QuecOpen EC2X&AG35 GPS API 使用指导

About the Document

本文档适用于 EC2X 和 AG35 平台

History

Revision	Date	Author	Description
1.0	2018-01-10	Navy.Qiu	Initial
1.1	2018-2-8	Navy.Qiu	Update, Add AGPS related API

目录

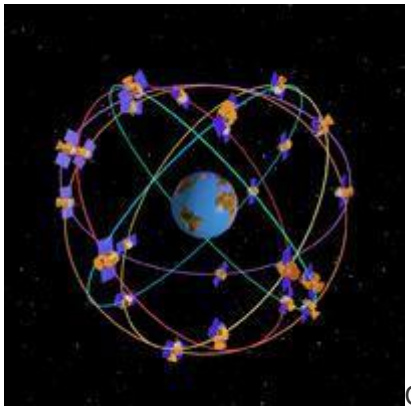
QuecOpen EC2X&AG35 GPS API 使用指导 0
About the Document 1
1. GPS 全球定位业务 3
1.1 定位原理 3
1.2 定位精度 5
2. GPS Location 接口函数 5
2.1 QL_LOC_Client_Init 5
2.2 QL_LOC_Client_Deinit 5
2.3 QL_LOC_AddRxIndMsgHandler 5
2.4 QL_LOC_Set_Indications 6
2.5 QL_LOC_Start_Navigation 6
2.6 QL_LOC_Stop_Navigation 7
2.7 QL_LOC_Set_Position_Mode 7
2.8 QL_LOC_Get_Current_Location 7
2.9 QL_LOC_RxIndMsgHandlerFunc_t 8
2.10 QL_LOC_Delete_Aiding_Data 8
2.11 QL_LOC_InjectTime 9
2.12 QL_LOC_InjectLocation 10
2.13 QL_LOC_Xtra_InjectData 10
2.14 QL_LOC_Xtra_InjectFile 10
2.15 QL_LOC_Agps_DataConnOpen 11
2.16 QL_LOC_Agps_DataConnClose 11
2.17 QL_LOC_Agps_NfyDataConnFailed 12
2.18 QL_LOC_Agps_SetServer 12
2.19 QL_LOC_NI_SetResponse 13
2.20 QL_LOC_Agps_UpdateNWAvailability 13
3. GPS API 使用步骤 13
4. GPS daemon 演示步骤 14
2.1 命令执行 14
2.2 检查结果 14
GPS 实例代码 15
GPS 编译说明 19

1. GPS 全球定位业务

利用 GPS 定位卫星，在全球范围内实时进行定位、导航的系统，称为全球卫星定位系统，简称 GPS。GPS 是由美国国防部研制建立的一种具有全方位、全天候、全时段、高精度的卫星导航系统，能为全球用户提供低成本、高精度的三维位置、速度和精确定时等导航信息，是卫星通信技术在导航领域的应用典范，它极大地提高了地球社会的信息化水平，有力地推动了数字经济的发展。

1.1 定位原理

GPS 导航系统的基本原理是测量出已知位置的卫星到用户接收机之间的距离，然后综合多颗卫星的数据就可知道接收机的具体位置。要达到这一目的，卫星的位置可以根据星载时钟所记录的时间在卫星星历中查出。而用户到卫星的距离则通过记录卫星信号传播到用户所经历的时间，再将其乘以光速得到（由于大气层电离层的干扰，这一距离并不是用户与卫星之间的真实距离，而是伪距（PR，）：当 GPS 卫星正常工作时，会不断地用 1 和 0 二进制码元组成的伪随机码（简称伪码）发射导航电文。GPS 系统使用的伪码一共有两种，分别是民用的 C/A 码和军用的 P(Y)码。C/A 码频率 1.023MHz，重复周期一毫秒，码间距 1 微秒，相当于 300m；P 码频率 10.23MHz，重复周期 266.4 天，码间距 0.1 微秒，相当于 30m。而 Y 码是在 P 码的基础上形成的，保密性能更佳。导航电文包括卫星星历、工作状况、时钟改正、电离层时延修正、大气折射修正等信息。它是从卫星信号中解调制出来，以 50b/s 调制在载频上发射的。导航电文每个主帧中包含 5 个子帧每帧长 6s。前三帧各 10 个字码；每三十秒重复一次，每小时更新一次。后两帧共 15000b。导航电文中的内容主要有遥测码、转换码、第 1、2、3 数据块，其中最重要的则为星历数据。当用户接受到导航电文时，提取出卫星时间并将其与自己的时钟做对比便可得知卫星与用户的距离，再利用导航电文中的卫星星历数据推算出卫星发射电文时所处位置，用户在 WGS-84 大地坐标系中的位置速度等信息便可得知。


