AG35 Series QuecOpen GNSS API Reference Manual

Automotive Module Series

Version: 1.0

Date: 2020-09-08

Status: Released

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

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236 Email: info@quectel.com

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

For technical support, or to report documentation errors, please visit:
http://www.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.

DISCLAIMER

WHILE QUECTEL HAS MADE EFFORTS TO ENSURE THAT THE FUNCTIONS AND FEATURES UNDER DEVELOPMENT ARE FREE FROM ERRORS, IT IS POSSIBLE THAT THESE FUNCTIONS AND FEATURES COULD CONTAIN ERRORS, INACCURACIES AND OMISSIONS. UNLESS OTHERWISE PROVIDED BY VALID AGREEMENT, QUECTEL MAKES NO WARRANTIES OF ANY KIND, IMPLIED OR EXPRESS, WITH RESPECT TO THE USE OF FEATURES AND FUNCTIONS UNDER DEVELOPMENT. TO THE MAXIMUM EXTENT PERMITTED BY LAW, QUECTEL EXCLUDES ALL LIABILITY FOR ANY LOSS OR DAMAGE SUFFERED IN CONNECTION WITH THE USE OF THE FUNCTIONS AND FEATURES UNDER DEVELOPMENT, REGARDLESS OF WHETHER SUCH LOSS OR DAMAGE MAY HAVE BEEN FORESEEABLE.

COPYRIGHT

About the Document

Revision History

Version	Date	Author	Description
1.0	2020-09-08	Raigor ZHAO	Initial

Contents

Table Index

1 Introduction

Global navigation satellite system (GNSS) is a general term describing any satellite constellation that provides positioning, navigation, and timing (PNT) services on a global or regional basis. GNSS can also refer to augmentation systems.

- Global navigation satellite systems include GPS, GLONASS, BeiDou and Galileo.
- Regional navigation satellite systems include QZSS and NavIC (IRNSS).
- Satellite-based augmentation systems (SBAS) include WAAS, SDCM, EGNOS, MSAS, GAGAN

This document introduces how to realize the GNSS feature of Quectel AG35 series module in QuecOpen® solution, through GNSS APIs in SDK provided by Quectel.

2 GNSS Features Overview

Quectel AG35 series QuecOpen module integrates the IZat Gen8C GNSS engine which supports GPS, BeiDou, GLONASS and Galileo systems. This, coupled with SUPL and gpsOneXTRA Assistance technologies, allows the module to provide quicker, more accurate and more dependable positioning.

This chapter introduces the supported GNSS features and performance of Quectel AG35 series QuecOpen module.

2.1. GNSS Features

- Supports multi-constellation GNSS systems including GPS, GLONASS, BeiDou, Galileo and QZSS.
- Supports SBAS including WAAS, EGNOS, MSAS and GAGAN.
- Supports gpsOneXTRA Assistance technology to deliver more accurate positioning with greater sensitivity.
- Supports AGPS such as SUPL to improve the startup performance such as TTFF.
- Supports precise point positioning and DGPS.
- Supports output positioning information in multiple frequencies, such as 1 Hz, 2 Hz, 5 Hz and 10 Hz.

2.2. GNSS Performance

Table 1: Positioning Performance

Item	Performance	Comment
2D positioning accuracy (50%, 68%, 95%)	< 2 m, < 2.5 m, < 5 m	Standalone mode in open sky
3D positioning accuracy (50%, 68%, 95%)	< 2.5 m, < 3 m, < 6 m	Standalone mode in open sky
Number of channels tracked simultaneously	40	
TTFF @ cold start	29 s	Standalone mode in open sky
TTFF @ warm start	27 s	Standalone mode in open sky

TTFF @ hot start	1 s	Standalone mode in open sky
Reacquisition time after a loss of lock of 30 s	≈ 1 s	In open sky
Reacquisition time after a loss of lock of 5 mins	≈ 2 s	In open sky
Acquisition sensitivity (cold start, 95%)	> -149 dBm	Cold start; 300 s timeout value
Tracking sensitivity	> -163 dBm	Standalone or MSB mode
Velocity accuracy (68%, 95%)	0.15 m/s, 0.3 m/s	Drive in a straight line at 30 m/s
Heading accuracy (68%, 95%)	0.2 deg, 0.5 deg	Drive in a straight line at 30 m/s
Maximum speed	1852 km/h	

NOTE

For details about positioning mode, refer to ***Chapter 3.4.5.1***.

