SEMA® EAPI Guide

(Applicable to SEMA Linux Version ≥ 4.4.0
and SEMA Windows Version ≥ 4.2.15)

February 17, 2026

Leading EDGE COMPUTING

ADLINK
LEADING EDGE COMPUTING

Copyright 2026 ADLINK Technology, Inc.

Disclaimer

The information in this document is subject to change without prior notice in order to improve reliability, design, and function and does not represent a commitment on the part of the manufacturer. In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages. This document contains proprietary information protected by copyright.

Trademark Information

Product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

Revision History

Revision	Release Date	Description of Changes
1.0	17/02/2026	Initial version of User manual for SEMA Windows and Linux

Contents

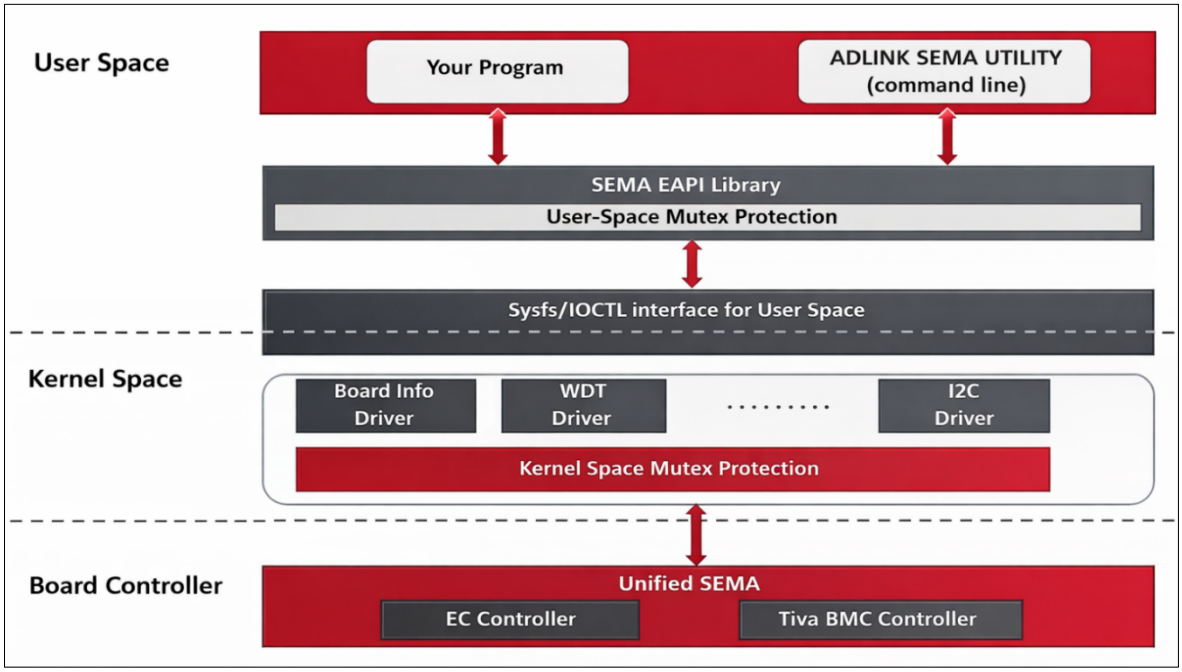
1. Acronyms / Definitions

Abbreviation	Description
EC	Embedded Controller
ACPI	Advanced Configuration Power Interface – standard to implement power saving modes in PC-AT systems
EAPI	Embedded Application Programming Interface
SEMA	Smart Embedded Management Agent
BIOS	Basic Input Output System
Boot Counter	Boot Counter is incremented after a HW or SW reset or after a successful power-up. A lifetime system count of the number of times the EFI/BIOS's Post is completed. Post completion is defined as being just prior to processor control being passed to the first boot vector.
Carrier Board	An application specific circuit board that accepts a COM Express Module
COM Express	PICMG definition for Computer-On-Modules
EEPROM	Electrically Erasable Programmable Read-Only Memory
EFI	Extensible Firmware Interface
GPIO	General Purpose Input Output
GPI	General Purpose Input
GPO	General Purpose Output
I2C	Inter Integrated Circuit - 2 wire (clock and data) signalling scheme allowing communication between integrated circuits, primarily used to read and load register values.
LAN	Local Area Network
LVDS	Low Voltage Differential Signalling – widely used as a physical interface for TFT flat panels. LVDS can be used for many high-speed signalling applications. In this document, it refers only to TFT flat-panel applications.
OEM/ODM	Original Equipment/Design Manufacturer
OS	Operating System
PCI	Peripheral Component Interface
PCIE	PCI Express– next-generation high speed Serialized I/O bus
PNP	Plug and Play
Power-up Watchdog	Used to handle system reset or boot failures. The reset cycle is repeated until the system successfully boots and the application services the watchdog.
ROM	Read Only Memory – a legacy term – often the device referred to as a ROM can actually be written to, in a special mode. Such writable ROMs are sometimes called Flash ROMs. BIOS is stored in ROM or Flash ROM

Abbreviation	Description
RTC	Real Time Clock – battery backed circuit in PC-AT systems that keeps system time and date as well as certain system setup parameters
Run-time watchdog	Used to recover from operating system hangs, application software hangs, or other system malfunctions. When the runtime watchdog timer expires, the EC/BMC triggers a system reset and automatically stops the runtime watchdog.
Running Time Meter	A lifetime system counts of the number of whole minutes that the system reset has been inactive.
S0, S3, S4, S5	System states describing the power and activity level S0 – Full power, all devices powered S3 – Suspend to RAM system- Context stored in RAM; RAM is in standby S4 – Suspend to Disk system- Context stored on disk S5 – Soft Off – Main power rail off, only standby rail present
SMBus	System Management Bus
SPI	Serial Peripheral Interface
VGA	Video Graphics Adapter
Watchdog	A hardware or software count-down timer that must be repeatedly reset or triggered at regular intervals by an operating program to prevent time-out. If the timer is not reset in time (meaning the operating program has stalled or has failed in some way), the timer times-out and alerts the system and/or operator of the failure.
Watchdog Trigger	The process or technique or resting the Watchdog counter, preventing a Watchdog timeout event.
WDT	Watchdog Timer

2. Overview

2.1. For Linux Platform

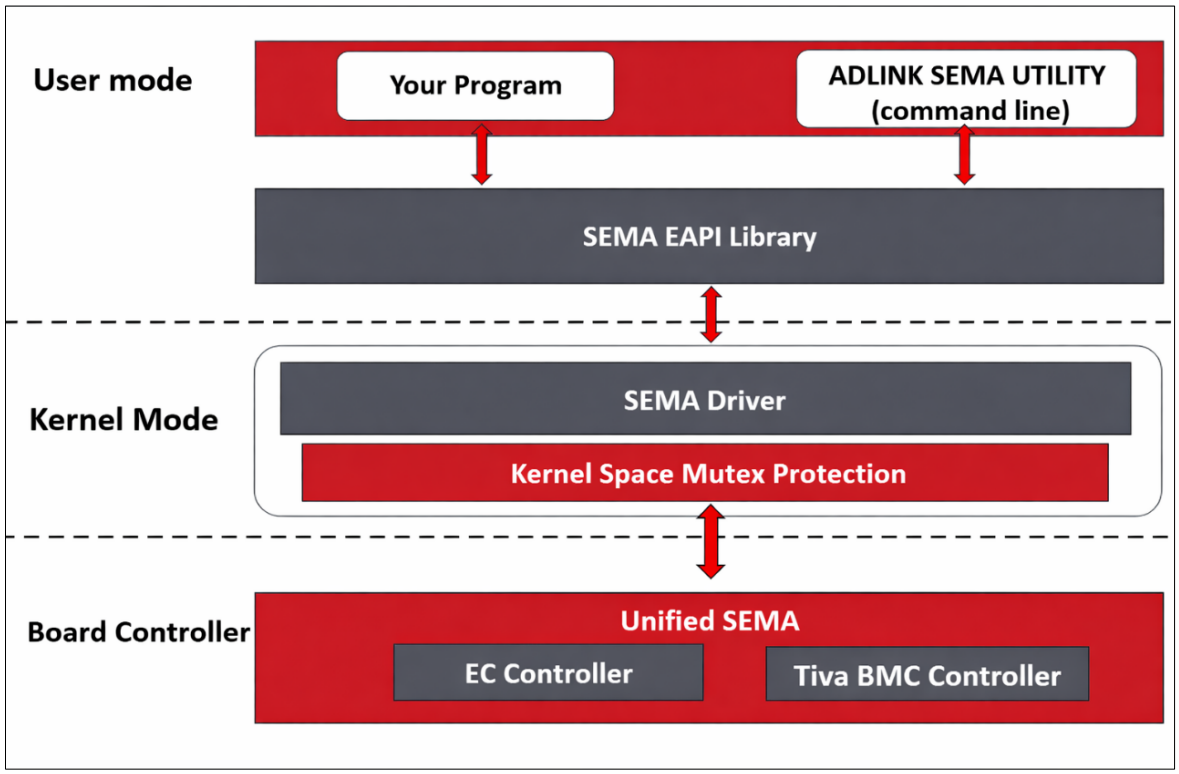


- **User Space Layer:**
  User applications and the ADLINK SEMA utility access hardware services through the SEMA EAPI library. User-space mutex protection ensures safe multi-threaded access before requests are forwarded to the kernel.

- **Kernel Space Layer:**
  Hardware drivers (Board Info, WDT, I^2C, etc.) receive requests via the **Sysfs/IOCTL** interface. Kernel-level mutex protection synchronizes driver operations and prevents race conditions during hardware access.

- **Board Controller Layer:**
  The Unified SEMA interface communicates with embedded controllers such as the EC and Tiva BMC. This layer executes low-level control and monitoring functions for reliable board management.

2.2. For Windows Platform



- **User Mode:**
  User applications and the ADLINK SEMA command-line utility access hardware services through the SEMA EAPI Library. This layer provides a unified API interface for board management and monitoring operations.
- **Kernel Mode:**
  The SEMA Driver receives requests from user space and acts as the main communication bridge between applications and the hardware controllers. It handles command routing and low-level device coordination.
- **Board Controller Layer:**
  The EC Controller and Tiva BMC Controller execute hardware control and monitoring functions through the Unified SEMA interface. This layer directly manages board-level sensors, power, and system features.

3. Type Definitions

__IN

Parameter Type	Characteristics
Immediate value	Input value that must be specified and is essential.
Pointer	Valid pointer to an initialized buffer/variable.

__OUT

Parameter Type	Characteristics
Pointer	Valid pointer to a destination buffer/variable.

__INOPT

Parameter Type	Characteristics
Pointer	Valid pointer to an initialized buffer/variable or NULL pointer. **Note:** Refer to function specification for specifics.

__OUTOPT

Parameter Type	Characteristics
Pointer	Valid pointer to a destination buffer/variable or NULL pointer. **Note:** Refer to function specification for specifics.

__INOUT

Parameter Type	Characteristics
Pointer	Valid pointer to an initialized buffer/variable. Contents of buffer/variable updated before return.

4. Status Codes

Refer section **9.1 Error Codes** for macros and their values.

Status Code	Description	Action
EAPI_STATUS_NOT_INITIALIZED	The EAPI library is not yet or unsuccessfully initialized	**EApiLibInitialize** needs to be called prior to the first access of any other EAPI function
EAPI_STATUS_INITIALIZED	The EAPI Library is successfully initialized	None
EAPI_STATUS_ALLOC_ERROR	Memory Allocation Error	Free memory and try again
EAPI_STATUS_INVALID_PARAMETER	One or more of the EAPI function call parameters are out of the defined range	Verify Function Parameters
EAPI_STATUS_INVALID_BLOCK_ LENGTH	This means that the Block length is too long	Use relevant Capabilities information to correct select block lengths
EAPI_STATUS_INVALID_BLOCK_ ALIGNMENT	The Block Alignment is incorrect.	Use Alignment Capabilities information to correctly align write access
EAPI_STATUS_INVALID_DIRECTION	The current Direction Argument attempts to set GPIOs to an unsupported direction I.E. Setting GPI to Output	Use pInputs and pOutputs to correctly select input and outputs
EAPI_STATUS_INVALID_BITMASK	The Bitmask Selects bits/GPIOs which are not supported for the current ID	Use pInputs and pOutputs to probe supported bits
EAPI_STATUS_UNSUPPORTED	This function or ID is not supported at the actual hardware environment	None
EAPI_STATUS_NOT_FOUND	Selected device was not found. Eg: The I2C device address is not Acknowledged, device is not present or inactive	None
EAPI_STATUS_BUSY_COLLISION	The selected device or ID is busy or a data collision was detected.	Retry

	Eg: • The addressed I2C bus is busy or there is a bus collision. • The I2C bus is in use. • Either CLK or DAT are low. • Arbitration loss or bus Collision, data remains low when writing a 1	
EAPI_STATUS_RUNNING	Watchdog timer already started.	Call EApiWDogStop, before retrying.
EAPI_STATUS_TIMEOUT	Function call timed out. Eg: I2C operation lasted too long.	Retry
EAPI_STATUS_READ_ERROR	An error was detected during a read operation. Eg: I2C Read function was not successful	Retry
EAPI_STATUS_WRITE_ERROR	An error was detected during a write operation. Eg: • I2C Write function was not successful. • No Acknowledge was received after writing any byte after the first address byte. • Can be caused by unsupported device command/index. • 10Bit Address Device Not Present • Storage Write Error	Retry
EAPI_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Storage buffer overflow was prevented. Read count was larger than the defined buffer length.	Either increase the buffer size or reduce the block length
EAPI_STATUS_ERROR	Generic error message. No further error details are available.	None
EAPI_STATUS_SUCCESS	The operation was successful. The value for this status code is defined as 0.	None

5. For Linux Platforms

5.1 Initialization Functions

The Initialization functions prepare the underlying communication layer, allocate required resources, and establish access to the board controller (EC/BMC).

5.1.1 EApiLibInitialize

Function Definition

```
uint32_t
EAPI_CALLTYPE
EApiLibInitialize(void);
```

Description

General initialization of the EAPI. Prior to calling any EAPI function, the library needs to be initialized by calling this function.

Parameters

None.

Return Status Code

Condition	Return Value
Initialization Success	EAPI_STATUS_SUCCESS
Initialization Failure	EAPI_STATUS_NOT_INITIALIZED

5.1.2 EApiLibUnInitialize

Function Definition

```
uint32_t
EAPI_CALLTYPE
EApiLibUnInitialize(void);
```

Description

General function to uninitialize the EAPI library. Should be called before program exit.

Parameters

None.

Return Status Code

Condition	Return Value
Success	EAPI_STATUS_SUCCESS

5.2 Board Information Functions

SEMA EAPI functions provide an interface to control or get board's information, including

- Static board and manufacturer information (e.g. BIOS version, Manufacturer name)
- Failure forensics (e.g. restart event)

5.2.1 EApiBoardGetStringA

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiBoardGetStringA(
    __IN     uint32_t  Id ,          /* Name Id */
    __OUT    char      *pBuffer ,    /* Destination pBuffer */
    __INOUT uint32_t *pBufLen       /* pBuffer Length */
);
```

Description

Text information about the system and manufacturer details for diagnostic and inventory purposes.

Parameters

Id

__IN Selects the EAPI Id corresponding to the data. All the data retrieved is in **string format**.

Id	Description
EAPI_ID_BOARD_MANUFACTURER_STR	Board Manufacturer Name
EAPI_ID_BOARD_NAME_STR	Board Name
EAPI_ID_BOARD_SERIAL_STR	Board Serial Number
EAPI_ID_BOARD_BIOS_REVISION_STR	Board BIOS Revision
EAPI_ID_BOARD_HW_REVISION_STR	Hardware Revision
EAPI_ID_BOARD_PLATFORM_TYPE_STR	Board Platform Type
EAPI_SEMA_ID_BOARD_BOOT_VERSION_STR	Bootloader Revision (Only for BMC boards)
EAPI_SEMA_ID_BOARD_APPLICATION_VERSION_STR	Firmware Revision
EAPI_SEMA_ID_BOARD_RESTART_EVENT_STR	Board Restart Event
EAPI_SEMA_ID_BOARD_REPAIR_DATE_STR	Board Repair Date
EAPI_SEMA_ID_BOARD_MANUFACTURE_DATE_STR	Board Manufacturing Date
EAPI_SEMA_ID_BOARD_MAC_1_STRING	Board MAC Address 1
EAPI_SEMA_ID_BOARD_MAC_2_STRING	Board MAC Address 2
EAPI_SEMA_ID_BOARD_2ND_HW_REVISION_STR	Board Secondary Hardware Revision
EAPI_SEMA_ID_BOARD_2ND_SERIAL_STR	Board Secondary Serial Number

***pBuffer**

__OUT Pointer to a buffer that receives the value's data. This parameter can be NULL if the data is not required

***pBufLen**

__INOUT Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the **pBuffer** parameter. When the function returns, this variable contains the size of the data copied to **pBuffer** including the terminating null character.

Return Status Code

Condition	Return Value
pBufLen==NULL or pData==NULL	EAPI_STATUS_INVALID_PARAMETER
Unsupported EAPI Id	EAPI_STATUS_UNSUPPORTED
Failed	EAPI_STATUS_READ_ERROR
Success	EAPI_STATUS_SUCCESS

5.3 Board Values

- Allows access to board-specific parameters such as temperature limits, thresholds, or configuration values.
- These values are typically used for monitoring, diagnostics, and system control logic.

5.3.1 EApiBoardGetValue

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiBoardGetValue(
    __IN  uint32_t Id          , /* Value Id */
    __OUT uint32_t *pValue       /* Return Value */
);
```

