforms: [Fam-1Ah\\_Mod-00h-0Fh](#)

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>d_in</i>	struct <a href="#">oob_config_d_in</a> containing [0] mca_oob_misc0_ec_enable, [2:1] dram_cecc_oob_ec_mode, [7:3] dram_cecc_leak_rate, [8] pcie_err_reporting_en, [31] core_mca_err_reporting_en.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.38 get\_bmc\_ras\_oob\_config()

```
oob_status_t get_bmc_ras_oob_config (
    uint8_t soc_num,
    uint32_t * oob_config )
```

Reads the current status of OOB state infrastructure in SoC.

This function will read the current status of OOB state configuration in the SoC. Supported platforms: [Fam-1Ah\\_Mod-00h-0Fh](#)

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>oob_config</i>	containing [0] mca_oob_misc0_ec_enable, [2:1] dram_cecc_oob_ec_mode, [7:3] dram_cecc_leak_rate, [8] pcie_err_reporting_en, [11] MCA Thresholding Interrupt Enable value [12] DRAM CECC Thresholding Interrupt Enable value [13] PCIe error Thresholding Interrupt Enable value [18:15] MCA MaxIntRate value [22:19] DRAM CECC MaxIntRate1 value [26:23] PCIe error MaxIntRate1 value [31] core_mca_err_reporting_en

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.4.2.39 read\_ppin\_fuse()

```
oob_status_t read_ppin_fuse (
    uint8_t soc_num,
    uint64_t * data )
```

Get the 64 bit PPIN fuse.

This function will read the 64 bit PPIN fuse available via OPN\_PPIN fuse. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>data</i>	PPIN fuse data

## 7.4.2.40 read\_ras\_df\_err\_validity\_check()

```
oob_status_t read_ras_df_err_validity_check (
    uint8_t soc_num,
    uint8_t df_block_id,
    struct ras_df_err_chk * err_chk )
```

Read number of instances of DF blocks of type DF\_BLOCK\_ID with errors.

This function reads the number instance of DF blocks of type DF\_BLOCK\_ID that have an error log to report. Supported platforms: [Fam-19h\\_Mod-10h-1Fh](#), [Fam-1Ah\\_Mod-00h-0Fh](#) and [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>df_block_id</i>	DF block ID.
out	<i>err_chk</i>	<a href="#">ras_df_err_chk</a> Struct containing number of DF block instances and length of error log in bytes per instance.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.4.2.41 read\_ras\_df\_err\_dump()

```
oob_status_t read_ras_df_err_dump (
    uint8_t soc_num,
    union ras_df_err_dump ras_err,
    uint32_t * data )
```

Read RAS DF error dump.

This function reads 32 bits of data from the offset provided for DF block instance reported by read\_ras\_df\_err\_validity\_check. Supported platforms: Fam-19h\_Mod-10h-1Fh, Fam-1Ah\_Mod-00h-0Fh and Fam-19h\_Mod-90h-9Fh.

##### Parameters

in	soc_num	Socket index.
in	ras_err	ras_df_err_dump Union containing 4 byte offset, DF block ID and block ID instance.
out	data	output data from offset of DF block instance.

##### Return values

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

#### 7.4.2.42 read\_rtc()

```
oob_status_t read_rtc (
    uint8_t soc_num,
    uint64_t * rtc )
```

Read RTC.

Read RTC timer value. RTC time represents the year, month, day, hour, minute and seconds value in a 64b encoding. Supported platforms: Fam-1Ah\_Mod-00h-0Fh

##### Parameters

in	soc_num	Socket index.
out	rtc	[63:0] = RTC value, [31:0]=> DataIn==0 & [63:32]=> DataIn==4 If DataIn==0 RTC=DD_hh_mm_ss If DataIn==4 RTC=00_YYYY_MM All digits in BCD format.

##### Return values

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

## 7.4.2.43 read\_dimm\_spd\_register()

```
oob_status_t read_dimm_spd_register (
    uint8_t soc_num,
    struct dimm_spd_d_in spd_d_in,
    uint32_t * spd_data )
```

Get SPD SB data.

Supported platforms: [Fam-1Ah\\_Mod-00h-0Fh](#)

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>spd_d_in</i>	struct <a href="#">dimm_spd_d_in</a> containing DIMM address, LID, register offset, register space.
out	<i>spd_data</i>	containing spd data as per input

## 7.4.2.44 get\_dimm\_serial\_num()

```
oob_status_t get_dimm_serial_num (
    uint8_t soc_num,
    uint8_t dimm_addr,
    uint32_t * serial_num )
```

Get DIMM serial number.

Supported platforms: [Fam-1Ah\\_Mod-00h-0Fh](#)

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>dimm_addr</i>	DIMM address
out	<i>serial_num</i>	containing DIMM serial number

## 7.5 esmi\_mailbox\_nda.h File Reference

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

## Data Structures

- struct [status\\_query\\_payload](#)

- *data for status query payload. Members contain repair entry number.*
- struct [get\\_ras\\_action\\_data\\_in](#)  
*data input for the get BMC RAS action status. Members contain repair entry number and ras action id.*
- struct [ras\\_action\\_status](#)  
*Output data for the get BMC RAS action status. Members contain repair result, repair entry number and ras action id.*
- struct [action\\_payload](#)  
*data for the action payload. Members contain payload, offset, repair entry number.*
- struct [set\\_ras\\_action\\_data\\_in](#)  
*data input for set ras action status. Members contain struct [action\\_payload](#), ras action ID and eom flag.*

## Enumerations

- enum [esb\\_mailbox\\_nda\\_commands](#) { [SET\\_BMC\\_RAS\\_ACTION](#) = 0x66, [GET\\_BMC\\_RAS\\_ACTION\\_STATUS](#) }
- *Mailbox message IDs defined in the APML library (NDA)*

## Functions

- [oob\\_status\\_t set\\_bmc\\_ras\\_action\\_status](#) (uint8\_t soc\_num, struct [set\\_ras\\_action\\_data\\_in](#) data\_in, uint32\_t \*status)  
*Set BMC RAS action status.*
- [oob\\_status\\_t get\\_bmc\\_ras\\_action\\_status](#) (uint8\_t soc\_num, struct [get\\_ras\\_action\\_data\\_in](#) data\_in, struct [ras\\_action\\_status](#) \*status)  
*Get BMC RAS action status.*

### 7.5.1 Detailed Description

Header file for the NDA Mailbox messages supported by APML library. All required function, structure, enum, etc. definitions should be defined in this file.

This header file contains the following: NDA APIs prototype of the Mailbox messages exported by the APML library. Description of the API, arguments and return values. The Error codes returned by the API.

### 7.5.2 Function Documentation

#### 7.5.2.1 set\_bmc\_ras\_action\_status()

```
oob_status_t set_bmc_ras_action_status (
    uint8_t soc_num,
    struct set_ras_action_data_in data_in,
    uint32_t * status )
```

Set BMC RAS action status.

This function will set BMC RAS action status for the specified action ID. Supported platforms: [Fam-1Ah\\_Mod-00h-0Fh](#)

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>data_in</i>	struct <a href="#">set_ras_action_data_in</a> containing struct <a href="#">action_payload</a> , ras action ID and eom flag.
out	<i>status</i>	response for ras action.

## 7.5.2.2 get\_bmc\_ras\_action\_status()

```
oob_status_t get_bmc_ras_action_status (
    uint8_t soc_num,
    struct get_ras_action_data_in data_in,
    struct ras_action_status * status )
```

Get BMC RAS action status.

This function will read BMC RAS action status for the specified action ID. Supported platforms: [Fam-1Ah\\_Mod-00h-0Fh](#)

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>data_in</i>	struct <a href="#">get_ras_action_data_in</a> containing struct <a href="#">status_query_payload</a> and ras action id.
out	<i>status</i>	struct <a href="#">ras_action_status</a> containing repair result and struct <a href="#">get_ras_action_data_in</a> .

## 7.6 esmi\_rmi.h File Reference

```
#include "apml_err.h"
```

## Macros

- #define [MAX\\_ALERT\\_REG](#) 32  
*Max alert register //.*
- #define [MAX\\_THREAD\\_REG\\_V20](#) 32  
*Max thread register for rev 20 //.*
- #define [MAX\\_THREAD\\_REG\\_V10](#) 16  
*Max thread register for rev 10 //.*

## Enumerations

- enum [sbrmi\\_status\\_code](#) {  
**SBRMI\_SUCCESS** = 0x0, **SBRMI\_CMD\_TIMEOUT** = 0x11, **SBRMI\_WARM\_RESET** = 0x22, **SBRMI\_UNKNOWN\_CMD\_FORMAT** = 0x40,  
**SBRMI\_INVALID\_READ\_LENGTH** = 0x41, **SBRMI\_EXCESSIVE\_DATA\_LENGTH** = 0x42, **SBRMI\_INVALID\_THREAD** = 0x44, **SBRMI\_UNSUPPORTED\_CMD** = 0x45,  
**SBRMI\_CMD\_ABORTED** = 0x81 }

Error codes returned by APML mailbox functions.