3 GNSS APIs

3.1. Header File Path

The header file path: *ql-ol-sdk/ql-ol-extsdk/include/ql_mcm_gps.h*.

3.2. Example Path

The example path: *ql-ol-sdk/ql-ol-extsdk/example/example_gps.c*.

3.3. API Overview

Table 2: API Overview

Function	Description
QL_LOC_Client_Init()	Initializes a GNSS client.
QL_LOC_Client_Deinit()	Deregisters a GNSS client.
QL_LOC_AddRxIndMsgHandler()	Registers a callback function to process GNSS data.
QL_LOC_Set_Indications()	Sets callback data.
QL_LOC_Set_Position_Mode()	Sets position configuration items.
QL_LOC_Start_Navigation()	Starts GNSS.
QL_LOC_Stop_Navigation()	Stops GNSS.
QL_LOC_Get_Current_Location()	Gets current location data.
QL_LOC_Delete_Aiding_Data()	Deletes GNSS aiding data.

QL_LOC_InjectTime()	Injects UTC time to GNSS.
QL_LOC_InjectLocation()	Injects location data to GNSS.
QL_LOC_Xtra_InjectData()	Injects gpsOneXTRA data to GNSS.
QL_LOC_Xtra_InjectFile()	Injects gpsOneXTRA file to GNSS.
QL_LOC_Agps_SetServer()	Sets SUPL server address and port.
QL_LOC_Install_supl_cert()	Installs an SUPL certificate.
QL_LOC_unInstall_supl_cert()	Uninstalls an SUPL certificate.
QL_LOC_Set_Auto_Inject_Xtra()	Sets automatic injection of gpsOneXTRA data.
QL_LOC_Xtra_GetValidity()	Queries the validity of injected gpsOneXTRA data.

NOTES

1. The APIs introduced in this document are not applicable to the module that supports QDR and PPE. For details, please contact Quectel Technical Support (support@quectel.com).
2. Never call any of the above GNSS APIs in any callback function.

3.4. API Description

3.4.1. QL_LOC_Client_Init

This function initializes a GNSS client to create a GNSS session.

- **Prototype**

```
int QL_LOC_Client_Init(loc_client_handle_type    *ph_loc);
```

- **Parameter**

ph_loc:
[Out] The handle that is returned after the GNSS client is initialized and the GNSS session is created. This parameter is used in subsequent GNSS APIs.

- **Return Value**

0 Created a GNSS session successfully.
Others Failed to create a GNSS session.

3.4.2. QL_LOC_Client_Deinit

This function deregisters a GNSS client to release the GNSS session.

- **Prototype**

```
int QL_LOC_Client_Deinit(loc_client_handle_type    *ph_loc);
```

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

- **Return Value**

0 Released the GNSS session successfully.
Others Failed to release the GNSS session.

3.4.3. QL_LOC_AddRxIndMsgHandler

This function registers a callback function to process GNSS data.

- **Prototype**

```
int QL_LOC_AddRxIndMsgHandler(QL_LOC_RxIndMsgHandlerFunc_t handlerPtr, void* contextPtr);
```

- **Parameter**

handlerPtr:
[In] The callback function used to process GNSS data.

contextPtr:
[In] The parameters required by the callback function. See ***Chapter 3.4.3.1***.

- **Return Value**

0 Registered the callback function successfully.
Others Failed to register the callback function.

3.4.3.1. QL_LOC_RxIndMsgHandlerFunc_t

This callback function processes the GNSS data.

