Implementation Guide

AAEON Windows EAPI (Embedded Application Programming Interface)

Revision 2.0 : January 15, 2020

Lists AAEON Windows EAPI (Embedded Application Programming Interface) for customers to develop their applications.

Contents

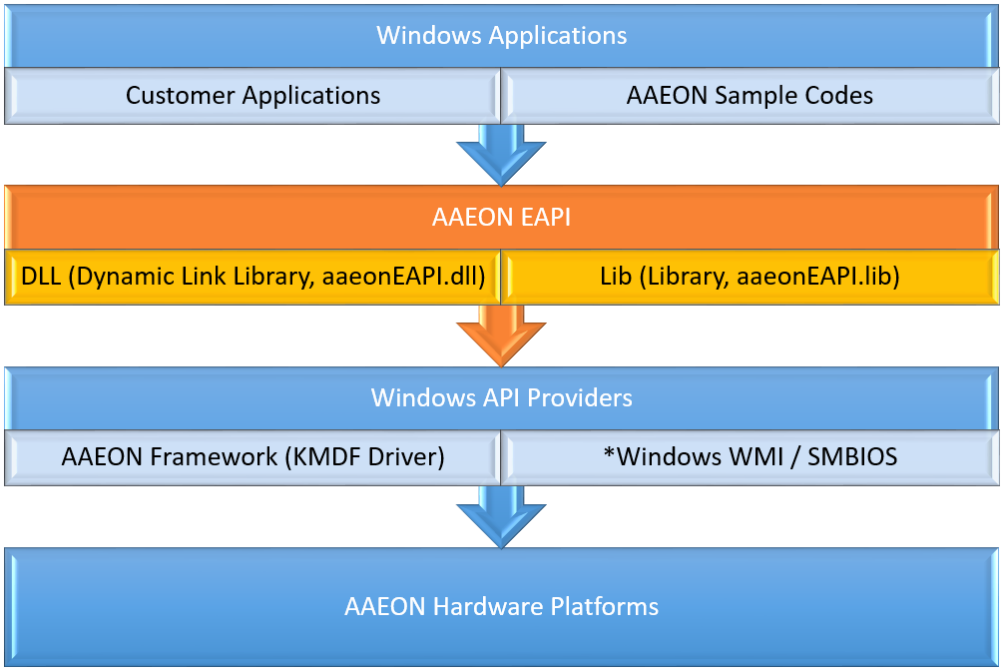
Revision History

Revision	Date	Description
2.0	January 15, 2020	New revised and updated version

Section 1: Overview

AAEON Windows Team provides AAEON Framework (KMDF Driver), DLL (Dynamic Link Library, aaeonEAPI.dll) / Lib (Library, aaeonEAPI.lib) and SDK (Software Development Kit) with sample codes for customers to develop their applications for Windows 7, 8, 8.1 and 10.

This document lists AAEON Windows EAPI (Embedded Application Programming Interface) for customers to develop their applications. The following diagram shows how these components work:



*Windows WMI / SMBIOS help get System Information from Windows.

Section 2: Definitions

This section lists AAEON Windows EAPI (Embedded Application Programming Interface) code definitions to help customers understand AAEON Windows Sample Codes more easily.

2.1: Parameter Prefix

2.1.1 __IN

__IN	
Arg Type	Characteristics
Immediate value	Input value that must be specified and is essential to function operation.
Pointer	Valid pointer to initialized buffer/variable.

2.1.2 __INOPT

__INOPT	
Arg Type	Characteristics
Pointer	Valid pointer to initialized buffer/variable, or NULL Pointer. Note: refer to function specification for specifics.

2.1.3 __OUT

__OUT	
Arg Type	Characteristics
Pointer	Valid pointer to destination buffer/variable.

2.1.4 __OUTOPT

__OUTOPT	
Arg Type	Characteristics
Pointer	Valid pointer to destination buffer/variable, or NULL Pointer. Note: refer to function specification for specifics.

2.1.5 __INOUT

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

2.2 Returned Status Codes

2.2.1 EAPI_STATUS_NOT_INITIALIZED

EAPI_STATUS_NOT_INITIALIZED	
Description	The EAPI library is not yet or unsuccessfully initialized. EApiLibInitialize needs to be called prior to the first access of any EAPI function. Actions: Call EApiLibInitialize.
Value	0xFFFFFFFF

2.2.2 EAPI_STATUS_INITIALIZED

EAPI_STATUS_INITIALIZED	
Description	Library is initialized. Actions: None.
Value	0xFFFFFFFE

2.2.3 EAPI_STATUS_ALLOC_ERROR

EAPI_STATUS_ALLOC_ERROR	
Description	Memory Allocation Error. Actions: Free memory and try again.
Value	0xFFFFFFFD

2.2.4 EAPI_STATUS_DRIVER_TIMEOUT

EAPI_STATUS_DRIVER_TIMEOUT	
Description	Time out in driver. This is Normally caused by hardware/software semaphore timeout. Actions: Retry.
Value	0xFFFFFFFC

2.2.5 EAPI_STATUS_DEVICE_NOT_READY

EAPI_STATUS_DRIVER_TIMEOUT	
Description	Hardware is not ready. Actions: Check BIOS setting or HW jumper settings whichever is applicable.
Value	0xFFFFFFFB

2.2.6 EAPI_STATUS_INVALID_PARAMETER

EAPI_STATUS_INVALID_PARAMETER	
Description	One or more of the EAPI function call parameters are out of range. Possible Reasons could be NULL Pointer Invalid Offset Invalid Length Undefined Value Storage Write Incorrectly Aligned Offset Invalid Write Length Actions: Verify Function Parameters.
Value	0xFFFFFEFF

2.2.7 EAPI_STATUS_INVALID_BLOCK_ALIGNMENT

EAPI_STATUS_INVALID_BLOCK_ALIGNMENT	
Description	The Block Alignment is incorrect. Actions: Use pInputs and pOutputs to correctly select input and outputs
Value	0xFFFFFEFE

2.2.8 EAPI_STATUS_INVALID_BLOCK_LENGTH

EAPI_STATUS_INVALID_BLOCK_LENGTH	
Description	Block length is too long. Actions: Use Alignment Capabilities information to correctly align write access.
Value	0xFFFFFEFD