- enum [sbrmi\\_registers](#) {  
**SBRMI\_REVISION** = 0x0, **SBRMI\_CONTROL**, **SBRMI\_STATUS**, **SBRMI\_READSIZE**,  
**SBRMI\_THREADENABLESTATUS0**, **SBRMI\_ALERTSTATUS0** = 0x10, **SBRMI\_ALERTSTATUS15** =  
0x1F, **SBRMI\_ALERTMASK0** = 0x20,  
**SBRMI\_ALERTMASK15** = 0x2F, **SBRMI\_SOFTWAREINTERRUPT** = 0x40, **SBRMI\_THREADNUMBER**,  
**SBRMI\_THREAD128CS** = 0x4B,  
**SBRMI\_RASSTATUS**, **SBRMI\_THREADNUMBERLOW** = 0x4E, **SBRMI\_THREADNUMBERHIGH** = 0x4F,  
**SBRMI\_ALERTSTATUS16** = 0x50,  
**SBRMI\_ALERTSTATUS31** = 0x5F, **SBRMI\_MP0OUTBNDMSG0** = 0x80, **SBRMI\_MP0OUTBNDMSG7** =  
0x87, **SBRMI\_ALERTMASK16** = 0xC0,  
**SBRMI\_ALERTMASK31** = 0xCF }

SB-RMI(Side-Band Remote Management Interface) features register access.

## Functions

- [oob\\_status\\_t read\\_sbrmi\\_revision](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
Read revision from SB\_RMI register command. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_control](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
Read Control byte from SB\_RMI register command. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_status](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
Read one byte of Status value from SB\_RMI register command. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_readsize](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
This register specifies the number of bytes to return when using the block read protocol to read SBRMI↔\_x[4F:10]. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_threadenablestatus](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
Read one byte of Thread Status from SB\_RMI register command. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_multithreadenablestatus](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
Read one byte of Thread Status from SB\_RMI register command. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_swinterrupt](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
This register is used by the SMBus master to generate an interrupt to the processor to indicate that a message is available.. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_threadnumber](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
This register indicates the maximum number of threads present. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_mp0\\_msg](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
This register will read the message running on the MP0. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_alert\\_status](#) (uint8\_t soc\_num, uint8\_t num\_of\_alert\_status\_reg, uint8\_t \*\*buffer)  
This function will read bit vector for all the threads. Value of 1 indicates MCE occurred for the thread and is set by hardware. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_alert\\_mask](#) (uint8\_t soc\_num, uint8\_t num\_of\_alert\_mask\_reg, uint8\_t \*\*buffer)  
This function will read bit vector for all the threads. Value of 1 indicates alert signaling disabled for corresponding SBRMI::AlertStatus[MceStat] for the thread. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbrmi\\_inbound\\_msg](#) (uint8\_t soc\_num, uint8\_t \*buffer)



*This register will read the inbound message. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*

- `oob_status_t read_sbrmi_outbound_msg` (uint8\_t soc\_num, uint8\_t \*buffer)

*This register will read the outbound message. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*

- `oob_status_t read_sbrmi_threadnumberlow` (uint8\_t soc\_num, uint8\_t \*buffer)

*This register indicates the low part of maximum number of threads. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*

- `oob_status_t read_sbrmi_threadnumberhi` (uint8\_t soc\_num, uint8\_t \*buffer)

*This register indicates the upper part of maximum number of threads. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*

- `oob_status_t read_sbrmi_thread_cs` (uint8\_t soc\_num, uint8\_t \*buffer)

*This register is used to read the thread cs. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*

- `oob_status_t read_sbrmi_ras_status` (uint8\_t soc\_num, uint8\_t \*buffer)

*This register will read the ras status. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*

- `oob_status_t clear_sbrmi_ras_status` (uint8\_t soc\_num, uint8\_t \*buffer)

*This API will clear ras status register. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*

- `oob_status_t esmi_get_threads_per_socket` (uint8\_t soc\_num, uint32\_t \*threads\_per\_socket)

*Get the number of threads per socket.*

## Variables

- `const uint8_t thread_en_reg_v10` [MAX\_THREAD\_REG\_V10]

*thread enable register revision 0x10*

- `const uint8_t thread_en_reg_v20` [MAX\_THREAD\_REG\_V20]

*thread enable register revision 0x20*

- `const uint8_t alert_status` [MAX\_ALERT\_REG]

*alert status register*

- `const uint8_t alert_mask` [MAX\_ALERT\_REG]

*alert mask*

### 7.6.1 Detailed Description

Header file for the APML library for SB-RMI functionality access. All required function, structure, enum, etc. definitions should be defined in this file for SB-RMI Register accessing.

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

## 7.7 esmi\_tsi.h File Reference

```
#include "apml_err.h"
```

## Macros

- `#define TEMP_INC 0.125`

*Register encode the temperature to increase in 0.125 In decimal portion one increase in byte is equivalent to 0.125.*

## Enumerations

- enum `sbtsi_registers` {  
**SBTSI\_CPUTEMPINT** = 0x1, **SBTSI\_STATUS**, **SBTSI\_CONFIGURATION**, **SBTSI\_UPDATERATE**,  
**SBTSI\_HIEMPINT** = 0x7, **SBTSI\_LOTEMPINT**, **SBTSI\_CONFIGWR**, **SBTSI\_CPUTEMPDEC** = 0x10,  
**SBTSI\_CPUTEMPOFFINT**, **SBTSI\_CPUTEMPOFFDEC**, **SBTSI\_HIEMPDEC**, **SBTSI\_LOTEMPDEC**,  
**SBTSI\_TIMEOUTCONFIG** = 0x22, **SBTSI\_ALERTTHRESHOLD** = 0x32, **SBTSI\_ALERTCONFIG** = 0xBF,  
**SBTSI\_MANUFID** = 0xFE,  
**SBTSI\_REVISION** = 0xFF }

*SB-TSI(Side-Band Temperature Sensor Interface) commands register access. The below registers mentioned as per Genesis PPR.*

- enum `sbtsi_config_write` { **ARA\_MASK** = 0x2, **READORDER\_MASK** = 0x20, **RUNSTOP\_MASK** = 0x40,  
**ALERTMASK\_MASK** = 0x80 }

*Bitfield values to be set for SBTSI confirwr register [7] Alert mask [6] RunStop [5] ReadOrder [1] AraDis.*

## Functions

- `oob_status_t read_sbtsi_cpuinttemp` (uint8\_t soc\_num, uint8\_t \*buffer)  
*integer CPU temperature value The CPU temperature is calculated by adding the CPU temperature offset(S↔BTSL::CpuTempOffInt, SBTSI::CpuTempOffDec) to the processor control temperature (Tctl). SBTSI::CpuTempInt and SBTSI::CpuTempDec combine to return the CPU temperature. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- `oob_status_t read_sbtsi_status` (uint8\_t soc\_num, uint8\_t \*buffer)  
*Status register is Read-only, volatile field If SBTSI::AlertConfig[AlertCompEn] == 0 , the temperature alert is latched high until the alert is read. If SBTSI::AlertConfig[AlertCompEn] == 1, the alert is cleared when the temperature does not meet the threshold conditions for temperature and number of samples. Supported platforms↔: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- `oob_status_t read_sbtsi_config` (uint8\_t soc\_num, uint8\_t \*buffer)  
*The bits in this register are Read-only and can be written by Writing to the corresponding bits in SBTSI↔::ConfigWr Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- `oob_status_t read_sbtsi_updaterate` (uint8\_t soc\_num, float \*buffer)  
*This register value specifies the rate at which CPU temperature is compared against the temperature thresholds to determine if an alert event has occurred. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- `oob_status_t write_sbtsi_updaterate` (uint8\_t soc\_num, float uprate)  
*This register value specifies the rate at which CPU temperature is compared against the temperature thresholds to determine if an alert event has occurred. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- `oob_status_t read_sbtsi_hitempint` (uint8\_t soc\_num, uint8\_t \*buffer)  
*This value specifies the integer portion of the high temperature threshold. The high temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is greater than or equal to the threshold. Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*
- `oob_status_t read_sbtsi_lotempint` (uint8\_t soc\_num, uint8\_t \*buffer)  
*This value specifies the integer portion of the low temperature threshold. The low temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is less than or equal to the threshold Supported platforms: Fam-19h\_Mod-00h-0Fh, Fam-19h\_Mod-10h-1Fh, Fam-19h\_Mod-90h-9Fh and Fam-1Ah\_Mod-00h-0Fh.*

