MRM IVCP User Manual

5/4/2017
SSID.SRP

Table of contents

1 Introduction

1.1 IVCP SDK Overview

MRM IVCP SDK is a set of libraries for controlling IVCP (Intelligent Vehicle Co-Processor).

MRM IVCP SDK is composed of the following API modules:

***NOTE: Not all API modules are available on all device platforms. Please refer to the hardware specification for corresponding functions.**

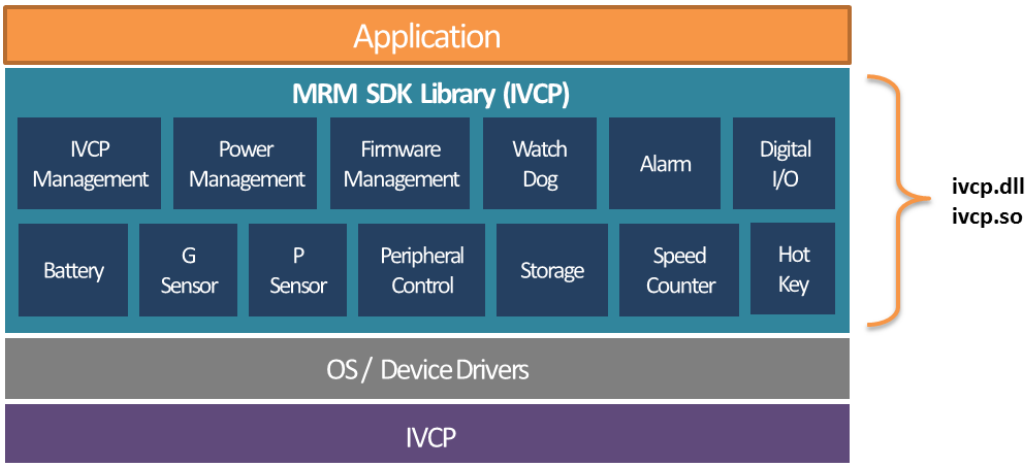
Module	Feature
IVCP Management	The basic functions(ex:initialization, deinitialization) of IVCP SDK. Please also refer to the **IVCP convention** section.
Firmware Management	Save/load the default configurations of IVCP functions.
Power Management	Power management functions such as boot control, Ignition control, event delay adjustments and low voltage protection.
Watchdog	A watch dog timer mechanism which guarantees to shutdown the device in specific expiry time.
Alarm	Device alarm wakeup functions. You can set time to IVCP and set the rules to wakeup device at specific time.
Digital IO Control	Get/set status of Digital input/output pins.
Battery	Backup battery status monitoring,
G-Sensor	Gravity sensor functions. You can get data from gravity sensor and set gravity sensor as device wakeup source with specified threshold.
P-Sensor	Pressure sensor functions. You can get data from pressure sensor.
Sensor	System sensor functions. You can get data from internal system sensor.
Peripheral Control	Control power an relative functions of various peripheral devices though IVCP. (ex: WWAN module, WIFI module, GPS, Rear view camera.)

Storage	Functions for access the permanent storage on device. You can store arbitrary data to the storage and the data will not be cleared due to the hard disk format or crash.
Speed Counter	Functions for monitoring rotation of tire.
Hotkey	Get/set keycode of each on-device hardware hotkey. Get/set the LED brightness of the hotkeys.

MRM IVCP SDK supports the following platforms:

For Windows / Linux:

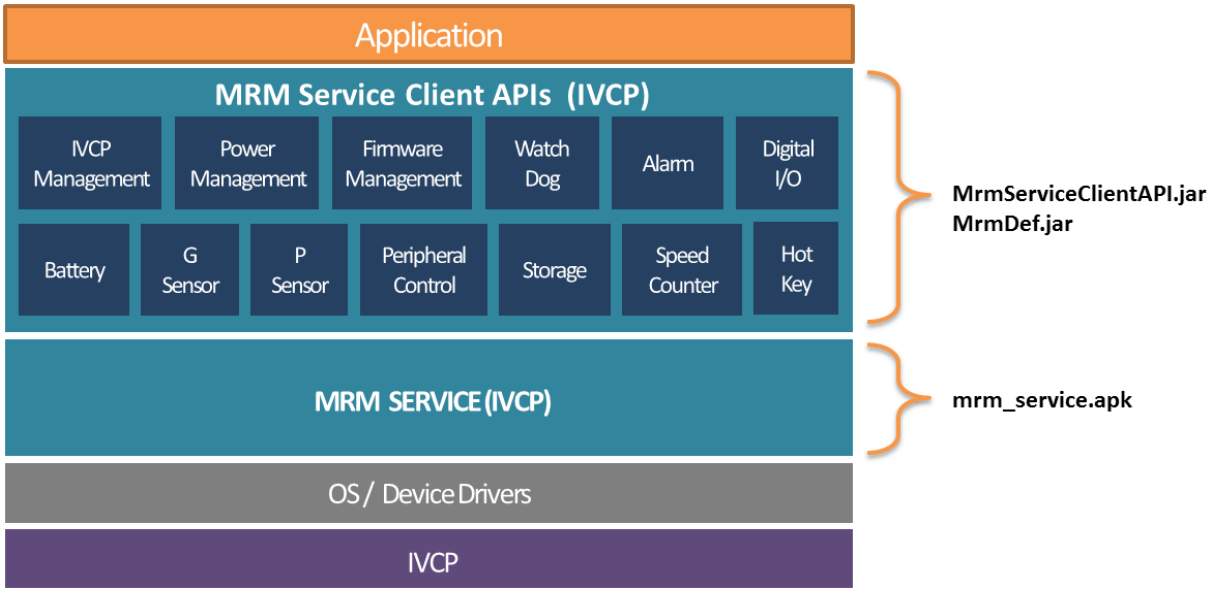
The MRM IVCP SDK is designed as a library (**ivcp.dll** / **ivcp.so**) for the customer's APP to load and access.



Prior to use IVCP APIs, you should call i**vcp_init()**. Normally the API should return 32-bits error code **MRM_ERR_NO_ERROR(0).** You must confirm the return value to ensure that the command is work. After calling **ivcp_init()**, the library will initialize IVCP and start to accept command.

For Android:

The MRM IVCP API is designed as a background service which acts as a proxy for client APPs to access the IVCP functions.



The MRM IVCP API for Android includes three parts:

- MRM Services (**mrm_service.apk**)
- MRM Service Client APIs (**MrmServiceClientAPI.jar**)
- MRM Data Classes And Constants Definitions (**MrmDef.jar**)

To use the IVCP service, you must install MRM Services APK to your device and import the above jar libraries in your APP project.
In your APP, you must call **ivcp_bind_service()** to connect your APP process with the IVCP Service before accessing the IVCP APIs and call **ivcp_unbind_service()** to disconnect service before your APP is destroyed.

1.2 System Requirements

Hardware:

- IVCP Module

Operating system:

- Windows
- Linux
- Android

Recommended Development Tool

- Visual Studio 2008 or above
- GCC 4.6+
- Android Studio 1.3.2+

2 Using IVCP

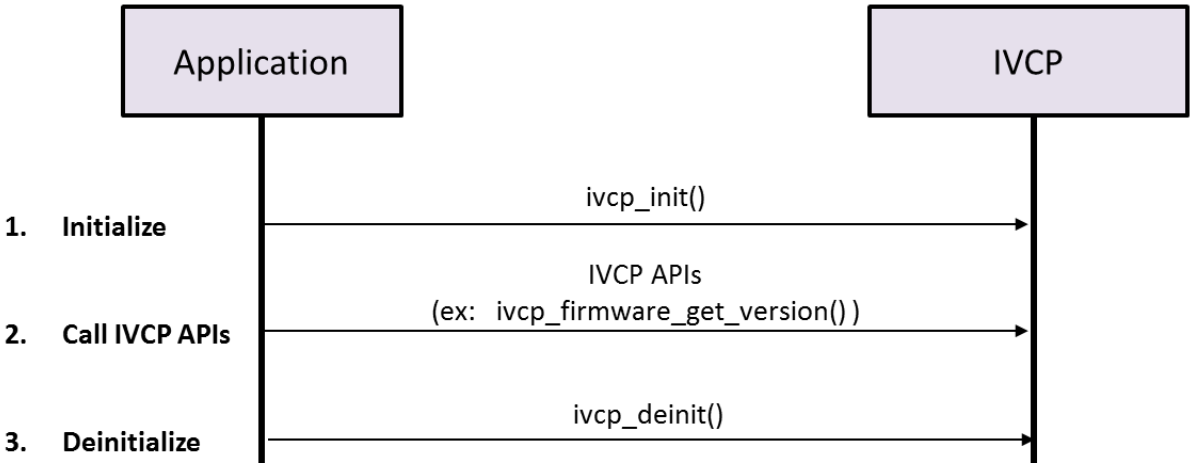
2.1 IVCP Conventions

2.1.1 Windows/Linux

- **Basic API usage**
  - Each IVCP API has a prefix "**ivcp**_" immediately followed by the function name and the operation name.
  - To usage the IVCP APIs, you must first initialize the library and deinitialize before your APP is closed.

    The flow is described as following:

    1. You must call **ivcp_init()** before using the other IVCP APIs.
    2. Call IVCP APIs.
    3. You must call **ivcp_deinit()** before you APP closed.



  - Any read type API using the pass by pointer. You should ensure the pointer be not released during the API processing and the pointer is valid.
  - You should always check the return value equal MRM_ERR_NO_ERROR(0), in order to ensure the command working.

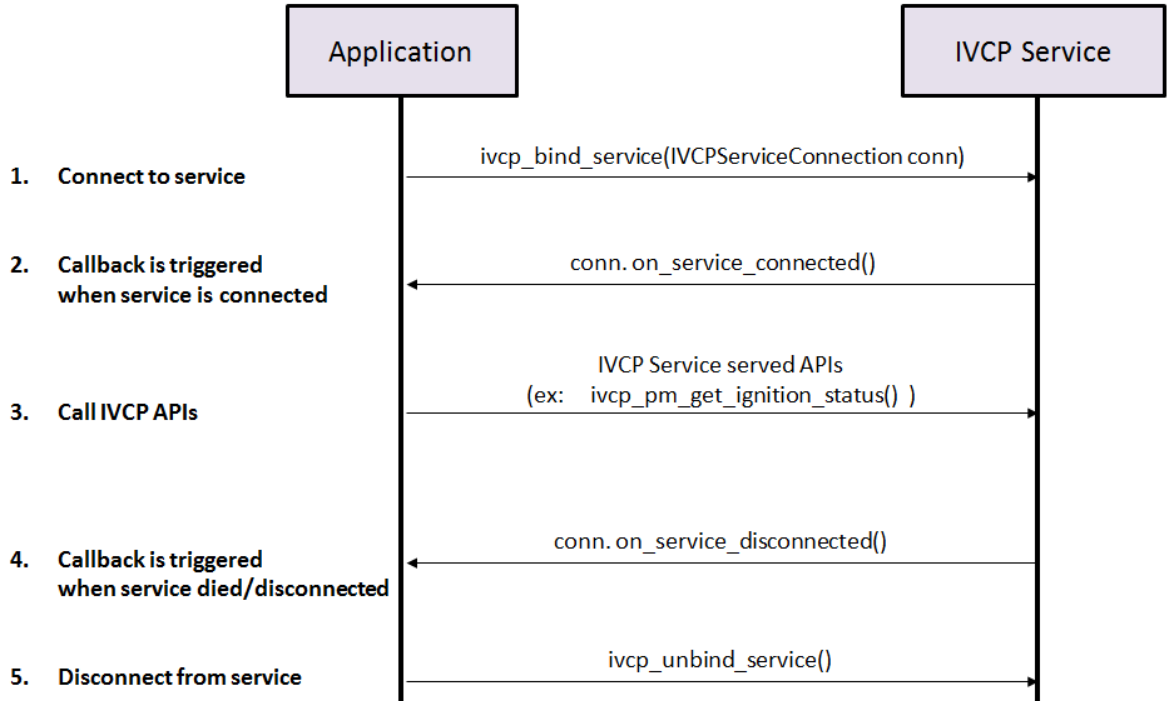
2.1.2 Android

- **SDK Namespaces**
  - **mrm.client.IVCPServiceClient**
    - The main class of MRM Service Client API. Use this class to access IVCP features.
  - **mrm.client.IVCPServiceConnection**
    - The service connection interface used by **ivcp_bind_service**().
  - **mrm.define.IVCP**
    - The definition of data structures used by MRM Service Client API.
  - **mrm.define.MRM_ENUM**
    - The definition of enumeration values used by MRM Service Client API.
  - **define.MRM_CONSTANTS**
    - The definition of constants used by MRM Service Client API.

- **Basic API usage**
  - Each IVCP API has a prefix "**ivcp**_" immediately followed by the function name and the operation name
  - You must create a instance of mrm.client.**IVCPServiceClient** to usage IVCP APIs .
  - To usage the IVCP APIs, you must first "bind" your APP's process to the IVCP service and "unbind" before your APP is closed.

    The flow is described as following:
    1. Create an instance of mrm.client.**IVCPServiceConnection** and implement the interface **on_service_connected()** and **on_service_disconnected()**,
       The interface is used to will inform your APP when the service is connected/disconnected. Please refer to the document of ivcp_bind_service() and IVCP sample code for further details.
    2. Call **ivcp_bind_service()** with created **IVCPServiceConnection** instance to connect your APP process to the IVCP Service.
    3. Call IVCP Served APIs
    4. Call **ivcp_unbind_service()** before your APP's application context is destroyed.

    NOTE: Please refer to the IVCP sample code for the details.

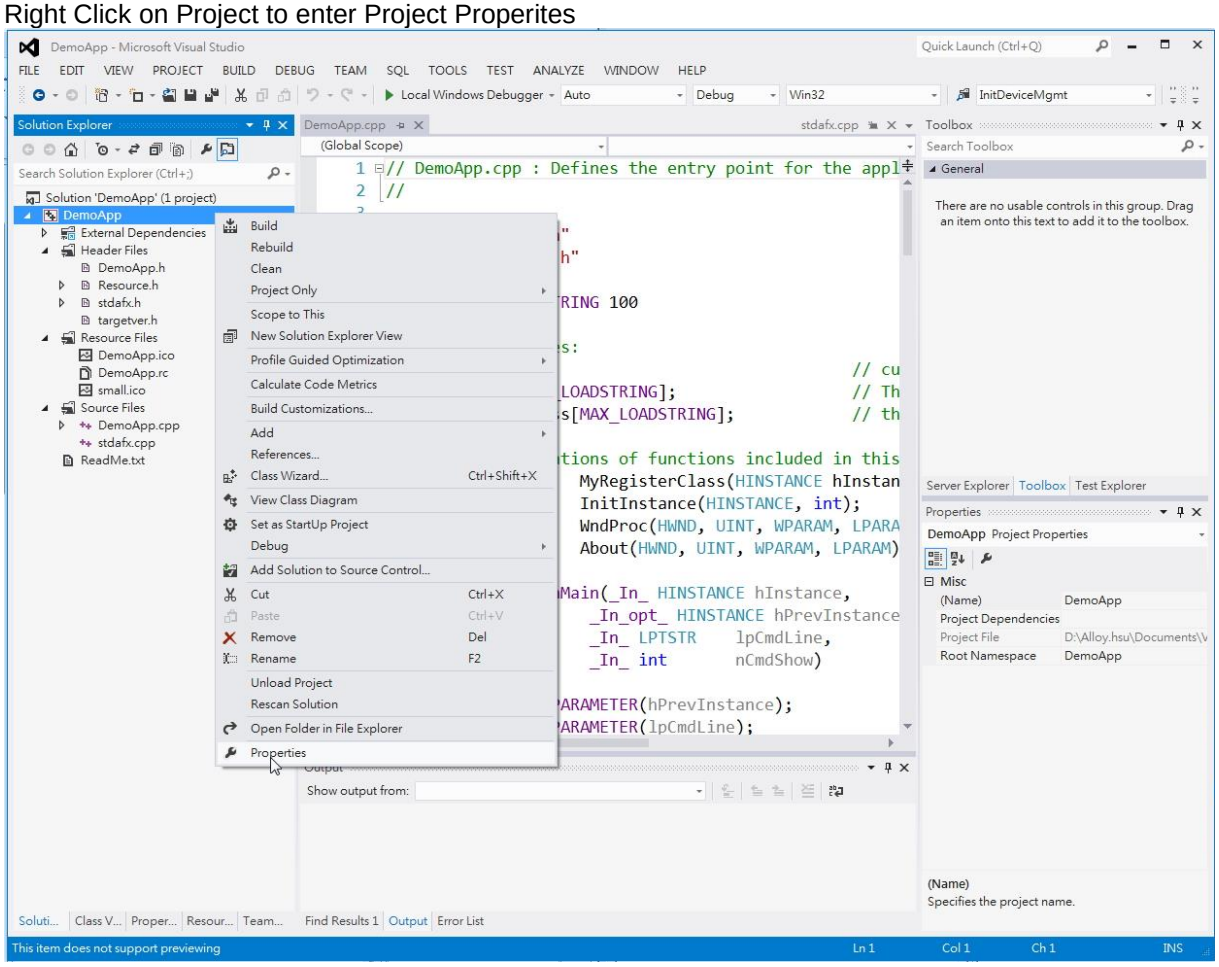


- APIs for reading data need an array for argument to store data. The array should be allocated before you pass it to the API and the data will be stored at **index 0 of the array.**
- You should always check the return value of APIs for error checking. The value should equal to MRM_ERR_NO_ERROR(0) when success or other value when failed.

- **IVCP Service Behavior**
  - The IVCP Service will be started when **ivcp_bind_service()** is called and keep alive when client APP unbind.
  - The IVCP Service might be stopped **by user manually** or **by system automatically (ex: when low memory)**.

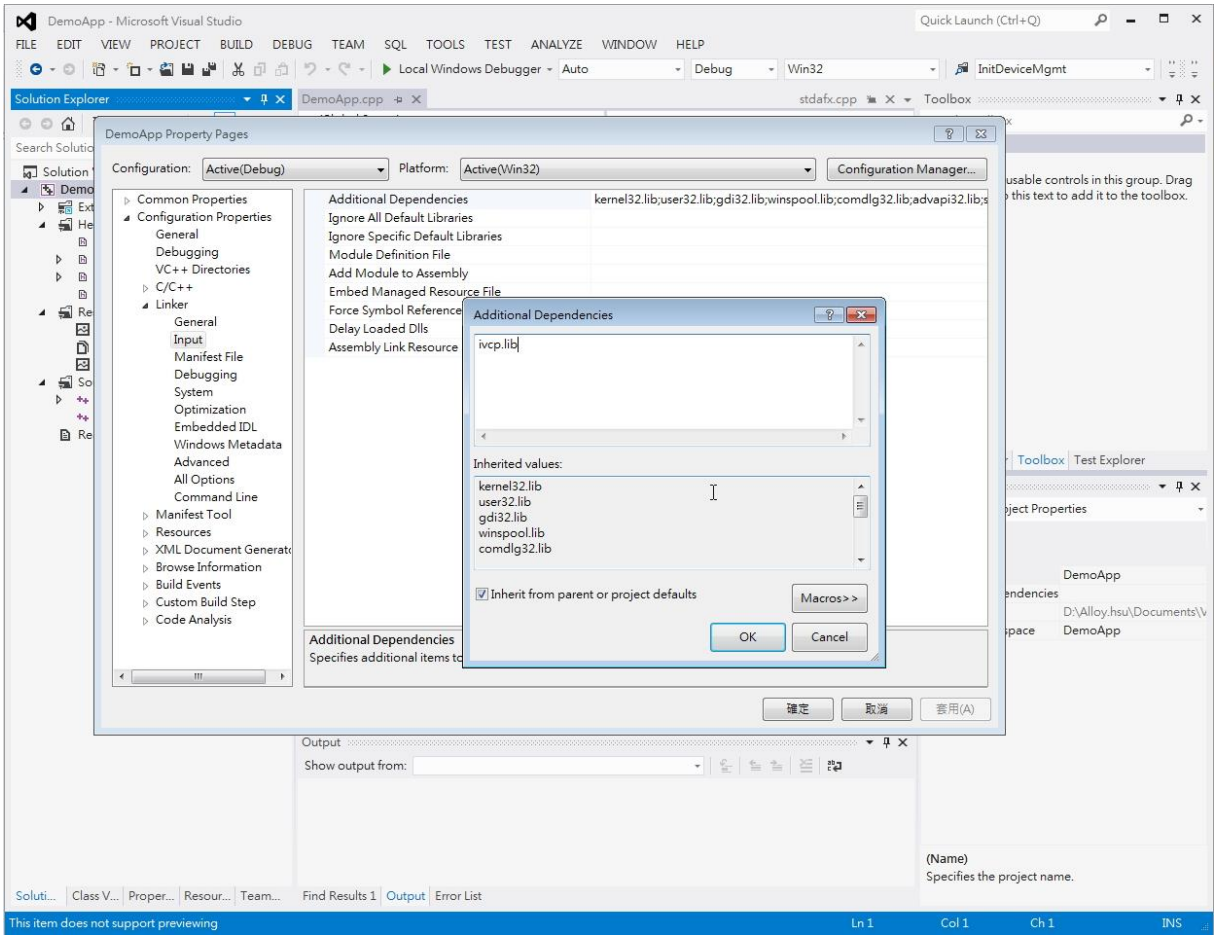
2.2 C++ with Visual Studio

1. Right Click on Project to enter Project Properites



2. Add #include "ivcp/ivcp.h" to your program

3. Select **Linker** and **Input** page. Set Additional Dependencies "ivcp.lib"


DemoApp - Microsoft Visual Studio
DemoApp Property Pages
Configuration: Active(Debug)
Platform: Active(Win32)
Configuration Manager...
Common Properties
Configuration Properties
General
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Command Line
Manifest Tool
Resources
Browse Information
Build Events
Custom Build Step
Code Analysis
Additional Dependencies
Ignore All Default Libraries
Ignore Specific Default Libraries
Module Definition File
Add Module to Assembly
Embed Managed Resource File
Force Symbol Reference
Delay Loaded Dlls
Assembly Link Resource
kernel32.lib;user32.lib;gdi32.lib;winspool.lib;comdlg32.lib;advapi32.lib;
Additional Dependencies
ivcp.lib
Inherited values:
kernel32.lib
user32.lib
gdi32.lib
winspool.lib
comdlg32.lib
Inherit from parent or project defaults
Macros>>
OK
Cancel
Additional Dependencies
Specifies additional items to
確定
取消
套用(A)
Output
Show output from:
(Name)
Specifies the project name.
This item does not support previewing
Ln 1
Col 1
Ch 1
INS

