aaeonEAPI User Guide

Document History

Version	Date	Author	Description
1.00		Tim Lin	First Release
1.06	2017.10.26	Tim Lin	Modify some typo
1.07	2017.11.10	Jerry Liao	Adjust some format incorrect
1.08	2018.04.11	Albert Wu	Add parameter for EApiWDogStart()
1.09	2018.05.14	Jerry Liao	Change to EApiSfanGetCaps
1.10	2018.01.25	Albert Wu	Add eDP backlight control function

Function Documentation

Initialize/UnInitialize API

EApiStatus_t EAPI_CALLTYPE EApiLibInitialize (void)

Should be called before calling any other API function is called.

EApiStatus_t EAPI_CALLTYPE EApiLibUnInitialize (void)

Should be called before program exit.

EApiStatus_t EAPI_CALLTYPE EApiVGALibInitialize (void)

Should be called before calling any other video backlight API function is called.

EApiStatus_t EAPI_CALLTYPE EApiVGALibUnInitialize (void)

Should be called before video backlight program exit.

Board Information API

EApiStatus_t EAPI_CALLTYPE EApiBoardGetStringA (__IN EApiId_t *Id*, __OUT char * *pBuffer*, __INOUT uint32_t * *pBufLen*)

Parameters:

Id	Name Id: EAPI_ID_BOARD_NAME_STR
Id	Name Id: EAPI_ID_BOARD_SERIAL_STR
Id	Name Id: EAPI_ID_BOARD_MANUFACTURER_STR
Id	Name Id: EAPI_ID_BOARD_BIOS_REVISION_STR
Id	Name Id: EAPI_ID_EC_REVISION_STR

Return values:

**pBuffer*	Destination pBuffer

Parameters:

**pBufLen*	pBuffer Length

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_INVALID_PARAMETER:

pBufLen==NULL/pBufLen!=NULL&&*pBufLen&&pBuffer==NULL

EAPI_STATUS_UNSUPPORTED: unknown Id

EAPI_STATUS_MORE_DATA: strlen(Id)+1>*pBufLen

EAPI_STATUS_SUCCESS: Else

Hardware Monitor API

Hardware Monitor SDK is API for customer to use Hardware Monitor feature on AAEON products. Features include Temperature, Voltage, and Fan speed information.

The API dll file is aaeonEAPI.dll

The API lib file is aaeonEAPI.lib

The API header file is aaeonEAPI.h

EApiStatus_t EAPI_CALLTYPE EApiHWMONGetCaps (__IN EApiId_t Id, __OUT uint32_t * *pFanEnable*, __OUT uint32_t * *pTempEnable*, __OUT uint32_t * *pVoltEnable*)

Parameters:

Id	Value Id: EAPI_ID_HWMON_FAN_CPU
Id	Value Id: EAPI_ID_HWMON_FAN_CHIPSET
Id	Value Id: EAPI_ID_HWMON_FAN_SYSTEM
Id	Value Id: EAPI_ID_HWMON_CPU_TEMP
Id	Value Id: EAPI_ID_HWMON_CHIPSET_TEMP
Id	Value Id: EAPI_ID_HWMON_SYSTEM_TEMP
Id	Value Id: EAPI_ID_HWMON_VOLTAGE_VCORE
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_2V5
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_3V3
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_VBAT
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_5V
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_5VSB
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_12V
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_DIMM
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_3VSB
Id	Value Id: EAPI_ID_BOARD_DRIVER_VERSION_VAL
Id	Value Id: EAPI_ID_AONCUS_HISAFE_FUCTION

Return values:

**pFanEnable*	FanEnable 1:Enable; 0:Disable
**pTempEnable*	TempEnable 1:Enable; 0:Disable
**pVoltEnable*	VoltEnable 1:Enable; 0:Disable

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_UNSUPPORTED: unknown Id

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiBoardGetValue (__IN EApiId_t *Id*, __OUT uint32_t * *pValue*)

Parameters:

Id	Value Id: EAPI_ID_HWMON_FAN_CPU
Id	Value Id: EAPI_ID_HWMON_FAN_CHIPSET
Id	Value Id: EAPI_ID_HWMON_FAN_SYSTEM
Id	Value Id: EAPI_ID_HWMON_CPU_TEMP
Id	Value Id: EAPI_ID_HWMON_CHIPSET_TEMP
Id	Value Id: EAPI_ID_HWMON_SYSTEM_TEMP
Id	Value Id: EAPI_ID_HWMON_VOLTAGE_VCORE
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_2V5
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_3V3
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_VBAT
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_5V
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_5VSB
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_12V
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_DIMM
Id	Value Id:EAPI_ID_HWMON_VOLTAGE_3VSB
Id	Value Id: EAPI_ID_BOARD_DRIVER_VERSION_VAL
Id	Value Id: EAPI_ID_AONCUS_HISAFE_FUCTION