- [oob\\_status\\_t read\\_sbtsi\\_configwrite](#) (uint8\_t soc\_num, uint8\_t \*buffer)  
This register provides write access to SBTSI::Config Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_cputempdecimal](#) (uint8\_t soc\_num, float \*buffer)  
The value returns the decimal portion of the CPU temperature Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_cputempoffint](#) (uint8\_t soc\_num, uint8\_t \*temp\_int)  
SBTSI::CpuTempOffInt and SBTSI::CpuTempOffDec combine to specify the CPU temperature offset Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_cputempoffdec](#) (uint8\_t soc\_num, float \*temp\_dec)  
This value specifies the decimal/fractional portion of the CPU temperature offset added to Tctl to calculate the CPU temperature. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_hitempdecimal](#) (uint8\_t soc\_num, float \*temp\_dec)  
This value specifies the decimal portion of the high temperature threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_lotempdecimal](#) (uint8\_t soc\_num, float \*temp\_dec)  
value specifies the decimal portion of the low temperature threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_timeoutconfig](#) (uint8\_t soc\_num, uint8\_t \*timeout)  
value specifies 0=SMBus defined timeout support disabled. 1=SMBus defined timeout support enabled. SMBus timeout enable. If SB-RMI is in use, SMBus timeouts should be enabled or disabled in a consistent manner on both interfaces. SMBus defined timeouts are not disabled for SB-RMI when this bit is set to 0. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_alertthreshold](#) (uint8\_t soc\_num, uint8\_t \*samples)  
Specifies the number of consecutive CPU temperature samples for which a temperature alert condition needs to remain valid before the corresponding alert bit is set. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_alertconfig](#) (uint8\_t soc\_num, uint8\_t \*mode)  
Status register is Read-only, volatile field If SBTSI::AlertConfig[AlertCompEn] == 0, the temperature alert is latched high until the alert is read. If SBTSI::AlertConfig[AlertCompEn] == 1, the alert is cleared when the temperature does not meet the threshold conditions for temperature and number of samples. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_manufid](#) (uint8\_t soc\_num, uint8\_t \*man\_id)  
Returns the AMD manufacture ID Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t read\\_sbtsi\\_revision](#) (uint8\_t soc\_num, uint8\_t \*revision)  
Specifies the SBI temperature sensor interface revision Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t sbtsi\\_get\\_cputemp](#) (uint8\_t soc\_num, float \*cpu\_temp)  
CPU temperature value The CPU temperature is calculated by adding SBTSI::CpuTempInt and SBTSI::CpuTempDec combine to return the CPU temperature. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t sbtsi\\_get\\_temp\\_status](#) (uint8\_t soc\_num, uint8\_t \*loalert, uint8\_t \*hialert)  
Status register is Read-only, volatile field If SBTSI::AlertConfig[AlertCompEn] == 0, the temperature alert is latched high until the alert is read. If SBTSI::AlertConfig[AlertCompEn] == 1, the alert is cleared when the temperature does not meet the threshold conditions for temperature and number of samples. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t sbtsi\\_get\\_config](#) (uint8\_t soc\_num, uint8\_t \*al\_mask, uint8\_t \*run\_stop, uint8\_t \*read\_ord, uint8\_t \*ara)  
The bits in this register are Read-only and can be written by Writing to the corresponding bits in SBTSI::ConfigWr Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).
- [oob\\_status\\_t sbtsi\\_set\\_configwr](#) (uint8\_t soc\_num, uint8\_t mode, uint8\_t config\_mask)  
The bits in this register are defined sbtsi\_config\_write and can be written by writing to the corresponding bits in SBTSI::ConfigWr Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).

- [oob\\_status\\_t sbtsi\\_get\\_timeout](#) (uint8\_t soc\_num, uint8\_t \*timeout\_en)  
*To verify if timeout support enabled or disabled Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).*
- [oob\\_status\\_t sbtsi\\_set\\_timeout\\_config](#) (uint8\_t soc\_num, uint8\_t mode)  
*To enable/disable timeout support Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).*
- [oob\\_status\\_t sbtsi\\_set\\_hitemp\\_threshold](#) (uint8\_t soc\_num, float hitemp\_thr)  
*This value set the high temperature threshold. The high temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is greater than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).*
- [oob\\_status\\_t sbtsi\\_set\\_lotemp\\_threshold](#) (uint8\_t soc\_num, float lotemp\_thr)  
*This value set the low temperature threshold. The low temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is less than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).*
- [oob\\_status\\_t sbtsi\\_get\\_hitemp\\_threshold](#) (uint8\_t soc\_num, float \*hitemp\_thr)  
*This value specifies the high temperature threshold. The high temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is greater than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).*
- [oob\\_status\\_t sbtsi\\_get\\_lotemp\\_threshold](#) (uint8\_t soc\_num, float \*lotemp\_thr)  
*This value specifies the low temperature threshold. The low temperature threshold specifies the CPU temperature that causes ALERT\_L to assert if the CPU temperature is less than or equal to the threshold. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).*
- [oob\\_status\\_t read\\_sbtsi\\_cputempoffset](#) (uint8\_t soc\_num, float \*temp\_offset)  
*SBTSI::CpuTempOffInt and SBTSI::CpuTempOffDec combine to specify the CPU temperature offset Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).*
- [oob\\_status\\_t write\\_sbtsi\\_cputempoffset](#) (uint8\_t soc\_num, float temp\_offset)  
*SBTSI::CpuTempOffInt and SBTSI::CpuTempOffDec combine to set the CPU temperature offset Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).*
- [oob\\_status\\_t sbtsi\\_set\\_alert\\_threshold](#) (uint8\_t soc\_num, uint8\_t samples)  
*Specifies the number of consecutive CPU temperature samples for which a temperature alert condition needs to remain valid before the corresponding alert bit is set. Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).*
- [oob\\_status\\_t sbtsi\\_set\\_alert\\_config](#) (uint8\_t soc\_num, uint8\_t mode)  
*Alert comparator mode enable Supported platforms: [Fam-19h\\_Mod-00h-0Fh](#), [Fam-19h\\_Mod-10h-1Fh](#), [Fam-19h\\_Mod-90h-9Fh](#) and [Fam-1Ah\\_Mod-00h-0Fh](#).*

### 7.7.1 Detailed Description

Header file for the APML library for SB-TSI functionality access. All required function, structure, enum, etc. definitions should be defined in this file for SB-TSI Register accessing.

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

## 7.8 rmi\_mailbox\_mi300.h File Reference

```
#include "apml_common.h"
#include "apml_err.h"
```

## Data Structures

- struct [max\\_mem\\_bw](#)  
*Structure for Max DDR/HBM bandwidth and utilization. It contains max bandwidth(16 bit data) in GBps, current utilization bandwidth(Read+Write)(16 bit data) in GBps.*
- struct [svi\\_port\\_domain](#)  
*struct containing port and slave address.*
- struct [freq\\_limits](#)  
*struct containing max frequency and min frequency limit*
- struct [mclk\\_fclk\\_pstates](#)  
*struct containing memory clock and fabric clock pstate mappings.*
- struct [statistics](#)  
*struct containing statistics parameter of interest and output control pstate mappings.*
- struct [xgmi\\_speed\\_rate\\_n\\_width](#)  
*struct containing xgmi speed rate in MHZ and link width in units of Gpbs. If link\_width[0] = 1 then XGMI link X2 is supported. If link\_width[1] = 1 then XGMI link X4 is supported. If Link\_width[2] = 1 then XGMI link X4 is supported and similarly if link\_width[3] = 1 then XGMI link X8 is supported.*
- struct [hbm\\_device\\_info](#)  
*struct containing device vendor, part number and total memory size in GBs.*
- struct [host\\_status](#)  
*struct containing power management controlling status, driving running status.*

## Macros

- #define [MI300A\\_ENCODING\\_SIZE](#) 10  
*MI300A encoding size //.*

## Enumerations

- enum [esb\\_mi300\\_mailbox\\_commands](#) {  
**SET\_MAX\_GFX\_CORE\_CLOCK** = 0x81, **SET\_MIN\_GFX\_CORE\_CLOCK**, **SET\_MAX\_PSTATE**, **GET\_P**↵  
**STATES**,  
**GET\_CURR\_XGMI\_PSTATE**, **SET\_XGMI\_PSTATE**, **UNSET\_XGMI\_PSTATE**, **GET\_XGMI\_PSTATES**,  
**GET\_XCC\_IDLE\_RESIDENCY**, **GET\_ENERGY\_ACCUMULATOR** = 0x90, **GET\_RAS\_ALARMS**, **GET**↵  
**PM\_ALARMS**,  
**GET\_PSN**, **GET\_LINK\_INFO**, **GET\_ABS\_MAX\_MIN\_GFX\_FREQ** = 0x96, **GET\_SVI\_TELEMETRY\_BY**↵  
**RAIL**,  
**GET\_DIE\_TYPE**, **GET\_ACT\_GFX\_FREQ\_CAP\_SELECTED** = 0x9c, **GET\_DIE\_HOT\_SPOT\_INFO** = 0xA0,  
**GET\_MEM\_HOT\_SPOT\_INFO**,  
**GET\_MAX\_OP\_TEMP**, **GET\_SLOW\_DOWN\_TEMP**, **GET\_STATUS** = 0xA4, **GET\_MAX\_MEM\_BW\_UTI**↵  
**LIZATION** = 0xB0,  
**GET\_HBM\_THROTTLE**, **SET\_HBM\_THROTTLE**, **GET\_HBM\_STACK\_TEMP**, **GET\_GFX\_CLK\_FREQ**↵  
**LIMITS**,  
**GET\_FCLK\_FREQ\_LIMITS**, **GET\_SOCKETS\_IN\_SYSTEM**, **GET\_HBM\_DEVICE\_INFO**, **GET\_PCIE\_ST**↵  
**ATS** = 0xBA,  
**GET\_BIST\_RESULTS** = 0xBC, **QUERY\_STATISTICS**, **CLEAR\_STATISTICS** }  
*Mailbox message types defined in the APLM library.*
- enum [range\\_type](#) { **MIN** = 0, **MAX** }  
*APLM range type used by GFX core clock frequency. Max, MIN are the values. Min is 0 and Max is 1.*
- enum [clk\\_type](#) { **GFX\_CLK** = 0, **F\_CLK** }  
*APLM clock frequency type. GFX\_CLK or F\_CLK GFX\_CLK value is 0 and F\_CLK value is 1.*
- enum [alarms\\_type](#) { **PM** }  
*APLM alarms type. PM\_ALARMS. PM is 0.*