Description

Information about the hardware platform in value format

Parameters

Id

__IN Selects the Get Value Sub Function Id

Id	Description	Units/Format
EAPI_ID_GET_EAPI_SPEC_VERSION	EAPI Specification Version used to implement API	Refer **Specification Version Number Format**
EAPI_ID_BOARD_BOOT_COUNTER_VAL	Boot Counter	boots
EAPI_ID_BOARD_RUNNING_TIME_METER_VAL	Running Time Meter	minutes
EAPI_ID_BOARD_LIB_VERSION_VAL	Vendor Specific Library Version	Refer **General Version number Format**
EAPI_ID_HWMON_CPU_TEMP	CPU Temperature	0.1 Kelvins
EAPI_ID_HWMON_BOARD_TEMP	Board Temperature	0.1 Kelvins
EAPI_ID_HWMON_VOLTAGE_VCORE	CPU Core Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_2V5	2.5V Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_3V3	3.3V Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_VBAT	Battery Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_5V	5V Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_5VSB	5V Standby Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_12V	12V Voltage	millivolts
EAPI_ID_HWMON_FAN_CPU	CPU Fan speed	RPM
EAPI_ID_HWMON_FAN_SYSTEM	System Fan speed	RPM

EAPI_SEMA_ID_BOARD_POWER_UP_TIME	Get the operating time after power up	seconds
EAPI_SEMA_ID_BOARD_RESTART_EVENT	Get the restart event	Refer **Restart Event**
EAPI_SEMA_ID_BOARD_CAPABILITIES	Get the capabilities of this board or system	Refer **Firmware Capability**
EAPI_SEMA_ID_BOARD_CAPABILITIES_EX	Get the extended EC capabilities	Refer **Extended Firmware Capability**
EAPI_SEMA_ID_BOARD_MIN_TEMP	Board minimum temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_MAX_TEMP	Board maximum temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_STARTUP_TEMP	Board startup temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_CPU_MIN_TEMP	CPU minimum temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_CPU_MAX_TEMP	CPU maximum temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_CPU_STARTUP_TEMP	CPU startup temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_MAIN_CURRENT	Get main power current	Unit: mA, Resolution=1/10 mA
EAPI_SEMA_ID_HWMON_VOLTAGE_GFX_VCORE	GFX voltage	millivolts
EAPI_SEMA_ID_HWMON_VOLTAGE_1V05	1.05V voltage	millivolts
EAPI_SEMA_ID_HWMON_VOLTAGE_1V5	1.5V voltage	millivolts
EAPI_SEMA_ID_HWMON_VOLTAGE_VIN	Vin voltage	millivolts
EAPI_SEMA_ID_HWMON_FAN_SYSTEM_2	2nd system fan	RPM
EAPI_SEMA_ID_HWMON_FAN_SYSTEM_3	3rd system fan	RPM
EAPI_SEMA_ID_BOARD_2ND_SYSTEM_TEMP	2nd Board temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_2ND_SYSTEM_MIN_TEMP	2nd Board minimum temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_2ND_SYSTEM_MAX_TEMP	2nd Board maximum temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_2ND_SYSTEM_STARTUP_TEMP	2nd Board startup temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_POWER_CYCLE	Power cycle counter	cycles

EAPI_SEMA_ID_BOARD_BMC_FLAG	Status of the EC	Refer **EC/BMC Flags**
EAPI_SEMA_ID_BOARD_BMC_STATUS	Status of the EC (not supported for EC boards)	Information for problem analysis
EAPI_SEMA_ID_IO_CURRENT	IO current	mA
EAPI_SEMA_ID_HWMON_SYSTEM_TEMP	System Temperature	0.1 Kelvins
EAPI_SEMA_ID_HWMON_SYSTEM_MIN_TEMP	System minimum temperature	0.1 Kelvins
EAPI_SEMA_ID_HWMON_SYSTEM_MAX_TEMP	System maximum temperature	0.1 Kelvins
EAPI_SEMA_ID_HWMON_SYSTEM_STARTUP_TEMP	System startup temperature	0.1 Kelvins

***pValue**

__OUT Pointer to a buffer that receives the value.

Return Status Code

Condition	Return Value
pValue==NULL	EAPI_STATUS_INVALID_PARAMETER
Hardware monitor device handle access failed	EAPI_STATUS_UNSUPPORTED
Unsupported EAPI Id	EAPI_STATUS_UNSUPPORTED
Failed	EAPI_STATUS_READ_ERROR
Success	EAPI_STATUS_SUCCESS

5.4 Watchdog Control

The **Runtime Watchdog** is used to recover from operating system hangs, application software hangs, or other system malfunctions. When the runtime watchdog timer expires, the EC/BMC triggers a system reset and automatically stops the runtime watchdog.

The **Powerup Watchdog** is used to handle system reset or boot failures. The reset cycle is repeated until the system successfully boots and the application services the watchdog.

5.4.1 EApiWDogGetCap

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiWDogGetCap (
    __OUTOPT uint32_t   *pMaxDelay    ,    /* Max. supported
                                              delay in sec */
    __OUTOPT uint32_t *pMaxEventTimeout,/* Max. supported Event
                                              Timeout in sec */
    __OUTOPT uint32_t  *pMaxResetTimeout /*  Max. supported Reset
                                              Timeout in sec */
);
```

Description

Get the supported maximum delay, event timeout and reset value of the Runtime Watchdog timer.

Parameters

***pMaxDelay**

__OUTOPT Pointer to a buffer that receives maximum supported initial delay time of the watchdog timer in seconds.

***pMaxEventTimeout**

__OUTOPT Pointer to a buffer that receives maximum supported event timeout of the watchdog timer in seconds.

***pMaxResetTimeout**

__OUTOPT Pointer to a buffer that receives maximum supported reset timeout of the watchdog timer in seconds.

Return Status Code

Condition	Return Value
pMaxDelay==NULL && pMaxEventTimeout==NULL && pMaxResetTimeout==NULL	EAPI_STATUS_INVALID_PARAMETER
Success	EAPI_STATUS_SUCCESS
Read capabilities fail	EAPI_STATUS_READ_ERROR

5.4.2 EApiWDogStart

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiWDogStart (
    __IN uint32_t Delay,            /* Delay in sec */
    __IN uint32_t EventTimeout,     /* Event Timeout in sec */
    __IN uint32_t ResetTimeout      /* Reset Timeout in sec */
);
```

Description

Start the runtime watchdog timer and set the parameters.

Parameters

delay

__IN Initial delay for the watchdog timer in seconds. (currently not supported by SEMA EAPI, set it as 0)

EventTimeout

__IN Watchdog timeout interval in seconds to trigger an event. (currently not supported by SEMA EAPI, set it as 0)

ResetTimeout

__IN Watchdog timeout interval in seconds to trigger a reset.
(Supported range 1-65535 seconds)

Return Status Code

Condition	Return Value
ResetTimeout not within the supported range	EAPI_STATUS_INVALID_PARAMETER
Watchdog device handle access fail	EAPI_STATUS_UNSUPPORTED
Watchdog timer already active/running	EAPI_STATUS_RUNNING
Success	EAPI_STATUS_SUCCESS
Watchdog start timer failed	EAPI_STATUS_WRITE_ERROR

5.4.3 EApiWDogTrigger

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiWDogTrigger(void);
```

Description

Trigger the Runtime watchdog timer to previously set timeout value.

Parameter

None.

Return Status Code

Condition	Return Value
Watchdog device handle access fail	EAPI_STATUS_UNSUPPORTED
Watchdog timer not active/running	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS
Watchdog trigger failed	EAPI_STATUS_WRITE_ERROR

5.4.4 EApiWDogStop

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiWDogStop(void);
```

Description

Stops the operation of the runtime watchdog timer.

Parameter

None.

Return Status Code

Condition	Return Value
Watchdog device handle access fail	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Watchdog stop failed	EAPI_STATUS_WRITE_ERROR

5.4.5 EApiPwrUpWDogStart

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiPwrUpWDogStart (
    __IN uint32_t  Timeout         /* Timeout in sec */
);
```

Description

Start the Powerup watchdog timer and set the parameters.

Parameter

ResetTime

__IN Powerup watchdog timeout interval in seconds to trigger a reset. (Supported Range 60-65535).

Return Status Code

Condition	Return Value
Timeout not within the supported range	EAPI_STATUS_INVALID_PARAMETER
Watchdog device handle access fail	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS
Powerup watchdog start failure	EAPI_STATUS_WRITE_ERROR

5.4.6 EApiPwrUpWDogStop

Function definition

```
uint32_t
EAPI_CALLTYPE
 EApiPwrUpWDogStop(void);
```

Description

Stops the operation of the powerup watchdog timer

Parameter

None

Return Status Code

Condition	Return Value
Watchdog device handle access fail	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS
Powerup watchdog stop failure	EAPI_STATUS_WRITE_ERROR

5.5 Storage Access

Provides read and write access to non-volatile storage such as on-board EEPROM storage **(Eg: EEPROM type: Microchip AT25256B)** connected to the EC/BMC.

Storage Id

Id	Description
EAPI_ID_STORAGE_STD	User Region
EAPI_ID_STORAGE_SCR	Secure Region (Unsupported for BMC boards)
EAPI_ID_STORAGE_ODM	ODM Region (Unsupported for BMC boards)

Region	Address range	Base address	Valid Address to Read/Write	Block size
1-User	0x0000 to 0x03FF (1KB)	0x0000	0 to 1020	4
2-Secure	0x6000 to 0x67FF (2KB)	0x6000	0 to 2044	4
3-ODM	0x0C00 to 0x0FFF (1KB)	0x0C00	0 to 1020	4

5.5.1 EApiStorageCap

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiStorageCap (
    __IN   uint32_t Id,                 /* Storage Area Id*/
    __OUT uint32_t *pStorageSize,       /* Max. Storage Size */
    __OUT uint32_t *pBlockLength        /* Max. block length */
);
```

Description

Get the maximum storage size and block length of the selected storage area.

Parameters

Id

__IN Refer **Storage Id.**

***pStorageSize**

__OUT Pointer to a buffer that receives storage area size.

***pBlockLength**

__OUT Pointer to a buffer that receives the storage area alignment/block size.

Return Status Code

Condition	Return Value
Unsupported Storage Id	EAPI_STATUS_UNSUPPORTED
pStorageSize == NULL && pBlockLength == NULL	EAPI_STATUS_INVALID_PARAMETER
Sysfs file Read Error	EAPI_STATUS_READ_ERROR
Success	EAPI_STATUS_SUCCESS

5.5.2 EApiStorageAreaRead

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiStorageAreaRead(
    __IN  uint32_t Id,        /* Storage Area Id*/
    __IN  uint32_t Offset,    /* Byte Offset */
    __OUT void     *pBuffer, /* Pointer to Data pBuffer */
    __IN  uint32_t BufLen,    /* Data pBuffer Size in
                                 bytes */
    __IN  uint32_t ByteCnt    /* Number of bytes to read*/
);
```

Description

Reads string data from the selected storage area.

Parameters

Id

__IN Refer **Storage Id**.

Offset

__IN Storage area start address offset in bytes.

***pBuffer**

__OUT Pointer to a buffer that receives the read data.

BufLen

__IN Size, in bytes, of the buffer pointed to by the pBuffer parameter.

ByteCnt

__IN Size, in bytes, of the information read to the buffer pointed to by the pBuffer parameter.

Return Status Code

Condition	Return Value
Unsupported Storage Id	EAPI_STATUS_UNSUPPORTED
Storage access library not initialized	EAPI_STATUS_NOT_INITIALIZED
pBuffer == NULL || ByteCnt == 0 || BufLen == 0	EAPI_STATUS_INVALID_PARAMETER
ByteCnt > BufLen	EAPI_STATUS_MORE_DATA
Offset + ByteCnt > StorageSize	EAPI_STATUS_INVALID_BLOCK_LENGTH
Storage area device handle access failed	EAPI_STATUS_ERROR
Storage area data read failed	EAPI_STATUS_READ_ERROR
Success	EAPI_STATUS_SUCCESS

5.5.3 EApiStorageAreaWrite

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiStorageAreaWrite (
    __IN uint32_t Id,        /* Storage Area Id*/
    __IN uint32_t Offset,    /* Byte Offset */
    __IN    void *pBuffer,   /* Pointer to Data pBuffer */
    __IN uint32_t ByteCnt    /* Data pBuffer Size in
                                bytes */
);
```

Description

Writes string data to the selected storage area.

Parameters

Id

__IN Refer **Storage Id**.

Offset

__IN Storage area start address offset in bytes.

***pBuffer**

__IN Pointer to a buffer containing the data to be stored.

ByteCnt

__IN Size, in bytes, of the information pointed to by the pBuffer parameter.

Return Status Code

Condition	Return Value
Unsupported Storage Id	EAPI_STATUS_UNSUPPORTED
Storage access library not initialized	EAPI_STATUS_NOT_INITIALIZED
ByteCnt == 0 || pBuffer == NULL	EAPI_STATUS_INVALID_PARAMETER
Offset + ByteCnt > StorageSize	EAPI_STATUS_INVALID_BLOCK_LENGTH
(Offset % pBlockLength) != 0	EAPI_STATUS_INVALID_BLOCK_ALIGNMENT
Storage area device handle access failed	EAPI_STATUS_ERROR
Storage area data write failed	EAPI_STATUS_WRITE_ERROR
Success	EAPI_STATUS_SUCCESS

5.5.4 EApiStorageHexRead

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiStorageHexRead(
     __IN uint32_t  Id      , /* Storage Area Id */
     __IN uint32_t Offset   , /* Byte Offset */
     __OUT    void *pBuffer , /* Pointer to Data pBuffer */
     __IN uint32_t BufLen   , /* Data buffer size in
                                 bytes*/
     __IN uint32_t Bytecnt    /*Number of bytes to
                                 count */
);
```

Description

Reads data in hexadecimal from the selected storage area.

Parameters

Id

__IN Refer **Storage Id**.

Offset

__IN Storage area start address offset in bytes.

***pBuffer**

__OUT Pointer to a buffer that receives the read data in hexadecimal.

BufLen

__IN Size, in bytes, of the buffer pointed to by the pBuffer parameter.

Bytecnt

__IN Size, in bytes, of the information read to the buffer pointed to by the pBuffer.

Return Status Code

Condition	Return Value
Unsupported Storage Id	EAPI_STATUS_UNSUPPORTED
Storage access library not initialized	EAPI_STATUS_NOT_INITIALIZED
pBuffer == NULL || ByteCnt == 0 || BufLen == 0	EAPI_STATUS_INVALID_PARAMETER
ByteCnt > BufLen	EAPI_STATUS_MORE_DATA
Offset + ByteCnt > StorageSize	EAPI_STATUS_INVALID_BLOCK_LENGTH
Storage area device handle access failed	EAPI_STATUS_ERROR
Storage area data read failed	EAPI_STATUS_READ_ERROR
Success	EAPI_STATUS_SUCCESS

5.5.5 EApiStorageHexWrite

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiStorageHexWrite(
     __IN   uint32_t   Id,       /* Storage Area Id */
     __IN   uint32_t   Offset,   /* Byte Offset */
     __IN   void   *pBuffer, /* Pointer to Data pBuffer */
     __IN   uint32_t ByteCnt /* Data buffer size in
                                bytes*/
);
```

Description

Writes data in hexadecimal to the selected storage area.

Parameters

Id

__IN Refer **Storage Id**.

Offset

__IN Storage area start address offset in bytes.

***pBuffer**

__OUT Pointer to a buffer that receives the read data in hexadecimal.

BufLen

__IN Size, in bytes, of the buffer pointed to by the pBuffer parameter.

ByteCnt

__IN Size, in bytes, of the information pointed to by the pBuffer.

Return Status Code

Condition	Return Value
Unsupported Storage Id	EAPI_STATUS_UNSUPPORTED
Storage access library not initialized	EAPI_STATUS_NOT_INITIALIZED
ByteCnt == 0 || pBuffer == NULL	EAPI_STATUS_INVALID_PARAMETER
Offset + ByteCnt > StorageSize	EAPI_STATUS_INVALID_BLOCK_LENGTH
(Offset % pBlockLength) != 0	EAPI_STATUS_INVALID_BLOCK_ALIGNMENT
Storage area device handle access failed	EAPI_STATUS_ERROR
Storage area data write failed	EAPI_STATUS_WRITE_ERROR

5.5.6 EApiStorageUnLock

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiStorageUnLock(
     __IN   uint32_t   Id         ,   /* Storage Area Id */
     __IN   uint32_t   Permission, /* Storage Area Permission */
     __IN   char    *passcode    /* Storage Area Passcode */
);
```

Description

Unlock the Secure or ODM storage region to access data.

Parameters

Id

__IN Only Secure or ODM regions. Refer **Storage Id**.

Permission

__IN Supports 2 permissions.

Id	Description
1	Read only
2	Read / Write

***passcode**

__IN Pointer to a buffer containing passcode to unlock the selected region.

Return Status Code

Condition	Return Value
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid Storage Id	EAPI_STATUS_INVALID_PARAMETER
Unsupported functionality	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Failed	EAPI_STATUS_ERROR

5.5.7 EApiStorageLock

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiStorageLock(
     __IN   uint32_t  Id          /* Storage Area Id */
);
```

Description

To lock the Secure or ODM storage region.

Parameters

Id

__IN Only Secure or ODM Storage Id. Refer **Storage Id**.

Return Status Code

Condition	Return Value
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid Storage Id	EAPI_STATUS_INVALID_PARAMETER
Unsupported functionality	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Failed	EAPI_STATUS_ERROR

5.5.8 EApiGUIDWrite

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiGUIDWrite(
     __IN   uint32_t   Id,        /* Storage Area Id */
     __IN   uint32_t   Offset,    /* Byte Offset */
     __IN   void   *pBuffer,      /* Pointer to Data pBuffer */
     __IN   uint32_t   ByteCnt    /* Data buffer size in
                                     bytes*/
);
```

Description

Writes type 4 - UUID in hexadecimal to the selected storage area. The GUID can be read using **EApiStorageHexRead.**

Parameters

Id

__IN Only ODM Storage Id. Refer **Storage Id**.

Offset

__IN Offset **0x100** of ODM region is used for storing GUID.

***pBuffer**

__IN Pointer to a buffer array of 16 elements containing the GUID [in hexadecimal] to be stored.

ByteCnt

__IN Size, in bytes, of the information pointed to by the pBuffer. [ByteCnt=16]

Return Status Code

Condition	Return Value
Unsupported Storage Id	EAPI_STATUS_UNSUPPORTED
Storage access library not initialized	EAPI_STATUS_NOT_INITIALIZED
Offset + ByteCnt > StorageSize	EAPI_STATUS_INVALID_BLOCK_LENGTH
Storage area device handle access failed	EAPI_STATUS_ERROR
Failed	EAPI_STATUS_WRITE_ERROR
Success	EAPI_STATUS_SUCCESS

5.6 Voltage Monitor

- Allows monitoring of system and component voltage levels in real time.
- Helps detect power irregularities and ensures operation within safe voltage ranges.

5.6.1 EApiBoardGetVoltageCap

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiBoardGetVoltageCap(
    __OUTOPT  uint32_t *value,      /* buffer */
);
```

Description

To display the capabilities of voltage monitor.