2.2.9 EAPI_STATUS_INVALID_DIRECTION

EAPI_STATUS_INVALID_DIRECTION	
Description	The current Direction Argument attempts to set GPIOs to an unsupported directions. I.E. Setting GPIO Input to Output. Actions: Use pInputs and pOutputs to correctly select input and outputs.
Value	0xFFFFFEFC

2.2.10 EAPI_STATUS_INVALID_BITMASK

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.
Value	0xFFFFFEFB

2.2.11 API_STATUS_RUNNING

API_STATUS_RUNNING	
Description	Watchdog timer already started. Actions: Call EApiWDogStop, before retrying.
Value	0xFFFFFEFA

2.2.12 EAPI_STATUS_UNSUPPORTED

EAPI_STATUS_UNSUPPORTED	
Description	This function or ID is not supported in this hardware configuration. Actions: None.
Value	0xFFFFFCFF

2.2.13 EAPI_STATUS_NOT_FOUND

EAPI_STATUS_NOT_FOUND	
Description	I2C Device Error No Acknowledge for Device Address, 7 Bit Address Only. 10 Bit Address may cause Write error if two 10 Bit addressed devices present on the bus. Actions: None.
Value	0xFFFFFBFF

2.2.14 EAPI_STATUS_TIMEOUT

EAPI_STATUS_TIMEOUT	
Description	EApi I2C functions specific. 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 Actions: Retry.
Value	0xFFFFFBFD

2.2.15 EAPI_STATUS_READ_ERROR

EAPI_STATUS_READ_ERROR	
Description	I2C Read Error Not Possible to detect. Storage Read Error .... Actions: Retry.
Value	0xFFFFFAFF

2.2.16 EAPI_STATUS_WRITE_ERROR

EAPI_STATUS_WRITE_ERROR	
Description	I2C Write Error No Acknowledge received after writing any Byte after the First Address Byte. Can be caused by unsupported Device Command/Index; Ext Command/Index used on Standard Command/Index Device; 10 Bit Address Device Not Present; Storage Write Error; ... Actions: Retry.
Value	0xFFFFFAFE

2.2.17 EAPI_STATUS_MORE_DATA

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. Read Count > Buffer Length. Actions: Either increase the buffer size or reduce the block length.
Value	0xFFFFF9FF

2.2.18 EAPI_STATUS_ERROR

EAPI_STATUS_ERROR	
Description	Generic error message. No further error details are available. Actions: None.
Value	0xFFFFF0FF

2.2.19 EAPI_STATUS_SUCCESS

EAPI_STATUS_SUCCESS	
Description	Successful operation Actions: None.
Value	0x00000000

2.3 Alias Names

2.3.1 EAPI_CALLTYPE

```
# define EAPI_CALLTYPE __declspec(dllexport)
```

2.3.2 uint8_t

```
typedef unsigned __int8   uint8_t;
```

2.3.3 uint16_t

```
typedef unsigned __int16 uint16_t;
```

2.3.4 uint32_t

```
typedef unsigned __int32 uint32_t;
```

2.3.5 EAPI_UINT8_C(x)

```
# define EAPI_UINT8_C(x)      ((uint8_t)(x))
```

2.3.6 EAPI_UINT16_C(x)

```
# define EAPI_UINT16_C(x)     ((uint16_t)(x))
```

2.3.7 EAPI_UINT32_C(x)

```
# define EAPI_UINT32_C(x)     ((uint32_t)(x))
```

2.3.8 EApiStatus_t

typedef uint32_t EApiStatus_t;

2.3.9 EApiId_t

typedef uint32_t EApiId_t;

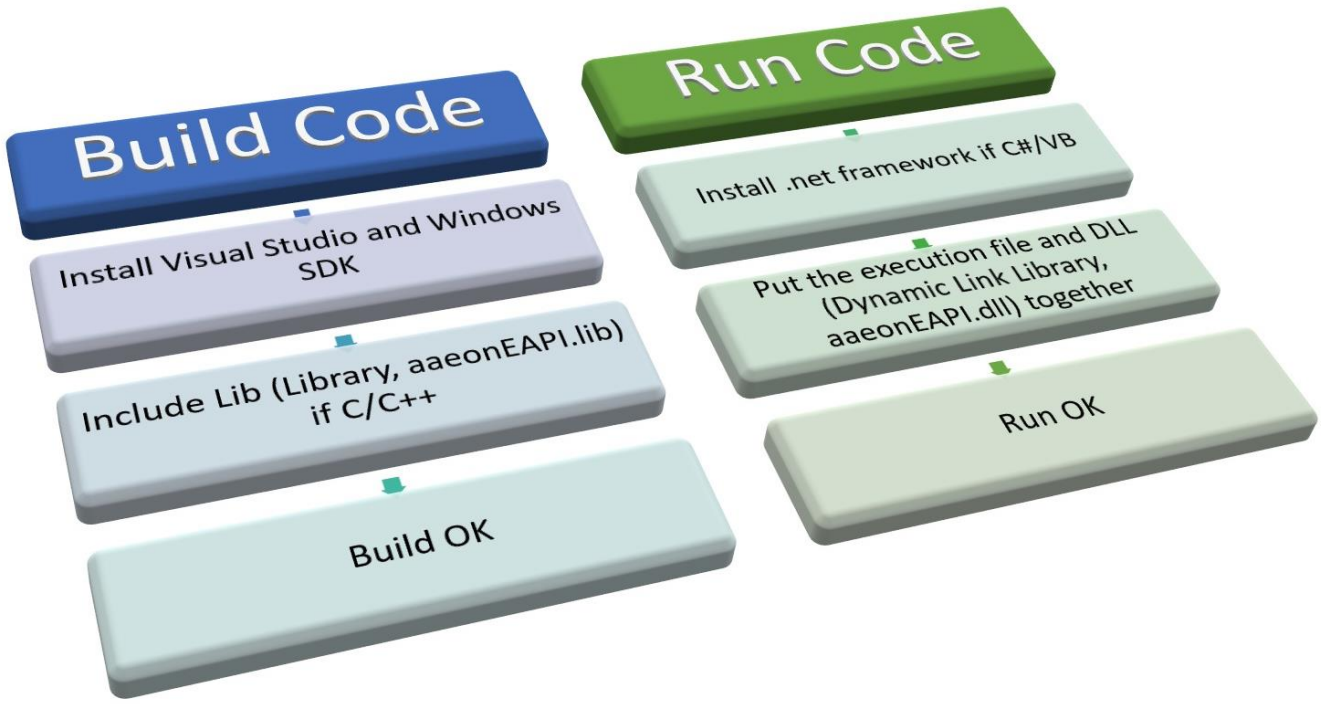
Section 3: Function Description

This section lists AAEON Windows EAPI (Embedded Application Programming Interface) supported functions.