- **Prototype**

```
typedef void (*QL_LOC_RxIndMsgHandlerFunc_t)
(
    loc_client_handle_type    ph_loc,
    E_QL_LOC_NFY_MSG_ID_T     e_msg_id,
    void                      *pv_data,
    void                      *contextPtr
);
```

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

e_msg_id:
[In] Callback data ID.

pv_data:
[In] Callback data. See ***Chapter 4*** for the details.

contextPtr:
[In] Customized data.

- **Return Value**

None.

3.4.4. QL_LOC_Set_Indications

This function sets the callback data.

- **Prototype**

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

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

bit_mask:
[In] Bit mask of callback data. It is defined as below:

```
LOC_IND_LOCATION_INFO_ON        (1 << 0)    //Location data.
LOC_IND_STATUS_INFO_ON          (1 << 1)    //GNSS engine status data.
```

```
LOC_IND_SV_INFO_ON                      (1 << 2)        //Satellites related data.
LOC_IND_NMEA_INFO_ON                    (1 << 3)        //NMEA sentences.
LOC_IND_CAP_INFO_ON                     (1 << 4)        //Not supported
LOC_IND_UTC_TIME_REQ_ON                 (1 << 5)        //Request of UTC time injection
LOC_IND_XTRA_DATA_REQ_ON                (1 << 6)        //Request of gpsOneXTRA data injection
LOC_IND_AGPS_DATA_CONN_CMD_REQ_ON       (1 << 7)        //Not supported
LOC_IND_NI_NFY_USER_RESP_REQ_ON         (1 << 8)        //Not supported
```

- **Return Value**

0 Set callback data successfully.
Others Failed to set callback data.

3.4.5. QL_LOC_Set_Position_Mode

This function sets positioning configuration items.

- **Prototype**

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

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

pt_mode:
[In] Positioning configuration items. See ***Chapter 3.4.5.1***.

- **Return Value**

0 Set positioning configuration items successfully.
Others Failed to set positioning configuration items.

3.4.5.1. QL_LOC_POS_MODE_INFO_T

The structure of positioning configuration items is defined as below:

```
typedef struct
{
  E_QL_LOC_POS_MODE_T mode;     //Positioning mode. Only standalone and MSB are supported
E_QL_LOC_POS_RECURRENCE_T recurrence;       //Positioning recurrence mode.
  uint32_t                  min_interval;           //Positioning interval. Unit: ms.
uint32_t                    preferred_accuracy;     //Horizontal positioning accuracy. Unit: meter.
```

```
uint32_t                preferred_time;         //Positioning timeout. Unit: ms.
}QL_LOC_POS_MODE_INFO_T;
```

- **Parameter**

Type	Parameter	Description
E_QL_LOC_POS_MODE_T	*mode*	Positioning mode. *E_QL_LOC_POS_MODE_STANDALONE*: Standalone mode. *E_QL_LOC_POS_MODE_MS_BASED*: MSB mode (speed up positioning). *E_QL_LOC_POS_MODE_MS_ASSISTED*: MSA mode (not supported currently).
E_QL_LOC_POS_RECURRENCE_T	*recurrence*	Positioning recurrence mode. *E_QL_LOC_POS_RECURRENCE_PERIODIC*: Periodic positioning. *E_QL_LOC_POS_RECURRENCE_SINGLE*: Single positioning.
uint32_t	*min_interval*	Positioning interval. Unit: ms. Valid values are 100, 200, 500 and multiplies of 1000.
uint32_t	*preferred_accuracy*	Horizontal positioning accuracy. Unit: meter.
uint32_t	*preferred_time*	Positioning timeout. Unit: ms.

3.4.6. QL_LOC_Start_Navigation

This function starts the GNSS.

- **Prototype**

```
int QL_LOC_Start_Navigation(loc_client_handle_type ph_loc);
```

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

- **Return Value**

0 Started GNSS successfully.
Others Failed to start GNSS.

3.4.7. QL_LOC_Stop_Navigation

This function stops the GNSS.