Return values:

**pValue*	Return Value

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_INVALID_PARAMETER: pValue==NULL

EAPI_STATUS_UNSUPPORTED: unknown Id

EAPI_STATUS_SUCCESS: Else

Digital IO API

DIO SDK is API for customer to use DIO feature on AAEON products. This SDK will get DIO port on the current AAEON product.

The API dll file is aaeonEAPI.dll

The API lib file is aaeonEAPI.lib

The API header file is aaeonEAPI.h

EApiStatus_t EAPI_CALLTYPE EApiGPIOGetCaps (__IN EApiId_t *Id*, __OUTOPT uint32_t * *PinCount*, __OUTOPT uint32_t * *pDioDisable*)

Parameters:

Id	GPIO Id: **EAPI_GPIO_GPIO_ID(GPIO_NUM)**

Return values:

**PinCount*	Pin Count
**pDioDisable*	Pin Enable or Disable 1:Disable; 0:Enable

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_INVALID_PARAMETER: Bitmask==0

EAPI_STATUS_UNSUPPORTED: Unsupported Id

EAPI_STATUS_INVALID_BITMASK: (Bitmask&~(pInputs|pOutputs))

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiGPIOGetDirection (__IN EApiId_t *Id*, __IN uint32_t *Bitmask*, __OUT uint32_t * *pDirection*)

Parameters:

Id	GPIO Id: **EAPI_GPIO_GPIO_ID(GPIO_NUM)**
Bitmask	Bit mask of Affected Bits: 0xFFFFFFFF

Return values:

**pDirection*	Current Direction

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_INVALID_PARAMETER: Bitmask==0

EAPI_STATUS_UNSUPPORTED: Unsupported Id

EAPI_STATUS_INVALID_BITMASK: (Bitmask&~(pInputs|pOutputs))

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiGPIOGetLevel (__IN EApiId_t *Id*, __IN uint32_t *Bitmask*, __OUT uint32_t * *pLevel*)

Parameters:

Id	GPIO Id: **EAPI_GPIO_GPIO_ID(GPIO_NUM)**
Bitmask	Bit mask of Affected Bits: 0xFFFFFFFF

Return values:

**pLevel*	Current Level

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_INVALID_PARAMETER: Bitmask==0

EAPI_STATUS_UNSUPPORTED: Unsupported Id

EAPI_STATUS_INVALID_BITMASK: (Bitmask&~(pInputs|pOutputs))

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiGPIOSetDirection (__IN EApiId_t *Id*, __IN uint32_t *Bitmask*, __IN uint32_t *Direction*)

Parameters:

Id	GPIO Id: **EAPI_GPIO_GPIO_ID(GPIO_NUM)**
Bitmask	Bit mask of Affected Bits: 0xFFFFFFFF
Direction	Direction

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_INVALID_PARAMETER: Bitmask==0

EAPI_STATUS_UNSUPPORTED: Unsupported Id

EAPI_STATUS_INVALID_BITMASK: (Bitmask&~(pInputs|pOutputs))

EAPI_STATUS_INVALID_DIRECTION:

(Bitmask&pDirection)&~pInputs/(Bitmask&~pDirection)&~pOutputs

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiGPIOSetLevel (__IN EApiId_t *Id*, __IN uint32_t *Bitmask*, __IN uint32_t *Level*)

Parameters:

Id	GPIO Id: **EAPI_GPIO_GPIO_ID(GPIO_NUM)**
Bitmask	Bit mask of Affected Bits: 0xFFFFFFFF
Level	Level

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_INVALID_PARAMETER: Bitmask==0

EAPI_STATUS_UNSUPPORTED: Unsupported Id

EAPI_STATUS_SUCCESS: Else

SMBUS/I2C API

Smbus SDK is supported to control the smbus/I2C function, it can read or write data for device connected to smbus/I2C. Through the API interface, the customer can control the smbus function easily.

