MDM9x07&MDM9628

GPS API

User Guide

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
7th Floor, Hongye Building, No.1801 Hongmei Road, Xuhui District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://quectel.com/support/sales.htm

For technical support, or to report documentation errors, please visit:
http://quectel.com/support/technical.htm
Or email to: support@quectel.com

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT

About the Document

This document applies to MDM9628 and MDM9X07.

History

Revision	Date	Author	Description
1.0	2018-01-10	Navy.Qiu	Initial

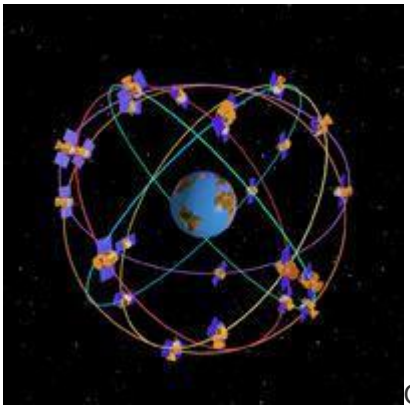
Contents

1. Global Position System (GPS) Introduction

The global positioning and navigation system, called the global satellite positioning system (GPS), is a global positioning and navigation system using GPS to locate the satellite. GPS is developed by the United States Department of Defense established a comprehensive, all-weather, full-time, high-precision satellite navigation system, for the world to provide users with low cost and high precision three-dimensional position, velocity and timing accurate navigation information, it is the application model for satellite communication technology in navigation field. GPS greatly improves the information level of the Earth Society, effectively promotes the development of the digital economy.

1.1 Positioning Principle

The basic principle of GPS navigation system is to measure the distance between the satellite from the known location and the user receiver, then get the location of the receiver by combining the data from multiple satellites. To achieve this goal, the location of the satellite can be detected in the satellite ephemeris according to the time recorded by the satellite clock. The distance from satellite to user is the satellite signal travelled time to the user then multiplied by the speed of light. (due to the atmosphere ionosphere interference, this distance is not really for the distance between the user and satellite but pseudo range. (PR,) : When the GPS satellites work normally, they will continuously transmit navigation messages with pseudo-random codes(referred to as pseudo code) consisting of 1 and 0 binary symbols. There are two kinds of pseudo code used in GPS system, respectively are civil C/A code and military P (Y) code. The frequency of C/A code is 1.023MHz, the repetition period is one millisecond, the code interval is 1 microseconds, equal to 300m, the P code frequency is 10.23MHz, the repetition period is 266.4 days, and the code interval is 0.1 microseconds, which is equal to 30m. The Y code is formed on the basis of P code and has better security performance. Navigation messages include satellite ephemeris, working condition, clock correction, ionospheric delay correction, atmospheric refraction correction and so on. It is modulated from the satellite signal and is transmitted on the carrier frequency by 50b/s modulation. Each main frame of the navigation message contains 5 sub-frames with a length of 6S per frame. The first three frames each have 10 words; repeated every thirty seconds, updated every hour. The last two frames are 15000b totally. The navigation message mainly include telemetry code, conversion code, and first, second, third data blocks, and the most important is the ephemeris data. When the user receives the navigation message, Extract the satellite time and compare it with own clock to get the distance between the satellite and the user, then use the satellite ephemeris data in the navigation message to calculate the position of the satellite when transmitting the message, position, speed which users in the WGS-84 geodetic coordinate system and other information can be obtained.

 GPS

It can be seen that the role of the satellite part of the GPS navigation system is to continuously launch the navigation message. However, it may not always synchronized that the user receives the clock and satellite machine using a carrier clock, so in addition to the 3D coordinates of x, y, z user, but also introduce Δt that is the time difference between the satellite and the receiver as unknown, and then use the four equations to get this four unknowns. So if you want to know the location of the receiver, signals from at least 4 satellites should be received.

The GPS receiver can receive accurate up to nanosecond-level time information that can be used for timing; a forecast ephemeris for forecasting the approximate location of the satellite in the coming months; a broadcast ephemeris for calculating the satellite coordinates required for positioning, the accuracy is a few meters to tens of meters (changed at any time due to different satellites); GPS system information, such as satellite status and so on.

The GPS receiver can get the distance from the satellite to the receiver by measuring the code, since it contains the error of the receiver's satellite clock and the atmospheric propagation error, that distance is called pseudo range. The pseudo range measured by CA code is called the pseudo range of CA code, the precision is about 20 meters. The pseudo distance measured by P code is called the pseudo distance of the P code, and the precision is about 2 meters.