2.3 JAVA with Android Studio

1. **Install MRM IVCP Service to your device**

   Please find the **mrm_service.apk** in the MRM SDK package.
   Connect you device to your computer with ADB, then execute the following ADB commands to install MRM IVCP Service

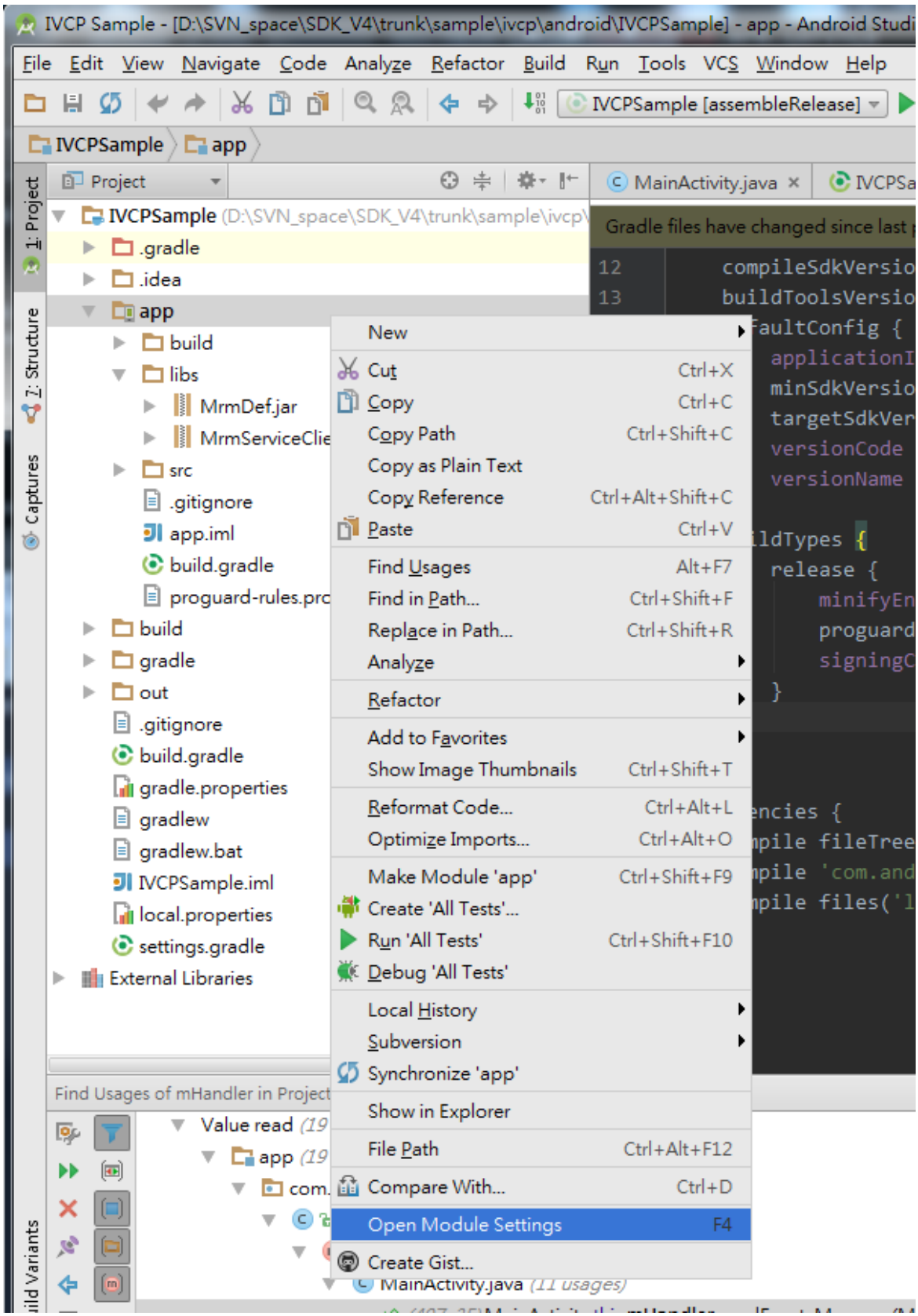
```
adb install .\bin\mrm_service.apk
```

2. **Import MRM IVCP Service Client API library to your project**

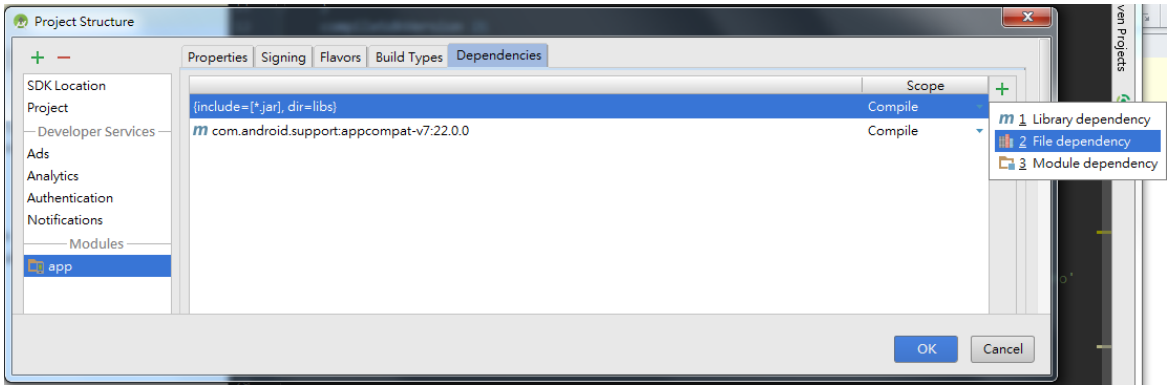
   To access MRM IVCP Service, you must import the MRM IVCP Service Client API lib into you project.

   Please find the **MrmServiceClientAPI.jar** and **MrmDef.jar** in the MRM SDK package. Copy the libraries to the directory **/[Module Name]/libs/** in you Android Studio project (the default module name might be "app").
   Then import the libraries by following the steps below:

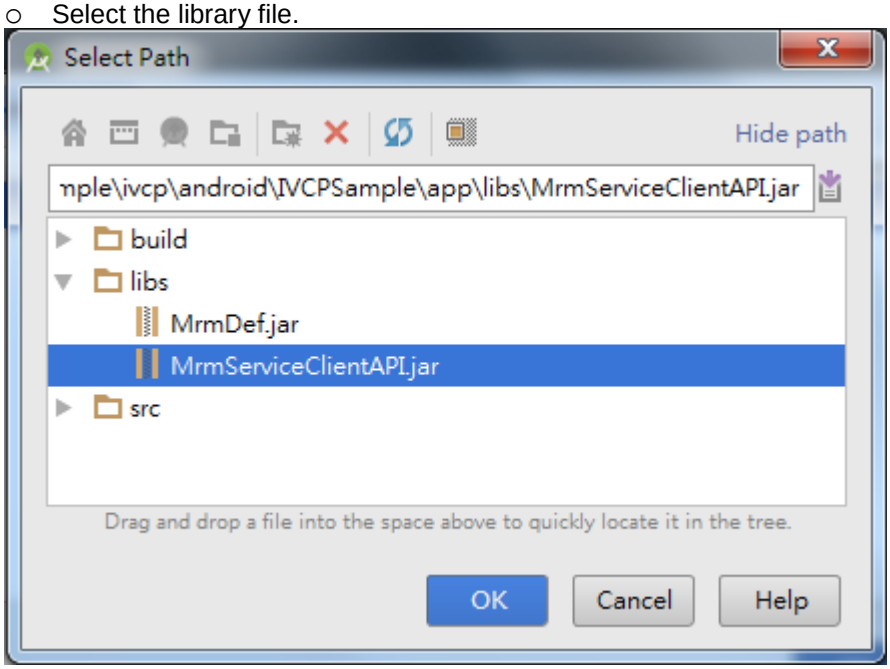
   - Right click on you APP module. Click "**Open module settings**"



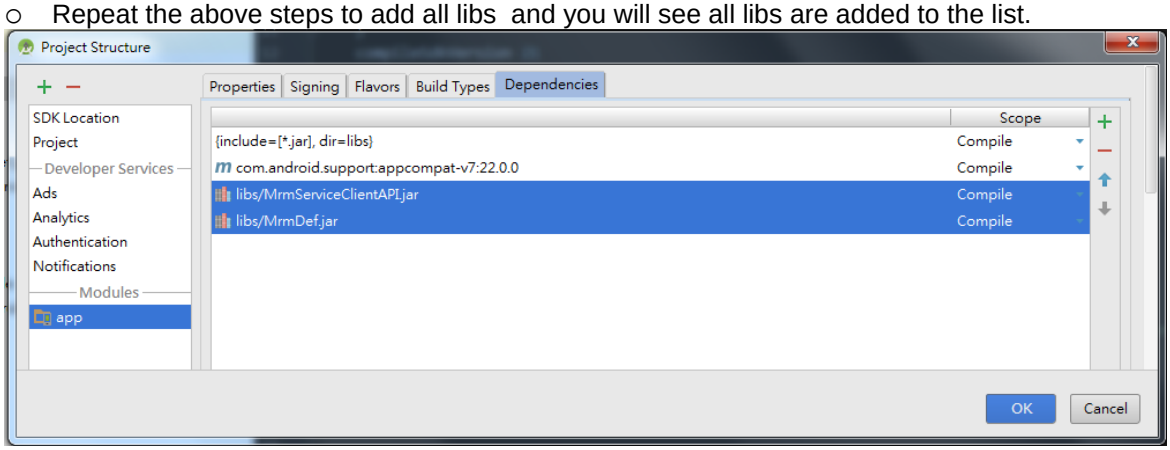
- Click the "**Dependency**" tab. Then click "**+**" -> "**Library dependency**"



- Select the library file.



- Repeat the above steps to add all libs and you will see all libs are added to the list.



3 Function Overview

3.1 Power Management Module

- **Power On**

Power management provide a flexibly power check and boot mode control. You can according to the demand change your setting. The TREK device supported many kinds of wakeup source. For instance, boot from G-Sensor, WWAN SMS, Alarm, Ignition, Power Button, Reset Button,Keep a live, AT-mode. Boot from Ignition can divided **three** type: boot from Keep a live mode power on, AT-mode power on and Ignition Off to On power on. The priority like above order. Keep alive mode power on priority then AT-mode power on. What's Keep a live mode power on? When Ignition keep the status "On". If power status at Power off or Suspend, the system will power on Immediately. At this stage, the Ignition on delay will **ignored**; On the other hand, the AT-mode power on and Ignition off to on power on have the Ignition on delay. Normally this delay is a short period time, may be 1~2s, waiting signal stable. After the IVCP to confirm the signal steadied than power on. If the signal violating the conditions in the Ignition on delay, the computer wouldn't boot. What's AT-mode power on? When Ignition status keep on and the computer first the electric current is turned on. After Ignition on delay the ignition status keep on than the system will power on Immediately. Finally, the Ignition off to on power on is when you enable the ignition wakeup and the ignition status off to on than the system boot immediately.

- **Low Voltage Protection**

The power management sub-module also provide low voltage protection (LVP) feature. The system boot divided two parts. Before system power on calling pre-boot. At this time point you can decided whether to enable the pre-boot low voltage power check. If the you enable the pre-boot LVP and the power voltage low then threshold the system would not boot. After system power on calling post-boot. At this stage the IVCP will continues check the power voltage until power off. When the power voltage start low then threshold, the countdown timer start countdown. This time calling the low voltage delay time. If current voltage recover th normal at low voltage delay time, the countdown timer will stop and return to power on otherwise you will enter the low voltage hard delay time. At this moment, the system ready to go power off. At this stage the voltage return to normal would be affect power off processing. You can use ivcp_pm_get_lvp_range() and ivcp_pm_reset_lvp_threshold() to get the range and default threshold.

- **Power off**

Normal power off divided the below method: OS shut down, Ignition On to Off and push power button. When Ignition off to on or manually OS shutdown, the system will countdown Ignition off delay. If the times up, the system will countdown the ignition off hard delay, the system will power off. When push power button there have not Ignition on to off delay, but have Ignition off hard delay. The difference between ignition off delay and ignition hard delay is returnable ability. When you enter the ignition off delay, the signal condition is change, the power off processing will aborted(The OS processing not aborted). It can avoided the ignition signal not stable to cause unexpected power off.

Please refer to Power Management Usage to check detail operation.

3.2 Watchdog Module

Watchdog sub-module provide a mechanism to avoid system Hang. First of all, you need use **ivcp_watchdog_set_time()** to setup countdown time. Second, you need use **ivcp_watchdog_enable()** to start watchdog function. The time decrease over time. When the countdown timer equal to zero, the system will automotive reboot. You should keep use **ivcp_watchdog_trigger**() period a time in order to prevent system reboot. If the system hang up, The IVCP not receive any trigger signal. The system will force reboot Immediately. Default this function is disable. You should enable this function each boot.

4 Application Programming Interface

4.1 IVCP Management Functions

4.1.1 Usage

4.1.1.1 Windows/Linux

Please refer to the IVCP Conventions for basic usage.

4.1.1.2 Android

Please refer to the IVCP Conventions for basic usage.

4.1.2 Constant

4.1.2.1 Android

Class:
mrm.define.**MRM_CONSTANTS**

Fields:

Field Name	Type	Value
IVCP_EVENT_ID_UNKNOWN	int	-1
IVCP_EVENT_ID_GSENSOR_ALARM	int	0

4.1.3 APIs

4.1.3.1 ivcp_init

Syntax:

Windows / Linux	mrm_err **ivcp_init**(void)
Android	-

Description:

General initialization of the IVCP library.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

- Prior to calling any IVCP API function the library needs to be initialized by calling this function. The return code for all IVCP API function will be **MRM_ERR_LIBRARY_NOT_INIT** unless this function is called.
- For Android, you **do not** have to call this function. The IVCP Service will initialize SDK automatically when it is started.

4.1.3.2 ivcp_deinit

Syntax:

Windows / Linux	mrm_err **ivcp_deinit**(void)
Android	-

Description:

General uninitialization of the IVCP library.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

- For Android, you do not have to call this function.
  The IVCP Service will deinitialize SDK automatically when it is stopped.

4.1.3.3 ivcp_bind_service

Syntax:

Windows / Linux	-
Android	int **ivcp_bind_service**(IVCPServiceConnection conn)

Description:

Connect current application context(ex: Activity) to IVCP Service.

Parameters:

conn:

A instance of **mrm.client.IVCPServiceClient.IVCPServiceConnection**.

You must implement the following interfaces of **IVCPServiceConnection**:

public void on_service_connected()

This is a callback which is called when the service is connected.

It will be trigged after you called ivcp_bind_service().

public void on_service_disonnected()

This is a callback which is called when the service is disconnected.

It will be trigged when the IVCP service process is stopped or died.

IVCP Service might be stopped **by user manually** or **by system automatically (ex: when low memory)**.

Please refer to the IVCP sample code for the detailed usage.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

- You can only access IVCP API after **ivcp_bind_service()** is called and the **on_service_connected()** callback is triggered. Before that, for most IVCP API calls, you will get error code **MRM_ERR_ANDROID_CLIENT_SERVICE_DISCONNECTED.**
- If IVCP Service is in stopped status, the service will be started when APP called **ivcp_bind_service().**
- Due to the nature of Android, a background service(e.g. IVCP Service) might be killed by system automatically, you should carefully implement **on_service_disonnected()** callback to deal with the service disconnect event.
- You can call **ivcp_bind_service()** in **on_service_disonnected()** callback to re-connect if you need to.
- You should call **ivcp_unbind_service()** before your application context(ex: Activity) is destroyed or it

may cause memory leak.

4.1.3.4 ivcp_unbind_service

Syntax:

Windows / Linux	-
Android	int **ivcp_unbind_service**()

Description:

Disconnect current application context from IVCP Service.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

You should call this function before your application context(ex: Activity) is destroyed or it may cause memory leak.

4.1.3.5 ivcp_is_service_connected

Syntax:

Android	int **ivcp_is_service_initialized**(boolean[] status)

Description:

Get initialization status of IVCP Service.

Parameters:

status

An allocated array of size 1 for storing returned initialization status.

The returned value will be store at index 0.

The value is TRUE, if IVCP Service is initialized and all served APIs are available.

The value is FALSE, if IVCP Service is not initialized.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remark:

-

4.1.3.6 ivcp_is_service_initialized

Syntax:

Android	boolean **ivcp_is_service_connected**()

Description:

Check whether the client application process is connected to IVCP Service.

Parameters:

-

Returns:

Returns TRUE, if connected.

Returns FALSE, if disconnected.

Remark:

-

4.1.3.7 ivcp_get_version

Syntax:

Windows / Linux	mrm_err **ivcp_get_version**(char *version)
Android	int **ivcp_get_version**(byte[] version)

Description:

Get the version of SDK.

Parameters:

version [out]

Pointer to a buffer that will hold the version of SDK. The buffer is C string that end of '\0'. The content of unused bytes filled 0x00.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remark:

- The maximum length of version string is **IVCP_MAXIMUM_LIBRARY_STRING_LENGTH**(24)
- For Android, you can find the constant value **IVCP_MAXIMUM_LIBRARY_STRING_LENGTH** under namespace **mrm.define**

4.1.3.8 ivcp_get_platform_name

Syntax:

Windows / Linux	mrm_err **ivcp_get_platform_name**(char *name)
Android	int **ivcp_get_platform_name**(byte[] name)

Description:

Get the platform name.

Parameters:

name [out]

Pointer to a buffer that will hold the version of SDK. The buffer is C string that end of '\0'. The content of unused bytes filled 0x00.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remark:

- The maximum length of version string is **IVCP_MAXIMUM_PLATFORM_STRING_LENGTH**(16)
- For Android, you can find the constant value **IVCP_MAXIMUM_PLATFORM_STRING_LENGTH** under namespace **mrm.define**

4.1.3.9 ivcp_get_device_serial_number

Syntax:

Windows / Linux	mrm_err **ivcp_get_device_serial_number**(char *serial_number)
Android	int **ivcp_get_device_serial_number**(byte[] serial_number)

Description:

Get the platform name.

Parameters:

serial_number [out]

Pointer to a buffer that will hold the serial number of device. The buffer is C string that end of '\0'.

The content of unused bytes filled 0x00.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remark:

- The maximum length of version string is **IVCP_MAXIMUM_DEVICE_SERIAL_NUMBER_STRING_LENGTH**(64)
- For Android, you can find the constant value **IVCP_MAXIMUM_DEVICE_SERIAL_NUMBER_STRING_LENGTH** under namespace **mrm.define**

4.2 Firmware Management Functions

4.2.1 Usage

4.2.1.1 Windows/Linux

Please refer to the IVCP Conventions for basic usage.

4.2.1.2 Android

Please refer to the IVCP Conventions for basic usage.

4.2.2 APIs

4.2.2.1 ivcp_firmware_get_version

Syntax:

Windows / Linux	mrm_err **ivcp_firmware_get_version**(char *version)
Android	int **ivcp_firmware_get_version**(byte[] version)

Description:

Get the version of firmware.