AAEON Framework (KMDF Driver) must be installed before calling these functions. *EapiLibInitialize()* should be the first to call before calling other EAPI functions. *EApiLibUnInitialize()* should be called to release resources before program exit.

When building C/C++ apps, Lib (Library, aaeonEAPI.lib) is needed.
aaeonEAPI.lib is needed for C/C++ based app, please put the lib files and executable files in same folder.

The following shows how to build and run codes:



3.1 Initialization Functions

Prior to calling any EAPI function these functions must be called to initialize the AAEON library.
The status code for all EAPI functions will be EAPI_STATUS_NOT_INITIALIZED unless this initialization function is called.
Before program exit, un-initialization functions must be called to release resources.

3.1.1 *EApiLibInitialize(void)*

EApiLibInitialize(void)	
Description	Should be called before calling any other API function.
Parameter(s)	None
Condition	**Return Values**
Library Already initialized	EAPI_STATUS_INITIALIZED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.1.2 *EApiVGALibInitialize(void)*

EApiVGALibInitialize(void)	
Description	Should be called before calling any other VIDEO backlight API function.
Parameter(s)	None
Condition	**Return Values**
Library Already initialized	EAPI_STATUS_INITIALIZED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.1.3 *EApiLibUnInitialize(void)*

EApiLibUnInitialize(void)	
Description	Should be called before program exit.
Parameter(s)	None
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.1.4 *EApiVGALibUnInitialize(void)*

EApiVGALibUnInitialize(void)	
Description	Should be called before VIDEO backlight program exit.
Parameter(s)	None
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.2 Information Functions

These functions return information provided by AAEON Windows EAPI.

3.2.1 *EapiBoardGetStringA()*

EApiBoardGetStringA(*__IN EApiId_t Id,* */* Name Id */* *__OUT char *pBuffer,* */* Destination pBuffer */* *__INOUT uint32_t *pBufLen* */* pBuffer Length */* *);*	
Description	Get information about the AAEON hardware platform in string format.
Parameter(s)	Id: EAPI_ID_BOARD_NAME_STR /* Board Name String */ EAPI_ID_BOARD_SERIAL_STR /* Board ID */ EAPI_ID_BOARD_MANUFACTURER_STR /* Board Manufacturer Name String */ EAPI_ID_BOARD_BIOS_REVISION_STR /* Board Bios Revision String */ EAPI_ID_EC_REVISION_STR /* EC Revision */ EAPI_ID_BASEBOARD_SERIAL_STR /* SMBIOS Baseboard Serial Number */ *pBuffer: Destination Buffer *pBufLen: Buffer Length
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
pBufLen==NULL	EAPI_STATUS_INVALID_PARAMETER
pBufLen!=NULL && *pBufLen && pBuffer==NULL	EAPI_STATUS_INVALID_PARAMETER
Unknown Id	EAPI_STATUS_UNSUPPORTED
strlen(Id)+1 > *pBufLen	EAPI_STATUS_MORE_DATA
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

Note:

1. EC Revision is only applicable when platform has an EC chip.
2. Baseboard serial number is only applicable when platform has a ComExpress module and a baseboard.

3.2.2 *EapiBoardGetValue()*

EApiBoardGetValue(*__IN EApiId_t Id,* */* Value Id */* *__OUT uint32_t *pValue* */* Return Value */* *);*	
Description	Get information about the AAEON hardware platform in value format.
Parameter(s)	Id: EAPI_ID_HWMON_FAN_CPU /* CPU Fan Speed (RPM) */ EAPI_ID_HWMON_FAN_CHIPSET /* Chipset Fan Speed (RPM) */ EAPI_ID_HWMON_FAN_SYSTEM /* System Fan Speed (RPM) */ EAPI_ID_HWMON_CPU_TEMP /* CPU Temperature */ EAPI_ID_HWMON_CHIPSET_TEMP /* Chipset Temperature */ EAPI_ID_HWMON_SYSTEM_TEMP /* System Temperature */ 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_VOLTAGE_DIMM /* DIMM/RAM Voltage (millivolts) */ EAPI_ID_HWMON_VOLTAGE_3VSB /* 3V Standby Voltage (millivolts) */ EAPI_ID_BOARD_DRIVER_VERSION_VAL /* AAEON Framework Driver Version */ *pValue: Return Value
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
pValue==NULL	EAPI_STATUS_INVALID_PARAMETER
Unknown Id	EAPI_STATUS_UNSUPPORTED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

Note: not all HWMON information are available to all platform. If you have question, please consult

with AAEON support team.

3.3 Backlight Functions

These functions control backlight for integrated flat panel displays, typically LVDS. Please note, these functions are only applicable to the platforms support LCD backlight driving.

3.3.1 *EApiBKLIGHTGetCaps()*

EApiBKLIGHTGetCaps(***__OUT uint32_t *pBLCDevCount*** ***)***	
Description	Get the count of backlight control
Parameter(s)	*pBLCDevCount: Number of backlight control
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.3.2 *EapiVgaGetBacklightEnable()*

EApiVgaGetBacklightEnable(***__IN EApiId_t Id, /* Backlight Id */*** ***__OUT uint32_t *pEnable /* Backlight Enable */*** ***)***	
Description	Get current Backlight Enable state for specified Flat Panel.
Parameter(s)	Id: EAPI_ID_BACKLIGHT_1 /* Backlight Panel 1 (LVDS1) */ EAPI_ID_BACKLIGHT_2 /* Backlight Panel 2 (LVDS2)*/ EAPI_ID_BACKLIGHT_3 /* Backlight Panel 3 (Reserved)*/ EAPI_ID_BACKLIGHT_4 /* Backlight Panel 4 (eDP)*/ *pEnable: FALSE (0) = Off, TRUE (1) = On
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
pEnable==NULL	EAPI_STATUS_INVALID_PARAMETER
Unknown Id	EAPI_STATUS_UNSUPPORTED

Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.3.3 *EapiVgaGetBacklightBrightness()*