## Functions

- [oob\\_status\\_t set\\_gfx\\_core\\_clock](#) (uint8\_t soc\_num, enum [range\\_type](#) freq\_type, uint32\_t freq)  
*Set maximum/minimum gfx core clock frequency.*
- [oob\\_status\\_t set\\_mclk\\_fclk\\_max\\_pstate](#) (uint8\_t soc\_num, uint32\_t pstate)  
*Set maximum mem and fclk pstate.*
- [oob\\_status\\_t get\\_mclk\\_fclk\\_pstates](#) (uint8\_t soc\_num, uint8\_t pstate\_ind, struct [mclk\\_fclk\\_pstates](#) \*pstate)  
*Get memory and fabric clock power state mappings.*
- [oob\\_status\\_t set\\_xgmi\\_pstate](#) (uint8\_t soc\_num, uint32\_t pstate)  
*Sets the XGMI Pstate.*
- [oob\\_status\\_t unset\\_xgmi\\_pstate](#) (uint8\_t soc\_num)  
*Resets the XGMI Pstate.*
- [oob\\_status\\_t get\\_xgmi\\_pstates](#) (uint8\_t soc\_num, uint8\_t pstate\_ind, struct [xgmi\\_speed\\_rate\\_n\\_width](#) \*xgmi\_pstate)  
*Read XGMI power state mappings.*
- [oob\\_status\\_t get\\_xcc\\_idle\\_residency](#) (uint8\_t soc\_num, uint32\_t \*gfx\_cores\_idle\_res)  
*Read xcc idle residency percentage.*
- [oob\\_status\\_t get\\_energy\\_accum\\_with\\_timestamp](#) (uint8\_t soc\_num, uint64\_t \*energy, uint64\_t \*time\_stamp)  
*Read energy accumulator with time stamp.*
- [oob\\_status\\_t get\\_alarms](#) (uint8\_t soc\_num, enum [alarms\\_type](#) type, uint32\_t \*buffer)  
*Read PM alarm status based on enumeration type [alarms\\_type](#).*
- [oob\\_status\\_t get\\_psn](#) (uint8\_t soc\_num, uint32\_t die\_index, uint64\_t \*buffer)  
*Reads public serial number (PSN).*
- [oob\\_status\\_t get\\_link\\_info](#) (uint8\_t soc\_num, uint8\_t \*link\_config, uint8\_t \*module\_id)  
*Read link Info.*
- [oob\\_status\\_t get\\_max\\_min\\_gfx\\_freq](#) (uint8\_t soc\_num, uint16\_t \*max\_freq, uint16\_t \*min\_freq)  
*Read maximum and minimum allowed GFX engine frequency.*
- [oob\\_status\\_t get\\_act\\_gfx\\_freq\\_cap](#) (uint8\_t soc\_num, uint16\_t \*freq)  
*Read Actual GFX frequency cap selected.*
- [oob\\_status\\_t get\\_svi\\_rail\\_telemetry](#) (uint8\_t soc\_num, struct [svi\\_port\\_domain](#) port, uint32\_t \*pow)  
*Read SVI based telemetry for individual rails.*
- [oob\\_status\\_t get\\_die\\_hotspot\\_info](#) (uint8\_t soc\_num, uint8\_t \*die\_id, uint16\_t \*temp)  
*Reads local ID of the hottest die and its temperature.*
- [oob\\_status\\_t get\\_mem\\_hotspot\\_info](#) (uint8\_t soc\_num, uint8\_t \*hbm\_stack\_id, uint16\_t \*hbm\_temp)  
*Reads local ID of the HBM stack and its temperature.*
- [oob\\_status\\_t get\\_host\\_status](#) (uint8\_t soc\_num, struct [host\\_status](#) \*status)  
*Reads the status in a bit vector.*
- [oob\\_status\\_t get\\_max\\_mem\\_bw\\_util](#) (uint8\_t soc\_num, struct [max\\_mem\\_bw](#) \*bw)  
*Reads max memory bandwidth utilization.*
- [oob\\_status\\_t get\\_hbm\\_throttle](#) (uint8\_t soc\_num, uint32\_t \*mem\_th)  
*Reads HBM throttle.*
- [oob\\_status\\_t set\\_hbm\\_throttle](#) (uint8\_t soc\_num, uint32\_t mem\_th)  
*writes HBM throttle.*
- [oob\\_status\\_t get\\_hbm\\_temperature](#) (uint8\_t soc\_num, uint32\_t index, uint16\_t \*temp)  
*Reads hbm stack temperature.*
- [oob\\_status\\_t get\\_clk\\_freq\\_limits](#) (uint8\_t soc\_num, enum [clk\\_type](#) type, struct [freq\\_limits](#) \*limit)  
*Reads GFX/F clk frequency limits based on enumeration type [clk\\_type](#) .*
- [oob\\_status\\_t get\\_sockets\\_in\\_system](#) (uint8\_t soc\_num, uint32\_t \*sockets\_count)  
*Reads number of sockets in system.*
- [oob\\_status\\_t get\\_bist\\_results](#) (uint8\_t soc\_num, uint8\_t die\_id, uint32\_t \*bist\_result)

- Reads die level bist result status from package.*
- [oob\\_status\\_t get\\_statistics](#) (uint8\_t soc\_num, struct [statistics](#) stat, uint32\_t \*param)  
*Reads statistics for a given parameter.*
- [oob\\_status\\_t clear\\_statistics](#) (uint8\_t soc\_num)  
*Clears statistics.*
- [oob\\_status\\_t get\\_die\\_type](#) (uint8\_t soc\_num, uint32\_t data\_in, uint32\_t \*data\_out)  
*Get die type.*
- [oob\\_status\\_t get\\_curr\\_xgmi\\_pstate](#) (uint8\_t soc\_num, uint8\_t \*xgmi\_pstate)  
*Get current xgmi pstate Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).*
- [oob\\_status\\_t get\\_max\\_operating\\_temp](#) (uint8\_t soc\_num, uint32\_t ctf\_type, uint16\_t \*temp)  
*Get maximum operating temperature.*
- [oob\\_status\\_t get\\_slow\\_down\\_temp](#) (uint8\_t soc\_num, uint32\_t ctf\_type, uint16\_t \*slow\_down\_temp)  
*Get slow down temperature Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).*
- [oob\\_status\\_t get\\_hbm\\_dev\\_info](#) (uint8\_t soc\_num, struct [hbm\\_device\\_info](#) \*dev\_info)  
*Get hbm device information.*
- [oob\\_status\\_t get\\_pciestats](#) (uint8\_t soc\_num, uint32\_t pcie\_stat\_select, uint32\_t \*pcie\_stats)  
*Get PCIe statistics.*

### 7.8.1 Detailed Description

Header file for the MI300 mailbox messages supported by APML library. All required function, structure, enum, etc. definitions should be defined in this file.

This header file contains the following: APIs prototype of the Mailbox messages for MI300 exported by the APML library. Description of the API, arguments and return values. The Error codes returned by the API.

### 7.8.2 Function Documentation

#### 7.8.2.1 set\_gfx\_core\_clock()

```
oob_status_t set_gfx_core_clock (
    uint8_t soc_num,
    enum range_type freq_type,
    uint32_t freq )
```

Set maximum/minimum gfx core clock frequency.

This function sets user provided frequency as MAX GFX/MIN GFX core clock frequency in MHZ based on enumeration type [range\\_type](#). Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>freq_type</i>	enumeration type <a href="#">range_type</a> containing "MIN" = 0 or "MAX" = 1.
in	<i>freq</i>	frequency in MHZ.



## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

7.8.2.2 `set_mclk_fclk_max_pstate()`

```
oob_status_t set_mclk_fclk_max_pstate (
    uint8_t soc_num,
    uint32_t pstate )
```

Set maximum mem and fclk pstate.

This function sets the maximum memory and fabric clock power state. Mappings from memory and fabric clock pstate to MEMCLK/FCLK frequency can be found by issuing GetPstates command. Supported platforms↵: [Fam-19h\\_Mod-90h-9Fh](#).

In case if the in-band has also set the maximum Pstate, then lower of the limits is used.

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>pstate</i>	maximum pstate.Valid pstate range is 0 - 3.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

7.8.2.3 `get_mclk_fclk_pstates()`

```
oob_status_t get_mclk_fclk_pstates (
    uint8_t soc_num,
    uint8_t pstate_ind,
    struct mclk_fclk_pstates * pstate )
```

Get memory and fabric clock power state mappings.

This function returns the memory and fabric clock power state mappings. Returns MEMCLK/FCLK frequency in units of 1MHz for the available clock power states (Pstates). Each MEMCLK/FCLK frequency pair is returned independently for each pstate. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>pstate_ind</i>	index of the pstate.
out	<i>pstate</i>	struct <a href="#">mclk_fclk_pstates</a> containing mem_clk and f_clk frequency in MHz.



## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.8.2.4 set\_xgmi\_pstate()

```
oob_status_t set_xgmi_pstate (
    uint8_t soc_num,
    uint32_t pstate )
```

Sets the XGMI Pstate.

This function sets the specified XGMI Pstate. This disables all active XGMI Pstate management although XGMI power down modes will still be supported. Only 2 XGMI states are supported (0/1). Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>pstate</i>	power state. valid values are 0 - 1.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.8.2.5 unset\_xgmi\_pstate()

```
oob_status_t unset_xgmi_pstate (
    uint8_t soc_num )
```

Resets the XGMI Pstate.

This function resets the XGMI Pstate specified in the SetXgmiPstate, causing XGMI link speed/width to be actively managed by the GPU. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
----	----------------	---------------

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 7.8.2.6 get\_xgmi\_pstates()

```
oob_status_t get_xgmi_pstates (
    uint8_t soc_num,
    uint8_t pstate_ind,
    struct xgmi_speed_rate_n_width * xgmi_pstate )
```

Read XGMI power state mappings.