Parameters:

version [out]

Pointer to a buffer that will hold the version of firmware. The buffer is C string that end of '\0'. The content of unused bytes filled 0x00.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remark:

- The maximum length of version string is **IVCP_MAXIMUM_FIRMWARE_VERSION_LENGTH**(16)
- For Android, you can find the constant value **IVCP_MAXIMUM_FIRMWARE_VERSION_LENGTH** under namespace **mrm.define**

4.2.2.2 ivcp_firmware_load_default

Syntax:

Windows / Linux	mrm_err **ivcp_firmware_load_default**(void)
Android	int **ivcp_firmware_load_default**()

Description:

Load the default configuration to current configuration.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

- This function will take about 300ms to 600ms for loading the default configuration.
- You should call **ivcp_firmware_save_default()** once to save your user default configuration to the device before you call this API to load,
  or you will get error code **MRM_ERR_IVCP_USER_DEFAULT_IS_EMPTY**,

4.2.2.3 ivcp_firmware_save_default

Syntax:

Windows / Linux	mrm_err **ivcp_firmware_save_default**(void)
Android	int **ivcp_firmware_save_default**()

Description:

Save current configuration as the default configuration.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

This function will take about 300ms to 600ms for saving the default configuration.

4.3 Power Management Functions

4.3.1 Usage

4.3.1.1 Ignition Control

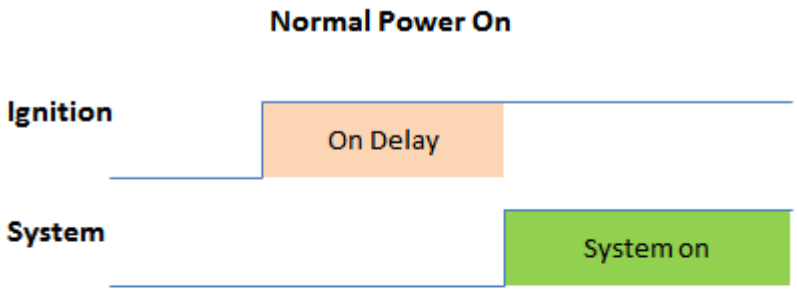
The IVCP monitors the events of ignition ON/OFF and handles the power sequence of TREK.

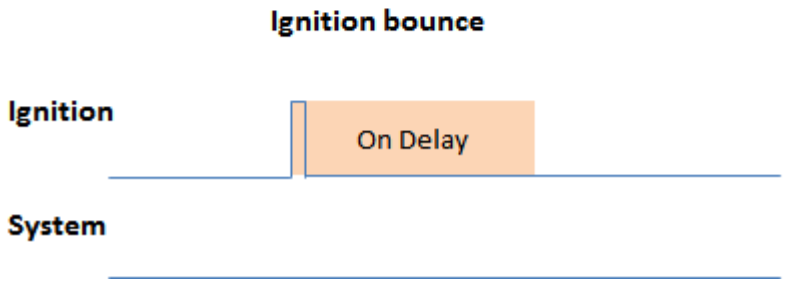
- **Ignition ON Event**

When Ignition status is switched from OFF to ON, the VPM will count "Ignition ON delay" period to wait signal become stable. After count, the IVCP will power on the TREK device.

If the ignition signal unstable the conditions (e.g switched from ON to OFF) during the "Ignition on delay", the TREK will not be booted.

The "Ignition ON delay" can be adjust by using ivcp_pm_set_event_delay(IVCP_EVENT_IGNITION_ON, times) and can be get by using ivcp_pm_get_event_delay(IVCP_EVENT_IGNITION_ON, times) .





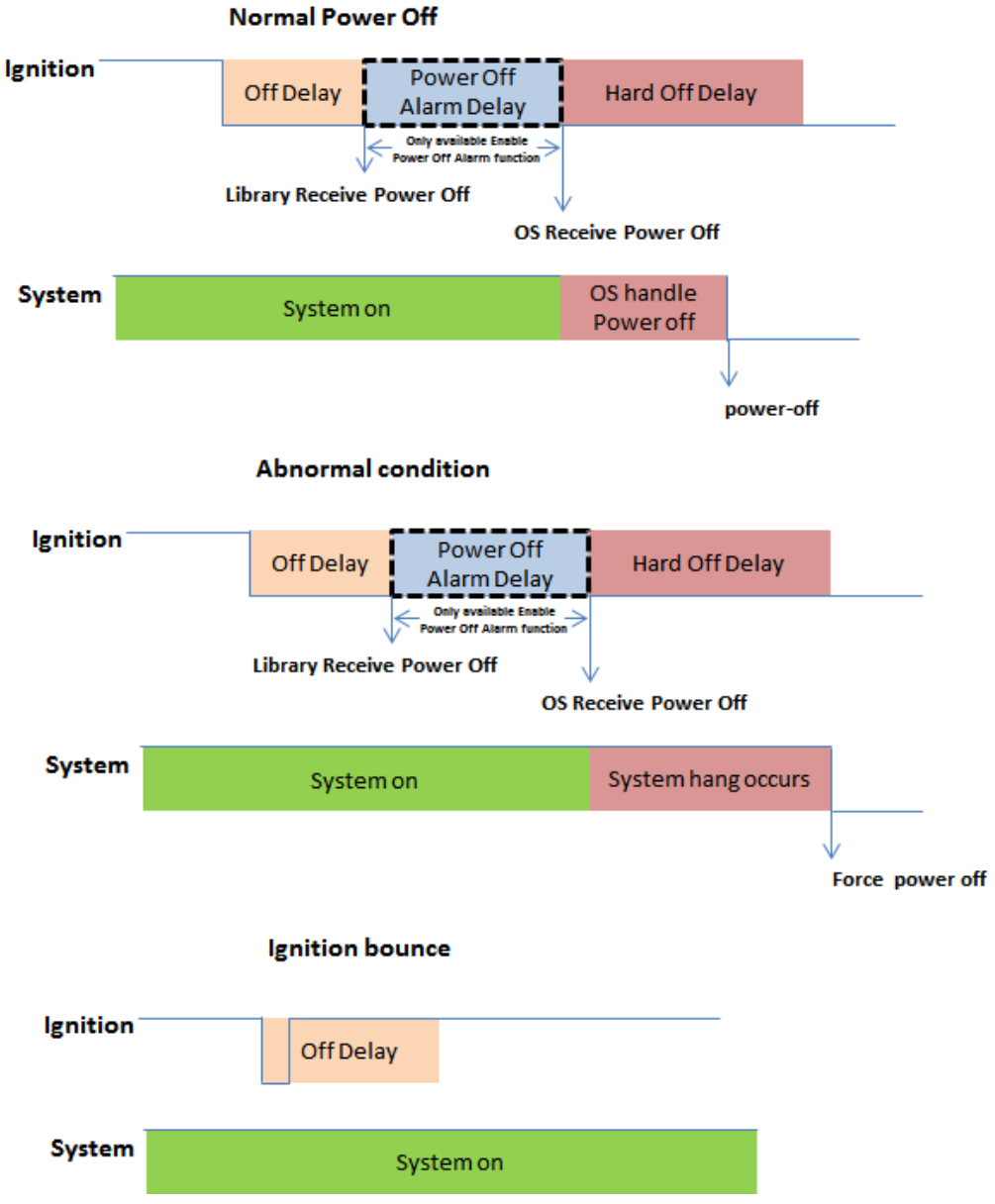
- **Ignition OFF Event**

When Ignition status is switched from ON to OFF, the IVCP will start counting "Ignition OFF delay". If the ignition signal unstable conditions (e.g switched from OFF to ON) during the period, the IVCP will stop counting and stop the power off sequence.

After the "Ignition OFF delay" period expired,the IVCP may enter "Power off alarm delay" when power off alarm function enable. This time prepare for Application have full time to handle shutdown process. After "Power off alarm delay" time period expired, the IVCP then start power off sequence and start counting "Hard OFF delay" and then power off the TREK device. Once the IVCP start power sequence, the count can not be canceled. The TREK device will be forced power off by the VPM when "Hard OFF delay"

expired to avoid unexpected OS freeze.

The "Hard OFF delay" can be adjust by using ivcp_pm_set_event_delay(IVCP_EVENT_IGNITION_OFF_HARD, times) and can be get by using ivcp_pm_get_event_delay(IVCP_EVENT_IGNITION_OFF_HARD, times) .



4.3.1.2 AT Mode / Keep Alive Mode

There are two mode settings for VPM to control how the TREK device should be booted up - the AT-Mode and the Keep-Alive-Mode :

- **Keep-Alive-Mode**

This mode decides whether TREK device should keeps power on.
The Keep-Alive-Mode setting can be enabled or disabled. If Keep-Alive-Mode is enabled, IVCP will check whether the ignition status is ON when when the TREK device goes to power off or suspend status. If ignition status is ON, IVCP will boot up TREK device immediately , otherwise the TREK device will not be booted up..
You can use **ivcp_pm_set_alive_mode**() to set status of Keep-Alive-Mode and use **ivcp_pm_get_alive_mode**() to get status of Keep-Alive-Mode.

- **AT-Mode**

This mode decides whether TREK device should be booted up when the electric current is turned on.
The AT-Mode setting can be enabled or disabled. If AT-Mode is enabled, IVCP will check whether the ignition status is ON when when the electric current is turned on. If ignition status is ON, IVCP will start counting ignition ON delay then boot up TREK device, otherwise the TREK device will not be booted up..
For the definition of ignition on delay and the related sequence, please refer to ignition control section.
You can use **ivcp_pm_set_at_mode**() to set status of AT-Mode and use **ivcp_pm_get_at_mode**() to get status of AT-Mode.

If both mode are enabled, the Keep-Alive-Mode is prior than AT-Mode.

4.3.1.3 Wakeup Control

The computer may waken up by many source, such as ignition, alarm, WWAN SMS, reset button and power button.

- **Power button**

TREK device normally have power button by connect TREK 3XX device. The power button normal behind screen. You can use power button to wake up device. This wake up source can't disable.

- **Ignition**

You can use **ivcp_pm_ignition_wakeup_enable**() to enable ignition wake up function and use **ivcp_pm_get_ignition_wakeup_status**() to get ignition wake up function's status. The default of ignition wake up is enable. Some situation you may want disable ignition wakeup function by **ivcp_pm_ignition_wakeup_disable**().

- **Alarm**

IVCP have an external RTC for reading real time. It is based on the RTC's time to wake up device. First of all, you need use **ivcp_alarm_set_real_time**() to setup real time. Alarm wake up support three mode to wake up device, for example, wake up by hour, wake up by day and wake up by weeks. You can use **ivcp_alarm_set_wakeup_mode**() to set wake up mode and use **ivcp_alarm_set_wakeup_time**() to setup specific time to wake up. For instance, if you want wake up 10 minutes per hour. You can set wake up by hour and minute of wake up alarm time to 10. Finally, you need use **ivcp_alarm_wakeup_enable**() to enable Alarm wake up function. The default of alarm wake up is disable.

- **G-sensor**

You can use **ivcp_gsensor_wakeup_enable**() to enable G-sensor wake up function and use **ivcp_gsensor_get_wakeup_status**() to get G-sensor wake up function's status. The default of G-sensor wake up is disable. First of all, you need use **ivcp_gsensor_set_wakeup_threshold**() to setup the threshold of G-sensor wake up. When the device's Acceleration is greater than the threshold. IVCP will wake up its. Normal the threshold bigger than 4G when you want to detect tap's behavior. You should not set the threshold too low. Otherwise the TREK device will easy to boot up.

- **WWAN SMS**

You can use **ivcp_peripheral_wwan_wakeup_enable**() to enable WWAN SMS wake up function and use **ivcp_peripheral_get_wwan_wakeup_status**() to get function's status. The default of WWAN SMS wake up is disable. IVCP will keep WWAN module standby power when this function is enable. In other word, this function will cause more power consumption. When TREK device insert SIM card and received any SMS, IVCP will wake up its.

4.3.1.4 Low Battery Protection

IVCP provides low battery protection (LBP) functions which monitors whether the voltage of battery is above specified threshold and control the power of TREK device.
If LBP function is enabled, when the battery voltage is under the specified threshold (low battery), IVCP will power off the TREK device for safety purpose. Default this function is **disable**.

The LBP functions includes Pre-boot LBP and Post-boot LBP:

- **Pre-boot LBP**

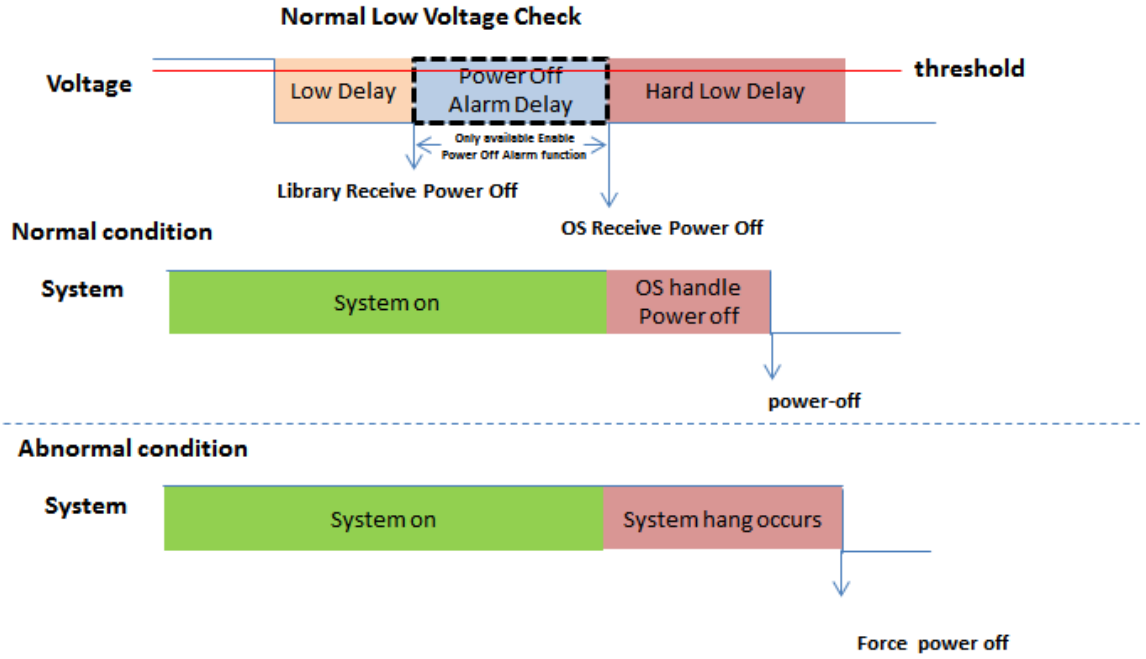
"Pre-boot" is defined as "before system boot up". The pre-boot LBP function is able to do the battery voltage check before the system boot up and decide whether to boot up the TREK device.
If Pre-boot LBP enabled, IVCP check whether battery voltage is equal to or above the specified threshold for a specified "LBP low delay" period, if the battery voltage is stable, IVCP allow system to boot up. Otherwise, the its will not boot up.

For setting/getting the LBP low delay, you can use
ivcp_pm_set_event_delay(**IVCP_EVENT_LOW_VOLTAGE**) and

ivcp_pm_get_event_delay(IVCP_EVENT_LOW_VOLTAGE) .

And you can enable/disable pre-boot LBP function by using **ivcp_pm_lvp_preboot_enable**() and get the status of pre-boot LBP function by using **ivcp_pm_lvp_preboot_get_status**().

- **Post-boot LBP**



"Post-boot" is defined as "after OS boot up". The post-boot LBP function is able to do the battery voltage check after the system boot up and decide whether to power off the TREK device when battery low. If Post-boot LBP enabled, IVCP check whether battery voltage is under the specified threshold for a specified "LBP low delay" period, if the battery voltage is stable and low, IVCP asks OS to shutdown and start counting "LBP low hard delay" period. If OS can not be shutdown during "LBP low hard delay" period, IVCP will force power off TREK device.

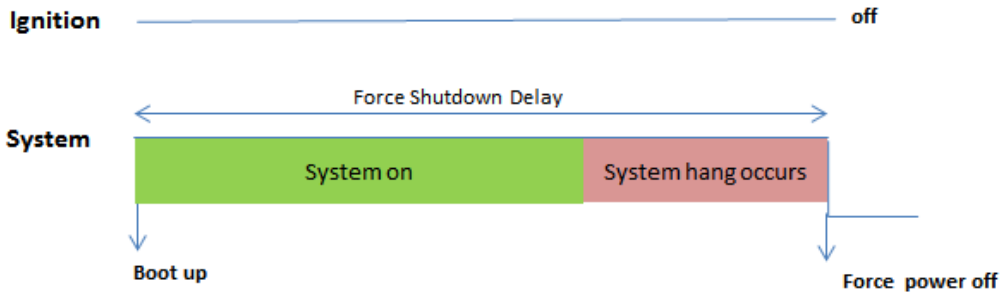
For setting/getting the LBP low hard delay, you can use
ivcp_pm_set_event_delay(IVCP_EVENT_LOW_VOLTAGE_HARD) and
ivcp_pm_get_event_delay(IVCP_EVENT_LOW_VOLTAGE_HARD) .

And you can enable/disable pre-boot LBP function by using **ivcp_pm_lvp_postboot_enable**() and get the status of pre-boot LBP function by using **ivcp_pm_lvp_postboot_get_status**().

4.3.1.5 Force Shutdown Control

Some scenarios you want remote boot up TREK device to do some maintenance process. But in this condition, if system hang up, you have no any interface to shutdown TREK device. You can use force shutdown function to prevent battery run out. This condition assume you not near by TREK device. The ignition signal should be keep off. If ignition is on, this function will automotive disable.

You can use **ivcp_pm_force_shutdown_enable**() to enable this function and use **ivcp_pm_get_force_shutdown_status**() to get function status.



4.3.2 Enumeration

4.3.2.1 Windows/Linux

ivcp_power_mode Enum

- (1) **IVCP_POWER_MODE_48V** - The power mode operation on 48V.
- (2) **IVCP_POWER_MODE_24V** - The power mode operation on 24V.
- (3) **IVCP_POWER_MODE_12V** - The power mode operation on 12V.

ivcp_pm_wakeup_type Enum

- (0) **IVCP_WAKEUP_TYPE_POWER_BUTTON** - Computer wakeup power button pushed.
- (1) **IVCP_WAKEUP_TYPE_IGNITION** - Computer wakeup form ignition off to on.
- (2) **IVCP_WAKEUP_TYPE_WWAN** - Computer wakeup form WWAN module.
- (3) **IVCP_WAKEUP_TYPE_GSENSOR** - Computer wakeup form gsensor.
- (4) **IVCP_WAKEUP_TYPE_DI1** - Computer wakeup form digital input 1.
- (5) **IVCP_WAKEUP_TYPE_DI2** - Computer wakeup form digital input 2.
- (6) **IVCP_WAKEUP_TYPE_ALARM** - Computer wakeup form RTC alarm.
- (7) **IVCP_WAKEUP_TYPE_HOTKEY** - Computer wakeup form hotkey board.
- (8) **IVCP_WAKEUP_TYPE_DI3** - Computer wakeup form digital input 3.
- (9) **IVCP_WAKEUP_TYPE_DI4** - Computer wakeup form digital input 4.
- (10) **IVCP_WAKEUP_TYPE_KEEP_ALIVE_MODE** - Computer wakeup form keep a live mode.
- (11) **IVCP_WAKEUP_TYPE_AT_MODE** - Computer wakeup form AT mode.
- (12) **IVCP_WAKEUP_TYPE_RESET** - Computer wakeup form software reset or reset button.

ivcp_pm_shutdown_mask Enum

- (0) **IVCP_SHUTDOWN_MASK_POWER_BUTTON** - The shut-down mask of power button source.
- (1) **IVCP_SHUTDOWN_MASK_IGNITION** - The shutdown mask of ignition on to off source.

ivcp_pm_event Enum

- (0) **IVCP_EVENT_LOW_VOLTAGE** - The low voltage event.
- (1) **IVCP_EVENT_LOW_VOLTAGE_HARD** - The low voltage hard time event.
- (2) **IVCP_EVENT_IGNITION_ON** - The ignition off to on event.
- (3) **IVCP_EVENT_IGNITION_OFF_TO_POWER_OFF** - The ignition on to off and goto power off event.
- (4) **IVCP_EVENT_IGNITION_OFF_HARD** - The ignition on to off hard time event.
- (5) **IVCP_EVENT_POST_BOOT_POWER_CHECK** - The post-boot power check event.
- (6) **IVCP_EVENT_IGNITION_OFF_TO_SUSPEND** - The ignition on to off and goto suspend event.
- (7) **IVCP_EVENT_POWER_OFF_ALARM** - The power off alarm event.

Remarks:

- For Android, you can get the enumeration values from the enum class **mrm.define.MRM_ENUM.IVCP_PM_EVENT.**
  (ex: MRM_ENUM.IVCP_PM_EVENT.**IVCP_EVENT_IGNITION_OFF_TO_POWER_OFF**.getValue())
  Please refer to sample code for detailed usage.

4.3.2.2 Android

ivcp_power_mode

class:

mrm.define.MRM_ENUM.IVCP_PM_POWER_MODE

Enum:

Name	Type	value	Comment
IVCP_POWER_MODE_48V	int	1	The power mode operation on 48V.
IVCP_POWER_MODE_24V	int	2	The power mode operation on 24V.
IVCP_POWER_MODE_12V	int	3	The power mode operation on 12V.

Remark:

Use the method **getValue()** to get the enum value.

ex: MRM_ENUM.IVCP_PM_POWER_MODE.IVCP_POWER_MODE_48V**.getValue()** returns 1.

