ASUSTek Computer Inc.

ASUS API Programming Guide

Manual Rev.: 1.02

Revision Date: 2022/02/23

Revision History

Revision	Date	Change
1.00	2021/10/08	Initial release
1.01	2021/11/15	Add Tinker Board 2/2S, Tinker Edge R support
1.02	2022/02/23	Add APIs of Vehicle Co-Processors. Add APIs of AsusBoardGet/SetComPortMode Add APIs of AsusBoardGet/SetModeOfMiniPCIeSlot Add APIs of AsusBoardGet/SelectSIMSlot Add PE100A support on Debian 10 OS

Table of Contents

1 Introduction

ASUS API is a layer between hardware drivers and user applications. When a user application wants to access hardware resources (fan, watchdog, GPIO), it calls the ASUS API function and this function will use dirver or system calls to perform the task.

Supported functions:

- Obtaining general information about the system
- System monitoring: thermal, voltage, fan, etc.
- Watchdog, GPIO control
- Power scheduling
- Connectivity management

ASUS API is compatible with EAPI specification and goes a step further to offer additional features. ASUS API is released in the form of the dynamic-link library, so it can be easily used by an arbitrary application delveloped in C++, C# or higher programming languages. To use ASUS API, application developers only have to add the ASUS API library to their project.

1.1 File Description

To use the ASUS API, copy the following files to your application folder. We provide the sample code of how to use the API library, facilitating the development of your programs.
The provided files are:

Linux:	
output\root\install_asus_library.sh	The shell script to install the library files, the header files, and the binary files to the environment of the ASUS device product
output\root\usr\lib\libasusapi.so	Shared object library
output\root\usr\lib\libasusapi.a	Archive library
output\root\usr\include\asusapi\asusapi.h	Header file
output\root\usr\bin\asusapiapp	Sample application
output\root\examples\dynamic\README.md	Usage of sample application
output\root\examples\dynamic\asusapiapp.c	Code of samepl application

1.2 ASUS API Supported Functions

ASUS API Functions	PE100A	Tinker Edge R	Tinker Board 2/2S	PV100A
EApiBoardGetStringA				
EAPI_ID_BOARD_MANUFACTURER_STR	V	V	V	V
EAPI_ID_BOARD_NAME_STR	V	V	V	V
EAPI_ID_BOARD_REVISION_STR	V	V	V	V
EAPI_ID_BOARD_SERIAL_STR				
EAPI_ID_BOARD_BIOS_REVISION_STR				
EAPI_ID_BOARD_HW_REVISION_STR	V	V	V	V
EAPI_ID_BOARD_PLATFORM_TYPE_STR				
EApiBoardGetValue				
EAPI_ID_GET_EAPI_SPEC_VERSION	V	V	V	V
EAPI_ID_BOARD_BOOT_COUNTER_VAL				
EAPI_ID_BOARD_RUNNING_TIME_METER_VAL				
EAPI_ID_BOARD_PNPID_VAL	V	V	V	V
EAPI_ID_BOARD_PLATFORM_REV_VAL				
EAPI_ID_BOARD_DRIVER_VERSION_VAL				
EAPI_ID_BOARD_LIB_VERSION_VAL	V	V	V	V
EAPI_ID_HWMON_CPU_TEMP				
EAPI_ID_HWMON_CHIPSET_TEMP				
EAPI_ID_HWMON_SYSTEM_TEMP				
EAPI_ID_HWMON_VOLTAGE_VCORE				
EAPI_ID_HWMON_VOLTAGE_2V5				
EAPI_ID_HWMON_VOLTAGE_3V3				
EAPI_ID_HWMON_VOLTAGE_VBAT				
EAPI_ID_HWMON_VOLTAGE_5V				
EAPI_ID_HWMON_VOLTAGE_5VSB				
EAPI_ID_HWMON_VOLTAGE_12V				
EAPI_ID_HWMON_FAN_CPU				
EAPI_ID_HWMON_FAN_SYSTEM				
ASUS_ID_HWMON_VOLTAGE_VTT				
ASUS_ID_HWMON_VOLTAGE_DCIN				V
ASUS_ID_HWMON_CURRENT_SYSTEM				
ASUS_ID_HWMON_SENSOR_01_TEMP				V
ASUS_ID_HWMON_SENSOR_02_TEMP				V
ASUS_ID_HWMON_SENSOR_03_TEMP				V
AsusBoardSetValue				
AsusBoardGetComPortMode				V
AsusBoardSetComPortMode				V
AsusBoardGetModeOfMiniPCIeSlot				V
AsusBoardSetModeOfMiniPCIeSlot				V
AsusBoardGetSelectedSIMSlot				V
AsusBoardSelectSIMSlot				V
EApiI2CGetBusCap			V	
EApiI2CWriteReadRaw			V	
EApiI2CReadTransfer			V	
EApiI2CWriteTransfer			V	
EApiI2CProbeDevice			V	
EApiWDogGetCap	V		V	
EApiWDogStart	V		V	
EApiWDogTrigger	V		V	
EApiWDogStop	V		V	
AsusWDogStartWdtService				
EApiGPIOGetDirectionCaps	V		V	V
EApiGPIOGetDirection	V		V	V
EApiGPIOSetDirection	V		V	V
EApiGPIOGetLevel	V		V	V
EApiGPIOSetLevel	V		V	V

ASUS API Functions	PE100A	Tinker Edge R	Tinker Board 2/2S	PV100A
AsusConnMgrModemGetNumberofModems	V1	V1		
AsusConnMgrModemGetModemInfo	V1	V1		
AsusConnMgrModemStartNetwork	V1	V1		
AsusConnMgrModemStopNetwork	V1	V1		
AsusConnMgrModemPowerOn	V1	V1		
AsusConnMgrModemPowerOff	V1	V1		
AsusConnMgrModemRestart	V1	V1		
AsusConnMgrModemSwitchSIM				
AsusConnMgrModemGetAttachedStatus	V1	V1		
AsusConnMgrModemGetStatus	V1	V1		
AsusConnMgrModemUnlockSIMByPIN	V1	V1		
AsusConnMgrModemSetFlightMode	V1	V1		
AsusConnMgrModemSetAPN	V1	V1		
AsusConnMgrModemSetUser	V1	V1		
AsusConnMgrModemSetPassword	V1	V1		
AsusConnMgrModemSetIPType	V1	V1		
AsusConnMgrModemCheckCarrier	V1	V1		
AsusConnMgrModemSwitchCarrier	V1	V1		
AsusConnMgrModemGetICCID	V1	V1		
AsusConnMgrModemGetIMSI	V1	V1		
AsusConnMgrModemGetSignalStrength	V1	V1		
AsusConnMgrModemGetAdvancedSignalInfo	V1	V1		
AsusConnMgrModemGetProfile	V1	V1		
AsusConnMgrModemResetProfile	V1	V1		
AsusConnMgrModemGetCellLocationInfo	V1	V1		
AsusConnMgrModemSetKeepAlive	V1	V1		
AsusConnMgrModemGetKeepAliveStatus	V1	V1		
AsusConnMgrSetFailover	V1	V1		
AsusConnMgrGetFailoverStatus	V1	V1		
AsusConnMgrSetFailoverGroup	V1	V1		
AsusConnMgrGetFailoverGroup	V1	V1		
AsusVcpGSensorAvailable				V
AsusVcpGSensorEnable				V
AsusVcpGSensorDisable				V
AsusVcpGSensorGetStatus				V
AsusVcpGSensorRead				V
AsusVcpGSensorGetResolution				V
AsusVcpGSensorSetResolution				V
AsusVcpGSensorGetOffset				V
AsusVcpGSensorSetOffset				V
AsusVcpRtcTimeSet				V
AsusVcpRtcTimeGet				V
AsusVcpComPortInit				V
AsusVcpComPortEnable				V
AsusVcpComPortDisable				V
AsusVcpComPortRead				V
AsusVcpComPortWrite				V
AsusVcpWakeupEnable				V
AsusVcpWakeupDisable				V
AsusVcpGetWakeupStatus				V
AsusVcpGetLastWakeupSource				V
AsusVcpPowerMgmtSetPowerOnEventDelay				V
AsusVcpPowerMgmtGetPowerOnEventDelay				V
AsusVcpPowerMgmtSetPowerOffEventDelay				V
AsusVcpPowerMgmtGetPowerOffEventDelay				V
AsusVcpPowerMgmtSetLowBatteryProtectionPreBootThreshold				V
AsusVcpPowerMgmtGetLowBatteryProtectionPreBootThreshold				V

Notes:

V1 The support of the API depends on the modem module products. Please contact with the vendor for more information.

1.3 Supported OS

The following table contains supported Operation Systems for the ASUS products allowed to use ASUS API.

ASUS Product	Operation System
PE100A	Yocto 3.2 and Debian 10 since the image version v0.0.4
Tinker Edge R	Debian 10 since the image version v2.0.2
Tinker Board 2	Debian 10 since the image version v2.0.2
PV100A	Yocto Sumo since the image version v1.0.2

2 Function Documentation

2.1 Initailization Functions

2.1.1 EApiLibInitialize

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

Description

Initialization of ASUS API. Prior to calling any ASUS API function, the library needs to be initialized by calling this function. The status code for all API function will be ASUS_API_STATUS_NOT_INITIALIZED unless this function is called.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_INITIALIZED	Library initialized.

2.1.2 EApiLibUnInitialize

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

Description

The function to uninitialized the API library. Should be called before program exit.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.

2.2 System Monitoring & Board Information

2.2.1 EApiBoardGetStringA

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

Description

Text information about the hardware platform. Supports EAPI Id and ASUS Id.

Parameters

In/Out	Parameter Name	Description
__IN	Id	Selects the Get String Sub function Id (refer to the following EAPI Id Table and the ASUS Id Table)
__OUT	pBuffer	Pointer to a buffer that receives the value's data
__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 Table

Id	Description	Units/Format
EAPI_ID_BOARD_MANUFACTURER_STR	Board Manufacturer Name	string
EAPI_ID_BOARD_NAME_STR	Board Name	string
EAPI_ID_BOARD_REVISION_STR	Board Revision	string
EAPI_ID_BOARD_SERIAL_STR	Board Serial Number	string
EAPI_ID_BOARD_BIOS_REVISION_STR	Board BIOS Revision	string
EAPI_ID_BOARD_HW_REVISION_STR	Board HW Revision	string
EAPI_ID_BOARD_PLATFORM_TYPE_STR	Board Platform Id	string

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supported.
ASUS_API_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Storage buffer overflow was prevented. Read count was larger than the defined buffer length.

2.2.2 EApiBoardGetValue

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

Description

Information about the hardware platform in value format. Supports EAPI Id and ASUS Id.

Parameters

In/Out	Parameter Name	Description
__IN	Id	Selects the Get Value Sub function Id (refer to the following EAPI Id Table and the ASUS Id Table)
__OUT	pValue	Pointer to a buffer that receives the value's data

EAPI Id Table

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	
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
EAPI_ID_HWMON_CHIPSET_TEMP	Chipset Temperature	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	2.5V Voltage	Millivolts
EAPI_ID_HWMON_VOLTAGE_3V3	3.3V Voltage	Millivolts
EAPI_ID_HWMON_VOLTAGE_VBAT	Battery Voltage	Millivolts
EAPI_ID_HWMON_VOLTAGE_5V	5V Voltage	Millivolts
EAPI_ID_HWMON_VOLTAGE_5VSB	5V Standby Voltage	Millivolts
EAPI_ID_HWMON_VOLTAGE_12V	12V Voltage	Millivolts
EAPI_ID_HWMON_FAN_CPU	CPU Fan	RPM
EAPI_ID_HWMON_FAN_SYSTEM	System Fan	RPM

ASUS Id Table

Id	Description	Units/Format
ASUS_ID_HWMON_VOLTAGE_VTT	VTT Voltage	Millivolts

ASUS_ID_HWMON_VOLTAGE_DCIN	DC-IN Voltage	Millivolts
ASUS_ID_HWMON_CURRENT_SYSTEM	System Current	Milliamperes
ASUS_ID_HWMON_SENSOR_01_TEMP	Sensor 01 Temperature. In PV100A, it indicates the temperature around MB SoC	0.1 Kelvins
ASUS_ID_HWMON_SENSOR_02_TEMP	Sensor 02 Temperature. In PV100A, it indicates the temperature around LTE module	0.1 Kelvins
ASUS_ID_HWMON_SENSOR_03_TEMP	Sensor 03 Temperature. In PV100A, it indicates the temperature around WiFi module	0.1 Kelvins

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.2.3 **AsusBoardSetValue**

```
uint32_t
ASUS_CALLTYPE
AsusBoardSetValue (
    __IN        uint32_t   Id,
    __IN        uint32_t   Value);
```

Description

Sets the value of the system hardware component.