This function reads the XGMI power state mappings. Reads the supported XGMI link speeds and widths available to the SetXgmiPstate message. Link speeds reported in units of 1Gpbs. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>pstate_ind</i>	xgmi pstate index for speed rate.
out	<i>xgmi_pstate</i>	struct <a href="#">xgmi_speed_rate_n_width</a> containing speed rate in MHz and link width

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 7.8.2.7 get\_xcc\_idle\_residency()

```
oob_status_t get_xcc_idle_residency (
    uint8_t soc_num,
    uint32_t * gfx_cores_idle_res )
```

Read xcc idle residency percentage.

This function will provide the average xcc idle residency across all GFX cores in the socket. 100% specifies that all enabled GFX cores in the socket are running in idle. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
out	<i>gfx_cores_idle_res</i>	idle residency in percentage

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.8.2.8 get\_energy\_accum\_with\_timestamp()

```
oob_status_t get_energy_accum_with_timestamp (
    uint8_t soc_num,
    uint64_t * energy,
    uint64_t * time_stamp )
```

Read energy accumulator with time stamp.

This function will read 64 bits energy accumulator and the 56 bit time stamp. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>energy</i>	accumulator 2 <sup>16</sup> J.
out	<i>time_stamp</i>	time stamp (units:10ns).

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.8.2.9 get\_alarms()

```
oob_status_t get_alarms (
    uint8_t soc_num,
    enum alarms_type type,
    uint32_t * buffer )
```

Read PM alarm status based on enumeration type [alarms\\_type](#).

This function provides PM alarm status if the enumeration type [alarms\\_type](#) is PM = 0 then it will retrieve PM alarm status . If buffer value is 1 the status is VRHOT. If the buffer value is 2 status is die over temp. If the buffer value is 4 status is HBM over temp and if the buffer is 8 then status is PWRBRK. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>type</i>	enumeration type <a href="#">alarms_type</a> . PM = 0.
out	<i>buffer</i>	returns PSP fw return data.

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 7.8.2.10 get\_psn()

```
oob_status_t get_psn (
    uint8_t soc_num,
    uint32_t die_index,
    uint64_t * buffer )
```

Reads public serial number (PSN).

This function will return 64 bit public serial number (PSN) unique to each die. Supported platforms↵: [Fam-19h\\_Mod-90h-9Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>die_index</i>	core/die index.
out	<i>buffer</i>	returns 64 bit unique public serial number.

#### Return values

<i>OOB_SUCCESS</i>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 7.8.2.11 get\_link\_info()

```
oob_status_t get_link_info (
    uint8_t soc_num,
    uint8_t * link_config,
    uint8_t * module_id )
```

Read link Info.

This function will read link info. Function will read the module ID and link config reflecting strapping pins. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
out	<i>link_config</i>	link configuration.
out	<i>module↵_id</i>	module ID.

#### Return values

<i>OOB_SUCCESS</i>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.8.2.12 get\_max\_min\_gfx\_freq()

```
oob_status_t get_max_min_gfx_freq (
    uint8_t soc_num,
    uint16_t * max_freq,
    uint16_t * min_freq )
```

Read maximum and minimum allowed GFX engine frequency.

This function will read maximum and minimum allowed GFX engine frequency. Supported platforms↵: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>max_freq</i>	maximum GFX frequency in MHZ.
out	<i>min_freq</i>	minimum GFX frequency in MHZ.

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.8.2.13 get\_act\_gfx\_freq\_cap()

```
oob_status_t get_act_gfx_freq_cap (
    uint8_t soc_num,
    uint16_t * freq )
```

Read Actual GFX frequency cap selected.

This function will read current seleted GFX engine clock frequency.It reflects minimum of all frequency caps seleted via in-band and out-of-band controls. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>freq</i>	maximum GFX frequency in MHZ.

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 7.8.2.14 get\_svi\_rail\_telemetry()

```
oob_status_t get_svi_rail_telemetry (
    uint8_t soc_num,
    struct svi_port_domain port,
    uint32_t * pow )
```

Read SVI based telemetry for individual rails.

This function will read SVI based telemetry for individual rails. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>port</i>	struct svi_telemetry_domain containing port and slave address.
out	<i>pow</i>	power in milliwatts.

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 7.8.2.15 get\_die\_hotspot\_info()

```
oob_status_t get_die_hotspot_info (
    uint8_t soc_num,
    uint8_t * die_id,
    uint16_t * temp )
```

Reads local ID of the hottest die and its temperature.

This function will read local ID of the hottest die and its corresponding die temperature. Measured in every 1 ms and the most recently measured temperature in °C is reported. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
out	<i>die_id</i>	Hottest die ID.
out	<i>temp</i>	Die hot spot temperature in °C.

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.8.2.16 get\_mem\_hotspot\_info()

```
oob_status_t get_mem_hotspot_info (
    uint8_t soc_num,
    uint8_t * hbm_stack_id,
    uint16_t * hbm_temp )
```

Reads local ID of the HBM stack and its temperature.

This function will read local ID of the HBM stack and its corresponding HBM stack temperature. Measured in every 1 ms and the most recently measured temperature is reported. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>hbm_stack_id</i>	Local ID of the hottest HBM stack.
out	<i>hbm_temp</i>	temperature in units of 1 °C.

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.8.2.17 get\_host\_status()

```
oob_status_t get_host_status (
    uint8_t soc_num,
    struct host_status * status )
```

Reads the status in a bit vector.

This function will read PM controller status and driver running status in a bit vector Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>status</i>	struct <a href="#">host_status</a> containing power management controller status and driver running status.

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 7.8.2.18 get\_max\_mem\_bw\_util()

```
oob_status_t get_max_mem_bw_util (
    uint8_t soc_num,
    struct max_mem_bw * bw )
```

Reads max memory bandwidth utilization.

This function will provide theoretic.al maximum HBM/memory bandwidth of the system in GB/s, utilized bandwidth in GB/S. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
out	<i>bw</i>	struct <a href="#">max_mem_bw</a> containing max bw, utilized b/w.

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 7.8.2.19 get\_hbm\_throttle()

```
oob_status_t get_hbm_throttle (
    uint8_t soc_num,
    uint32_t * mem_th )
```

Reads HBM throttle.

This function will read HBM throttle in percentage. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
out	<i>mem_th</i>	hbm throttle in percentage (0 - 100%).

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

### 7.8.2.20 set\_hbm\_throttle()

```
oob_status_t set_hbm_throttle (
    uint8_t soc_num,
    uint32_t mem_th )
```



writes HBM throttle.

This function will write HBM throttle. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>mem_th</i>	hbm throttle in percentage (0 - 80%).

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.8.2.21 get\_hbm\_temperature()

```
oob_status_t get_hbm_temperature (
    uint8_t soc_num,
    uint32_t index,
    uint16_t * temp )
```

Reads hbm stack temperature.

This function will read particular hbm stack temperature. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

#### Parameters

in	<i>soc_num</i>	Socket index.
in	<i>index</i>	hbm stack index ( 0 - 7 for MI300).
out	<i>temp</i>	temperature in units of 1 °C.

#### Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

#### 7.8.2.22 get\_clk\_freq\_limits()

```
oob_status_t get_clk_freq_limits (
    uint8_t soc_num,
    enum clk_type type,
    struct freq_limits * limit )
```

Reads GFX/F clk frequency limits based on enumeration type [clk\\_type](#) .

This function will provide socket's GFX/F clk max and min frequency limits based on enumeration type [clk\\_type](#). The function reads GFX clk frequency limits if the enumeration type [clk\\_type](#) is GFX\_CLK = 0 else it will read F\_CLK frequency limits. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>type</i>	enumeration type <a href="#">clk_type</a> . Values are "GFX_CLK" = 0 or "F_CLK" = 1.
out	<i>limit</i>	struct <a href="#">freq_limits</a> containing max and min GFX/F_clk frequency in MHZ.

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

7.8.2.23 `get_sockets_in_system()`

```
oob_status_t get_sockets_in_system (
    uint8_t soc_num,
    uint32_t * sockets_count )
```

Reads number of sockets in system.

This function will read number of sockets in system. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>sockets_count</i>	Numbers of sockets in system

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

7.8.2.24 `get_bist_results()`

```
oob_status_t get_bist_results (
    uint8_t soc_num,
    uint8_t die_id,
    uint32_t * bist_result )
```

Reads die level bist result status from package.

This function will read die level bist result status from package. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>die_id</i>	die level id.
out	<i>bist_result</i>	constituent bist results depending on MI300X/A/C configuration.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

7.8.2.25 `get_statistics()`

```
oob_status_t get_statistics (
    uint8_t soc_num,
    struct statistics stat,
    uint32_t * param )
```

Reads statistics for a given parameter.

This function will read statistics for a given parameter since the last clear statistics command. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>stat</i>	struct statistics containing statistics parameter of interest and output control.
out	<i>param</i>	parameter or timestamp HI/Lo value.

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

7.8.2.26 `clear_statistics()`

```
oob_status_t clear_statistics (
    uint8_t soc_num )
```

Clears statistics.

This function will clear all stored query statistics timestamps and then resumes data collection or aggregation. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
----	----------------	---------------

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
------------------------------------	-----------------------------------

## Return values

<i>Non-zero</i>	is returned upon failure.
-----------------	---------------------------

## 7.8.2.27 get\_die\_type()

```
oob_status_t get_die_type (
    uint8_t soc_num,
    uint32_t data_in,
    uint32_t * data_out )
```

Get die type.