GPS

可见 GPS 导航系统卫星部分的作用就是不断地发射导航电文。然而，由于用户接受机使用的时钟与卫星星载时钟不可能总是同步，所以除了用户的三维坐标 x、y、z 外，还要引进一个 Δt 即卫星与接收机之间的时间差作为未知数，然后用 4 个方程将这 4 个未知数解出来。所以如果想知道接收机所处的位置，至少要能接收到 4 个卫星的信号。

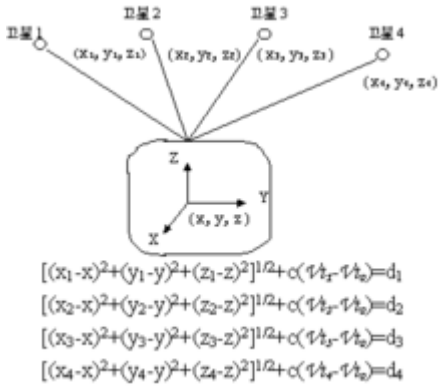
GPS 接收机可接收到可用于授时的准确至纳秒级的时间信息；用于预报未来几个月内卫星所处概略位置的预报星历；用于计算定位时所需卫星坐标的广播星历，精度为几米至几十米（各个卫星不同，随时变化）；以及 GPS 系统信息，如卫星状况等。

GPS 接收机对码的量测就可得到卫星到接收机的距离，由于含有接收机卫星钟的误差及大气传播误差，故称为伪距。对 CA 码测得的伪距称为 CA 码伪距，精度约为 20 米左右，对 P 码测得的伪距称为 P 码伪距，精度约为 2 米左右。

GPS 接收机对收到的卫星信号，进行解码或采用其它技术，将调制在载波上的信息去掉后，就可以恢复载波。严格而言，载波相位应被称为载波拍频相位，它是收到的受多普勒频 移影响的卫星信号载波相位与接收机本机振荡产生信号相位之差。一般在接收机钟确定的历元时刻量测，保持对卫星信号的跟踪，就可记录下相位的变化值，但开始观测时的接收机和卫星振荡器的相位初值是不知道的，起始历元的相位整数也是不知道的，即整周模糊度，只能在数据处理中作为参数解算。相位观测值的精度高至毫米，但前提是解出整周模糊度，因此只有在相对定位、并有一段连续观测值时才能使用相位观测值，而要达到优于米级的定位 精度也只能采用相位观测值。

按定位方式，GPS 定位分为单点定位和相对定位（差分定位）。单点定位就是根据一台接收机的观测数据来确定接收机位置的方式，它只能采用伪距观测量，可用于车船等的概略导航定位。相对定位（差分定位）是根据两台以上接收机的观测数据来确定观测点之间的相对位置的方法，它既可采用伪距观测量也可采用相位观测量，大地测量或工程测量均应采用相位观测值进行相对定位。

在 GPS 观测量中包含了卫星和接收机的钟差、大气传播延迟、多路径效应等误差，在定位计算时还要受到卫星广播星历误差的影响，在进行相对定位时大部分公共误差被抵消或削弱，因此定位精度将大大提高，双频接收机可以根据两个频率的观测量抵消大气中电离层误差的主要部分，在精度要求高，接收机间距离较远时（大气有明显差别），应选用双频接收机。