The API dll file is aaeonEAPI.dll

The API lib file is aaeonEAPI.dll

The API header file is aaeonEAPI.dll

EApiStatus_t EAPI_CALLTYPE EApiI2CWriteReadRaw (__IN EApiId_t *Id*, __IN uint8_t *Addr*, __INOPT void* *pWBuffer*, __IN uint32_t *WriteBCnt*, __OUTOPT void * *pRBuffer*, __IN uint32_t *RBufLen*, __IN uint32_t *ReadBCnt*)

```
#define EApiI2CWriteRaw(Id, Addr, pBuffer, ByteCnt) \
        EApiI2CWriteReadRaw(Id, Addr, pBuffer, ByteCnt, NULL, 0, 0)
#define EApiI2CReadRaw(Id, Addr, pBuffer, BufLen, ByteCnt) \
        EApiI2CWriteReadRaw(Id, Addr, NULL, 0, pBuffer, BufLen, ByteCnt)
```

Parameters:

Id	EAPI_ID_I2C_EXTERNAL
Id	EAPI_ID_I2C_LVDS_1
Id	EAPI_ID_I2C_LVDS_2
Id	EAPI_ID_AONCUS_I2C_EXTERNAL_2
Id	EAPI_ID_AONCUS_I2C_EXTERNAL_3
Id	EAPI_ID_AONCUS_SMBUS_EXTERNAL_1
Id	EAPI_ID_AONCUS_SMBUS_EXTERNAL_2
Id	EAPI_ID_AONCUS_SMBUS_EXTERNAL_3
Addr	Encoded 7/10Bit I2C Device Address
**pWBuffer*	Transfer Write Data Buffer
WriteBCnt	Data Length
***pRBuffer*	Transfer Read Data Buffer
BufLen	Data pBuffer Length: Byte=1;
ReadBCnt	Data Length

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_INVALID_PARAMETER: pBuffer==NULL/BufLen==0/ByteCnt==0

EAPI_STATUS_UNSUPPORTED: unknown Id

EAPI_STATUS_INVALID_BLOCK_LENGTH: ByteCnt>pMaxBlkLen

EAPI_STATUS_BUSY_COLLISION: Bus Busy SDA/SDC low/Arbitration Error/Collision Error

EAPI_STATUS_TIMEOUT: Time-out due to clock stretching

EAPI_STATUS_NOT_FOUND: start<Addr Byte 1><R>Nak

EAPI_STATUS_SUCCESS: Else

The following macro is used to define the "Cmd" whether **byte** or **word**:

```
#define EAPI_I2C_ENC_STD_CMD(x)     EAPI_UINT32_C(((x)&0xFF)|EAPI_I2C_STD_CMD)
#define EAPI_I2C_ENC_EXT_CMD(x) EAPI_UINT32_C(((x)&0xFFFF)|EAPI_I2C_EXT_CMD)
```

EApiStatus_t EAPI_CALLTYPE EApiI2CReadTransfer (__IN EApiId_t *Id*, __IN uint32_t *Addr*, __IN uint32_t *Cmd*, __OUT void * *pBuffer*, __IN uint32_t *BufLen*, __IN uint32_t *ByteCnt*)

Parameters:

Id	EAPI_ID_I2C_EXTERNAL
Id	EAPI_ID_I2C_LVDS_1
Id	EAPI_ID_I2C_LVDS_2
Id	EAPI_ID_AONCUS_I2C_EXTERNAL_2
Id	EAPI_ID_AONCUS_I2C_EXTERNAL_3
Id	EAPI_ID_AONCUS_SMBUS_EXTERNAL_1
Id	EAPI_ID_AONCUS_SMBUS_EXTERNAL_2
Id	EAPI_ID_AONCUS_SMBUS_EXTERNAL_3
Addr	Encoded 7/10Bit I2C Device Address
Cmd	I2C Command/Offset

Return values:

**pBuffer*	Transfer Data pBuffer

Parameters:

BufLen	Data pBuffer Length: Byte=1; Word=2
ByteCnt	Byte Count to read: Byte=1; Word=2

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_INVALID_PARAMETER: pBuffer==NULL/BufLen==0/ByteCnt==0

EAPI_STATUS_UNSUPPORTED: unknown Id

EAPI_STATUS_INVALID_BLOCK_LENGTH: ByteCnt>pMaxBlkLen

EAPI_STATUS_BUSY_COLLISION: Bus Busy SDA/SDC low/Arbitration Error/Collision Error