- **Prototype**

```
int QL_LOC_Stop_Navigation(loc_client_handle_type ph_loc);
```

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

- **Return Value**

0 Stopped GNSS successfully.
Others Failed to stop GNSS.

3.4.8. QL_LOC_Get_Current_Location

This function gets the current location data.

- **Prototype**

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

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

pt_loc_info:
[Out] Location data. See ***Chapter 3.4.8.1***.

timeout_sec:
[In] Positioning timeout. Unit: ms.

- **Return Value**

0 Got location data successfully.
Others Failed to get location data.

3.4.8.1. QL_LOC_LOCATION_INFO_T

The structure of location data is defined as below:

```
typedef struct
{
uint32_t       size;
E_QL_LOC_LOCATION_VALID_FLAG flags;
E_QL_LOC_ULP_LOCATION_SOURCE position_source;
double          latitude;
double          longitude;
      double          altitude;
      float           speed;
      float           bearing;
      float           accuracy;
      int64_t         timestamp;
      int32_t         is_indoor;
      float           floor_number;
      uint32_t        raw_data_len;
      uint8_t         raw_data[QL_LOC_GPS_RAW_DATA_LEN_MAX];
      char            map_url[QL_LOC_GPS_LOCATION_MAP_URL_SIZE + 1];
      uint8_t         map_index[QL_LOC_GPS_LOCATION_MAP_IDX_SIZE];
}QL_LOC_LOCATION_INFO_T;
```

- **Parameter**

Type	Parameter	Description
uint32_t	*size*	Size of this structure.
E_QL_LOC_LOCATION_VALID_FLAG	*flags*	Data validity indication. If the mask is 1, it means the data in this structure is valid, and 0 indicates invalid. The value is a combination of the following masks. *E_QL_LOC_LOCATION_LAT_LONG_VALID*: Longitude and latitude. *E_QL_LOC_LOCATION_ALTITUDE_VALID*: Altitude. *E_QL_LOC_LOCATION_SPEED_VALID*: Speed. *E_QL_LOC_LOCATION_BEARING_VALID*: Bearing. *E_QL_LOC_LOCATION_ACCURACY_VALID*: Positioning accuracy. *E_QL_LOC_LOCATION_SOURCE_INFO_VALID*: Not supported. *E_QL_LOC_LOCATION_IS_INDOOR_VALID*: Not supported.

		E_QL_LOC_LOCATION_FLOOR_NUMBE_VALID: Not supported. *E_QL_LOC_LOCATION_MAP_URL_VALID*: Not supported. *E_QL_LOC_LOCATION_MAP_INDEX_VALID*: Not supported.
E_QL_LOC_ULP_LOCATION_SOURCE	*position_source*	Location data source.
double	*latitude*	Latitude. Range: -90 to 90. Unit: degree.
double	*longitude*	Longitude. Range: 0–180. Unit: degree.
double	*altitude*	Altitude. Unit: meter.
float	*speed*	Speed. Range: 0–540. Unit: m/s.
float	*bearing*	Bearing. Range: 0–360. Unit: degree.
float	*accuracy*	Horizontal accuracy. Unit: meter.
int64_t	*timestamp*	UTC time. Unit: ms.
int32_t	*is_indoor*	Indoor or not. (Invalid data)
float	*floor_number*	Floor number. (Invalid data)
int32_t	*raw_data_len*	Length of raw data. Range: 0–256. (Invalid data)
uint8_t	*raw_data*	Raw data. (Invalid data)
char	*map_url*	(Invalid data)
uint8_t	*map_index*	(Invalid data)

3.4.9. QL_LOC_Delete_Aiding_Data

This function deletes the GNSS aiding data.

- **Prototype**

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

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

flags:
[In] Specifies the type of data to be deleted. See ***Chapter 3.4.9.1***.