Please refer to sample code for detailed usage.

ivcp_pm_wakeup_type

class:

mrm.define.MRM_ENUM.IVCP_PM_WAKEUP_TYPE

Enum:

Name	Type	value	Comment
IVCP_WAKEUP_TYPE_POWER_BUTTON	int	0	Computer wakeup power button pushed.
IVCP_WAKEUP_TYPE_IGNITION	int	1	Computer wakeup form ignition off to on.
IVCP_WAKEUP_TYPE_WWAN	int	2	Computer wakeup form WWAN module.
IVCP_WAKEUP_TYPE_GSENSOR	int	3	Computer wakeup form gsensor.
IVCP_WAKEUP_TYPE_DI1	int	4	Computer wakeup form digital input 1.
IVCP_WAKEUP_TYPE_DI2	int	5	Computer wakeup form digital input 2.
IVCP_WAKEUP_TYPE_ALARM	int	6	Computer wakeup form RTC alarm.
IVCP_WAKEUP_TYPE_HOTKEY	int	7	Computer wakeup form hotkey board.
IVCP_WAKEUP_TYPE_DI3	int	8	Computer wakeup form digital input 3.
IVCP_WAKEUP_TYPE_DI4	int	9	Computer wakeup form digital input 4.
IVCP_WAKEUP_TYPE_KEEP_ALIVE_MODE	int	10	Computer wakeup form keep a live mode.
IVCP_WAKEUP_TYPE_AT_MODE	int	11	Computer wakeup form AT mode.
IVCP_WAKEUP_TYPE_RESET	int	12	Computer wakeup form software reset or reset button.

Remark:

Use the method **getValue()** to get the enum value.

ex: MRM_ENUM.IVCP_PM_WAKEUP_TYPE.IVCP_WAKEUP_TYPE_IGNITION**.getValue()** returns 1.

Please refer to sample code for detailed usage.

ivcp_pm_shutdown_mask

class:

mrm.define.MRM_ENUM.IVCP_PM_SHUTDOWN_MASK

Enum:

Name	Type	value	Comment
IVCP_SHUTDOWN_MASK_POWER_BUTTON	int	0	The shut-down mask of power button source.
IVCP_SHUTDOWN_MASK_IGNITION	int	1	The shutdown mask of ignition on to off source.

Remark:

Use the method **getValue()** to get the enum value.

ex: MRM_ENUM.IVCP_PM_SHUTDOWN_MASK.IVCP_SHUTDOWN_MASK_IGNITION**.getValue()** returns 1.

Please refer to sample code for detailed usage.

ivcp_pm_event

class:

mrm.define.MRM_ENUM.**IVCP_PM_EVENT**

Enum:

Name	Type	value	Comment
IVCP_EVENT_LOW_VOLTAGE	int	0	The low voltage event.
IVCP_EVENT_LOW_VOLTAGE_HARD	int	1	The low voltage hard time event.
IVCP_EVENT_IGNITION_ON	int	2	The ignition off to on event.
IVCP_EVENT_IGNITION_OFF_TO_POWER_OFF	int	3	The ignition on to off and goto power off event.
IVCP_EVENT_IGNITION_OFF_HARD	int	4	The ignition on to off hard time event.
IVCP_EVENT_POST_BOOT_POWER_CHECK	int	5	The post-boot power check event.
IVCP_EVENT_IGNITION_OFF_TO_SUSPEND	int	6	The ignition on to off and goto suspend event.
IVCP_EVENT_POWER_OFF_ALARM	int	7	The power off alarm event.

Remark:

Use the method **getValue()** to get the enum value.

ex: MRM_ENUM.IVCP_PM_EVENT.IVCP_EVENT_LOW_VOLTAGE_HARD**.getValue()** returns 1.

Please refer to sample code for detailed usage.

4.3.3 APIs

4.3.3.1 ivcp_pm_power_off

Syntax:

Windows / Linux	-
Android	int **ivcp_pm_power_off**()

Description:

Trigger the shutdown process of the operating system by setting the working mode of IVCP firmware to power off mode.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.2 ivcp_pm_get_ignition_status

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_ignition_status**(char *status)
Android	int **ivcp_pm_get_ignition_status**(boolean[] status)

Description:

Get status of ignition.

Parameters:

status [out]

Ignition status. The value 1 is On otherwise 0 is Off.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.3 ivcp_pm_ignition_wakeup_enable

Syntax:

Windows / Linux	mrm_err **ivcp_pm_ignition_wakeup_enable**(void)
Android	int **ivcp_pm_ignition_wakeup_enable**()

Description:

Enable the ignition wakeup function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.4 ivcp_pm_ignition_wakeup_disable

Syntax:

Windows / Linux	mrm_err **ivcp_pm_ignition_wakeup_disable**(void)
Android	int **ivcp_pm_ignition_wakeup_disable**()

Description:

Disable the ignition wakeup function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.5 ivcp_pm_get_ignition_wakeup_status

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_ignition_wakeup_status**(char *status)
Android	int **ivcp_pm_get_ignition_wakeup_status**(boolean[] status)

Description:

Get status of ignition wakeup function.

Parameters:

status [out]

Ignition wakeup function status. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.6 ivcp_pm_set_ignition_poweroff_event

Syntax:

Windows / Linux	mrm_err **ivcp_pm_set_ignition_poweroff_event**(void *poweroff_event))
Android	-

Description:

Set a user define event in order to let IVCP library notify the specified event when Ignition on to off. This event only trigger when power off alarm function enable. User can using ivcp_pm_power_off_alarm_enable() to enable this function

Parameters:

poweroff_event [in]

Pointer to the power off event. In windows, the poweroff event will pointer to a Windows Events HANDLE. In linux, the poweroff_event will pointer to a structure which consists of a pthread_mutex_t and pthread_cond_t.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

You should not release the event before deinitialize IVCP library.

4.3.3.7 ivcp_pm_set_shutdown_mask

Syntax:

Windows / Linux	mrm_err **ivcp_pm_set_shutdown_mask**(ivcp_pm_shutdown_mask mask, char status)
Android	int **ivcp_pm_set_shutdown_mask**(int mask, boolean status)

Description:

Set shut-down mask This function use to enable or disable the shut-down event source. When the shutdown source set enable, platform can using specifies source to shutdown the machine.

Parameters:

mask [in]

mask enum type. The detail please refer to ivcp_pm_shutdown_mask section. Some platform may not support to control specifies mask.

status [in]

Enable of disable mask. 1 is Enable and 0 is Disable.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

4.3.3.8 ivcp_pm_get_shutdown_mask

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_shutdown_mask**(ivcp_pm_shutdown_mask mask, char *status)
Android	int **ivcp_pm_get_shutdown_mask**(int mask, boolean[] status)

Description:

Get shut-down mask This function use to enable or disable the shut-down event source. When the shutdown source set enable, platform can using specifies source to shutdown the machine.

Parameters:

mask [in]

mask enum type. The detail please refer to ivcp_pm_shutdown_mask section. Some platform may not support to control specifies mask.

status [out]

Enable of disable mask. 1 is Enable and 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.9 ivcp_pm_power_off_alarm_enable

Syntax:

Windows / Linux	mrm_err **ivcp_pm_power_off_alarm_enable**(void)
Android	int **ivcp_pm_power_off_alarm_enable**()

Description:

Enable the power off alarm function. MCU will notify host machine when system go to power off.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.10 ivcp_pm_power_off_alarm_disable

Syntax:

Windows / Linux	mrm_err **ivcp_pm_power_off_alarm_disable**(void)
Android	int **ivcp_pm_power_off_alarm_disable**()

Description:

Disable the power off alarm function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.11 ivcp_pm_get_power_off_alarm_status

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_power_off_alarm_status**(char *status)
Android	int **ivcp_pm_get_power_off_alarm_status**(boolean[] status)

Description:

Get status of power off alarm function.

Parameters:

status [out]

Power off alarm function status. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.12 ivcp_pm_get_voltage

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_voltage**(float *voltage)
Android	int **ivcp_pm_get_voltage**(float[] voltage)

Description:

Get computer current voltage.

Parameters:

voltage [out]

Voltage of computer. The unit is volt.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.13 ivcp_pm_get_alive_mode

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_alive_mode**(char *mode)
Android	int **ivcp_pm_get_alive_mode**(boolean[] mode)

Description:

Get keep a live mode status.

Parameters:

mode [out]

Keep a live mode. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.14 ivcp_pm_set_alive_mode

Syntax:

Windows / Linux	mrm_err **ivcp_pm_set_alive_mode**(char mode)
Android	int **ivcp_pm_set_alive_mode**(boolean mode)

Description:

Set keep a live mode status.

Parameters:

mode [in]

Keep a live mode. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.15 ivcp_pm_get_power_mode

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_power_mode**(char *mode)
Android	int **ivcp_pm_get_power_mode**(byte[] mode)

Description:

Get power mode.

Parameters:

mode [out]

Power mode. The value please refer to ivcp_power_mode.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.16 ivcp_pm_set_power_mode

Syntax:

Windows / Linux	mrm_err **ivcp_pm_set_power_mode**(char mode)
Android	int **ivcp_pm_set_power_mode**(byte mode)

Description:

Set the power mode.

Parameters:

mode [in]

Power mode. The value please refer to ivcp_power_mode.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.17 ivcp_pm_get_power_status

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_power_status**(char *status)
Android	int **ivcp_pm_get_power_status**(boolean[] status)

Description:

Get car power status.

Parameters:

status [out]

Car power exists status. The value is 1(true) if exists, 0(false) if not exists.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.18 ivcp_pm_get_at_mode

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_at_mode**(char *mode)
Android	int **ivcp_pm_get_at_mode**(boolean[] mode)

Description:

Get AT mode status.

Parameters:

mode [out]

AT mode. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.19 ivcp_pm_set_at_mode

Syntax:

Windows / Linux	mrm_err **ivcp_pm_set_at_mode**(char mode)
Android	int **ivcp_pm_set_at_mode**(boolean mode)

Description:

Set AT mode status.

Parameters:

mode [in]

AT mode. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.20 ivcp_pm_get_event_delay

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_event_delay**(ivcp_pm_event event_type, unsigned int *second)
Android	int **ivcp_pm_get_event_delay**(int event_type, int[] second)

Description:

Get delay time of specifies event.

Parameters:

event_type [in]

Control event type. The detail please refer to ivcp_pm_event section. Some platform may not support to contorl specifes event.

second [out]

The event time. The unit is second.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.21 ivcp_pm_set_event_delay

Syntax:

Windows / Linux	mrm_err **ivcp_pm_set_event_delay**(ivcp_pm_event event_type, unsigned int second)
Android	int **ivcp_pm_set_event_delay**(int event_type, int second)

Description:

Set delay time of specifies event.

Parameters:

event_type [in]

Control event type. The detail please refer to ivcp_pm_event section. Some platform may not support to control specifies event.

second [in]

The event time. The unit is second.

Returns:

- **MRM_ERR_NO_ERROR** - On success.
  Otherwise see the error code list.
- Value range (NOTE: These ranges may change based on firmware)
  - (0) IVCP_EVENT_LOW_VOLTAGE
    - 1 ~ 3600 sec.
  - (1) IVCP_EVENT_LOW_VOLTAGE_HARD
    - 1 ~ 3600 sec.
  - (2) IVCP_EVENT_IGNITION_ON
    - 1 ~ 18000 sec.
  - (3) IVCP_EVENT_IGNITION_OFF_TO_POWER_OFF
    - 1 ~ 18000 sec.
  - (4) IVCP_EVENT_IGNITION_OFF_HARD
    - 1 ~ 18000 sec.
  - (5) IVCP_EVENT_POST_BOOT_POWER_CHECK
    - 1 ~ 18000 sec.

4.3.3.22 ivcp_pm_get_last_wakeup_source

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_last_wakeup_source**(ivcp_pm_wakeup_type *source)
Android	int **ivcp_pm_get_last_wakeup_source**(int[] source)

Description:

Get the last wakeup source.

Parameters:

source [out]

The last wakeup source. The detail please refer to ivcp_pm_wakeup_type section.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.23 ivcp_pm_get_lvp_range

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_lvp_range**(float *min, float *max, float *defualt)
Android	int **ivcp_pm_get_lvp_range**(float[] min, float[] max, float[] default)

Description:

Get low voltage protection control range and default value.

Parameters:

min [out]

The minimum voltage value can be set.

max [out]

The minimum voltage value can be set.

default [out]

The default voltage value.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.24 ivcp_pm_get_lvp_preboot_threshold

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_lvp_preboot_threshold**(float *voltage)
Android	int **ivcp_pm_get_lvp_preboot_threshold**(float[] voltage)

Description:

Get preboot low voltage protection threshold.

Parameters:

voltage [out]

The threshold. The unit is volt.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.25 ivcp_pm_set_lvp_preboot_threshold

Syntax:

Windows / Linux	mrm_err **ivcp_pm_set_lvp_preboot_threshold**(float voltage)
Android	int **ivcp_pm_set_lvp_preboot_threshold**(float voltage)

Description:

Set preboot low voltage protection threshold.

Parameters:

voltage [in]

The threshold. The unit is volt.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.26 ivcp_pm_get_lvp_postboot_threshold

Syntax:

Windows / Linux	mrm_err **ivcp_pm_get_lvp_postboot_threshold**(float *voltage)
Android	int **ivcp_pm_get_lvp_postboot_threshold**(float[] voltage)

Description:

Get postboot low voltage protection threshold.

Parameters:

voltage [out]

The threshold. The unit is volt.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.27 ivcp_pm_set_lvp_postboot_threshold

Syntax:

Windows / Linux	mrm_err **ivcp_pm_set_lvp_postboot_threshold**(float voltage)
Android	int **ivcp_pm_set_lvp_postboot_threshold**(float voltage)

Description:

Set postboot low voltage protection threshold.

Parameters:

voltage [in]

The threshold. The unit is volt.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.28 ivcp_pm_reset_lvp_threshold

Syntax:

Windows / Linux	mrm_err **ivcp_pm_reset_lvp_threshold**(void)
Android	int **ivcp_pm_reset_lvp_threshold**()

Description:

Reset all the low voltage protection threshold to default.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.29 ivcp_pm_lvp_preboot_enable

Syntax:

Windows / Linux	mrm_err **ivcp_pm_lvp_preboot_enable**(void)
Android	int **ivcp_pm_lvp_preboot_enable**()

Description:

Enable preboot low voltage protection.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.30 ivcp_pm_lvp_preboot_disable

Syntax:

Windows / Linux	mrm_err **ivcp_pm_lvp_preboot_disable**(void)
Android	int **ivcp_pm_lvp_preboot_disable**()

Description:

Disable preboot low voltage protection.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.31 ivcp_pm_lvp_postboot_enable

Syntax:

Windows / Linux	mrm_err **ivcp_pm_lvp_postboot_enable**(void)
Android	int **ivcp_pm_lvp_postboot_enable**()

Description:

Enable prostboot low voltage protection.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.32 ivcp_pm_lvp_postboot_disable

Syntax:

Windows / Linux	mrm_err **ivcp_pm_lvp_postboot_disable**(void)
Android	int **ivcp_pm_lvp_postboot_disable**()

Description:

Disable postboot low voltage protection.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.33 ivcp_pm_lvp_preboot_get_status

Syntax:

Windows / Linux	mrm_err **ivcp_pm_lvp_preboot_get_status**(char *status)
Android	int **ivcp_pm_lvp_preboot_get_status**(boolean[] status)

Description:

Get status of preboot low voltage protection.

Parameters:

status [out]

The preboot low voltage protection status. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.34 ivcp_pm_lvp_postboot_get_status

Syntax:

Windows / Linux	mrm_err **ivcp_pm_lvp_postboot_get_status**(char *status)
Android	int **ivcp_pm_lvp_postboot_get_status**(boolean[] status)

Description:

Get status of postboot low voltage protection.

Parameters:

status [out]

The postboot low voltage protection status. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.35 ivcp_pm_set_lvp_poweroff_event

Syntax:

Windows / Linux	mrm_err **ivcp_pm_set_lvp_poweroff_event**(void *poweroff_event))
Android	-

Description:

Set a user define event in order to let IVCP library notify the specified event when car voltage low than threshold. This event only trigger when power off alarm function enable. User can using ivcp_pm_power_off_alarm_enable() to enable this function

Parameters:

poweroff_event [in]

Pointer to the power off event. In windows, the poweroff_event will pointer to a Windows Events HANDLE. In linux, the poweroff_event will pointer to a structure which consists of a pthread_mutex_t and pthread_cond_t.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

You should not release the event before deinitialize IVCP library.

4.3.3.36 ivcp_pm_force_shutdown_enable

Syntax:

Windows / Linux	-
Android	int **ivcp_pm_force_shutdown_enable**()

Description:

Enable force shutdown function.

This function is disabled by default.

If this function is enabled, when device is woken up and ignition status is off (e.g alarm wakeup is triggered), the device will be automatically shutdown after an specified delay time.

You can set delay time by using ivcp_pm_set_force_shutdown_delay() .

To disable this function, you can use ivcp_pm_force_shutdown_disable().

To get the current enable status of this function, you can use ivcp_pm_get_force_shutdown_status().

Parameters:

none

Returns:

IMC_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.37 ivcp_pm_force_shutdown_disable

Syntax:

Windows / Linux	-
Android	int **ivcp_pm_force_shutdown_disable**()

Description:

Disable force shutdown function.

Parameters:

none

Returns:

IMC_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.38 ivcp_pm_get_force_shutdown_status

Syntax:

Windows / Linux	-
Android	int **ivcp_pm_get_force_shutdown_status**(boolean[] status)

Description:

Get the enable status of force shutdown function.

Parameters:

status [out]

The enable status. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.39 ivcp_pm_get_force_shutdown_delay

Syntax:

Windows / Linux	-
Android	int **ivcp_pm_get_force_shutdown_delay**(int[] second)

Description:

Get delay time of force shutdown function.

Parameters:

second [out]

The delay time. The unit is second.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.3.3.40 ivcp_pm_set_force_shutdown_delay

Syntax:

Windows / Linux	-
Android	int **ivcp_pm_set_force_shutdown_delay**(int second)

Description:

Set delay time of force shutdown function.

Parameters:

second [in]

The delay time. The unit is second.

Returns:

- **MRM_ERR_NO_ERROR** - On success.
  Otherwise see the error code list.
- The available range is 1 ~ 3600 sec.

4.4 Watch Dog Functions

4.4.1 Usage

4.4.1.1 Windows/Linux

Please refer to the IVCP Conventions for basic usage.

4.4.1.2 Android

Please refer to the IVCP Conventions for basic usage.

4.4.2 APIs

4.4.2.1 ivcp_watchdog_enable

Syntax:

Windows / Linux	mrm_err **ivcp_watchdog_enable**(void)
Android	int **ivcp_watchdog_enable**()

Description:

Enable watchdog function. CAUTION!! When enable the watchdog timer, watchdog start coutdown. If the coutdown timer equal zero, The computer will auto reboot immediately. You should call ivcp_watchdog_trigger periodicaly in order to prevent the computer reboot. This mechanism can prevent the program stuck and get a chance to repair itself.

The watchdog default is stop. You should manual enable after each boot if you want enable the watchdog function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.4.2.2 ivcp_watchdog_disable

Syntax:

Windows / Linux	mrm_err **ivcp_watchdog_disable**(void)
Android	int **ivcp_watchdog_disable**()

Description:

Disable watchdog function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.4.2.3 ivcp_watchdog_trigger

Syntax:

Windows / Linux	mrm_err **ivcp_watchdog_trigger**(void)
Android	int **ivcp_watchdog_trigger**()

Description:

Trigger watchdog in order to prevent the computer reboot. After called ivcp_watchdog_trigger function the watchdog countdown timer will reset to setting time.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.4.2.4 ivcp_watchdog_get_time

Syntax:

Windows / Linux	mrm_err **ivcp_watchdog_get_time**(unsigned short *second)
Android	int **ivcp_watchdog_get_time**(int[] second)

Description:

Get setting countdown time of watchdog.

Parameters:

second [out]

The countdown time. The unit is second.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.4.2.5 ivcp_watchdog_set_time

Syntax:

Windows / Linux	mrm_err **ivcp_watchdog_set_time**(unsigned short second)
Android	int **ivcp_watchdog_set_time**(int second)

Description:

Set setting countdown time of watchdog.

Parameters:

second [in]

The countdown time. The unit is second.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.4.2.6 ivcp_watchdog_get_current_time

Syntax:

Windows / Linux	mrm_err **ivcp_watchdog_get_current_time**(unsigned short *second)
Android	int **ivcp_watchdog_get_current_time**(int[] second)

Description:

Get current countdown time of watchdog.

Parameters:

second [out]

The current countdown time. The unit is second.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.5 Alarm Functions

4.5.1 Usage

4.5.1.1 Windows/Linux

Please refer to the IVCP Conventions for basic usage.

4.5.1.2 Android

Please refer to the IVCP Conventions for basic usage.

4.5.2 Structure/Classes

4.5.2.1 Windows/Linux

ivcp_real_time_t Structure

Syntax:

Windows / Linux:

```
typedef struct
{
        unsigned short year;
        unsigned char month;
        unsigned char day;
        unsigned char hour;
        unsigned char minute;
        unsigned char second;
        unsigned char week;
} ivcp_alarm_real_time_t;
```

Android:

```
public class IVCP_ALARM_REAL_TIME
{
        public int year;
        public byte month;
        public byte day;
        public byte hour;
        public byte minute;
        public byte second;
        public byte week;
}
```

Description:

This data structure defines the real time of internal RTC.