GPS 定位原理

GPS 定位的基本原理是根据高速运动的卫星瞬间位置作为已知的起算数据，采用空间距离后方交会的方法，确定待测点的位置。如图所示，假设 t 时刻在地面待测点上安置 GPS 接收机，可以测定 GPS 信号到达接收机的时间△t，再加上接收机所接收到的卫星星历等其它数据可以确定以下四个方程式。

1.2 定位精度

28 颗卫星(其中 4 颗备用)早已升空，分布在 6 条交点互隔 60 度的轨道面上，距离地面约 20000 千米。已经实现单机导航精度约为 10 米，综合定位的话，精度可达厘米级和毫米级。但民用领域开放的精度约为 10 米。

2. GPS Location 接口函数

2.1 QL_LOC_Client_Init

1. 函数原型：
   int QL_LOC_Client_Init(loc_client_handle_type *ph_loc);
2. 参数说明：
   1) ph_loc：OUT location 句柄指针
3. 返回说明：int，非 0 表示错误
4. 功能描述：
   获取 Location 功能使用句柄初始化

2.2 QL_LOC_Client_Deinit

1. 函数原型：
   int QL_LOC_Client_Deinit(loc_client_handle_type h_loc);
2. 参数说明：
   1) h_loc：IN Location 句柄
3. 返回说明：int, 非 0 表示错误
4. 功能描述：
   相关 Location 功能资源销毁

2.3 QL_LOC_AddRxIndMsgHandler

1. 函数原型：
   int QL_LOC_AddRxIndMsgHandler(QL_LOC_RxIndMsgHandlerFunc_t handlerPtr,
   void* contextPtr);
2. 参数说明：
   1) handlerPtr： IN location 状态回调函数

2) contextPtr： IN （主要是用于区分是哪个 client 返回的消息）

3. 返回说明：int, 非 0 表示错误
4. 功能描述：

注册 location 状态的回调函数，该回调函数能接收到的消息由 QL_LOC_Set_Indications 确定；

2.4 QL_LOC_Set_Indications

1. 函数原型：

```
int QL_LOC_Set_Indications(loc_client_handle_type h_loc, int bit_mask);
```

2. 参数说明：
   1) h_voice：IN voice 句柄
   2) bit_mask：IN 功能使能开关，每个 bit 含义如下：

```
#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. 返回说明：int, 非 0 表示错误
4. 功能描述：

设置回调信息类型；

2.5 QL_LOC_Start_Navigation

1. 函数原型：

```
int QL_LOC_Start_Navigation(loc_client_handle_type h_loc);
```

2. 参数说明：
   1) h_loc：IN location 句柄
3. 返回说明：int, 非 0 表示错误
4. 功能描述：

开始获取导航数据；

2.6 QL_LOC_Stop_Navigation

1. 函数原型：
   int QL_LOC_Stop_Navigation(loc_client_handle_type h_loc);
2. 参数说明：
   1) h_loc：IN location 句柄
3. 返回说明：int, 非 0 表示错误
4. 功能描述：
   停止获取导航数据；

2.7 QL_LOC_Set_Position_Mode

函数原型：

```
int QL_LOC_Set_Position_Mode(loc_client_handle_type h_loc,
                             QL_LOC_POS_MODE_INFO_T *pt_mode);
```

1. 参数说明：
   1) h_loc：IN location 句柄
2. 返回说明：int, 非 0 表示错误
3. 功能描述：
   设置导航参数，如导航模式，获取数据间隔，精度等信息；

2.8 QL_LOC_Get_Current_Location

1. 函数原型：

```
int QL_LOC_Get_Current_Location(loc_client_handle_type      h_loc,
                                QL_LOC_LOCATION_INFO_T *pt_loc_info,
                                int                         timeout_sec);
```

2. 参数说明：
   1) h_loc：IN location 句柄
   2) pt_loc_info: IN location 参数设置
   3) timeout_sec: IN 超时时间
3. 返回说明：int, 非 0 表示错误，其中-2 表示超时
4. 功能描述：
   同步方式获取当前位置信息，给定时间内未获取到就返回超时 ；

2.9 QL_LOC_RxIndMsgHandlerFunc_t

1. 函数原型：

