ADLINK TECHNOLOGY INC.

ADLINK SEMA® 3.5

SEMA® Extended EAPI Programming Guide

Manual Rev.: 1.01

Revision Date: December 2016

Part No.: 50-10050-1010

Revision History

Revision	Date	Changes
1.00	September 2016	Initial release
1.01	December 2016	Update based on SEMA® 3.5 R7 Added the following new functions: • SemaEApiGPIOPinGetCap • SemaEApiGPIOPinSetConfig • SemaEApiGPIOFuncSetPinConfig • SemaEApiGPIOPinGetADCValue • SemaEApiBordGetCurrentPosErrorLog • SemaEApiBoardGetExceptionDescription Added Chapter 2.18 A/D Conversion

Copyright 2016 ADLINK Technology, Inc.

Table of Contents

1 SEMA Extended EAPI

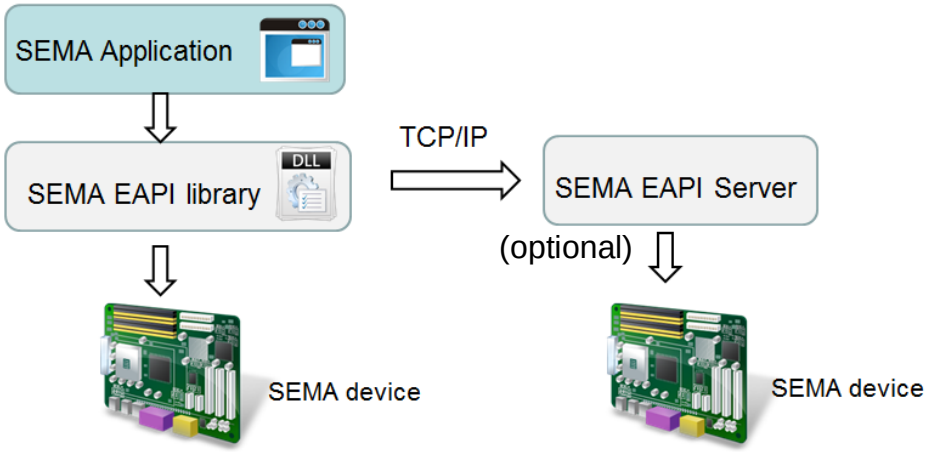
1.1 Overview

The ADLINK Smart Embedded Management Agent (SEMA) is an intelligent middleware that monitors and controls your devices. SEMA includes an API (SEMA Extended Embedded API) that allows customers to easily integrate all SEMA functions into their applications, including the following:

- Access board information
- Access CPU/memory/network information
- Control CPU operating mode
- Control backlight, I^2C, GPIO, watchdog, fan, 1-Wire Device
- Access storage drive S.M.A.R.T. data
- Update BMC/BIOS firmware
- Access Forensic information (BMC Power-Up Error Log)

In this document we use the abbreviation “SEMA EAPI” to refer to the SEMA Extended EAPI. SEMA EAPI includes all EAPI functions defined in PICMG EAPI specification (http://picmg.org//wp-content/uploads/COM_EAPI_R1_0.pdf) and also provides special functions to enable users accessing additional information and controlling their devices. All SEMA EAPI functions can (optionally) be called remotely from another computer via network (TCP/IP). To secure the data transported via a TCP/IP network, SEMA EAPI also provides password authentication and SSL/TLS encryption.

1.2 Architecture



1.3 Supported Operating Systems

The following operating systems are supported:

- ► Windows 7 (32/64-bit)
- ► Windows 8/8.1 (32/64-bit)
- ► Windows 10 (32/64-bit)
- ► Linux (3.2.x) or higher

2 Function Documentation

2.1 Initialization Functions

2.1.1 SemaEApiLibInitialize

```
uint32_t
EAPI_CALLTYPE
SemaEApiLibInitialize(
    __IN    bool     ssl,
    __IN    IP_Version ipv,
    __IN    char    *pHostIP,
    __IN    uint32_t     port,
    __IN    char     *pPasswd,
    __OUT   uint32_t    *pHandler
);
```

Description

Initialization of SEMA EAPI. Prior to calling any SEMA EAPI function, the library needs to be initialized by calling this function. The status code for all EAPI functions will be **EAPI_STATUS_NOT_INITIALIZED** until this function is called.

Parameters

In/Out	Parameter Name	Description
__IN	ssl	True for security connection
__IN	IP_Version	Select IP_V4 for IPv4 and IP_V6 for IPv6
__IN	pHostIP	IP address of a remote computer. If it is set to NULL or "localhost", then localhost is considered to be the target machine
__IN	port	Default port number which the SEMA EAPI server listens to is 9999
__IN	pPasswd	Password to access the remote target board
__OUT	pHandler	Returned pointer to the handler of selected board. *BoardHandle* == 0 means localhost.

```
typedef enum _IPVERSION{
        IP_V4,
        IP_V6
}IP_Version;
```

2.1.2 SemaEApiLibUnInitialize

```
uint32_t
EAPI_CALLTYPE
SemaEApiLibUnInitialize(
      __IN   uint32_t Handler
);
```

Description

Uninitialization of SEMA EAPI on the selected board.

Parameters

In/Out	Parameter Name	Description
__IN	Handler	The handler of selected board to be uninitialized

2.2 Board

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

- Static board information (e.g. model name, BIOS version, manufacturer name)
- Dynamic board information (e.g. voltage, current, boot count, temperature)
- Failure forensics (e.g. restart event)

2.2.1 SemaEApiBoardGetStringA

```
uint32_t
EAPI_CALLTYPE
SemaEApiBoardGetStringA(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Name Id */
__OUT char *pBuffer , /* Destination pBuffer */
__INOUT uint32_t *pBufLen /* pBuffer Length */
);
```

Description

Text information about the hardware platform. Supports EAPI ID and SEMA Extend ID. Please refer to PICMG EAPI 1.0 for more details about EAPI ID.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board.
__IN	Id	Selects the Get String Sub function ID (refer to PICMG EAPI 1.0)
__OUT	pBuffer	Pointer to a buffer that receives the value's data. This parameter can be NULL if the data is not required.
__INOUT	pBufLen	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.

EAPI ID

ID	Description	Units/Format
EAPI_ID_BOARD_MANUFACTURER_STR	Board manufacturer name	string
EAPI_ID_BOARD_NAME_STR	Board name	string
EAPI_ID_BOARD_SERIAL_STR	Serial number	string
EAPI_ID_BOARD_BIOS_REVISION_STR	Board BIOS revision	string
EAPI_ID_BOARD_PLATFORM_TYPE_STR	Platform ID	string
EAPI_ID_BOARD_HW_REVISION_STR	HW revision string	string

SEMA Extend ID

ID	Description	Units/Format
SEMA_EAPI_ID_BOARD_BOOT_VERSION_STR	Boot version string	string
SEMA_EAPI_ID_BOARD_APPLICATION_VERSION_STR	Firmware application version string	string
SEMA_EAPI_ID_BOARD_CHIPSET_ID_STR	Chipset ID string	string
SEMA_EAPI_ID_BOARD_RESTART_EVENT_STR	Restart Event string	string
SEMA_EAPI_ID_BOARD_DEVICE_ID_STR	Device ID string	string
SEMA_EAPI_ID_BOARD_REPAIR_DATE_STR	Last Repair Date	string
SEMA_EAPI_ID_BOARD_MANUFACTURE_DATE_STR	Manufacture date	string
SEMA_EAPI_ID_BOARD_MAC_STRING	MAC address on module	string
SEMA_EAPI_ID_BOARD_2ND_HW_REVISION_STR	2nd HW revision string	string
SEMA_EAPI_ID_BOARD_2ND_SERIAL_STR	2nd HW serial string	string

2.2.2 SemaEApiBoardSetStringA

```
uint32_t
EAPI_CALLTYPE
SemaEApiBoardSetStringA(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Name Id */
__IN char *pBuffer , /* String to write */
__OUT uint32_t *pBufLen /* pBuffer Length */
);
```

Description

Set text information about the hardware platform.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	Selects the Get String Sub function Id (refer to PICMG EAPI 1.0)
__IN	pBuffer	Pointer to a buffer that store string data.
__OUT	pBufLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the ***pBuffer*** parameter.

SEMA Extend ID

ID	Description	Units/Format
SEMA_EAPI_ID_BOARD_DEVICE_ID_STR	Device ID string	string

2.2.3 SemaEApiBoardGetValue

```
uint32_t
EAPI_CALLTYPE
SemaEApiBoardGetValue(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Value Id */
__OUT uint32_t *pValue /* Return Value */
);
```

Description

Text information about the hardware platform. Supports EAPI ID and SEMA Extend ID. For more details about EAPI ID, please refer to PICMG EAPI 1.0.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	Selects the Get Value Sub function Id (refer to PICMG EAPI 1.0 and SEMA Extend ID)
__OUT	pValue	Pointer to a buffer that receives the value's data

EAPI ID

ID	Description	Units/Format
EAPI_ID_GET_EAPI_SPEC_VERSION	EAPI Specification Version used to implement API	
EAPI_ID_BOARD_BOOT_COUNTER_VAL	Boot Counter	boots
EAPI_ID_BOARD_RUNNING_TIME_METER_VAL	Running Time Meter	minutes
EAPI_ID_BOARD_PNPID_VAL	Board Vendor PNPID	Compressed ASCII PNPID
EAPI_ID_BOARD_PLATFORM_REV_VAL	Platform Specification Version used to create Board	[31:24] Version [23:16] Revision [15:0] 0
EAPI_ID_BOARD_DRIVER_VERSION_VAL	Vendor Specific Driver Version	
EAPI_ID_BOARD_LIB_VERSION_VAL	Vendor Specific Library Version	
EAPI_ID_HWMON_CPU_TEMP	CPU Temperature	0.1 Kelvins

ID	Description	Units/Format
EAPI_ID_HWMON_CHIPSET_TEMP	Chipset Temperature (not supported)	0.1 Kelvins
EAPI_ID_HWMON_SYSTEM_TEMP	System Temperature	0.1 Kelvins
EAPI_ID_HWMON_VOLTAGE_VCORE	CPU Core Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_2V5	2.5 V Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_3V3	3.3 V Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_VBAT	Battery Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_5V	5 V Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_5VSB	5 V Standby Voltage	millivolts
EAPI_ID_HWMON_VOLTAGE_12V	12 V Voltage	millivolts
EAPI_ID_HWMON_FAN_CPU	CPU Fan	RPM
EAPI_ID_HWMON_FAN_SYSTEM	System Fan	RPM