- **Return Value**

0 Deleted GNSS aiding data successfully.
Others Failed to delete GNSS aiding data.

NOTE

This function takes a certain time to delete the specified data, so wait for at least 2 seconds before calling the next API.

3.4.9.1. E_QL_LOC_DELETE_AIDING_DATA_TYPE_T

The enumeration of GNSS aiding data to be deleted is defined as below:

```
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;
```

- **Parameter**

Parameter	Description
E_QL_LOC_DELETE_EPHEMERIS	Deletes ephemeris data.
E_QL_LOC_DELETE_ALMANAC	Deletes almanac data.
E_QL_LOC_DELETE_POSITION	Deletes location data.
E_QL_LOC_DELETE_TIME	Deletes time information.
E_QL_LOC_DELETE_IONO	Deletes ionospheric data.
E_QL_LOC_DELETE_UTC	Deletes UTC data.
E_QL_LOC_DELETE_HEALTH	Deletes satellite health status data.
E_QL_LOC_DELETE_SVDIR	Deletes satellite direction data.
E_QL_LOC_DELETE_SVSTEER	Deletes satellite rotation speed data.
E_QL_LOC_DELETE_SADATA	Deletes satellite data.
E_QL_LOC_DELETE_RTI	Deletes RTI data.
E_QL_LOC_DELETE_CELLDB_INFO	Deletes cell database data.
E_QL_LOC_DELETE_ALMANAC_CORR	Deletes almanac correction data.
E_QL_LOC_DELETE_FREQ_BIAS_EST	Deletes frequency bias estimation.
E_QL_LOC_DELETE_EPHEMERIS_GLO	Deletes GLONASS ephemeris data.
E_QL_LOC_DELETE_ALMANAC_GLO	Deletes GLONASS almanac data.
E_QL_LOC_DELETE_SVDIR_GLO	Deletes GLONASS satellite direction data.
E_QL_LOC_DELETE_SVSTEER_GLO	Deletes GLONASS satellite rotation speed data.
E_QL_LOC_DELETE_ALMANAC_CORR_GLO	Deletes GLONASS almanac correction data.
E_QL_LOC_DELETE_TIME_GPS	Deletes GPS time data.
E_QL_LOC_DELETE_TIME_GLO	Deletes GLONASS time data.

E_QL_LOC_DELETE_ALL	Deletes all data.

3.4.10. QL_LOC_InjectTime

This function injects the UTC time to GNSS engine. Injecting UTC time before starting GNSS with *QL_LOC_Start_Navigation()* can speed up positioning.

- **Prototype**

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

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

pt_info:
[In] UTC time data to be injected. See ***Chapter 3.4.10.1***.

- **Return Value**

0 Injected the UTC time successfully.
Others Failed to inject the UTC time.

NOTE

The difference between the injected time and current UTC time should be less than 10 seconds, otherwise the positioning time will be increased.

3.4.10.1. QL_LOC_INJECT_TIME_INTO_T

The structure of the injected UTC time is defined as below:

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

- **Parameter**

Type	Parameter	Description
int64_t	*time*	The current UTC time, that is the number of milliseconds that have elapsed since 00:00:00 UTC on January 1, 1970.
int64_t	*time_reference*	Invalid parameter. Set it to 0.
int32_t	*uncertainty*	Time accuracy. Set it to 3500. Unit: ms.

3.4.11. QL_LOC_InjectLocation

This function injects the location data to GNSS engine.

- **Prototype**

```
int QL_LOC_InjectLocation( loc_client_handle_type ph_loc, QL_LOC_INJECT_LOCATION_INTO_T *pt_info);
```

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

pt_info:
[In] Location data to be injected. See ***Chapter 3.4.11.1***.

- **Return Value**

0 Injected the location data successfully.
Others Failed to inject the location data.

3.4.11.1. QL_LOC_INJECT_LOCATION_INTO_T

The structure of the location data to be injected is defined as below:

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

- **Parameter**

Type	Parameter	Description
double	*latitude*	Latitude. Unit: degree.
double	*longitude*	Longitude. Unit: degree.
float	*accuracy*	Accuracy. Unit: meter.