Parameters

In/Out	Parameter Name	Description
__IN	Id	Selects the Id of the system hardware component (refer to the following EAPI Id Table and the ASUS Id Table)

__IN	Value	Sets the value of the selected system hardware component.

EAPI Id Table

Id	Description	Units/Format
EAPI_ID_HWMON_FAN_CPU	CPU Fan	RPM
EAPI_ID_HWMON_FAN_SYSTEM	System Fan	RPM

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
AUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.2.4 AsusBoardGetComPortMode

```
uint32_t
ASUS_CALLTYPE
AsusBoardGetComPortMode (
    __IN        uint32_t  Id,
    __OUT       enum ASUS_COM_PORT_MODE *pMode);
```

Description

Gets the com port mode of the com port Id.

Parameters

In/Out	Parameter Name	Description
__IN	Id	The com port Id
__OUT	pMode	Pointer to a variable that receives the com port mode. Refer to the following enum: ASUS_COM_PORT_MODE.

Enum of ASUS_COM_PORT_MODE

Parameter	Description

ASUS_COM_PORT_MODE_RS232	RS232
ASUS_COM_PORT_MODE_RS422	RS422
ASUS_COM_PORT_MODE_RS485	RS485

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supported.

2.2.5 AsusBoardSetComPortMode

```
uint32_t
ASUS_CALLTYPE
AsusBoardSetComPortMode (
    __IN        uint32_t   Id,
    __IN        enum ASUS_COM_PORT_MODE Mode);
```

Description

Sets the com port mode of the com port Id.

Parameters

In/Out	Parameter Name	Description
__IN	Id	The com port Id
__IN	Mode	The com port mode. Refer to the following enum: ASUS_COM_PORT_MODE.

Enum of ASUS_COM_PORT_MODE

Parameter	Description
ASUS_COM_PORT_MODE_RS232	RS232
ASUS_COM_PORT_MODE_RS422	RS422

ASUS_COM_PORT_MODE_RS485	RS485

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supported.

2.2.6 AsusBoardGetModeOfMiniPCIeSlot

```
uint32_t
ASUS_CALLTYPE
AsusBoardGetModeOfMiniPCIeSlot (
    __IN    enum ASUS_BOARD_ID_OF_MPCIE_SLOT Id,
    __OUT   enum ASUS_BOARD_MODE_OF_MPCIE_SLOT *pMode);
```

Description

Gets the mode of the mPCIe slot.

Parameters

In/Out	Parameter Name	Description
__IN	Id	The mPCIe slot Id. Refer to the following enum: ASUS_BOARD_ID_OF_MPCIE_SLOT.
__OUT	pMode	Pointer to a variable that receives the mode of the mPCIe slot. Refer to the following enum: ASUS_BOARD_MODE_OF_MPCIE_SLOT.

Enum of ASUS_BOARD_ID_OF_MPCIE_SLOT

Parameter	Description
MPCIE_SLOT_ID00	The mPCIe slot Id 0

Enum of ASUS_BOARD_MODE_OF_MPCIE_SLOT

Parameter	Description
PCIE_MODE_OF_MPCIE_SLOT	The PCIe mode
SATA_MODE_OF_MPCIE_SLOT	The SATA mode

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supported.

2.2.7 AsusBoardSetModeOfMiniPCIeSlot

```
uint32_t
ASUS_CALLTYPE
AsusBoardSetModeOfMiniPCIeSlot (
    __IN        enum ASUS_BOARD_ID_OF_MPCIE_SLOT Id,
    __IN        enum ASUS_BOARD_MODE_OF_MPCIE_SLOT Mode);
```

Description

Sets the mode of the mPCIe slot.

Parameters

In/Out	Parameter Name	Description
__IN	Id	The mPCIe slot Id. Refer to the following enum: ASUS_BOARD_ID_OF_MPCIE_SLOT.
__IN	Mode	The mode of the mPCIe slot. Refer to the following enum: ASUS_BOARD_MODE_OF_MPCIE_SLOT.

Enum of ASUS_BOARD_ID_OF_MPCIE_SLOT

Parameter	Description
MPCIE_SLOT_ID00	The mPCIe slot Id 0

Enum of ASUS_BOARD_MODE_OF_MPCIE_SLOT

Parameter	Description
PCIE_MODE_OF_MPCIE_SLOT	The PCIe mode
SATA_MODE_OF_MPCIE_SLOT	The SATA mode

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supported.

2.2.8 AsusBoardGetSelectedSIMSlot

```
uint32_t
ASUS_CALLTYPE
AsusBoardGetSelectedSIMSlot (
    __OUT       enum ASUS_BOARD_ID_OF_SIM_SLOT *pId);
```

Description

Gets the current selected SIM slot Id.

Parameters

In/Out	Parameter Name	Description
__OUT	pId	Pointer to a variable that receives the current selected SIM slot Id. Refer to the following enum: ASUS_BOARD_ID_OF_SIM_SLOT.

Enum of ASUS_BOARD_ID_OF_SIM_SLOT

Parameter	Description
BOARD_SIM_SLOT_ID00	The SIM slot Id 0
BOARD_SIM_SLOT_ID01	The SIM slot Id 1

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supported.

2.2.9 AsusBoardSelectSIMSlot

```
uint32_t
ASUS_CALLTYPE
AsusBoardSelectSIMSlot (
    __IN        enum ASUS_BOARD_ID_OF_SIM_SLOT Id);
```

Description

Selects the SIM slot to be used.

Parameters

In/Out	Parameter Name	Description
__IN	Id	The SIM slot Id. Refer to the following enum: ASUS_BOARD_ID_OF_SIM_SLOT.

Enum of ASUS_BOARD_ID_OF_SIM_SLOT

Parameter	Description
BOARD_SIM_SLOT_ID00	The SIM slot Id 0
BOARD_SIM_SLOT_ID01	The SIM slot Id 1

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.

ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supported.

2.3 GPIO Functions

2.3.1 EApiGPIOGetDirectionCaps

```
uint32_t
EAPI_CALLTYPE
EApiGPIOGetDirectionCaps (
    __IN        uint32_t  Id,
    __OUTPUT    uint32_t  *pInputs,
    __OUTPUT    uint32_t  *pOutputs);
```

Description

Reads the capabilities of the current GPIO implementation from the selected GPIO interface. The direction of this port can be configurred by EApiGPIOSetDirection.

Parameters

In/Out	Parameter Name	Description
__IN	Id	GPIO Ids
__OUTPUT	pInputs	Pointer to a buffer that receives the bit mask of the supported inputs
__OUTPUT	pOutputs	Pointer to a buffer that receives the bit mask of the supported outputs

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.3.2 EApiGPIOGetDirection

```
uint32_t
EAPI_CALLTYPE
EApiGPIOGetDirection (
    __IN        uint32_t  Id,
    __IN        uint32_t  Bitmask,
    __OUT       uint32_t  *pDirection);
```

Description

Reads the current configuration of the selected GPIO ports.

Parameters

In/Out	Parameter Name	Description
__IN	Id	GPIO Ids
__IN	Bitmask	Bit mask. Only selected bits are returned. Unselected bits return 0
__OUT	pDirection	Pointer to a buffer that receives the direction of the selected GPIO ports. Bits with the value EAPI_GPIO_INPUT are inputs, bits with EAPI_GPIO_OUTPUT are outputs

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_BITMASK	The bitmask selects bits/GPIOs which are not supported for the current Id.

2.3.3 EApiGPIOSetDirection

```
uint32_t
EAPI_CALLTYPE
EApiGPIOSetDirection (
    __IN        uint32_t  Id,
    __IN        uint32_t  Bitmask,
    __IN        uint32_t  Direction);
```

Description

Sets the configuration for the selected GPIO ports.

Parameters

In/Out	Parameter Name	Description
__IN	Id	GPIO Ids
__IN	Bitmask	Bit mask. Only selected bits are changed. Unselected bits remain unchanged
__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

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_BITMASK	The bitmask selects bits/GPIOs which are not supported for the current Id.
ASUS_API_STATUS_INVALID_DIRECTION	The current direction argument attemps to set GPIOs to unsupportted directions.

2.3.4 EApiGPIOGetLevel

```
uint32_t
EAPI_CALLTYPE
EApiGPIOGetLevel (
    __IN        uint32_t  Id,
    __IN        uint32_t  Bitmask,
    __OUT       uint32_t  *pLevel);
```

Description

Read the level from GPIO ports.

Parameters

In/Out	Parameter Name	Description
__IN	Id	GPIO Ids
__IN	Bitmask	Bit mask. Only selected bits are returned. Unselected bits return 0
__OUT	pLevel	Pointer to a buffer that receives the level of the selected GPIO ports. Bits with the value EAPI_GPIO_HIGH are high levels, bits with EAPI_GPIO_LOW are low levels

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_BITMASK	The bitmask selects bits/GPIOs which are not supported for the current Id.

2.3.5 EApiGPIOSetLevel

```
uint32_t
EAPI_CALLTYPE
EApiGPIOSetLevel (
    __IN        uint32_t  Id,
    __IN        uint32_t  Bitmask,
    __IN        uint32_t  Level);
```

Description

Write level 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.

Parameters

In/Out	Parameter Name	Description
__IN	Id	GPIO Ids
__IN	Bitmask	Bit mask. Only selected bits are changed. Unselected bits remain unchanged
__IN	Level	Sets the levels of the selected GPIO ports. Bits with the value EAPI_GPIO_HIGH are high levels, bits with EAPI_GPIO_LOW are low levels

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_BITMASK	The bitmask selects bits/GPIOs which are not supported for the current Id.

2.4 Watchdog

2.4.1 EApiWDogGetCap

```
uint32_t
EAPI_CALLTYPE
EApiWDogGetCap (
    __OUTPUT    uint32_t   *pMaxDelay,
    __OUTPUT    uint32_t   *pMaxEventTimeout,
    __OUTPUT    uint32_t   *pMaxResetTimeout);
```

Description

Get the capibilities of the watchdog timer.

Parameters

In/Out	Parameter Name	Description
__OUTPUT	pMaxDelay	Pointer to a buffer that receives maximum supported initial delay time of the watchdog timer in milliseconds. If the value returns 0, it means not support event timeouot.
__OUTPUT	pMaxEventTimeout	Pointer to a buffer that receives maximum supported event timeout of the watchdog timer in milliseconds. If the value returns 0, it means not support event timeouot.
__OUTPUT	pMaxResetTimeout	Pointer to a buffer that receives maximum supported reset timeout of the watchdog timer in milliseconds.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the watchdog. The caller should have root permission.

2.4.2 EApiWDogStart

```
uint32_t
EAPI_CALLTYPE
EApiWDogStart (
    __IN    uint32_t  Delay,
    __IN    uint32_t  EventTimeout,
    __IN    uint32_t  ResetTimeout);
```

Description

Start the watchdog timer and set the parameters. To adjust the parameters, the watchdog must be stopped via EApiWDogStop and then EApiWDogStart must be called again with the new values.

In Linux, the watchdog daemon should be disabled before calling this function. That is to say, “RuntimeWatchdogSec” and “ShutdownWatchdogSec” need to be commented in the file /etc/systemd/system.conf

Parameters

In/Out	Parameter Name	Description
__IN	Delay	Initial delay for the watchdog timer in milliseconds.
__IN	EventTimeout	Watchdog timeout interval in milliseconds to trigger an event.
__IN	ResetTimeout	Watchdog timeout interval in milliseconds to trigger a reset.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_RUNNING	Watchdog timer already started.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the watchdog. The caller should have root permission.