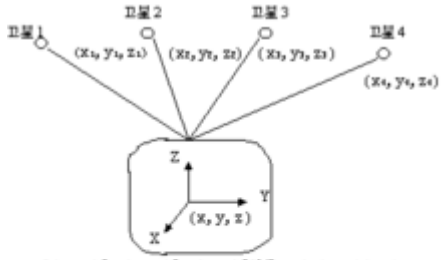
The GPS receiver decodes the received satellite signals or uses other techniques to restore the carrier by removing the information that is modulated on the carrier. Strictly speaking, the carrier phase should be called the carrier beat phase. It is the difference between the carrier phase of the satellite signal and the phase of the receiver's oscillation, which is affected by the Doppler frequency shift. Usually measured at the epoch time determined by the receiver clock to keep track of the satellite signal, the phase change value can be recorded, but the initial phase of the receiver and satellite oscillator at the beginning of the observation is unknown, the phase integer of the initial epoch is unknown, i.e. integer ambiguity, only as a parameter solution in data processing. The accuracy of the phase observation is as high as millimeter, but the premise is to solve the integer ambiguity. Therefore, only under the relative location and a continuous observation value, phase observation value can be used, and accuracy must use the phase observation value to achieve the positioning that better than the meter level.

According to the positioning mode, GPS positioning is divided into single point positioning and relative positioning (differential positioning). Single point positioning is based on the observation data of

a receiver to determine the location of the receiver. It can only use pseudo range measurement, and it can be used for rough navigation and location of vehicles and ships. Relative positioning (differential positioning) is the way to determine the relative position between the observation points which according to the observation data of more than 2 receivers, it can use the pseudo range measurement also can take phase measurement, geodetic or engineering measurement should use the phase observation value of relative positioning.

The GPS measurements include satellite and receiver clock error, atmospheric propagation delay, multipath error, but also influenced by the satellite broadcast ephemeris error in the positioning calculation, the relative positioning of most public errors are offset or weakened, thus the positioning accuracy will be greatly improved, dual-frequency receiver can offset the main part of the ionospheric error in the atmosphere according to the observation measurement of two frequencies, when the accuracy is high required and the distance between receivers is far (the atmosphere is obviously different), should choose the dual-frequency receiver.

The basic principle of GPS positioning is based on the instantaneous position of high speed satellite as the known initial data, and take the method of space distance resection to determine the location of the point to be measured. As shown in the figure, assuming that the GPS receiver is placed on the ground at the time t, the time △t at which the GPS signal reaches the receiver can be measured, and the other data such as the satellite ephemeris received by the receiver can determine the following four Formulas.



$$[(x_1-x)^2+(y_1-y)^2+(z_1-z)^2]^{1/2}+c(\Delta t_r-\Delta t_s)=d_1$$
$$[(x_2-x)^2+(y_2-y)^2+(z_2-z)^2]^{1/2}+c(\Delta t_r-\Delta t_s)=d_2$$
$$[(x_3-x)^2+(y_3-y)^2+(z_3-z)^2]^{1/2}+c(\Delta t_r-\Delta t_s)=d_3$$
$$[(x_4-x)^2+(y_4-y)^2+(z_4-z)^2]^{1/2}+c(\Delta t_r-\Delta t_s)=d_4$$

1.2 Positioning Accuracy

28 satellites (of which 4 are spare) have been launched for long and are distributed on the 60 - degree track of 6 intersection points, about 20000 kilometers from the ground. The precision of the single machine navigation is about 10 meters, and the precision can reach the centimeter level and the millimeter level. But the accuracy of the civil field is about 10 meters.

2. GPS Location Interface

2.1 QL_LOC_Client_Init

1. function：
   int QL_LOC_Client_Init(loc_client_handle_type *ph_loc);
2. parameter：
   1) ph_loc：OUT location handle
3. return：int , <0: error, 0: success, >0: partial success
4. description：
   Initiate location function module and return the handle.

2.2 QL_LOC_Client_Deinit

1. function：
   int QL_LOC_Client_Deinit(loc_client_handle_type h_loc);
2. parameter：
   1) h_loc：IN Location handle
3. return：int, 0: success, other: error
4. description：
   De-initiate location related resource.