3.4.12. QL_LOC_Xtra_InjectData

This function injects the gpsOneXTRA assistance data to GNSS engine.

- **Prototype**

```
Int QL_LOC_Xtra_InjectData(loc_client_handle_type ph_loc, char *data, int length);
```

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

data:
[In] gpsOneXTRA data.

length:
[In] Length of gpsOneXTRA data.

- **Return Value**

0 Injected the gpsOneXTRA data successfully.
Others Failed to inject the gpsOneXTRA data.

3.4.13. QL_LOC_Xtra_InjectFile

This function injects the gpsOneXTRA file to GNSS engine.

- **Prototype**

```
int QL_LOC_Xtra_InjectFile( loc_client_handle_type ph_loc, char *filename);
```

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

filename:
[In] gpsOneXTRA file.

- **Return Value**

0 Injected the gpsOneXTRA file successfully.
Others Failed to inject the gpsOneXTRA file.

3.4.14. QL_LOC_Agps_SetServer

This function sets the SUPL server address and port.

- **Prototype**

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

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

pt_info:
[In] SUPL server address and port. See ***Chapter 3.4.14.1***.

- **Return Value**

0 Set the SUPL server address and port successfully.
Others Failed to set the SUPL server address and port.

3.4.14.1. QL_LOC_AGPS_SERVER_INTO_T

The structure of SUPL server address and port is defined as below:

```
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;
```

- **Parameter**

Type	Parameter	Description
E_QL_LOC_AGPS_TYPE_T	*e_agps_type*	Protocol type.
char	*host_name*	Server address.
uint32_t	*port*	Port.

3.4.15. QL_LOC_Install_supl_cert

This function installs an SUPL certificate.

- **Prototype**

```
int  QL_LOC_Install_supl_cert(loc_client_handle_type    ph_loc,
                              uint8_t    suplCertificateId,
                              uint16_t  suplCertificateLen,
                              const  char*    suplCertificatePtr);
```

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

suplCertificateId:
[In] SUPL certificate ID. Customized parameter. Range: 0–9.

suplCertificateLen:
[In] Length of the SUPL certificate. Unit: byte.

suplCertificatePtr:
[In] Certificate content.

- **Return Value**

0 Installed the SUPL certificate successfully.
Others Failed to install the SUPL certificate.

3.4.16. QL_LOC_unInstall_supl_cert

This function uninstalls an SUPL certificate.

- **Prototype**

```
int QL_LOC_unInstall_supl_cert(loc_client_handle_type ph_loc, mcm_gps_supl_cert_slot_t_v01 suplCertificateId);
```

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

suplCertificateId:
[In] SUPL certificate ID. Range: 0–9.

- **Return Value**

0 Uninstalled the SUPL certificate successfully.
Others Failed to uninstall the SUPL certificate.

3.4.17. QL_LOC_Set_Auto_Inject_Xtra

This function sets automatic injection of gpsOneXTRA data after the previous data is expired.

- **Prototype**

```
int QL_LOC_Set_Auto_Inject_Xtra(loc_client_handle_type ph_loc, unsigned int autoInject);
```

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

autoInject:
[In] Whether to enable automatic injection of gpsOneXTRA data after the previous data is expired.
0 Disable
1 Enable

- **Return Value**

0 Set successfully.
Others Failed to set.

3.4.18. QL_LOC_Xtra_GetValidity

This function queries the validity of the injected gpsOneXTRA data.

- **Prototype**

```
int  QL_LOC_Xtra_GetValidity(loc_client_handle_type  ph_loc,
                             uint64_t*  startTimeUtc,
                             uint16_t*  durationHour);
```

- **Parameter**

ph_loc:
[In] The handle returned by *QL_LOC_Client_Init()*.

startTimeUtc:
[Out] gpsOneXTRA file start time, that is the number of seconds that have elapsed since 00:00:00 UTC on January 1, 1970.

durationHour:
[Out] Effective duration of a gpsOneXTRA file. Unit: hour.