SEMA Extend ID

ID	Description	Units/Format
SEMA_EAPI_ID_BOARD_POWER_UP_TIME	Get power uptime	seconds
SEMA_EAPI_ID_BOARD_POWER_CYCLE	Power cycle counter	
SEMA_EAPI_ID_BOARD_RESTART_EVENT	Get restart event	See note 1
SEMA_EAPI_ID_BOARD_CAPABILITIES	Get BMC capabilities	See note 2
SEMA_EAPI_ID_BOARD_CAPABILITIES_EX	Get extended BMC capabilities	See note 3
SEMA_EAPI_ID_BOARD_SYSTEM_MIN_TEMP	Board Min Temperature	0.1 Kelvins
SEMA_EAPI_ID_BOARD_SYSTEM_MAX_TEMP	Board Max Temperature	0.1 Kelvins
SEMA_EAPI_ID_BOARD_SYSTEM_STARTUP_TEMP	Board Startup Temperature	0.1 Kelvins
SEMA_EAPI_ID_BOARD_CPU_MIN_TEMP	CPU Min Temperature	0.1 Kelvins
SEMA_EAPI_ID_BOARD_CPU_MAX_TEMP	CPU Max Temperature	0.1 Kelvins
SEMA_EAPI_ID_BOARD_CPU_STARTUP_TEMP	CPU Startup	0.1 Kelvins

ID	Description	Units/Format
	Temperature	
SEMA_EAPI_ID_ BOARD_MAIN_CURRENT	Get main power current	unit: mA, resolution=1/10 mA
SEMA_EAPI_ID_HWMON_VOLTAGE_GFX_VCORE	GFX voltage	millivolts
SEMA_EAPI_ID_HWMON_VOLTAGE_1V05	1.05 V voltage	millivolts
SEMA_EAPI_ID_HWMON_VOLTAGE_1V5	1.5 V voltage	millivolts
SEMA_EAPI_ID_HWMON_VOLTAGE_VIN	Vin Voltage	millivolts
SEMA_EAPI_ID_HWMON_FAN_SYSTEM_2	2nd System Fan	RPM
SEMA_EAPI_ID_HWMON_FAN_SYSTEM_3	3rd System Fan	RPM
SEMA_EAPI_ID_BOARD_2ND_SYSTEM_TEMP	2nd board Temperature	0.1 Kelvins
SEMA_EAPI_ID_BOARD_2ND_SYSTEM_MIN_TEMP	2nd board Min Temperature	0.1 Kelvins
SEMA_EAPI_ID_BOARD_2ND_SYSTEM_MAX_TEMP	2nd board Max Temperature	0.1 Kelvins
SEMA_EAPI_ID_BOARD_2ND_SYSTEM_STARTUP_TEMP	2nd board Startup Temperature	0.1 Kelvins
SEMA_EAPI_ID_BOARD_BMC_FLAG	status of the BMC.	See note 4
SEMA_EAPI_ID_BOARD_BMC_STATUS	status of the BMC.	Information for problem analysis

Note 1:

Restart reason

0x00: UNKNOWN

0x20: SW_RESET

0x30: HW_RESE

0x40: WATCHDOG

0x50: BIOS_FAUL

0x60: POWER_DOWN

0x70: POWER_LOSS

0x80: POWER_CYCLE

0x90: VIN_DROP

0xA0: PWR_FAIL

Note 2:

The meaning of BMC capabilities bits

Bit 0: Uptime & power cycles counter

Bit 1: System restart event

Bit 2: 1k USER flash memory available

Bit 3: Watchdog

Bit 4: Temperature sensors installed/available

Bit 5: Voltage sensors installed/available

Bit 6: Storage of failure reason available (exception code)

Bit 7: Bootloader timeout programmable

Bit 8: Display control available

Bit 9: Power up watchdog available

Bit 10: Current sensors installed/available

Bit 11: Bootcounter

Bit 12: Input-Voltage0 V10:00=5 V, 01=12 V, 10+11=reserved

Bit 13: Input-Voltage1 V10:00=5 V, 01=12 V, 10+11=reserved

Bit 14: Rsense for Input-Voltage: 0=8 mR, 1=4 mR

Bit 15: Fail-Safe-BIOS supported

Bit 16: Ext. I^2C bus #1 available

Bit 17: Ext. I^2C bus #2 available

Bit 18: Programmable CPU fan available

Bit 19: Programmable system fan available

Bit 20: AT/ATX mode supported

Bit 21: Thermal SCI supported

Bit 22: Power up to last state

Bit 23: Backlight restore

Bit 24: DTS register

Bit 25: DTS register offset

Bit 26: Smart fan #3

Bit 27: Smart fan #4

Bit 28: TIVA GPIOs support (12 GPIOs)

Bit 29: Ext. I^2C bus #3 available

Bit 30: Ext. I^2C bus #4 available

Bit 31: BMC is from TIVA type

Note 3:

The meaning of extended BMC capabilities bits

Bit 32: Board2 Temperature: 0 = not available

Bit 33: PEC protocol: 1 = supported

Bit 34: SoftFan: 1 = supported

Bit 35: Error log: 1 = supported

Bit 36: 1-Wire: 1 = supported

Bit 37: Wake-by-BMC: 1 = supported

Bit 38: GPIO ADC: 1 = supported

Note 4:

Bit 7: BIOS Select

0=Standard BIOS, 1=Fail-Safe-BIOS is active

Bit 6: ATX mode

0=AT mode, 1=ATX mode

Bit 5: reserved

Bit 0-4: Exception code (note: code is board specific, please refer to respective HW manual or use SemaEApiBoardGetExceptionDescription).

2.2.4 SemaEApiBoardGetVoltageMonitor

```
uint32_t
EAPI_CALLTYPE
SemaEApiBoardGetVoltageMonitor (
__IN uint32_t Handler , /* handler for remote target */
__IN uint32_t    Id          , /* Value Id */
__OUT uint32_t *pValue ,       /* Return Value */
__OUT char *pBuffer /* Return Voltage description */
);
```

Description

Get text information and values of platform‘s voltage.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	Selects the Get Value Sub function ID (refer to SEMA Extend ID)
__OUT	pValue	Pointer to a buffer that receives the value's data
__OUT	pBuffer	Pointer to a buffer that receives the description string

SEMA Extend ID

ID	Description	Units/Format
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN0	Get voltage from channel 0	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN1	Get voltage from channel 1	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN2	Get voltage from channel 2	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN3	Get voltage from channel 3	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN4	Get voltage from channel 4	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN5	Get voltage from channel 5	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN6	Get voltage from channel 6	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN7	Get voltage from channel 7	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN8	Get voltage from channel 8	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN9	Get voltage from channel 9	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN10	Get voltage from channel 10	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN11	Get voltage from channel 11	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN12	Get voltage from channel 12	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN13	Get voltage from channel 13	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN14	Get voltage from channel 14	millivolts & string
SEMA_EAPI_ID_HWMON_VOLTAGE_AIN15	Get voltage from channel 15	millivolts & string

2.2.5 SemaEApiBoardGetErrorLog

```
uint32_t
EAPI_CALLTYPE
SemaEApiBoardGetErrorLog (
__IN uint32_t Handler , /* handler for remote target */
__IN uint32_t position          , /* The position of error log */
__OUT uint32_t *ErrorNum;
__OUT uint8_t *Flags;
__OUT uint8_t *RestartEvent;
__OUT uint32_t *PwrCycles;
__OUT uint32_t *Bootcount;
__OUT uint32_t *Time;
__OUT uint16_t *Status;
__OUT signed char *CPUtemp;
__OUT signed char *Boardtemp;
);
```

Description

Get Error number history in the BMC. Failures in the Power Sequence are shown on the BMC status LED at the time where the issue occurs. The Error Log buffer stores those Power Sequence issues in an Error Log buffer. Beside of the displayed Error Code, the Error Log also stores 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

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Position	The pointer point to the log to be read
__OUT	ErrorNum	Error Number. To get detailed error description, please use SemaEApiBoardGetErrorNumberDescription
__OUT	Flags	Exception Code, selected BIOS and Power Mode (see SemaEApiBoardGetValue function and SEMA_EAPI_ID_BOARD_BMC_FLAG parameter) To get detailed error description from exception code, please use SemaEApiBoardGetExceptionDescription
__OUT	RestartEvent	System Restart Event (see SemaEApiBoardGetValue

		function and SEMA_EAPI_ID_BOARD_RESTART_EVENT parameter
__OUT	PwrCycles	Power Cycles Counter
__OUT	Bootcount	Boot Counter
__OUT	Time	Ontime Counter (seconds)
__OUT	Status	Status information from BMC status command
__OUT	CPUtemp	Current CPU temperature (if available)
__OUT	Boardtemp	Current Board temperature (if available)

2.2.6 SemaEApiBoardGetCurrentPosErrorLog

```
uint32_t
EAPI_CALLTYPE
SemaEApiBoardGetCurrentPosErrorLog (
__IN uint32_t Handler , /* handler for remote target */
__OUT uint32_t *ErrorNum;
__OUT uint8_t *Flags;
__OUT uint8_t *RestartEvent;
__OUT uint32_t *PwrCycles;
__OUT uint32_t *Bootcount;
__OUT uint32_t *Time;
__OUT uint16_t *Status;
__OUT signed char *CPUtemp;
__OUT signed char *Boardtemp;
);
```

Description

Get Error number history in the BMC. Failures in the Power Sequence are shown on the BMC status LED at the time where the issue occurs. The Error Log buffer stores those Power Sequence issues in an Error Log buffer. Beside of the displayed Error Code, the Error Log also stores information about the actual state and counters for better tracking of the issues. After retrieving the Error Log, the position will be automatically moved to next position (previous record).