```
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. 参数说明：

   1) h_loc：OUT　　location 句柄

   2) e_msg_id: OUT　　消息 ID

   3) pv_data: OUT　　消息内容，格式取决于消息 ID，如下说明

```
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. 返回说明：void

4. 功能描述：

   根据 QL_LOC_Set_Indications 和 QL_LOC_Set_Position_Mode 设置，在有对应事件发生时候，通过该接口接相关信息送上来 ；

2.10 QL_LOC_Delete_Aiding_Data

1. 函数原型：

```
int QL_LOC_Delete_Aiding_Data(    loc_client_handle_type                h_loc,
                                  E_QL_LOC_DELETE_AIDING_DATA_TYPE_T    flags);
```

2. 参数说明：

   1) h_loc：IN　　location 句柄

   2) flags:　IN　　Aiding data 功能标记位，定义如下：

```
typedef enum
{
    E_QL_LOC_DELETE_EPHEMERIS       = (1 <<  0),    /**<   Delete ephemeris data.   */
    E_QL_LOC_DELETE_ALMANAC         = (1 <<  1),    /**<   Delete almanac data.   */
```

```
    E_QL_LOC_DELETE_POSITION          = (1 <<  2),    /**<  Delete position data.  */
    E_QL_LOC_DELETE_TIME              = (1 <<  3),    /**<  Delete time data.  */
    E_QL_LOC_DELETE_IONO              = (1 <<  4),    /**<  Delete IONO data.  */
    E_QL_LOC_DELETE_UTC               = (1 <<  5),    /**<  Delete UTC data.  */
    E_QL_LOC_DELETE_HEALTH            = (1 <<  6),    /**<  Delete health data.  */
    E_QL_LOC_DELETE_SVDIR             = (1 <<  7),    /**<  Delete SVDIR data.  */
    E_QL_LOC_DELETE_SVSTEER           = (1 <<  8),    /**<  Delete SVSTEER data.  */
    E_QL_LOC_DELETE_SADATA            = (1 <<  9),    /**<  Delete SA data.  */
    E_QL_LOC_DELETE_RTI               = (1 << 10),    /**<  Delete RTI data.  */
    E_QL_LOC_DELETE_CELLDB_INFO       = (1 << 11),    /**<  Delete cell DB information.  */
    E_QL_LOC_DELETE_ALMANAC_CORR      = (1 << 12),    /**<  Delete almanac correction data.  */
    E_QL_LOC_DELETE_FREQ_BIAS_EST     = (1 << 13),    /**<  Delete frequency bias estimate.  */
    E_QL_LOC_DELETE_EPHEMERIS_GLO     = (1 << 14),    /**<  Delete ephemeris GLO data.  */
    E_QL_LOC_DELETE_ALMANAC_GLO       = (1 << 15),    /**<  Delete almanac GLO data.  */
    E_QL_LOC_DELETE_SVDIR_GLO         = (1 << 16),    /**<  Delete SVDIR GLO data.  */
    E_QL_LOC_DELETE_SVSTEER_GLO       = (1 << 17),    /**<  Delete SVSTEER GLO data.  */
    E_QL_LOC_DELETE_ALMANAC_CORR_GLO= (1 << 18),    /**<  Delete almanac correction GLO data.  */
    E_QL_LOC_DELETE_TIME_GPS          = (1 << 19),    /**<  Delete time GPS data.  */
    E_QL_LOC_DELETE_TIME_GLO          = (1 << 20),    /**<  Delete time GLO data.  */
    E_QL_LOC_DELETE_ALL               = 0xFFFFFFFF,   /**<  Delete all location data.  */
}E_QL_LOC_DELETE_AIDING_DATA_TYPE_T;
```

3. 返回说明：int ，非 0 表示错误
4. 功能描述：

   根据 flags 所设置功能类型，删除对应辅助数据；

2.11 QL_LOC_InjectTime

1. 函数原型：

```
int QL_LOC_InjectTime(   loc_client_handle_type          h_loc,
                         QL_LOC_INJECT_TIME_INTO_T     *pt_info);
```

2. 参数说明：

   1) h_loc：IN location 句柄

   2) pt_info: IN 时间参数信息，定义如下：