- **Return Value**

0 Got the validity of the injected gpsOneXTRA data successfully.
Others Failed to get the validity of the injected gpsOneXTRA data.

4 Callback Data

This chapter introduces the callback data of *QL_LOC_RxIndMsgHandlerFunc_t()*.

4.1. Satellite Status Data

The structure of satellite status *QL_LOC_SV_STATUS_T* is defined as below:

```
typedef struct
{
    uint32_t                size;                                   /**<  Set to the size of mcm_gps_sv_status_t. */
    int                     num_svs;                                /**<  Number of SVs currently visible. */
    QL_LOC_SV_INFO_T        sv_list[QL_LOC_GPS_SUPPORT_SVS_MAX];    /**<  Contains an array of SV information. */
    uint32_t                ephemeris_mask;                         /**<  Bitmask indicating which SVs have ephemeris data.  */
    uint32_t                almanac_mask;                           /**<  Bitmask indicating which SVs have almanac data.  */
    uint32_t                used_in_fix_mask;                       /**<  Bitmask indicating which SVs were used for computing the most recent position fix. */
}QL_LOC_SV_STATUS_T;
```

- **Parameter**

Type	Parameter	Description
uint32_t	*size*	Size of this structure.
int	*num_svs*	Number of the visible satellites.
QL_LOC_SV_INFO_T	*sv_list*	Satellite number. Range: 0–32. See ***Chapter 4.2***.
uint32_t	*ephemeris_mask*	Bitmask indicating whether there is valid ephemeris.
uint32_t	*almanac_mask*	Bitmask indicating whether there is valid almanac.
uint32_t	*used_in_fix_mask*	Bitmask indicating whether the satellite is used in

	positioning.

4.2. General Satellite Data

The structure of general satellite data *QL_LOC_SV_INFO_T* is defined as below:

```
typedef struct
{
    uint32_t    size;          /**<  Set to the size of mcm_gps_sv_info_t. */
    int         prn;           /**<  Pseudo-random number for the SV. */
    float       snr;           /**<  Signal-to-noise ratio. */
    float       elevation;     /**<  Elevation of the SV in degrees. */
    float       azimuth;       /**<  Azimuth of the SV in degrees. */
}QL_LOC_SV_INFO_T;
```

- **Parameter**

Type	Parameter	Description
uint32_t	*size*	Size of this structure.
int	*prn*	Satellite ID.
float	*snr*	Signal to noise ratio. Range: 0–99.
float	*elevation*	Satellite elevation angle. Range: 0–90. Unit: degree.
float	*azimuth*	Satellite azimuth angle. Range: 0–360. Unit: degree.

5 gpsOneXTRA Assistance Technology

5.1. gpsOneXTRA Features

Quectel AG35 series QuecOpen module applies gpsOneXTRA Assistance technology to speed up GNSS positioning. This technology has the following features.

- Expands capabilities of standalone GNSS.
- Provides assistance data, such as almanac, ephemeris, ionospheric data and satellite data.
- Supports GPS, GLONASS, BeiDou and Galileo navigation systems.
- Injects gpsOneXTRA data to reduce TTFF by 10–18 s, achieving an average TTFF of 18–30 s.

5.2. gpsOneXTRA Data Downloading

The gpsOneXTRA data can be obtained from either of the following addresses:

http://xtrapath4.izatcloud.net/xtra2.bin
http://xtrapath5.izatcloud.net/xtra2.bin
http://xtrapath6.izatcloud.net/xtra2.bin
http://xtrapath4.izatcloud.net/xtra3grc.bin
http://xtrapath5.izatcloud.net/xtra3grc.bin
http://xtrapath6.izatcloud.net/xtra3grc.bin

5.3. gpsOneXTRA Data Injection

Make sure the following conditions are satisfied before injecting the gpsOneXTRA data by calling *QL_LOC_Xtra_InjectData()*.