Members:

year

The year. The valid values for this member are 2000 through 2099.

month

The month. 1 for January, 2 for February..and so on.

day

The day. The valid values for this member are 1 through 31.

hour

The hour. The valid values for this member are 0 through 23.

minute

The minute. The valid values for this member are 0 through 59.

second

The second. The valid values for this member are 0 through 59.

week

The week. 0 for Sunday, 1 for Monday..and so on.

ivcp_alarm_wakeup_time_t Structure

Syntax:

Windows / Linux:

```
typedef struct
{
	unsigned char day_of_week;
	unsigned char hour;
	unsigned char minute;
} ivcp_alarm_wakeup_time_t;
```

Android:

```
public class IVCP_ALARM_WAKEUP_TIME
{
	public byte day_of_week;
	public byte hour;
	public byte minute;
}
```

Description:

This data structure defines the alarm wake time of internal RTC.

Members:

day_of_week

The day of week. 0 for sunday, 1 for Monday..and so on.

hour

The hour. The valid values for this member are 0 through 23.

minute

The minute. The valid values for this member are 0 through 59.

4.5.2.2 Android

IVCP_ALARM_REAL_TIME

class:

mrm.define.IVCP.**IVCP_ALARM_REAL_TIME**

Fields:

Field Name	Type	Input/Output	Comment
year	int	in, out	The year. The valid values for this member are 2000 through 2099.
month	byte	in, out	The month. 1 for January, 2 for February..and so on.
day	byte	in, out	The day. The valid values for this member are 1 through 31.
hour	byte	in, out	The hour. The valid values for this member are 0 through 23.
minute	byte	in, out	The minute. The valid values for this member are 0 through 59.
second	byte	in, out	The second. The valid values for this member are 0 through 59.
week	byte	in, out	The week. 0 for Sunday, 1 for Monday and so on.

IVCP_ALARM_WAKEUP_TIME

class:

mrm.define.IVCP.**IVCP_ALARM_WAKEUP_TIME**

Fields:

Field Name	Type	Input/Output	Comment
day_of_week	byte	in, out	The day of week. 0 for sunday, 1 for Monday..and so on.
hour	byte	in, out	The hour. The valid values for this member are 0 through 23.
minute	byte	in, out	The minute. The valid values for this member are 0 through 59.

4.5.3 Enumeration

4.5.3.1 Windows/Linux

ivcp_alarm_mode Enum

- (0) **IVCP_ALARM_MODE_NO_ALARM** - The computer is not enable alarm wakeup.
- (1) **IVCP_ALARM_MODE_HOURLY** - The computer is wakeup hourly by setting alarm minute.
- (2) **IVCP_ALARM_MODE_DAILY** - The computer is wakeup daily by setting alarm hour and minute.
- (3) **IVCP_ALARM_MODE_WEEKLY** - The computer is wakeup weekly by setting alarm day of week, hour and minute.

4.5.3.2 Android

IVCP_ALARM_MODE

class:

mrm.define.MRM_ENUM.**IVCP_ALARM_MODE**

Enum:

Name	Type	value	Comment
IVCP_ALARM_MODE_NO_ALARM	int	-1	The computer is not enable alarm wakeup.
IVCP_ALARM_MODE_HOURLY	int	0	The computer is wakeup hourly by setting alarm minute.
IVCP_ALARM_MODE_DAILY	int	1	The computer is wakeup daily by setting alarm hour and minute.
IVCP_ALARM_MODE_WEEKLY	int	2	The computer is wakeup weekly by setting alarm day of week, hour and minute.

Remark:

Use the method **getValue()** to get the enum value.

ex: MRM_ENUM.IVCP_ALARM_MODE.IVCP_ALARM_MODE_DAILY**.getValue()** returns 1.

Please refer to sample code for detailed usage.

4.5.4 APIs

4.5.4.1 ivcp_alarm_get_real_time

Syntax:

Windows / Linux	mrm_err **ivcp_alarm_get_real_time**(ivcp_alarm_real_time_t *realTime)
Android	int **ivcp_alarm_get_real_time**(IVCP_ALARM_REAL_TIME realTime)

Description:

Get real time from internal RTC.

Parameters:

realTime [out]

Windows/Linux:

Pointer to ivcp_real_time_t struct which is used to store the gotten values.

Android:

Instance of IVCP_ALARM_REAL_TIME which is used to store the gotten values.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.5.4.2 ivcp_alarm_set_real_time

Syntax:

Windows / Linux	mrm_err **ivcp_alarm_set_real_time**(ivcp_alarm_real_time_t realTime)
Android	int **ivcp_alarm_set_real_time**(IVCP_ALARM_REAL_TIME realTime)

Description:

Set real time to internal RTC.

Parameters:

realTime [in]

Windows/Linux:

Pointer to ivcp_real_time_t struct. You must assign the values to be set to each field of **realTime.**

Android:

Instance of IVCP_ALARM_REAL_TIME. You must assign the values to be set to each field of **realTime.**

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.5.4.3 ivcp_alarm_get_wakeup_time

Syntax:

Windows / Linux	mrm_err **ivcp_alarm_get_wakeup_time**(ivcp_alarm_wakeup_time_t *wakeupTime)
Android	int **ivcp_alarm_get_wakeup_time**(IVCP_ALARM_WAKEUP_TIME wakeupTime)

Description:

Get alarm setting time from internal RTC.

Parameters:

wakeupTime [out]

Windows/Linux:

Pointer to ivcp_alarm_wakeup_time_t struct which is used to store the gotten values.

Android:

Instance of IVCP_ALARM_WAKEUP_TIME which is used to store the gotten values.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.5.4.4 ivcp_alarm_set_wakeup_time

Syntax:

Windows / Linux	mrm_err **ivcp_alarm_set_wakeup_time**(ivcp_alarm_wakeup_time_t wakeupTime)
Android	int **ivcp_alarm_set_wakeup_time**(IVCP_ALARM_WAKEUP_TIME wakeupTime)

Description:

Set alarm setting time from internal RTC.

Parameters:

wakeupTime [out]

Windows/Linux:

Pointer to ivcp_alarm_wakeup_time_t struct. You must assign the values to be set to each field of **wakeupTime.**

Android:

Instance of IVCP_ALARM_WAKEUP_TIME. You must assign the values to be set to each field of **wakeupTime.**

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.5.4.5 ivcp_alarm_get_wakeup_mode

Syntax:

Windows / Linux	mrm_err **ivcp_alarm_get_wakeup_mode**(ivcp_alarm_mode *mode)
Android	int **ivcp_alarm_get_wakeup_mode**(int[] mode)

Description:

Get alarm wakeup mode.

Parameters:

mode [out]

The alarm wakeup mode. The detail please refer to ivcp_alarm_mode.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.5.4.6 ivcp_alarm_set_wakeup_mode

Syntax:

Windows / Linux	mrm_err **ivcp_alarm_set_wakeup_mode**(ivcp_alarm_mode mode)
Android	int **ivcp_alarm_set_wakeup_mode**(int mode)

Description:

Set alarm wakeup mode.

Parameters:

mode [in]

The alarm wakeup mode. The detail please refer to ivcp_alarm_mode.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.5.4.7 ivcp_alarm_wakeup_enable

Syntax:

Windows / Linux	mrm_err **ivcp_alarm_wakeup_enable**(void)
Android	int **ivcp_alarm_wakeup_enable**()

Description:

Enable the alarm wakeup function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.5.4.8 ivcp_alarm_wakeup_disable

Syntax:

Windows / Linux	mrm_err **ivcp_alarm_wakeup_disable**(void)
Android	int **ivcp_alarm_wakeup_disable**()

Description:

Disable the alarm wakeup function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.5.4.9 ivcp_alarm_get_wakeup_status

Syntax:

Windows / Linux	mrm_err **ivcp_alarm_get_wakeup_status**(char *status)
Android	int **ivcp_alarm_get_wakeup_status**(boolean[] status)

Description:

Get status of alarm wakeup function.

Parameters:

status [out]

The alarm wakeup function status. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.6 Digital IO Control Functions

4.6.1 Usage

4.6.1.1 Windows/Linux

Please refer to the IVCP Conventions for basic usage.

4.6.1.2 Android

Please refer to the IVCP Conventions for basic usage.

4.6.2 Enumeration

4.6.2.1 Windpws/Linux

ivcp_dio_pin_id Enum

- (0) **IVCP_DIGITAL_PIN0** - The digital pin 0.
- (1) **IVCP_DIGITAL_PIN1** - The digital pin 1.
- (2) **IVCP_DIGITAL_PIN2** - The digital pin 2.
- (3) **IVCP_DIGITAL_PIN3** - The digital pin 3.
- (4) **IVCP_DIGITAL_PIN4** - The digital pin 4.
- (5) **IVCP_DIGITAL_PIN5** - The digital pin 5.

ivcp_dio_input_type Enum

- (0) **IVCP_DIO_INPUT_TYPE_WET_CONTACT** - Wet contact digital input.
- (1) **IVCP_DIO_INPUT_TYPE_DRY_CONTACT** - Dry contact digital input.

4.6.2.2 Android

IVCP_DIO_PIN_ID

class:

mrm.define.MRM_ENUM.**IVCP_DIO_PIN_ID**

Enum:

Name	Type	value	Comment
IVCP_DIGITAL_PIN0	int	0	The digital pin 0.
IVCP_DIGITAL_PIN1	int	1	The digital pin 1.
IVCP_DIGITAL_PIN2	int	2	The digital pin 2.
IVCP_DIGITAL_PIN3	int	3	The digital pin 3.
IVCP_DIGITAL_PIN4	int	4	The digital pin 4.
IVCP_DIGITAL_PIN5	int	5	The digital pin 5.

Remark:

Use the method **getValue()** to get the enum value.

ex: MRM_ENUM.IVCP_DIO_PIN_ID.IVCP_DIGITAL_PIN1**.getValue()** returns 1.

Please refer to sample code for detailed usage.

IVCP_DIO_INPUT_TYPE

class:

mrm.define.MRM_ENUM**.IVCP_DIO_INPUT_TYPE**

Enum:

Name	Type	value	Comment
IVCP_DIO_INPUT_TYPE_WET_CONTACT	int	0	Wet contact digital input.
IVCP_DIO_INPUT_TYPE_DRY_CONTACT	int	1	Dry contact digital input.

Remark:

Use the method **getValue()** to get the enum value.

ex: MRM_ENUM.IVCP_DIO_INPUT_TYPE.IVCP_DIO_INPUT_TYPE_DRY_CONTACT**.getValue()** returns 1.

Please refer to sample code for detailed usage.

4.6.3 APIs

4.6.3.1 ivcp_dio_get_input_number

Syntax:

Windows / Linux	mrm_err **ivcp_dio_get_input_number**(char *number)
Android	int **ivcp_dio_get_input_number**(byte[] number)

Description:

Get digital input supported number.

Parameters:

number [out]

The digital input supported number.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.6.3.2 ivcp_dio_get_output_number

Syntax:

Windows / Linux	mrm_err **ivcp_dio_get_output_number**(char *number)
Android	int **ivcp_dio_get_output_number**(byte[] number)

Description:

Get digital output supported number.

Parameters:

number [out]

The digital output supported number.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.6.3.3 ivcp_dio_read_input

Syntax:

Windows / Linux	mrm_err **ivcp_dio_read_input**(ivcp_dio_pin_id id, char *status)
Android	int **ivcp_dio_read_input**(int id, boolean[] status)

Description:

Read the specifies digital input pin status.

Parameters:

id [in]

The specified input pin.

For Windows/Linux, please refer to ivcp_dio_pin_id.

For Android, please refer to IVCP_DIO_PIN_ID.

status [out]

The specifies pin status. The value 1 is High otherwise 0 is Low.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.6.3.4 ivcp_dio_read_input_multiple

Syntax:

Windows / Linux	mrm_err **ivcp_dio_read_input_multiple**(unsigned int *status)
Android	int **ivcp_dio_read_input_multiple**(int[] status)

Description:

Read the all digital output pin status. The bit-mask indicate which pin changing in this operation. The bit-0 is **IVCP_DIGITAL_PIN0**, bit-1 is **IVCP_DIGITAL_PIN1, etc. T**he detail please refer to ivcp_dio_pin_id section.

For example, if you want get **IVCP_DIGITAL_PIN0** and **IVCP_DIGITAL_PIN1** the bit-mask will be 0x000000003.

Parameters:

status [out]

The specifies pin status using bit-mask. The bit value 1 is High otherwise 0 is Low.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.6.3.5 ivcp_dio_write_output

Syntax:

Windows / Linux	mrm_err **ivcp_dio_write_output**(ivcp_dio_pin_id id, char status)
Android	int **ivcp_dio_write_output**(int id, boolean status)

Description:

Control the specifies digital output pin status.

Parameters:

id [in]

The specified input pin.

For Windows/Linux, please refer to ivcp_dio_pin_id.

For Android, please refer to IVCP_DIO_PIN_ID.

status [in]

The specifies pin status. The value 1 is High otherwise 0 is Low.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.6.3.6 ivcp_dio_read_output

Syntax:

Windows / Linux	mrm_err **ivcp_dio_read_output**(ivcp_dio_pin_id id, char *status)
Android	int **ivcp_dio_read_output**(int id, boolean[]status)

Description:

Read the specifies digital output pin status.

Parameters:

id [in]

The specified output pin.

For Windows/Linux, please refer to ivcp_dio_pin_id.

For Android, please refer to IVCP_DIO_PIN_ID.

status [out]

The specifies pin status. The value 1 is High otherwise 0 is Low.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.6.3.7 ivcp_dio_get_input_type

Syntax:

Windows / Linux	-
Android	int **ivcp_dio_get_input_type**(int[] type)

Description:

Get current digital input type.

Parameters:

type [out]

The digital input type enum value.

For Android, Please refer to IVCP_DIO_INPUT_TYPE.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.6.3.8 ivcp_dio_set_input_type

Syntax:

Windows / Linux	-
Android	int **ivcp_dio_set_input_type**(int type)

Description:

Set digital input type.

Parameters:

type [in]

The digital input type enum value.

For Android, Please refer to IVCP_DIO_INPUT_TYPE.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.6.3.9 ivcp_dio_get_pin_input_type

Syntax:

Windows / Linux	-
Android	int **ivcp_dio_get_pin_input_type**(int id, int[] type)

Description:

Get the specifies digital input pin type.

Parameters:

id [in]

The specified input pin.

For Windows/Linux, please refer to ivcp_dio_pin_id.

For Android, please refer to IVCP_DIO_PIN_ID.

type [out]

The digital input type enum value.

For Android, Please refer to IVCP_DIO_INPUT_TYPE.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.6.3.10 ivcp_dio_set_pin_input_type

Syntax:

Windows / Linux	-
Android	int **ivcp_dio_set_pin_input_type**(int id, int type)

Description:

Set the specifies digital input pin type.

Parameters:

id [in]

The specified input pin.

For Windows/Linux, please refer to ivcp_dio_pin_id.

For Android, please refer to IVCP_DIO_PIN_ID.

type [in]

The digital input type enum value.

For Android, Please refer to IVCP_DIO_INPUT_TYPE.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.6.3.11 ivcp_dio_get_reference_voltage

Syntax:

Windows / Linux	-
Android	int **ivcp_dio_get_reference_voltage**(float[] voltage)

Description:

Get current reference voltage.

Parameters:

voltage [out]

The reference voltage value.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.6.3.12 ivcp_dio_set_reference_voltage

Syntax:

Windows / Linux	-
Android	int **ivcp_dio_set_reference_voltage**(float voltage)

Description:

Set reference voltage.

Parameters:

voltage [in]

The reference voltage value.

Returns:

- **MRM_ERR_NO_ERROR** - On success.
  Otherwise see the error code list.

Remarks:

- The available range for reference voltage is 0V ~ 3.3V.
- The IVCP firmware configures the reference voltage from 0V to 3.3V with 4096 levels. 0.0008056V for each level. **ivcp_dio_set_reference_voltage()** helps you to convert the **voltage** float value parameter to appropriate integer level, and set the level to IVCP.
- Due to that the voltage float value set/get through **ivcp_dio_set_reference_voltage()/ivcp_dio_get_reference_voltage()** will be converted to/from IVCP firmware reference voltage levels, the voltage float value you set/get through **ivcp_dio_set_reference_voltage()/ivcp_dio_get_reference_voltage()** might not be exactly the same.

4.7 Battery Functions

4.7.1 Usage

4.7.1.1 Windows/Linux

Please refer to the IVCP Conventions for basic usage.

4.7.1.2 Android

Please refer to the IVCP Conventions for basic usage.

4.7.2 APIs

4.7.2.1 ivcp_battery_get_status

Syntax:

Windows / Linux	mrm_err **ivcp_battery_get_status**(char *status)
Android	int **ivcp_battery_get_status**(boolean[] status)

Description:

Get backup battery exist status.

Parameters:

status [out]

Backup power exists status. The value is 1(true) if exists, 0(false) if not exists.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.7.2.2 ivcp_battery_get_voltage

Syntax:

Windows / Linux	mrm_err **ivcp_battery_get_voltage**(int *mV)
Android	int **ivcp_battery_get_voltage**(int[] mV)

Description:

Get backup battery current voltage.

Parameters:

voltage [out]

Voltage of backup battery. The unit is millivolts (mV). The value is 0~65535.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.7.2.3 ivcp_battery_get_average_current

Syntax:

Windows / Linux	mrm_err **ivcp_battery_get_average_current**(int *mA)
Android	int **ivcp_battery_get_average_current**(int[] mA)

Description:

Get backup battery average current.

Parameters:

voltage [out]

Average current of backup battery. The value is -32768 ~ 32727. The unit is milli Amps(mA).

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.7.2.4 ivcp_battery_get_state_of_charge

Syntax:

Windows / Linux	mrm_err **ivcp_battery_get_state_of_charge**(int *percentage)
Android	int **ivcp_battery_get_state_of_charge**(int[] percentage)

Description:

Get backup battery current state of charge.

Parameters:

percentage [out]

Backup power current state of charge. The unit is percentage(%). The value is 0~100 %.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.7.2.5 ivcp_battery_get_time_to_empty

Syntax:

Windows / Linux	mrm_err **ivcp_battery_get_time_to_empty**(int *minute)
Android	int **ivcp_battery_get_time_to_empty**(int[] minute)

Description:

Get backup battery time to empty. The unit is minute.

Parameters:

minute [out]

Backup power time to empty. The value is 0~65535.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.7.2.6 ivcp_battery_get_temperature

Syntax:

Windows / Linux	mrm_err **ivcp_battery_get_temperature**(float *temperature)
Android	int **ivcp_battery_get_temperature**(float[] temperature)

Description:

Get backup battery temperature. The unit is Celsius.

Parameters:

temperature [out]

Backup power temperature. The value is 0~6280.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8 G-Sensor Functions

4.8.1 Usage

4.8.1.1 Basic Usage

Windows/Linux

Please refer to the IVCP Conventions for basic usage.

Android

Please refer to the IVCP Conventions for basic usage.

4.8.1.2 G Sensor Alarm Event

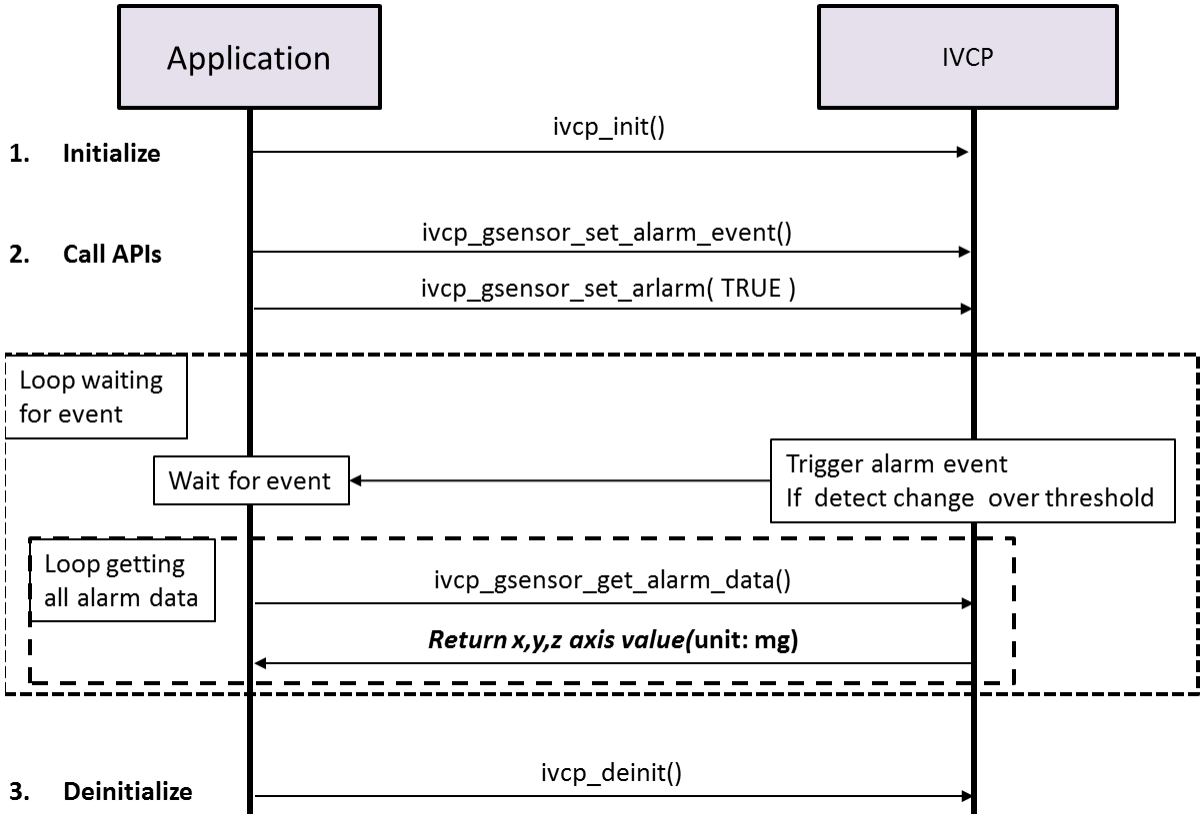
IVCP is able to send an alarm event to your APP when G sensor senses impact on the device.
The following sections describes the usage of related APIs.