EApiVgaGetBacklightBrightness(***__IN EApiId_t Id,*** ***/* Backlight Id */*** ***__OUT uint32_t *pBright*** ***/* Backlight Brightness */*** ***)***	
Description	Get the brightness level of the selected flat panel display.
Parameter(s)	Id: EAPI_ID_BACKLIGHT_1 /* Backlight Panel 1 (LVDS1) */ EAPI_ID_BACKLIGHT_2 /* Backlight Panel 2 (LVDS2)*/ EAPI_ID_BACKLIGHT_3 /* Backlight Panel 3 (Reserved)*/ EAPI_ID_BACKLIGHT_4 /* Backlight Panel 4 (eDP)*/ *pBright: The value of brightness level, from 0 to 255
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
pBright==NULL	EAPI_STATUS_INVALID_PARAMETER
Unknown Id	EAPI_STATUS_UNSUPPORTED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.3.4 *EapiVgaSetBacklightBrightness()*

EApiVgaSetBacklightBrightness(*__IN EApiId_t Id,* */* Backlight Id */* *__IN uint32_t Bright* */* Backlight Brightness */* *)*	
Description	Set the brightness level of the selected flat panel display.
Parameter(s)	Id: EAPI_ID_BACKLIGHT_1 /* Backlight Panel 1 (LVDS1) */ EAPI_ID_BACKLIGHT_2 /* Backlight Panel 2 (LVDS2)*/ EAPI_ID_BACKLIGHT_3 /* Backlight Panel 3 (Reserved)*/ EAPI_ID_BACKLIGHT_4 /* Backlight Panel 4 (eDP)*/ Bright: The value of brightness level, from 0 to 255
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Bright > 255	EAPI_STATUS_INVALID_PARAMETER
Unknown Id	EAPI_STATUS_UNSUPPORTED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.4 I²C|SMBUS Functions

I²C/SMBUS functions support 7/10 bit standard/extended commands. AAEON Windows EAPI supports up to eight I²C controllers and three SMBUS controllers.

3.4.1 *EApiI2CWriteReadRaw()*

EApiI2CWriteReadRaw(***__IN EApiId_t Id, /* I²C|SMBUS Bus Id */*** ***__IN uint8_t Addr, /* Encoded 7Bit I²C|SMBUS Device Address */*** ***__INOPT void *pWBuffer, /* Write Data pBuffer */*** ***__IN uint32_t WriteBCnt, /* Number of Bytes to write plus 1 */*** ***__OUTOPT void *pRBuffer, /* Read Data pBuffer */*** ***__IN uint32_t RBufLen, /* Data pBuffer Length */*** ***__IN uint32_t ReadBCnt /* Number of Bytes to Read plus 1 */*** ***)***	
Description	Common function for read/write commands to the I²C/SMBUS devices.
Parameter(s)	Id: EAPI_ID_I2C_EXTERNAL /* 1st I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_2 /* 2nd I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_3 /* 3rd I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_4 /* 4th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_5 /* 5th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_6 /* 6th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_7 /* 7th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_8 /* 8th I²C bus */ EAPI_ID_AONCUS_SMBUS_EXTERNAL_1 /* 1st SMBUS bus */ EAPI_ID_AONCUS_SMBUS_EXTERNAL_2 /* 2nd SMBUS bus */ EAPI_ID_AONCUS_SMBUS_EXTERNAL_3 /* 3rd SMBUS bus */ Addr: Encoded 7/10 Bit I²C|SMBUS Device Address Cmd: I²C|SMBUS Command/Offset *pWBuffer: Write Data Buffer WriteBCnt: Number of Bytes to write plus 1 *pRBuffer: Read Data Buffer RBufLen: Data Buffer Length ReadBCnt: Number of Bytes to Read plus 1

Condition	Return Values
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
(WriteBCnt>1)&&(pWBuffer==NULL)	EAPI_STATUS_INVALID_PARAMETER
(ReadBCnt>1)&&(pRBuffer==NULL)	EAPI_STATUS_INVALID_PARAMETER
(ReadBCnt>1)&&(RBufLen==0)	EAPI_STATUS_INVALID_PARAMETER
((WriteBCnt==0)&&(ReadBCnt==0))	EAPI_STATUS_INVALID_PARAMETER
Unknown Id	EAPI_STATUS_UNSUPPORTED
WriteBCnt>(pMaxBlkLen+1)	EAPI_STATUS_INVALID_BLOCK_LENGTH
ReadBCnt>(RBufLen+1)	EAPI_STATUS_MORE_DATA
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.4.2 *EApiI2CReadTransfer()*

EApiI2CReadTransfer(*__IN EApiId_t Id, /* I²C|SMBUS Bus Id */* *__IN uint32_t Addr, /* Encoded 7/10Bit I²C|SMBUS Device Address */* *__IN uint32_t Cmd, /* I²C|SMBUS Command/Offset */* *__OUT void *pBuffer, /* Transfer Data pBuffer */* *__IN uint32_t BufLen, /* Data pBuffer Length */* *__IN uint32_t ByteCnt /* Byte Count to read */* *)*	
Description	Addr Byte 1 Below Designates Addr MSB in a 10 bit address transfer and the complete address in an 7 bit address transfer.
Parameter(s)	Id: EAPI_ID_I2C_EXTERNAL /* 1st I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_2 /* 2nd I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_3 /* 3rd I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_4 /* 4th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_5 /* 5th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_6 /* 6th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_7 /* 7th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_8 /* 8th I²C bus */ EAPI_ID_AONCUS_SMBUS_EXTERNAL_1 /* 1st SMBUS bus */ EAPI_ID_AONCUS_SMBUS_EXTERNAL_2 /* 2nd SMBUS bus */ EAPI_ID_AONCUS_SMBUS_EXTERNAL_3 /* 3rd SMBUS bus */ Addr: Encoded 7/10 Bit I²C|SMBUS Device Address Cmd: I²C|SMBUS Command/Offset *pBuffer: Transfer Data pBuffer BufLen: Data pBuffer Length, Byte=1; Word=2 ByteCnt: Byte Count to read, Byte=1; Word=2

Condition	Return Values
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
pBuffer == NULL || BufLen == 0 || ByteCnt == 0	EAPI_STATUS_INVALID_PARAMETER
ByteCnt > MAX_BLOCK_LENGTH(0x100) - Cmd	EAPI_STATUS_INVALID_BLOCK_LENGTH
ByteCnt > BufLen	EAPI_STATUS_MORE_DATA
Unknown Id	EAPI_STATUS_UNSUPPORTED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.4.3 *EApiI2CWriteTransfer()*