2.3 QL_LOC_AddStateHandler

1. function：
   int QL_LOC_AddRxIndMsgHandler(QL_LOC_RxIndMsgHandlerFunc_t handlerPtr,
   void* contextPtr);
2. parameter：
   1) handlerPtr: IN location status callback function
   2) contextPtr: IN callback function tag.
3. return：int, 0: success, other: error
4. description：
   Register location status callback function. The kind of callback message type depends on the mask set via QL_LOC_Set_Indications ；

2.4 QL_LOC_Set_Indications

1. function：
   int QL_LOC_Set_Indications(loc_client_handle_type h_loc, int bit_mask);
2. parameter：
   1) h_loc：IN Location handle
   2) bit_mask：IN function on-off control bit，each bit means as following:
      #define LOC_IND_LOCATION_INFO_ON (1 << 0)
      #define LOC_IND_STATUS_INFO_ON (1 << 1)
      #define LOC_IND_SV_INFO_ON (1 << 2)
      #define LOC_IND_NMEA_INFO_ON (1 << 3)
      #define LOC_IND_CAP_INFO_ON (1 << 4)
      #define LOC_IND_UTC_TIME_REQ_ON (1 << 5)
      #define LOC_IND_XTRA_DATA_REQ_ON (1 << 6)
      #define LOC_IND_AGPS_DATA_CONN_CMD_REQ_ON (1 << 7)
      #define LOC_IND_NI_NFY_USER_RESP_REQ_ON (1 << 8)
3. return：int, 0: success, other: error
4. description：
   Set callback function message type；

2.5 QL_LOC_Start_Navigation

1. function：
   int QL_LOC_Start_Navigation(loc_client_handle_type h_loc);
2. parameter：
   1) h_loc：IN location handle
3. return：int, 0: success, other: error
4. description：
   Start getting location data.

2.6 QL_LOC_Stop_Navigation

1. function：
   int QL_LOC_Stop_Navigation(loc_client_handle_type h_loc);
2. parameter：
   1) h_loc：IN location handle
3. return：int, 0: success, other: error
4. description：

stop getting location data.

2.7 QL_LOC_Set_Position_Mode

1. function：
   int QL_LOC_Set_Position_Mode(loc_client_handle_type h_loc,
   QL_LOC_POS_MODE_INFO_T *pt_mode);
2. parameter：
   1) h_loc：IN location handle
3. return：int, 0: success, other: error
4. description：
   Set location callback parameter, i.e. navigator mode，getting data interval, accuracy, detail infor please refer to QL_LOC_POS_MODE_INFO_T；

2.8 QL_LOC_Get_Current_Location

1. function：
   int QL_LOC_Get_Current_Location(loc_client_handle_type h_loc,
   QL_LOC_LOCATION_INFO_T *pt_loc_info,
   int timeout_sec);
2. parameter：
   1) h_loc：IN location handle
   2) pt_loc_info: IN location param setting
   3) timeout_sec: IN timeout value
3. return：int, 0: success, other: error，and -2 means timeout
4. description：
   Get current location synchronously.

2.9 QL_LOC_RxIndMsgHandlerFunc_t

1. function：
   typedef void (*QL_LOC_RxIndMsgHandlerFunc_t)
   (loc_client_handle_type h_loc,
   E_QL_LOC_NFY_MSG_ID_T e_msg_id,
   void *pv_data,
   void *contextPtr);
2. parameter：

1) h_loc：OUT location handle

2) e_msg_id: OUT message ID

3) pv_data: OUT message content，detail format depends on the message ID as following:

```
typedef enum
{
    E_QL_LOC_NFY_MSG_ID_STATUS_INFO = 0,          /**<  pv_data = &E_QL_LOC_STATUS_VALUE_T  */
    E_QL_LOC_NFY_MSG_ID_LOCATION_INFO,            /**<  pv_data = &QL_LOC_LOCATION_INFO_T  */
    E_QL_LOC_NFY_MSG_ID_SV_INFO,                  /**<  pv_data = &QL_LOC_SV_STATUS_T  */
    E_QL_LOC_NFY_MSG_ID_NMEA_INFO,                /**<  pv_data = &QL_LOC_NMEA_INFO_T  */
    E_QL_LOC_NFY_MSG_ID_CAPABILITIES_INFO,        /**<  pv_data = &E_QL_LOC_CAPABILITIES_T  */
    E_QL_LOC_NFY_MSG_ID_AGPS_STATUS,              /**<  pv_data = &QL_LOC_AGPS_STATUS_T  */
    E_QL_LOC_NFY_MSG_ID_NI_NOTIFICATION,          /**<  pv_data = &QL_LOC_NI_NOTIFICATION_INTO_T  */
    E_QL_LOC_NFY_MSG_ID_XTRA_REPORT_SERVER,                                   /**<  pv_data =
&QL_LOC_XTRA_REPORT_SERVER_INTO_T  */
}E_QL_LOC_NFY_MSG_ID_T;
```

4) contextPtr: OUT 回调 tag

3. return：void
4. description：

When specified event happens set in QL_LOC_Set_Indications and QL_LOC_Set_Position_Mode，it will callback via this function；

3. GPS API Manual Guide

You can refer to example/API/api_test_main.c or example/gnss/example_gps.c for help.