This function will read die-type, counts and AID base die based on die-ID input. If the bit[0] of input/data\_in is 1 then it will get maximum die-ID (0 - 255). If the bit[0] of input/data\_in is 0 then the data\_out[7:0] will be die type i.e. 0 Not available, 1 means AID, 2 means XCD, 3 means CCD and 4 means HBM stack and 5- 255 are reserved. data\_out[15:8] means maximum count of currently indexed die type. Data\_out[19:16] means AID associated with specified die-ID. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>data_in</i>	input to get the die-type, counts and AID base.
out	<i>data_out</i>	maximum die id idnex or the die type based on die-id input.

## Return values

<a href="#">OOB_SUCCESS</a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.8.2.28 get\_curr\_xgmi\_pstate()

```
oob_status_t get_curr_xgmi_pstate (
    uint8_t soc_num,
    uint8_t * xgmi_pstate )
```

Get current xgmi pstate Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>xgmi_pstate</i>	current xgmi pstate

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

7.8.2.29 `get_max_operating_temp()`

```
oob_status_t get_max_operating_temp (
    uint8_t soc_num,
    uint32_t ctf_type,
    uint16_t * temp )
```

Get maximum operating temperature.

This function will get critical temperature fault core and HBM temperature. Supported platforms↵  
: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>ctf_type</i>	critical temperature fault type. 0 for AID, 1 for CCD, 2 for XCD and 3 for HBM.
out	<i>temp</i>	Max operating temp in °C

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

7.8.2.30 `get_slow_down_temp()`

```
oob_status_t get_slow_down_temp (
    uint8_t soc_num,
    uint32_t ctf_type,
    uint16_t * slow_down_temp )
```

Get slow down temperature Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>ctf_type</i>	critical temperature fault type. 0 for AID, 1 for CCD, 2 for XCD and 3 for HBM.
out	<i>slow_down_temp</i>	slow down temperature in °C

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.8.2.31 get\_hbm\_dev\_info()

```
oob_status_t get_hbm_dev_info (
    uint8_t soc_num,
    struct hbm_device_info * dev_info )
```

Get hbm device information.

This function will get hbm device information i.e. device vendor, part number and total memory size (GBs). Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
out	<i>dev_info</i>	struct <a href="#">hbm_device_info</a> containing device vendor, part number and total memory size (GBs).

## Return values

<a href="#"><i>OOB_SUCCESS</i></a>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.

## 7.8.2.32 get\_pciestats()

```
oob_status_t get_pciestats (
    uint8_t soc_num,
    uint32_t pcie_stat_select,
    uint32_t * pcie_stats )
```

Get PCIe statistics.

This function will get PCIe statistics including total transitions from L0 to recovery state, total number of replays issued on the PCIe link, total number of NAKs issued on the PCIe link by the device, and total number of NAKs issued on the PCIe link by the receiver. Supported platforms: [Fam-19h\\_Mod-90h-9Fh](#).

## Parameters

in	<i>soc_num</i>	Socket index.
in	<i>pcie_stat_select</i>	PCIe stat selector. 0 for L0 to recovery count, 1 for replay count, 2 for NAK sent count and 3 for NAK recieved count.
out	<i>pcie_stats</i>	pcie statistics

## Return values

<i>OOB_SUCCESS</i>	is returned upon successful call.
<i>Non-zero</i>	is returned upon failure.



# Index

- action\_payload, 65
- apml\_encodings, 65
- apml\_err.h, 87
  - oob\_status\_t, 88
- apml\_recover\_dev
  - apml\_recovery.h, 89
- apml\_recovery.h, 89
  - apml\_recover\_dev, 89
- Auxiliary functions, 13
  - errno\_to\_oob\_status, 13
  - esmi\_get\_err\_msg, 14
- clear\_sbrmi\_ras\_status
  - SB-RMI Register Read Byte Protocol, 44
- clear\_statistics
  - rmi\_mailbox\_mi300.h, 142
- core\_mca\_err\_reporting\_en
  - oob\_config\_d\_in, 75
- cpuid\_reg
  - esmi\_cpuid\_msr.h, 91
- Current, Min, Max TDP, 24
  - read\_max\_tdp, 24
  - read\_min\_tdp, 25
  - read\_tdp, 24
- dimmm\_power, 66
- dimmm\_spd\_d\_in, 66
- dimmm\_thermal, 67
- dpm\_level, 68
- Dram and other features Query, 28
  - read\_ccd\_bist\_result, 31
  - read\_cclk\_freq\_limit, 32
  - read\_ccx\_bist\_result, 31
  - read\_ddr\_bandwidth, 33
  - read\_dram\_throttle, 28
  - read\_iod\_bist, 30
  - read\_nbio\_error\_logging\_register, 30
  - read\_socket\_c0\_residency, 32
  - read\_vddio\_mem\_power, 29
  - write\_dram\_throttle, 29
- dram\_cecc\_leak\_rate
  - oob\_config\_d\_in, 75
- dram\_cecc\_oob\_ec\_mode
  - oob\_config\_d\_in, 75
- err\_type
  - run\_time\_threshold, 82
- errno\_to\_oob\_status
  - Auxiliary functions, 13
- esmi\_cpuid\_msr.h, 90
  - cpuid\_reg, 91
- esmi\_get\_err\_msg
  - Auxiliary functions, 14
- esmi\_get\_logical\_cores\_per\_socket
  - using CPUID Register Access, 35
- esmi\_get\_processor\_info
  - using CPUID Register Access, 34
- esmi\_get\_threads\_per\_core
  - using CPUID Register Access, 35
- esmi\_get\_threads\_per\_socket
  - SB-RMI Register Read Byte Protocol, 45
- esmi\_get\_vendor\_id
  - using CPUID Register Access, 34
- esmi\_mailbox.h, 91
  - get\_bmc\_ras\_oob\_config, 116
  - get\_bmc\_ras\_run\_time\_err\_validity\_ck, 114
  - get\_bmc\_ras\_run\_time\_error\_info, 115
  - get\_dimm\_serial\_num, 119
  - get\_post\_code, 114
  - override\_delay\_reset\_on\_sync\_flood, 113
  - read\_bmc\_control\_pcie\_gen5\_rate, 109
  - read\_bmc\_cpu\_base\_frequency, 108
  - read\_bmc\_rapl\_units, 108
  - read\_bmc\_ras\_fch\_reset\_reason, 100
  - read\_bmc\_ras\_mca\_msr\_dump, 99
  - read\_bmc\_ras\_mca\_validity\_check, 99
  - read\_bmc\_ras\_pcie\_config\_access, 98
  - read\_current\_dfpstate\_frequency, 107
  - read\_current\_io\_bandwidth, 104
  - read\_current\_xgmi\_bandwidth, 104
  - read\_dimm\_power\_consumption, 101
  - read\_dimm\_spd\_register, 119
  - read\_dimm\_temp\_range\_and\_refresh\_rate, 100
  - read\_dimm\_thermal\_sensor, 101
  - read\_lclk\_dpm\_level\_range, 112
  - read\_ppin\_fuse, 117
  - read\_pwr\_current\_active\_freq\_limit\_core, 102
  - read\_pwr\_current\_active\_freq\_limit\_socket, 102
  - read\_pwr\_svi\_telemetry\_all\_rails, 103
  - read\_rapl\_core\_energy\_counters, 109
  - read\_rapl\_pckg\_energy\_counters, 110
  - read\_ras\_df\_err\_dump, 118
  - read\_ras\_df\_err\_validity\_check, 117
  - read\_rtc, 118
  - read\_socket\_freq\_range, 103
  - read\_ucode\_revision, 112
  - reset\_on\_sync\_flood, 113
  - set\_bmc\_ras\_err\_threshold, 115
  - set\_bmc\_ras\_oob\_config, 116