Parameters

***value**

__OUTOPT Pointer to buffer that receives the voltage capability value.

Return Status Code

Condition	Return Value
Success	EAPI_STATUS_SUCCESS
Voltage cap read failed	EAPI_STATUS_ERROR

5.6.2 EApiBoardGetVoltageMonitor

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiBoardGetVoltageMonitor(
    __IN    uint32_t  id,              /* Channel Id */
    __OUT   uint32_t  *mVolts,         /* Value buffer */
    __OUT   char   *pBuf               /* Description Buffer */
    __OUT   uint32_t  size             /* size of buffer */
);
```

Description

To get the voltage values and its description.

Parameters

channel

__IN Select the channel ID [channel range 0 -15].

***mVolts**

__OUT Pointer to buffer that receives the voltage value.

***pBuf**

__OUT Pointer to buffer that receives the voltage description string.

size

__IN Size of pBuf in bytes.

Return Status Code

Condition	Return Value
Invalid buffers	EAPI_STATUS_INVALID_PARAMETER
Unsupported voltage channel Id	EAPI_STATUS_UNSUPPORTED
Voltage read failed	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

5.7 GPIO

SEMA helps control general purpose I/O pins connected to the board controller (EC/BMC).

5.7.1 EApiGPIOGetDirectionCaps

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiGPIOGetDirectionCaps(
     __IN     uint32_t   Id      ,     /* GPIO Id*/
     __OUTOPT uint32_t   *pInputs,     /* Supported GPIO
                                       Output Bit Mask */
     __OUTOPT uint32_t   *pOutputs    /* Supported GPIO
                                       Output Bit Mask */
);
```

Description

Reads the capabilities of the current GPIO implementation from the selected GPIO interface. The ports where both input and output bit masks are 1 are GPIOs.

Parameters

Id

__IN GPIO Id (currently Id= 0 only is supported by SEMA EAPI).

***pInputs**

__OUTOPT Pointer to a buffer that receives the bit mask of the supported inputs.

***pOutputs**

__OUTOPT Pointer to a buffer that receives the bit mask of the supported outputs.

Return Status Code

Condition	Return Value
Unsupported GPIO Functionality / Id	EAPI_STATUS_UNSUPPORTED
Bitmask==NULL	EAPI_STATUS_INVALID_PARAMETER
Success	EAPI_STATUS_SUCCESS

5.7.2 EApiGPIOGetDirection

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiGPIOGetDirection(
     __IN  uint32_t   Id         , /* GPIO Id*/
     __IN  uint32_t   Bitmask    , /* Bitmask of affected bits */
     __OUT uint32_t   *pDirection  /* Current Direction */
);
```

Description

Reads the current configuration of the selected GPIO pins.

Parameters

Id

__IN GPIO Id. (currently Id= 0 only is supported by SEMA EAPI).

Bitmask

__IN Bitmask Only selected bits are returned. Unselected bits return **0.**

***pDirection**

__OUT Pointer to a buffer that receives the direction of the supported GPIO ports. Bits with the value **1** are inputs, bits with **0** are outputs.

Return Status Code

Condition	Return Value
GPIO library not initialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported GPIO Functionality / Bitmask	EAPI_STATUS_UNSUPPORTED
Bitmask==NULL / pDirection==NULL	EAPI_STATUS_INVALID_PARAMETER
Failed	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

5.7.3 EApiGPIOSetDirection

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiGPIOSetDirection(
     __IN   uint32_t    Id        , /* GPIO Id*/
     __IN   uint32_t   Bitmask    , /* Bitmask of affected bits */
     __IN   uint32_t    Direction   /* Direction */
);
```

Description

Sets the configuration of the selected GPIO pins.

Parameters

Id

__IN GPIO Id (currently Id= 0 only is supported by SEMA EAPI).

Bitmask

__IN Bitmask. The bits for which the direction is to be changed are set to 1 and other bits are set to 0.

Direction

__IN Sets the direction of the selected GPIO ports. Bits with the value **1** are inputs, bits with **0** are outputs.

Return Status Code

Condition	Return Value
GPIO library not initialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported GPIO Functionality	EAPI_STATUS_UNSUPPORTED
Bitmask==NULL	EAPI_STATUS_INVALID_PARAMETER
GPIO device handle access failed	EAPI_STATUS_ERROR
GPIO direction write failed	EAPI_STATUS_WRITE_ERROR
Success	EAPI_STATUS_SUCCESS

5.7.4 EApiGPIOGetLevel

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiGPIOGetLevel(
     __IN   uint32_t    Id         , /* GPIO Id*/
     __IN   uint32_t   Bitmask     , /* Bitmask of affected bits*/
     __OUT  uint32_t   *pLevel      /* Current Level */
);
```

Description

Reads level from the GPIO pins.

Parameters

Id

__IN GPIO Id (currently Id= 0 only is supported by SEMA EAPI).

Bitmask

__IN Bitmask. Only selected bits are returned. Unselected bits return **0.**

***pLevel**

__OUT Pointer to a buffer that receives the GPIO level. Results can be read on a bit level. Bits with the value **1** are high, bits with **0** are low.

Return Status Code

Condition	Return Value
GPIO library not initialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported GPIO Functionality	EAPI_STATUS_UNSUPPORTED
Bitmask==NULL	EAPI_STATUS_INVALID_PARAMETER
Failed	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

5.7.5 EApiGPIOSetLevel

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiGPIOSetLevel(
     __IN  uint32_t  Id        , /* GPIO Id*/
     __IN  uint32_t  Bitmask   , /* Bitmask of affected bits */
     __OUT uint32_t  Level       /* Current Level */
);
```

Description

Write to GPIO ports. Only level of the GPO pin can be changed.

Parameters

Id

__IN GPIO Id (currently Id= 0 only is supported by SEMA EAPI).

Bitmask

__IN Value for a bit mask. Only selected bits are changed. Unselected bits remain unchanged.

Level

__IN Input level of the selected GPIO port. Output for single ports is on a bit level. Bits with the value **1** are high, bits with **0** are low.

Return Status Code

Condition	Return Value
GPIO library not initialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported GPIO Functionality	EAPI_STATUS_UNSUPPORTED
Bitmask==NULL	EAPI_STATUS_INVALID_PARAMETER
GPIO device handle access failed	EAPI_STATUS_ERROR
Failed	EAPI_STATUS_WRITE_ERROR
Success	EAPI_STATUS_SUCCESS

5.8. Generic I2C Access

Supported only for EC boards. The SEMA I2C functions Provides low-level I2C read and write access to devices connected on the I2C bus.

I2C Bus Id

Id	Description
SEMA_EXT_IIC_BUS_1	Extended I2C bus 1
SEMA_EXT_IIC_BUS_2	Extended I2C bus 2
SEMA_EXT_IIC_BUS_3	Extended I2C bus 3
SEMA_EXT_IIC_BUS_4	Extended I2C bus 4

I2C Cmd Type

CmdType	Description	Encoding Condition
EAPI_I2C_NO_CMD	No command/index	Bit 30 of **Cmd** must be 1
EAPI_I2C_ENC_STD_CMD	Extended standard 8 bits CMD	Original 32 bit **Cmd**
EAPI_I2C_ENC_EXT_CMD	Extended standard 10 bits CMD	Bit 31 of **Cmd** must be 1

5.8.1 EApiI2CGetBusCap

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiI2CGetBusCap(
     __IN uint32_t   Id          , /* I2C Bus Id*/
     __OUT uint32_t *pMaxBlkLen    /* I2C Bus Status */
 );
```

Description

Returns maximum block length if the selected I2C bus is supported.

Parameters

Id

__IN Refer **I2C Bus Id.**

pMaxBlkLen

__OUT size in bytes. Pointer to a buffer that receives the maximum transfer block length for the given interface.

Note: Maximum length of data byte to write is **29 bytes** and to read is **32 bytes.**

Return Status Code

Condition	Return Value
pMaxBlkLen == NULL	EAPI_STATUS_INVALID_PARAMETER
Unsupported I2C functionality / I2C bus Id	EAPI_STATUS_UNSUPPORTED
Failed	EAPI_STATUS_READ_ERROR
Success	EAPI_STATUS_SUCCESS

5.8.2 EApiI2CGetBusSts

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiI2CGetBusSts(
     __IN  uint32_t  Id          , /* I2C Bus Id*/
     __OUT uint8_t   *Bus_Sts      /* I2C Bus Status */
);
```

Description

Reflects the I2C status of the most recently used I2C transaction

Parameters

Id

__IN Refer **I2C Bus Id.**

***Bus_Sts**

__OUT Pointer to a buffer that receives the data.

Id	Bit 7
Transaction cannot complete normally Eg: - The device does not respond ACK - Pull down in the bus - Any other error	0
Transaction complete normally Eg: - Transfer complete - The device respond ACK	1

Return Status Code

Condition	Return Value
Bus_sts==NULL	EAPI_STATUS_INVALID_PARAMETER
Unsupported I2C functionality / I2C bus Id	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS

5.8.3 EApiI2CProbeDevice

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiI2CProbeDevice(
     __IN uint32_t Id ,       /* I2C Bus Id*/
     __IN uint32_t Addr       /* 8-bit I2C Device Address */
 );
```

Description

Returns EAPI_STATUS_SUCCESS if the device with selected I2C address is present on the specified I2C bus.

Parameters

Id

__IN Refer **I2C Bus Id.**

Addr

__IN 8-bit I2C slave device address.

Return Status Code

Condition	Return Value
Unsupported I2C functionality / I2C bus Id	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Failed	EAPI_STATUS_READ_ERROR

5.8.4 EApiI2CReadTransfer

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiI2CReadTransfer(
     IN   uint32_t   Id       , /* I2C Bus Id*/
     IN   uint32_t   Addr     , /* 8-Bit I2C
                                   Device Address */
     IN   uint32_t   Cmd      , /* No of Bytes to write*/
     OUT  void      *pBuffer  , /* Read Data pBuffer */
     IN   uint32_t   BufLen   , /* Data pBuffer Length */
     IN   uint32_t   ByteCnt    /* Number of Bytes to Read */
);
```

Description

Reads a specific register in the selected I2C device.

Parameters

Id

__IN Refer **I2C Bus Id.**

Addr

__IN 8-bit I2C device address.

Cmd

__IN Encoded I2C device command / index. Refer **I2C Cmd Type** for command encoding.

***pBuffer**

__OUT Pointer to a buffer that receives the read data.

BufLen

__IN Size (in bytes), of the information pointed to by the **pBuffer** parameter.

ByteCnt

__IN Size (in bytes), of the information read to the buffer pointed by the **pBuffer** parameter.

Return Status Code

Condition	Return Value
Unsupported I2C functionality / I2C bus Id	EAPI_STATUS_UNSUPPORTED
I2C device handle access failed	EAPI_STATUS_READ_ERROR
pBuffer == NULL || ByteCnt == 0 || BufLen == 0 || ByteCnt > MAX_BLOCK || ByteCnt > BufLen	EAPI_STATUS_INVALID_PARAMETER
Success	EAPI_STATUS_SUCCESS
Failure	EAPI_STATUS_ERROR

5.8.5 EApiI2CWriteTransfer

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiI2CWriteTransfer(
      __IN uint32_t Id,          /* I2C Bus Id*/
      __IN uint32_t Addr,        /* 8 Bit I2C Device Address */
      __IN unit32_t Cmd,         /* I2C Command/Offset */
      __IN void     *pBuffer,    /* Data pBuffer */
      __IN uint32_t ByteCnt      /* Byte Count to write */
);
```

Description

Write to a specific register in the selected I2C device.
Writes to an I2C device at the I2C address **(Addr)** the number of bytes **(ByteCnt)** from the buffer **(*pBuffer)** while using the device specific command **(Cmd)**.
Depending on the addressed I2C device command **(Cmd)** can be a specific command or a byte offset.

Parameters

Id

__IN Refer **I2C Bus Id.**

Addr

__IN 8-bit I2C device address.

Cmd

__IN Encoded I2C device command / index. Refer **I2C Cmd Type** for command encoding.

***pBuffer**

__IN Pointer to a buffer containing the data to be transferred.

ByteCnt

__IN Size (in bytes), of the information pointed to by the **pBuffer** parameter.

Return Status Code

Condition	Return Value
Unsupported I2C functionality	EAPI_STATUS_UNSUPPORTED
I2C device handle access failed	EAPI_STATUS_READ_ERROR
pBuffer == NULL || ByteCnt == 0	EAPI_STATUS_INVALID_PARAMETER
ByteCnt > MAX_BLOCK	EAPI_STATUS_INVALID_BLOCK_LENGTH
Success	EAPI_STATUS_SUCCESS

5.8.6 EApiI2CWriteReadRaw

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiI2CWriteReadRaw(
     IN   uint32_t   Id,          /* I2C Bus Id*/
     IN   uint8_t    Addr,        /* 8Bit I2C Device Address */
     INOPT  void     *pWBuffer,   /* Write Data pWBuffer*/
     IN   uint32_t   WriteBCnt,   /* Number of Bytes to write*/
     OUTOPT  void    *pRBuffer,   /* Read Data pRBuffer */
     IN   uint32_t    RBufLen,    /* Data pRBuffer Length*/
     IN   uint32_t    ReadBCnt    /* Number of Bytes to Read */
);
```

Description

Universal function used for combined write/read transactions (**I2C repeat start** feature).

Parameters

Id __IN **Refer I2C Bus Id.**

Addr

__IN 8-bit I2C slave device address.

***pWBuffer**

__INOPT Pointer to a buffer containing the data to be transferred. This parameter can be NULL if the data is not required. 0th index of pWBuffer holds the encoded **Cmd**.

WriteBCnt

__IN Size, in bytes, of the information pointed to by the pWBuffer parameter.

***pRBuffer**

__OUTOPT Pointer to a buffer that receives the read data.

RBufLen

__IN Size, in bytes, of the buffer pointed to by the pRBuffer parameter.

ReadBCnt

__IN Size (in bytes), specifying the number of bytes to read.

Return Status Code

Condition	Return Value
Unsupported I2C functionality	EAPI_STATUS_UNSUPPORTED
Invalid ReadBCnt, RBufLen or WriteBCnt	EAPI_STATUS_INVALID_PARAMETER
WriteBCnt / ReadBCnt greater than MaxBlockLen	EAPI_STATUS_INVALID_BLOCK_LENGTH
Write Failure	EAPI_STATUS_WRITE_ERROR
Read Failure	EAPI_STATUS_READ_ERROR
Success	EAPI_STATUS_SUCCESS

5.9. Error Log Description

- Retrieves hardware and firmware error logs recorded by the system.
- Useful for post-failure analysis, debugging, and reliability assessment.

5.9.1 EApiBoardGetErrorLog

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiBoardGetErrorLog(
     __IN    uint32_t    position,          /* Exception code */
     __OUT   uint32_t    *ErrorNumber,
     __OUT   uint8_t     *Flags,
     __OUT   uint8_t     *RestartEvent,
     __OUT   uint32_t    *PwrCycles,
     __OUT   uint32_t    *Bootcount,
     __OUT   uint32_t    *Time,
     __OUT   uint8_t     *Status,
     __OUT   signed char *CPUtemp,
     __OUT   signed char *Boardtemp
     __OUT   uint32_t     *TotalOnTime,
     __OUT   uint8_t      *BiosSel
 );
```

Description

Get error number history of selected ports from the EC/BMC. The error log buffer stores the power sequence issues, information about the actual state and counters for better tracking of the issues.
The latest entry in the error log buffer is always found on position 0. The previous entry is found on position 1 and so on.

Parameters

position

__IN Position points to the log to be read.

***ErrorNumber**

__OUT Pointer to buffer that stores the Error number. To get the detailed description, use **EApiGetErrorNumberDescription.**

***Flags**

Exception Code, selected BIOS and Power Mode.

***RestartEvent**

__OUT Pointer to buffer that stores the system restart event.

***PwrCycles**

__OUT Pointer to buffer that stores the value of power cycles counter.

***Bootcount**

__OUT Pointer to buffer that stores the value of boot counter.

***Time**

__OUT Pointer to buffer that stores the value of ontime counter in seconds.

***Status**

__OUT Pointer to buffer that stores the value of BMC status information.

***CPUtemp**

__OUT Pointer to buffer that stores the value of current CPU temperature in celsius.

***Boardtemp**

__OUT Pointer to buffer that stores the value of current board temperature in celsius.

***TotalOnTime**

__OUT Pointer to buffer that stores the value of total on time in minutes.

***BiosSel**

__OUT Pointer to buffer that stores the value of BIOS Selected.

Return Status Code

Condition	Return Value
Success	EAPI_STATUS_SUCCESS
If Buffers are NULL	EAPI_STATUS_INVALID_PARAMETER
Write position failed	EAPI_STATUS_WRITE_ERROR
Read Error log failed	EAPI_STATUS_READ_ERROR

5.9.2 EApiBoardGetCurPosErrorLog

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiBoardGetCurPosErrorLog (
    __OUT  uint32_t     *ErrorNumber,
    __OUT  uint8_t      *Flags,
    __OUT  uint8_t      *RestartEvent,
    __OUT  uint32_t     *PwrCycles,
    __OUT  uint32_t     *Bootcount,
    __OUT  uint32_t     *Time,
    __OUT  uint8_t      *Status,
    __OUT  signed char  *CPUtemp,
    __OUT  signed char  *Boardtemp
    __OUT  uint32_t    *TotalOnTime,
    __OUT  uint8_t      *BiosSel
);
```

Description

Get latest error number history from EC/BMC. The error log buffer stores the power sequence issues, information about the actual state and counters for better tracking of the issues.