Windows/Linux

At first, you can use ivcp_gsensor_set_alarm_threshold() to set alarm threshold and use ivcp_gsensor_get_alarm_threshold() to check current alarm threshold value.

After threshold is set, to enable the G sensor alarm event mechanism, you need to call ivcp_gsensor_set_alarm_event() to register an alarm event to IVCP library. Then call ivcp_gsensor_set_arlarm(true) to enable alarm event function of VPM.
When G-sensor detect sensor data changes over specified threshold, the registered event will be triggered.

You need to hold an event listening loop in your APP to handle the registered event. SDK holds an internal buffer to store the alarm data. When the event is triggered, you can run loop to calling ivcp_gsensor_get_alarm_data() to consume the alarm data from the buffer.
After all alarm data are consumed, you should keep listening the registered event.
Please refer to the sample code for the details of implementation.



Android

At first, you can use ivcp_gsensor_set_alarm_threshold() to set alarm threshold and use ivcp_gsensor_get_alarm_threshold() to check current alarm threshold value.

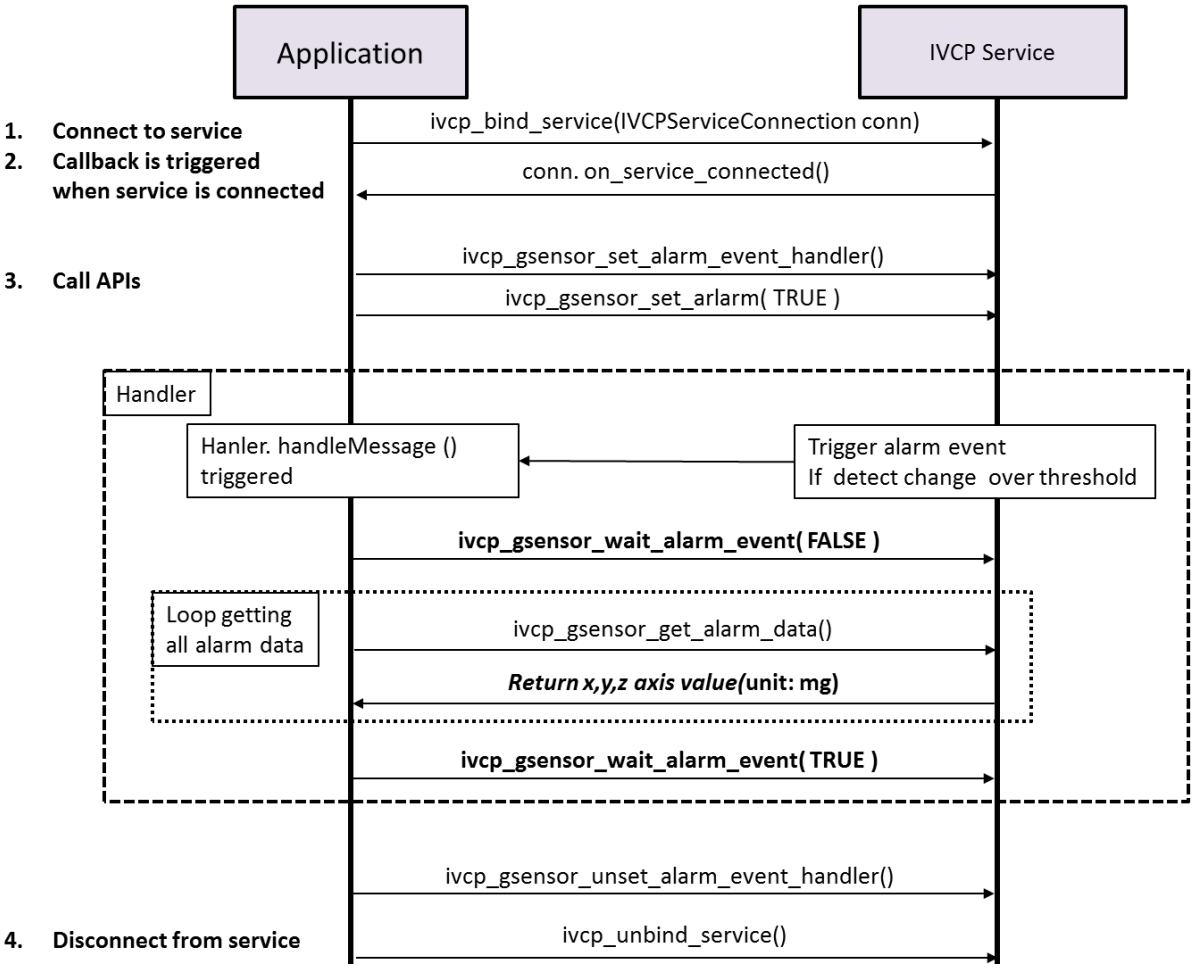
MRM IVCP SDK leverage the Android Handler mechanism to inform G sensor alarm event. To enable the G sensor alarm event mechanism in Android, you need to create an instance of Handler and call ivcp_gsensor_set_alarm_event_handler() to register the Handler instance to SUSI service.
Then call ivcp_gsensor_set_arlarm(true) to enable alarm event function of VPM.

When G-sensor detect sensor data changes over specified threshold, The registered event will be triggered and the handleMessage() callback of registered handler instance will be triggered.
SDK holds an internal buffer to store the alarm data. When the event is triggered, you can run loop to calling ivcp_gsensor_get_alarm_data() to consume the alarm data from the buffer.

Due to that IVCP service will trigger alarm event when it get a alarm data, to avoid from getting multiple unnecessary alarm events, you should call ivcp_gsensor_wait_alarm_event(FALSE) to ask IVCP service temporarily stop pass alarm event to handler before your APP start the alarm data getting loop. After your APP gets all alarm data, you should call ivcp_gsensor_wait_alarm_event(TRUE) to ask IVCP service continue the event informing.

If you do not need to listen to alarm event anymore, you should call ivcp_gsensor_unset_alarm_event_handler() to unregister the Handler instance from IVCP service.

Please refer to the sample code for the details of implementation.


Application
IVCP Service
1. Connect to service
2. Callback is triggered when service is connected
ivcp_bind_service(IVCPServiceConnection conn)
conn. on_service_connected()
3. Call APIs
ivcp_gsensor_set_alarm_event_handler()
ivcp_gsensor_set_arlarm(TRUE)
Handler
Hanler. handleMessage () triggered
Trigger alarm event If detect change over threshold
ivcp_gsensor_wait_alarm_event(FALSE)
Loop getting all alarm data
ivcp_gsensor_get_alarm_data()
Return x,y,z axis value(unit: mg)
ivcp_gsensor_wait_alarm_event(TRUE)
ivcp_gsensor_unset_alarm_event_handler()
4. Disconnect from service
ivcp_unbind_service()

4.8.2 Structure/Classes

4.8.2.1 Windows/Linux

ivcp_gsensor_value_t Structure

Syntax:

```
typedef struct
{
        int x_mg;
        int y_mg;
        int z_mg;
} ivcp_gsensor_value_t;
```

Description:

This data structure defines the each 3-axis static acceleration of gravity value of G-Sensor.

Members:

x_mg

The x-axis acceleration of gravity value. The unit is mg.

y_mg

The y-axis acceleration of gravity value. The unit is mg.

z_mg

The z-axis acceleration of gravity value. The unit is mg.

4.8.2.2 Android

IVCP_GSENSOR_VALUE

class:

mrm.define.IVCP.**IVCP_GSENSOR_VALUE**

Description:

This data structure defines the each 3-axis static acceleration of gravity value of G-Sensor.

Fields:

Field Name	Type	Input/Output	Comment
x_mg	int	in, out	The x-axis acceleration of gravity value. The unit is mg.
y_mg	int	in, out	The y-axis acceleration of gravity value. The unit is mg.
z_mg	int	in, out	The z-axis acceleration of gravity value. The unit is mg.

4.8.3 Enumeration

4.8.3.1 Windows/Linux

ivcp_gsensor_res Enum

- (0) **IVCP_GSENSOR_RES_2G** - The G-Sensor resolution of +2g/-2g.
- (1) **IVCP_GSENSOR_RES_4G** - The G-Sensor resolution of +4g/-4g.
- (2) **IVCP_GSENSOR_RES_8G** - The G-Sensor resolution of +8g/-8g.
- (3) **IVCP_GSENSOR_RES_16G** - The G-Sensor resolution of +16g/-16g.

Remarks:

- For Android, you can get the enumeration value from the class **mrm.define.MRM_ENUM.IVCP_GSENSOR_RES.**
  (ex: MRM_ENUM.IVCP_GSENSOR_RES.**IVCP_GSENSOR_RES_2G**.getValue())
  Please refer to sample code for detailed usage.

4.8.3.2 Android

IVCP_GSENSOR_RES

class:

mrm.define.MRM_ENUM.IVCP_GSENSOR_RES

Enum:

Name	Type	value	Comment
IVCP_GSENSOR_RES_2G	int	0	The G-Sensor resolution of +2g/-2g.
IVCP_GSENSOR_RES_4G	int	1	The G-Sensor resolution of +4g/-4g.
IVCP_GSENSOR_RES_8G	int	2	The G-Sensor resolution of +8g/-8g.
IVCP_GSENSOR_RES_16G	int	3	The G-Sensor resolution of +16g/-16g.

Remark:

Use the method **getValue()** to get the enum value.

ex: MRM_ENUM.IVCP_GSENSOR_RES.IVCP_GSENSOR_RES_4G**.getValue()** returns 1.

Please refer to sample code for detailed usage.

4.8.4 Constant

4.8.4.1 Android

Class:
mrm.define.**MRM_CONSTANTS**

Fields:

Field Name	Type	Value
IVCP_EVENT_ID_UNKNOWN	int	-1
IVCP_EVENT_ID_GSENSOR_ALARM	int	0

4.8.5 APIs

4.8.5.1 ivcp_gsensor_available

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_available**(char *supported)
Android	int **ivcp_gsensor_available**(boolean[] supported)

Description:

Get supported status of G-Sensor.

Parameters:

supported [out]

The G-Sensor supported status. The value 1 is supported on this platform otherwise 0 is unsupported.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.2 ivcp_gsensor_enable

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_enable**(void)
Android	int **ivcp_gsensor_enable**()

Description:

Enable G-Sensor function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.3 ivcp_gsensor_disable

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_disable**(void)
Android	int **ivcp_gsensor_disable**()

Description:

Disable G-Sensor function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.4 ivcp_gsensor_get_status

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_get_status**(char *status)
Android	int **ivcp_gsensor_get_status**(boolean[] status)

Description:

Get status of G-Sensor.

Parameters:

status [out]

The G-Sensor status. The unit is second. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.5 ivcp_gsensor_read

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_read**(ivcp_gsensor_value_t *value)
Android	int **ivcp_gsensor_read**(IVCP_GSENSOR_VALUE value)

Description:

Get acceleration of gravity value from G-Sensor. You should enable G-Sensor before using this function otherwise you may get a error code(**MRM_ERR_IVCP_GSENSOR_DATA_INVALID**) and meaningless value.

Parameters:

value [out]

Windows/Linux:

Pointer to ivcp_gsensor_value_t struct which is used to store the gotten values.

Android:

A instance of IVCP_GSENSOR_VALUE which is used to store the gotten values.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.6 ivcp_gsensor_get_resolution

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_get_resolution**(ivcp_gsensor_res *res)
Android	int **ivcp_gsensor_get_resolution**(int[] res)

Description:

Get acceleration's resolution of G-Sensor.

Parameters:

res [out]

The acceleration's resolution. Please refer to .ivcp_gsensor_res.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.7 ivcp_gsensor_set_resolution

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_set_resolution**(ivcp_gsensor_res res)
Android	int **ivcp_gsensor_set_resolution**(int res)

Description:

Set acceleration's resolution of G-Sensor. The higher, the more inaccurate.

Parameters:

res [in]

The acceleration's resolution. Please refer to .ivcp_gsensor_res.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.8 ivcp_gsensor_wakeup_enable

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_wakeup_enable**(void)
Android	int **ivcp_gsensor_wakeup_enable**()

Description:

Enable G-Sensor wakeup function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.9 ivcp_gsensor_wakeup_disable

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_wakeup_disable**(void)
Android	int **ivcp_gsensor_wakeup_disable**()

Description:

Disable G-Sensor wakeup function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.10 ivcp_gsensor_get_wakeup_status

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_get_wakeup_status**(char *status)
Android	int **ivcp_gsensor_get_wakeup_status**(boolean[] status)

Description:

Get status of G-Sensor wakeup function.

Parameters:

status [out]

G-Sensor wakeup function status. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.11 ivcp_gsensor_get_wakeup_threshold

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_get_wakeup_threshold**(int *mg)
Android	int **ivcp_gsensor_get_wakeup_threshold**(int[] mg)

Description:

Get acceleration wakeup threshold.

Parameters:

mg [out]

The acceleration wakeup threshold. The unit is mg.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.12 ivcp_gsensor_set_wakeup_threshold

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_set_wakeup_threshold**(int mg)
Android	int **ivcp_gsensor_set_wakeup_threshold**(int mg)

Description:

Set acceleration wakeup threshold. Threshold scale only 62.5 mg, value will be automatically rounded up.The range threshold is 63mg to 16000mg.

Parameters:

mg [in]

The acceleration wakeup threshold. The unit is mg.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.13 ivcp_gsensor_set_alarm_event

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_set_alarm_event**(void *gsensor_alarm_event)
Android	-

Description:

Set a user define event in order to let IVCP library notify the specifies event when G-sensor alarm is triggered.

Parameters:

gsensor_alarm_event [in]

Pointer to the G-sensor alarm trigger event. In windows, the gsensor_alarm_event will pointer to a Windows Events HANDLE. In linux, the gsensor_alarm_event will pointer to a structure which consists of a pthread_mutex_t and pthread_cond_t.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

You should not close the event before deinitialization IVCP library.

4.8.5.14 ivcp_gsensor_set_alarm_event_handler

Syntax:

Windows / Linux	-
Android	int **ivcp_gsensor_set_alarm_event_handler**(Handler handler)

Description:

Set a Handler which handles the event from IVCP Service when G-sensor alarm is triggered.

Parameters:

handler [in]

Handler instance which handles the event from IVCP Service when G-sensor alarm is triggered.

The Handler's handlerMessage() call back will receive a Message with "what" field equals to IVCP_EVENT_ID_GSENSOR_ALARM

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

Please refer to the section G Sensor Alarm Event for the usage.

4.8.5.15 ivcp_gsensor_unset_alarm_event_handler

Syntax:

Windows / Linux	-
Android	int **ivcp_gsensor_unset_alarm_event_handler**()

Description:

Unset a registered Handler which handles the G sensor alarm event from IVCP Service.

Parameters:

-

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

Please refer to the section G Sensor Alarm Event for the usage.

4.8.5.16 ivcp_gsensor_wait_alarm_event

Syntax:

Android	int **ivcp_gsensor_wait_alarm_event**(boolean status)

Description:

Allow/Disallow IVCP service to pass G sensor alarm event to registered Handler when alarm event is triggered.

Parameters:

status [in]

The status of whether IVCP service should pass G sensor alarm event to registered Handler

TRUE: Inform

FLASE: Not to inform

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

Please refer to the section G Sensor Alarm Event for the usage.

4.8.5.17 ivcp_gsensor_get_alarm_threshold

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_get_alarm_threshold**(int *mg)
Android	int **ivcp_gsensor_get_alarm_threshold**(int[] mg)

Description:

Get G-sensor alarm trigger threshold.

Parameters:

mg [out]

The G-sensor alarm trigger threshold. This threshold The unit is mg.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.18 ivcp_gsensor_set_alarm_threshold

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_set_alarm_threshold**(int mg)
Android	int **ivcp_gsensor_set_alarm_threshold**(int mg)

Description:

Set G-sensor alarm trigger threshold. The available range is 2000mg to 16000mg. Threshold should be multiple of 62.5 mg or the value will be automatically rounded up.

Parameters:

mg [in]

The acceleration wakeup threshold. The unit is mg.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.19 ivcp_gsensor_get_alarm_data

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_get_alarm_data**(ivcp_gsensor_value_t *value)
Android	int **ivcp_gsensor_get_alarm_data**(IVCP_GSENSOR_VALUE value)

Description:

Get the trigger acceleration of gravity value from library queue. The library will keep the last alarm trigger value to queue. If user not get the alarm value, the queue will be overwritten. return code **MRM_ERR_IVCP_GSENSOR_DATA_NOT_READY** mean the queue is empty.

Parameters:

value [out]

Windows/Linux:

Pointer to ivcp_gsensor_value_t struct which is used to store the gotten values.

Android:

A instance of IVCP_GSENSOR_VALUE which is used to store the gotten values.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.20 ivcp_gsensor_get_arlarm

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_get_arlarm**(char *enable)
Android	int **ivcp_gsensor_get_alarm_threshold**(int[] mg)

Description:

Get status of alarm trigger function.

Parameters:

enable [out]

The alarm trigger function status. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.8.5.21 ivcp_gsensor_set_arlarm

Syntax:

Windows / Linux	mrm_err **ivcp_gsensor_set_arlarm**(char enable)
Android	int **ivcp_gsensor_set_alarm_threshold**(int mg)

Description:

Enable or disable the alarm trigger function.

Parameters:

enable [in]

The alarm trigger function status. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.9 P-Sensor Functions

4.9.1 Usage

4.9.1.1 Windows/Linux

Please refer to the IVCP Conventions for basic usage.

4.9.1.2 Android

Please refer to the IVCP Conventions for basic usage.

4.9.2 APIs

4.9.2.1 ivcp_psensor_available

Syntax:

Windows / Linux	mrm_err **ivcp_psensor_available**(char *supported)
Android	int **ivcp_psensor_available**(boolean[] supported)

Description:

Get supported status of P-Sensor.

Parameters:

supported [out]

The P-Sensor supported status. The value 1 is supported on this platform otherwise 0 is unsupported.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.9.2.2 ivcp_psensor_enable

Syntax:

Windows / Linux	mrm_err **ivcp_psensor_enable**(void)
Android	int **ivcp_psensor_enable**()

Description:

Enable P-Sensor function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.9.2.3 ivcp_psensor_disable

Syntax:

Windows / Linux	mrm_err **ivcp_psensor_disable**(void)
Android	int **ivcp_psensor_disable**()

Description:

Disable P-Sensor function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.9.2.4 ivcp_psensor_get_status

Syntax:

Windows / Linux	mrm_err **ivcp_psensor_get_status**(char *status)
Android	int **ivcp_psensor_get_status**(boolean[] status)

Description:

Get status of P-Sensor.

Parameters:

status [out]

The P-Sensor status. The unit is second. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.9.2.5 ivcp_psensor_get_pressure

Syntax:

Windows / Linux	mrm_err **ivcp_psensor_get_pressure**(int *mbar)
Android	int **ivcp_psensor_get_pressure**(int[] mbar)

Description:

Get millibar from P-Sensor. You should enable P-Sensor before using this function otherwise you may get a error code(**MRM_ERR_IVCP_PSENSOR_DATA_INVALID**) and meaningless value.

Parameters:

mbar [out]

The millibar.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.10 Sensor Functions

4.10.1 Usage

4.10.1.1 Windows/Linux

Please refer to the IVCP Conventions for basic usage.

4.10.2 APIs

4.10.2.1 ivcp_sensor_read_cpu_temperature

Syntax:

Windows / Linux	mrm_err **ivcp_sensor_read_cpu_temperature**(char *val)
Android	-

Description:

Get internal CPU Temperature. only support partial platform.

Parameters:

val [out]

Internal CPU Temperature. The unit is 1 Celsius.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

- IOPort driver need 1.4+ version.

4.10.2.2 ivcp_sensor_read_system_temperature1

Syntax:

Windows / Linux	mrm_err **ivcp_sensor_read_system_temperature1**(char *val)
Android	-

Description:

Get system temperature. only support partial platform.