```
typedef struct
{
    int64_t time;                 /**<  Inject time.*/
    int64_t time_reference;       /**<  Time reference.*/
    int32_t uncertainty;          /**<  Uncertainty.*/
}QL_LOC_INJECT_TIME_INTO_T;   /* Message */
```

3. 返回说明：int，非 0 表示错误
4. 功能描述：

   注入时间信息，用于判断所注入的 xtra data 是否有效；

2.12 QL_LOC_InjectLocation

1. 函数原型：
   int QL_LOC_InjectLocation(loc_client_handle_type h_loc,
   QL_LOC_INJECT_LOCATION_INTO_T *pt_info);
2. 参数说明：
   1) h_loc：IN location 句柄
   2) pt_info: IN 位置信息指针，定义如下：

```
typedef struct
{
    double  latitude;   /**<   Latitude.*/
    double  longitude;  /**<   Longitude.*/
    float   accuracy;   /**<   Accuracy.*/
}QL_LOC_INJECT_LOCATION_INTO_T;
```

3. 返回说明：int, 非 0 表示错误
4. 功能描述：
   注入位置信息，用于快速定位 ；

2.13 QL_LOC_Xtra_InjectData

1. 函数原型：
   int QL_LOC_Xtra_InjectData(loc_client_handle_type h_loc,
   char *data,
   int length);
2. 参数说明：
   1) h_loc：IN location 句柄
   2) data: IN 存储 xtra data 数据指针
   3) length: IN xtra data 数据长度
3. 返回说明：int, 非 0 表示错误
4. 功能描述：
   导入 xtra 数据，由于 IPC 机制限制，此 buffer 最大只能支持到 0xFC00；

2.14 QL_LOC_Xtra_InjectFile

1. 函数原型：
   int QL_LOC_Xtra_InjectFile(loc_client_handle_type h_loc,
   char *filename);
2. 参数说明：

1) h_loc：IN location 句柄

2) filename: IN xtra 文件完整路径

3. 返回说明：int, 非 0 表示错误
4. 功能描述：

导入 filename 所指定的 xtra 数据，由于 IPC 机制限制，目前此处文件最大只能支持到 0xFC00；

2.15 QL_LOC_Agps_DataConnOpen

1. 函数原型：

```
int QL_LOC_Agps_DataConnOpen( loc_client_handle_type                h_loc,
                              QL_LOC_AGPS_DATA_CONN_OPEN_INTO_T *pt_info);
```

2. 参数说明：

1) h_loc：IN location 句柄

2) pt_info: IN 数据连接服务器信息，定义如下：

```
#define QL_LOC_APN_NAME_LENGTH_MAX 100
typedef struct
{
    E_QL_LOC_AGPS_TYPE_T            e_agps_type;                                /**<  AGPS type.*/
    char                            apn[QL_LOC_APN_NAME_LENGTH_MAX + 1];/**<  APN.*/
    E_QL_LOC_AGPS_APN_BEARER_TYPE_T e_bearer_type;                              /**<  Bearer type.*/
}QL_LOC_AGPS_DATA_CONN_OPEN_INTO_T;
```

3. 返回说明：int, 非 0 表示错误
4. 功能描述：

标记 AGPS 数据连接已经打开；

2.16 QL_LOC_Agps_DataConnClose

1. 函数原型：

```
int QL_LOC_Agps_DataConnClose(loc_client_handle_type    h_loc,
                              E_QL_LOC_AGPS_TYPE_T      atype);
```

2. 参数说明：

1) h_loc：IN location 句柄

2) atype: IN agps 类型，定义如下：

```
typedef enum
{
    E_QL_LOC_AGPS_TYPE_INVALID          = -1,   /**<  Invalid.  */
    E_QL_LOC_AGPS_TYPE_ANY              = 0,    /**<  Any.  */
    E_QL_LOC_AGPS_TYPE_SUPL             = 1,    /**<  SUPL.  */
    E_QL_LOC_AGPS_TYPE_C2K              = 2,    /**<  C2K.  */
    E_QL_LOC_AGPS_TYPE_WWAN_ANY         = 3,    /**<  WWAN any.  */
    E_QL_LOC_AGPS_TYPE_WIFI             = 4,    /**<  Wi-Fi.  */
```