Parameters

***ErrorNumber**

__OUT Pointer to buffer that stores the Error number. To get the detailed description, please use **EApiGetErrorNumberDescription.**

***Flags**

Exception Code, selected BIOS and Power Mode

***RestartEvent**

__OUT Pointer to buffer that stores the system restart event.

***PwrCycles**

__OUT Pointer to buffer that stores the value of power cycles counter.

***Bootcount**

__OUT Pointer to buffer that stores the value of boot counter.

***Time**

__OUT Pointer to buffer that stores the value of ontime counter in seconds.

***Status**

__OUT Pointer to buffer that stores the value of status information from BMC status command.

***CPUtemp**

__OUT Pointer to buffer that stores the value of current CPU temperature in Celsius.

***Boardtemp**

__OUT Pointer to buffer that stores the value of current board temperature in Celsius.

***TotalOnTime**

__OUT Pointer to buffer that stores the value of total on time in minutes.

***BiosSel**

__OUT Pointer to buffer that stores the value of BIOS Selected.

Return Status Code

Condition	Return Value
Success	EAPI_STATUS_SUCCESS
If buffers are NULL	EAPI_STATUS_INVALID_PARAMETER
Read Error log failed	EAPI_STATUS_READ_ERROR

5.9.3 EApiBoardGetErrorNumDesc

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiBoardGetErrorNumDesc(
    __IN   uint32_t  Pos,      /* Pos */
    __OUT  char      *pBuf     /* Buffer */
    __OUT  uint32_t   size     /* Buffer size */
);
```

Description

To get the description of the given error number.

Parameters

Pos

__IN Position number of the log to be read.

***pBuf**

__OUT Pointer to buffer that receives the description string.

size

__OUT Size of pBuf in bytes.

Return Status Code

Condition	Return Value
Success	EAPI_STATUS_SUCCESS
Invalid buffer	EAPI_STATUS_INVALID_PARAMETER
Position write failed	EAPI_STATUS_WRITE_ERROR
Read Error log failed	EAPI_STATUS_READ_ERROR

5.10. BIOS Source

Allows configuring the BIOS source mode such as **External BIOS**, **Standard BIOS**, or **Fail-Safe BIOS**.

5.10.1 EApiGetBiosSource

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiGetBiosSource(
    __OUT   uint8_t   *data       /* Buffer */
);
```

Description

To get the BIOS Source.

Parameters

***data**

__OUT Pointer to buffer that receives the data.

Return Status Code

Condition	Return Value
Invalid data buffer	EAPI_STATUS_INVALID_PARAMETER
Success	EAPI_STATUS_SUCCESS
BIOS source read failed	EAPI_STATUS_READ_ERROR

5.10.2 EApiSetBiosSource

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiSetBiosSource(
    __IN   uint8_t data,          /* Buffer */
);
```

Description

Set the BIOS Source.

Parameters

data

__IN Pointer to buffer that stores the data.

Return Status Code

Condition	Return Value
Failed	EAPI_STATUS_WRITE_ERROR
Success	EAPI_STATUS_SUCCESS

5.10.3 EApiGetBiosStatus

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiGetBiosStatus(
     __OUT   uint8_t *data         /* Buffer */
 );
```

Description

Displays the BIOS status.

Parameters

***data**

__OUT Pointer to buffer that receives the data.

Return Status Code

Condition	Return Value
Invalid data buffer	EAPI_STATUS_INVALID_PARAMETER
Success	EAPI_STATUS_SUCCESS
BIOS source read failed	EAPI_STATUS_READ_ERROR

5.11. Smart Fan

- Allows monitoring and control of fan speed based on temperature or predefined policies.
- Helps optimize cooling performance while reducing noise and power consumption.

Fan Id

Fan Id	Description
0	CPU Fan
1	System fan 1
2	System fan 2 (Unsupported for EC boards)
3	System fan 3 (Unsupported for EC boards)

Fan Mode

Fan Mode	Description
0	Auto
1	Off
2	On
3	Soft

Fan Temperature Source

TempSrc	Description
0	CPU Sensor
1	Board Sensor

For System fan 1 of COM-HPC boards,

TempSrc	Description
0	CPU Sensor
1	Carrier Sensor

5.11.1. EApiSmartFanSetTempSetpoints

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiSmartFanSetTempSetpoints(
     __IN int     id,              /* Fan Id */
     __IN int     Level1,          /* Buffer */
     __IN int     Level2,          /* Buffer */
     __IN int     Level3,          /* Buffer */
     __IN int     Level4           /* Buffer */
 );
```

Description

To set temperature setpoints. The valid temp level range is from -128 to 128.

Parameters

id

__IN Refer **Fan Id.**

Level1

__IN Pointer to buffer that stores the temperature level 1 setpoint.

Level2

__IN Pointer to buffer that stores the temperature level 2 setpoint.

Level3

__IN Pointer to buffer that stores the temperature level 3 setpoint.

Level4

__IN Pointer to buffer that stores the temperature level 4 setpoint.

Return Status Code

Condition	Return Value
Invalid Fan Id or Temperature setpoint level out of range	EAPI_STATUS_INVALID_PARAMETER
adl-bmc-hwmon driver not loaded successfully / Fan functionality unsupported	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Temperature points write failed	EAPI_STATUS_WRITE_ERROR

5.11.2. EApiSmartFanGetTempSetpoints

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiSmartFanSetTempSetpoints(
    __IN int       id,          /* Fan Id */
    __OUT  int     *pLevel1,    /* Buffer */
    __OUT  int     *pLevel2,    /* Buffer */
    __OUT  int     *pLevel3,    /* Buffer */
    __OUT  int     *pLevel4     /* Buffer */
);
```

Description

To get temperature set points.

Parameters

id

__IN Refer **Fan Id.**

***pLevel1**

__IN Pointer to buffer that receives the temperature level 1.

***pLevel2**

__IN Pointer to buffer that receives the temperature level 2.

***pLevel3**

__IN Pointer to buffer that receives the temperature level 3.

***pLevel4**

__IN Pointer to buffer that receives the temperature level 4.

Return Status Code

Condition	Return Value
adl-bmc-hwmon driver not loaded successfully / Fan Id unsupported	EAPI_STATUS_UNSUPPORTED
Sysfs access error	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS
Failed	EAPI_STATUS_READ_ERROR

5.11.3. EApiSmartFanSetPWMSetpoints

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiSmartFanSetPWMSetpoints (
    __IN   int     id,              /* Fan Id */
    __IN   int     pwm_Level1,      /* Buffer */
    __IN   int     pwm_Level2,      /* Buffer */
    __IN   int     pwm_Level3,      /* Buffer */
    __IN   int     pwm_Level4       /* Buffer */
);
```

Description

To set PWM setpoints. The valid PWM range is from 0 to 100.

Parameters

id

__IN Refer **Fan Id.**

***pwm_Level1**

__IN Pointer to buffer that receives the PWM level 1.

***pwm_Level2**

__IN Pointer to buffer that receives the PWM level 2.

***pwm_Level3**

__IN Pointer to buffer that receives the PWM level 3.

***pwm_Level4**

__IN Pointer to buffer that receives the PWM level 4.

Return Status Code

Condition	Return Value
Invalid Fan Id or PWM setpoint out of range	EAPI_STATUS_INVALID_PARAMETER
adl-bmc-hwmon driver not loaded successfully / Fan functionality unsupported	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Failed	EAPI_STATUS_WRITE_ERROR

5.11.4. EApiSmartFanGetPwmSetpoints

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiSmartFanGetPwmSetpoints (
    __IN int      id,                /* Id */
    __OUT  int    *pwm_Level1,       /* Buffer */
    __OUT  int    *pwm_Level2,       /* Buffer */
    __OUT  int    *pwm_Level3,       /* Buffer */
    __OUT  int    *pwm_Level4        /* Buffer */
);
```

Description

To get PWM setpoints.

Parameters

id

__IN Refer **Fan Id.**

***pLevel1**

__IN Pointer to buffer that receives the PWM level 1.

***pLevel2**

__IN Pointer to buffer that receives the PWM level 2.

***pLevel3**

__IN Pointer to buffer that receives the PWM level 3.

***pLevel4**

__IN Pointer to buffer that receives the PWM level 4.

Return Status Code

Condition	Return Value
Invalid Fan Id / Invalid PWM level buffers	EAPI_STATUS_INVALID_PARAMETER
adl-bmc-hwmon driver not loaded successfully / Fan functionality unsupported	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Failed	EAPI_STATUS_READ_ERROR

5.11.5. EApiSmartFanSetMode

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiSmartFanSetMode (
    __IN  int id,                    /* Fan Id */
    __IN  int fan_mode               /* Buffer */
);
```

Description

To set fan mode

Parameters

id

__IN Refer **Fan Id**.

fan_mode

__IN Refer **Fan Mode**.

Return Status Code

Condition	Return Value
id == NULL || fan_mode == NULL	EAPI_STATUS_INVALID_PARAMETER
adl-bmc-hwmon driver not loaded successfully / Fan functionality unsupported	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Failed	EAPI_STATUS_WRITE_ERROR

5.11.6. EApiSmartFanGetMode

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiSmartFanGetMode (
    __IN  int  id,              /* Fan Id */
    __OUT int *fan_mode         /* Buffer */
);
```

Description

To get fan mode.

Parameters

id

__IN Refer **Fan Id**.

***fan_mode**

__IN Refer **Fan Mode**.

Return Status Code

Condition	Return Value
id == NULL || fan_mode == NULL	EAPI_STATUS_INVALID_PARAMETER
adl-bmc-hwmon driver not loaded successfully / Fan functionality unsupported	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Failed	EAPI_STATUS_READ_ERROR

5.11.7. EApiSmartFanSetTempSrc

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiSmartFanSetTempSrc(
    __IN  int  id,                  /* Fan Id */
    __IN  int  TempSrc              /* Buffer */

);
```

Description

To set Temperature source of specific fan id.

Parameters

id

__IN Refer **Fan Id**.

TempSrc

__IN Refer **Fan Temperature Source**.

Return Status Code

Condition	Return Value
Unsupported Fan ID or TempSrc	EAPI_STATUS_INVALID_PARAMETER
Fan Driver not loaded or Unsupported	EAPI_STATUS_UNSUPPORTED
Failed	EAPI_STATUS_WRITE_ERROR
Success	EAPI_STATUS_SUCCESS

5.11.8. EApiSmartFanGetTempSrc

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiSmartFanSetMode(
    __IN   int   id,              /* Id */
    __OUT  int   *pTempSrc        /* Buffer */
    );
```

Description

To get Temperature source of specific fan id

Parameters

id

__IN Refer **Fan Id**.

***pTempSrc**

__IN Refer **Fan Temperature Source.**

Return Status Code

Condition	Return Value
Invalid Fan Id / Invalid temperature source buffer	EAPI_STATUS_INVALID_PARAMETER
adl-bmc-hwmon driver not loaded successfully / Fan functionality unsupported	EAPI_STATUS_UNSUPPORTED
Fan mode read failed	EAPI_STATUS_READ_ERROR
Success	EAPI_STATUS_SUCCESS

5.12 Backlight

- Enables control of display backlight brightness levels.
- Used to adjust visibility and reduce power consumption in embedded displays.

Backlight Id

Id	Description
EAPI_ID_BACKLIGHT_1	Backlight Local Flat Panel 1

PWM Level

Backlight Enable values	Description
1	Requests/Signifies that the Backlight be Enabled
0	Requests/Signifies that the Backlight be Disabled

5.12.1 EApiVgaSetBacklightEnable

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiVgaSetBacklightEnable (
     __IN   uint32_t Id          , /* Backlight Id */
     __IN   uint32_t Enable        /* Backlight Enable */
 );
```

Description

Enables the backlight of the selected flat panel display and set brightness level 200 by default.

Parameters

Id

__IN **Backlight Id.**

Enable

__IN Refer **PWM Level** Pointer to a buffer that stores the backlight enable state.

Return Status Code

Condition	Return Value
Unsupported Backlight Id	EAPI_STATUS_UNSUPPORTED
Enable == NULL	EAPI_STATUS_INVALID_PARAMETER
Success	EAPI_STATUS_SUCCESS
Write Failure	EAPI_STATUS_WRITE_ERROR

5.12.2 EApiVgaGetBacklightEnable

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiVgaGetBacklightEnable (
     __IN  uint32_t Id          , /* Backlight Id */
     __OUT uint32_t *pEnable      /* Backlight Enable */
 );
```

Description

Get the backlight enable status of the selected flat panel display.

Parameters

Id

__IN Refer **Backlight Id.**

*pEnable

__IN Refer **PWM Level** Pointer to a buffer that receives the backlight enable state.

Return Status Code

Condition	Return Value
Unsupported Backlight Id	EAPI_STATUS_UNSUPPORTED
pEnable == NULL	EAPI_STATUS_INVALID_PARAMETER
Success	EAPI_STATUS_SUCCESS
Read Failure	EAPI_STATUS_READ_ERROR

5.12.3 EApiVgaSetBacklightBrightness

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiVgaSetBacklightBrightness (
    __IN   uint32_t   Id         , /* Backlight Id */
    __IN   uint32_t   Bright       /* Backlight Brightness */
);
```

Description

Sets the brightness of the selected flat panel display.

Parameters

Id

__IN Refer **Backlight Id.**

Bright

__IN Backlight brightness level. The range of PWM Level is from 0 to 255.

Return Status Code

Condition	Return Value
Unsupported Backlight Id	EAPI_STATUS_UNSUPPORTED
Bright == NULL	EAPI_STATUS_INVALID_PARAMETER
Backlight sysfs file read failed	EAPI_STATUS_READ_ERROR
Write Failure	EAPI_STATUS_WRITE_ERROR
Backlight disabled	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

5.12.4 EApiVgaGetBacklightBrightness

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiVgaGetBacklightBrightness (
    __IN  uint32_t  Id         , /* Backlight Id */
    __OUT uint32_t  *pBright     /* Backlight Brightness */
);
```

Description

Reads the current brightness of the selected flat panel display

Parameters

Id

__IN Refer **Backlight Id.**

***pBright**

__OUT Pointer to a buffer that receives the current backlight brightness level.

Return Status Code

Condition	Return Value
Unsupported Backlight Id	EAPI_STATUS_UNSUPPORTED
pBright == NULL	EAPI_STATUS_INVALID_PARAMETER
Backlight brightness read failed	EAPI_STATUS_READ_ERROR
Backlight disabled	EAPI_STATUS_ERRROR
Success	EAPI_STATUS_SUCCESS

5.13. Exception Description

- Provides detailed descriptions of error and exception codes returned by APIs.
- Helps developers quickly diagnose failures and understand error conditions.

5.13.1. EApiBoardGetExcepDesc

Function definition

```
uint32_t
EAPI_CALLTYPE
EApiBoardGetExcepDesc(
    __IN  uint32_t  Exceptioncode, /* Exception code */
    __OUT char      *pBuf          /* Buffer */
    __INOUT  uint32_t  size           /* size in bytes */
);
```

Description

To get text information of the Exception code.

Parameters

Exceptioncode

__IN The Error number (Exception Code).

***pBuf**

__OUT Pointer to buffer that receives Error number.

size

__INOUT Size of pBuf in bytes.

Return Status Code

Condition	Return Value
Invalid buffer	EAPI_STATUS_INVALID_PARAMETER
Exception code write failed	EAPI_STATUS_WRITE_ERROR
Read exception description failed	EAPI_STATUS_READ_ERROR
Success	EAPI_STATUS_SUCCESS

5.14. SMBus Slave Device Access

- Provides access to SMBus devices for reading and writing data.
  Commonly used to communicate with power management devices, sensors, and controllers.

5.14.1. EApiSMBReadTrans

Function Definition

```
uint32_t
EAPI_CALLTYPE
EApiSMBReadTrans (
     __IN  uint32_t  Addr  , /* 7-bit SMBus Device Address */
     __IN  uint32_t  Cmd   , /* SMBus Device command */
     __OUT void   *pBuffer  , /* Read Data pBuffer */
     __IN  uint32_t  nByteCnt /* Number of Bytes to Read*/
);
```

Description

Reads from a specific register in the selected SMBus device.
Reads from SMBus device at the SMBus address **(Addr)** and stores in the buffer **pBuffer** while using the device specific command **Cmd.**

Parameters

Addr

__ IN Encoded 7-bit SMBus Slave device address.

Cmd

__IN SMBus Device Command / Index.

***pBuffer**

__OUT Pointer to a buffer that receives the read data.

nByteCnt

__IN Size (in bytes), of the information read to the buffer pointed to by pBuffer.
nByteCnt is 2 for ReadWord operation.
nByteCnt is 1 for ReadByte operation.

Return Status

Condition	Return Value
Invalid pbuffer or nByteCnt	EAPI_STATUS_INVALID_PARAMETER
Sysfs access failed / **adl-bmc-i2c** driver not loaded successfully	EAPI_STATUS_UNSUPPORTED
SMBus Read failed	EAPI_STATUS_READ_ERROR
Success	EAPI_STATUS_SUCCESS

5.14.2. EApiSMBWriteTrans

Function Definition

```
uint32_t
EAPI_CALLTYPE
EApiSMBWriteTrans (
     __IN uint32_t   Addr  , /* 7-bit SMBUS Device Address */
     __IN uint32_t   Cmd   , /* I2C Command/Offset */
     __IN void      *pBuffer,/* Data buffer */
     __IN uint32_t   nByteCnt /* Bytes Count to Write */
);
```

Description

Writes to a specific register in the selected SMBus device.