EApiI2CWriteTransfer(*__IN EApiId_t Id, /* I²C|SMBUS Bus Id */* *__IN uint32_t Addr, /* Encoded 7/10Bit I²C|SMBUS Device Address */* *__IN uint32_t Cmd, /* I²C|SMBUS Command/Offset */* *__IN void *pBuffer, /* Transfer Data pBuffer */* *__IN uint32_t ByteCnt /* Byte Count to write */* *)*	
Description	Addr Byte 1 Below Designates Addr MSB in a 10 bit address transfer and the complete address in an 7 bit address transfer.
Parameter(s)	Id: EAPI_ID_I2C_EXTERNAL /* 1st I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_2 /* 2nd I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_3 /* 3rd I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_4 /* 4th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_5 /* 5th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_6 /* 6th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_7 /* 7th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_8 /* 8th I²C bus */ EAPI_ID_AONCUS_SMBUS_EXTERNAL_1 /* 1st SMBUS bus */ EAPI_ID_AONCUS_SMBUS_EXTERNAL_2 /* 2nd SMBUS bus */ EAPI_ID_AONCUS_SMBUS_EXTERNAL_3 /* 3rd SMBUS bus */ Addr: Encoded 7/10 Bit I²C|SMBUS Device Address Cmd: I²C|SMBUS Command/Offset *pBuffer: Transfer Data pBuffer ByteCnt: Byte Count to read, Byte=1; Word=2

Condition	Return Values
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
pBuffer == NULL || ByteCnt == 0	EAPI_STATUS_INVALID_PARAMETER
ByteCnt > MAX_BLOCK_LENGTH(0x100) - Cmd	EAPI_STATUS_INVALID_BLOCK_LENGTH
Unknown Id	EAPI_STATUS_UNSUPPORTED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.4.4 *EApiI2CProbeDevice()*

EApiI2CProbeDevice(***__IN EApiId_t Id,*** ***/* I²C|SMBUS Bus Id */*** ***__IN uint32_t Addr*** ***/* Encoded 7/10Bit I²C|SMBUS Device Address */*** ***)***	
Description	I²C Probe Types Probe Type 1: Address Format : 7Bit Start <Addr Byte> <W> Ack Stop Probe Type 2: Address Format : 10Bit Start <Addr Byte MSB> <W> Ack <Addr Byte LSB> Ack Stop
Parameter(s)	Id: EAPI_ID_I2C_EXTERNAL /* 1st I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_2 /* 2nd I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_3 /* 3rd I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_4 /* 4th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_5 /* 5th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_6 /* 6th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_7 /* 7th I²C bus */ EAPI_ID_AONCUS_I2C_EXTERNAL_8 /* 8th I²C bus */ EAPI_ID_AONCUS_SMBUS_EXTERNAL_1 /* 1st SMBUS bus */ EAPI_ID_AONCUS_SMBUS_EXTERNAL_2 /* 2nd SMBUS bus */ EAPI_ID_AONCUS_SMBUS_EXTERNAL_3 /* 3rd SMBUS bus */ Addr: Encoded 7/10 Bit I²C|SMBUS Device Address

Condition	Return Values
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unknown Id	EAPI_STATUS_UNSUPPORTED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.5 WATCHDOG Functions

Two scenarios to invoke WDT functions

Use EApiWDogStart

After EApiWDogStart

```
|<- Delay ->|<- Event Timeout ->|<- Reset Timeout ->|
A-----------B-------------------C-------------------D
```

Use EApiWDogTrigger

After EApiWDogTrigger

```
|<- Event Timeout ->|<- Reset Timeout ->|
E-------------------F-------------------G
```

Stage A

Watchdog is started.

Stage B

Initial Delay Period.

Stage C/F

Event is triggered, NMI, IRQ, or PIN is Triggered.

This allows for possible Software Recovery.

Stage D/G

System is reset.

Stage E

Watchdog is Triggered.

EApiWDogStop must be called before Stage C/F to prevent event from being generated.

EApiWDogStop must be called before Stage D/G to prevent system from being reset.

3.5.1 *EapiWDogGetCap()*

```
EApiWDogGetCap(
    __OUTOPT uint32_t *pMaxDelay,           /* Maximum Supported Delay in milliseconds*/
    __OUTOPT uint32_t *pMaxEventTimeout, /* Maximum Supported Event Timeout in
                                            milliseconds
                                            0 == Unsupported
                                         */
    __OUTOPT uint32_t *pMaxResetTimeout /* Maximum Supported Reset Timeout in
                                            milliseconds
                                         */
)
```

Description	Get maximum Supported Delay / Supported Event Timeout / Supported Reset Timeout of the watchdog timer.
Parameter(s)	*pMaxDelay: Maximum Supported Delay in milliseconds *pMaxEventTimeout: Maximum Supported Event Timeout in milliseconds 0 == Unsupported *pMaxResetTimeout: Maximum Supported Reset Timeout in milliseconds

Condition	Return Values
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
pMaxDelay == NULL && pMaxResetTimeout == NULL && pMaxEventTimeout == NULL	EAPI_STATUS_INVALID_PARAMETER
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.5.2 *EapiWDogStart()*

EApiWDogStart(*__IN uint32_t Delay, /* Delay in milliseconds */* *__IN uint32_t Minute, /* Control Minute or Second */* *__IN uint32_t EventTimeout, /* Event Timeout in milliseconds */* *__IN uint32_t ResetTimeout /* Reset Timeout in milliseconds */* *)*	
Description	Start the watchdog timer and set the timeout values, watchdog must be stopped via EApiWDogStop and then EApiWDogStart must be called again with the new values. If the hardware implementation of the watchdog timer does not allow to set exactly the select time, the EAPI shall select the next available and longer time.
Parameter(s)	Delay: Delay in milliseconds Minute: Control Minute or Second EventTimeout: Event Timeout in milliseconds ResetTimeout: Reset Timeout in milliseconds
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
(Delay > gMaxDelay) || (EventTimeout > gMaxEventTimeout) || (ResetTimeout > gMaxResetTimeout)	EAPI_STATUS_INVALID_PARAMETER
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.5.3 *EapiWDogTrigger()*

EapiWDogTrigger(void)	
Description	Trigger the watchdog timer.
Parameter(s)	None
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Watchdog Not Started	EAPI_STATUS_ERROR
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.5.4 *EapiWDogStop()*