2.4.3 AsusWDogStartWdtService

```
uint32_t
ASUS_CALLTYPE
AsusWDogStartWdtService (
    __IN    uint32_t  Delay,
    __IN    uint32_t  ResetTimeout);
```

Description

Start the Watchdog Timer Service (the WDT Service) and set the parameters. To adjust the parameters, the WDT Service must be stopped via EApiWDogStop and then AsusWDogStartWdtService must be called again with the new values.

After the WDT Service has been started by the AsusWDogStartWdtService function, the WDT Service will automatically send the first trigger within (Delay + ResetTimeouot) milliseconds as set with AsusWDogStartWdtService function, following the first trigger every subsequent trigger will be sent by the WDT Service within (ResetTimeout) milliseconds.

Parameters

In/Out	Parameter Name	Description
__IN	Delay	Initial delay for the watchdog timer in milliseconds.
__IN	ResetTimeout	Watchdog timeout interval in milliseconds to trigger a reset.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_RUNNING	Watchdog timer already started.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the watchdog. The caller should have root permission.

2.4.4 EApiWDogTrigger

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

Description

Trigger the watchdog timer.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the watchdog. The caller should have root permission.

2.4.5 EApiWDogStop

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

Description

Stops the operation of the watchdog timer.

The Watchdog Timer Service (the WDT Service) will be stopped as well if the WDT Service is running.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the watchdog. The caller should have root permission.

2.5 Power Scheduling

2.5.1 AsusSystemBootSet

```
uint32_t
ASUS_CALLTYPE
AsusSystemBootSet (
    __IN    uint32_t  Id,
    __IN    uint32_t  Frequency,
    __IN    uint32_t  DayofMonth,
    __IN    uint32_t  DayofWeek,
    __IN    uint32_t  Hour,
    __IN    uint32_t  Minute,
    __IN    uint32_t  Second);
```

Description

Power Scheduling provides the boot up, shutdown, restartr, and sleep function for the system state management.

Be able to set Id for the action of Power Scheduling, such as boot up, shutdown, restart, sleep, etc., and be able to schedule the freguency, which can be daily, weekly, monthly, or one shot, on that action.

Parameters

In/Out	Parameter Name	Description
__IN	Id	Selects Power Scheduling function Id (refer to the following ASUS Power Scheduling Id Table)
__IN	Frequency	Selects Power Scheduling Frequency (refer to the following ASUS Power Scheduling Frequency Table)
__IN	DayofMonth	Selects the day of month, such as 1, 2, …, 30, and 31.
__IN	DayofWeek	Selects the day of week (refer to the following ASUS DayofWeek Table)
__IN	Hour	Selects the hour, such as 0, 1, …, 22, and 23.
__IN	Minute	Selects the minute, such as 0, 1, …, 58, and 59.
__IN	Second	Selects the second, such as 0, 1, …, 58, and 59.

ASUS Power Scheduling Id Table

Id	Description
ASUS_ID_SYSTEM_BOOT_BOOTUP	Boot up or wake up function. Boot up or wake up the system.
ASUS_ID_SYSTEM_BOOT_SHUTDOWN	Shutdown function. Shutdown the system.
ASUS_ID_SYSTEM_BOOT_SLEEP	Sleep function. Sleep the system.
ASUS_ID_SYSTEM_BOOT_RESTART	Restart function. Restart the system.

ASUS Power Scheduling Frequency Table

Id	Description
ASUS_SYSTEM_BOOT_DISABLE_FREQUENCY	Disable the selected function. The arguments DayofMonth, DayofWeek, Hour, Minute, and Second are useless.
ASUS_SYSTEM_BOOT_ENABLE_FREQUENCY_DAILY	The selected function will action daily. The arguments DayofMonth, DayofWeek are useless.
ASUS_SYSTEM_BOOT_ENABLE_FREQUENCY_WEEKLY	The selected function will action weekly. The argument DayofMonth is useless.
ASUS_SYSTEM_BOOT_ENABLE_FREQUENCY_MONTHLY	The selected function will action monthly. The argument DayofWeek is useless.
ASUS_SYSTEM_BOOT_ENABLE_FREQUENCY_ONE_SHOT	The selected function will action one shot. The arguments DayofMonth, DayofWeek are useless.

ASUS Power Scheduling Frequency Table

Id
ASUS_SYSTEM_BOOT_WEEKLY_SUNDAY
ASUS_SYSTEM_BOOT_WEEKLY_MONDAY
ASUS_SYSTEM_BOOT_WEEKLY_TUESDAY
ASUS_SYSTEM_BOOT_WEEKLY_WEDNESDAY
ASUS_SYSTEM_BOOT_WEEKLY_THURSDAY
ASUS_SYSTEM_BOOT_WEEKLY_FRIDAY

ASUS_SYSTEM_BOOT_WEEKLY_SATURDAY

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.5.2 AsusSystemBootGet

```
uint32_t
ASUS_CALLTYPE
AsusSystemBootGet (
    __IN    uint32_t  Id,
    __OUT   uint32_t  *pFrequency,
    __OUT   uint32_t  *pDayofMonth,
    __OUT   uint32_t  *pDayofWeek,
    __OUT   uint32_t  *pHour,
    __OUT   uint32_t  *pMinute,
    __OUT   uint32_t  *pSecond);
```

Description

Reads the frequency and the configuration of the Power Scheduling function from the selected function Id.

The frequency and the configuration of the Power Scheduling function can be set by the AsusSystemBootSet().

Parameters

In/Out	Parameter Name	Description
__IN	Id	Selects Power Scheduling function Id (refer to the following ASUS Power Scheduling Id Table)
__OUT	pFrequency	Pointer to a buffer that receives the frequency of the selected function Id (refer to the following ASUS Power Scheduling

		Frequency Table)
__OUT	pDayofMonth	Pointer to a buffer that receives the day of month of the selected function Id
__OUT	pDayofWeek	Pointer to a buffer that receives the day of week of the selected function Id
__OUT	pHour	Pointer to a buffer that receives the hour of the selected function Id
__OUT	pMinute	Pointer to a buffer that receives the minute of the selected function Id
__OUT	pSecond	Pointer to a buffer that receives the second of the selected function Id

ASUS Power Scheduling Id Table

Id	Description
ASUS_ID_SYSTEM_BOOT_BOOTUP	Boot up or wake up function.
ASUS_ID_SYSTEM_BOOT_SHUTDOWN	Shutdown function.
ASUS_ID_SYSTEM_BOOT_SLEEP	Sleep function.
ASUS_ID_SYSTEM_BOOT_RESTART	Restart function.

ASUS Power Scheduling Frequency Table

Id	Description
ASUS_SYSTEM_BOOT_DISABLE_FREQUENCY	The selected function is disabled. The arguments DayofMonth, DayofWeek, Hour, Minute, and Second are useless.
ASUS_SYSTEM_BOOT_ENABLE_FREQUENCY_DAILY	The selected function actions daily. The arguments DayofMonth, DayofWeek are useless.
ASUS_SYSTEM_BOOT_ENABLE_FREQUENCY_WEEKLY	The selected function actions weekly. The argument DayofMonth is useless.
ASUS_SYSTEM_BOOT_ENABLE_FREQUENCY_MONTHLY	The selected function actions monthly. The argument DayofWeek is useless.
ASUS_SYSTEM_BOOT_ENABLE_FREQUENCY_ONE_SHOT	The selected function actions one shot. The arguments DayofMonth, DayofWeek are useless.

ASUS Power Scheduling Frequency Table

Id
ASUS_SYSTEM_BOOT_WEEKLY_SUNDAY
ASUS_SYSTEM_BOOT_WEEKLY_MONDAY
ASUS_SYSTEM_BOOT_WEEKLY_TUESDAY
ASUS_SYSTEM_BOOT_WEEKLY_WEDNESDAY
ASUS_SYSTEM_BOOT_WEEKLY_THURSDAY
ASUS_SYSTEM_BOOT_WEEKLY_FRIDAY
ASUS_SYSTEM_BOOT_WEEKLY_SATURDAY

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.6 Functions for the I2C Bus

2.6.1 EApiI2CGetBusCap

```
uint32_t
EAPI_CALLTYPE
EApiI2CGetBusCap (
    __IN        uint32_t  Id,
    __OUTPUT    uint32_t  *pMaxBlkLen);
```

Description

Gets the capabilities of the selected I2C bus.

Parameters

In/Out	Parameter Name	Description

__IN	Id	I2C bus Id
__OUTPUT	pMaxBlkLen	Size in bytes. Pointer to a buffer that receives the maximum transfer block length for the given interface.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.6.2 EApiI2CWriteReadRaw

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

Description

Universal function for read and write operations to the I2C bus.

Parameters

In/Out	Parameter Name	Description
__IN	Id	I2C bus Id
__IN	Addr	Encoded 7Bit I2C Device Address. It can be derived by the macro for address encoding: EAPI_I2C_ENC_7BIT_ADDR()
__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	Pointerr 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 ASUS_API_STATUS_MORE_DATA. If pRBuffer is NILL, 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.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_BLOCK_LENGTH	This means that the block length is too long.
ASUS_API_STATUS_BUSY_COLLISION	The selected device or Id is busy or a data collision was detected.
ASUS_API_STATUS_NOT_FOUND	Selected device was not found.
ASUS_API_STATUS_WRITE_ERROR	An error was detected during a write operation.
ASUS_API_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Storage buffer overflow was prevented. Read count was larger than the defined buffer length.
ASUS_API_STATUS_TIMEOUT	Timeout due to clock stretching

2.6.3 EApiI2CReadTransfer

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

Description

Reads from a specific register in the selected I2C device.

Reads from I2C device at the I2C address Addr the amount of ByteCnt bytes to the buffer pBuffer while using the device specific command Cmd. Depending on the addressed I2C device, Cmd can be a specific command or a byte offset.

Parameters

In/Out	Parameter Name	Description
__IN	Id	I2C bus Id
__IN	Addr	Encoded 7Bit I2C Device Address. It can be derived by the macro for address encoding: EAPI_I2C_ENC_7BIT_ADDR()
__IN	Cmd	I2C command/offset
__OUT	pBuffer	Pointerr 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 ASUS_API_STATUS_MORE_DATA.
__IN	ByteCnt	Size, in bytes, of data to be read

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.

ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_BLOCK_LENGTH	This means that the block length is too long.
ASUS_API_STATUS_BUSY_COLLISION	The selected device or Id is busy or a data collision was detected.
ASUS_API_STATUS_NOT_FOUND	Selected device was not found.
ASUS_API_STATUS_WRITE_ERROR	An error was detected during a write operation.
ASUS_API_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Storage buffer overflow was prevented. Read count was larger than the defined buffer length.
ASUS_API_STATUS_TIMEOUT	Timeout due to clock stretching

2.6.4 **EApiI2CWriteTransfer**

```
uint32_t
EAPI_CALLTYPE
EApiI2CWriteTransfer (
    __IN        uint32_t  Id,
    __IN        uint32_t  Addr,
    __IN        uint32_t  Cmd,
    __IN        void      *pBuffer,
    __IN        uint32_t  ByteCnt);
```

Description

Write to a specific register in the selected I2C device.

Write to an I2C device at the I2C address Addr the amount of ByteCnt bytes from the buffer *pBuffer while using the device specific command Cmd. Depending on the addressed I2C device, Cmd can be a specific command or a byte offset.

Parameters

In/Out	Parameter Name	Description
__IN	Id	I2C bus Id
__IN	Addr	Encoded 7Bit I2C Device Address. It can be derived by the macro for

		address encoding: EAPI_I2C_ENC_7BIT_ADDR()
__IN	Cmd	I2C command/offset
__IN	pBuffer	Pointerr to a buffer containing the data to be transferred.
__IN	ByteCnt	Size, in bytes, of the information pointed to by the pBuffer parameter

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_BLOCK_LENGTH	This means that the block length is too long.
ASUS_API_STATUS_BUSY_COLLISION	The selected device or Id is busy or a data collision was detected.
ASUS_API_STATUS_NOT_FOUND	Selected device was not found.
ASUS_API_STATUS_WRITE_ERROR	An error was detected during a write operation.
ASUS_API_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Storage buffer overflow was prevented. Read count was larger than the defined buffer length.
ASUS_API_STATUS_TIMEOUT	Timeout due to clock stretching

2.6.5 EApiI2CProbeDevice

```
uint32_t
EAPI_CALLTYPE
EApiI2CProbeDevice (
    __IN        uint32_t  Id,
    __IN        uint32_t  Addr);
```

Description

Probes I2C address to test I2C device present.