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__OUT	ErrorNum	Error Number. To get detailed error description, please

		use SemaEApiBoardGetErrorNumberDescription
__OUT	Flags	Exception Code, selected BIOS and Power Mode (see SemaEApiBoardGetValue function and SEMA_EAPI_ID_BOARD_BMC_FLAG parameter) To get detailed error description from exception code, please use SemaEApiBoardGetExceptionDescription
__OUT	RestartEvent	System Restart Event (see SemaEApiBoardGetValue function and SEMA_EAPI_ID_BOARD_RESTART_EVENT parameter
__OUT	PwrCycles	Power Cycles Counter
__OUT	Bootcount	Boot Counter
__OUT	Time	Ontime Counter (seconds)
__OUT	Status	Status information from BMC status command
__OUT	CPUtemp	Current CPU temperature (if available)
__OUT	Boardtemp	Current Board temperature (if available)

2.2.7 SemaEApiBoardGetErrorNumberDescription

```
uint32_t
EAPI_CALLTYPE
SemaEApiBoardGetErrorNumberDescription (
__IN uint32_t Handler , /* handler for remote target */
__IN uint32_t ErrorNum, /* Error number */
__OUT char *pBuffer /* Return error description */
);
```

Description

Get text information of the Error number in the error log (get from SemaEApiBoardGetErrorLog).

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	ErrorNum	The Error number (Exception Code) defined in HW manual
__OUT	pBuffer	The description of error number

2.2.8 SemaEApiBoardGetExceptionDescription

```
uint32_t
EAPI_CALLTYPE
SemaEApiBoardGetExceptionDescription(
        __IN uint32_t BoardHandler, /* handler for remote target */
        __IN uint8_t ExceptionCode, /* Exception Code*/
        __OUT      char * pBuffer/* Return exception description */
);
```

Description

Get text information of the Exception Code.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	ExceptionCode	The Exception Code (defined in HW manual)
__OUT	pBuffer	The description of exception code

2.3 CPU

SEMA EAPI functions provide an interface to control or access CPU information including:

- Get static information (e.g. name, cache, frequency)
- Get dynamic information (e.g. CPU usage)
- Get/set CPU operating mode

2.3.1 SemaEApiCPUGetString

```
uint32_t
EAPI_CALLTYPE
SemaEApiCPUGetString(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Value Id */
__OUT char *pBuffer , /* Destination pBuffer */
__INOUT uint32_t *pBufLen /* pBuffer Length */
);
```

Description

Detailed text information about the CPU.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	Selects the Get Value Sub function Id (refer to SEMA Extend ID)
__OUT	pBuffer	Pointer to a buffer that receives the string data
__INOUT	pBufLen	pBuffer length
__OUT	pValue	Pointer to a buffer that receives the value's data

SEMA Extend ID

ID	Description	Units/Format
SEMA_EAPI_ID_MODEL_NAME	CPU model name	string

2.3.2 SemaEApiCPUGetValue

```
uint32_t
EAPI_CALLTYPE
SemaEApiCPUGetValue(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Value Id */
__OUT uint32_t *pValue /* Return Value */
);
```

Description

Detailed information about the CPU.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	Selects the Get Value Sub function Id (refer to SEMA Extend ID)
__OUT	pValue	Pointer to a buffer that receives the value's data

SEMA Extend ID

ID	Description	Units/Format
SEMA_EAPI_ID_CPU_FREQUENCY	Provides information about CPU frequency	MHz
SEMA_EAPI_ID_CPU_OPERATION_MODE	Provides information about the current CPU operating mode	See SEMA_CPU_PERFORMANCE_ID
SEMA_EAPI_ID_CPU_COUNT	Number of CPUs	
SEMA_EAPI_ID_CPU_CORE_COUNT	Number of cores of each CPU	
SEMA_EAPI_ID_CPU_L1_CACHE	L1 cache size	kB
SEMA_EAPI_ID_CPU_L2_CACHE	L2 cache size	kB
SEMA_EAPI_ID_CPU_L3_CACHE	L3 cache size	kB
SEMA_EAPI_ID_CPU_USAGE	The current CPU usage	percentage

SEMA_CPU_PERFORMANCE_ID

Linux

To enable different CPU performance levels in Linux, edit the /etc/rc.modules file using any text editor. Add the following lines to the rc.modules file:

- modprobe cpufreq_conservative
- modprobe cpufreq_ondemand
- modprobe cpufreq_powersave
- modprobe cpufreq_stats
- modprobe cpufreq_userspace

Enable rc.modules to be executabled

chmod +x /etc/rc.modules

Note: The command may need to be run as super user or root.

The support of these modes is different with the Linux distributions and CPU chipset. Please contact Linux vendor for detail.

ID	Description	Note
CPU_HIGH_PERFORMANCE_MODE	Sets the CPU statically to the highest frequency	
CPU_POWERSAVE_MODE	Sets CPU statically to the lowest frequency	
CPU_ONDEMAND_MODE	Sets the CPU depending on the current usage	
CPU_CONSERVATIVE_MODE	Sets the CPU depending on the current usage. It differs in behavior to CPU_ONDEMAND_MODE that it gracefully increases and decreases the CPU speed rather than jumping to max speed the moment there is any load on the CPU.	
CPU_USERSPACE_MODE	Does not support this mode in set operation. *1	

*1: If customer wants to configure userspace mode manually, please use the following command.

1.set the governor:

#> cpupower frequency-set --governor userspace

2.set the frequency:

#> cpupower --cpu all frequency-set --freq ???MHz

(???MHz-> the MHz you want set. For example 800 MHz)

Windows

ID	Effective setting in Windows
CPU_HIGH_PERFORMANCE_MODE	Powercfg -setacvalueindex scheme_current sub_processor PROCTHROTTLEMAX 100 Powercfg -setacvalueindex scheme_current sub_processor CPMINCORES 100
CPU_POWERSAVE_MODE	Powercfg -setacvalueindex scheme_current sub_processor PROCTHROTTLEMAX 50 Powercfg -setacvalueindex scheme_current sub_processor CPMINCORES 0 Powercfg -setacvalueindex scheme_current sub_processor CPMAXCORES 0
CPU_ONDEMAND_MODE	Powercfg -setacvalueindex scheme_current sub_processor PROCTHROTTLEMAX 80 Powercfg -setacvalueindex scheme_current sub_processor CPMINCORES 0 Powercfg -setacvalueindex scheme_current sub_processor CPMAXCORES 75
CPU_CONSERVATIVE_MODE	Powercfg -setacvalueindex scheme_current sub_processor PROCTHROTTLEMAX 75 Powercfg -setacvalueindex scheme_current sub_processor CPMINCORES 0 Powercfg -setacvalueindex scheme_current sub_processor CPMAXCORES 50
CPU_USERSPACE_MODE	Not supported

2.3.3 SemaEApiCPUSetValue

```
uint32_t
EAPI_CALLTYPE
SemaEApiCPUSetValue(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Value Id */
__IN uint32_t Value /* Value */
);
```

Description

Set CPU performance mode.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	Selects the Get Value Sub function ID (refer to SEMA Extend ID)
__IN	Value	The value's data

SEMA Extend ID

ID	Description	Units/Format
SEMA_EAPI_ID_CPU_OPERATION_MODE	Gets information about the current CPU operating mode	See SEMA_CPU_PERFORMANCE_ID

2.4 Memory

SEMA EAPI functions provide an interface to access memory information, including:

- Get static information (e.g. memory size, frequency)
- Get dynamic information (e.g. memory usage)

2.4.1 SemaEApiMemoryGetValue

```
uint32_t
EAPI_CALLTYPE
SemaEApiMemoryGetValue(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Value Id */
__OUT uint32_t *pValue /* Return Value */
);
```

Description

Information about the memory.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	Selects the Get Value Sub function ID (refer to SEMA Extend ID)
__OUT	pValue	Pointer to a buffer that receives the value's data

SEMA Extend ID

ID	Description	Units/Format
SEMA_EAPI_ID_MEMORY_FREQUENCY	Provides information about memory frequency	MHz
SEMA_EAPI_ID_MEMORY_TOTAL	The total size of memory	MB
SEMA_EAPI_ID_MEMORY_FREE	The size of free memory	MB

2.5 Network

SEMA EAPI functions provide an interface to access network information, including:

- Get static information: IP addr, Mac addr …
- Get dynamic information: TX/RX byte count …

SEMA Device ID

ID	Description
SEMA_EAPI_ID_NETWORK_ETH0	Ethernet 0
SEMA_EAPI_ID_NETWORK_ETH1	Ethernet 1
SEMA_EAPI_ID_NETWORK_ETH2	Ethernet 2
SEMA_EAPI_ID_NETWORK_ETH3	Ethernet 3

2.5.1 SemaEApiNetworkGetString

```
uint32_t
EAPI_CALLTYPE
SemaEApiNetworkGetString(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Value Id */
__IN uint32_t DeviceId , /* Device Id */
__OUT char *pBuffer , /* Destination pBuffer */
__INOUT uint32_t *pBufLen /* pBuffer Length */
);
```

Description

Information about the network interface.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	Selects the Get Value Sub function ID (refer to SEMA Extend ID)
__IN	DeviceId	Selects the device Id (refer to SEMA device ID)
__OUT	pBuffer	Pointer to a buffer that receives the string data
__INOUT	pBufLen	pBuffer Length

SEMA Extend ID

ID	Description	Units/Format
SEMA_EAPI_ID_NETWORK_ADAPTER_NAME	Description of the network interface	string
SEMA_EAPI_ID_NETWORK_IP_ADDRESS	IP address	string
SEMA_EAPI_ID_NETWORK_SUBMASK	Submask	string
SEMA_EAPI_ID_NETWORK_GATEWAY	Gateway	string
SEMA_EAPI_ID_NETWORK_MAC	MAC address	string

2.5.2 SemaEApiNetworkGetValue

```
uint32_t
EAPI_CALLTYPE
SemaEApiNetworkGetValue(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Value Id */
__IN uint32_t DeviceId , /* Device Id */
__OUT uint32_t *pValue /* Return Value */
);
```

Description

Information about the network.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	Selects the Get Value Sub function ID (refer to SEMA Extend ID)
__IN	DeviceId	Selects the device ID (refer to SEMA device ID)
__OUT	pValue	Pointer to a buffer that receives the value's data

SEMA Extend ID

ID	Description	Units/Format
SEMA_EAPI_ID_NETWORK_TX	The current tx byte count (not supported by Windows)	Byte
SEMA_EAPI_ID_NETWORK_RX	The current rx byte count (not supported by Windows)	Byte

2.6 Backlight

The SEMA EAPI functions support backlight brightness and backlight enable.