EApiWDogStop(void)	
Description	Close Watchdog Instance
Parameter(s)	None
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.5.5 *EapiWDogReloadTimer()*

EApiWDogReloadTimer(void)	
Description	Reload the Timeout count
Parameter(s)	None
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.5.6 *EapiWDogGetStatus()*

EApiWDogGetStatus(*__OUTOPT uint32_t *pwdtMinute,* *__OUTOPT uint32_t *pwdtCountTime,* *__OUTOPT uint32_t *pwdtReloadTime* *)*	
Description	Get watchdog timer mode, time count value and reload timer.
Parameter(s)	pwdtMinute: Get the mode of minute or second *pwdtCountTime: Get WDT time count *pwdtReloadTime: Get WDT ReloadTime
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.5.7 *EapiWDogSetStatus()*

EApiWDogSetStatus(*__IN uint32_t wdtMinute,* *__IN uint32_t wdtCountTime,* *__IN uint32_t wdtReloadTime* *)*	
Description	Set watchdog timer mode, time count value and reload timer.
Parameter(s)	wdtMinute: Set the mode of minute or second wdtCountTime: Set WDT time count wdtReloadTime: Set WDT ReloadTime
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.6 GPIO (DIO) Functions

AAEON hardware platforms assign pins for general purpose I/Os. AAEON Windows EAPI provides a set of functions to control these hardware GPIO pins. For actual hardware pin mapping, please consult user manual or contact AAEON support team.

3.6.1 *EApiGPIOGetDirectionCaps()*

EApiGPIOGetDirectionCaps(***__IN EApiId_t Id,*** ***/* GPIO Id */*** ***__OUTOPT uint32_t *pInputs,*** ***/* Supported GPIO Input Bit Mask */*** ***__OUTOPT uint32_t *pOutputs*** ***/* Supported GPIO Output Bit Mask */*** ***)***	
Description	Get the capabilities of GPIO.
Parameter(s)	Id: Individual ID Per GPIO Mapping – EAPI_ID_GPIO_GPIO00: 'GPIO 0' EAPI_ID_GPIO_GPIO01: 'GPIO 1' EAPI_ID_GPIO_GPIO02: 'GPIO 2' EAPI_ID_GPIO_GPIO03: 'GPIO 3' EAPI_ID_GPIO_GPIO04: 'GPIO 4' EAPI_ID_GPIO_GPIO05: 'GPIO 5' EAPI_ID_GPIO_GPIO06: 'GPIO 6' EAPI_ID_GPIO_GPIO07: 'GPIO 7' Multiple GPIOs Per ID Mapping – EAPI_ID_GPIO_BANK00: GPIO 0-31 of bank 0 EAPI_ID_GPIO_BANK01: GPIO 0-31 of bank 1 EAPI_ID_GPIO_BANK02: GPIO 0-31 of bank 2 *pInputs: Pointer to a buffer that receives the bit mask of the supported inputs. *pOutputs: Pointer to a buffer that receives the bit mask of the supported outputs

Condition	Return Values
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
(pOutputs==NULL)&&(pInputs==NULL)	EAPI_STATUS_INVALID_PARAMETER
Unsupported ID	EAPI_STATUS_UNSUPPORTED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.6.2 *EApiGPIOGetDirection()*

EApiGPIOGetDirection(*__IN EApiId_t Id, /* GPIO Id */* *__IN uint32_t Bitmask, /* Bit mask of Affected Bits */* *__OUT uint32_t *pDirection /* Current Direction */* *)*	
Description	Get the direction of the selected GPIO(s).
Parameter(s)	Id: Individual ID Per GPIO Mapping – EAPI_ID_GPIO_GPIO00: 'GPIO 0' EAPI_ID_GPIO_GPIO01: 'GPIO 1' EAPI_ID_GPIO_GPIO02: 'GPIO 2' EAPI_ID_GPIO_GPIO03: 'GPIO 3' EAPI_ID_GPIO_GPIO04: 'GPIO 4' EAPI_ID_GPIO_GPIO05: 'GPIO 5' EAPI_ID_GPIO_GPIO06: 'GPIO 6' EAPI_ID_GPIO_GPIO07: 'GPIO 7' Multiple GPIOs Per ID Mapping – EAPI_ID_GPIO_BANK00: GPIO 0-31 of bank 0 EAPI_ID_GPIO_BANK01: GPIO 0-31 of bank 1 EAPI_ID_GPIO_BANK02: GPIO 0-31 of bank 2 Bitmask: Bit mask of Affected Bits: 0xFFFFFFFF *pDirection: Current Direction

Condition	Return Values
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Bitmask == 0	EAPI_STATUS_INVALID_PARAMETER
Unsupported ID	EAPI_STATUS_UNSUPPORTED
GetLastError() == 0x15	API_STATUS_DEVICE_NOT_READY
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.6.3 *EApiGPIOSetDirection()*

EApiGPIOSetDirection(*__IN EApiId_t Id, /* GPIO Id */* *__IN uint32_t Bitmask, /* Bit mask of Affected Bits */* *__IN uint32_t Direction /* Direction */* *)*	
Description	Set the direction of the selected GPIO(s).
Parameter(s)	Id: Individual ID Per GPIO Mapping – EAPI_ID_GPIO_GPIO00: 'GPIO 0' EAPI_ID_GPIO_GPIO01: 'GPIO 1' EAPI_ID_GPIO_GPIO02: 'GPIO 2' EAPI_ID_GPIO_GPIO03: 'GPIO 3' EAPI_ID_GPIO_GPIO04: 'GPIO 4' EAPI_ID_GPIO_GPIO05: 'GPIO 5' EAPI_ID_GPIO_GPIO06: 'GPIO 6' EAPI_ID_GPIO_GPIO07: 'GPIO 7' Multiple GPIOs Per ID Mapping – EAPI_ID_GPIO_BANK00: GPIO 0-31 of bank 0 EAPI_ID_GPIO_BANK01: GPIO 0-31 of bank 1 EAPI_ID_GPIO_BANK02: GPIO 0-31 of bank 2 Bitmask: Bit mask of Affected Bits: 0xFFFFFFFF Direction: Direction

Condition	Return Values
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Bitmask == 0	EAPI_STATUS_INVALID_PARAMETER
Unsupported ID	EAPI_STATUS_UNSUPPORTED
GetLastError() == 0x15	EAPI_STATUS_DEVICE_NOT_READY
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.6.4 *EApiGPIOGetLevel()*