- write\_apb\_disable, 106
- write\_apb\_enable, 106
- write\_bmc\_report\_dimm\_power, 97
- write\_bmc\_report\_dimm\_thermal\_sensor, 98
- write\_df\_pstate\_range, 111
- write\_gmi3\_link\_width\_range, 105
- write\_lclk\_dpm\_level\_range, 107
- write\_pwr\_efficiency\_mode, 110
- write\_xgmi\_link\_width\_range, 105
- esmi\_mailbox\_nda.h, 119
  - get\_bmc\_ras\_action\_status, 121
  - set\_bmc\_ras\_action\_status, 120
- esmi\_oob\_cpuid
  - SB-RMI CPUID Register Access, 38
- esmi\_oob\_cpuid\_eax
  - SB-RMI CPUID Register Access, 39
- esmi\_oob\_cpuid\_ebx
  - SB-RMI CPUID Register Access, 39
- esmi\_oob\_cpuid\_ecx
  - SB-RMI CPUID Register Access, 40
- esmi\_oob\_cpuid\_edx
  - SB-RMI CPUID Register Access, 40
- esmi\_oob\_read\_msr
  - SB-RMI Read Processor Register Access, 37
- esmi\_rmi.h, 121
- esmi\_tsi.h, 123
- freq\_limits, 68
- get\_act\_gfx\_freq\_cap
  - rmi\_mailbox\_mi300.h, 135
- get\_alarms
  - rmi\_mailbox\_mi300.h, 133
- get\_bist\_results
  - rmi\_mailbox\_mi300.h, 141
- get\_bmc\_ras\_action\_status
  - esmi\_mailbox\_nda.h, 121
- get\_bmc\_ras\_oob\_config
  - esmi\_mailbox.h, 116
- get\_bmc\_ras\_run\_time\_err\_validity\_ck
  - esmi\_mailbox.h, 114
- get\_bmc\_ras\_run\_time\_error\_info
  - esmi\_mailbox.h, 115
- get\_clk\_freq\_limits
  - rmi\_mailbox\_mi300.h, 139
- get\_curr\_xgmi\_pstate
  - rmi\_mailbox\_mi300.h, 143
- get\_die\_hotspot\_info
  - rmi\_mailbox\_mi300.h, 136
- get\_die\_type
  - rmi\_mailbox\_mi300.h, 143
- get\_dimm\_serial\_num
  - esmi\_mailbox.h, 119
- get\_energy\_accum\_with\_timestamp
  - rmi\_mailbox\_mi300.h, 132
- get\_hbm\_dev\_info
  - rmi\_mailbox\_mi300.h, 145
- get\_hbm\_temperature
  - rmi\_mailbox\_mi300.h, 139
- get\_hbm\_throttle
  - rmi\_mailbox\_mi300.h, 138
- get\_host\_status
  - rmi\_mailbox\_mi300.h, 137
- get\_link\_info
  - rmi\_mailbox\_mi300.h, 134
- get\_max\_mem\_bw\_util
  - rmi\_mailbox\_mi300.h, 137
- get\_max\_min\_gfx\_freq
  - rmi\_mailbox\_mi300.h, 135
- get\_max\_operating\_temp
  - rmi\_mailbox\_mi300.h, 144
- get\_mclk\_fclk\_pstates
  - rmi\_mailbox\_mi300.h, 130
- get\_mem\_hotspot\_info
  - rmi\_mailbox\_mi300.h, 136
- get\_pciestats
  - rmi\_mailbox\_mi300.h, 145
- get\_post\_code
  - esmi\_mailbox.h, 114
- get\_psn
  - rmi\_mailbox\_mi300.h, 134
- get\_ras\_action\_data\_in, 69
- get\_slow\_down\_temp
  - rmi\_mailbox\_mi300.h, 144
- get\_sockets\_in\_system
  - rmi\_mailbox\_mi300.h, 141
- get\_statistics
  - rmi\_mailbox\_mi300.h, 142
- get\_svi\_rail\_telemetry
  - rmi\_mailbox\_mi300.h, 135
- get\_xcc\_idle\_residency
  - rmi\_mailbox\_mi300.h, 132
- get\_xgmi\_pstates
  - rmi\_mailbox\_mi300.h, 132
- hbm\_device\_info, 69
- host\_status, 70
- input
  - ras\_df\_err\_dump, 79
- lclk\_dpm\_level\_range, 70
- link\_id
  - link\_id\_bw\_type, 71
- link\_id\_bw\_type, 71
  - link\_id, 71
- max\_ddr\_bw, 71
- max\_mem\_bw, 72
- mca\_bank, 72
- mca\_oob\_misc0\_ec\_enable
  - oob\_config\_d\_in, 74
- mclk\_fclk\_pstates, 73
- nbio\_err\_log, 73
- oob\_config\_d\_in, 74
  - core\_mca\_err\_reporting\_en, 75
  - dram\_cecc\_leak\_rate, 75

- dram\_cecc\_oob\_ec\_mode, 75
- mca\_oob\_misc0\_ec\_enable, 74
- pcie\_err\_reporting\_en, 75
- oob\_status\_t
  - apml\_err.h, 88
- Out-of-band Performance (Boost limit) Control, 22
  - write\_esb\_boost\_limit, 22
  - write\_esb\_boost\_limit\_allcores, 22
- override\_delay\_reset\_on\_sync\_flood
  - esmi\_mailbox.h, 113
- pci\_address, 75
- pcie\_err\_reporting\_en
  - oob\_config\_d\_in, 75
- Performance (Boost limit) Monitor, 20
  - read\_bios\_boost\_fmax, 20
  - read\_esb\_boost\_limit, 20
- Platforms, 18
- Power Control, 19
  - write\_socket\_power\_limit, 19
- Power Monitor, 16
  - read\_max\_socket\_power\_limit, 17
  - read\_socket\_power, 16
  - read\_socket\_power\_limit, 16
- processor\_info, 76
- Prochot, 26
  - read\_prochot\_residency, 26
  - read\_prochot\_status, 26
- pstate\_freq, 77
- ras\_action\_status, 77
- ras\_df\_err\_chk, 78
- ras\_df\_err\_dump, 78
  - input, 79
- ras\_override\_delay, 79
- ras\_rt\_err\_req\_type, 80
- ras\_rt\_valid\_err\_inst, 80
- read\_bios\_boost\_fmax
  - Performance (Boost limit) Monitor, 20
- read\_bmc\_control\_pcie\_gen5\_rate
  - esmi\_mailbox.h, 109
- read\_bmc\_cpu\_base\_frequency
  - esmi\_mailbox.h, 108
- read\_bmc\_rapl\_units
  - esmi\_mailbox.h, 108
- read\_bmc\_ras\_fch\_reset\_reason
  - esmi\_mailbox.h, 100
- read\_bmc\_ras\_mca\_msr\_dump
  - esmi\_mailbox.h, 99
- read\_bmc\_ras\_mca\_validity\_check
  - esmi\_mailbox.h, 99
- read\_bmc\_ras\_pcie\_config\_access
  - esmi\_mailbox.h, 98
- read\_ccd\_bist\_result
  - Dram and other features Query, 31
- read\_cclk\_freq\_limit
  - Dram and other features Query, 32
- read\_ccx\_bist\_result
  - Dram and other features Query, 31
- read\_current\_dfpstate\_frequency
  - esmi\_mailbox.h, 107
- read\_current\_io\_bandwidth
  - esmi\_mailbox.h, 104
- read\_current\_xgmi\_bandwidth
  - esmi\_mailbox.h, 104
- read\_ddr\_bandwidth
  - Dram and other features Query, 33
- read\_dimm\_power\_consumption
  - esmi\_mailbox.h, 101
- read\_dimm\_spd\_register
  - esmi\_mailbox.h, 119
- read\_dimm\_temp\_range\_and\_refresh\_rate
  - esmi\_mailbox.h, 100
- read\_dimm\_thermal\_sensor
  - esmi\_mailbox.h, 101
- read\_dram\_throttle
  - Dram and other features Query, 28
- read\_esb\_boost\_limit
  - Performance (Boost limit) Monitor, 20
- read\_iod\_bist
  - Dram and other features Query, 30
- read\_lclk\_dpm\_level\_range
  - esmi\_mailbox.h, 112
- read\_max\_socket\_power\_limit
  - Power Monitor, 17
- read\_max\_tdp
  - Current, Min, Max TDP, 24
- read\_max\_threads\_per\_l3
  - SB-RMI CUID Register Access, 41
- read\_min\_tdp
  - Current, Min, Max TDP, 25
- read\_nbio\_error\_logging\_register
  - Dram and other features Query, 30
- read\_ppin\_fuse
  - esmi\_mailbox.h, 117
- read\_prochot\_residency
  - Prochot, 26
- read\_prochot\_status
  - Prochot, 26
- read\_pwr\_current\_active\_freq\_limit\_core
  - esmi\_mailbox.h, 102
- read\_pwr\_current\_active\_freq\_limit\_socket
  - esmi\_mailbox.h, 102
- read\_pwr\_svi\_telemetry\_all\_rails
  - esmi\_mailbox.h, 103
- read\_rapl\_core\_energy\_counters
  - esmi\_mailbox.h, 109
- read\_rapl\_pckg\_energy\_counters
  - esmi\_mailbox.h, 110
- read\_ras\_df\_err\_dump
  - esmi\_mailbox.h, 118
- read\_ras\_df\_err\_validity\_check
  - esmi\_mailbox.h, 117
- read\_rtc
  - esmi\_mailbox.h, 118
- read\_sbrmi\_alert\_mask
  - SB-RMI Register Read Byte Protocol, 44