Backlight ID

Selects the flat panel display.

Description

ID	Description
EAPI_ID_BACKLIGHT_1	Backlight Local Flat Panel 1
EAPI_ID_BACKLIGHT_2	Backlight Local Flat Panel 2
EAPI_ID_BACKLIGHT_3	Backlight Local Flat Panel 3

Refer to PICMG EAPI 1.0 for more details.

2.6.1 SemaEApiVgaGetBacklightEnable

```
uint32_t
EAPI_CALLTYPE
SemaEApiVgaGetBacklightEnable(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Backlight Id */
__OUT uint32_t *pEnable /* Backlight Enable */
);
```

Description

Returns current Backlight Enable state for specified Flat Panel

Refer to PICMG EAPI 1.0, **EApiVgaGetBacklightEnable**.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__OUT	pEnable	Pointer to a buffer that receives the current backlight enable state.

2.6.2 SemaEApiVgaSetBacklightEnable

```
uint32_t
EAPI_CALLTYPE
SemaEApiVgaSetBacklightEnable(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Backlight Id */
__IN uint32_t Enable /* Backlight Enable */
);
```

Description

Refer to PICMG EAPI 1.0 **EApiVgaSetBacklightEnable**.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Enable	Backlight enable options

Backlight Enable Values

Name	Description
EAPI_BACKLIGHT_SET_ON	Requests/Signifies that the backlight is enabled
EAPI_BACKLIGHT_SET_OFF	Requests/Signifies that the backlight is disabled

2.6.3 SemaEApiVgaGetBacklightBrightness

```
uint32_t
EAPI_CALLTYPE
SemaEApiVgaGetBacklightBrightness (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Backlight Id */
__OUT uint32_t *pBright /* Backlight Brightness */
);
```

Description

Refer to PICMG EAPI 1.0 **EApiVgaGetBacklightBrightness**.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__OUT	pBright	Pointer to a buffer that receives the current backlight brightness level

2.6.4 SemaEApiVgaSetBacklightBrightness

```
uint32_t
EAPI_CALLTYPE
SemaEApiVgaSetBacklightBrightness (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Backlight Id */
__IN uint32_t Bright /* Backlight Brightness */
);
```

Description

Refer to PICMG EAPI 1.0 **EApiVgaSetBacklightBrightness**.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Bright	Backlight Brightness level

2.7 Storage

Each board can have several storage areas. A storage area is a piece of physical memory that usually provides persistent storage for the user's application(s). The only storage currently supported is EEPROM. The EAPI defines one user storage area with a minimal size of 32 bytes.

Storage Ids

ID	Description
EAPI_ID_STORAGE_STD	Standard Storage Area >=32 bytes for read/write access

2.7.1 SemaEApiStorageCap

```
uint32_t
EAPI_CALLTYPE
SemaEApiStorageCap (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Storage Area Id */
__OUT uint32_t *pStorageSize,    /* Total */
__OUT uint32_t *pBlockLength /* Write block length & alignment */
);
```

Description

Get the capabilities of the selected storage area. Refer to PICMG EAPI 1.0 **EApiStorageCap**

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	Storage Area ID
__OUT	pStorageSize	Pointer to a buffer that receives storage area size. This parameter can be NULL if the data is not required.
__OUT	pBlockLength	Pointer to a buffer that receives the storage areas alignment/Block size. This parameter can be NULL if the data is not required. The value must be used to calculate write block alignment and size.

2.7.2 SemaEApiStorageAreaRead

```
uint32_t
EAPI_CALLTYPE
SemaEApiStorageAreaRead (
__IN uint32_t BoardHandler, /* handler for remote target */
__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

Read data to the selected user data area.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	Storage Area ID
__IN	Offset	Storage area start address offset in bytes
__OUT	pBuffer	Pointer to a buffer that receives the read data.
__IN	BufLen	Size, in bytes, of the buffer pointed to by the pBuffer parameter
__IN	ByteCnt	Size, in bytes, of the information read to the buffer pointed to by the pBuffer parameter

2.7.3 SemaEApiStorageAreaWrite

```
uint32_t
EAPI_CALLTYPE
SemaEApiStorageAreaWrite (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* Storage Area Id */
__IN uint32_t Offset , /* Byte Offset */
__IN void *pBuffer , /* Pointer to Data pBuffer */
__IN uint32_t ByteCnt /* Number of bytes to write*/
);
```

Description

Refer to PICMG EAPI 1.0 **EApiStorageAreaWrite**

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	Storage Area ID
__IN	Offset	Storage area start address offset in bytes. This value must be a multiple of *pBlockLength.
__IN	pBuffer	Pointer to a buffer containing the data to be stored
__IN	ByteCnt	Size, in bytes, of the information pointed to by the pBuffer parameter

2.8 I²C Bus

The SEMA I²C functions provide an interface to access the onboard I²C. You can use these functions to access I²C devices. (Please refer the hardware spec. to know how many I²C interfaces been supported on your products). Refer to PICMG EAPI 1.0 for more details.

I2C Bus Ids

ID	Description
EAPI_ID_I2C_EXTERNAL	Baseboard I²C Interface
EAPI_ID_I2C_LVDS_1	LVDS/EDP 1 Interface
EAPI_ID_I2C_LVDS_2	LVDS/EDP 2 Interface
SEMA_EAPI_ID_I2C_EXTERNAL_2	2nd external I²C interface
SEMA_EAPI_ID_I2C_EXTERNAL_3	3nd external I²C interface

2.8.1 SemaEApiI2CGetBusCap

```
uint32_t
EAPI_CALLTYPE
SemaEApiI2CGetBusCap (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* I2C Bus Id */
__OUT uint32_t *pMaxBlkLen /* Max Block Length Supported on this interface */
);
```

Description

Returns the capabilities of the selected I²C bus. Refer to PICMG EAPI 1.0 **EapiI2CGetBusCap**.

Parameters

in/out	Parameter name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	I²C Bus ID
__OUT	pMaxBlkLen	Size in bytes, Pointer to a buffer that receives the maximum transfer block length for the given interface. Please note that care must be taken if using in combination with ***EapiI2CWriteTransfer*** as the maximum data payload length will be ***pMaxBlkLen***-(write overhead). So for example a 10 Bit Addressed device with extended command has a write overhead of 3 bytes. Address Byte 2 and 2 bytes command.

2.8.2 SemaEApiI2CWriteReadRaw

```
uint32_t
EAPI_CALLTYPE
SemaEApiI2CWriteReadRaw (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* I2C Bus Id */
__IN uint8_t Addr , /* Encoded 7Bit I2C Device Address*/
__INOPT void *pWBuffer , /* Write Data pBuffer */
__IN uint32_t WriteBCnt, /* Number of Bytes to write */
__OUTOPT void *pRBuffer , /* Read Data pBuffer */
__IN uint32_t RBufLen , /* Data pBuffer Length */
__IN uint32_t ReadBCnt /* Number of Bytes to Read*/
);
```

Description

Universal function for read and write operations to the I2C bus. Refer to PICMG EAPI 1.0 **EApiI2CWriteReadRaw**.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	I^2C Bus ID
__IN	Addr	First Byte of I^2C Device Address.
__INOPT	pWBuffer	Pointer to a buffer containing the data to be transferred. This parameter can be NULL if the data is not required.
__IN	WriteBCnt	Size, in bytes, of the information pointed to by the **pWBuffer** parameter plus 1 If **pWBuffer** is NULL this must be zero or one.
__OUTOPT	pRBuffer	Pointer to a buffer that receives the read data. This parameter can be NULL if the data is not required.
__IN	RBufLen	Size, in bytes, of the buffer pointed to by the **pRBuffer** parameter. If the buffer specified by **pRBuffer** parameter is not large enough to hold the data, the function returns the value **EAPI_STATUS_MORE_DATA** If **pRBuffer** is NULL this must be zero.
__IN	ReadBCnt	Size, in bytes, to be read to **pRBuffer** plus 1 If **pRBuffer** is NULL this must be zero or one.

2.8.3 SemaEApiI2CReadTransfer

```
uint32_t
EAPI_CALLTYPE
SemaEApiI2CReadTransfer (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* I2C Bus Id */
__IN uint32_t Addr , /* Encoded 7 Bit I2C Device Address*/
__IN uint32_t Cmd , /* I2C Command/Offset */
__OUT void *pBuffer , /* Transfer Data pBuffer */
__IN uint32_t BufLen , /* Data pBuffer Length */
__IN uint32_t ByteCnt /* Byte Count to read */
);
```

Description

Reads from a specific register in the selected I^2C device.

Reads from I^2C device at the I^2C address ***Addr*** the amount of ***ByteCnt*** bytes to the buffer ***pBuffer*** while using the device specific command ***Cmd***. Depending on the addressed I^2C device ***Cmd*** can bea specific command or a byte offset.

Refer to PICMG EAPI 1.0 **EApiI2CReadTransfer**

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	IVC Bus ID
__IN	Addr	Encoded 7 Bit I^2C Device Address
__IN	Cmd	Encoded I^2C Device Command / Index.
__OUT	pBuffer	Pointer to a buffer that receives the read data. This parameter can be NULL if the data is not required.
__IN	BufLen	Size, in bytes, of the buffer pointed to by the ***pBuffer*** parameter. If the buffer specified by ***pBuffer*** parameter is not large enough to hold the data, the function returns the value ***EAPI_STATUS_MORE_DATA***
__IN	ByteCnt	Size in bytes of data to be read

2.8.4 SemaEapiI2CWriteTransfer

```
uint32_t
EAPI_CALLTYPE
SemaEapiI2CWriteTransfer (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* I2C Bus Id */
__IN uint32_t Addr , /* Encoded 7 Bit I2C Device Address */
__IN uint32_t Cmd , /* I2C Command/Offset */
__IN void *pBuffer , /* Transfer Data pBuffer */
__IN uint32_t ByteCnt /* Byte Count to write */
);
```

Description

Refer to PICMG EAPI 1.0 **EapiI2CWriteTransfer**

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	I^2C Bus ID
__IN	Addr	Encoded 7 Bit I^2C Device Address
__IN	Cmd	Encoded I^2C Device Command / Index
__IN	pBuffer	Pointer to a buffer containing the data to be transferred
__IN	ByteCnt	Size, in bytes, of the information pointed to by the **pBuffer** parameter

2.8.5 SemaEApiI2CProbeDevice