EAPI_STATUS_TIMEOUT: Time-out due to clock stretching

EAPI_STATUS_NOT_FOUND: start<Addr Byte 1><R>Nak

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiI2CWriteTransfer (__IN EApiId_t *Id*, __IN uint32_t *Addr*, __IN uint32_t *Cmd*, __IN void * *pBuffer*, __IN uint32_t *ByteCnt*)

Parameters:

Id	EAPI_ID_I2C_EXTERNAL
Id	EAPI_ID_I2C_LVDS_1
Id	EAPI_ID_I2C_LVDS_2
Id	EAPI_ID_AONCUS_I2C_EXTERNAL_2
Id	EAPI_ID_AONCUS_I2C_EXTERNAL_3
Id	EAPI_ID_AONCUS_SMBUS_EXTERNAL_1
Id	EAPI_ID_AONCUS_SMBUS_EXTERNAL_2
Id	EAPI_ID_AONCUS_SMBUS_EXTERNAL_3
Addr	Encoded 7/10Bit I2C Device Address
Cmd	I2C Command/Offset; Seq&Page Mode-->ByteCnt
**pBuffer*	Transfer Data pBuffer
ByteCnt	Byte Count to write: Byte=1; Word=2

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_INVALID_PARAMETER: pBuffer==NULL/BufLen==0/ByteCnt==0

EAPI_STATUS_UNSUPPORTED: unknown Id

EAPI_STATUS_INVALID_BLOCK_LENGTH: ByteCnt+(overhead)>pMaxBlkLen

EAPI_STATUS_BUSY_COLLISION: Bus Busy SDA/SDC low/Arbitration Error/Collision Error

EAPI_STATUS_TIMEOUT: Time-out due to clock stretching

EAPI_STATUS_NOT_FOUND: start<Addr Byte 1><W>Nak

EAPI_STATUS_WRITE_ERROR: start<Addr Byte 1><W>Ack<Addr Byte 2>Nak/<CMD Byte 1>Nak/<Data Byte 1>Nak

EAPI_STATUS_SUCCESS: Else

Smart Fan API

Smart fan SDK is supported to control the fan function; it can control fan speed manually.

The API dll file is aaeonEAPI.dll

The API lib file is aaeonEAPI.dll

The API header file is aaeonEAPI.dll

EApiStatus_t EAPI_CALLTYPE EApiSfanGetCaps (__IN EApiId_t *Id*, __OUT uint32_t * *pAutoModeCap*)

Parameters:

Id	Sfan Id: EAPI_ID_SFAN00
Id	Sfan Id: EAPI_ID_SFAN01
Id	Sfan Id: EAPI_ID_SFAN02

Return values:

**pAutoModeCap*	Auto mode enable 1:Enable; 0:Disable

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiSfanGetStatus (__IN EApiId_t *Id*, __OUT uint32_t * *pFanAutoMode*, __OUT uint32_t * *pFullSpeedTemp*, __OUT uint32_t * *pLowSpeedTemp*, __OUT uint32_t * *pManualSpeed*)

Parameters:

Id	Sfan Id: EAPI_ID_SFAN00
Id	Sfan Id: EAPI_ID_SFAN01
Id	Sfan Id: EAPI_ID_SFAN02

Return values:

**pFanAutoMode*	Mode for Fan 1:AutoMode; 0: ManualMode
**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*	ManualSpeed 0-255

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_UNSUPPORTED: Unsupported Id

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiSfanSetStatus (__IN EApiId_t *Id*, __IN uint32_t *FanAutoMode*, __IN uint32_t *FullSpeedTemp*, __IN uint32_t *LowSpeedTemp*, __IN uint32_t *ManualSpeed*)

Parameters:

Id	Sfan Id: EAPI_ID_SFAN00
Id	Sfan Id: EAPI_ID_SFAN01
Id	Sfan Id: EAPI_ID_SFAN02
FanAutoMode	Mode for Fan 1:AutoMode; 0: ManualMode
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	ManualSpeed 0-255

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_UNSUPPORTED: Unsupported Id

EAPI_STATUS_SUCCESS: Else

BACKLIGHT API

BACKLIGHT Controller SDK is API to control the brightness of LVDS in **Voltage** and **PWM** mode. Through the API interface, the customer can determine the brightness of LVDS LCD panel easily.