Parameters:

val [out]

System Temperature of CPU. The unit is 1 Celsius.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11 Peripheral Control Functions

4.11.1 Usage

4.11.1.1 Windows/Linux

Please refer to the IVCP Conventions for basic usage.

4.11.1.2 Android

Please refer to the IVCP Conventions for basic usage.

4.11.2 Structure/Classes

4.11.2.1 Windows/Linux

ivcp_peripheral_comport_mode Structure

Syntax:

Windows:

```
typedef struct
{
        ivcp_peripheral_comport_modes comport_mode;
        char term_status;
} ivcp_peripheral_comport_mode_t;
```

Description:

This data structure defines the com port and com port mode.

Members:

comport_mode

The com port mode. Current support RS232,RS485,RS422.

term_status

com port termination status.

4.11.3 Enumeration

4.11.3.1 Windows/Linux

ivcp_peripheral_comport_modes Enum

- (0) **IVCP_PERIPHERAL_COMPORT_MODE_RS232** - The com port mode using RS232.
- (1) **IVCP_PERIPHERAL_COMPORT_MODE_RS485** - The com port mode using RS485.
- (2) **IVCP_PERIPHERAL_COMPORT_MODE_RS422** - The com port mode using RS422.

ivcp_peripheral_comport Enum

- (0) **IVCP_PERIPHERAL_COMPORT_A** - IVCP using com port A.
- (1) **IVCP_PERIPHERAL_COMPORT_B** - IVCP using com port B.
- (2) **IVCP_PERIPHERAL_COMPORT_C** - IVCP using com port C.
- (3) **IVCP_PERIPHERAL_COMPORT_D** - IVCP using com port D.
- (4) **IVCP_PERIPHERAL_COMPORT_E** - IVCP using com port E.
- (5) **IVCP_PERIPHERAL_COMPORT_F** - IVCP using com port F.

ivcp_peripheral_control_type Enum

- (0) **IVCP_PERIPHERAL_WWAN_POWER** - The control type of WWAN power.
- (1) **IVCP_PERIPHERAL_WIFI_POWER** - The control type of WIFI power.
- (2) **IVCP_PERIPHERAL_GPS_POWER** - The control type of GPS power.
- (3) **IVCP_PERIPHERAL_SIM_SWITCH** - The control type of SIM card switch.
- (4) **IVCP_PERIPHERAL_REARVIEW_SWITCH** - The control type of rear view switch.
- (5) **IVCP_PERIPHERAL_MIC_SWITCH** - The control type of MIC switch.

ivcp_peripheral_power_id Enum

- (0) **IVCP_PERIPHERAL_PID_WWAN** - The power control ID of WWAN.
- (1) **IVCP_PERIPHERAL_PID_WIFI** - The power control ID of WIFI.
- (2) **IVCP_PERIPHERAL_PID_GPS** - The power control ID of GPS.

ivcp_peripheral_rearview_id Enum

- (0) **IVCP_PERIPHERAL_RID_MAIN** - Display to LVDS view.
- (1) **IVCP_PERIPHERAL_RID_EXTERNEL1** - Display to Camera view.
- (2) **IVCP_PERIPHERAL_RID_EXTERNEL2** - reserve.
- (3) **IVCP_PERIPHERAL_RID_EXTERNEL3** - reserve.

ivcp_peripheral_voice_input Enum

- (0) **IVCP_PERIPHERAL_VOICE_INPUT_FRONT_MIC** - Voice from Front Panel MIC.
- (1) **IVCP_PERIPHERAL_VOICE_INPUT_MIC_IN** - Voice from MIC-IN. Normally on the HDC or GIO cable.
- (2) **IVCP_PERIPHERAL_VOICE_BLUETOOTH** - Voice from Bluetooth handset.

4.11.3.2 Android

IVCP_PERIPHERAL_COMPORT_MODES

class:

mrm.define.MRM_ENUM.**IVCP_PERIPHERAL_COMPORT_MODES**

Enum:

Name	Type	value	Comment
IVCP_PERIPHERAL_COMPORT_MODE_RS232	int	0	The com port mode using RS232.
IVCP_PERIPHERAL_COMPORT_MODE_RS485	int	1	The com port mode using RS485.
IVCP_PERIPHERAL_COMPORT_MODE_RS422	int	2	The com port mode using RS422.

Remark:

Use the method **getValue()** to get the enum value.

ex:

MRM_ENUM.**IVCP_PERIPHERAL_COMPORT_MODES**.IVCP_PERIPHERAL_COMPORT_MODE_RS485**.getValue()** returns 1.

Please refer to sample code for detailed usage.

IVCP_PERIPHERAL_COMPORT

class:

mrm.define.MRM_ENUM.**IVCP_PERIPHERAL_COMPORT**

Enum:

Name	Type	value	Comment
IVCP_PERIPHERAL_COMPORT_A	int	0	IVCP using com port A.
IVCP_PERIPHERAL_COMPORT_B	int	1	IVCP using com port B.
IVCP_PERIPHERAL_COMPORT_C	int	2	IVCP using com port C.
IVCP_PERIPHERAL_COMPORT_D	int	3	IVCP using com port D.
IVCP_PERIPHERAL_COMPORT_E	int	4	IVCP using com port E.
IVCP_PERIPHERAL_COMPORT_F	int	5	IVCP using com port F.

Remark:

Use the method **getValue()** to get the enum value.

ex: MRM_ENUM.IVCP_PERIPHERAL_COMPORT.IVCP_PERIPHERAL_COMPORT_B**.getValue()** returns 1.

Please refer to sample code for detailed usage.

IVCP_PERIPHERAL_CONTROL_TYPE

class:

mrm.define.MRM_ENUM.**IVCP_PERIPHERAL_CONTROL_TYPE**

Enum:

Name	Type	value	Comment
IVCP_PERIPHERAL_WWAN_POWER	int	0	The control type of WWAN power.
IVCP_PERIPHERAL_WIFI_POWER	int	1	The control type of WIFI power.
IVCP_PERIPHERAL_GPS_POWER	int	2	The control type of GPS power.
IVCP_PERIPHERAL_SIM_SWITCH	int	3	The control type of SIM card switch.
IVCP_PERIPHERAL_REARVIEW_SWITCH	int	4	The control type of rear view switch.

Remark:

Use the method **getValue()** to get the enum value.

ex: MRM_ENUM.IVCP_PERIPHERAL_CONTROL_TYPE.IVCP_PERIPHERAL_WIFI_POWER**.getValue()** returns 1.

Please refer to sample code for detailed usage.

IVCP_PERIPHERAL_POWER_ID

mrm.define.MRM_ENUM.**IVCP_PERIPHERAL_POWER_ID**

Enum:

Name	Type	value	Comment
IVCP_PERIPHERAL_PID_WWAN	int	0	The power control ID of WWAN.
IVCP_PERIPHERAL_PID_WIFI	int	1	The power control ID of WIFI.
IVCP_PERIPHERAL_PID_GPS	int	2	The power control ID of GPS.

Remark:

Use the method **getValue()** to get the enum value.

ex: MRM_ENUM.IVCP_PERIPHERAL_POWER_ID.IVCP_PERIPHERAL_PID_WIFI**.getValue()** returns 1.

Please refer to sample code for detailed usage.

IVCP_PERIPHERAL_REARVIEW_ID

class:

mrm.define.MRM_ENUM.**IVCP_PERIPHERAL_REARVIEW_ID**

Enum:

Name	Type	value	Comment
IVCP_PERIPHERAL_RID_MAIN	int	0	Display to LVDS view.
IVCP_PERIPHERAL_RID_EXTERNEL1	int	1	Display to Camera view.
IVCP_PERIPHERAL_RID_EXTERNEL2	int	2	reserve
IVCP_PERIPHERAL_RID_EXTERNEL3	int	3	reserve

Remark:

Use the method **getValue()** to get the enum value.

ex:

MRM_ENUM.IVCP_PERIPHERAL_REARVIEW_ID.IVCP_PERIPHERAL_RID_EXTERNEL1**.getValue()**

returns 1.

Please refer to sample code for detailed usage.

4.11.4 APIs

4.11.4.1 ivcp_peripheral_control_available

Syntax:

Windows / Linux	mrm_err **ivcp_peripheral_control_available**(ivcp_peripheral_control_type type, char *supported)
Android	int **ivcp_peripheral_control_available**(int controlType, out boolean[] supported)

Description:

Get supported status of control type.

Parameters:

id [in]

The type of peripheral device.

For Windows/Linux:, please refer to ivcp_peripheral_control_type.

For Android, please refer to IVCP_PERIPHERAL_CONTROL_TYPE.

supported [out]

The control type supported status. The value 1 is supported on this platform otherwise 0 is unsupported.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11.4.2 ivcp_peripheral_power_on

Syntax:

Windows / Linux	mrm_err **ivcp_peripheral_power_on**(ivcp_peripheral_power_id id)
Android	int **ivcp_peripheral_power_on**(int id)

Description:

Turn on the power of specifies device.

Parameters:

id [in]

The power ID of peripheral device.

For Windows/Linux, please refer to ivcp_peripheral_power_id .

For Android, please refer to IVCP_PERIPHERAL_POWER_ID.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

- In TREK-973 Windows, use this function to turn on the power of WiFi or BT. Windows WiFi or BT power icon doesn't work.
  Recommend use SetupDiGetClassDevs function or Windows Device Console (Devcon.exe) to turn on WiFi or BT power.

4.11.4.3 ivcp_peripheral_power_off

Syntax:

Windows / Linux	mrm_err **ivcp_peripheral_power_off**(ivcp_peripheral_power_id id)
Android	int **ivcp_peripheral_power_off**(int id)

Description:

Turn off the power of specifies device.

Parameters:

id [in]

The power ID of peripheral device.

For Windows/Linux, please refer to ivcp_peripheral_power_id .

For Android, please refer to IVCP_PERIPHERAL_POWER_ID.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

- In TREK-973 Windows, use this function to turn off the power of WiFi or BT. Windows WiFi or BT power icon doesn't work.
  Recommend use SetupDiGetClassDevs function or Windows Device Console (Devcon.exe) to turn off WiFi or BT power.

4.11.4.4 ivcp_peripheral_get_power_status

Syntax:

Windows / Linux	mrm_err **ivcp_peripheral_get_power_status**(ivcp_peripheral_power_id id, char *status)
Android	int **ivcp_peripheral_get_power_status**(int id, out boolean[] status)

Description:

Get power status of specifies device.

Parameters:

id [in]

The power ID of peripheral device.

For Windows/Linux, please refer to ivcp_peripheral_power_id .

For Android, please refer to IVCP_PERIPHERAL_POWER_ID.

status [out]

The power status of specifies device. The unit is second. The value 1 is On otherwise 0 is Off.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11.4.5 ivcp_peripheral_wwan_wakeup_enable

Syntax:

Windows / Linux	mrm_err **ivcp_peripheral_wwan_wakeup_enable**(void)
Android	int **ivcp_peripheral_wwan_wakeup_enable**()

Description:

Enable WWAN wakeup function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11.4.6 ivcp_peripheral_wwan_wakeup_disable

Syntax:

Windows / Linux	mrm_err **ivcp_peripheral_wwan_wakeup_disable**(void)
Android	int **ivcp_peripheral_wwan_wakeup_disable**()

Description:

Disable WWAN wakeup function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11.4.7 ivcp_peripheral_get_wwan_wakeup_status

Syntax:

Windows / Linux	mrm_err **ivcp_peripheral_get_wwan_wakeup_status**(char *status)
Android	int **ivcp_peripheral_get_wwan_wakeup_status**(boolean[] status)

Description:

Get status of WWAN wakeup function.

Parameters:

status [out]

The WWAN wakeup function status. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11.4.8 ivcp_peripheral_get_rearview

Syntax:

Windows / Linux	mrm_err **ivcp_peripheral_get_rearview**(ivcp_peripheral_rearview_id *id)
Android	int **ivcp_peripheral_get_rearview**(int[] id)

Description:

Get display view specifies id.

Parameters:

id [out]

The display view id.

For Windows/Linux, please refer to ivcp_peripheral_rearview_id

For Android, please refer to IVCP_PERIPHERAL_REARVIEW_ID

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11.4.9 ivcp_peripheral_set_rearview

Syntax:

Windows / Linux	mrm_err **ivcp_peripheral_set_rearview**(ivcp_peripheral_rearview_id id)
Android	int **ivcp_peripheral_set_rearview**(int id)

Description:

Set display view specifies id.

Parameters:

id [in]

The display view id.

For Windows/Linux, please refer to ivcp_peripheral_rearview_id

For Android, please refer to IVCP_PERIPHERAL_REARVIEW_ID

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11.4.10 ivcp_peripheral_auto_rearview_enable

Syntax:

Windows / Linux	mrm_err **ivcp_peripheral_auto_rearview_enable**(void)
Android	int **ivcp_peripheral_auto_rearview_enable**()

Description:

Enable the automatic switches to display Camera view if reverse gear detected and display LVDS view if reverse gear absent.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

If you enable this function, the digital input 1 will use to detect reverse gear status. The detail please check the out the product user manual.

4.11.4.11 ivcp_peripheral_auto_rearview_disable

Syntax:

Windows / Linux	mrm_err **ivcp_peripheral_auto_rearview_disable**(void)
Android	int **ivcp_peripheral_auto_rearview_disable**()

Description:

Disable the automatic switches to display Camera view if reverse gear detected and display LVDS view if reverse gear absent. If your current display on Camera view, system will force to switch LVDS view if you calling this function.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11.4.12 ivcp_peripheral_get_auto_rearview_status

Syntax:

Windows / Linux	mrm_err **ivcp_peripheral_get_auto_rearview_status**(char *status)
Android	int **ivcp_peripheral_get_auto_rearview_status**(boolean[] status)

Description:

Get status of automatic detect reverse gear function.

Parameters:

status [out]

Automatic detect reverse gear function status. The value 1 is Enable otherwise 0 is Disable.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11.4.13 ivcp_peripheral_get_comport_mode

Syntax:

Windows	-
Android	int **ivcp_peripheral_get_comport_mode**(int[] mode, boolean[] terminationStatus)

Description:

Get com port operating mode and status of termination.

Parameters:

mode [out]

The com port operating mode. Please refer to IVCP_PERIPHERAL_COMPORT_MODES.

terminationStatus [out]

The status of termination. The value is 1(True) when enabled, 0(false) when disabled.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11.4.14 ivcp_peripheral_set_comport_mode

Syntax:

Windows	-
Android	int **ivcp_peripheral_set_comport_mode**(int mode, boolean terminationStatus)

Description:

Set com port operating mode and status of termination.

Parameters:

mode [out]

The com port operating mode. Please refer to IVCP_PERIPHERAL_COMPORT_MODES.

terminationStatus [out]

The status of termination. Set value to 1(True) for enabled, 0(false) for disabled.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11.4.15 ivcp_peripheral_get_audio_input

Syntax:

Windows / Linux	mrm_err ivcp_peripheral_get_voice_input(ivcp_peripheral_voice_input *type)
Android	-

Description:

Get voice input source.

Parameters:

type [out]

The voice input source.

For Windows/Linux, please refer to ivcp_peripheral_audio_input_type

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11.4.16 ivcp_peripheral_set_audio_input

Syntax:

Windows / Linux	mrm_err ivcp_peripheral_get_voice_input(ivcp_peripheral_voice_input type)
Android	-

Description:

Set voice input source.

Parameters:

type [in]

The voice input source.

For Windows/Linux, please refer to ivcp_peripheral_audio_input_type

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.11.4.17 ivcp_peripheral_get_gps_antenna_status

Syntax:

Windows / Linux	mrm_err **ivcp_peripheral_get_gps_antenna_status**(char *status)
Android	int **ivcp_peripheral_get_gps_antenna_status**(byte[] status)

Description:

Get status of GPS antenna function.

Parameters:

status [out]

The GPS antenna function status. The value 0x00 is antenna removed, 0x01 is antenna short and 0x02 antenna normal status

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.12 Storage Functions

4.12.1 Usage

4.12.1.1 Windows/Linux

Please refer to the IVCP Conventions for basic usage.

4.12.1.2 Android

Please refer to the IVCP Conventions for basic usage.

4.12.2 APIs

4.12.2.1 ivcp_storage_get_size

Syntax:

Windows / Linux	mrm_err **ivcp_storage_get_size**(int *bytes)
Android	int **ivcp_storage_get_size**(int[] bytes)

Description:

Get the storage size.

Parameters:

bytes [out]

The storage size. The unit is byte.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.12.2.2 ivcp_storage_read

Syntax:

Windows / Linux	mrm_err **ivcp_storage_read**(unsigned int address, unsigned char *data, unsigned int size, unsigned int *readbyte)
Android	int **ivcp_storage_read**(int address, byte[] data, int size, int[] readbyte)

Description:

Read the specifies address of storage to the buffer.

Parameters:

address [in]

The specifies address.

data [out]

Pointer to a buffer that will hold the data of storage.

size [in]

The specifies data size to read.

readbyte [out]

The actual the number of bytes read

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.12.2.3 ivcp_storage_write

Syntax:

Windows / Linux	mrm_err **ivcp_storage_write**(unsigned int address, unsigned char *data, unsigned int size, unsigned int *writebyte)
Android	int **ivcp_storage_write**(int address, byte[] data, int size, int[] writebyte)

Description:

Write data to specifies address of storage.

Parameters:

address [in]

The specifies address.

data [in]

Pointer to a buffer that will hold the data of storage.

size [in]

The specifies data size to write.

writebyte [out]

The actual the number of bytes write

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.12.2.4 ivcp_storage_read_byte

Syntax:

Windows / Linux	mrm_err **ivcp_storage_read_byte**(unsigned int address, unsigned char *data)
Android	int **ivcp_storage_read_byte**(int address, byte[] data)

Description:

Read a byte from a specifies storage address.

Parameters:

address [in]

The specifies address.

data [out]

Pointer to a buffer that will hold the data of storage.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.12.2.5 ivcp_storage_write_byte

Syntax:

Windows / Linux	rm_err **ivcp_storage_write_byte**(unsigned int address, unsigned char data)
Android	int **ivcp_storage_write_byte**(int address, byte data)

Description:

Write a byte to a specifies storage address.

Parameters:

address [in]

The specifies address.

data [in]

Pointer to a buffer that will hold the data of storage.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.13 Speed Counter Functions

4.13.1 Usage

4.13.1.1 Windows/Linux

Please refer to the IVCP Conventions for basic usage.

4.13.1.2 Android

Please refer to the IVCP Conventions for basic usage.

4.13.2 APIs

4.13.2.1 ivcp_speedcounter_get_counter

Syntax:

Windows / Linux	mrm_err **ivcp_speedcounter_get_counter**(int *counter)
Android	int **ivcp_speedcounter_get_counter**(int[] counter)

Description:

Get the pulse count from the IVCP firmware managed buffer.

Parameters:

counter [out]

Pulse count.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

Remarks:

- It is recommended to call ivcp_speedcounter_reset_counter() once before this API is called.
- The size of buffer is 2 bytes. Buffer overflow will occurs if the pulse count is over 65535.

4.13.2.2 ivcp_speedcounter_reset_counter

Syntax:

Windows / Linux	mrm_err **ivcp_speedcounter_reset_counter**()
Android	int **ivcp_speedcounter_reset_counter**()

Description:

Clean the IVCP firmware managed buffer for pulse count. Reset the pulse count to 0.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.13.2.3 ivcp_speedcounter_get_and_reset_counter

Syntax:

Windows / Linux	mrm_err **ivcp_speedcounter_get_and_reset_counter**(int *counter)
Android	int **ivcp_speedcounter_get_and_reset_counter**(int[] counter)

Description:

Get the pulse count from the IVCP firmware managed buffer, then reset the count immediately.

Parameters:

counter [out]

Pulse count.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

Remarks:

- It is recommended to call ivcp_speedcounter_reset_counter() once before this API is called.
- The size of buffer is 2 bytes. Buffer overflow will occurs if the pulse count is over 65535.

4.14 Hotkey Functions

4.14.1 Usage

4.14.1.1 Windows/Linux

Please refer to the IVCP Conventions for basic usage.

4.14.1.2 Android

Please refer to the IVCP Conventions for basic usage.

4.14.2 APIs

4.14.2.1 ivcp_hotkey_get_keycode

Syntax:

Windows / Linux	-
Android	int **ivcp_hotkey_get_keycode**(int hotkeyId, int[] keycode)

Description:

Get the key code of specified hardware hot key .

Parameters:

hotkeyId [in]

ID of the target hot key. The ID is a zero-based index assigned to each hot key on the device.

ex: If there are 5 hot key on the device, the available ID is 0~4.

keycode [out]

The key code.

For the keycode value of Android, please refer to the key code list.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

4.14.2.2 ivcp_hotkey_set_keycode

Syntax:

Windows / Linux	-
Android	int **ivcp_hotkey_set_keycode**(int hotkeyId, int keycode)

Description:

Set the key code to specified hardware hot key .

Parameters:

hotkeyId [in]

ID of the target hot key. The ID is a zero-based index assigned to each hot key on the device.

ex: If there are 5 hot key on the device, the available ID is 0~4.

keycode [in]

The key code.