Parameters

In/Out	Parameter Name	Description
__IN	Id	I2C bus Id
__IN	Addr	Encoded 7Bit I2C Device Address. It can be derived by the macro for address encoding: EAPI_I2C_ENC_7BIT_ADDR()

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_BUSY_COLLISION	The selected device or Id is busy or a data collision was detected.
ASUS_API_STATUS_NOT_FOUND	Selected device was not found.
ASUS_API_STATUS_TIMEOUT	Timeout due to clock stretching

2.7 Connectivity Management

2.7.1 AsusConnMgrModemGetNumberofModems

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemGetNumberofModems (
    __OUT    uint32_t   *pValue);
```

Description

Get the number of the modems.

Parameters

In/Out	Parameter Name	Description
__OUT	pValue	Pointer to a variable that specifies the number of the modems.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supported.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.2 AsusConnMgrModemGetModemInfo

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemGetModemInfo (
    __OUT       struct ConnMgrModemInfo_s *pModemList,
    __INOUT     uint32_t   *pModemCnt);
```

Description

Gets the information of the modems.

Parameters

In/Out	Parameter Name	Description
__OUT	pModemList	An array of the structures that receives the information of the modems (refer to the following structure: ConnMgrModemInfo_s).

__INOUT	pModemCnt	Pointer to a variable that specifies the number of the ConnMgrModemInfo_s structures pointed to by the pModemList parameter. When the function returns, this variable contains the number of the modems. If the buffer specified by pModemList parameter is not large enough to hold the number of the modems, the function returns the value ASUS_API_STATUS_MORE_DATA.

Structure of ConnMgrModemInfo_s

Parameter Type	Parameter Name	Description
char	Index	The index of the modem
char[] (The array size is defined by MAX_CHAR_LENGTH_MODEM_INFO)	Path	The path of the modem
char[] (The array size is defined by MAX_CHAR_LENGTH_MODEM_INFO)	Manufacturer	The manufacturer of the modem
char[] (The array size is defined by MAX_CHAR_LENGTH_MODEM_INFO)	ModemName	The modem name
char[] (The array size is defined by MAX_CHAR_LENGTH_MODEM_INFO)	FirmwareRevision	The firmware revision of the modem

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supported.
ASUS_API_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Storage buffer overflow was prevented. Read count was larger than the defined buffer length.

ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.3 **AsusConnMgrModemStartNetwork**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemStartNetwork (__IN uint32_t Index);
```

Description

Start the network connectivity.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.4 AsusConnMgrModemStopNetwork

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemStopNetwork (__IN uint32_t Index);
```

Description

Stop the network connectivity.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.5 AsusConnMgrModemPowerOn

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemPowerOn (__IN uint32_t Index);
```

Description

Power on the modem.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.7.6 AsusConnMgrModemPowerOff

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemPowerOff (__IN uint32_t Index);
```

Description

Power off the modem.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.

ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.7 **AsusConnMgrModemRestart**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemRestart (__IN uint32_t Index);
```

Description

Power off and power on the modem.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.8 AsusConnMgrModemGetKeepAliveStatus

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemGetKeepAliveStatus (
    __OUT       uint32_t *pStatus);
```

Description

Get the status of the keep alive feature.

Parameters

In/Out	Parameter Name	Description
__OUT	pStatus	Pointer to a variable that receives the value's data. The value with ASUS_CMM_KEEP_ALIVE_ON means the keep alive feature is on. The value with ASUS_CMM_KEEP_ALIVE_OFF means the keep alive feature is off.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.9 AsusConnMgrModemSetKeepAlive

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemSetKeepAlive (__IN uint32_t Enable);
```

Description

Allow to enable or disable the keep alive feature.

Parameters

In/Out	Parameter Name	Description
__IN	Enable	The value with ASUS_CMM_KEEP_ALIVE_ON means to turn on the keep alive feature. The value with ASUS_CMM_KEEP_ALIVE_OFF means to turn off.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.10 **AsusConnMgrModemGetStatus**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemGetStatus (
    __IN        uint32_t Index,
    __OUT       struct ConnMgrModemStatus_s *pStatus);
```

Description

Get the status of the network connection and the information of IP.

Parameters

In/Out	Parameter Name	Description

__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().
__OUT	pStatus	Pointer to a structure that receives the status of the network connection and the information of IP (refer to the following structure: ConnMgrModemStatus_s).

Structure of ConnMgrModemStatus_s

In/Out Parameter Type	Parameter Name	Description
__OUT char *	pConnected	Pointer to a variable that specifies the status of the network connection. The variable might be the string “yes” or the string “no”.
__INOUT uint32_t *	pConnectedLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pConnected parameter. When the function returns, this variable contains the size of the data copied to pConnected including the terminating null character. If the buffer specified by pConnected parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pInterface	Pointer to a buffer that receives the interface name.
__INOUT uint32_t *	pInterfaceLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pInterface parameter. When the function returns, this variable contains the size of the data copied to pInterface including the terminating null character. If the buffer specified by pInterface parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pApn	Pointer to a buffer that receives the APN name.
__INOUT uint32_t *	pApnLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pApn parameter. When the function returns, this

		variable contains the size of the data copied to pApn including the terminating null character. If the buffer specified by pApn parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pRoaming	Pointer to a variable that specifies the status of the roaming. The variable might be the string “allowed” or the string “forbidden”.
__INOUT uint32_t *	pRoamingLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pRoaming parameter. When the function returns, this variable contains the size of the data copied to pRoaming including the terminating null character. If the buffer specified by pRoaming parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT struct ConnMgrModemIpv4Status_s *	pIpv4Status	Pointer to a structure that receives the information of IPv4 (refer to the following structure: ConnMgrModemIpv4Status_s).
__OUT struct ConnMgrModemIpv6Status_s *	pIpv6Status	Pointer to a structure that receives the information of IPv6 (refer to the following structure: ConnMgrModemIpv6Status_s).

Structure of ConnMgrModemIpv4Status_s

In/Out Parameter Type	Parameter Name	Description
__OUT char *	pAddress	Pointer to a buffer that receives the IP address.
__INOUT uint32_t *	pAddressLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pAddress parameter. When the function returns, this variable contains the size of the data copied to pAddress including the terminating null character. This variable will be zero if the IP address is invalid. If the buffer specified by pAddress parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pGateway	Pointer to a buffer that receives the gateway address.
__INOUT uint32_t *	pGatewayLen	Pointer to a variable that specifies the size, in bytes, of

		the buffer pointed to by the pGateway parameter. When the function returns, this variable contains the size of the data copied to pGateway including the terminating null character. This variable will be zero if the gateway address is invalid. If the buffer specified by pGateway parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pMtu	Pointer to a buffer that receives the MTU.
__INOUT uint32_t *	pMtuLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pMtu parameter. When the function returns, this variable contains the size of the data copied to pMtu including the terminating null character. This variable will be zero if the MTU is invalid. If the buffer specified by pMtu parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pDns	Pointer to a buffer that receives the DNS address.
__INOUT uint32_t *	pDnsLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pDns parameter. When the function returns, this variable contains the size of the data copied to pDns including the terminating null character. This variable will be zero if the DNS address is invalid. If the buffer specified by pDns parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.

Structure of ConnMgrModemIpv6Status_s

In/Out Parameter Type	Parameter Name	Description
__OUT char *	pAddress	Pointer to a buffer that receives the IP address.
__INOUT uint32_t *	pAddressLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pAddress parameter. When the function returns, this variable contains the size of the data copied to pAddress including the terminating null character. This variable will be zero if the IP address is invalid. If the buffer specified by pAddress parameter is not large enough to hold the data, the function returns

		the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pGateway	Pointer to a buffer that receives the gateway address.
__INOUT uint32_t *	pGatewayLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pGateway parameter. When the function returns, this variable contains the size of the data copied to pGateway including the terminating null character. This variable will be zero if the gateway address is invalid. If the buffer specified by pGateway parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pMtu	Pointer to a buffer that receives the MTU.
__INOUT uint32_t *	pMtuLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pMtu parameter. When the function returns, this variable contains the size of the data copied to pMtu including the terminating null character. This variable will be zero if the MTU is invalid. If the buffer specified by pMtu parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pDns	Pointer to a buffer that receives the DNS address.
__INOUT uint32_t *	pDnsLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pDns parameter. When the function returns, this variable contains the size of the data copied to pDns including the terminating null character. This variable will be zero if the DNS address is invalid. If the buffer specified by pDns parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.

ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Stroage buffer overflow was prevented. Read count was larger than the defined buffer length.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.11 **AsusConnMgrModemGetAttachedStatus**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemGetAttachedStatus (
    __IN        uint32_t Index,
    __OUT       struct ConnMgrModemAttachedStatus_s *pStatus);
```

Description

Get the attached status of the modem, including the state of the modem and the access technology that the modem uses when registered with or connected to a network.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().
__OUT	pStatus	Pointer to a structure that receives the attached status of the modem (refer to the following structure: ConnMgrModemAttachedStatus_s).

Structure of ConnMgrModemAttachedStatus_s

Parameter Type	Parameter Name	Description

enum ASUS_CMM_REGISTRATION_STATE	RegistrationState	Refer to the following enum: ASUS_CMM_REGISTRATION_STATE
enum ASUS_CMM_RADIO_INTERFACE	RadioInterface	Refer to the following enum: ASUS_CMM_RADIO_INTERFACE
enum ASUS_CMM_FLIGHT_MODE_STATE	FlightModeState	Refer to the following enum: ASUS_CMM_FLIGHT_MODE_STATE

Enum of ASUS_CMM_REGISTRATION_STATE

Parameter	Description
CMMODEM_STATE_FAILED	The modem is unusable.
CMMODEM_STATE_UNKNOWN	State unknown or not reportable.
CMMODEM_STATE_INITIALIZING	The modem is currently being initialized.
CMMODEM_STATE_LOCKED	The modem needs to be unlocked.
CMMODEM_STATE_DISABLED	The modem is not enabled and is powered down.
CMMODEM_STATE_DISABLING	The modem is currently transitioning to the MM_MODEM_STATE_DISABLED state.
CMMODEM_STATE_ENABLING	The modem is currently transitioning to the MM_MODEM_STATE_ENABLED state.
CMMODEM_STATE_ENABLED	The modem is enabled and powered on but not registered with a network provider and not available for data connections.
CMMODEM_STATE_SEARCHING	The modem is searching for a network provider to register with.
CMMODEM_STATE_REGISTERED	The modem is registered with a network provider, and data connections and messaging may be available for use.
CMMODEM_STATE_DISCONNECTING	The modem is disconnecting and deactivating the last active packet data bearer. This state will not be entered if more than one packet data bearer is active and one of the active bearers is deactivated.
CMMODEM_STATE_CONNECTING	The modem is activating and connecting the first packet data bearer. Subsequent bearer activations when another bearer is already active do not cause this state to be entered.
CMMODEM_STATE_CONNECTED	One or more packet data bearers is active and connected.

Enum of ASUS_CMM_RADIO_INTERFACE

Parameter	Description
CMMODEM_ACCESS_TECHNOLOGY_UNKNOWN	The access technology used is unknown.
CMMODEM_ACCESS_TECHNOLOGY_POTS	Analog wireline telephone.
CMMODEM_ACCESS_TECHNOLOGY_GSM	GSM.
CMMODEM_ACCESS_TECHNOLOGY_GSM_COMPACT	Compact GSM.
CMMODEM_ACCESS_TECHNOLOGY_GPRS	GPRS.
CMMODEM_ACCESS_TECHNOLOGY_EDGE	EDGE (ETSI 27.007: "GSM w/EGPRS").
CMMODEM_ACCESS_TECHNOLOGY_UMTS	UMTS (ETSI 27.007: "UTRAN").
CMMODEM_ACCESS_TECHNOLOGY_HSDPA	HSDPA (ETSI 27.007: "UTRAN w/HSDPA").
CMMODEM_ACCESS_TECHNOLOGY_HSUPA	HSUPA (ETSI 27.007: "UTRAN w/HSUPA").
CMMODEM_ACCESS_TECHNOLOGY_HSPA	HSPA (ETSI 27.007: "UTRAN w/HSDPA and HSUPA").
CMMODEM_ACCESS_TECHNOLOGY_HSPA_PLUS	HSPA+ (ETSI 27.007: "UTRAN w/HSPA+").
CMMODEM_ACCESS_TECHNOLOGY_1XRTT	CDMA2000 1xRTT.
CMMODEM_ACCESS_TECHNOLOGY_EVDO0	CDMA2000 EVDO revision 0.
CMMODEM_ACCESS_TECHNOLOGY_EVDOA	CDMA2000 EVDO revision A.
CMMODEM_ACCESS_TECHNOLOGY_EVDOB	CDMA2000 EVDO revision B.
CMMODEM_ACCESS_TECHNOLOGY_LTE	LTE (ETSI 27.007: "E-UTRAN")
CMMODEM_ACCESS_TECHNOLOGY_5GNR	5GNR (ETSI 27.007: "NG-RAN"). Since 1.14.
CMMODEM_ACCESS_TECHNOLOGY_ANY	Mask specifying all access technologies.

Enum of ASUS_CMM_FLIGHT_MODE_STATE

Parameter	Description
ASUS_CMM_FLIGHT_MODE_OFF	The flight mode is off.
ASUS_CMM_FLIGHT_MODE_ON	The flight mode is on.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.12 **AsusConnMgrModemSwitchSIM**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemSwitchSIM (
    __IN        uint32_t Index,
    __IN        uint32_t  SimSlotId);
```