```
uint32_t
EAPI_CALLTYPE
SemaEApiI2CProbeDevice (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* I2C Bus Id */
__IN uint32_t Addr /* Encoded 7 Bit I2C Device Address */
);
```

Description

Refer to PICMG EAPI 1.0 **EApiI2CProbeDevice**

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	I^2C Bus ID
__IN	Addr	Encoded 7 bit I^2C device address

2.9 Watchdog

The SEMA EAPI library watchdog functions support two watchdog controls of the board. One is the run-time watchdog, another one is the power-up watchdog. If the watchdog begins and reaches the pre-set time, it will access the CPU's RESET signal to reset the system. The run-time watchdog is used in the OS to monitor the process or the application. It's a onetime watchdog and will be cleared automatically after system reboot. The power-up watchdog is supervising the whole power-up and boot process as well as the loading the operating systems. It will continuously keep resetting the system if the user doesn't stop it. The power-up watchdog is used to guard the system power-up.

Caution: Please make sure your application stops the power-up watchdog after the system boots up successfully. Also make sure you set the correct timer to let your application stop the watchdog in time. Too short timer setting will cause the system to reboot infinitely.

2.9.1 SemaEApiWDogGetCap

```
uint32_t
EAPI_CALLTYPE
SemaEApiWDogGetCap (
__IN uint32_t BoardHandler, /* handler for remote target */
__OUTOPT uint32_t *pMaxDelay ,/* Max. supported delay in msec */
__OUTOPT uint32_t *pMaxEventTimeout ,/* Max. supported event timeout in
msec, 0 == Unsupported */
__OUTOPT uint32_t *pMaxResetTimeout/* Max. supported reset timeout in
msec */
);
```

Description

Get the capabilities of the watchdog timer. Refer to PICMG EAPI 1.0 **EApiWDogGetCap**.

This function supports only the run-time watchdog.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__OUTOPT	pMaxDelay	Pointer to a buffer that receives maximum supported initial delay time of the watchdog timer in miliseconds
__OUTOPT	pMaxEventTimeout	Pointer to a buffer that receives maximum supported event timeout of the watchdog timer in miliseconds
__OUTOPT	pMaxResetTimeout	Pointer to a buffer that receives maximum supported event timeout of the watchdog timer in miliseconds

2.9.2 SemaEApiWDogStart

```
uint32_t
EAPI_CALLTYPE
SemaEApiWDogStart (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t delay , /* Delay in msec */
__IN uint32_t EventTimeout /* Timeout in msec */
__IN uint32_t ResetTimeout /* Reset in msec */
);
```

Description

Start the watchdog timer and set the parameters. This function supports only the run-time watchdog.

To adjust the parameters, the watchdog must be stopped via **Sema*EApiWDogStop*** and then **Sema*EApiWDogStart*** must be called again with the new values. If the hardware implementation of the watchdog timer does not allow to set exactly the selected timing, the SEMA EAPI ***shall*** select the next possible longer timing.

Refer to PICMG EAPI 1.0 **EApiWDogStart**.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	delay	Initial delay for the watchdog timer in milliseconds. (currently not supported by SEMA EAPI, set it as 0)
__IN	EventTimeout	Watchdog timeout interval in milliseconds to trigger an event. (currently not supported by SEMA EAPI, set it as 0)
__IN	ResetTimeout	Watchdog timeout interval in milliseconds to trigger a reset.

2.9.3 SemaEApiWDogTrigger

```
uint32_t
EAPI_CALLTYPE
SemaEApiWDogTrigger (
__IN uint32_t BoardHandler/* handler for remote target */
);
```

Description

Trigger the watchdog timer. Refer to PICMG EAPI 1.0 **EApiWDogTrigger**.

This function supports only the run-time watchdog.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board

2.9.4 SemaEApiWDogStop

```
uint32_t
EAPI_CALLTYPE
SemaEApiWDogStop (
__IN uint32_t BoardHandler/* handler for remote target */
);
```

Description

Stops the operation of the watchdog timer. Refer to PICMG EAPI 1.0 **EApiWDogStop**.

This function supports only the run-time watchdog.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board

2.9.5 SemaEApiPwrUpWDogStart

```
uint32_t
EAPI_CALLTYPE
SemaEApiPwrUpWDogStart (
__IN uint32_t BoardHandler/* handler for remote target */
__IN uint32_t ResetTimeout /* Reset in sec */

);
```

Description

Start the power-up watchdog timer with timeout. This function supports only the power-up watchdog.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	ResetTimeout	Watchdog timeout interval in milliseconds to trigger a reset.

2.9.6 SemaEApiPwrUpWDogStop

```
uint32_t
EAPI_CALLTYPE
SemaEApiPwrUpWDogStop (
__IN uint32_t BoardHandler/* handler for remote target */
);
```

Description

Stops the operation of the power-up watchdog timer. This function supports only the power-up watchdog.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board

2.10 GPIO

SEMA device specifies pins for general purpose I/Os. The SEMA EAPI provides a set of functions to control these hardware GPIO pins. Refer to PICMG EAPI 1.0 for more details. Moreover some SEMA devices allow to assign 1-Wire (see chapter 2.17) or A/D conversion capability to GPIO pins. Those GPIO pins can be used to connect 1-Wire bus devices or to perform measurements with the integrated A/D converter then.

Single GPIO addressing

Each GPIO pin can be addressed individually. Please refer to the platform specific header file (COM Express: EApiCOM0.h) for the detailed pin assignment. ***_GPIO_ID0***

Individual GPIO Ids	Description
EAPI_GPIO_ID0	'GPIO 0' Bit mapped to Bit 0
EAPI_GPIO_ID1	'GPIO 1' Bit mapped to Bit 0
EAPI_GPIO_ID2	'GPIO 2' Bit mapped to Bit 0
EAPI_GPIO_ID3	'GPIO 3' Bit mapped to Bit 0
EAPI_GPIO_ID4	'GPIO 4' Bit mapped to Bit 0
EAPI_GPIO_ID5	'GPIO 5' Bit mapped to Bit 0
EAPI_GPIO_ID6	'GPIO 6' Bit mapped to Bit 0
EAPI_GPIO_ID7	'GPIO 7' Bit mapped to Bit 0

Parallel GPIO addressing

A group of selected GPIO pins can be addressed simultaneously.

Multiple GPIO Ids	Description
EAPI_ID_GPIO_BANK00	GPIO 0-31 Bit mapped to Bit 0-31

Bit-mask Bit States

Multiple GPIO Ids	Description
EAPI_GPIO_BITMASK_SELECT	Used to specify that the Specific GPIO is selected
EAPI_GPIO_BITMASK_NOSELECT	Used to specify that the Specific GPIO is not selected, and should be ignored.

Level Bit States

Name	Description
EAPI_GPIO_LOW	Used to specify/signify that the Specific GPIO is low (not activated).
EAPI_GPIO_HIGH	Used to specify/signify that the Specific GPIO is high (activated).

Direction Bit States

Name	Description
EAPI_GPIO_INPUT	Used to specify/signify that the Specific GPIO is in input mode.
EAPI_GPIO_OUTPUT	Used to specify/signify that the Specific GPIO is in output mode.

2.10.1 SemaEApiGPIOGetDirectionCaps

```
uint32_t
EAPI_CALLTYPE
SemaEApiGPIOGetDirectionCaps (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* GPIO Id */
__OUTOPT uint32_t *pInputs , /* Supported GPIO Input 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. The direction of this ports can be configured by **SEMAEApiGPIOSetDirection.** Refer to PICMG EAPI 1.0 **EApiGPIOGetDirectionCaps**.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	GPIO ID
__OUTOPT	pInputs	Pointer to a buffer that receives the bit mask of the supported inputs.
__OUTOPT	pOutputs	Pointer to a buffer that receives the bit mask of the supported inputs.

2.10.2 SemaEApiGPIOGetDirection

```
uint32_t
EAPI_CALLTYPE
SemaEApiGPIOGetDirection (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* GPIO Id */
__IN uint32_t Bitmask , /* Bit mask of Affected Bits */
__OUT uint32_t *pDirection /* Current Direction */
);
```

Description

Reads the current configuration of the selected GPIO ports. Refer to PICMG EAPI 1.0 **EApiGPIOGetDirection**.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	GPIO ID
__IN	Bitmask	Bit mask
__OUT	pDirection	Pointer to a buffer that receives the direction of the selected GPIO ports

2.10.3 SemaEApiGPIOSetDirection

```
uint32_t
EAPI_CALLTYPE
SemaEApiGPIOSetDirection (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* GPIO Id */
__IN uint32_t Bitmask , /* Bit mask of Affected Bits*/
__IN uint32_t Direction /* Direction */
);
```

Description

Sets the configuration for the selected GPIO ports. Refer to PICMG EAPI 1.0 **EApiGPIOSetDirection**.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	GPIO ID
__IN	Bitmask	Bit mask. Only the ports with the level ***EAPI_GPIO_HIGH*** are processed.
__IN	Direction	Sets the direction of the selected GPIO ports. Bits with the value **EAPI_GPIO_INPUT** are inputs, bits with ***EAPI_GPIO_OUTPUT*** are outputs.

2.10.4 SemaEApiGPIOGetLevel

```
uint32_t
EAPI_CALLTYPE
SemaEApiGPIOGetLevel (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* GPIO Id */
__IN uint32_t Bitmask , /* Bit mask of Affected Bits*/
__OUT uint32_t *pLevel /* Current Level */
);
```

Description

Read the from GPIO ports. Refer to PICMG EAPI 1.0 **EApiGPIOGetLevel**.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Others	GPIO ID
__IN	Bitmask	Bit mask. Only selected bits are returned. Unselected bits return ***EAPI_GPIO_LOW.***
__OUT	pLevel	Pointer to a buffer that receives the GPIO level. Results can be read on a bit level.