Parameters

Addr

__ IN Encoded 7-bit SMBus Slave device address.

Cmd

__IN SMBus Device Command / Index.

***pBuffer**

__IN Pointer to a buffer that stores the data to be read.

nByteCnt

__IN Size (in bytes), of the buffer pointed to by the pBuffer parameter.

Return Status

Condition	Return Value
Invalid pbuffer or nByteCnt	EAPI_STATUS_INVALID_PARAMETER
nByteCnt greater than maximum supported block length	EAPI_STATUS_INVALID_BLOCK_LENGTH
Sysfs access failed / **adl-bmc-i2c** driver not loaded successfully	EAPI_STATUS_UNSUPPORTED
SMBus Write failed	EAPI_STATUS_WRITE_ERROR
Success	EAPI_STATUS_SUCCESS

6. For Windows Platforms

6.1 Initialization Functions

6.1.1 SemaEApiLibInitialize

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiLibInitialize (void);
```

Description

General initialization of the EAPI. Prior to calling any EAPI function the library needs to be initialized by calling this function.
The status code for all EAPI function will be EAPI_STATUS_NOT_INITIALIZED unless this function is called.

Parameters

None.

Return Status

Condition	Return Value
Mutex Creation failed	EAPI_STATUS_ALLOC_ERROR
Mutex timeout	EAPI_STATUS_ERROR
EC/BMC chip not detected	EAPI_STATUS_NOT_FOUND
Initialization failed	EAPI_STATUS_NOT_INITIALIZED
Initialization success	EAPI_STATUS_INITIALIZED

6.1.2 SemaEApiUnInitialize

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiUnInitialize (void);
```

Description

General function to uninitialize the EAPI library. Should be called before program exit. Closes the driver handles and releases OS resources (file descriptors, mutexes, threads, etc).

Parameters

None.

Return Status

Condition	Return Value
Success	EAPI_STATUS_SUCCESS

6.2 Board Information

SEMA EAPI functions provide an interface to control or get board's information, including

- Static board and manufacturer information (e.g. BIOS version, Manufacturer name)
- Failure forensics (e.g. restart event)

6.2.1 SemaEApiBoardGetStringA

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiBoardGetStringA (
      __IN       uint32_t    Id,          /* EAPI ID */
      __OUT      uint8_t     *pData,      /* Data buffer */
      __INOUT    uint32_t    *pBufLen     /* Buffer Length */
);
```

Description

Text information about the system and manufacturer details for diagnostic and inventory purposes.

Parameters

Id

__IN Selects the EAPI Id corresponding to the data. All the data retrieved is in **string format**.

Id	Description
EAPI_ID_BOARD_MANUFACTURER_STR	Board Manufacturer Name
EAPI_ID_BOARD_NAME_STR	Board Name
EAPI_ID_BOARD_SERIAL_STR	Board Serial Number
EAPI_ID_BOARD_BIOS_REVISION_STR	Board BIOS Revision
EAPI_ID_BOARD_HW_REVISION_STR	Hardware Revision
EAPI_ID_BOARD_PLATFORM_TYPE_STR	Board Platform Type
EAPI_SEMA_ID_BOARD_BOOT_VERSION_STR	Bootloader Revision (Only for BMC boards)
EAPI_SEMA_ID_BOARD_APPLICATION_VERSION_STR	Firmware Revision
EAPI_SEMA_ID_BOARD_RESTART_EVENT_STR	Board Restart Event
EAPI_SEMA_ID_BOARD_REPAIR_DATE_STR	Board Repair Date
EAPI_SEMA_ID_BOARD_MANUFACTURE_DATE_STR	Board Manufacturing Date
EAPI_SEMA_ID_BOARD_MAC_1_STRING	Board MAC Address 1
EAPI_SEMA_ID_BOARD_MAC_2_STRING	Board MAC Address 2
EAPI_SEMA_ID_BOARD_2ND_HW_REVISION_STR	Board Secondary Hardware Revision
EAPI_SEMA_ID_BOARD_2ND_SERIAL_STR	Board Secondary Serial Number

***pData**

__OUT Pointer to a buffer that receives the data.

***pBufLen**

__INOUT Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the **pBuffer** parameter.

When the function returns, this variable contains the size of the data copied to **pBuffer** including the terminating null character.

Return Status

Condition	Return Value
Library not initialized	EAPI_STATUS_NOT_INITIALIZED
pBufLen == NULL || pBufLen == 0 || pData == NULL	EAPI_STATUS_INVALID_PARAMETER
pBufLen is less than data retrieved	EAPI_STATUS_MORE_DATA
Unsupported EAPI Id	EAPI_STATUS_UNSUPPORTED
Failed	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.3 Board Values

- Allows access to real-time system and hardware parameters [corresponds to a specific sensor reading or board metric such as temperature, Voltage, etc].
- These values are typically used for monitoring and diagnostics.

6.3.1 SemaEApiBoardGetValue

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiBoardGetValue (
      __IN       uint32_t    Id,          /* EAPI ID */
      __OUT      uint32_t    *pData       /* Value buffer */
);
```

Description

Read the real-time system and hardware parameters [corresponds to a specific sensor reading or board metric].

Parameters

Id

__IN Selects the EAPI Id corresponding to the data.

Id	Description	Units/Format
EAPI_ID_GET_EAPI_SPEC_VERSION	EAPI Specification Version used to implement API	Refer **Specification Version Number Format**
EAPI_ID_BOARD_BOOT_COUNTER_VAL	Boot Counter	boots
EAPI_ID_BOARD_RUNNING_TIME_METER_VAL	Running Time Meter	minutes
EAPI_ID_BOARD_LIB_VERSION_VAL	Vendor Specific Library Version	Refer **General Version number Format**
EAPI_ID_HWMON_CPU_TEMP	CPU Temperature	0.1 Kelvins
EAPI_ID_HWMON_BOARD_TEMP	Board Temperature	0.1 Kelvins
EAPI_ID_HWMON_VOLTAGE_VCORE	CPU Core Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_2V5	2.5V Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_3V3	3.3V Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_VBAT	Battery Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_5V	5V Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_5VSB	5V Standby Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_12V	12V Voltage	millivolts
EAPI_ID_HWMON_FAN_CPU	CPU Fan speed	RPM

EAPI_ID_HWMON_FAN_SYSTEM	System Fan speed	RPM
EAPI_SEMA_ID_BOARD_POWER_UP_TIME	Get the operating time after power up	seconds
EAPI_SEMA_ID_BOARD_RESTART_EVENT	Get the restart event	Refer **Restart Event**
EAPI_SEMA_ID_BOARD_CAPABILITIES	Get the capabilities of this board or system	Refer **Firmware Capability**
EAPI_SEMA_ID_BOARD_CAPABILITIES_EX	Get the extended EC capabilities	Refer **Extended Firmware Capability**
EAPI_SEMA_ID_BOARD_MIN_TEMP	Board minimum temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_MAX_TEMP	Board maximum temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_STARTUP_TEMP	Board startup temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_CPU_MIN_TEMP	CPU minimum temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_CPU_MAX_TEMP	CPU maximum temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_CPU_STARTUP_TEMP	CPU startup temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_MAIN_CURRENT	Get main power current	Unit: mA, Resolution=1/10 mA
EAPI_SEMA_ID_HWMON_VOLTAGE_GFX_VCORE	GFX voltage	millivolts
EAPI_SEMA_ID_HWMON_VOLTAGE_1V05	1.05V voltage	millivolts
EAPI_SEMA_ID_HWMON_VOLTAGE_1V5	1.5V voltage	millivolts
EAPI_SEMA_ID_HWMON_VOLTAGE_VIN	Vin voltage	millivolts
EAPI_SEMA_ID_HWMON_FAN_SYSTEM_2	2nd system fan	RPM
EAPI_SEMA_ID_HWMON_FAN_SYSTEM_3	3rd system fan	RPM
EAPI_SEMA_ID_BOARD_2ND_SYSTEM_TEMP	2nd Board temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_2ND_SYSTEM_MIN_TEMP	2nd Board minimum temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_2ND_SYSTEM_MAX_TEMP	2nd Board maximum temperature	0.1 Kelvins
EAPI_SEMA_ID_BOARD_2ND_SYSTEM_STARTUP_TEMP	2nd Board startup temperature	0.1 Kelvins

EAPI_SEMA_ID_BOARD_POWER_CYCLE	Power cycle counter	cycles
EAPI_SEMA_ID_BOARD_BMC_FLAG	Status of the EC	Refer **EC/BMC Flags**
EAPI_SEMA_ID_BOARD_BMC_STATUS	Status of the EC (not supported for EC boards)	Information for problem analysis
EAPI_SEMA_ID_IO_CURRENT	IO current	mA
EAPI_SEMA_ID_HWMON_SYSTEM_TEMP	System Temperature	0.1 Kelvins
EAPI_SEMA_ID_HWMON_SYSTEM_MIN_TEMP	System minimum temperature	0.1 Kelvins
EAPI_SEMA_ID_HWMON_SYSTEM_MAX_TEMP	System maximum temperature	0.1 Kelvins
EAPI_SEMA_ID_HWMON_SYSTEM_STARTUP_TEMP	System startup temperature	0.1 Kelvins

***pData**

__OUT Pointer to a buffer that receives the data.

Return Status

Condition	Return Value
Library not initialized	EAPI_STATUS_NOT_INITIALIZED
pData == NULL	EAPI_STATUS_INVALID_PARAMETER
Unsupported EAPI Id / Functionality	EAPI_STATUS_UNSUPPORTED
Mutex Failure / Read Value Failed	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.4. Watchdog Control

The **Runtime Watchdog** is used to recover from operating system hangs, application software hangs, or other system malfunctions. When the runtime watchdog timer expires, the EC/BMC triggers a system reset and automatically stops the runtime watchdog.

The **Powerup Watchdog** is used to handle system reset or boot failures. The reset cycle is repeated until the system successfully boots and the application services the watchdog.

6.4.1. SemaEApiWDogGetCap

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiWDogGetCap (
    __OUTOPT uint32_t *pMaxDelay    ,   /* Max. supported
                                           delay in sec */
    __OUTOPT uint32_t *pMaxEventTimeout,/* Max. supported Event
                                           Timeout in sec */
    __OUTOPT uint32_t *Resetvalue      /* Max. supported Reset
                                           Timeout in sec */
);
```

Description

Get the supported maximum delay, event timeout and reset value of the Runtime Watchdog timer.

Parameters

***pMaxDelay**

__OUTOPT Pointer to a buffer that receives maximum supported initial delay time of the watchdog timer in seconds.

***pMaxEventTimeout**

__OUTOPT Pointer to a buffer that receives maximum supported event timeout of the watchdog timer in seconds.

***Resetvalue**

__OUTOPT Pointer to a buffer that receives maximum supported reset timeout of the watchdog timer in seconds.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
pMaxDelay==NULL && pMaxEventTimeout==NULL && Resetvalue==NULL	EAPI_STATUS_INVALID_PARAMETER
Success	EAPI_STATUS_SUCCESS
Runtime watchdog unsupported	EAPI_STATUS_UNSUPPORTED

6.4.2. SemaEApiWDogStart

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiWDogStart (
    __IN uint32_t delay,          /* Delay in sec */
    __IN uint32_t EventTimeout,   /* Event Timeout in sec */
    __IN uint32_t ResetTime       /* Reset Timeout in sec */
);
```

Description

Start the runtime watchdog timer and set the parameters.

Parameters

delay

__IN Initial delay for the watchdog timer in seconds. (currently not supported by SEMA EAPI, set it as 0).

EventTimeout

__IN Watchdog timeout interval in seconds to trigger an event. (currently not supported by SEMA EAPI, set it as 0).

ResetTime

__IN Watchdog timeout interval in seconds to trigger a reset. (Supported range 1-65535 seconds)

Return Status

Condition	Return Value
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Runtime watchdog unsupported	EAPI_STATUS_UNSUPPORTED
Invalid delay or EventTimeout	EAPI_STATUS_INVALID_PARAMETER
ResetTime not within the supported range	EAPI_STATUS_INVALID_PARAMETER
Watchdog timer already active/running	EAPI_STATUS_RUNNING
Mutex failure / Watchdog Start Failure	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.4.3. SemaEApiWDogTrigger

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiWDogTrigger (void);
```

Description

Trigger the Runtime watchdog timer to previously set timeout value.

Parameters

None.

Return Status

Condition	Return Value
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Runtime watchdog unsupported	EAPI_STATUS_UNSUPPORTED
Mutex failure / Watchdog timer not started / Watchdog Trigger failed	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.4.4. SemaEApiWDogStop

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiWDogStop (void);
```

Description

Stops the operation of the Runtime watchdog timer.

Parameters

None.

Return Status

Condition	Return Value
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Runtime watchdog unsupported	EAPI_STATUS_UNSUPPORTED
Mutex failure / Watchdog timer not started / Watchdog Trigger failed	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.4.5. SemaEApiPwrUpWDogStart

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiPwrUpWDogStart (
    __IN uint32_t ResetTime  /* Timeout in sec */
);
```

Description

Start the Powerup watchdog timer and set the parameters.

Parameters

ResetTime

__IN Powerup Watchdog timeout interval in seconds to trigger a reset. (Supported Range 60-65535).

Return Status

Condition	Return Value
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Powerup watchdog unsupported	EAPI_STATUS_UNSUPPORTED
ResetTime not within the supported range	EAPI_STATUS_INVALID_PARAMETER
Mutex failure / Watchdog Start Failure	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.4.6. SemaEApiPwrUpWDogStop

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiPwrUpWDogStop (void);
```

Description

Stops the operation of the Powerup watchdog timer.

Parameters

None.

Return Status

Condition	Return Value
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Runtime watchdog unsupported	EAPI_STATUS_UNSUPPORTED
Watchdog timer not started	EAPI_STATUS_WRITE_ERROR
Mutex failure / Watchdog Trigger failed	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.5. Storage Access

Provides read and write access to non-volatile storage such as on-board EEPROM storage **(Eg: EEPROM type: Microchip AT25256B)** connected to the EC/BMC.

Storage Id

Id	Description
EAPI_ID_STORAGE_STD	User Region
EAPI_ID_STORAGE_SCR	Secure region (Unsupported for BMC Boards)
EAPI_ID_STORAGE_ODM	ODM Region (Unsupported for BMC Boards)

Region	Address range	Base address	Valid Address to Read/Write	Block size
1-User	0x0000 to 0x03FF (1KB)	0x0000	0 to 1020	4
2-Secure	0x6000 to 0x67FF (2KB)	0x6000	0 to 2044	4
3-ODM	0x0C00 to 0x0FFF (1KB)	0x0C00	0 to 1020	4

6.5.1. SemaEApiStorageCap

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiStorageCap (
     __IN    uint32_t   Id,              /* Channel Id */
     __OUT   uint32_t   *pStorageSize, /* Value buffer */
     __OUT   uint32_t   *pBlockLength);/* Total write block length
                                          & alignment*/
```

Description

Get the maximum supported storage size and block length of the selected storage area.

Parameters

Id

__IN Refer **Storage Id**.

***pStorageSize**

__OUT Pointer to a buffer that receives storage area size.

***pBlockLength**

__OUT Pointer to a buffer that receives the storage area alignment/block size.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported Storage Id or Invalid buffers	EAPI_STATUS_INVALID_PARAMETER
Unsupported Id	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS

6.5.2. SemaEApiStorageAreaRead

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiStorageAreaRead(
      __IN uint32_t Id,        /* Storage Area Id*/
      __IN uint32_t Offset,    /* Byte Offset */
      __OUT    void *pBuffer, /* Pointer to Data pBuffer */
      __IN uint32_t nBufLen,   /* pBuffer size in bytes */
      __IN uint32_t nByteCnt   /* Number of bytes to read */
);
```

Description

Reads string data from the selected storage area.

Parameters

Id

__IN Refer **Storage Id**.

Offset

__IN Storage area start address offset in bytes.

***pBuffer**

__OUT Pointer to a buffer that receives the read data.

nBufLen

__IN Size, in bytes, of the buffer pointed to by the pBuffer parameter

nByteCnt

__IN Size, in bytes, of the information read to the buffer pointed to by the pBuffer parameter.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported Storage area Id or (pBuffer == NULL || nBufLen < nByteCnt || ByteCnt < 0)	EAPI_STATUS_INVALID_PARAMETER
(nByteCnt % BlockLength) != 0	EAPI_STATUS_INVALID_BLOCK_LENGTH
(nByteCnt + nOffset) > StorageSize	EAPI_STATUS_MORE_DATA
Mutex Failure / Read failure	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.5.3. SemaEApiStorageAreaWrite

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiStorageAreaWrite(
    __IN uint32_t Id,        /* Storage Area Id*/
    __IN uint32_t nOffset,   /* Byte Offset */
    __IN     void *pBuffer,  /* Pointer to Data pBuffer */
    __IN uint32_t nByteCnt   /* Number of bytes to read */
);
```

Description

Writes string data to the selected storage area.

Parameters

Id

__IN Refer **Storage Id**.

nOffset

__IN Storage area start address offset in bytes.

***pBuffer**

__IN Pointer to a buffer containing the data to be stored.

nByteCnt

__IN Size, in bytes, of the information pointed to by the pBuffer parameter.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Storage Access Unsupported	EAPI_STATUS_UNSUPPORTED
Invalid Storage area Id or (pBuffer == NULL || nByteCnt < 0)	EAPI_STATUS_INVALID_PARAMETER
nOffset != 0 && (nOffset % BlockLength) != 0	EAPI_STATUS_INVALID_BLOCK_ALIGNMENT
nByteCnt % BlockLength != 0	EAPI_STATUS_INVALID_BLOCK_LENGTH
(nByteCnt + nOffset) > StorageSize	EAPI_STATUS_MORE_DATA
Mutex Failure / Write failure	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.5.4. SemaEApiStorageHexRead

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiStorageHexRead(
          __IN uint32_t Id,        /* Storage Area Id*/
          __IN uint32_t Offset,    /* Byte Offset */
          __OUT    void *pBuffer, /* Pointer to Data pBuffer */
          __IN uint32_t nBufLen,    /* Data pBuffer Size in
                                       bytes */
          __IN uint32_t nByteCnt    /* Number of bytes to
                                       read */
);
```

Description

Reads data in hexadecimal from the selected storage area.