Description

Switch the SIM.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().
__IN	SimSlotId	SIM slot Ids (refer to the following ASUS SIM Slot Id Table)

ASUS SIM Slot Id Table

Id	Description	Value
ASUS_CMM_SIM_SLOT_0	SIM Slot 0	0
ASUS_CMM_SIM_SLOT_1	SIM Slot 1	1

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.13 **AsusConnMgrModemUnlockSIMByPIN**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemUnlockSIMByPIN (
    __IN        uint32_t Index,
    __IN        char *pPinCode);
```

Description

Unlock the SIM by PIN code.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().
__IN	pPinCode	Pointer to a buffer containing the PIN code to be transferred.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.7.14 **AsusConnMgrModemSetFlightMode**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemSetFlightMode (
    __IN        uint32_t Index,
    __IN        enum ASUS_CMM_FLIGHT_MODE_STATE FlightMode);
```

Description

Turm on or turn off the flight mode.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().
__IN	FlightMode	Refer to the following enum: ASUS_CMM_FLIGHT_MODE_STATE The value with ASUS_CMM_FLIGHT_MODE_ON means to turn on the flight mode. The value with ASUS_CMM_FLIGHT_MODE_OFF means to turn off.

Enum of ASUS_CMM_FLIGHT_MODE_STATE

Parameter	Description
ASUS_CMM_FLIGHT_MODE_OFF	The flight mode is off.
ASUS_CMM_FLIGHT_MODE_ON	The flight mode is on.

Return Status Code

Return Value	Description

ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.15 **AsusConnMgrModemSetAPN**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemSetAPN (
    __IN        char *pApn);
```

Description

Set the APN to the profile.

Parameters

In/Out	Parameter Name	Description
__IN	pApn	Pointer to a buffer containing the APN to be transferred.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.7.16 **AsusConnMgrModemSetUser**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemSetUser (
    __IN        char *pUser);
```

Description

Set the user name to the profile.

Parameters

In/Out	Parameter Name	Description
__IN	pUser	Pointer to a buffer containing the user name to be transferred.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.17 **AsusConnMgrModemSetPassword**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemSetPassword (
    __IN        char *pPassword);
```

Description

Set the password to the profile.

Parameters

In/Out	Parameter Name	Description
__IN	pPassword	Pointer to a buffer containing the password to be transferred.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.18 **AsusConnMgrModemSetIPType**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemSetIPType (
    __IN        uint32_t  IpType);
```

Description

Set the IP type to the profile.

Parameters

In/Out	Parameter Name	Description
__IN	IpType	The IP type (refer to the following ASUS IP Type Table)

ASUS IP Type Table

Id	Description	Value
ASUS_CMM_IP_TYPE_IPV4	IPv4 method	0
ASUS_CMM_IP_TYPE_IPV6	IPv6 method	1

ASUS_CMM_IP_TYPE_IPV4V6	IPv4/IPv6 method	2

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.19 **AsusConnMgrModemGetProfile**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemGetProfile (
    __OUT        struct ConnMgrModemProfile_s *pProfile);
```

Description

Get the information of the profile.

Parameters

In/Out	Parameter Name	Description
__OUT	pProfile	Pointer to a structure that receives the information of the profile (refer to the following structure: ConnMgrModemProfile_s).

Structure of ConnMgrModemProfile_s

In/Out Parameter Type	Parameter Name	Description
__OUT char *	pApn	Pointer to a buffer that receives the APN name of the profile.
__INOUT uint32_t *	pApnLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pApn parameter. When

		the function returns, this variable contains the size of the data copied to pApn including the terminating null character. If the buffer specified by pApn parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pUser	Pointer to a buffer that receives the user name of the profile.
__INOUT uint32_t *	pUserLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pUser parameter. When the function returns, this variable contains the size of the data copied to pUser including the terminating null character. If the buffer specified by pUser parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pPassword	Pointer to a buffer that receives the password of the profile.
__INOUT uint32_t *	pPasswordLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pPassword parameter. When the function returns, this variable contains the size of the data copied to pPassword including the terminating null character. If the buffer specified by pPassword parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pIpv4Method	Pointer to a buffer that receives the IPv4 configuration method which supports "disabled", "auto", "manual", "link-local", etc.
__INOUT uint32_t *	pIpv4MethodLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pIpv4Method parameter. When the function returns, this variable contains the size of the data copied to pIpv4Method including the terminating null character. If the buffer specified by pIpv4Method parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pIpv6Method	Pointer to a buffer that receives the IPv6 configuration method which supports "disabled", "auto", "manual", "link-local", etc.
__INOUT uint32_t *	pIpv6MethodLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pIpv6Method parameter.

		When the function returns, this variable contains the size of the data copied to pIpv6Method including the terminating null character. If the buffer specified by pIpv6Method parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Stroage buffer overflow was prevented. Read count was larger than the defined buffer length.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.20 **AsusConnMgrModemResetProfile**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemResetProfile ();
```

Description

Set the profile to the default value.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.

ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.21 **AsusConnMgrModemSwitchCarrier**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemSwitchCarrier (
    __IN        uint32_t Index,
    __IN        struct ConnMgrModemCarrierMcc_s *pMcc
    __IN        struct ConnMgrModemCarrierMnc_s *pMnc);
```

Description

Switch to the carrier network with the input of the carrier name.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().
__IN	pMcc	Pointer to a structure containing the data to be transferred (refer to the following structure: ConnMgrModemCarrierMcc_s).
__IN	pMnc	Pointer to a structure containing the data to be transferred (refer to the following structure: ConnMgrModemCarrierMnc_s).

Structure of ConnMgrModemCarrierMcc_s

In/Out Parameter Type	Parameter Name	Description

__IN char *	pMcc	Pointer to a buffer containing the MCC to be transferred.
__IN uint32_t *	pMccLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pMcc parameter.

Structure of ConnMgrModemCarrierMnc_s

In/Out Parameter Type	Parameter Name	Description
__IN char *	pMnc	Pointer to a buffer containing the MNC to be transferred.
__IN uint32_t *	pMncLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pMnc parameter.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.7.22 **AsusConnMgrModemCheckCarrier**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemCheckCarrier (
    __IN        uint32_t Index,
    __OUT       struct ConnMgrModemCarrierInfo_s *pCarrierInfo);
```

Description

Get the information of the carrier including MCC, MNC, and the name of the carrier.

Parameters

In/Out	Parameter Name	Description

__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().
__OUT	pCarrierInfo	Pointer to a structure that receives the information of the carrier (refer to the following structure: ConnMgrModemCarrierInfo_s).

Structure of ConnMgrModemCarrierInfo_s

In/Out Parameter Type	Parameter Name	Description
__OUT struct ConnMgrModemCarrierMcc_s *	pMcc	Pointer to a structure that receives MCC (refer to the following structure: ConnMgrModemCarrierMcc_s).
__OUT struct ConnMgrModemCarrierMnc_s *	pMnc	Pointer to a structure that receives MNC (refer to the following structure: ConnMgrModemCarrierMnc_s).
__OUT struct ConnMgrModemCarrierName_s *	pName	Pointer to a structure that receives the name of the carrier (refer to the following structure: ConnMgrModemCarrierName_s).

Structure of ConnMgrModemCarrierMcc_s

In/Out Parameter Type	Parameter Name	Description
__OUT char *	pMcc	Pointer to a buffer that receives MCC.
__INOUT uint32_t *	pMccLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pMcc parameter. When the function returns, this variable contains the size of the data copied to pMcc including the terminating null character. If the buffer specified by pMcc parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.

Structure of ConnMgrModemCarrierMnc_s

In/Out Parameter Type	Parameter Name	Description
__OUT char *	pMnc	Pointer to a buffer that receives MNC.

__INOUT uint32_t *	pMncLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pMnc parameter. When the function returns, this variable contains the size of the data copied to pMnc including the terminating null character. If the buffer specified by pMnc parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.

Structure of ConnMgrModemCarrierName_s

In/Out Parameter Type	Parameter Name	Description
__OUT char *	pOperatorName	Pointer to a buffer that receives the name of the carrier.
__INOUT uint32_t *	pOperatorNameLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pOperatorName parameter. When the function returns, this variable contains the size of the data copied to pOperatorName including the terminating null character. If the buffer specified by pOperatorName parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Stroage buffer overflow was prevented. Read count was larger than the defined buffer length.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.23 **AsusConnMgrModemGetICCID**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemGetICCID (
    __IN        uint32_t Index,
    __OUT       char *pBuffer,
    __INOUT     uint32_t *pBufLen);
```

Description

Get the Integrate Circuit Card Identity.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().
__OUT	pBuffer	Pointer to a buffer that receives the value's data.
__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. If the buffer specified by pBuffer parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined

	range.
ASUS_API_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Stroage buffer overflow was prevented. Read count was larger than the defined buffer length.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.24 **AsusConnMgrModemGetIMSI**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemGetIMSI (
    __IN        uint32_t Index,
    __OUT       char *pBuffer,
    __INOUT     uint32_t *pBufLen);
```

Description

Get the International Mobile Subscriber Identity.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().
__OUT	pBuffer	Pointer to a buffer that receives the value's data.
__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. If the buffer specified by pBuffer parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Stroage buffer overflow was prevented. Read count was larger than the defined buffer length.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.25 **AsusConnMgrModemGetSignalStrength**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemGetSignalStrength (
    __IN        uint32_t Index,
    __OUT       uint32_t *pValue);
```

Description

Get the percentage of the signal strength.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().
__OUT	pValue	Pointer to a variable that receives the value's data.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.26 **AsusConnMgrModemGetAdvancedSignalInfo**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemGetAdvancedSignalInfo (
    __IN        uint32_t Index,
    __OUT       struct ConnMgrModemAdvSignalInfo_s *pAdvSignalInfo);
```

Description

Get the signal strength of the different measurement.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().
__OUT	pAdvSignalInfo	Pointer to a structure that receives the signal strength of the different measurement (refer to the following structure: ConnMgrModemAdvSignalInfo_s).

Structure of ConnMgrModemAdvSignalInfo_s

Parameter Type	Parameter Name	Description

struct ConnMgrModemEvdoSignalInfo_s	Evdo	A structure that describes the signal strength of the different measurement of EVDO technology (refer to the following structure: ConnMgrModemEvdoSignalInfo_s).
struct ConnMgrModemGsmSignalInfo_s	Gsm	A structure that describes the signal strength of the different measurement of GSM technology (refer to the following structure: ConnMgrModemGsmSignalInfo_s).
struct ConnMgrModemUmtsSignalInfo_s	Umts	A structure that describes the signal strength of the different measurement of UMTS technology (refer to the following structure: ConnMgrModemUmtsSignalInfo_s).
struct ConnMgrModemLteSignalInfo_s	Lte	A structure that describes the signal strength of the different measurement of LTE technology (refer to the following structure: ConnMgrModemLteSignalInfo_s).