2.10.5 SemaEApiGPIOSetLevel

```
uint32_t
EAPI_CALLTYPE
SemaEApiGPIOSetLevel (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* GPIO Id */
__IN uint32_t Bitmask , /* Bit mask of Affected Bits*/
__IN uint32_t Level /* Level */
);
```

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. Any GPO level setting to the pins witch are configured with “Input” will not take effect. This function call will not return this as an error. Please make sure the Bitmask also

mask the Input pins. Refer to PICMG EAPI 1.0 **EApiGPIOSetLevel**.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	GPIO ID
__IN	Bitmask	Value for a bit mask. Only selected bits are changed. Unselected bits remain unchanged.
__IN	Level	Input level of the selected GPIO port. Output for single ports is on a bit level.

2.10.6 SemaEApiGPIOPinGetCap

```
uint32_t
EAPI_CALLTYPE
SemaEApiGPIOPinGetCap (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint8_t Id , /* GPIO Id */
__OUT uint32_t *Capabilites /* GPIO pins capabilities value */
);
```

Description

Read the capabilities value of GPIO pins.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	GPIO ID
__OUT	Capabilities	Pointer to a buffer that receives the value's data. 0x1 = GPIO 0x2 = 1-Wire 0x4 = A/D converter (ADC) 0x0 = not available

2.10.7 SemaEApiGPIOPinSetConfig

```
uint32_t
SemaEApiGPIOPinSetConfig (
__IN uint32_t BoardHandler, /* handler for remote target */
```

```
__IN uint8_t Id , /* GPIO Id */
__IN uint32_t Configuration /* PIN configuration value */
);
```

Description

Sets the configuration of GPIO pins for the selected GPIO capabilities.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	GPIO ID
__IN	Configuration	0x0 = GPIO_OUTPUT 0x1 = GPIO_INPUT 0x2 = GPIO_1WIRE 0x4 = GPIO_ A/D converter (ADC)

2.10.8 SemaEApiGPIOPinGetConfig

```
uint32_t
SemaEApiGPIOPinGetConfig (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint8_t Id , /* GPIO Id */
__OUT uint32_t *Configuration /* PIN configuration value */
);
```

Description

Read the configuration value of GPIO pins for the selected GPIO capabilities.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	GPIO ID
__IN	Configuration	0x0 = GPIO_OUTPUT 0x1 = GPIO_INPUT 0x2 = GPIO_1WIRE 0x4 = GPIO_ A/D converter (ADC)

2.11 Fan

SEMA EAPI support the control of the smart fan. It supports 4 level of scale. 4 power level can be set according the temperature.

2.11.1 Fan ID

Smartfan IDs	Description
SEMA_EAPI_ID_FAN_CPU	CPU fan
SEMA_EAPI_ID_FAN_SYSTEM_1	System fan 1
SEMA_EAPI_ID_FAN_SYSTEM_2	System fan 2
SEMA_EAPI_ID_FAN_SYSTEM_3	System fan 3

Fan mode	Description
SEMA_EAPI_FAN_MODE_AUTO	Smartfan on
SEMA_EAPI_FAN_MODE_ON	Turn on fan
SEMA_EAPI_FAN_MODE_OFF	Turn off fan

Temperature source IDs	Description
SEMA_FAN_TEMP_CPU	Temperature source from CPU
SEMA_FAN_TEMP_SYS	Temperature source from System

2.11.2 SemaEApiSmartFanSetTempSetpoints

```
uint32_t
EAPI_CALLTYPE
SemaEApiSmartFanSetTempSetpoints(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN int32_t   Id , /* FAN Id */
__IN int32_t   Level1, /* Level 1 */
__IN int32_t   Level2 /* Level 2*/
__IN int32_t   Level3, /* Level 3*/
__IN int32_t   Level4 /* Level 4*/
);
```

Description

Set temperature setpoints (signed, degree Celsius)

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	FAN ID
__IN	Level1	Level 1
__IN	Level2	Level 2
__IN	Level3	Level 3
__IN	Level3	Level 4

2.11.3 SemaEApiSmartFanGetTempSetpoints

```
uint32_t
EAPI_CALLTYPE
SemaEApiSmartFanGetTempSetpoints(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN int32_t Id , /* FAN Id */
__OUT int32_t * pLevel1, /* Level 1 */
__OUT int32_t * pLevel2 /* Level 2*/
__OUT int32_t * pLevel3, /* Level 3*/
__OUT int32_t * pLevel4 /* Level 4*/
);
```

Description

Get temperature setpoints (signed, degree Celsius)

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__OUT	pLevel1	Level 1
__OUT	pLevel2	Level 2
__OUT	pLevel3	Level 3
__OUT	pLevel3	Level 4

2.11.4 SemaEApiSmartFanSetPWMSetpoints

```
uint32_t
EAPI_CALLTYPE
SemaEApiSmartFanSetPWMSetpoints(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t    Id , /* FAN Id */
__IN uint32_t    Level1, /* Level 1 */
__IN uint32_t    Level2 /* Level 2*/
__IN uint32_t    Level3, /* Level 3*/
__IN uint32_t    Level4 /* Level 4*/
);
```

Description

Set PWM setpoints (valid range: 0... 100)

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	FAN ID
__IN	Level1	Level 1
__IN	Level2	Level 2
__IN	Level3	Level 3
__IN	Level3	Level 4

2.11.5 SemaEApiSmartFanGetPWMSetpoints

```
uint32_t
EAPI_CALLTYPE
SemaEApiSmartFanGetTempSetpoints(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* FAN Id */
__OUT uint32_t * pLevel1, /* Level 1 */
__OUT uint32_t * pLevel2 /* Level 2*/
__OUT uint32_t * pLevel3, /* Level 3*/
__OUT uint32_t * pLevel4 /* Level 4*/
);
```

Description

Get PWM setpoints (valid range: 0 ... 100)

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	FAN ID
__OUT	pLevel1	Level 1
__OUT	pLevel2	Level 2
__OUT	pLevel3	Level 3
__OUT	pLevel3	Level 4

2.11.6 SemaEApiSmartFanGetMode

```
uint32_t
EAPI_CALLTYPE
SemaEApiSmartFanGetMode(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* FAN Id */
__OUT uint32_t * pMode, /* Fan mode */
);
```

Description

Get Fan Mode.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	FAN ID
__OUT	pMode	Fan mode

2.11.7 SemaEApiSmartFanSetMode

```
uint32_t
EAPI_CALLTYPE
SemaEApiSmartFanSetMode(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* FAN Id */
__IN uint32_t Mode, /* Fan mode */
);
```

Description

Set Fan Mode.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	FAN ID
__IN	Mode	Fan mode

2.11.8 SemaEApiSmartFanGetTempSrc

```
uint32_t
EAPI_CALLTYPE
SemaEApiSmartFanGetTempSrc(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* FAN Id */
__OUT uint32_t* pTemp_src, /* Temperature source */
);
```

Description

Get temperature source of specified Fan ID.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	FAN ID
__OUT	pTemp_src	Temperature source

2.11.9 SemaEApiSmartFanSetTempSrc

```
uint32_t
EAPI_CALLTYPE
SemaEApiSmartFanSetTempSrc (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id , /* FAN Id */
__IN uint32_t Temp_src, /* Temperature source ID*/
);
```

Description

Set temperature source of specified Fan ID. After call this function, the smartfan will be triggered by the temperature source. Only CPU and System temperature could be set as temperature source of FANs. Some platform may support more system temperature, however the temperature can't be set as the source of FANs.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	FAN ID
__IN	Temp_src	Temperature source ID

2.12 BIOS/Bootloader

The SEMA EAPI provides the function to control active BIOS (X86) /Bootloader (ARM) and firmware update. Be careful when you using these functions. Make sure you have the right BIOS or Bootloader binary file and do not turn off the power when the update is proceeding. In the ARM system, please make sure that the write-protection of bootloader has been removed.

2.12.1 SemaEApiGetActiveBIOSBank

```
uint32_t
EAPI_CALLTYPE
SemaEApiGetActiveBIOSBank(
__IN uint32_t BoardHandler, /* handler for remote target */
__OUT uint32_t * pbank , /*active bank of BIOS/Bootloader */
);
```

Description

Gets selected BIOS/Bootloader (only available if Faile-Safe-BIOS/Bootloader is supported).

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__OUT	pbank	BIOS number/index: 0: Standard BIOS 1: Fail-Safe BIOS 2: External BIOS (SPI0 on carrier) 3: Internal BIOS (SPI0 on module)

2.12.2 SemaEApiSetActiveBIOSBank

```
uint32_t
EAPI_CALLTYPE
SemaEApiSetActiveBIOSBank(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t bank , /*active bank of BIOS/Bootloader */
);
```

Description

Select BIOS/Bootloader (only available if Fail-Safe-BIOS/Bootloader is supported). In some platforms, this only take effect after system full reset. Please power off and power on the system to make sure it work normally.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	bank	Bank number 0: Standard BIOS 1: Fail-Safe BIOS 2: External BIOS (SPI0 on carrier) 3: Internal BIOS (SPI0 on carrier)

2.12.3 SemaEApiBIOSUpdate

```
uint32_t
EAPI_CALLTYPE
SemaEApiBIOSUpdate(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN char* filename,    /*the filename of BIOS/Bootloader firmware */
__IN char* checksum    /*the MD5 checksum of BIOS/Bootloader firmware */
);
```

Description

Updating the BIOS/Bootloader firmware.

Only support for American Megatrends (AMI) BIOS.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	filename	The filename of BIOS/Bootloader firmware
__IN	checksum	The MD5 checksum of BIOS/Bootloader firmware

2.13 BMC Firmware

The SEMA EAPI provides the BMC firmware update function. Be careful when you using this function. Make sure you have the right firmware file and not turn off the power when the update is proceeding. With SEMA BMC 2, system will always reboot after delay. With SEMA BMC 3, reboot or shutdown is not necessary. To know which BMC on the board, please contact the ADLINK support team for more details.

2.13.1 SemaEApiBMCUpdate

```
uint32_t
EAPI_CALLTYPE
SemaEApiBMCUpdate(
__IN uint32_t BoardHandler, /* handler for remote target */
__IN char* FileName,    /*the filename of BMC firmware */
__IN char* CheckSum    /*the MD5 checksum of BMC firmware */
__IN bool bReboot,    /* Reboot operating system after FW update?    */
__IN uint32_t Delay    /*the delay in seconds for the reboot */

);
```

Description

Updating the BMC firmware.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	filename	The filename of BMC firmware
__IN	checksum	The MD5 checksum of BMC firmware
__IN	bReboot	System reboot after update
__IN	Delay	the delay in seconds for the reboot

2.14 System Boot

System boot provide the shutdown, restart, suspend and hibernate function for system state management.