For the keycode value of Android, please refer to the key code list.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

4.14.2.3 ivcp_hotkey_get_led_brightness

Syntax:

Windows / Linux	-
Android	int **ivcp_hotkey_get_led_brightness**(int[] brightness)

Description:

Get brightness of hotkey LED.

Parameters:

brightness [out]

The LED brightness. The valid range is 0 to 100 in unit of percentage.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

4.14.2.4 ivcp_hotkey_set_led_brightness

Syntax:

Windows / Linux	-
Android	int **ivcp_hotkey_set_led_brightness**(int brightness)

Description:

Set brightness of hotkey LED.

Parameters:

brightness [in]

The LED brightness. The valid range is 0 to 100 in unit of percentage.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

4.14.3 Key Coode List

4.14.3.1 Android(US)

Keycode Value	Description
1	ESCAPE
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
11	0
12	MINUS
13	EQUALS
14	DEL
15	TAB
16	Q
17	W
18	E
19	R
20	T
21	Y
22	U
23	I
24	O
25	P
26	LEFT_BRACKET
27	RIGHT_BRACKET
28	ENTER
29	CTRL_LEFT
30	A
31	S
32	D

33	F
34	G
35	H
36	J
37	K
38	L
39	SEMICOLON
40	APOSTROPHE
41	GRAVE
42	SHIFT_LEFT
43	BACKSLASH
44	Z
45	X
46	C
47	V
48	B
49	N
50	M
51	COMMA
52	PERIOD
53	SLASH
54	SHIFT_RIGHT
55	NUMPAD_MULTIPLY
56	ALT_LEFT
57	SPACE
58	CAPS_LOCK
59	F1
60	F2
61	F3
62	F4
63	F5
64	F6
65	F7
66	F8
67	F9
68	F10

69	NUM_LOCK
70	SCROLL_LOCK
71	NUMPAD_7
72	NUMPAD_8
73	NUMPAD_9
74	NUMPAD_SUBTRACT
75	NUMPAD_4
76	NUMPAD_5
77	NUMPAD_6
78	NUMPAD_ADD
79	NUMPAD_1
80	NUMPAD_2
81	NUMPAD_3
82	NUMPAD_0
83	NUMPAD_DOT
85	ZENKAKU_HANKAKU
86	BACKSLASH
87	F11
88	F12
89	RO
92	HENKAN
93	KATAKANA_HIRAGANA
94	MUHENKAN
95	NUMPAD_COMMA
96	NUMPAD_ENTER
97	CTRL_RIGHT
98	NUMPAD_DIVIDE
99	SYSRQ
100	ALT_RIGHT
102	MOVE_HOME
103	DPAD_UP
104	PAGE_UP
105	DPAD_LEFT
106	DPAD_RIGHT
107	MOVE_END
108	DPAD_DOWN

109	PAGE_DOWN
110	INSERT
111	FORWARD_DEL
113	VOLUME_MUTE
114	VOLUME_DOWN
115	VOLUME_UP
116	POWER
117	NUMPAD_EQUALS
119	BREAK
121	NUMPAD_COMMA
122	KANA
123	EISU
124	YEN
125	META_LEFT
126	META_RIGHT
127	MENU
128	MEDIA_STOP
139	MENU
140	CALCULATOR
142	SLEEP
143	WAKEUP
150	EXPLORER
152	POWER
155	ENVELOPE
156	BOOKMARK
158	BACK
159	FORWARD
160	MEDIA_CLOSE
161	MEDIA_EJECT
162	MEDIA_EJECT
163	MEDIA_NEXT
164	MEDIA_PLAY_PAUSE
165	MEDIA_PREVIOUS
166	MEDIA_STOP
167	MEDIA_RECORD
168	MEDIA_REWIND

169	CALL
171	MUSIC
172	HOME
177	PAGE_UP
178	PAGE_DOWN
179	NUMPAD_LEFT_PAREN
180	NUMPAD_RIGHT_PAREN
200	MEDIA_PLAY
201	MEDIA_PAUSE
207	MEDIA_PLAY
208	MEDIA_FAST_FORWARD
212	CAMERA
213	MUSIC
215	ENVELOPE
217	SEARCH
224	BRIGHTNESS_DOWN
225	BRIGHTNESS_UP
226	HEADSETHOOK
256	BUTTON_1
257	BUTTON_2
258	BUTTON_3
259	BUTTON_4
260	BUTTON_5
261	BUTTON_6
262	BUTTON_7
263	BUTTON_8
264	BUTTON_9
265	BUTTON_10
266	BUTTON_11
267	BUTTON_12
268	BUTTON_13
269	BUTTON_14
270	BUTTON_15
271	BUTTON_16
288	BUTTON_1
289	BUTTON_2

290	BUTTON_3
291	BUTTON_4
292	BUTTON_5
293	BUTTON_6
294	BUTTON_7
295	BUTTON_8
296	BUTTON_9
297	BUTTON_10
298	BUTTON_11
299	BUTTON_12
300	BUTTON_13
301	BUTTON_14
302	BUTTON_15
303	BUTTON_16
304	BUTTON_A
305	BUTTON_B
306	BUTTON_C
307	BUTTON_X
308	BUTTON_Y
309	BUTTON_Z
310	BUTTON_L1
311	BUTTON_R1
312	BUTTON_L2
313	BUTTON_R2
314	BUTTON_SELECT
315	BUTTON_START
316	BUTTON_MODE
317	BUTTON_THUMBL
318	BUTTON_THUMBR
353	DPAD_CENTER
362	GUIDE
366	DVR
377	TV
397	CALENDAR
402	CHANNEL_UP
403	CHANNEL_DOWN

429	CONTACTS
464	FUNCTION
465	ESCAPE FUNCTION
466	F1 FUNCTION
467	F2 FUNCTION
468	F3 FUNCTION
469	F4 FUNCTION
470	F5 FUNCTION
471	F6 FUNCTION
472	F7 FUNCTION
473	F8 FUNCTION
474	F9 FUNCTION
475	F10 FUNCTION
476	F11 FUNCTION
477	F12 FUNCTION
478	1 FUNCTION
479	2 FUNCTION
480	D FUNCTION
481	E FUNCTION
482	F FUNCTION
483	S FUNCTION
484	B FUNCTION
580	APP_SWITCH
582	VOICE_ASSIST

4.15 Cradle Functions

4.15.1 Usage

4.15.1.1 Windows/Linux

Please refer to the IVCP Conventions for basic usage.

4.15.1.2 Android

Please refer to the IVCP Conventions for basic usage.

4.15.2 Enumeration

4.15.2.1 Windows/Linux

US keyboard keycode table

- **KEYCODE_A** 0x41
- **KEYCODE_B** 0x42
- **KEYCODE_C** 0x43
- **KEYCODE_D** 0x44
- **KEYCODE_E** 0x45
- **KEYCODE_F** 0x46
- **KEYCODE_G** 0x47
- **KEYCODE_H** 0x48
- **KEYCODE_I** 0x49
- **KEYCODE_J** 0x4A
- **KEYCODE_K** 0x4B
- **KEYCODE_L** 0x4C
- **KEYCODE_M** 0x4D
- **KEYCODE_N** 0x4E
- **KEYCODE_O** 0x4F
- **KEYCODE_P** 0x50
- **KEYCODE_Q** 0x51
- **KEYCODE_R** 0x52
- **KEYCODE_S** 0x53
- **KEYCODE_T** 0x54
- **KEYCODE_U** 0x55
- **KEYCODE_V** 0x56
- **KEYCODE_W** 0x57
- **KEYCODE_X** 0x58
- **KEYCODE_Y** 0x59
- **KEYCODE_Z** 0x5A
- **KEYCODE_1** 0x31
- **KEYCODE_2** 0x32
- **KEYCODE_3** 0x33
- **KEYCODE_4** 0x34
- **KEYCODE_5** 0x35
- **KEYCODE_6** 0x36
- **KEYCODE_7** 0x37

- **KEYCODE_8** 0x38
- **KEYCODE_9** 0x39
- **KEYCODE_0** 0x30
- **KEYCODE_F1** 0x70
- **KEYCODE_F2** 0x71
- **KEYCODE_F3** 0x72
- **KEYCODE_F4** 0x73
- **KEYCODE_F5** 0x74
- **KEYCODE_F6** 0x75
- **KEYCODE_F7** 0x76
- **KEYCODE_F8** 0x77
- **KEYCODE_F9** 0x78
- **KEYCODE_F10** 0x79
- **KEYCODE_F11** 0x7A
- **KEYCODE_F12** 0x7B
- **KEYCODE_ESC** 0x1B
- **KEYCODE_GRAVE_ACCENT** 0xC0
- **KEYCODE_TAB** 0x09
- **KEYCODE_CAPS_LOCK** 0x14
- **KEYCODE_SHIFT** 0x10
- **KEYCODE_CTRL** 0x11
- **KEYCODE_L_WIN** 0x5B
- **KEYCODE_ALT** 0x12
- **KEYCODE_MINUS** 0xBD
- **KEYCODE_EQUAL** 0xBB
- **KEYCODE_BACKSPACE** 0x08
- **KEYCODE_L_QUOTATION** 0xDB
- **KEYCODE_R_QUOTATION** 0xDD
- **KEYCODE_BACKSLASH** 0xDC
- **KEYCODE_SEMICOLON** 0xBA
- **KEYCODE_APOSTROPHE** 0xDE
- **KEYCODE_ENTER** 0x0D
- **KEYCODE_COMMA** 0xBC
- **KEYCODE_PERIOD** 0xBE
- **KEYCODE_SLASH** 0xBF
- **KEYCODE_APPLICATION** 0x5D
- **KEYCODE_R_WIN** 0x5C
- **KEYCODE_SPACEBAR** 0x20
- **KEYCODE_PRINT_SCREEN** 0x2C
- **KEYCODE_SCROLL_LOCK** 0x91
- **KEYCODE_PAUSE** 0x13
- **KEYCODE_INSERT** 0x2D
- **KEYCODE_HOME** 0x24
- **KEYCODE_PAGEUP** 0x21
- **KEYCODE_DELETE** 0x2E
- **KEYCODE_END** 0x23
- **KEYCODE_PAGEDOWN** 0x22
- **KEYCODE_LEFT_ARROW** 0x25
- **KEYCODE_UP_ARROW** 0x26
- **KEYCODE_RIGHT_ARROW** 0x27

- **KEYCODE_DOWN_ARROW** 0x28
- **KEYCODE_NUM_LOCK** 0x90

Remarks:

- The key code current only support for US standard keyboard.

ivcp_cradle_funkey_id

- (0) **IVCP_FUNCTION_KEY1** - The function key 0.
- (1) **IVCP_FUNCTION_KEY2** - The function key 1.
- (2) **IVCP_FUNCTION_KEY3** - The function key 2.
- (3) **IVCP_FUNCTION_KEY4** - The function key 3.
- (4) **IVCP_FUNCTION_KEY5** - The function key 4.

4.15.3 APIs

4.15.3.1 ivcp_cradle_set_detach_event

Syntax:

Windows	mrm_err **ivcp_cradle_set_detach_event**(void *cradle_event)
Android	-

Description:

Set a user define event in order to let IVCP library notify the specifies event when detect cradle detach.

Parameters:

cradle_event [in]

Pointer to the cradle detach event. In windows, the cradle_detach_event will pointer to a Windows Events HANDLE.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

4.15.3.2 ivcp_cradle_set_attach_event

Syntax:

Windows	mrm_err **ivcp_cradle_set_attach_event**(void *cradle_event)
Android	-

Description:

Set a user define event in order to let IVCP library notify the specifies event when detect cradle attach.

Parameters:

cradle_event [in]

Pointer to the cradle attach event. In windows, the cradle_attach_event will pointer to a Windows Events HANDLE.

Returns:

- **MRM_ERR_NO_ERROR** - On success.
  Otherwise see the error code list.

4.15.3.3 ivcp_cradle_get_status

Syntax:

Windows	mrm_err **ivcp_cradle_get_status**(char *status)
Android	-

Description:

Get current status of cradle.

Parameters:

status [out]

cradle detach or attach status. The value 1 is Attach otherwise 0 is detach.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

4.15.3.4 ivcp_cradle_get_led_brightness

Syntax:

Windows	mrm_err **ivcp_cradle_get_led_brightness**(unsigned char *brightness)
Android	-

Description:

Get brightness of LED brightness function.

Parameters:

brightness [out]

The LED brightness. The unit is 0~100 %.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

4.15.3.5 ivcp_cradle_set_led_brightness

Syntax:

Windows	mrm_err **ivcp_cradle_set_led_brightness**(unsigned char brightness)
Android	-

Description:

Set brightness of LED brightness function.

Parameters:

brightness [in]

The LED brightness. The unit is 0~100 %.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

4.15.3.6 ivcp_cradle_get_keycode_us

Syntax:

Windows	mrm_err **ivcp_cradle_get_keycode_us**(ivcp_cradle_funkey_id id,unsigned int *keycode)
Android	-

Description:

Get the key code of specifies function key (Hot Key) .

Parameters:

id [in]

The specifies function key. Please refer to cardle_funkey_id.

keycode [out]

The keyboard key codes. The key code current only support for US standard keyboard. Please refer to US keyboard keycode table.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

4.15.3.7 ivcp_cradle_set_keycode_us

Syntax:

Windows	mrm_err **ivcp_cradle_set_keycode_us**(ivcp_cradle_funkey_id id,unsigned int keycode)
Android	-

Description:

Get the key code to specifies function key (Hot Key) .

Parameters:

id [in]

The specifies function key. Please refer to cardle_funkey_id.

keycode [in]

The keyboard key codes. The key code current only support for US standard keyboard.Please refer to US keyboard keycode table.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

4.15.3.8 ivcp_cradle_get_keycode_hid

Syntax:

Windows	mrm_err **ivcp_cradle_get_keycode_hid**(ivcp_cradle_funkey_id id,unsigned int *keycode)
Android	-

Description:

Get the key code of specifies function key (Hot Key) .

Parameters:

id [in]

The specifies function key. Please refer to cardle_funkey_id.

keycode [out]

The USB HID key codes. Please refer to USB HID keycode.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

4.15.3.9 ivcp_cradle_set_keycode_hid

Syntax:

Windows	mrm_err **ivcp_cradle_set_keycode_hid**(ivcp_cradle_funkey_id id,unsigned int keycode)
Android	-

Description:

Get the key code to specifies function key (Hot Key) .

Parameters:

id [in]

The specifies function key. Please refer to cardle_funkey_id.

keycode [in]

The USB HID key codes. Please refer to USB HID keycode.

Returns:

- **MRM_ERR_NO_ERROR** - On success.

  Otherwise see the error code list.

5 Error Code List

5.1 Comman Error

- (0x00000000) **MRM_ERR_NO_ERROR** - On success.
- (0x00000001) **MRM_ERR_INVALID_POINTER** - Encounter invalid pointer.
- (0x00000002) **MRM_ERR_INVALID_ARGUMENT** - Encounter invalid argument. Please check out the API parameter.
- (0x00000003) **MRM_ERR_UNSUPPORT_OPERATION** - Encounter unsupported operation. Please check out spec. of the platform supported.
- (0x00000005) **MRM_ERR_LIBRARY_NOT_INIT** - Function call before the library init.
- (0x00000010) **MRM_ERR_ILLEGAL_OPERATION** - Encounter illegal operation.
- (0x00000011) **MRM_ERR_LIBRARY_ALREADY_INIT** - Function call before the library init.
- (0x00000012) **MRM_ERR_ARRAY_OUR_OF_RANGE** - Encounter the access array out of range.
- (0x00000013) **MRM_ERR_OPERATION_FAIL** - Encounter operation fail.

- (0x10000001) **MRM_ERR_ANDROID_JNI_NULL_POINTER** - Null pointer error occurred on service side(JNI).
- (0x10000002) **MRM_ERR_ANDROID_JNI_OUT_OF_MEMORY** - Out of memory error occurred on service side(JNI).
- (0x10000003) **MRM_ERR_ANDROID_JNI_EVENT_INIT_FAILED** - Failed to init event handle on service side(JNI).
- (0x10000004) **MRM_ERR_ANDROID_JNI_EVENT_DEINIT_FAILED** - Failed to init event handle on service side(JNI).
- (0x10000005) **MRM_ERR_ANDROID_JNI_EVENT_LISTENING_THREAD_ALREADY_RUNNING** - Event listening thread in service is already running(JNI).
- (0x10000006) **MRM_ERR_ANDROID_JNI_EVENT_LISTENING_THREAD_CREATE_FAILED** - Failed to create event listening thread in service(JNI).
- (0x10100001) **MRM_ERR_ANDROID_SERVICE_NULL_POINTER** - Null pointer error occurred on service side.
- (0x10100002) **MRM_ERR_ANDROID_SERVICE_UNKNWON_EXCEPTION** - Unknown exception occurred on service side.
- (0x10100003) **MRM_ERR_ANDROID_SERVICE_REMOTE_CALLBACK_LIST_NOT_FOUND** - Remote callback list not found on service side.
- (0x10100004) **MRM_ERR_ANDROID_SERVICE_REMOTE_CALLBACK_REGISTER_FAILED** - Failed to register remote callback on service side.

- (0x10100005) **MRM_ERR_ANDROID_SERVICE_REMOTE_CALLBACK_UNREGISTER_FAILED** - Failed to unregister remote callback on service side.
- (0x10100006) **MRM_ERR_ANDROID_SERVICE_REMOTE_CALLBACK_UPDATE_FAILED** - Failed to update remote callback cookie on service side.
- (0x10200001) **MRM_ERR_ANDROID_CLIENT_NULL_POINTER** - Null pointer error occurred on client side in IVCP Service Client API lib.
- (0x10200002) **MRM_ERR_ANDROID_CLIENT_FAILED_TO_BIND_SERVICE** - Unable to connect client application context to the service.
- (0x10200003) **MRM_ERR_ANDROID_CLIENT_SERVICE_ALREADY_CONNECTED** - Current client application context is already connected to the service.
- (0x10200004) **MRM_ERR_ANDROID_CLIENT_SERVICE_DISCONNECTED** - Current client application context is not connected to the service.
- (0x10200005) **MRM_ERR_ANDROID_CLIENT_REMOTE_EXCEPTION** - Remote exception received at client side. Failed to execute required task.
- (0x10200006) **MRM_ERR_ANDROID_CLIENT_FAILED_TO_UNBIND_SERVICE** - Unable to disconnect client application context to the service.

5.2 IVCP Error

- (0x01000001) **MRM_ERR_IVCP_DEVICE_NODE_OPEN_FAIL** - Open IVCP device node fail. Please checkout IVCP is exist or the device not use by another application.
- (0x01000002) **MRM_ERR_IVCP_DEVICE_NODE_WRITE_FAIL** - Encounter write operation fail. Please retry operation.
- (0x01000003) **MRM_ERR_IVCP_DEVICE_NODE_READ_FAIL** - Encounter read operation fail. Please retry operation.
- (0x01000004) **MRM_ERR_IVCP_IS_BUSY** - Device is busy. try again.
- (0x01000005) **MRM_ERR_IVCP_ERROR_COMMAND** - Device is not recognize this command operation.
- (0x01000006) **MRM_ERR_IVCP_ERROR_RESPONSE_FORMAT_NOT_MATCH** - Device is not recognize this response.
- (0x01000007) **MRM_ERR_IVCP_PARAMETER_OUT_OF_RANGE** - The parameter out of range.
- (0x01000008) **MRM_ERR_IVCP_DEVICE_NODE_READ_TIMEOUT** - Device out of expect time.
- (0x01000009) **MRM_ERR_IVCP_NO_EEPROM_CHIP_FIND** - Device can't find the storage to save the config. Please contract your FAE to check out the hardware.
- (0x0100000A) **MRM_ERR_IVCP_DEVICE_NODE_WRITE_NOT_MATCH** - Device is not recognize this command operation.
- (0x0100000B) **MRM_ERR_IVCP_NO_GSENSOR_CHIP_FIND** - Device can't find the G-Sensor chip on the platform. Please contract your FAE to get supported.
- 0x0100000C) **MRM_ERR_IVCP_UNKNOW_RETURN_VALUE** - Library can not recognize response value.
- 0x0100000D) **MRM_ERR_IVCP_GSENSOR_DATA_INVALID** - The G-Sensor value is invalid.
- 0x0100000E) **MRM_ERR_IVCP_UNKNOW_ERROR_CODE** - Library can not recognize error code.
- 0x0100000F) **MRM_ERR_IVCP_PSENSOR_DATA_INVALID** - The P-Sensor value is invalid.
- 0x01000010) **MRM_ERR_IVCP_NO_PSENSOR_CHIP_FIND** - Device can't find the P-Sensor chip on the platform. Please contract your FAE to get supported.
- 0x01000011) **MRM_ERR_IVCP_SMBUS_ENCOUTER_ERROR** - Library read SMBus encounter error. The operation aborted.
- 0x01000012) **MRM_ERR_IVCP_GSENSOR_DATA_NOT_READY** - The G-Sensor alarm data queue is empty.
- 0x01000013) **MRM_ERR_IVCP_USER_DEFAULT_IS_EMPTY** - The current user default configuration is empty and can not be loaded.