Structure of ConnMgrModemEvdoSignalInfo_s

Parameter Type	Parameter Name	Description
float	rssi	The CDMA EV-DO RSSI (Received Signal Strength Indication), in dBm, given as a floating point value.
float	ecio	The CDMA EV-DO Ec/Io, in dBm, given as a floating point value.
float	sinr	CDMA EV-DO SINR level, in dB, given as a floating point value.
float	io	The CDMA EV-DO Io, in dBm, given as a floating point value.
uint32_t	validFlag	Used to specify that the specific signal strength is valid or not. RSSI mapped to Bit 0. Ec/Io mapped to Bit 1. SINR mapped to Bit 3. Io mapped to Bit 4.

Structure of ConnMgrModemGsmSignalInfo_s

Parameter Type	Parameter Name	Description
float	rssi	The GSM RSSI (Received Signal Strength Indication), in

		dBm, given as a floating point value.
uint32_t	validFlag	Used to specify that the specific signal strength is valid or not. RSSI mapped to Bit 0.

Structure of ConnMgrModemUmtsSignalInfo_s

Parameter Type	Parameter Name	Description
float	rssi	The UMTS RSSI (Received Signal Strength Indication), in dBm, given as a floating point value.
float	rscp	The UMTS RSCP (Received Signal Code Power), in dBm, given as a floating point value.
float	ecio	The UMTS Ec/Io, in dB, given as a floating point value.
uint32_t	validFlag	Used to specify that the specific signal strength is valid or not. RSSI mapped to Bit 0. RSCP mapped to Bit 1. Ec/Io mapped to Bit 3.

Structure of ConnMgrModemLteSignalInfo_s

Parameter Type	Parameter Name	Description
float	rssi	The LTE RSSI (Received Signal Strength Indication), in dBm, given as a floating point value.
float	rsrq	The LTE RSRQ (Reference Signal Received Quality), in dB, given as a floating point value.
float	rsrp	The LTE RSRP (Reference Signal Received Power), in dBm, given as a floating point value.
float	snr	The LTE S/R ratio, in dB, given as a floating point value.
uint32_t	validFlag	Used to specify that the specific signal strength is valid or not. RSSI mapped to Bit 0. RSRQ mapped to Bit 1. RSRP mapped to Bit 3. S/R ratio mapped to Bit 4.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.

ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.27 **AsusConnMgrModemGetCellLocationInfo**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrModemGetCellLocationInfo (
    __IN        uint32_t Index,
    __OUT       struct ConnMgrModemCellLocationInfo_s *pCellLocInfo);
```

Description

Get the information of the cell location.

Parameters

In/Out	Parameter Name	Description
__IN	Index	A variable that specifies the index of the modem that can be derived from the API: AsusConnMgrModemGetModemInfo ().
__OUT	pCellLocInfo	Pointer to a structure that receives the information of the cell location (refer to the following structure: ConnMgrModemCellLocationInfo_s).

Structure of ConnMgrModemCellLocationInfo_s

In/Out Parameter Type	Parameter Name	Description
__OUT char *	pOperatorCode	Pointer to a buffer that receives the operator code.
__INOUT uint32_t *	pOperatorCodeLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pOperatorCode parameter. When the function returns, this variable contains the size of the data copied to pOperatorCode including the terminating null character. If the buffer specified by pOperatorCode parameter is not large enough to hold the data,

		the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pOperatorName	Pointer to a buffer that receives the operator name.
__INOUT uint32_t *	pOperatorNameLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pOperatorName parameter. When the function returns, this variable contains the size of the data copied to pOperatorName including the terminating null character. If the buffer specified by pOperatorName parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pLocationAreaCode	Pointer to a buffer that receives the location area code.
__INOUT uint32_t *	pLocationAreaCodeLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pLocationAreaCode parameter. When the function returns, this variable contains the size of the data copied to pLocationAreaCode including the terminating null character. If the buffer specified by pLocationAreaCode parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pTrackingAreaCode	Pointer to a buffer that receives the tracking area code.
__INOUT uint32_t *	pTrackingAreaCodeLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pTrackingAreaCode parameter. When the function returns, this variable contains the size of the data copied to pTrackingAreaCode including the terminating null character. If the buffer specified by pTrackingAreaCode parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.
__OUT char *	pCellId	Pointer to a buffer that receives the cell Id.
__INOUT uint32_t *	pCellIdLen	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pCellId parameter. When the function returns, this

		variable contains the size of the data copied to pCellId including the terminating null character. If the buffer specified by pCellId parameter is not large enough to hold the data, the function returns the value ASUS_API_STATUS_MORE_DATA.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Stroage buffer overflow was prevented. Read count was larger than the defined buffer length.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.28 **AsusConnMgrSetFailover**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrSetFailover (
    __IN        uint32_t Enable);
```

Description

Allow to enable or disable the failover feature.

Parameters

In/Out	Parameter Name	Description

__IN	Enable	The value with ASUS_CM_FAILOVER_ON means to turn on the failover feature. The value with ASUS_CM_FAILOVER_OFF means to turn off.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.29 **AsusConnMgrGetFailoverStatus**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrGetFailoverStatus (
    __OUT        uint32_t *pStatus);
```

Description

Get the status of the failover feature.

Parameters

In/Out	Parameter Name	Description
__OUT	pStatus	Pointer to a variable that receives the value's data. The value with ASUS_CM_FAILOVER_ON means the failover feature is on. The value with ASUS_CM_FAILOVER_OFF means the failover feature is off.

Return Status Code

Return Value	Description

ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.30 **AsusConnMgrSetFailoverGroup**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrSetFailoverGroup (
    __IN        char **pInterface,
    __IN        uint32_t InterfaceCnt);
```

Description

Set the failover group on the failover feature to determine the priority of the network interface.

Parameters

In/Out	Parameter Name	Description
__IN	pInterface	Pointer to a buffer containing the list of the network interface name to be transferred.
__IN	InterfaceCnt	The number of the network interfaces in the group.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined

	range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.7.31 **AsusConnMgrGetFailoverGroup**

```
uint32_t
ASUS_CALLTYPE
AsusConnMgrGetFailoverGroup (
    __OUT       char **pInterface,
    __INOUT     uint32_t *ByteCntOfRow,
    __IN        uint32_t ByteCntOfColumn);
```

Description

Get the failover group of the failover feature. The output of this function is the list of the network interface name.

Parameters

In/Out	Parameter Name	Description
__OUT	pInterface	Pointer to a buffer that receives the list of the network interface name.
__INOUT	ByteCntOfRow	Pointer to a variable that specifies the size, in bytes, of the buffer pointed to by the pInterface parameter. When the function returns, this variable contains the number of the network interfaces. This variable will be zero if there is no group set on the failover feature. If the buffer specified by pInterface parameter is not large enough to hold the number of the network interfaces, the function returns the value ASUS_API_STATUS_MORE_DATA.
__IN	ByteCntOfColumn	The column size, in bytes, of the buffer pointed to by the pInterface parameter. If ByteCntOfColumn parameter is not large enough to hold the size, in bytes, of the return interface name, the function returns the value ASUS_API_STATUS_MORE_DATA.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Stroage buffer overflow was prevented. Read count was larger than the defined buffer length.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the Connectivity Management. The caller should have root permission.

2.8 **LED Control**

2.8.1 **AsusLedGetInfo**

```
uint32_t
ASUS_CALLTYPE
AsusLedGetInfo (
    __OUT       struct LedInfo_s *pLedInfo,
    __INOUT     uint32_t   *pLedCnt);
```

Description

Gets the information of the LEDs.

Parameters

In/Out	Parameter Name	Description
__OUT	pLedInfo	An array of the structures that receives the information of the LEDs (refer to the following structure: LedInfo_s).

__INOUT	pLedCnt	Pointer to a variable that specifies the number of the LedInfo_s structures pointed to by the pLedInfo parameter. When the function returns, this variable contains the number of the LEDs. This variable will be zero if there is no LED support. If the buffer specified by pLedInfo parameter is not large enough to hold the number of the LEDs, the function returns the value ASUS_API_STATUS_MORE_DATA.

Structure of LedInfo_s

Parameter Type	Parameter Name	Description
uint32_t	LedId	LED Id
uint32_t	SupportedColor	Used to specify that the specific color is supported or not by this LED. For instance, Bit 0 corresponds to “Color Blue”, and Bit 1 to “Color Green”. If Bit 0 is set, the LED is able to light up blue. Refer to the following LED Color Table for the detailed color assignment.
uint32_t	SystemOccupied	Used to specify whether this LED is occupied by the system or not. The value with LED_OCCUPIED_BY_SYSTEM means the LED is only controlled by the system. The value with LED_OCCUPIED_BY_USER means users or applications have the permission to control the LED.

LED Color Table

Id	Value	Description
LED_LIGHT_COLOR_BLUE	0x00000001	“Color Blue” mapped to Bit 0 of the SupportedColor parameter.
LED_LIGHT_COLOR_GREEN	0x00000002	“Color Green” mapped to Bit 1 of the SupportedColor parameter.
LED_LIGHT_COLOR_RED	0x00000004	“Color Red” mapped to Bit 2 of the SupportedColor parameter.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.

ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supported.
ASUS_API_STATUS_MORE_DATA	The amount of available data exceeds the buffer size. Storage buffer overflow was prevented. Read count was larger than the defined buffer length.

2.8.2 AsusLedGetNumberofLeds

```
uint32_t
ASUS_CALLTYPE
AsusLedGetNumberofLeds (
    __OUT          uint32_t *pValue);
```

Description

Get the number of LEDs that can be controlled.

Parameters

In/Out	Parameter Name	Description
__OUT	pValue	Pointer to a variable that specifies the number of the LEDs.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.8.3 AsusLedTurnOn

```
uint32_t
ASUS_CALLTYPE
AsusLedTurnOn (
    __IN        uint32_t LedId,
    __IN        uint32_t Color);
```

Description

Turn on the LED with the specific color.

Parameters

In/Out	Parameter Name	Description
__IN	LedId	LED Id
__IN	Color	The value to specify that the specific color (refer to the following LED Color Table for the detailed color assignment).

LED Color Table

Id	Value	Description
LED_LIGHT_COLOR_BLUE	0x00000001	Color Blue
LED_LIGHT_COLOR_GREEN	0x00000002	Color Green
LED_LIGHT_COLOR_RED	0x00000004	Color Red

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the LED. Please use AsusLedSetSystemOccupied() with the parameter: LED_OCCUPIED_BY_USER to allow the system not to occupy the

	LED.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.8.4 AsusLedTurnOff

```
uint32_t
ASUS_CALLTYPE
AsusLedTurnOff (
    __IN        uint32_t LedId);
```

Description

Turn off the LED.

Parameters

In/Out	Parameter Name	Description
__IN	LedId	LED Id

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_INVALID_PERMISSION_ACCESS	Invalid permission access to the LED. Please use AsusLedSetSystemOccupied() with the parameter: LED_OCCUPIED_BY_USER to allow the system not to occupy the LED.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.8.5 AsusLedSetSystemOccupied

```
uint32_t
ASUS_CALLTYPE
AsusLedSetSystemOccupied (
    __IN        uint32_t LedId,
    __IN        uint32_t SystemOccupied);
```

Description

Allow the system to occupy the control permission of the LED or not.

Parameters

In/Out	Parameter Name	Description
__IN	LedId	LED Id
__IN	SystemOccupied	Used to specify whether this LED is occupied by the system or not. The value with LED_OCCUPIED_BY_SYSTEM means the LED will be only controlled by the system. The value with LED_OCCUPIED_BY_USER means users or applications will have the permission to control the LED.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9 Vehicle Co-Processors

2.9.1 AsusVcpRtcTimeSet

```
uint32_t
ASUS_CALLTYPE
AsusVcpRtcTimeSet (
    __IN        uint32_t  Year,
    __IN        uint32_t  Month,
    __IN        uint32_t  DayofMonth,
    __IN        uint32_t  Hour,
    __IN        uint32_t  Minute,
    __IN        uint32_t  Second,
    __IN        uint32_t  DayofWeek);
```

Description

Sets time to RTC.