Action ID

ID	Description
SEMA_EAPI_ID_SYSTEM_BOOT_SW_SHUTDOWN	Software shutdown the system
SEMA_EAPI_ID_SYSTEM_BOOT_SW_RESTART	Software restart the system
SEMA_EAPI_ID_SYSTEM_BOOT_SW_SUSPEND	Software suspend to ram (S3)
SEMA_EAPI_ID_SYSTEM_BOOT_SW_HIBERNATE	Software hibernate to disk (S4)

2.14.1 SemaEApiSystemBootSet

```
uint32_t
EAPI_CALLTYPE
SemaEApiSystemBootSet (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t Id ,/* The id of action    */
__IN uint32_t Delay /* The delay in seconds to trigger the action */
);
```

Description

System power management.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	The ID of action
__IN	delay	The delay in seconds to trigger the action

2.15 Heartbeat

Heartbeat service provides notification when target device are failed. The service only has the peer to peer connection without any hierarchy or cluster topology. This function does not support “localhost” heartbeat.

2.15.1 SemaEApiHeartbeat

```
uint32_t
EAPI_CALLTYPE
SemaEApiHeartbeat (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint32_t    Keepalive,    /* The time interval to check the alive */
__IN uint32_t    Deadtime,    /*The expired time for the determination of dead link*/
__IN char* Action /* filename of script or execute file to run when heartbeat expired */
);
```

Description

After starting the heartbeat, the remote server will keep watching it. If the remote server can't receive the heart beat with in deadtime, it will execute the action. Note if **SemaEApiLibUnInitialize** called without **SemaEApiHeartBeatStop** , this will make the client hearbeat stop and remote monitor will trigger the action after deadtime expired. The script file or execute file should be placed under "/etc/SEMA/exec/"(Linux) or "C:\SEMA\exec\" (Windows).

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Keepalive	The time interval to check the alive (in seconds)
__IN	Deadtime	The expired time for the determination of dead link (in seconds)
__IN	Action	Callback function when target device is dead

2.15.2 SemaEApiHeartBeatStop

```
uint32_t
EAPI_CALLTYPE
SemaEApiHeartbeat (
__IN uint32_t BoardHandler /* handler for remote target */
);
```

Description

Stop the hearbeat. Call this function when you want gracefully shut down the hearbeat monitor in remote server.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board

2.16 Disk Information

Disk information service provides detailed information about the HDD storage.

S.M.A.R.T.

Self-Monitoring, Analysis, and Reporting Technology, or S.M.A.R.T., is a monitoring system for hard drives to detect and report various indicators of reliability, in the hope of anticipating failures. It is implemented inside the drives. S.M.A.R.T. provides several ways of monitoring hard drive health.

2.16.1 SemaEApiDiskNum

```
uint32_t
EAPI_CALLTYPE
SemaEApiDiskNum (
__IN uint32_t    BoardHandler, /* handler for remote target */
__OUT uint32_t*    pDiskNumber,    /*number of disks*/
);
```

Description

Get the total number of disks found on system.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__OUT	pDiskNumber	Number of disks

2.16.2 SemaEApiDiskInfo

```
uint32_t
EAPI_CALLTYPE
SemaEApiDiskInfo (
__IN uint32_t    BoardHandler, /* handler for remote target */
__IN uint32_t    DiskIndex, /* index of Disk, start from 0*/
__OUT char*        pInfo,    /* detailed S.M.A.R.T. information */
__INOUT uint32_t*    pLength    /*length of info*/
);
```

Description

Get the detailed information of specified disk. The information will be retrieved in a XML formatted string.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	DiskIndex	Index of Disk, start from 0
__OUT	pInfo	Detailed S.M.A.R.T. information
__INOUT	pLength	Length of info

String output example:

```
<xml>
<Model_Family>Seagate Barracuda 7200.14 (AF)</Model_Family>
<Device_Model>ST500DM002-1BD142</Device_Model>
<Serial_Number>W2AM44XD</Serial_Number>
<LU_WWN_Device_Id>5 000c50 05cfbde61</LU_WWN_Device_Id>
<Firmware_Version>KC45</Firmware_Version>
<User_Capacity>500,107,862,016 bytes [500 GB]</User_Capacity>
<Sector_Sizes>512 bytes logical, 4096 bytes physical</Sector_Sizes>
<Rotation_Rate>7200 rpm</Rotation_Rate>
<Device>In smartctl database [for details use: -P show]</Device>
<ATA_Version>ATA8-ACS T13/1699-D revision 4</ATA_Version>
<SATA_Version>SATA 3.0, 6.0 Gb/s (current: 3.0 Gb/s)</SATA_Version>
<Local_Time>Sat Jun 21 03:10:14 2014 CST</Local_Time>
<SMART_support>Available - device has SMART capability.</SMART_support>
<SMART_enabled>Enabled</SMART_enabled>
</xml>
```

2.16.3 SemaEApiDiskSMARTAll

```
uint32_t
EAPI_CALLTYPE
SemaEApiDiskSMARTAll (
__IN uint32_t   BoardHandler, /* handler for remote target */
__IN uint32_t   DiskIndex, /* index of Disk, start from 0*/
__OUT char*       pInfo,   /* detailed S.M.A.R.T. information */
__OUT uint32_t*   pLength,   /*length of info*/
);
```

Description

Get the all the S.M.A.R.T. information of assigned disk. The information will be retrieve in a XML formatted string.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	DiskIndex	Index of Disk, start from 0
__OUT	pInfo	Detailed S.M.A.R.T. information
__OUT	pLength	Length of info

String output example:

```
<xml>
<SMART>
<ID number="1">
        <ATTRIBUTE_NAME>Raw_Read_Error_Rate</ATTRIBUTE_NAME>
        <FLAG>0x000f</FLAG>
        <VALUE>102</VALUE>
        <WORST>099</WORST>
        <THRESH>006</THRESH>
        <TYPE>Pre-fail</TYPE>
        <UPDATED>Always</UPDATED>
        <WHEN_FAILED>-</WHEN_FAILED>
        <RAW_VALUE>0x000000428f40</RAW_VALUE>
</ID>
<ID number="3">
        <ATTRIBUTE_NAME>Spin_Up_Time</ATTRIBUTE_NAME>
        <FLAG>0x0003</FLAG>
```

```
        <VALUE>100</VALUE>
        <WORST>100</WORST>
        <THRESH>000</THRESH>
        <TYPE>Pre-fail</TYPE>
        <UPDATED>Always</UPDATED>
        <WHEN_FAILED>-</WHEN_FAILED>
        <RAW_VALUE>0x000000000000</RAW_VALUE>
</ID>
</xml>
```

2.16.4 SemaEApiDiskSMART

```
uint32_t
EAPI_CALLTYPE
SemaEApiDiskSMART (
__IN uint32_t    BoardHandler, /* handler for remote target */
__IN uint32_t    DiskIndex, /* index of Disk, start from 0*/
__IN uint32_t    ID,    /* S.M.A.R.T. id */
__OUT uint32_t*    pNormValue,    /* S.M.A.R.T. normalized value*/
__OUT char*    pRawValue    /* S.M.A.R.T. raw value*/
);
```

Description

Get the specified S.M.A.R.T. information of assigned disk. The ID number is mapped to S.M.A.R.T. ID. For more details, see http://en.wikipedia.org/wiki/S.M.A.R.T.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	DiskIndex	Index of Disk, start from 0
__IN	ID	S.M.A.R.T. ID
__OUT	NorValue	S.M.A.R.T. normalized value
__OUT	pRawValue	S.M.A.R.T. raw value in string (0x000000000000)

2.17 1-Wire

1-wire is a device communications bus system that provides low-speed data, signaling, and power over a single signal. SEMA 1-wire function provides an easy to use interface to connect and use 1-wire capable sensors or crypto chips. SEMA BMC will act as bus master and communicate with one or more 1-wire slaves. SEMA 1-Wire bus only support standard speed (15.4Kbps).

Please note that some products that use external GPIO IC for GPIO function do not support the 1-Wire API in this session. 1-Wire support requires GPIOs which are directly controlled by BMC.

2.17.1 SemaEApi1WireReset

```
uint32_t
EAPI_CALLTYPE
SemaEApi1WireReset(
__IN uint32_t    BoardHandler, /* handler for remote target */
_);
```

Description

Reset the 1-Wire bus slave devices and get them ready for a command.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board

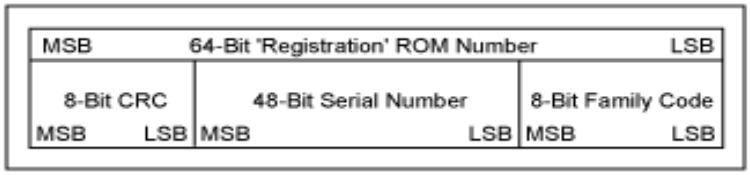
2.17.2 SemaEApi1WireProbeRomID

```
uint32_t
EAPI_CALLTYPE
SemaEApi1WireRead(
__IN uint32_t    BoardHandler, /* handler for remote target */
__OUT uint8_t *pFamilyCode , /* Family code of ROMID (1 byte)*/
__OUT uint8_t *pSerialNo, /* Serial number of ROMID(6 bytes) */
__OUT uint8_t *pCRC    /* CRC of ROMID (1 byte) */
);
```

Description

Each 1-Wire device will have 64-bit unique registration number in read-only memory (ROM) that is used to address it individually by a 1-Wire master in a 1-Wire bus.

