MRM GPS User Manual

5/4/2017
SSID.SRP

Table of contents

1 Introduction

1.1 GPS Overview

MRM GPS API is a lightweight interface between OS(Operating system) and GPS module. In order to flexibly meet customer demand, we provide developer a command-style to query GPS information. The MRM SDK offer many kinds of API to query GPS information without user deal any NMEA code parse. For the advanced user, you can using callback function for easy capture NMEA code without deal any serial port handling. At begin using GPS API, you should call **gps_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 **gps_init()**, the library will initialize GPS and start to accept command.

MRM GPS API(**gps.dll** or **gps.so**)

The detail please refer to the below section.

1.2 System Requirements

Hardware:

- GPS Module (u-blox)

Operating system:

- Windows
- Linux

2 Using GPS Library

2.1 GPS Library Conventions

- You should call **gps_init()** before using the other GPS API
- You should always check the return value equal **MRM_ERR_NO_ERROR**(0), in order to ensure the command working.
- 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.
- Any GPS API has a prefix "**gps**_" immediately followed by the function name and the operation name

2.2 C++ with Visual Studio

1. Right Click on Project to enter Project Properites
2. Add #include "gps/gps.h" to your program
3. Select **Linker** and **Input** page. Set Additional Dependencies "gps.lib"

2.3 C++ with gcc

1. Add the header path in CXXFLAG
2. Add #include "gps/gps.h" to your program
3. Add -lgps in LDFLAGS

4. gcc $(CXXFLAG) xxx.cpp -o xxx.o
5. gcc xxx.o $(LDFLAGS) -o output

3 Function Overview

3.1 GPS Information Query

MRM GPS SDK provide information as below table

Item	Related API
Latitude	gps_get_location
Longitude	gps_get_location
Altitude	gps_get_location
UTC Time	gps_get_utc_time
Fix status	gps_get_fix_status
Antenna status	gps_get_antenna_status
Speed over ground	gps_get_speed
Satellite Information	gps_get_satellite_in_view, gps_get_satellites
NMEA code	gps_set_nmea_callback, gps_clear_nmea_callback

4 Application Programming Interface (API)

4.1 GPS Functions

4.1.1 gps_fix_status Enum

- (0) **GPS_STATUS_NO_FIX** - No fix with GPS.
- (1) **GPS_STATUS_GPS** - Fix with GPS.
- (2) **GPS_STATUS_DGPS** - Fix with Differential GPS.
- (3) **GPS_STATUS_DR** - Estimated dead reckoning (linear extrapolation).

4.1.2 gps_antenna_status Enum

- (0) **GPS_ANTENNA_UNKNOWN** - Unknown GPS antenna status.
- (1) **GPS_ANTENNA_OK** - The GPS antenna is OK.
- (2) **GPS_ANTENNA_OPEN** - The GPS antenna is open.
- (3) **GPS_ANTENNA_SHORT** - The GPS antenna is short. Please check out the signal line is fine.

4.1.3 gps_location_t Structure

Syntax:

```
typedef struct
{
        double latitude;
        double longitude;
        double altitude;
} gps_location_t;
```

Description:

This data structure defines the current location.

Members:

latitude

The degree of latitude. The negative is south otherwise north.

longitude

The degree of longitude. The negative is west otherwise east.

altitude

The altitude. The unit is meter.

4.1.4 gps_time_t_t Structure

Syntax:

```
typedef struct
{
        unsigned short year;
        unsigned char month;
        unsigned char day;
        unsigned char hour;
        unsigned char minute;
        unsigned char second;
        unsigned short msecond;
} gps_time_t;
```

Description:

This data structure defines the current location.

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.

meecond

The mini second . The valid values for this member are 0 through 999.

4.1.5 gps_satellite_t Structure

Syntax:

```
typedef struct
{
    int prn;
    int elevation;
    int azimuth;
    int snr;
    int used;
} gps_satellite_t;
```

Description:

This data structure defines the current location.

Members:

prn

The satellite ID.

elevation

The elevation in degrees. The valid values for this member are 0 to 90

azimuth

The azimuth degrees from true North. The valid values for this member are 0 through 359.

snr

The satellite SNR in dB. The valid values for this member are 0 through 99.

used

The usage. The values 1 is used otherwise not.

4.1.6 gps_init

Syntax:

```
mrm_err gps_init(char *port)
```

Description:

General initialization of the GPS library.

Parameters:

port [in]

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

Example:

port = "\\.\COM4" at Windows

port = "/dev/ttyS3" at Linux

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

Prior to calling any GPS API function the library needs to be initialized by calling this function. The return code for all GPS API function will be **MRM_ERR_LIBRARY_NOT_INIT** unless this function is called.

4.1.7 gps_deinit

Syntax:

```
mrm_err gps_deinit(void)
```

Description:

General uninitialization of the GPS library.

Parameters:

none

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.1.8 gps_get_version

Syntax:

```
mrm_err gps_get_version(char *version)
```

Description:

Get the library version of GPS.

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 **GPS_MAXIMUM_LIBRARY_STRING_LENGTH**(24)

4.1.9 gps_get_location

Syntax:

```
mrm_err gps_get_location(gps_location_t *location)
```

Description:

Get the last location information. you should always check the GPS fix status before calling this function.