EApiGPIOGetLevel(*__IN EApiId_t Id, /* GPIO Id */* *__IN uint32_t Bitmask, /* Bit mask of Affected Bits */* *__OUT uint32_t *pLevel /* Current Level */* *)*	
Description	Get level value from GPIO(s).
Parameter(s)	Id: Individual ID Per GPIO Mapping – EAPI_ID_GPIO_GPIO00: 'GPIO 0' EAPI_ID_GPIO_GPIO01: 'GPIO 1' EAPI_ID_GPIO_GPIO02: 'GPIO 2' EAPI_ID_GPIO_GPIO03: 'GPIO 3' EAPI_ID_GPIO_GPIO04: 'GPIO 4' EAPI_ID_GPIO_GPIO05: 'GPIO 5' EAPI_ID_GPIO_GPIO06: 'GPIO 6' EAPI_ID_GPIO_GPIO07: 'GPIO 7' Multiple GPIOs Per ID Mapping – EAPI_ID_GPIO_BANK00: GPIO 0-31 of bank 0 EAPI_ID_GPIO_BANK01: GPIO 0-31 of bank 1 EAPI_ID_GPIO_BANK02: GPIO 0-31 of bank 2 Bitmask: Bit mask of Affected Bits: 0xFFFFFFFF *pLevel: Current Level
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Bitmask == 0	EAPI_STATUS_INVALID_PARAMETER
Unsupported ID	EAPI_STATUS_UNSUPPORTED
GetLastError() == 0x15	EAPI_STATUS_DEVICE_NOT_READY
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.6.5 *EApiGPIOSetLevel()*

EApiGPIOSetLevel(*__IN EApiId_t Id, /* GPIO Id */* *__IN uint32_t Bitmask, /* Bit mask of Affected Bits */* *__IN uint32_t Level /* Level */* *)*	
Description	Set level value of GPIO(s).
Parameter(s)	Id: Individual ID Per GPIO Mapping – EAPI_ID_GPIO_GPIO00: 'GPIO 0' EAPI_ID_GPIO_GPIO01: 'GPIO 1' EAPI_ID_GPIO_GPIO02: 'GPIO 2' EAPI_ID_GPIO_GPIO03: 'GPIO 3' EAPI_ID_GPIO_GPIO04: 'GPIO 4' EAPI_ID_GPIO_GPIO05: 'GPIO 5' EAPI_ID_GPIO_GPIO06: 'GPIO 6' EAPI_ID_GPIO_GPIO07: 'GPIO 7' Multiple GPIOs Per ID Mapping – EAPI_ID_GPIO_BANK00: GPIO 0-31 of bank 0 EAPI_ID_GPIO_BANK01: GPIO 0-31 of bank 1 EAPI_ID_GPIO_BANK02: GPIO 0-31 of bank 2 Bitmask: Bit mask of Affected Bits: 0xFFFFFFFF Level: Level

Condition	Return Values
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Bitmask == 0	EAPI_STATUS_INVALID_PARAMETER
Unsupported ID	EAPI_STATUS_UNSUPPORTED
GetLastError() == 0x15	EAPI_STATUS_DEVICE_NOT_READY
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.6.6 *EApiGPIOGetCaps()*

EApiGPIOGetCaps(*__IN EApiId_t Id,* */* GPIO Id */* *__OUTOPT uint32_t *PinCount,* */* Supported GPIO number */* *__OUTOPT uint32_t *pDioDisable* */* GPIO active or not */* *)*	
Description	Get GPIO input/output mapping bits.
Parameter(s)	Id: Individual ID Per GPIO Mapping – EAPI_ID_GPIO_GPIO00: 'GPIO 0' EAPI_ID_GPIO_GPIO01: 'GPIO 1' EAPI_ID_GPIO_GPIO02: 'GPIO 2' EAPI_ID_GPIO_GPIO03: 'GPIO 3' EAPI_ID_GPIO_GPIO04: 'GPIO 4' EAPI_ID_GPIO_GPIO05: 'GPIO 5' EAPI_ID_GPIO_GPIO06: 'GPIO 6' EAPI_ID_GPIO_GPIO07: 'GPIO 7' Multiple GPIOs Per ID Mapping – EAPI_ID_GPIO_BANK00: GPIO 0-31 of bank 0 EAPI_ID_GPIO_BANK01: GPIO 0-31 of bank 1 EAPI_ID_GPIO_BANK02: GPIO 0-31 of bank 2 *PinCount: Supported GPIO number *pDioDisable: GPIO active or not, 1: Inactive; 0: Active

Condition	Return Values
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
(PinCount == NULL) && (pDioDisable == NULL)	EAPI_STATUS_INVALID_PARAMETER
Unsupported ID	EAPI_STATUS_UNSUPPORTED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.7 Fan Control Functions

AAEON Windows EAPI provides a set of functions to control fan rotation. Customer can control fan speed manually or automatically [smart fan, depends on temperature(s)].

3.7.1 *EapiSfanGetStatus()*

EApiSfanGetStatus(***__IN EApiId_t Id, /* Fan Id */*** ***__OUT uint32_t *pFanAutoMode,*** ***__OUT uint32_t *pFullSpeedTemp,*** ***__OUT uint32_t *pLowSpeedTemp,*** ***__OUT uint32_t *pManualSpeed*** ***)***	
Description	Get the specified fan current configuration, which includes fan speed control mode, full/low fan speed temperature trigger point and manual set fan speed value.
Parameter(s)	Id: Fan Id EAPI_ID_SFAN00 /* 1st Fan */ EAPI_ID_SFAN01 /* 2nd Fan */ EAPI_ID_SFAN02 /* 3rd Fan */ EAPI_ID_SFAN03 /* 4th Fan */ EAPI_ID_SFAN04 /* 5th Fan */ EAPI_ID_SFAN05 /* 6th Fan */ *pFanAutoMode: Mode for Fan 1: Auto Mode; 0: Manual Mode *pFullSpeedTemp: Fan speed is full speed when exceeds the setting temperature 0-100 *pLowSpeedTemp: Fan speed is low speed when less than the setting temperature 0-100 *pManualSpeed: Manual Speed 0-255
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.7.2 *EapiSfanSetStatus()*