Usage：

User Case 1： (Generally use, notify event via callback function):

1, QL_LOC_Client_Init

2, QL_LOC_AddRxIndMsgHandler(pf_cb)

3, QL_LOC_Set_Indications

4, QL_LOC_Set_Position_Mode

5, QL_LOC_Start_Navigation

6, handle the events in pf_cb

7, QL_LOC_Stop_Navigation

8, QL_LOC_Client_Deinit

User case 2： (Get current location only once):

1, QL_LOC_Client_Init

2, QL_LOC_AddRxIndMsgHandler(pf_cb) ---- This can be omitted!

3, QL_LOC_Set_Indications, set bit_mask=LOC_IND_LOCATION_INFO_ON

4, QL_LOC_Set_Position_Mode Set single mode

5, QL_LOC_Get_Current_Location, if not timeout, it will return current position infor or use last stored one.
6, QL_LOC_Client_Deinit

4. GPS demo

2.1 Run example

root@mdm9607-perf:/# ./example_gps

2.2 Check result

```
root@mdm9607-perf:/# ./example_gps
=============== gps test start =============
please input test mode(0: sync_get_position_once, other:get_gps_info_by_cb): 1
Starting MCM RIL Services: done
[QL_MCM_Client_Init    529]:    mcm_client_init ret=0x2 with h_mcm=0x0 ==> Sleep 2s and Retry !
[QL_MCM_Client_Init    529]:    mcm_client_init ret=0x2 with h_mcm=0x0 ==> Sleep 2s and Retry !
[QL_MCM_Client_Init    536]: Client initialized successfully 0x3
[QL_MCM_Client_Init    546]:    mcm_client_init start up required service!
[ql_mcm_async_cb    252]: ####h_mcm=0x3    msg_id=0x800
[ql_mcm_client_srv_updown_async_cb    33]: ####h_mcm=0x3    msg_id=0x800
[loc_ind_cb    22]:
 ===== mcmlocservice UP ! =========
QL_LOC_Client_Init ret 0 with h_loc=3
QL_LOC_AddRxIndMsgHandler ret 0
Please input indication bitmask(NiNfy|AGPS|XTRA|UTC|CAP|NMEA|SV|Status|Location):
511         //511=0x1FF=01 1111 1111, means to turn on all bits
[ql_mcm_ind_cb    133]: ####h_mcm=0x3    msg_id=0x312
[ql_loc_rx_ind_msg_cb    8]: e_msg_id=4
[ql_mcm_ind_cb    133]: ####h_mcm=0x3    msg_id=0x312
[ql_loc_rx_ind_msg_cb    8]: e_msg_id=4
QL_LOC_Set_Indications ret 0
QL_LOC_Set_Position_Mode ret 0
QL_LOC_Start_Navigation ret=0
Wait and handle event ! You can input -1 to exit): [ql_mcm_ind_cb    133]: ####h_mcm=0x3    msg_id=0x30f
[ql_loc_rx_ind_msg_cb    8]: e_msg_id=0
[ql_mcm_ind_cb    133]: ####h_mcm=0x3    msg_id=0x30f
[ql_loc_rx_ind_msg_cb    8]: e_msg_id=0
[ql_mcm_ind_cb    133]: ####h_mcm=0x3    msg_id=0x30f
[ql_loc_rx_ind_msg_cb    8]: e_msg_id=0
[ql_mcm_ind_cb    133]: ####h_mcm=0x3    msg_id=0x310
```

```
[ql_loc_rx_ind_msg_cb    8]: e_msg_id=2
[ql_mcm_ind_cb    133]: ####h_mcm=0x3    msg_id=0x311
[ql_loc_rx_ind_msg_cb    8]: e_msg_id=3
NMEA info: timestamp=315964862708, length=17, nmea=$GPGSV,1,1,0,*65
[ql_mcm_ind_cb    133]: ####h_mcm=0x3    msg_id=0x311
[ql_loc_rx_ind_msg_cb    8]: e_msg_id=3
NMEA info: timestamp=315964862709, length=17, nmea=$GLGSV,1,1,0,*79
[ql_mcm_ind_cb    133]: ####h_mcm=0x3    msg_id=0x311
[ql_loc_rx_ind_msg_cb    8]: e_msg_id=3
NMEA info: timestamp=315964862709, length=29, nmea=$GPGSA,A,1,,,,,,,,,,,,,,,*1E
[ql_mcm_ind_cb    133]: ####h_mcm=0x3    msg_id=0x311
[ql_loc_rx_ind_msg_cb    8]: e_msg_id=3
NMEA info: timestamp=315964862709, length=24, nmea=$GPVTG,,T,,M,,N,,K,N*2C
[ql_mcm_ind_cb    133]: ####h_mcm=0x3    msg_id=0x311
[ql_loc_rx_ind_msg_cb    8]: e_msg_id=3
NMEA info: timestamp=315964862710, length=24, nmea=$GPRMC,,V,,,,,,,,N*53
[ql_mcm_ind_cb    133]: ####h_mcm=0x3    msg_id=0x311
[ql_loc_rx_ind_msg_cb    8]: e_msg_id=3
NMEA info: timestamp=315964862710, length=25, nmea=$GPGGA,,,,,,0,,,,,,,,*66    //If output this, means succsuful.
```