Parameters

Id

__IN Refer **Storage Id**.

Offset

__IN Storage area start address offset in bytes.

***pBuffer**

__OUT Pointer to a buffer that receives the read data in hexadecimal.

BufLen

__IN Size, in bytes, of the buffer pointed to by the pBuffer parameter.

ByteCnt

__IN Size, in bytes, of the information read to the buffer pointed to by the pBuffer parameter.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported Storage area Id or (pBuffer == NULL || nBufLen < nByteCnt || ByteCnt < 0)	EAPI_STATUS_INVALID_PARAMETER
(nByteCnt % BlockLength) != 0	EAPI_STATUS_INVALID_BLOCK_LENGTH
(nByteCnt + nOffset) > StorageSize	EAPI_STATUS_MORE_DATA
Mutex Failure / Read failure	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.5.5. SemaEApiStorageHexWrite

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiStorageHexWrite(
      __IN uint32_t Id,        /* Storage Area Id*/
      __IN uint32_t nOffset,   /* Byte Offset */
      __IN     char *pBuffer,  /* Pointer to Data pBuffer */
      __IN uint32_t nByteCnt   /* Number of bytes to read */
);
```

Description

Writes data in hexadecimal to the selected storage area.

Parameters

Id

__IN Refer **Storage Id**.

nOffset

__IN Storage area start address offset in bytes.

***pBuffer**

__IN Pointer to a buffer that stores the data to be written.

nByteCnt

__IN Size, in bytes, of the information pointed to by the pBuffer parameter.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Storage Access Unsupported	EAPI_STATUS_UNSUPPORTED
Invalid Storage area Id or (pBuffer == NULL || nByteCnt < 0)	EAPI_STATUS_INVALID_PARAMETER
nOffset != 0 && (nOffset % BlockLength) != 0	EAPI_STATUS_INVALID_BLOCK_ALIGNMENT
nByteCnt % BlockLength != 0	EAPI_STATUS_INVALID_BLOCK_LENGTH
(nByteCnt + nOffset) > StorageSize	EAPI_STATUS_MORE_DATA
Mutex Failure / Write failure	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.5.6. SemaEApiStorageUnlock

Function definition

```
uint32_t
SEMAEAPI_API
SemaEApiStorageUnlock(
     __IN   uint32_t   Id,          /* Storage Area Id */
     __IN   uint32_t   Permission,  /* Storage Area Permission */
     __IN   char       *Password    /* Storage Area Passcode */
     );
```

Description

Unlock the Secure or ODM storage region to access data. (Currently supported only for EC boards)

Parameters

Id

__IN Only Secure or ODM regions. Refer **Storage Id**.

Permission

__IN Supports 2 permissions.

Id	Description
1	Read only
2	Read / Write

***Password**

__IN Pointer to a buffer containing passcode to unlock the selected region

Return Status Code

Condition	Return Value
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid Storage Id	EAPI_STATUS_INVALID_PARAMETER
Unsupported functionality / Permission	EAPI_STATUS_UNSUPPORTED
Invalid Password	EAPI_STATUS_WRITE_ERROR
EC is busy	EAPI_STATUS_TIMEOUT
Success	EAPI_STATUS_SUCCESS
Mutex failure / Unlock failed	EAPI_STATUS_ERROR

6.5.7. SemaEApiStorageLock

Function definition

```
uint32_t
SEMAEAPI_API
SemaEApiStorageLock(
     __IN   uint32_t  Id          /* Storage Area Id */
);
```

Description

Lock the Secure or ODM storage region to protect data access. (Currently supported only for EC boards)

Parameters

Id

__IN Only Secure or ODM regions. Refer **Storage Id**.

Return Status Code

Condition	Return Value
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid Storage Id	EAPI_STATUS_INVALID_PARAMETER
Unsupported functionality / Permission	EAPI_STATUS_UNSUPPORTED
EC is busy	EAPI_STATUS_TIMEOUT
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Lock failed	EAPI_STATUS_ERROR

6.5.8. EApiGUIDWrite

Function definition

```
uint32_t
SEMAEAPI_API
EApiGUIDWrite(
     __IN   uint32_t   Id,        /* Storage Area Id */
     __IN   uint32_t   nOffset,   /* Byte Offset */
     __IN   void      *pBuffer,   /* Pointer to Data pBuffer */
     __IN   uint32_t   nByteCnt /* Data buffer size in
                                   bytes*/
     );
```

Description

Writes UUID type 4 in hexadecimal to the selected storage area. The GUID can be read using **SemaEApiStorageHexRead.**

NOTE: The ODM region should be unlocked before calling this function

Parameters

Id

__IN Refer ODM region in **Storage Id**.

nOffset

__IN Offset **0x100** of ODM region is used for storing GUID.

***pBuffer**

__IN Pointer to a buffer array of 16 elements containing the GUID [in hexadecimal] to be stored.

nByteCnt

__IN Size, in bytes, of the information pointed to by the pBuffer parameter [nByteCnt=16]

Return Status Code

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported Storage area Id	EAPI_STATUS_UNSUPPORTED
(nByteCnt + nOffset) > StorageSize	EAPI_STATUS_MORE_DATA
Mutex Failure / Write failure	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.6. Voltage Monitor

- Allows monitoring of system and component voltage levels in real time.
- Helps detect power irregularities and ensures operation within safe voltage ranges.

6.6.1. SemaEApiBoardGetVoltageMonitor

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiBoardGetVoltageMonitor (
     __IN    uint32_t  channel, /* Channel Id */
     __OUT   uint32_t  *pValue, /* Value buffer */
     __OUT   char   *pBuffer    /* Description Buffer */
);
```

Description

Get the Voltage description and values of the selected channel.

Parameters

channel

__IN Select the channel ID [channel range 0 -15].

***pValue**

__OUT Pointer to a buffer that receives the voltage value.

***pBuffer**

__OUT Pointer to a buffer that receives the voltage description string.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
pValue == NULL || pBuffer == NULL	EAPI_STATUS_INVALID_PARAMETER
Unsupported Channel	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Read voltage failure	EAPI_STATUS_ERROR

6.7. GPIO Control

SEMA helps control general purpose I/O pins connected to the board controller (EC/BMC).

6.7.1. SemaEApiGPIOGetDirectionCaps

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiGPIOGetDirectionCaps(
    __IN       uint8_t    GpioId      , /* GPIO Id*/
    __OUTOPT   uint32_t   *pnCapsIn    , /* Supported GPIO
                                           Output Bit Mask */
    __OUTOPT   uint32_t   *pnCapsOut     /* Supported GPIO
                                          Output Bit Mask */
);
```

Description

Reads the capabilities of the current GPIO implementation from the selected GPIO interface. The ports where both input and output bit masks are 1 are GPIOs.

Parameters

GpioId

__IN GPIO Id. Currently **EAPI_ID_GPIO_BANK00** only is supported by SEMA EAPI.

***pnCapsIn**

__OUTOPT Pointer to a buffer that receives the bit mask of the supported inputs.

***pnCapsOut**

__OUTOPT Pointer to a buffer that receives the bit mask of the supported outputs.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
(pnCapsOut == NULL) || (pnCapsIn == NULL)	EAPI_STATUS_INVALID_PARAMETER
Success	EAPI_STATUS_SUCCESS
Mutex Failure	EAPI_STATUS_ERROR
GPIO unsupported	EAPI_STATUS_UNSUPPORTED

6.7.2. SemaEApiGPIOGetDirection

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiGPIOGetDirection(
     __IN  uint8_t  GpioId     , /* GPIO Id*/
     __IN uint32_t  Bitmask    , /* Bitmask of affected bits */
     __OUT uint32_t *pDirection   /* Direction */
);
```

Description

Reads the current configuration of the selected GPIO pins.

Parameters

GpioId

__IN GPIO Id. Currently **EAPI_ID_GPIO_BANK00** only is supported by SEMA EAPI.

Bitmask

__IN Only selected bits are returned. Unselected bits return 0.

***pDirection**

__OUT Pointer to a buffer that receives the direction of the supported GPIO ports. Bits with the value 1 are inputs and bits with value 0 are outputs.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid pDirection or Bitmask	EAPI_STATUS_INVALID_PARAMETER
Bitmask validation failed	EAPI_STATUS_INVALID_BITMASK
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Read direction failed	EAPI_STATUS_ERROR
Unsupported	EAPI_STATUS_UNSUPPORTED

6.7.3. SemaEApiGPIOSetDirection

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiGPIOSetDirection(
     __IN  uint8_t     GpioId   , /* GPIO Id*/
     __IN  uint32_t    Bitmask  , /* Bitmask of affected bits */
     __IN  uint32_t    pDirection  /* Direction buffer */
);
```

Description

Sets the configuration of the selected GPIO pins.

Parameters

GpioId

__IN GPIO Id. Currently **EAPI_ID_GPIO_BANK00** only is supported by SEMA EAPI.

Bitmask

__IN The bits for which the direction is to be changed are set to 1 and other bits are set to 0.

pDirection

__IN Sets the direction of the selected GPIO ports. Bits with the value **1** are inputs, bits with **0** are outputs.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid Bitmask	EAPI_STATUS_INVALID_PARAMETER
Bitmask validation failed	EAPI_STATUS_INVALID_BITMASK
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Set direction failed	EAPI_STATUS_ERROR
Unsupported	EAPI_STATUS_UNSUPPORTED

6.7.4. SemaEApiGPIOGetLevel

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiGPIOGetLevel(
     __IN  uint8_t   GpioId   , /* GPIO Id*/
     __IN  uint32_t  Bitmask  , /* Bitmask of affected bits */
     __OUT uint32_t  *plevel    /* Level */
);
```

Description

Reads the current level of the selected GPIO pins.

Parameters

GpioId

__IN GPIO Id. Currently **EAPI_ID_GPIO_BANK00** only is supported by SEMA EAPI.

Bitmask

__IN Only selected bits are returned. Unselected bits return 0.

***plevel**

__OUT Pointer to a buffer that receives the GPIO level. Results can be read on a bit level. Bits with the value **1** are high, bits with **0** are low.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid Bitmask	EAPI_STATUS_INVALID_PARAMETER
Bitmask validation failed	EAPI_STATUS_INVALID_BITMASK
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Read level failed	EAPI_STATUS_ERROR
Unsupported	EAPI_STATUS_UNSUPPORTED

6.7.5. SemaEApiGPIOSetLevel

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiGPIOSetLevel(
     __IN   uint8_t     GpioId   , /* GPIO Id*/
     __IN   uint32_t    Bitmask , /* Bitmask of affected bits */
     __IN   uint32_t    plevel      /* Level buffer */
);
```

Description

Write to GPIO ports. Depending on the hardware implementation writing multiple GPIO ports with the bit mask option does not guarantee a time synchronous change of the output levels. Only level of the GPO pin can be changed.

Parameters

GpioId

__IN GPIO Id. Currently **EAPI_ID_GPIO_BANK00** only is supported by SEMA EAPI.

Bitmask

__IN The bits for which the direction is to be changed are set to 1 and other bits are set to 0.

plevel

__IN Input level of the selected GPIO port. Only level of the Output pins can be changed.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid Bitmask	EAPI_STATUS_INVALID_PARAMETER
Bitmask validation failed	EAPI_STATUS_INVALID_BITMASK
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Set level failed	EAPI_STATUS_ERROR
Unsupported	EAPI_STATUS_UNSUPPORTED

6.8. Generic I2C Access

The SEMA I2C functions Provides low-level I2C read and write access to devices connected on the I2C bus.

I2C Bus Id

Id	Description
SEMA_EXT_IIC_BUS_1	Extended I2C bus 1
SEMA_EXT_IIC_BUS_2	Extended I2C bus 2
SEMA_EXT_IIC_BUS_3	Extended I2C bus 3
SEMA_EXT_IIC_BUS_4	Extended I2C bus 4

I2C Cmd Type

CmdType	Description	Encoding Condition
EAPI_I2C_NO_CMD	No command/index	Bit 30 of **Cmd** must be 1
EAPI_I2C_ENC_STD_CMD	Extended standard 8 bits CMD	Original 32 bit **Cmd**
EAPI_I2C_ENC_EXT_CMD	Extended standard 10 bits CMD	Bit 31 of **Cmd** must be 1

6.8.1. SemaEApiI2CGetBusCap

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiI2CGetBusCap (
     __IN  uint32_t    Id         , /* I2C Bus Id */
     __OUT uint32_t   *pMaxBlkLen   /* Max BlockLength */
);
```

Description

Returns maximum block length if the selected I2C bus is supported.

Parameters

Id

__IN Refer **I2C Bus Ids**.

***pMaxBlkLen**

__OUT size in bytes. Pointer to a buffer that receives the maximum transfer block length for the given interface.

Note: Max length of data byte to write is **29 Bytes** and to read is **32 Bytes**.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
pMaxBlkLen ==NULL	EAPI_STATUS_INVALID_PARAMETER
Mutex Failure	EAPI_STATUS_ERROR
Success or I2C Bus Id supported	EAPI_STATUS_SUCCESS
I2C Bus Id unsupported	EAPI_STATUS_UNSUPPORTED

6.8.2. SemaEApiI2CGetBusSts

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiI2CGetBusSts (
     __IN     uint32_t    Id      , /* I2C Bus Id */
     __OUT    uint32_t   *pStatus    /* I2C Bus Status */
);
```

Description

Reflects the I2C status of the most recently used I2C transaction.

Parameters

Id

__IN Refer **I2C Bus Ids**.

***pStatus**

__OUT Pointer to a buffer that receives the data.

Id	Bit 7
Transaction cannot complete normally Eg: - The device does not respond ACK - Pull down in the bus - Any other error	0
Transaction complete normally Eg: - Transfer complete - The device respond ACK	1

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid pStatus buffer / Unsupported I2C Bus Id	EAPI_STATUS_UNSUPPORTED
Mutex Failure / Get bus status fail	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.8.3. SemaEApiI2CProbeDevice

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiI2CProbeDevice (
     __IN   uint8_t    Id,        /* I2C Bus Id */
     __IN   uint32_t   Addr       /* Encoded 7/10 Bit I2C Device
                                     Address */
);
```

Description

Returns EAPI_STATUS_SUCCESS if the device with selected I2C address is present on the specified I2C bus.

Parameters

Id

__IN I2C Bus Ids.

Addr

__IN Encoded 7/10-bit I2C device Address.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported I2C Bus Id	EAPI_STATUS_UNSUPPORTED
I2C busy	EAPI_STATUS_BUSY_COLLISION
Mutex Failure / I2C communication failed via EC/BMC	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS
Device not present in I2C bus	EAPI_STATUS_NOT_FOUND

6.8.4. SemaEApiI2CReadTransfer

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiI2CReadTransfer (
     __IN   uint32_t   Id       , /* I2C Bus Id*/
     __IN   uint16_t   Addr     , /* Encoded 7Bit I2C
                                     Device Address */
     __IN   uint32_t   Cmd      , /* No of Bytes to write*/
     __OUT     void   *pBuffer , /* Read Data pBuffer */
     __IN   uint32_t   BufLen   , /* Data pBuffer Length */
     __IN   uint32_t   ByteCnt    /* Number of Bytes to read */
);
```

Description

Reads a specific register from the selected I2C device.

Parameters

Id

__IN Refer **I2C Bus Ids**.

Addr

__IN Encoded 7/10-bit I2C device Address.

Cmd

__IN Encoded I2C device command / index. Refer **I2C Cmd Type** for command encoding.

***pBuffer**

__OUT Pointer to a buffer that receives the read data.

BufLen

__IN Size (in bytes), of the information pointed to by the **pBuffer** parameter.

ByteCnt

__IN Size (in bytes), of the information read to the buffer pointed to by the **pBuffer** parameter.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported I2C Bus Id	EAPI_STATUS_UNSUPPORTED
Invalid pBuffer, ByteCnt or BufLen	EAPI_STATUS_UNSUPPORTED
If ByteCnt is greater than maximum block length	EAPI_STATUS_INVALID_BLOCK_LENGTH
Buffer length smaller than ByteCnt	EAPI_STATUS_MORE_DATA
Mutex Failure / I2C communication failed via EC/BMC	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS
I2C Read Failed	EAPI_STATUS_READ_ERROR

6.8.5. SemaEApiI2CWriteTransfer

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiI2CWriteTransfer (
     __IN   uint32_t   Id       , /* I2C Bus Id*/
     __IN   uint32_t   Addr     , /* Encoded 7Bit I2C
                                     Device Address */
     __IN   uint32_t   Cmd      , /* No of Bytes to write*/
     __IN     void    *pBuffer , /* Data pBuffer */
     __IN   uint32_t   ByteCnt   /* Number of Bytes to write*/
);
```

Description

Writes to a specific register in the selected I2C device.