EApiSfanSetStatus(*__IN EApiId_t Id, /* Fan Id */* *__IN uint32_t FanAutoMode,* *__IN uint32_t FullSpeedTemp,* *__IN uint32_t LowSpeedTemp,* *__IN uint32_t ManualSpeed* *)*	
Description	Set the specified fan configuration, which includes fan speed control mode, full/low fan speed temperature trigger point and manual set fan speed value.
Parameter(s)	Id: Fan Id EAPI_ID_SFAN00 /* 1st Fan */ EAPI_ID_SFAN01 /* 2nd Fan */ EAPI_ID_SFAN02 /* 3rd Fan */ EAPI_ID_SFAN03 /* 4th Fan */ EAPI_ID_SFAN04 /* 5th Fan */ EAPI_ID_SFAN05 /* 6th Fan */ FanAutoMode: Mode for Fan 1: Auto Mode; 0: Manual Mode FullSpeedTemp: Fan speed is full speed when exceeds the setting temperature 0-100 LowSpeedTemp: Fan speed is low speed when less than the setting temperature 0-100 ManualSpeed: Manual Speed 0-255
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.7.3 *EapiSfanGetCaps()*

EApiSfanGetCaps(*__IN EApiId_t Id, /* Fan Id */* *__OUT uint32_t *pAutoModeCap* *)*	
Description	Check if the specified fan supports auto mode or not.
Parameter(s)	Id: Fan Id EAPI_ID_SFAN00 /* 1st Fan */ EAPI_ID_SFAN01 /* 2nd Fan */ EAPI_ID_SFAN02 /* 3rd Fan */ EAPI_ID_SFAN03 /* 4th Fan */ EAPI_ID_SFAN04 /* 5th Fan */ EAPI_ID_SFAN05 /* 6th Fan */ *pAutoModeCap: Auto mode supported or not 1 – Yes 0 – No
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.8 Hardware Monitor Functions

Hardware Monitor Functions provide Fan/Voltage/Temperature values for system health.

3.8.1 *EApiHWMONGetCaps()*

EApiHWMONGetCaps(***__IN EApiId_t Id, /* HWMON Id */*** ***__OUT uint32_t *pFanEnable,*** ***__OUT uint32_t *pTempEnable,*** ***__OUT uint32_t *pVoltEnable*** ***)***	
Description	Check if the specified monitor item active or not.
Parameter(s)	Id: Hardware Monitor Id EAPI_ID_HWMON_FAN_CPU /* CPU Fan */ EAPI_ID_HWMON_FAN_CHIPSET /* Chipset Fan */ EAPI_ID_HWMON_FAN_SYSTEM /* System Fan */ EAPI_ID_HWMON_CPU_TEMP /* CPU Temperature */ EAPI_ID_HWMON_CHIPSET_TEMP /* Chipset Temperature */ EAPI_ID_HWMON_SYSTEM_TEMP /* System Temperature */ EAPI_ID_HWMON_VOLTAGE_VCORE /* CPU Core Voltage */ EAPI_ID_HWMON_VOLTAGE_2V5 /* 2.5V Voltage */ EAPI_ID_HWMON_VOLTAGE_3V3 /* 3.3V Voltage */ EAPI_ID_HWMON_VOLTAGE_VBAT /* Battery Voltage */ EAPI_ID_HWMON_VOLTAGE_5V /* 5V Voltage */ EAPI_ID_HWMON_VOLTAGE_5VSB /* 5V Standby Voltage */ EAPI_ID_HWMON_VOLTAGE_12V /* 12V Voltage */ EAPI_ID_HWMON_VOLTAGE_DIMM /* DIMM/RAM Voltage */ EAPI_ID_HWMON_VOLTAGE_3VSB /* 3V Standby Voltage */ *pFanEnable: Fan active or not, 1 = active; 0 = inactive *pTempEnable: Temperature active or not, 1 = active; 0 = inactive *pVoltEnable: Voltage active or not, 1 = active; 0 = inactive
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unknown Id	EAPI_STATUS_UNSUPPORTED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.8.2 *EapiBoardGetValue()*

Please reference 3.2.2 to get Hardware Monitor values.

3.9 PWM Functions

PWM Functions provide PWM value Get/Set interfaces.

3.9.1 *EapiPwmGetValue()*

EApiPwmGetValue(***__IN EApiId_t Id,*** ***/* PWM Id */*** ***__OUT uint32_t *pPWMBaseUnitInt,*** ***/* PWM Base Unit Integer */*** ***__OUT uint32_t *pPWMBaseUnitFrac,*** ***/* PWM Base Unit Fractional */*** ***__OUT uint32_t *pPWMDutyCycle*** ***/* PWM Base Duty Cycle */*** ***)***	
Description	Get the integer/fractional portions of PWM Base Unit and the duty cycle.
Parameter(s)	Id: PWM Id EAPI_ID_PWM_1 1st PWM EAPI_ID_PWM_2 2nd PWM EAPI_ID_PWM_3 3rd PWM *pPWMBaseUnitInt: Get the integer portion of Base Unit *pPWMBaseUnitFrac: Get the fractional portion of Base Unit *pPWMDutyCycle: Get the duty cycle
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unknown Id	EAPI_STATUS_UNSUPPORTED
pPWMBaseUnitInt == NULL || pPWMBaseUnitFrac == NULL || pPWMDutyCycle == NULL	EAPI_STATUS_INVALID_PARAMETER
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

3.9.2 *EapiPwmSetValue()*

EApiPwmSetValue(*__IN EApiId_t Id, /* PWM Id */* *__IN uint32_t PWMBaseUnitInt, /* PWM Base Unit Integer */* *__IN uint32_t PWMBaseUnitFrac, /* PWM Base Unit Fractional */* *__IN uint32_t PWMDutyCycle /* PWM Base Unit Duty Cycle */* *)*	
Description	Set the integer/fractional portions of PWM Base Unit and the duty cycle.
Parameter(s)	Id: PWM Id EAPI_ID_PWM_1 1st PWM EAPI_ID_PWM_2 2nd PWM EAPI_ID_PWM_3 3rd PWM PWMBaseUnitInt: Set the integer portion of Base Unit PWMBaseUnitFrac: Set the fractional portion of Base Unit PWMDutyCycle: Set the duty cycle
Condition	**Return Values**
Library Uninitialized	EAPI_STATUS_NOT_INITIALIZED
Unknown Id	EAPI_STATUS_UNSUPPORTED
Common Error	Common Error Code
Others	EAPI_STATUS_SUCCESS

AAEON Company Profile

AAEON Corporate Video