GPS sample code

```
#include <ql_oe.h>

static void ql_loc_rx_ind_msg_cb(loc_client_handle_type  h_loc,
                                 E_QL_LOC_NFY_MSG_ID_T   e_msg_id,
                                 void                    *pv_data,
                                 void                    *contextPtr)
{
    QL_USER_LOG("e_msg_id=%d\n", e_msg_id);
    switch(e_msg_id)
    {//Convert pv_data to the right format and do some process based on the message id.
        case E_QL_LOC_NFY_MSG_ID_STATUS_INFO:
            break;
        case E_QL_LOC_NFY_MSG_ID_LOCATION_INFO:
```

```
        {
            QL_LOC_LOCATION_INFO_T *pt_location = (QL_LOC_LOCATION_INFO_T
*)pv_data;
            printf("**** flag=0x%X, Latitude = %f, Longitude=%f, accuracy = %f
****\n",
                    pt_location->flags,
                    pt_location->latitude,
                    pt_location->longitude,
                    pt_location->accuracy);
            break;
        }
        case E_QL_LOC_NFY_MSG_ID_SV_INFO:
            break;
        case E_QL_LOC_NFY_MSG_ID_NMEA_INFO:
        {
            QL_LOC_NMEA_INFO_T  *pt_nmea = (QL_LOC_NMEA_INFO_T  *)pv_data;

            printf("NMEA info: timestamp=%lld, length=%d, nmea=%s\n",
                   pt_nmea->timestamp, pt_nmea->length, pt_nmea->nmea);
            break;
        }
        case E_QL_LOC_NFY_MSG_ID_CAPABILITIES_INFO:
            break;
        case E_QL_LOC_NFY_MSG_ID_AGPS_STATUS:
            break;
        case E_QL_LOC_NFY_MSG_ID_NI_NOTIFICATION:
            break;
        case E_QL_LOC_NFY_MSG_ID_XTRA_REPORT_SERVER:
            break;
    }
}

void sync_get_position_once(void)
{
    int                 ret        = E_QL_OK;
    int                 h_loc      = 0;
    int                 bitmask    = 0;
    QL_LOC_POS_MODE_INFO_T  t_mode     = {0};
    QL_LOC_LOCATION_INFO_T  t_loc_info  = {0};
    int                 timeout_sec = 60;

    ret = QL_LOC_Client_Init(&h_loc);
    printf("QL_LOC_Client_Init ret %d with h_loc=%d\n", ret, h_loc);
```

```
    ret = QL_LOC_AddRxIndMsgHandler(ql_loc_rx_ind_msg_cb, (void*)h_loc);
    printf("QL_LOC_AddRxIndMsgHandler ret %d\n", ret);

    bitmask = 1; //force set to 1 to get location only.

    ret = QL_LOC_Set_Indications(h_loc, bitmask);
    printf("QL_LOC_Set_Indications ret %d\n", ret);

    t_mode.mode               = E_QL_LOC_POS_MODE_STANDALONE;
    t_mode.recurrence          = E_QL_LOC_POS_RECURRENCE_SINGLE;
    t_mode.min_interval        = 10;
    t_mode.preferred_accuracy   = 50;
    t_mode.preferred_time      = 90;  //You can adjust these param
    ret = QL_LOC_Set_Position_Mode(h_loc, &t_mode);
    printf("QL_LOC_Set_Position_Mode ret %d\n", ret);

    ret = QL_LOC_Get_Current_Location(h_loc, &t_loc_info, timeout_sec);
    printf(" QL_LOC_Get_Current_Location ret %d\n", ret);
    if(ret < 0)
    {
        if(ret == -2)
        {// -2: timeout, may need try again
            printf("QL_LOC_Get_Current_Location timeout, try again!\n");