Parameters:

location [out]

Pointer to gps_location_t structure that will hold the lalocation information.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

The library will keep the last valid GPS information until the new valid location information coming. At begin, library does not hold any validation data, instead return **MRM_ERR_GPS_DATA_NOT_READY**.

4.1.10 gps_get_speed

Syntax:

```
mrm_err gps_get_speed(double *mps)
```

Description:

Get the last speed over ground.

Parameters:

mps [out]

Pointer to a double that will hold the last speed over ground. The unit is meter per second.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.1.11 gps_get_utc_time

Syntax:

```
mrm_err gps_get_utc_time(gps_time_t *utc_time)
```

Description:

Get the last UTC time from GPS. you should always check the GPS fix status before calling this function. We recommend you using this time only for time synchronization. The GPS signal does not guarantee stability. When no signal, the time will hold the last updated time.

Parameters:

utc_time [out]

Pointer to gps_time_t structure that will hold the UTC time information.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

The library will keep the last valid GPS information until the new valid location information coming. At begin, library does not hold any validation data, instead return **MRM_ERR_GPS_DATA_NOT_READY**.

4.1.12 gps_get_satellite_in_view

Syntax:

```
mrm_err gps_get_satellite_in_view(int *count)
```

Description:

Get the number of satellites in view from GPS signal.

Parameters:

count [out]

Pointer to a int that will hold the number of satellites in view.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.1.13 gps_get_satellites

Syntax:

```
mrm_err gps_get_satellites(gps_satellite_t *satellite)
```

Description:

Get the last satellites information. You need check the satellite member of structure "used" to ensure the satellite is actual used for GPS.

Parameters:

satellite [out]

Pointer to gps_satellite_t structure array that will hold the satellites information. The data order as below

satellites 1, satellites 2,.. , and so on.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

The library will keep the last valid GPS information until the new valid location information coming. You should provide **enough memory** space for storing satellites information. We recommend you provide at least 32 * sizeof(gps_satellite_t).

4.1.14 gps_get_fix_status

Syntax:

```
mrm_err gps_get_fix_status(gps_fix_status *status)
```

Description:

Get the GPS fix status.

Parameters:

status [out]

Pointer to a status that will hold the fix GPS status. The detail please refer to gps_fix_status section.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.1.15 gps_get_antenna_status

Syntax:

```
mrm_err gps_get_antenna_status(gps_antenna_status *status)
```

Description:

Get the GPS antenna status.

Parameters:

status [out]

Pointer to a status that will hold the GPS antenna status. The detail please refer to gps_antenna_status section.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

Remarks:

When the status become **GPS_ANTENNA_SHORT**, the GPS module need a period time(30s+) to recovery to normal state after the troubleshooting.

4.1.16 gps_set_nmea_callback

Syntax:

```
typedef void ( _stdcall *gps_nmea_callback ) ( char *NMEA , int length)
mrm_err gps_set_nmea_callback(gps_nmea_callback callback)
```

Description:

This function is used to register the NMEA callback function. There can be only one function registered for this callback. The callback is called every time NMEA message coming from the GPS port. The function has as its arguments the NMEA string. This information can be used in the callback function to perform the appropriate action.

The callback function is used for asynchronous notification, somewhat like a hardware interrupt. We recommend that you note the information provided in the callback, but not process a long time work itself in the callback function. That should be done in the appropriate routine or thread dedicated to processing. If your callback function take too long time, its maybe cause other GPS API response time extend.

Parameters:

callback [in]

Pointer to a user defined callback function.

Returns:

MRM_ERR_NO_ERROR - On success.

Otherwise see the error code list.

4.1.17 gps_clear_nmea_callback

Syntax:

```
mrm_err gps_clear_nmea_callback(void)
```

Description:

Clear the callback registration of the GPS library.

Parameters:

none

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.
- (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.

5.2 GPS Error

- (0x04000001) **MRM_ERR_GPS_DEVICE_NODE_OPEN_FAIL** - Open GPS device node fail. Please checkout GPS is exist or the device not use by another application.
- (0x04000002) **MRM_ERR_GPS_DEVICE_NODE_WRITE_FAIL** - Encounter write operation fail.
- (0x04000003) **MRM_ERR_GPS_DEVICE_NODE_READ_FAIL** - Encounter read operation fail.
- (0x04000004) **MRM_ERR_GPS_IS_BUSY** - SDP device is busy. Try again.
- (0x04000008) **MRM_ERR_GPS_DEVICE_NODE_READ_TIMEOUT** - SDP encounter out of time when read operation.
- (0x04000009) **MRM_ERR_GPS_DATA_NOT_READY** - GPS in first initialization data not valid.
- (0x0400000A) **MRM_ERR_GPS_UNKNOWN_HARDWARE** - GPS not found on this platform. Please check out the hardware PCB version.
- (0x0400000B) **MRM_ERR_GPS_OPEN_FAIL_NO_DATA_IN** - Open GPS device node fail. No NMEA code detected in this node. Please checkout GPS is opened or the device not use by another application.