Parameters

In/Out	Parameter Name	Description
__IN	Year	The year.
__IN	Month	Selects the month, such as 1, 2, …, 11, and 12.
__IN	DayofMonth	Selects the day of month, such as 1, 2, …, 30, and 31.
__IN	Hour	Selects the hour, such as 0, 1, …, 22, and 23.
__IN	Minute	Selects the minute, such as 0, 1, …, 58, and 59.
__IN	Second	Selects the second, such as 0, 1, …, 58, and 59.
__IN	DayofWeek	Selects the day of week, such as 1, 2, …, 6, and 7.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supported.

2.9.2 AsusVcpRtcTimeGet

```
uint32_t
ASUS_CALLTYPE
AsusVcpRtcTimeGet (
    __OUT       uint32_t  *pYear,
    __OUT       uint32_t  *pMonth,
    __OUT       uint32_t  *pDayofMonth,
    __OUT       uint32_t  *pHour,
    __OUT       uint32_t  *pMinute,
    __OUT       uint32_t  *pSecond,
    __OUT       uint32_t  *pDayofWeek);
```

Description

Gets time from RTC.

Parameters

In/Out	Parameter Name	Description
__OUT	pYear	Pointer to a buffer that receives the year.
__OUT	pMonth	Pointer to a buffer that receives the month.
__OUT	pDayofMonth	Pointer to a buffer that receives the day of month.
__OUT	pHour	Pointer to a buffer that receives the hour.
__OUT	pMinute	Pointer to a buffer that receives the minute.
__OUT	pSecond	Pointer to a buffer that receives the second.
__OUT	pDayofWeek	Pointer to a buffer that receives the day of week.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supported.

2.9.3 AsusVcpComPortInit

```
uint32_t
ASUS_CALLTYPE
AsusVcpComPortInit (
    __IN        uint32_t Id,
    __IN        enum ASUS_COM_PORT_MODE Mode,
    __IN        enum ASUS_COM_PORT_BAUD_RATE BaudRate,
    __IN        enum ASUS_COM_PORT_PARITY Parity,
    __IN        enum ASUS_COM_PORT_DATA_WIDTH DataWidth,
    __IN        enum ASUS_COM_PORT_STOP_BIT StopBit,
    __IN        enum ASUS_COM_PORT_FLOW_CONTROL FlowControl);
```

Description

Initializes the com port with the settings.

Parameters

In/Out	Parameter Name	Description
__IN	Id	The com port Id.
__IN	Mode	The com port mode. Refer to the following enum: ASUS_COM_PORT_MODE.
__IN	BaudRate	The baud rate of the com port. Refer to the following enum: ASUS_COM_PORT_BAUD_RATE.
__IN	Parity	The parity of the com port. Refer to the following enum: ASUS_COM_PORT_PARITY.
__IN	DataWidth	The data width of the com port. Refer to the following enum: ASUS_COM_PORT_DATA_WIDTH.
__IN	StopBit	The stop bit of the com port. Refer to the following enum: ASUS_COM_PORT_STOP_BIT.
__IN	FlowControl	The flow control of the com port. Refer to the following enum: ASUS_COM_PORT_FLOW_CONTROL.

Enum of ASUS_COM_PORT_MODE

Parameter	Description
ASUS_COM_PORT_MODE_RS232	RS232
ASUS_COM_PORT_MODE_RS422	RS422
ASUS_COM_PORT_MODE_RS485	RS485

Enum of ASUS_COM_PORT_BAUD_RATE

Parameter	Description
ASUS_COM_PORT_BAUD_RATE_2400	A maximum of 2400 bits per second
ASUS_COM_PORT_BAUD_RATE_4800	A maximum of 4800 bits per second
ASUS_COM_PORT_BAUD_RATE_9600	A maximum of 9600 bits per second
ASUS_COM_PORT_BAUD_RATE_19200	A maximum of 19200 bits per second
ASUS_COM_PORT_BAUD_RATE_38400	A maximum of 38400 bits per second
ASUS_COM_PORT_BAUD_RATE_57600	A maximum of 57600 bits per second
ASUS_COM_PORT_BAUD_RATE_115200	A maximum of 115200 bits per second

Enum of ASUS_COM_PORT_PARITY

Parameter	Description
ASUS_COM_PORT_PARITY_NONE	no parity
ASUS_COM_PORT_PARITY_EVEN	even parity
ASUS_COM_PORT_PARITY_ODD	odd parity

Enum of ASUS_COM_PORT_DATA_WIDTH

Parameter	Description
ASUS_COM_PORT_DATA_7_BIT	7 data bits
ASUS_COM_PORT_DATA_8_BIT	8 data bits
ASUS_COM_PORT_DATA_9_BIT	9 data bits

Enum of ASUS_COM_PORT_STOP_BIT

Parameter	Description
ASUS_COM_PORT_STOP_0_5_BIT	0.5 stop bits
ASUS_COM_PORT_STOP_1_BIT	1 stop bits
ASUS_COM_PORT_STOP_1_5_BIT	1.5 stop bits
ASUS_COM_PORT_STOP_2_BIT	2 stop bits

Enum of ASUS_COM_PORT_FLOW_CONTROL

Parameter	Description
ASUS_COM_PORT_FLOW_CONTROL_DISABLED	Flow control on
ASUS_COM_PORT_FLOW_CONTROL_ENABLED	Flow control off

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.4 AsusVcpComPortEnable

```
uint32_t
ASUS_CALLTYPE
AsusVcpComPortEnable (
    __IN        uint32_t Id);
```

Description

Enables the com port.

Parameters

In/Out	Parameter Name	Description
__IN	Id	The com port Id.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.5 AsusVcpComPortDisable

```
uint32_t
ASUS_CALLTYPE
AsusVcpComPortDisable (
    __IN        uint32_t Id);
```

Description

Disables the com port.

Parameters

In/Out	Parameter Name	Description
__IN	Id	The com port Id.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.6 AsusVcpComPortRead

```
uint32_t
ASUS_CALLTYPE
AsusVcpComPortRead (
    __IN        uint32_t Id,
    __OUT       struct VcpComPortReadBuffer_s *pReadBuffer);
```

Description

Reads from the selected com port.

Parameters

In/Out	Parameter Name	Description

__IN	Id	The com port Id
__OUT	pReadBuffer	Pointer to a structure that receives the read data (refer to the following structure: VcpComPortReadBuffer_s).

Structure of VcpComPortReadBuffer_s

Parameter Type	Parameter Name	Description
char[] (The array size is defined by ASUS_VCP_COM_PORT_READ_BUFFER_LENGTH)	pBuffer	The read data

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.7 AsusVcpComPortWrite

```
uint32_t
ASUS_CALLTYPE
AsusVcpComPortWrite (
    __IN        uint32_t Id,
    __IN        struct VcpComPortWriteBuffer_s *pWriteBuffer);
```

Description

Writes to the selected com port.

Parameters

In/Out	Parameter Name	Description
__IN	Id	The com port Id
__IN	pWriteBuffer	Pointer to a structure containing the data to be written (refer to the following structure: VcpComPortWriteBuffer_s).

Structure of VcpComPortWriteBuffer_s

In/Out Parameter Type	Parameter Name	Description
__IN char[]	pBuffer	Pointer to a buffer containing the data to be written.
__IN uint32_t	ByteCnt	Size, in bytes, of data to be written

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.8 AsusVcpWakeupEnable

```
uint32_t
ASUS_CALLTYPE
AsusVcpWakeupEnable (
    __IN        enum VCP_WAKEUP_TYPE WakeupType);
```

Description

Enables the function of the selected wakeup type.

Parameters

In/Out	Parameter Name	Description
__IN	WakeupType	The type of wakeup. Refer to the following enum: VCP_WAKEUP_TYPE.

Enum of VCP_WAKEUP_TYPE

Parameter	Value	Description
VCP_WAKEUP_TYPE_DI0	0x00000001	The function to wakeup device form digital input 0
VCP_WAKEUP_TYPE_DI1	0x00000002	The function to wakeup device form digital

		input 1
VCP_WAKEUP_TYPE_DI2	0x00000004	The function to wakeup device form digital input 2
VCP_WAKEUP_TYPE_DI3	0x00000008	The function to wakeup device form digital input 3
VCP_WAKEUP_TYPE_IGNITION_OFF	0x00000010	The function to shutdown device if the ignition is switched off
VCP_WAKEUP_TYPE_GSENSOR	0x00000040	The function to wakeup device form GSensor
VCP_WAKEUP_TYPE_LTE	0x00000080	The function to wakeup device form LTE

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.9 AsusVcpWakeupDisable

```
uint32_t
ASUS_CALLTYPE
AsusVcpWakeupDisable (
    __IN        enum VCP_WAKEUP_TYPE WakeupType);
```

Description

Disables the function of the selected wakeup type.

Parameters

In/Out	Parameter Name	Description
__IN	WakeupType	The type of wakeup. Refer to the following enum: VCP_WAKEUP_TYPE.

Enum of VCP_WAKEUP_TYPE

Parameter	Value	Description
VCP_WAKEUP_TYPE_DI0	0x00000001	The function to wakeup device form digital input 0
VCP_WAKEUP_TYPE_DI1	0x00000002	The function to wakeup device form digital input 1
VCP_WAKEUP_TYPE_DI2	0x00000004	The function to wakeup device form digital input 2
VCP_WAKEUP_TYPE_DI3	0x00000008	The function to wakeup device form digital input 3
VCP_WAKEUP_TYPE_IGNITION_OFF	0x00000010	The function to shutdown device if the ignition is switched off
VCP_WAKEUP_TYPE_GSENSOR	0x00000040	The function to wakeup device form GSensor
VCP_WAKEUP_TYPE_LTE	0x00000080	The function to wakeup device form LTE

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.10 AsusVcpGetWakeupStatus

```
uint32_t
ASUS_CALLTYPE
AsusVcpGetWakeupStatus (
    __OUT       uint32_t *pWakeupType);
```

Description

Gets the status of each wakeup type.

Bit with the value 1 means Enable; otherwise 0 means Disable.

Parameters

In/Out	Parameter Name	Description
__OUT	pWakeupType	Pointer to a variable that specifies whether wakeup types are enabled or disabled. Each bit of this variable corresponds to a specific wakeup type. Refer to the enum of VCP_WAKEUP_TYPE and the following table: Condition Table of pWakeupType.

Condition Table of pWakeupType

Condition	Return Value	Description
(VCP_WAKEUP_TYPE_DI0 & (*pWakeupType))	0x01	Wakeup type of digital input 0 is enabled
(VCP_WAKEUP_TYPE_DI1 & (*pWakeupType))	0x02	Wakeup type of digital input 1 is enabled
(VCP_WAKEUP_TYPE_DI2 & (*pWakeupType))	0x04	Wakeup type of digital input 2 is enabled
(VCP_WAKEUP_TYPE_DI3 & (*pWakeupType))	0x08	Wakeup type of digital input 3 is enabled
(VCP_WAKEUP_TYPE_IGNITION_OFF & (*pWakeupType))	0x10	If the ignition is switched off, the device will be shutdown.
(VCP_WAKEUP_TYPE_IGNITION_ON & (*pWakeupType))	0x20	Wakeup type of ignition off to on is enabled
(VCP_WAKEUP_TYPE_GSENSOR & (*pWakeupType))	0x40	Wakeup type of gsensor is enabled
(VCP_WAKEUP_TYPE_LTE & (*pWakeupType))	0x80	Wakeup type of LTE is enabled
(VCP_WAKEUP_TYPE_PMIC & (*pWakeupType))	0x100	Wakeup type of PMIC is enabled

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.11 **AsusVcpGetLastWakeupSource**

```
uint32_t
ASUS_CALLTYPE
AsusVcpGetLastWakeupSource (
    __OUT        uint32_t *pWakeupType);
```

Description

Gets the last wakeup source.