Parameters

Id

__IN I2C Bus Ids.

Addr

__IN Encoded 7/10-bit I2C device Address.

Cmd

__IN Encoded I2C device command / index. Refer **I2C Cmd Type** for command encoding.

***pBuffer**

___IN Pointer to a buffer that stores the write data.

ByteCnt

__IN Size (in bytes), of the **pBuffer**.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported I2C Bus Id	EAPI_STATUS_UNSUPPORTED
Invalid pBuffer or ByteCnt	EAPI_STATUS_UNSUPPORTED
If ByteCnt is greater than maximum block length	EAPI_STATUS_INVALID_BLOCK_LENGTH
Error while allocating write buffer internally	EAPI_STATUS_ALLOC_ERROR
Mutex Failure / I2C communication failed via EC/BMC	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS
I2C write Failed	EAPI_STATUS_WRITE_ERROR

6.8.6. SemaEApiI2CWriteReadRaw

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiI2CWriteReadRaw (
     IN   uint32_t   Id           , /* I2C Bus Id*/
     IN   uint32_t   Addr         , /* Encoded 7Bit I2C
                                      *Device Address */
     INOPT  void    *pWBuffer     , /* Write Data pWBuffer*/
     IN   uint32_t    WriteBCnt   , /* Number of Bytes to
                                      *write */
     OUTOPT  void   *pRBuffer     , /* Read Data pRBuffer */
     IN   uint32_t    RBufLen     , /* Data pRBuffer Length*/
     IN   uint32_t    ReadBCnt      /* Number of Bytes to
                                      * Read */
);
```

Description

Universal function for Combined I2C read and write transaction. (**I2C repeat start** feature).

Parameters

Id __IN Refer **I2C Bus Ids**.

Addr __IN Encoded 7/10-bit I2C device Address.

***pWBuffer**

__INOPT Pointer to a buffer containing the data to be transferred. This parameter can be NULL if the data is not required. 0th index of pWBuffer holds the encoded **Cmd**.

WriteBCnt

__IN Size, in bytes, of the information pointed to by the pWBuffer parameter.

***pRBuffer**

___OUTOPT Pointer to a buffer that receives the read data.

RBufLen

__IN Size, in bytes, of the buffer pointed to by the pRBuffer parameter.

ReadBCnt

__IN Size (in bytes), specifying the number of bytes to read.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported I2C Bus Id	EAPI_STATUS_UNSUPPORTED
Invalid buffers or other parameters	EAPI_STATUS_INVALID_PARAMETER
If ByteCnt is greater than maximum block length	EAPI_STATUS_INVALID_BLOCK_LENGTH
Mutex Failure / I2C communication failed via EC	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS
I2C Read Failed	EAPI_STATUS_READ_ERROR

6.9. Error Log Description

- Retrieves hardware and firmware error logs recorded by the system.
- Useful for post-failure analysis, debugging, and reliability assessment.

6.9.1. SemaEApiBoardGetErrorLog

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiBoardGetErrorLog(
     __IN   uint32_t     position    ,     /* Exception code */
     __OUT  uint32_t     *ErrorNumber,
     __OUT  uint8_t      *Flags      ,
     __OUT  uint8_t      *RestartEvent,
     __OUT  uint32_t     *PwrCycles  ,
     __OUT  uint32_t     *Bootcount  ,
     __OUT  uint32_t     *Time       ,
     __OUT  uint32_t     *TotalOnTime,
     __OUT  uint8_t      *BIOSSel    ,
     __OUT  uint16_t     *Status     ,
     __OUT  signed char  *CPUtemp    ,
     __OUT  signed char  *Boardtemp
);
```

Description

Get error number history of selected ports from the EC/BMC. The error log buffer stores the power sequence issues, information about the actual state and counters for better tracking of the issues.

The latest entry in the error log buffer is always found on position 0. The previous entry is found on position 1 and so on.

Parameters

position

__IN Position points to the log to be read

***ErrorNumber**

__OUT Pointer to buffer that stores the Error number. To get the detailed description, use **SemaEApiGetErrorNumberDescription.**

***Flags**

Exception Code, selected BIOS and Power Mode.

***RestartEvent**

__OUT Pointer to buffer that stores the system restart event.

***PwrCycles**

__OUT Pointer to buffer that stores the value of power cycles counter.

***Bootcount**

__OUT Pointer to buffer that stores the value of boot counter.

***Time**

__OUT Pointer to buffer that stores the value of ontime counter in seconds.

***TotalOnTime**

__OUT Pointer to buffer that stores the value of total on time in minutes.

***BIOSSel**

__OUT Pointer to buffer that stores the value of BIOS Selected.

***Status**

__OUT Pointer to buffer that stores the value of status information from BMC status command.

***CPUtemp**

__OUT Pointer to buffer that stores the value of current CPU temperature.

***Boardtemp**

__OUT Pointer to buffer that stores the value of current board temperature.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Error log functionality unsupported	EAPI_STATUS_UNSUPPORTED
Invalid buffers	EAPI_STATUS_INVALID_PARAMETER
Mutex Failure / Read error log failure	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.9.2. SemaEApiBoardGetCurErrorLog

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiBoardGetCurErrorLog(
     __OUT  uint32_t     *ErrorNumber,
     __OUT  uint8_t      *Flags      ,
     __OUT  uint8_t      *RestartEvent,
     __OUT  uint32_t     *PwrCycles  ,
     __OUT  uint32_t     *Bootcount  ,
     __OUT  uint32_t     *Time       ,
     __OUT  uint32_t     *TotalOnTime,
     __OUT  uint8_t      *BIOSSel    ,
     __OUT  uint16_t     *Status     ,
     __OUT  signed char  *CPUtemp    ,
     __OUT  signed char  *Boardtemp
);
```

Description

Get latest error number history from EC/BMC. The error log buffer stores the power sequence issues, information about the actual state and counters for better tracking of the issues.

Parameters

***ErrorNumber**

__OUT Pointer to buffer that stores the Error number. To get the detailed description, use **SemaEApiGetErrorNumberDescription.**

***Flags**

Exception Code, selected BIOS and Power Mode.

***RestartEvent**

__OUT Pointer to buffer that stores the system restart event.

***PwrCycles**

__OUT Pointer to buffer that stores the value of power cycles counter.

***Bootcount**

__OUT Pointer to buffer that stores the value of boot counter.

***Time**

__OUT Pointer to buffer that stores the value of ontime counter in seconds.

***TotalOnTime**

__OUT Pointer to buffer that stores the value of total on time in minutes.

***BIOSSel**

__OUT Pointer to buffer that stores the value of BIOS Selected.

***Status**

__OUT Pointer to buffer that stores the value of status information from BMC status command.

***CPUtemp**

__OUT Pointer to buffer that stores the value of current CPU temperature.

***Boardtemp**

__OUT Pointer to buffer that stores the value of current board temperature.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Error log functionality unsupported	EAPI_STATUS_UNSUPPORTED
Invalid buffers	EAPI_STATUS_INVALID_PARAMETER
Mutex Failure / Read error log failure	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.9.3. SemaEApiBoardGetErrorNumberDescription

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiBoardGetErrorNumberDescription(
     __IN    uint32_t  ErrorNum,          /* Pos */
     __OUT   uint8_t   *pBuffer           /* Buffer */
);
```

Description

Returns the Exception code string corresponding to the error number.

Parameters

Id

__IN Valid Error Number. This should be taken from the **SemaEApiBoardGetErrorLog.**

***pBuffer**

__OUT Pointer to a buffer that receives Error number description.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Error log functionality unsupported	EAPI_STATUS_UNSUPPORTED
Invalid buffer or Error number less than 1	EAPI_STATUS_INVALID_PARAMETER
Mutex Failure / Read error number description failed	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.10. BIOS Information

- Allows configuring the BIOS source mode such as **External BIOS**, **Standard BIOS**, or **Fail-Safe BIOS**.
- Used to control system boot behaviour and enable recovery or redundancy mechanisms.

6.10.1. SemaEApiGetBIOSSource

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiGetBIOSSource (
     __OUT   uint8_t   *bit0,      /* Buffer */
     __OUT   uint8_t   *bit1       /* Buffer */
);
```

Description

To get the BIOS Source.

Parameters

***bit0**

__OUT Pointer to buffer that receives the data of Bit 0.

***bit1**

__OUT Pointer to buffer that receives the data of Bit 1.

bit 1	bit 0	Description
0	0	BIOS selected by hardware configuration
0	1	Switch to Fail-safe BIOS
1	0	Switch to External BIOS (SPI0 on carrier)
1	1	Switch to Internal BIOS (SPI0 on Module)

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid buffers	EAPI_STATUS_MORE_DATA
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Read Failure	EAPI_STATUS_ERROR

6.10.2. SemaEApiSetBIOSSource

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiSetBIOSSource (
     __IN   uint8_t   bit0,         /* Buffer */
     __IN   uint8_t   bit1          /* Buffer *
);
```

Description

To set the BIOS Source.

Parameters

bit0

__IN Pointer to buffer that receives the data of Bit 0.

bit1

__IN Pointer to buffer that receives the data of Bit 1.

bit 1	bit 0	Description
0	0	BIOS selected by hardware configuration
0	1	Switch to Fail-safe BIOS
1	0	Switch to External BIOS (SPI0 on carrier)
1	1	Switch to Internal BIOS (SPI0 on Module)

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid bits	EAPI_STATUS_MORE_DATA
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Read Failure	EAPI_STATUS_ERROR

6.10.3. SemaEApiGetBiosSourceSts

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiGetBiosSourceSts (
     __OUT   uint8_t   *bit0,      /* Buffer */
     __OUT   uint8_t   *bit1       /* Buffer */
     __OUT   uint8_t   *bit2       /* Buffer */
);
```

Description

To get the BIOS status. [Supported only for EC boards]

Parameters

***bit0**

__OUT Pointer to buffer that receives the data of Bit 0.

***bit1**

__OUT Pointer to buffer that receives the data of Bit 1.

***bit2**

__OUT Pointer to buffer that receives the data of Bit 2.

bit 2	bit 1	bit 0	Description	BIOS type
0	0	0	*M0 Module SPI0 / C1 Carrier SPI1 (Standard BIOS)	PICMG BIOS selected
0	0	1	*C0 Carrier SPI0 / M1 Module SPI1 (Fail-Safe BIOS)	
0	1	0	Not PICMG Mode / Unknown	
0	1	1	*M0/M1 (Standard BIOS	
1	0	0	Not Dual BIOS Mode / Unknown	Dual BIOS selected
1	0	1	Switch to Fail-safe BIOS	
1	1	0	Switch to External BIOS (SPI0 on carrier)	
1	1	1	Switch to Internal BIOS (SPI0 on Module)	

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported functionality	EAPI_STATUS_UNSUPPORTED
Invalid buffers	EAPI_STATUS_MORE_DATA
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Read Failure	EAPI_STATUS_ERROR

6.11. Fan Control

- This functionality allows to monitor and control the fan connected to the CPU or system.
- Each fan controller supports 4 temperature trigger levels, each associated with a corresponding PWM value. Whenever a temperature trigger level is crossed, the fan speed is adjusted based on the configured PWM value for that level.
- Helps optimize cooling performance while reducing noise and power consumption.

Fan Id

Id	Description
SEMA_FAN_CPU	CPU Fan
SEMA_FAN_SYSTEM_1	System Fan
SEMA_FAN_SYSTEM_2	System fan 2 (Unsupported for EC boards)
SEMA_FAN_SYSTEM_3	System fan 3 (Unsupported for EC boards)

Fan Mode

FanMode	Description
SEMA_FAN_MODE_AUTO	AUTO (Smart fan)
SEMA_FAN_MODE_OFF	OFF
SEMA_FAN_MODE_ON	ON
SEMA_FAN_MODE_SOFT_FAN	Soft Fan (Smart Fan with Interpolation)

Fan Temperature Source

TempSrc	Description
SEMA_FAN_TEMP_CPU	CPU Sensor
SEMA_FAN_TEMP_SYS	Board Sensor

For System fan 1 of COM-HPC boards,

TempSrc	Description
SEMA_FAN_TEMP_CPU	CPU Sensor
SEMA_FAN_TEMP_SYS	Carrier Sensor

6.11.1. SemaEApiSmartFanSetTempSetpoints

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiSmartFanSetTempSetpoints(
     __IN   uint32_t   FanID,       /* Id */
     __IN   uint32_t   Level1,      /* Buffer */
     __IN   uint32_t   Level2,      /* Buffer */
     __IN   uint32_t   Level3,      /* Buffer */
     __IN   uint32_t   Level4       /* Buffer */
);
```

Description

To set temperature setpoints.

Parameters

FanID

__IN Refer **Fan Id**.

Level1

__IN Pointer to buffer that stores the temperature level 1 setpoint.

Level2

__IN Pointer to buffer that stores the temperature level 2 setpoint.

Level3

__IN Pointer to buffer that stores the temperature level 3 setpoint.

Level4

__IN Pointer to buffer that stores the temperature level 4 setpoint.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported Fan Id	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Write Failure	EAPI_STATUS_ERROR

6.11.2. SemaEApiSmartFanGetTempSetpoints

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiSmartFanGetTempSetpoints(
     __IN    uint32_t   FanID,      /* Id */
     __OUT   uint32_t   *Level1,    /* Buffer */
     __OUT   uint32_t   *Level2,    /* Buffer */
     __OUT   uint32_t   *Level3,    /* Buffer */
     __OUT   uint32_t   *Level4     /* Buffer */
);
```

Description

To set temperature setpoints.

Parameters

FanID

__IN Refer **Fan Id**.

***Level1**

__IN Pointer to buffer that receives the temperature level 1 setpoint

***Level2**

__IN Pointer to buffer that receives the temperature level 2 setpoint

***Level3**

__IN Pointer to buffer that receives the temperature level 3 setpoint

***Level4**

__IN Pointer to buffer that receives the temperature level 4 setpoint

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported Fan Id	EAPI_STATUS_UNSUPPORTED
Invalid buffers	EAPI_STATUS_INVAID_PARAMETER
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Read Failure	EAPI_STATUS_ERROR

6.11.3. SemaEApiSmartFanSetPWMSetpoints

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiSmartFanSetPWMSetpoints(
     __IN   uint32_t   FanID,       /* Id */
     __IN   uint32_t   Level1,      /* Buffer */
     __IN   uint32_t   Level2,      /* Buffer */
     __IN   uint32_t   Level3,      /* Buffer */
     __IN   uint32_t   Level4       /* Buffer */
);
```

Description

To set PWM level triggers corresponding to the Temperature Level setpoints.

Parameters

FanID

__IN Refer **Fan Id**.

Level1

__IN Pointer to buffer that stores the PWM level 1 trigger.

Level2

__IN Pointer to buffer that stores the PWM level 2 trigger.

Level3

__IN Pointer to buffer that stores the PWM level 3 trigger.

Level4

__IN Pointer to buffer that stores the PWM level 4 trigger.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported Fan Id	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Write Failure	EAPI_STATUS_ERROR

6.11.4. SemaEApiSmartFanGetPWMSetpoints

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiSmartFanGetPWMSetpoints(
     __IN    uint32_t   FanID,         /* Id */
     __OUT   uint32_t   *pLevel1,     /* Buffer */
     __OUT   uint32_t   *pLevel2,     /* Buffer */
     __OUT   uint32_t   *pLevel3,     /* Buffer */
     __OUT   uint32_t   *pLevel4      /* Buffer */
);
```

Description

Returns the PWM level triggers corresponding to the Temperature Level setpoints.

Parameters

FanID

__IN Refer **Fan Id**.

***pLevel1**

__IN Pointer to buffer that receives the PWM level 1 trigger.

***pLevel2**

__IN Pointer to buffer that receives the PWM level 2 trigger.

***pLevel3**

__IN Pointer to buffer that receives the PWM level 3 trigger.

***pLevel4**

__IN Pointer to buffer that receives the PWM level 4 trigger.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported Fan Id	EAPI_STATUS_UNSUPPORTED
Invalid buffers	EAPI_STATUS_INVAID_PARAMETER
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Read Failure	EAPI_STATUS_ERROR

6.11.5. SemaEApiSmartFanSetMode

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiSmartFanSetMode(
     __IN   uint32_t   FanID,        /* Id */
     __IN   uint32_t   pFanMode      /* Buffer */
);
```

Description

To set fan mode.

Parameters

FanID

__IN Refer **Fan Id**.

pFanMode

__IN Refer **Fan Mode**.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported Fan Id / Fan mode	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Write Failure	EAPI_STATUS_ERROR

6.11.6. SemaEApiSmartFanGetMode

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiSmartFanGetMode(
     __IN  uint32_t    FanID,              /* Id */
     __OUT  uint32_t   *pFanMode           /* Buffer */
);
```

Description

To get fan mode.

Parameters

FanID

__IN Refer **Fan Id**.

pFanMode

__OUT Refer **Fan Mode**.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported Fan Id / Fan mode	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Read Failure	EAPI_STATUS_ERROR

6.11.7. SemaEApiSmartFanSetTempSrc

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiSmartFanSetTempSrc(
     __IN   uint32_t   FanID,        /* Id */
     __IN   uint32_t   pTempSrc      /* Buffer */
);
```

Description

To set temperature source of the specific fan Id.

Parameters

FanID

__IN Refer **Fan Id**.

pTempSrc

__IN Pointer to buffet that stores the temperature source. Refer **Fan Temperature Source.**

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported Fan Id / Fan mode	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Write Failure	EAPI_STATUS_ERROR

6.11.8. SemaEApiSmartFanGetTempSrc

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiSmartFanGetTempSrc(
     __IN  uint32_t    FanID,          /* Id */
     __OUT  uint32_t   *pTempSrc       /* Buffer */
);
```

Description

To get fan mode.

Parameters

FanID

__IN Refer **Fan Id**.

***pTempSrc**

__OUT Pointer to buffer that receives the temperature source. Refer **Fan Temperature Source.**

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported Fan Id / Fan mode	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Read Failure	EAPI_STATUS_ERROR

6.12. Backlight

- Enables control of display backlight brightness levels.
- Used to adjust visibility and reduce power consumption in embedded displays.

Backlight Id

Id	Description
EAPI_ID_BACKLIGHT_1	Backlight Local Flat Panel 1

PWM Level

Backlight Enable values	Description
1	Requests/Signifies that the Backlight be Enabled
0	Requests/Signifies that the Backlight be Disabled

6.12.1. SemaEApiVgaSetBacklightEnable

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiVgaSetBacklightEnable (
     __IN   uint32_t   PanelId    ,      /* Backlight Id */
     __IN   uint8_t    PWMLevel          /* Backlight Enable */
);
```

Description

Enables the backlight of the selected flat panel display and set brightness level 200 by default.

Parameters

PanelId

__IN Refer **Backlight Id**.

PWMLevel

__IN Pointer to a buffer that stores the current backlight enable state. Refer **PWM Level.**

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported functionality	EAPI_STATUS_UNSUPPORTED
Invalid buffers	EAPI_STATUS_INVALID_PARAMETER
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Write Failure	EAPI_STATUS_ERROR

6.12.2. SemaEApiVgaGetBacklightEnable

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiVgaGetBacklightEnable (
     __IN   uint32_t   PanelId    ,      /* Backlight Id */
     __OUT   uint8_t   *PWMLevel         /* Backlight Enable */
);
```

Description

Get the backlight enable status of the selected flat panel display.

Parameters

PanelId

__IN Refer **Backlight Id**.

PWMLevel

__OUT Pointer to a buffer that receives the current backlight enable state. Refer **PWM Level.**

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported functionality	EAPI_STATUS_UNSUPPORTED
Invalid buffers	EAPI_STATUS_INVALID_PARAMETER
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Write Failure	EAPI_STATUS_ERROR

6.12.3. SemaEApiVgaSetBacklightBrightness

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiVgaSetBacklightBrightness(
     __IN   uint32_t   PanelId    ,     /* Backlight Id */
     __IN   uint8_t    PWMLevel         /* Backlight Enable */
);
```

Description

Sets the brightness of the selected flat panel display to desired level.