The content of 64-bit



"SemaEApi1WireProbeRomID" provide the search function to probe the available RomID of devices on the 1-Wire bus. Once a probe is completed, the available device address is returned. Call SemaEApi1WireProbeRomID again will probe for next available RomID. To read out all RomID on the 1-Wire bus, customer should call this function repeatedly until the same RomID shows. If this function return error, customer can use SemaEApi1WireGetStatus to check the fail reason.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__OUT	pFamilyCode	Family code of ROMID (1 byte)
__OUT	pSerialNo	Serial number of ROMID (6 bytes)
__OUT	pCRC	CRC of ROMID (1 byte)

2.17.3 SemaEApi1WireGetStatus

```
uint32_t
EAPI_CALLTYPE
SemaEApi1ReadTransferID (
__IN uint32_t    BoardHandler, /* handler for remote target */
__OUT uint32_t * pStatus      /*1-Wire Operation Status */
);
```

Status

ID	Description
SEMA_EAPI_ONEWIRE_BUS_STATUS_BUSY	Bus is busy
SEMA_EAPI_ONEWIRE_BUS_STATUS_NO_SLAVE	No device on the bus
SEMA_EAPI_ONEWIRE_BUS_STATUS_STUCK	Bus stuck
SEMA_EAPI_ONEWIRE_ONEWIRE_NO_DEVICE_FOUND	No device found

Description

Read the bus status

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__OUT	pStatus	1-Wire Operation Status

2.17.4 SemaEApi1WireSendFunctionCmd

```
uint32_t
EAPI_CALLTYPE
SemaEApi1WireSendFunctionCmd (
__IN uint32_t    BoardHandler, /* Handler for remote target */
__IN uint8_t    FamilyCode , /* Family code of ROMID */
__IN uint8_t    pSerialNo, /* Serial number of ROMID */
__IN uint8_t    CRC,    /* CRC of ROMID */
__IN uint8_t *pComs , /* Array to store the commands */
__IN uint32_t CmdLen    /* Number of commands */
__IN uint8_t *pData , /* Array of data to be written */
__IN uint32_t DataLen    /* Length of data */
);
```

Description

Sent the function commands to 1-Wire device that match the ROM ID

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	FamilyCode	Family code of ROMID (1 byte)
__IN	pSerialNo	Serial number of ROMID (6 bytes)
__IN	CRC	CRC of ROMID (1 byte)
__IN	RomId	Device ROM ID
__IN	pComs	Array to store the commands
__IN	Cmd_num	Number of commands
__IN	pData	Pointer of data array to be written
__IN	DataLen	Number of bytes to be written

2.17.5 SemaEApi1WireRead

```
uint32_t
EAPI_CALLTYPE
SemaEApi1WireRead(
__IN uint32_t    BoardHandler, /* handler for remote target */
__OUT uint8_t *pBuffer , /* Pointer to data buffer */
__IN uint32_t BufLen , /* Data pBuffer Size in * bytes */
__IN uint32_t ByteCnt /* Number of bytes to read */
);
```

Description

Read out the data from 1-Wire device

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__OUT	pBuffer	Pointer to data buffer
__IN	BufLen	Data pBuffer Size in * bytes
__IN	ByteCnt	Number of bytes to read

2.17.6 SemaEApi1WireSelectGPIOPin

```
uint32_t
EAPI_CALLTYPE
SemaEApi1WireSelectGPIOPin(
__IN uint32_t    BoardHandler, /* handler for remote target */
__IN uint32_t GPIONum /* GPIO pin number for 1-Wire bus */
_);
```

Description

SEMA only support one GPIO pin as 1-Wire bus. Customer can select one of GPIO Pin from BMC to be act as 1-Wire bus. Note:

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	GPIONum	GPIO pin number for 1-Wire bus. GPIO pin number (0... NUMBER_OF_USER_GPIO-1). Set 0xFFFF as

In/Out	Parameter Name	Description
		no GPIO pin for 1-Wire function

2.17.7 SemaEApi1WireGetSelectGPIOPin

```
uint32_t
EAPI_CALLTYPE
SemaEApi1WireGetSelectGPIOPin(
__IN uint32_t    BoardHandler, /* handler for remote target */
__OUT uint32_t *pGPIONum /* GPIO pin number for 1-Wire bus */
_);
```

Description

Get the selected GPIO Pin for 1-Wire bus.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__OUT	pGPIONum	GPIO pin number for 1-Wire bus. GPIO pin number (0... NUMBER_OF_USER_GPIO-1) If return 0xFFFFF, no GPIO pin has been selected as 1-Wire.

2.18 A/D Conversion

Some SEMA devices can support basic A/D conversion functionality. Prerequisite is that the TI Tiva BMC is in use and that it is controlling directly the GPIO pins. Please refer to the BMC capabilities and to the SEMA commands for reading GPIO pin capability and for GPIO pin configuration (e.g. SemaEApiGPIOPinGetCap, SemaEApiGPIOPinSetConfig, SemaEApiGPIOPinGetConfig).

Up to 4 GPIO pins can be used for A/D conversion. Voltage range supported is 0..3.3 V. Resolution of the A/D converter output is 10-bit.

Note: physical connection to respective pins still require to add at least an external own simple filter, e.g. resistor-capacity set-up, or other suitable electric protection to protect input and to limit voltages to allowed range!

A/D conversion measurements can be performed using the SEMA command SemaEApiGPIOPinGetADCValue.

2.18.1 SemaEApiGPIOPinGetADCValue

```
uint32_t
SemaEApiGPIOPinGetADCValue (
__IN uint32_t BoardHandler, /* handler for remote target */
__IN uint8_t Id , /* GPIO Id */
__OUT uint32_t *Value /* ADC value */
);
```

Description

Read the A/D converter (ADC) value of GPIO pins for which the selected GPIO capability is ADC type.

Parameters

In/Out	Parameter Name	Description
__IN	BoardHandler	The handler of selected board
__IN	Id	GPIO ID
__OUT	Value	A/D conversion (ADC) value of PIN The resolution of the ADC inputs is 10-bit. The sampled voltage is referenced to 3.3V. Based on these facts, the voltage is calculated as follow: Vin = [ADC value] / 1023 * 3.3V (The number 1023 is the resolution of the ADC.)

3 Status Codes

EAPI_STATUS_NOT_INITIALIZED

Description

The EAPI library is not yet or unsuccessfully initialized. ***SemaEApiLibInitialize*** needs to be called prior to the first access of any other EAPI function.

Actions

Call ***SemaEApiLibInitialize***.

EAPI_STATUS_INITIALIZED

Description

Library is initialized.

Actions

None.

EAPI_STATUS_ALLOC_ERROR

Description

Memory Allocation Error.

Actions

Free memory and try again.

EAPI_STATUS_SW_TIMEOUT

Description

Software time out. This is normally caused by hardware/software semaphore timeout.

Actions

Retry.

EAPI_STATUS_INVALID_PARAMETER

Description

One or more of the EAPI function call parameters are out of the defined range.

Actions

Verify Function Parameters.

EAPI_STATUS_INVALID_BLOCK_LENGTH

Description

This means that the block length is too long.

Actions

Use relevant capabilities information to correct select block lengths.

EAPI_STATUS_INVALID_BLOCK_ALIGNMENT

Description

The block alignment is incorrect.

Actions

Use alignment capabilities information to correctly align write access.

EAPI_STATUS_INVALID_DIRECTION

Description

The current direction argument attempts to set GPIOs to an unsupported directions, i.e. Setting GPI to Output.

Actions

Use pInputs and pOutputs to correctly select input and outputs.

EAPI_STATUS_INVALID_BITMASK

Description

The bitmask selects bits/GPIOs which are not supported for the current ID.

Actions

Use pInputs and pOutputs to probe supported bits.

EAPI_STATUS_INVALID _LOCAL_TIME

Description

Invalid local time..

Actions

None.

EAPI_STATUS_UNSUPPORTED

Description

This function or ID is not supported at the actual hardware environment.

Actions

None.

EAPI_STATUS_NOT_FOUND

Description

Selected device was not found.

Example

The I^2C device address is not Acknowledged, device is not present or inactive.

Actions

None.

EAPI_STATUS_BUSY_COLLISION

Description

The selected device or ID is busy or a data collision was detected.

Example

The addressed I^2C bus is busy or there is a bus collision.

The I^2C bus is in use. Either CLK or DAT are low.

Arbitration loss or bus Collision, data remains low when writing a 1.

Actions

Retry.

EAPI_STATUS_RUNNING

Description

Watchdog timer already started.

Actions

Call EApiWDogStop, before retrying.

EAPI_STATUS_HW_TIMEOUT

Description

Function call timed out.

Example

I^2C operation lasted too long.

Actions

Retry.

EAPI_STATUS_READ_ERROR

Description

An error was detected during a read operation.

Example

I2C Read function was not successful

Actions

Retry.

EAPI_STATUS_WRITE_ERROR

Description

An error was detected during a write operation.

Example

I^2C 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.

10 bit address device not present.

Storage write error.

Actions

Retry.

EAPI_STATUS_MORE_DATA

Description

The amount of available data exceeds the buffer size.

Storage buffer overflow was prevented. Read count was larger than the defined buffer length.

Actions

Either increase the buffer size or reduce the block length.

EAPI_STATUS_ERROR

Description

Generic error message. No further error details are available.

Actions

None.

EAPI_STATUS_SUCCESS

The value for this status code is defined as 0.

Description

The operation was successful.

Actions

None.

Getting Service

Ask an Expert: http://askanexpert.adlinktech.com

ADLINK Technology, Inc.

9F, No.166 Jian Yi Road, Zhonghe District

New Taipei City 235, Taiwan

Tel: +886-2-8226-5877

Fax: +886-2-8226-5717

E-mail: service@adlinktech.com

Ampro ADLINK Technology, Inc.

5215 Hellyer Avenue, #110

San Jose, CA 95138, USA

Tel: +1-408-360-0200

Toll Free: +1-800-966-5200 (USA only)

Fax: +1-408-360-0222

E-mail: info@adlinktech.com

ADLINK Technology (China) Co., Ltd.

300 Fang Chun Rd., Zhangjiang Hi-Tech Park

Pudong New Area, Shanghai, 201203 China

Tel: +86-21-5132-8988

Fax: +86-21-5132-3588

E-mail: market@adlinktech.com

ADLINK Technology GmbH

Hans-Thoma-Strasse 11

D-68163 Mannheim, Germany

Tel: +49-621-43214-0

Fax: +49-621 43214-30

E-mail: emea@adlinktech.com

Please visit the Contact page at www.adlinktech.com for information on how to contact the nearest ADLINK regional office.