- read\_sbrmi\_alert\_status
  - SB-RMI Register Read Byte Protocol, [43](#)
- read\_sbtsi\_alertconfig
  - SBTSI Register Read Byte Protocol, [56](#)
- read\_sbtsi\_alertthreshold
  - SBTSI Register Read Byte Protocol, [55](#)
- read\_sbtsi\_config
  - SBTSI Register Read Byte Protocol, [50](#)
- read\_sbtsi\_configwrite
  - SBTSI Register Read Byte Protocol, [52](#)
- read\_sbtsi\_cpuinttemp
  - SBTSI Register Read Byte Protocol, [49](#)
- read\_sbtsi\_cputempdecimal
  - SBTSI Register Read Byte Protocol, [53](#)
- read\_sbtsi\_cputempoffdec
  - SBTSI Register Read Byte Protocol, [54](#)
- read\_sbtsi\_cputempoffint
  - SBTSI Register Read Byte Protocol, [53](#)
- read\_sbtsi\_cputempoffset
  - SBTSI Register Read Byte Protocol, [62](#)
- read\_sbtsi\_hitempdecimal
  - SBTSI Register Read Byte Protocol, [54](#)
- read\_sbtsi\_hitempint
  - SBTSI Register Read Byte Protocol, [51](#)
- read\_sbtsi\_lotempdecimal
  - SBTSI Register Read Byte Protocol, [55](#)
- read\_sbtsi\_lotempint
  - SBTSI Register Read Byte Protocol, [52](#)
- read\_sbtsi\_manufid
  - SBTSI Register Read Byte Protocol, [56](#)
- read\_sbtsi\_revision
  - SBTSI Register Read Byte Protocol, [57](#)
- read\_sbtsi\_status
  - SBTSI Register Read Byte Protocol, [50](#)
- read\_sbtsi\_timeoutconfig
  - SBTSI Register Read Byte Protocol, [55](#)
- read\_sbtsi\_updaterate
  - SBTSI Register Read Byte Protocol, [50](#)
- read\_socket\_c0\_residency
  - Dram and other features Query, [32](#)
- read\_socket\_freq\_range
  - esmi\_mailbox.h, [103](#)
- read\_socket\_power
  - Power Monitor, [16](#)
- read\_socket\_power\_limit
  - Power Monitor, [16](#)
- read\_tdp
  - Current, Min, Max TDP, [24](#)
- read\_ucose\_revision
  - esmi\_mailbox.h, [112](#)
- read\_vddio\_mem\_power
  - Dram and other features Query, [29](#)
- reset\_on\_sync\_flood
  - esmi\_mailbox.h, [113](#)
- rmi\_mailbox\_mi300.h, [126](#)
  - clear\_statistics, [142](#)
  - get\_act\_gfx\_freq\_cap, [135](#)
  - get\_alarms, [133](#)
  - get\_bist\_results, [141](#)
  - get\_clk\_freq\_limits, [139](#)
  - get\_curr\_xgmi\_pstate, [143](#)
  - get\_die\_hotspot\_info, [136](#)
  - get\_die\_type, [143](#)
  - get\_energy\_accum\_with\_timestamp, [132](#)
  - get\_hbm\_dev\_info, [145](#)
  - get\_hbm\_temperature, [139](#)
  - get\_hbm\_throttle, [138](#)
  - get\_host\_status, [137](#)
  - get\_link\_info, [134](#)
  - get\_max\_mem\_bw\_util, [137](#)
  - get\_max\_min\_gfx\_freq, [135](#)
  - get\_max\_operating\_temp, [144](#)
  - get\_mclk\_fclk\_pstates, [130](#)
  - get\_mem\_hotspot\_info, [136](#)
  - get\_pciestats, [145](#)
  - get\_psn, [134](#)
  - get\_slow\_down\_temp, [144](#)
  - get\_sockets\_in\_system, [141](#)
  - get\_statistics, [142](#)
  - get\_svi\_rail\_telemetry, [135](#)
  - get\_xcc\_idle\_residency, [132](#)
  - get\_xgmi\_pstates, [132](#)
  - set\_gfx\_core\_clock, [129](#)
  - set\_hbm\_throttle, [138](#)
  - set\_mclk\_fclk\_max\_pstate, [130](#)
  - set\_xgmi\_pstate, [131](#)
  - unset\_xgmi\_pstate, [131](#)
- run\_time\_err\_d\_in, [81](#)
- run\_time\_threshold, [81](#)
  - err\_type, [82](#)
- SB-RMI CPUID Register Access, [38](#)
  - esmi\_oob\_cpuid, [38](#)
  - esmi\_oob\_cpuid\_eax, [39](#)
  - esmi\_oob\_cpuid\_ebx, [39](#)
  - esmi\_oob\_cpuid\_ecx, [40](#)
  - esmi\_oob\_cpuid\_edx, [40](#)
  - read\_max\_threads\_per\_l3, [41](#)
- SB-RMI Mailbox Service, [15](#)
- SB-RMI Register Read Byte Protocol, [42](#)
  - clear\_sbrmi\_ras\_status, [44](#)
  - esmi\_get\_threads\_per\_socket, [45](#)
  - read\_sbrmi\_alert\_mask, [44](#)
  - read\_sbrmi\_alert\_status, [43](#)
  - thread\_en\_reg\_v10, [45](#)
- SB\_RMI Read Processor Register Access, [37](#)
  - esmi\_oob\_read\_msr, [37](#)
- SBTSI Register Read Byte Protocol, [47](#)
  - read\_sbtsi\_alertconfig, [56](#)
  - read\_sbtsi\_alertthreshold, [55](#)
  - read\_sbtsi\_config, [50](#)
  - read\_sbtsi\_configwrite, [52](#)
  - read\_sbtsi\_cpuinttemp, [49](#)
  - read\_sbtsi\_cputempdecimal, [53](#)
  - read\_sbtsi\_cputempoffdec, [54](#)
  - read\_sbtsi\_cputempoffint, [53](#)
  - read\_sbtsi\_cputempoffset, [62](#)

- read\_sbtsi\_hitempdecimal, [54](#)
- read\_sbtsi\_hitempint, [51](#)
- read\_sbtsi\_lotempdecimal, [55](#)
- read\_sbtsi\_lotempint, [52](#)
- read\_sbtsi\_manufid, [56](#)
- read\_sbtsi\_revision, [57](#)
- read\_sbtsi\_status, [50](#)
- read\_sbtsi\_timeoutconfig, [55](#)
- read\_sbtsi\_updaterate, [50](#)
- sbtsi\_get\_config, [58](#)
- sbtsi\_get\_cputemp, [57](#)
- sbtsi\_get\_hitemp\_threshold, [61](#)
- sbtsi\_get\_lotemp\_threshold, [62](#)
- sbtsi\_get\_temp\_status, [58](#)
- sbtsi\_get\_timeout, [59](#)
- sbtsi\_set\_alert\_config, [63](#)
- sbtsi\_set\_alert\_threshold, [63](#)
- sbtsi\_set\_configwr, [59](#)
- sbtsi\_set\_hitemp\_threshold, [60](#)
- sbtsi\_set\_lotemp\_threshold, [61](#)
- sbtsi\_set\_timeout\_config, [60](#)
- write\_sbtsi\_cputempoffset, [63](#)
- write\_sbtsi\_updaterate, [51](#)
- sbtsi\_get\_config
  - SBTSI Register Read Byte Protocol, [48](#)
- sbtsi\_get\_cputemp
  - SBTSI Register Read Byte Protocol, [57](#)
- sbtsi\_get\_hitemp\_threshold
  - SBTSI Register Read Byte Protocol, [61](#)
- sbtsi\_get\_lotemp\_threshold
  - SBTSI Register Read Byte Protocol, [62](#)
- sbtsi\_get\_temp\_status
  - SBTSI Register Read Byte Protocol, [58](#)
- sbtsi\_get\_timeout
  - SBTSI Register Read Byte Protocol, [59](#)
- sbtsi\_set\_alert\_config
  - SBTSI Register Read Byte Protocol, [63](#)
- sbtsi\_set\_alert\_threshold
  - SBTSI Register Read Byte Protocol, [63](#)
- sbtsi\_set\_configwr
  - SBTSI Register Read Byte Protocol, [59](#)
- sbtsi\_set\_hitemp\_threshold
  - SBTSI Register Read Byte Protocol, [60](#)
- sbtsi\_set\_lotemp\_threshold
  - SBTSI Register Read Byte Protocol, [61](#)
- sbtsi\_set\_timeout\_config
  - SBTSI Register Read Byte Protocol, [60](#)
- set\_bmc\_ras\_action\_status
  - esmi\_mailbox.nda.h, [120](#)
- set\_bmc\_ras\_err\_threshold
  - esmi\_mailbox.h, [115](#)
- set\_bmc\_ras\_oob\_config
  - esmi\_mailbox.h, [116](#)
- set\_gfx\_core\_clock
  - rmi\_mailbox\_mi300.h, [129](#)
- set\_hbm\_throttle
  - rmi\_mailbox\_mi300.h, [138](#)
- set\_mclk\_fclk\_max\_pstate
  - rmi\_mailbox\_mi300.h, [130](#)
- set\_ras\_action\_data\_in, [82](#)
- set\_xgmi\_pstate
  - rmi\_mailbox\_mi300.h, [131](#)
- statistics, [83](#)
- status\_query\_payload, [83](#)
- svi\_port\_domain, [84](#)
- temp\_refresh\_rate, [84](#)
- thread\_en\_reg\_v10
  - SB-RMI Register Read Byte Protocol, [45](#)
- unset\_xgmi\_pstate
  - rmi\_mailbox\_mi300.h, [131](#)
- using CUID Register Access, [34](#)
  - esmi\_get\_logical\_cores\_per\_socket, [35](#)
  - esmi\_get\_processor\_info, [34](#)
  - esmi\_get\_threads\_per\_core, [35](#)
  - esmi\_get\_vendor\_id, [34](#)
- write\_apb\_disable
  - esmi\_mailbox.h, [106](#)
- write\_apb\_enable
  - esmi\_mailbox.h, [106](#)
- write\_bmc\_report\_dimm\_power
  - esmi\_mailbox.h, [97](#)
- write\_bmc\_report\_dimm\_thermal\_sensor
  - esmi\_mailbox.h, [98](#)
- write\_df\_pstate\_range
  - esmi\_mailbox.h, [111](#)
- write\_dram\_throttle
  - Dram and other features Query, [29](#)
- write\_esb\_boost\_limit
  - Out-of-band Performance (Boost limit) Control, [22](#)
- write\_esb\_boost\_limit\_allcores
  - Out-of-band Performance (Boost limit) Control, [22](#)
- write\_gmi3\_link\_width\_range
  - esmi\_mailbox.h, [105](#)
- write\_lclk\_dpm\_level\_range
  - esmi\_mailbox.h, [107](#)
- write\_pwr\_efficiency\_mode
  - esmi\_mailbox.h, [110](#)
- write\_sbtsi\_cputempoffset
  - SBTSI Register Read Byte Protocol, [63](#)
- write\_sbtsi\_updaterate
  - SBTSI Register Read Byte Protocol, [51](#)
- write\_socket\_power\_limit
  - Power Control, [19](#)
- write\_xgmi\_link\_width\_range
  - esmi\_mailbox.h, [105](#)
- xgmi\_speed\_rate\_n\_width, [85](#)