1) GNSS is cold started, or the gpsOneXTRA data has expired.
2) Start the GNSS by calling *QL_LOC_Start_Navigation()*.
3) Inject the UTC time by calling *QL_LOC_InjectTime()*.

6 SUPL for AGPS

6.1. SUPL Description

AGPS significantly improves the TTFF of a GPS system. Quectel AG35 series QuecOpen module supports AGPS based on SUPL 2.0.

Secure User Plane Location (SUPL) is an Enabler that utilizes existing standards where available and possible, to transfer assistance data and positioning data over a User Plane bearer, such as IP, to aid network and SET based positioning technologies in the calculation of a SET's position.

NOTE

SUPL is available only when it is supported by the third-party service provider.

6.2. Implementation of SUPL

Follow the steps below to implement SUPL:

1. Call *QL_LOC_Agps_SetServer()* to set SUPL server address and port.
2. Call *QL_LOC_Install_supl_cert()* to install SUPL certificate obtained from the third-party service provider.
3. Start GNSS in MSB mode.

After implementing SUPL, the cold start TTFF is about 10 s in open sky.

6.3. Example

```
include <ql_oe.h>
int main(int argc, char *argv[])
{
```

```
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};

/*Init gnss client*/
ret = QL_LOC_Client_Init(&h_loc);
/*Registe callback*/
ret = QL_LOC_AddRxIndMsgHandler(ql_loc_rx_ind_msg_cb, (void*)h_loc);

/* Set what we want callbacks for */
ret = QL_LOC_Set_Indications(h_loc, 511);

/* Set position mode */
t_mode.mode                         = E_QL_LOC_POS_MODE_MS_BASED;
t_mode.recurrence                 = E_QL_LOC_POS_RECURRENCE_PERIODIC;
t_mode.min_interval               = 1000;    //report nmea frequency 1Hz
t_mode.preferred_accuracy     = 50;       // <50m
t_mode.preferred_time           = 90;       // 90s
ret = QL_LOC_Set_Position_Mode(h_loc, &t_mode);
/*Set Supl server*/
QL_LOC_AGPS_SERVER_INTO_T agps_server;
agps_server.e_agps_type = E_QL_LOC_AGPS_TYPE_SUPL;
strcpy(agps_server.host_name ,"supl-test.qxwz.com");
agps_server.port = 7275;
ret = QL_LOC_Agps_SetServer(h_loc,&agps_server);

/*Set Supl server*/
ret = QL_LOC_Install_AgpsCert(h_loc,"123456789",10);
printf("QL_LOC_Install_AgpsCert ret %d\n",ret);
sleep(2);

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

while(1)
{
    int finish_flag = 0;
    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);


}
```

7 Appendix A References

Table 3: Terms and Abbreviations

Abbreviation	Description
AGPS	Assisted Global Positioning System
API	Application Programming Interface
DB	Data Base
DGPS	Differential Global Positioning System
EGNOS	European Geostationary Navigation Overlay Service
GAGAN	GPS Aided Geo Augmented Navigation
GLONASS	Global Navigation Satellite System (Russia)
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IRNSS	Indian Regional Navigation Satellite System
MSA	Mobile Station Assisted
MSAS	Multi-functional Satellite Augmentation System (Japan)
MSB	Mobile Station Based
NavIC	Navigation with Indian Constellation
PPE	Precise Positioning Engine
QDR	Qualcomm Dead Reckoning
QZSS	Quasi-Zenith Satellite System
RTI	Real Time Integration
SA	Satellite
SBAS	Satellite-Based Augmentation System

SDCM	System of Differential Correction and Monitoring
SDK	Software Development Kit
SET	SUPL Enabled Terminal
SUPL	Secure User Plane Location
TTFF	Time To First Fix
UTC	Coordinated Universal Time
WAAS	Wide Area Augmentation System