Parameters

PanelId

__IN Refer **Backlight Id**.

PWMLevel

__IN Pointer to a buffer that stores the Backlight brightness level. The range of PWM Level is from 0 to 255.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported functionality	EAPI_STATUS_UNSUPPORTED
Invalid buffers	EAPI_STATUS_INVALID_PARAMETER
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Write Failure	EAPI_STATUS_ERROR

6.12.4. SemaEApiVgaGetBacklightBrightness

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiVgaGetBacklightBrightness(
     __IN   uint32_t   PanelId    ,      /* Backlight Id */
     __OUT   uint8_t   *PWMLevel         /* Backlight Enable */
);
```

Description

Reads the current brightness of the selected flat panel display.

Parameters

PanelId

__IN Refer **Backlight Id**.

PWMLevel

__OUT Pointer to a buffer that stores the Backlight brightness level. The range of PWM Level is from 0 to 255.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unsupported functionality	EAPI_STATUS_UNSUPPORTED
Invalid buffers	EAPI_STATUS_INVALID_PARAMETER
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Read Failure	EAPI_STATUS_ERROR

6.13 Exception Description

- Provides detailed descriptions of error and exception codes returned by APIs.
- Helps developers quickly diagnose failures and understand error conditions.

6.13.1 SemaEApiBoardGetExcepDesc

Function definition

```
uint32_t
EAPI_CALLTYPE
SemaEApiBoardGetExcepDesc(
    __IN  uint32_t  Exceptioncode, /* Exception code */
    __OUT char      *pBuf          /* Buffer */
    __INOUT  uint32_t  size           /* size in bytes */
);
```

Description

To get text information of the Exception code

Parameters

Exceptioncode

__IN The Error number (Exception Code).

***pBuf**

__OUT Pointer to buffer that receives Error number.

size

__INOUT Size of pBuf in bytes.

Return Status Code

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid buffer	EAPI_STATUS_INVALID_PARAMETER
Read exception description failed	EAPI_STATUS_ERROR
Success	EAPI_STATUS_SUCCESS

6.14. SMBus Slave Device Access

6.14.1. SemaEApiSMBReadTrans

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiSMBReadTrans (
     __IN   uint32_t   Id       , /* Bus Id*/
     __IN   uint16_t   Addr     , /* Encoded 7Bit SMBus
                                     Device Address */
     __IN   uint32_t   Cmd      , /* SMBus Device command */
     __OUT     void   *pBuffer , /* Read Data pBuffer */
     __IN   uint32_t   BufLen   , /* Data pBuffer Length */
     __IN   uint32_t   ByteCnt  /* Number of Bytes to
                                   Read*/
);
```

Description

Reads from a specific register in the selected SMBus device.

Reads from SMBus device at the SMBus address **(Addr)** and stores in the buffer **pBuffer** while using the device specific command **Cmd.**

Parameters

Id

__IN For SMBus Transfer, Id=0.

Addr

__ IN Encoded 7-bit SMBus device address. Refer hardware manual.

Cmd

__IN Encoded SMBus Device Command / Index.

***pBuffer**

__OUT Pointer to a buffer that receives the read data.

BufLen

__IN BufLen is 2 for ReadWord.
BufLen is 32 for ReadBlock operation.
BufLen is 1 for ReadByte operation.

ByteCnt

__IN Size (in bytes), of the information read to the buffer pointed to by pBuffer.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid buffers / SMBus functionality unsupported	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Read Failure	EAPI_STATUS_ERROR

6.14.2. SemaEApiSMBWriteTrans

Function Definition

```
uint32_t
SEMAEAPI_API
SemaEApiSMBWriteTrans (
     __IN   uint32_t   Id         , /* Bus Id*/
     __IN   uint32_t   Addr       , /* Encoded 7/10Bit I2C
                                       Device Address */
     __IN   uint32_t   Cmd        , /* I2C Command/Offset */
     __IN      void   *pBuffer    , /* Data buffer */
     __IN   uint32_t   BufLen     , /* Buffer Length */
     __IN   uint32_t   ByteCnt      /* Byte Count to write */
);
```

Description

Writes to a specific register in the selected SMBus device.

Parameters

Id

__IN For SMBus Transfer, Id=0.

Addr

__ IN Encoded 7-bit SMBus Slave device address.

Cmd

__IN Encoded SMBus Device Command / Index.

***pBuffer**

__IN Pointer to a buffer that stores the data to be read.

BufLen

__IN BufLen is 2 for WriteWord.
BufLen is 1 for WriteByte operation.

ByteCnt

__IN Size (in bytes), of the buffer pointed to by the pBuffer parameter.

Return Status

Condition	Return Value
Library uninitialized	EAPI_STATUS_NOT_INITIALIZED
Invalid buffers / SMBus functionality unsupported	EAPI_STATUS_UNSUPPORTED
Success	EAPI_STATUS_SUCCESS
Mutex Failure / Read Failure	EAPI_STATUS_ERROR

7. Data Interpretation

7.1. Specification Version

Definition

Bits	Description
[31:24]	Version
[23:16]	Revision
[15:0]	0

Example

Hex	Interpreted
0x01000000	1.0.0

7.2. General Version

Definition

Bits	Description
[31:24]	Major Version
[23:16]	Minor Version
[15:0]	Build Number

Example

Hex	Interpreted
0x05000000	5.0.0
0x0402000C	4.2.12

8. Annexure

8.1. Restart Event

Code	Code name	Description
0x00	SEMA_SRE_UNKNOWN	Unknown reason of restart (shown only on first BMC power-up)
0x20	SEMA_SRE_SW_RESET	A reset by software caused the restart of the system
0x30	SEMA_SRE_HW_RESET	A reset by hardware caused the restart of the system (e.g.: Reset button)
0x40	SEMA_SRE_WATCHDOG	The watchdog has restarted the system
0x50	SEMA_SRE_BIOS_FAULT	Standard BIOS is corrupted -> boot from Fail-Safe BIOS
0x60	SEMA_SRE_PWR_DOWN	The system was shut down (e.g. power button, ACPI shutdown)
0x70	SEMA_SRE_PWR_LOSS	The system was restarted after a power loss (e.g. external power supply instable or switched off while the system was running)
0x80	SEMA_SRE_PWR_CYCLE	The system is restarted after a power cycle (e.g. internal power supply has failed)
0x90	SEMA_SRE_VIN_DROP	The system is restarted after a voltage drop of the main input voltage
0xA0	SEMA_SRE_PWR_FAIL	The system is restarted after a power fail detection of an internal power supply circuit
0xB0	SEMA_SRE_CRIT_TEMP	The system was shut down by ACPI watchdog (CPU reached critical temperature)
0xC0	SEMA_SRE_WAKEUP	The system has received a wake event and resumes operation from a sleep

8.2. Firmware Capability

Firmware capabilities describe the functions that can be supported on the platform:

- When bit value goes to **1**, it indicates **enabled**
- When bit value goes to **0**, it indicates **disabled**

Bit	Description
Bit 0	Uptime & power cycles counter
Bit 1	System restart event
Bit 2	User flash size (0=512 bytes, 1=1024 bytes)
Bit 3	Runtime Watchdog
Bit 4	Temperatures
Bit 5	Voltage monitor
Bit 6	Storage of failure reason (Old SEMA support)
Bit 7	Boot loader timeout programmable (Old SEMA support)
Bit 8	Display Backlight control
Bit 9	Power up watchdog
Bit 10	Power monitor (Current sense)
Bit 11	Boot counter
Bit [13:12]	Input-Voltage (00=5V, 01=12V, 10=3V, 11=reserved) [Not supported by EC]
Bit 14	Rsense of Power monitor (0=8 mR, 1=4 mR) [Not supported by EC]
Bit 15	Dual-BIOS
Bit 16	I2C bus 1
Bit 17	I2C bus 2
Bit 18	CPU Fan
Bit 19	System Fan 1
Bit 20	AT/ATX mode
Bit 21	ACPI Thermal trigger
Bit 22	Power-up to last state
Bit 23	Backlight restore
Bit 24	DTS Temperature [Not supported by EC]
Bit 25	DTS offset registers [Not supported by EC]
Bit 26	System fan 2 [Not supported by EC]
Bit 27	System fan 3 [Not supported by EC]
Bit 28	Ext GPIO
Bit 29	I2C bus 3
Bit 30	I2C bus 4
Bit 31	BMC is from TIVA type

8.3. Extended Firmware Capability

Bit	Description
Bit 32	Board2 Temperature
Bit 33	PEC protocol
Bit 34	reserved
Bit 35	Error log
Bit 36	1-wire bus
Bit 37	Wake-by-BMC
Bit 38	GPIO alternate function
Bit 39	Soft Fan
Bit 40	Parameter memory
Bit 41	Extended I2C registers for status and data
Bit 42	Ext GPIO Input Interrupt (Supported only for EC boards)
Bit 43	Hardware Monitor Input String (Supported only for EC boards)
Bit 44	Ext-GPIO Pins Count (Supported only for EC boards)
Bit 45	Power-up Watchdog / Runtime Watchdog support action setting (Supported only for EC boards)
Bit 46	Switch BIOS immediately (Supported only for EC boards)

8.4. EC/BMC Flags

Bit		Description
Bit 7	0	Standard BIOS is active
	1	Fail-Safe BIOS is active
Bit 6	0	AT mode
	1	ATX mode
Bit 0 - 4	Exception Code. Refer **EApiBoardGetExcepDesc.**	

8.5. Sample Code

8.5.1. To retrieve Board name

```
// Test Application
#include <string.h>
#include <stdio.h>
#include "EApi.h"

int main (int argc, char* argv[])
{
   uint32_t eRet;
   unsigned char pData[100] = { 0 };
      unsigned int buflen = 100;

   // EAPI Library Initialization
   eRet = EApiLibInitialize();
   if (eRet!= EAPI_STATUS_INITIALIZED)
   {
        printf("\nEapi Library not initialized\n");
        return -1;
   }

   // Read Board Name
   eRet = EApiBoardGetStringA (EAPI_ID_BOARD_NAME_STR, pData,
&buflen);
   if (eRet == EAPI_STATUS_SUCCESS) {
        printf("\nBoard Name: %s\n ", pData);
   }
   else {
        printf("\nEApiBoardGetStringA failed with return code
0x%x ", eRet);
        return -1;
   }

   // To uninitialized the EAPI Library
   eRet= EApiUnInitialize();

   if (eRet == EAPI_STATUS_SUCCESS)
   {
        return 0;
   }
}
```

8.5.2. To demonstrate I2C read and write using transfer functions (Linux)

```
// Test Application (Assuming EEPROM is connected in I2C Bus 4
and 7-bit address 0x50)
#include <string.h>
#include <stdio.h>
#include "EApi.h"
int main (int argc, char* argv[]) {
   uint32_t eRet, BusID= 4, cmd= 0x00, BufLen= 2, nByteCnt= 2;
   uint16_t Address = 0x50;
   unsigned char Rdbuffer[2] = {0x0,0x0}, pBuffer[2] = {0xb,0xc};

   // EAPI Library Initialization
   eRet = EApiLibInitialize();
   if (eRet!= EAPI_STATUS_INITIALIZED) {
         printf("\nEapi Library not initialized\n");
         return -1;
   }

   // To write 2 bytes into EEPROM
   eRet = EApiI2CWriteTransfer (BusID - 1, Address, cmd, pBuffer
         ,nByteCnt);
   if (eRet == EAPI_STATUS_SUCCESS) {
         printf("\nI2CWriteTransfer success at address 0x%02x",
         Address);
   } else {
         printf("\nI2CWriteTransfer failed; Return code 0x%x ",
         eRet);
         return -1;
   }

   // To read 2 bytes from EEPROM
   eRet = EApiI2CReadTransfer (BusID - 1, Address, cmd, \
          Rdbuffer, BufLen, nByteCnt);
   if (eRet == EAPI_STATUS_SUCCESS) {
            printf("\nEApiI2CReadTransfer at address 0x%02x is
            successful. \nRead data: 0x%02x  0x%02x",
            Address, Rdbuffer[0], Rdbuffer[1]);
   } else {
         printf("\nI2CReadTransfer failed; Return code 0x%x ",
         eRet);
         return -1;
   }
   // Uninitializing EAPI Library
   eRet= EApiUnInitialize();
   if (eRet == EAPI_STATUS_SUCCESS)
         return 0;
}
```

8.5.3. To retrieve running time meter value

```
//Test Application
#include <stdio.h>
#include "eapi.h"

int main(int argc, char* argv[])
{
   uint32_t eRet;
   uint32_t runningTime = 0;

   // EAPI Library Initialization
   eRet = EApiLibInitialize();
   if (eRet != EAPI_STATUS_SUCCESS) {
       printf("EApiLibInitialize failed: 0x%x\n", eRet);
       return -1;
   }

   // Read Running Time Meter
   eRet = EApiBoardGetValue(
              EAPI_ID_BOARD_RUNNING_TIME_METER_VAL,
               &runningTime);

   if (eRet == EAPI_STATUS_SUCCESS) {
       printf("Running Time Meter: %u minutes\n", runningTime);
   } else {
       printf("EApiBoardGetValue failed: 0x%x\n", eRet);
       EApiLibUnInitialize();
       return -1;
   }

   // To uninitialized the EAPI Library
   eRet = EApiLibUnInitialize();
   if (eRet != EAPI_STATUS_SUCCESS) {
       printf("EApiLibUnInitialize failed: 0x%x\n", eRet);
       return -1;
   }
   return 0;
}
```

9. Appendix

9.1. Temperature Conversion

// To convert Celsius to Kelvin, use the below macro.

```
EAPI_ENCODE_CELCIUS(Celsius)
```

Note: Make sure the Celsius value is in integer data type

// To convert Kelvin to Celsius, use the below macro

```
EAPI_DECODE_CELCIUS(Kelvin)
```

Note: Make sure the Kelvin value is in integer data type

9.2. Error Codes

Error Code	Value
EAPI_STATUS_SUCCESS	0
EAPI_STATUS_ERROR	0xFFFFF0FF
EAPI_STATUS_MORE_DATA	0xFFFFF9FF
EAPI_STATUS_WRITE_ERROR	0xFFFFFAFE
EAPI_STATUS_READ_ERROR	0xFFFFFAFf
EAPI_STATUS_BUSY_COLLISION	0xFFFFFBFD
EAPI_STATUS_TIMEOUT	0xFFFFFBFE
EAPI_STATUS_NOT_FOUND	0xFFFFFBFF
EAPI_STATUS_UNSUPPORTED	0xFFFFFCFF
EAPI_STATUS_RUNNING	0xFFFFFEFA
EAPI_STATUS_INVALID_BITMASK	0xFFFFFEFB
EAPI_STATUS_INVALID_DIRECTION	0xFFFFFEFC
EAPI_STATUS_INVALID_BLOCK_LENGTH	0xFFFFFEFD
EAPI_STATUS_INVALID_BLOCK_ALIGNMENT	0xFFFFFEFE
EAPI_STATUS_INVALID_PARAMETER	0xFFFFFEFF
EAPI_STATUS_DRIVER_TIMEOUT	0xFFFFFFFC
EAPI_STATUS_ALLOC_ERROR	0xFFFFFFFD
EAPI_STATUS_INITIALIZED	0xFFFFFFFE
EAPI_STATUS_NOT_INITIALIZED	0xFFFFFFFF

10. FAQs & Troubleshooting

1. **Do I need kernel drivers to use SEMA EAPI on Linux/Windows?**
   Yes. SEMA relies on kernel-level drivers (EC, LPC, or I^2C related drivers depending on the platform). These drivers must be loaded successfully before running any user-space EAPI application.

2. **Do the header files need to be present while compiling a SEMA Linux EAPI application?**
   Yes. During compilation, ensure that eapi.h and conv.h are present. For simplicity, place them in the same directory as the source file. Missing headers will cause compilation to fail.

3. **Do the header files need to be present while compiling a SEMA Windows EAPI application?**
   Yes. During compilation, ensure that EApi.h and Error.h are present. For simplicity, place them in the same directory as the source file. Missing headers will cause compilation to fail.

4. **How to compile and execute the sample code for Linux?**
   - Install SEMA Linux Unified using the steps mentioned in the Installation Guide.
   - Copy the **Sample code** and name as sample.c. Then in the same folder add conv.h and eapi.h.
   - Compile the sample.c application using the command:

```
$ gcc sample.c -o sample -lsema
```

   - Run the sample code using the command:

```
$ ./sample
```

5. **How to compile and execute the sample code for Windows?**
   - Install SEMA Windows using the steps mentioned in the Installation Guide.
   - Create a C++ desktop application project in Visual studio.
   - Copy the code and name as sample.c. Then in the same folder add EApi.h, Error.h, SemaEApi.dll and SemaEApi.lib.
   - Link the SemaEApi.lib to sample.c using pragma comment or properties.

```
#pragma comment (lib, "SemaEapi.lib")
```

   - Build the solution file in Release x64 mode.
   - Run the generated application (sample.exe).

6. **What are common reasons for EApiLibInitialize() failure in Linux?**
   Common causes include:
   - SEMA kernel driver not loaded
   - Unsupported or mismatched BIOS/EC/BMC firmware
   - Insufficient permissions (Running without root access on Linux)

7. **What EAPI can be used for I2C repeat start feature in Windows?**
   SemaEApiI2CWriteReadRaw can be used for I2C repeat start feature.

8. **Explain the fan modes. What should be the setting of the fan if we require constant rpm value.**
   Fan modes:
   - Auto – the fan will be automatically controlled using the Trigger Settings described
   - Off – the fan is turned off completely
   - On – the fan runs at maximum RPM (PWM level 100%)
   - Soft – Auto mode with PWM Level Interpolation (i.e.) The PWM level is not adapted stepwise, but continuously.

   Fan mode changed to "**On"** to get the constant rpm value.