The API dll file is aaeonEAPI.dll

The API lib file is aaeonEAPI.dll

The API header file is aaeonEAPI.dll

EApiStatus_t EAPI_CALLTYPE EApiBKLIGHTGetCaps (__OUT uint32_t * *pBLCDevCount*)

Return values:

**pBLCDevCount*	Get the count of backlight control

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiVgaGetBacklightBrightness (__IN EApiId_t *Id*, __OUT uint32_t * *pBright*)

Parameters:

Id	Backlight Id: EAPI_ID_BACKLIGHT_1
Id	Backlight Id: EAPI_ID_BACKLIGHT_2
Id	Backlight Id: EAPI_ID_BACKLIGHT_3
Id	Backlight Id: EAPI_ID_BACKLIGHT_4

Return values:

**pBright*	Get the value of brightness, the value is from 0 to 255

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_INVALID_PARAMETER: pBright==NULL

EAPI_STATUS_UNSUPPORTED: unknown Id

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiVgaSetBacklightBrightness (__IN EApiId_t *Id*, __IN uint32_t *Bright*)

Parameters:

Id	Backlight Id: EAPI_ID_BACKLIGHT_1
Id	Backlight Id: EAPI_ID_BACKLIGHT_2
Id	Backlight Id: EAPI_ID_BACKLIGHT_3
Id	Backlight Id: EAPI_ID_BACKLIGHT_4
Bright	Set the value of brightness, the value is from 0 to 255

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_INVALID_PARAMETER: Bright>EAPI_BACKLIGHT_SET_BRIGHTEST

EAPI_STATUS_UNSUPPORTED: unknown Id

EAPI_STATUS_SUCCESS: Else

Watchdog Timer API

Watchdog Timer SDK is API for customer to use Watchdog feature on AAEON products. This Watchdog SDK is used when system requires auto-rebooting after system hangs up without updating the enabled Watchdog timer embedded in AAEON's board.

The API dll file is aaeonEAPI.dll

The API lib file is aaeonEAPI.dll

The API header file is aaeonEAPI.dll

EApiStatus_t EAPI_CALLTYPE EApiWDogGetCap (__OUTOPT uint32_t * *pMaxDelay*, __OUTOPT uint32_t * *pMaxEventTimeout*, __OUTOPT uint32_t * *pMaxResetTimeout*)

Return values:

**pMaxDelay*	Maximum Supported Delay in milliseconds
**pMaxEventTimeout*	Maximum Supported Event Timeout in milliseconds 0 == Unsupported
**pMaxResetTimeout*	Maximum Supported Reset Timeout in milliseconds

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_UNSUPPORTED: Unsupported

EAPI_STATUS_INVALID_PARAMETER: pMaxDelay==NULL && pMaxResetTimeout==NULL && pMaxEventTimeout==NULL

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiWDogGetStatus (__OUTOPT uint32_t * *pwdtMinute*, __OUTOPT uint32_t * *pwdtCountTime*, __OUTOPT uint32_t * *pwdtReloadTime*)

Return values:

**pwdtMinute*	Get the mode of minute or second
**pwdtCountTime*	Get WDT time count
**pwdtReloadTime*	Get WDT ReloadTime

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiWDogSetStatus (__IN uint32_t *wdtMinute*, __IN uint32_t *wdtCountTime*, __IN uint32_t *wdtReloadTime*)

Parameters:

wdtMinute	Set the mode of minute or second
wdtCountTime	Set WDT time count
wdtReloadTime	Set WDT ReloadTime

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiWDogReloadTimer (void)

Reload the Timeout count.

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_UNSUPPORTED: Unsupported

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiWDogStart (__IN uint32_t *Delay*, __IN uint32_t Minute, __IN uint32_t *EventTimeout*, __IN uint32_t *ResetTimeout*)

Parameters:

Delay	Delay in milliseconds
Minute	Control Minute or Second
EventTimeout	Event Timeout in milliseconds
ResetTimeout	Reset Timeout in milliseconds

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_UNSUPPORTED: Unsupported

EAPI_STATUS_INVALID_PARAMETER: Delay>pMaxDelay/EventTimeout>pMaxEventTimeout/ResetTimeout>pMaxResetTimeout

EAPI_STATUS_RUNNING: Already Running

EAPI_STATUS_SUCCESS: Else

EApiStatus_t EAPI_CALLTYPE EApiWDogStop (void)

Close Watchdog Instance.

Returns:

EAPI_STATUS_NOT_INITIALIZED: Library Uninitialized

EAPI_STATUS_UNSUPPORTED: Unsupported

EAPI_STATUS_SUCCESS: Else