Parameters

In/Out	Parameter Name	Description
__OUT	pWakeupType	Pointer to a variable that specifies the last wakeup source. Each bit of this variable corresponds to a specific wakeup type. Refer to the enum of VCP_WAKEUP_TYPE and the following table: Condition Table of pWakeupType.

Condition Table of pWakeupType

Condition	Return Value	Description
(VCP_WAKEUP_TYPE_DI0 & (*pWakeupType))	0x01	Wakeup type of digital input 0 is the last wakeup source
(VCP_WAKEUP_TYPE_DI1 & (*pWakeupType))	0x02	Wakeup type of digital input 1 is the last wakeup source
(VCP_WAKEUP_TYPE_DI2 & (*pWakeupType))	0x04	Wakeup type of digital input 2 is the last wakeup source
(VCP_WAKEUP_TYPE_DI3 & (*pWakeupType))	0x08	Wakeup type of digital input 3 is the last wakeup source
(VCP_WAKEUP_TYPE_IGNITION_ON & (*pWakeupType))	0x20	Wakeup type of ignition off to on is the last wakeup source
(VCP_WAKEUP_TYPE_GSENSOR & (*pWakeupType))	0x40	Wakeup type of gsensor is the last wakeup source
(VCP_WAKEUP_TYPE_LTE & (*pWakeupType))	0x80	Wakeup type of LTE is the last wakeup source
(VCP_WAKEUP_TYPE_PMIC & (*pWakeupType))	0x100	Wakeup type of PMIC is the last wakeup source

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.12 Vehicle Co-Processors Power Management (VPM)

The feature of Vehicle Co-Processors Power Management (VPM) is provided for users to fulfill the special requirements for in-vehicle applications.

2.9.12.1 AsusVcpPowerMgmtSetPowerOnEventDelay

```
uint32_t
ASUS_CALLTYPE
AsusVcpPowerMgmtSetPowerOnEventDelay (
    __IN        uint32_t   Second);
```

Description

Sets the delay time of the power on event.

When ignition status is switched from OFF to ON, the VPM will count “Power ON delay” period to wait signal become stable. After count, the device will be powered on.
If the ignition signal unstable the conditions (e.g switched from ON to OFF) during the “Power ON delay”, the device will not be booted.
The “Power ON delay” can be adjusted by AsusVcpPowerMgmtSetPowerOnEventDelay(seconds) and can be gotten by AsusVcpPowerMgmtGetPowerOnEventDelay(seconds).

Parameters

In/Out	Parameter Name	Description
__IN	Second	The delay time. The unit is second.

Return Status Code

Return Value	Description

ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.12.2 **AsusVcpPowerMgmtGetPowerOnEventDelay**

```
uint32_t
ASUS_CALLTYPE
AsusVcpPowerMgmtGetPowerOnEventDelay (
    __OUT        uint32_t   *pSecond);
```

Description

Gets the delay time of the power on event.

When ignition status is switched from OFF to ON, the VPM will count “Power ON delay” period to wait signal become stable. After count, the device will be powered on.
If the ignition signal unstable the conditions (e.g switched from ON to OFF) during the “Power ON delay”, the device will not be booted.
The “Power ON delay” can be adjusted by AsusVcpPowerMgmtSetPowerOnEventDelay(seconds) and can be gotten by AsusVcpPowerMgmtGetPowerOnEventDelay(seconds).

Parameters

In/Out	Parameter Name	Description
__OUT	pSecond	Pointer to a variable that specifies the delay time. The unit is second.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.

ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.12.3 **AsusVcpPowerMgmtSetPowerOffEventDelay**

```
uint32_t
ASUS_CALLTYPE
AsusVcpPowerMgmtSetPowerOffEventDelay (
    __IN            uint32_t   Second);
```

Description

Sets the delay time of the power off event.

When ignition status is switched from ON to OFF, the VPM will count “Power OFF delay”. After the “Power OFF delay” period expired, the system power will be cut off abruptly. Application programs could watch this event to do pre-defined tasks, like storing data and preparing to turn off the system. The “Power OFF delay” can be adjusted by AsusVcpPowerMgmtSetPoweOffEventDelay(seconds) and can be gotten by AsusVcpPowerMgmtGetPowerOffEventDelay(seconds).

Parameters

In/Out	Parameter Name	Description
__IN	Second	The delay time. The unit is second.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.12.4 **AsusVcpPowerMgmtGetPowerOffEventDelay**

```
uint32_t
ASUS_CALLTYPE
AsusVcpPowerMgmtGetPowerOffEventDelay (
    __OUT        uint32_t   *pSecond);
```

Description

Gets the delay time of the power off event.

When ignition status is switched from ON to OFF, the VPM will count “Power OFF delay”. After the “Power OFF delay” period expired, the system power will be cut off abruptly. Application programs could watch this event to do pre-defined tasks, like storing data and preparing to turn off the system. The “Power OFF delay” can be adjusted by AsusVcpPowerMgmtSetPoweOffEventDelay(seconds) and can be gotten by AsusVcpPowerMgmtGetPowerOffEventDelay(seconds).

Parameters

In/Out	Parameter Name	Description
__OUT	pSecond	Pointer to a variable that specifies the delay time. The unit is second.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.12.5 **AsusVcpPowerMgmtSetLowBatteryProtectionPreBootThreshold**

```
uint32_t
ASUS_CALLTYPE
AsusVcpPowerMgmtSetLowBatteryProtectionPreBootThreshold (
    __IN        float Voltage);
```

Description

VPM provides low battery protection (LBP) functions which monitor whether the voltage of battery is above specified threshold and control the power of the device.

VPM is able to do the battery voltage check before the system boot up and decide whether to boot up the device. If the battery voltage is above the specified threshold, VCM allows the system to boot up; otherwise, the system will not boot up.

This function is used to set the threshold.

Parameters

In/Out	Parameter Name	Description
__IN	Voltage	The threshold. The unit is voltage.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.12.6 **AsusVcpPowerMgmtGetLowBatteryProtectionPreBootThreshold**

```
uint32_t
ASUS_CALLTYPE
AsusVcpPowerMgmtGetLowBatteryProtectionPreBootThreshold (
    __OUT        float *pVoltage);
```

Description

VPM provides low battery protection (LBP) functions which monitor whether the voltage of battery is above specified threshold and control the power of the device.
VPM is able to do the battery voltage check before the system boot up and decide whether to boot up the device. If the battery voltage is above the specified threshold, VCM allows the system to boot up; otherwise, the system will not boot up.

This function is used to get the threshold.

Parameters

In/Out	Parameter Name	Description
__OUT	pVoltage	Pointer to a variable that specifies the threshold. The unit is voltage.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.13 **AsusVcpGSensorEnable**

```
uint32_t
ASUS_CALLTYPE
AsusVcpGSensorEnable ();
```

Description

Enables G-Sensor function.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.14 **AsusVcpGSensorDisable**

```
uint32_t
ASUS_CALLTYPE
AsusVcpGSensorDisable ();
```

Description

Disables G-Sensor function.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.15 **AsusVcpGSensorAvailable**

```
uint32_t
ASUS_CALLTYPE
AsusVcpGSensorAvailable (
    __OUT       char *pSupported);
```

Description

Gets supported status of G-Sensor.

Parameters

In/Out	Parameter Name	Description
__OUT	pSupported	Pointer to a variable that specifies the supported status of G-Sensor. The variable with the value 1 means the platform supports G-Sensor. Otherwise, the platform does not support G-Sensor.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.16 **AsusVcpGSensorGetStatus**

```
uint32_t
ASUS_CALLTYPE
AsusVcpGSensorGetStatus (
    __OUT       char *pStatus);
```

Description

Gets status of G-Sensor.

Parameters

In/Out	Parameter Name	Description

__OUT	pStatus	Pointer to a variable that specifies the status of G-Sensor. The variable with the value 1 means G-Sensor is enabled. Otherwise, G-Sensor is disabled.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.17 **AsusVcpGSensorRead**

```
uint32_t
ASUS_CALLTYPE
AsusVcpGSensorRead (
    __OUT          struct VcpGSensorValue_s *pGsensorValue);
```

Description

Gets the value of gravity acceleration from G-Sensor.

Parameters

In/Out	Parameter Name	Description
__OUT	pGsensorValue	Pointer to a structure that receives the value of gravity acceleration from G-Sensor (refer to the following structure: VcpGSensorValue_s).

Structure of VcpGSensorValue_s

Parameter Type	Parameter Name	Description
int	x_mg	The x-axis acceleration of gravity value. The unit is mg.
int	y_mg	The y-axis acceleration of gravity value. The unit is mg.
int	z_mg	The z-axis acceleration of gravity value. The unit is mg.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.18 **AsusVcpGSensorGetResolution**

```
uint32_t
ASUS_CALLTYPE
AsusVcpGSensorGetResolution (
    __OUT        enum VCP_GSENSOR_RES *pGsensorRes);
```

Description

Gets the resolution of G-Sensor.

Parameters

In/Out	Parameter Name	Description
__OUT	pGsensorRes	Pointer to a variable that receives the resolution of G-Sensor. Refer to the following enum: VCP_GSENSOR_RES.

Enum of VCP_GSENSOR_RES

Parameter	Description
VCP_GSENSOR_RES_2G	The G-Sensor resolution of +2g/-2g
VCP_GSENSOR_RES_4G	The G-Sensor resolution of +4g/-4g
VCP_GSENSOR_RES_8G	The G-Sensor resolution of +8g/-8g
VCP_GSENSOR_RES_16G	The G-Sensor resolution of +16g/-16g

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details

	are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.19 **AsusVcpGSensorSetResolution**

```
uint32_t
ASUS_CALLTYPE
AsusVcpGSensorSetResolution (
    __IN        enum VCP_GSENSOR_RES GsensorRes);
```

Description

Sets the resolution of G-Sensor.

Parameters

In/Out	Parameter Name	Description
__IN	GsensorRes	The resolution of G-Sensor. Refer to the following enum: VCP_GSENSOR_RES.

Enum of VCP_GSENSOR_RES

Parameter	Description
VCP_GSENSOR_RES_2G	The G-Sensor resolution of +2g/-2g
VCP_GSENSOR_RES_4G	The G-Sensor resolution of +4g/-4g
VCP_GSENSOR_RES_8G	The G-Sensor resolution of +8g/-8g
VCP_GSENSOR_RES_16G	The G-Sensor resolution of +16g/-16g

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.

ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.20 **AsusVcpGSensorGetOffset**

```
uint32_t
ASUS_CALLTYPE
AsusVcpGSensorGetOffset (
    __OUT          struct VcpGSensorValue_s *pGsensorValue);
```

Description

Gets the offset value of G-Sensor axis.

Parameters

In/Out	Parameter Name	Description
__OUT	pGsensorValue	Pointer to a structure that receives the offset value of G-Sensor axis (refer to the following structure: VcpGSensorValue_s).

Structure of VcpGSensorValue_s

Parameter Type	Parameter Name	Description
int	x_mg	The offset value of x-axis. The unit is mg.
int	y_mg	The offset value of y-axis. The unit is mg.
int	z_mg	The offset value of z-axis. The unit is mg.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.

2.9.21 **AsusVcpGSensorSetOffset**

```
uint32_t
ASUS_CALLTYPE
AsusVcpGSensorSetOffset (
    __IN        struct VcpGSensorValue_s GsensorValue);
```

Description

Sets the offset value of G-Sensor axis.

Parameters

In/Out	Parameter Name	Description
__IN	GsensorValue	The offset value of G-Sensor axis (refer to the following structure: VcpGSensorValue_s).

Structure of VcpGSensorValue_s

Parameter Type	Parameter Name	Description
int	x_mg	The offset value of x-axis. The unit is mg.
int	y_mg	The offset value of y-axis. The unit is mg.
int	z_mg	The offset value of z-axis. The unit is mg.

Return Status Code

Return Value	Description
ASUS_API_STATUS_SUCCESS	The operation was successful.
ASUS_API_STATUS_ERROR	Generic error message. No further error details are available.
ASUS_API_STATUS_NOT_INITIALIZED	Library uninitialized.
ASUS_API_STATUS_INVALID_PARAMETER	One or more of the API function call parameters are out of the defined range.
ASUS_API_STATUS_UNSUPPORTED	This function or Id is not supportted.