        }
        else
        {
            printf("QL_LOC_Get_Current_Location Fail, ret %d\n", ret);
        }
    }
    else
    {
        printf("**** Latitude = %lf, Longitude=%lf, altitude=%lf, accuracy = %f
****\n",
                t_loc_info.latitude, t_loc_info.longitude,
t_loc_info.altitude, t_loc_info.accuracy);
    }

    ret = QL_LOC_Client_Deinit(h_loc);
    printf("QL_LOC_Client_Deinit ret=%d\n", ret);

    return ;
}

void get_gps_info_by_cb(void)
```

```
{
    int                     ret        = E_QL_OK;
    int                     h_loc      = 0;
    int                     bitmask    = 0;
    QL_LOC_POS_MODE_INFO_T  t_mode     = {0};
    QL_LOC_LOCATION_INFO_T  t_loc_info  = {0};

    ret = QL_LOC_Client_Init(&h_loc);
    printf("QL_LOC_Client_Init ret %d with h_loc=%d\n", ret, h_loc);

    ret = QL_LOC_AddRxIndMsgHandler(ql_loc_rx_ind_msg_cb, (void*)h_loc);
    printf("QL_LOC_AddRxIndMsgHandler ret %d\n", ret);

    printf("Please input indication
bitmask(NiNfy|AGPS|XTRA|UTC|CAP|NMEA|SV|Status|Location):\n", ret);
    scanf("%d", &bitmask);   //turn on function callback bits as required

    /* Set what we want callbacks for */
    ret = QL_LOC_Set_Indications(h_loc, bitmask);
    printf("QL_LOC_Set_Indications ret %d\n", ret);

    t_mode.mode               = E_QL_LOC_POS_MODE_STANDALONE;
    t_mode.recurrence          = E_QL_LOC_POS_RECURRENCE_PERIODIC;
    t_mode.min_interval        = 10;
    t_mode.preferred_accuracy   = 50;
    t_mode.preferred_time      = 90;  //
    ret = QL_LOC_Set_Position_Mode(h_loc, &t_mode);
    printf("QL_LOC_Set_Position_Mode ret %d\n", ret);

    ret = QL_LOC_Start_Navigation(h_loc);
    printf("QL_LOC_Start_Navigation ret=%d\n", ret);

    while(1)
    {
        int finish_flag = 0;//Wait for message arrive and process
        printf("Wait and handle event ! You can input -1 to exit): ");
        scanf("%d", &finish_flag);
        if(finish_flag == -1)
        {
            break;
        }
    }
    ret = QL_LOC_Stop_Navigation(h_loc);
    printf("QL_LOC_Stop_Navigation ret=%d\n", ret);
```

```
    ret = QL_LOC_Client_Deinit(h_loc);
    printf("QL_LOC_Client_Deinit ret=%d\n", ret);
}


int main(int argc, char *argv[])
{
    int mode;

    printf("=============== gps test start =============\r\n");
    printf("please input test mode(0: sync_get_position_once,
other:get_gps_info_by_cb): ");
    scanf("%d", &mode);

    if(mode == 0)
    {
        sync_get_position_once();
    }
    else
    {
        get_gps_info_by_cb();
    }
    printf("=============== gps test end =============\r\n");
}
```

How to build GPS example

Build single example_voice.c：

1. Unzip ql-ol-sdk.tar.bz2：tar -jxvf ql-ol-sdk.tar.bz2
2. Enter ql-ol-sdk folder：cd ql-ol-sdk
3. source ql-ol-crosstool/ql-ol-crosstool-env-init（Please make sure FW version is same to SDK version, or else error may happened）
4. execute：cd ql-ol-extsdk/example/example_gps
5. execute：make clean;make;