```
    E_QL_LOC_AGPS_TYPE_SUPL_ES          = 5,    /**<  SUPL_ES.  */
}E_QL_LOC_AGPS_TYPE_T;
```

3. 返回说明：int，非 0 表示错误
4. 功能描述：

   标记 AGPS 数据连接已经关闭；

2.17 QL_LOC_Agps_NfyDataConnFailed

1. 函数原型：

```
int QL_LOC_Agps_NfyDataConnFailed(loc_client_handle_type    h_loc,
                                  E_QL_LOC_AGPS_TYPE_T      atype);
```

2. 参数说明：

   1) h_loc：IN location 句柄

   2) atype: IN AGPS 类型

3. 返回说明：int，非 0 表示错误
4. 功能描述：

   标记 AGPS 数据连接启动失败；

2.18 QL_LOC_Agps_SetServer

1. 函数原型：

```
int QL_LOC_Agps_SetServer(loc_client_handle_type          h_loc,
                          QL_LOC_AGPS_SERVER_INTO_T       *pt_info);
```

2. 参数说明：

   1) h_loc：IN location 句柄

   2) pt_info: IN AGPS 服务器信息，定义如下:

```
#define QL_LOC_SEVER_ADDR_LENGTH_MAX    255
typedef struct
{
    E_QL_LOC_AGPS_TYPE_T    e_agps_type;                                  /**<  AGPS type.*/
    char                    host_name[QL_LOC_SEVER_ADDR_LENGTH_MAX + 1];/**<  Host name.*/
    uint32_t                port;                                         /**<  Port.*/
}QL_LOC_AGPS_SERVER_INTO_T;
```

3. 返回说明：int，非 0 表示错误
4. 功能描述：

   设置 AGPS 服务器信息。

2.19 QL_LOC_NI_SetResponse

1. 函数原型：
   int QL_LOC_NI_SetResponse(loc_client_handle_type h_loc,
   QL_LOC_NI_RESPONSE_INTO_T *pt_info);
2. 参数说明：
   1) h_loc：IN location 句柄
   2) pt_info: IN AGPS NI 响应信息
3. 返回说明：int, 非 0 表示错误
4. 功能描述：
   发送 NI 用户响应信息。

2.20 QL_LOC_Agps_UpdateNWAvailability

1. 函数原型：
   int QL_LOC_Agps_UpdateNWAvailability(loc_client_handle_type h_loc,
   int available,
   const char *apn);
2. 参数说明：
   1) h_loc：IN location 句柄
   2) available: IN 标记是否可用
   2) apn: IN apn 名字
3. 返回说明：int, 非 0 表示错误
4. 功能描述：
   更新网络可用状态。

3. GPS API 使用步骤

请参考 example/API/api_test_main.c
说明：
使用案例一： (一般使用模式，通过回调函数将相关信息上报给 App):
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

使用案例二： (主动获取一次 location):
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 设置单次模式
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 daemon 演示步骤

2.1 命令执行

root@mdm9607-perf:/# ./example_gps

2.2 检查结果

```
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，表示打开所有 bit
[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 //确认这些信息可以正确输出

GPS 实例代码

```
#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)
    {//根据对应消息获取相应信息，并做对应处理
        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;  //参数可根据需要自行调整
    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);    //根据需要设置bitmask，打开对应回调消息

    /* 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;  //参数可根据需要自行调整
    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;//等待消息到来并在回调函数里面处理
        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");
}
```

GPS 编译说明

编译单个 example_voice.c 说明：

1. ql-ol-sdk.tar.bz2 解压：tar -jxvf ql-ol-sdk.tar.bz2
2. 进入 ql-ol-sdk 目录：cd ql-ol-sdk
3. source ql-ol-crosstool/ql-ol-crosstool-env-init（确保 SDK 版本与模块版本一致，否则可能出现错误）

4. 执行，命令：cd ql-ol-extsdk/example/example_gps
5. 执行：make clean;make;