EC200U Series QuecOpen(SDK) GNSS Development Guide

LTE Standard Module Series

Version: 1.0

Date: 2024-02-05

Status: Released

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our 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 offices. 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 us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Revision History

Version	Date	Author	Description
-	2021-11-05	Tyler LI/ Lambert ZHAO	Creation of the document
1.0	2024-02-05	Tyler LI/ Lambert ZHAO	First official release

Contents

About the Document **3**
Contents **4**
Table Index **6**

1 Introduction **7**

2 GNSS API **8**
2.1. Header File 8
2.2. API Overview 8
2.3. API Description 9
2.3.1. ql_gnss_switch 9
2.3.1.1. ql_GnssSW 9
2.3.1.2. ql_errcode_gnss 10
2.3.2. ql_gnss_callback_register 11
2.3.2.1. ql_gnss_callback 12
2.3.2.2. ql_uart_port_number_e 12
2.3.3. ql_gnss_agps_cfg 13
2.3.3.1. ql_AGPSGnssSW 13
2.3.4. ql_gnss_state_info_get 14
2.3.4.1. ql_GnssState 14
2.3.5. ql_gnss_nmea_get 15
2.3.6. ql_gnss_sys_type_cfg 15
2.3.7. ql_gnss_agps_param_cfg 16
2.3.8. ql_gnss_agps_param_cfg_ex 17
2.3.8.1. quec_agps_cfg_ex 17
2.3.8.2. quec_agps_apn_cfg 18
2.3.9. ql_gnss_agps_get_cfg 19
2.3.10. ql_auto_gnss_cfg 19
2.3.10.1. ql_AutoGnssSW 20
2.3.11. ql_gnss_apflash_cycle_update 20
2.3.11.1. ql_gnss_apflashupdate_e 20
2.3.12. ql_gnss_apflash_getpvdata 21
2.3.12.1. ql_gnss_data_t 21
2.3.13. ql_gnss_apflash_retry_enable 23
2.3.13.1. ql_gnss_apflashretry_e 23
2.3.14. ql_gnss_apflash_data_parse 24
2.3.15. ql_gnss_apflash_set_recv_status 24
2.3.15.1. ql_gnss_apflashdatarecv_e 25
2.3.16. ql_gnss_apflash_get_recv_status 25
2.3.17. ql_gnss_apflash_enable 25
2.3.17.1. ql_gnss_apflash_enable_e 26
2.3.18. ql_gnss_apflash_is_enable 26

3 Development Example and Debugging 27
3.1. Development Example 27
3.1.1. Example Description 27
3.1.2. Example Automatic Startup 28
3.1.3. Hibernation 28
3.1.4. AGPS 29
3.2. Debugging 29

4 Appendix References 31

Table Index

Table 1: API Overview 8
Table 2: Related Documents 31
Table 3: Terms and Abbreviations 31

1 Introduction

Quectel EC200U series module supports QuecOpen® solution. QuecOpen® is an embedded development platform based on RTOS. It is intended to simplify the design and development of IoT applications. For more information on QuecOpen®, see ***document [1]***.

This document is applicable to QuecOpen® solution based on SDK build environment. It provides information about GNSS, including GNSS API, development example, and application layer debugging on Quectel EC200U series module. Currently, EC200U series module only supports obtaining NMEA sentences output by GNSS chips and positioning information obtained through parsing.

EC200U series includes the following models:

- EC200U-CN
- EC200U-EU
- EC200U-AU

NOTE

1. Preset files for functions such as GNSS, Bluetooth and TTS in the file system cannot be deleted at will.
2. GNSS function is optional. Please contact Quectel Technical Support to confirm whether the model in use supports GNSS, if necessary.

2 GNSS API

2.1. Header File

ql_gnss.h, the header file of GNSS API, is in the *components\ql-kernel\inc* directory. Unless otherwise specified, all header files mentioned in this document are in this directory.

2.2. API Overview

Table 1: API Overview

Function	Description
ql_gnss_switch()	Enables or disables GNSS.
ql_gnss_callback_register()	Registers a callback function to read NMEA data.
ql_gnss_agps_cfg()	Enables or disables AGPS.
ql_gnss_state_info_get()	Gets GNSS state.
ql_gnss_nmea_get()	Gets GNSS NMEA sentences.
ql_gnss_sys_type_cfg()	Configures GNSS satellite constellation.
ql_gnss_agps_param_cfg()	Configures AGPS parameters.
ql_gnss_agps_param_cfg_ex()	Configures AGPS extended parameters.
ql_gnss_agps_get_cfg()	Gets the AGPS extended parameter configuration.
ql_auto_gnss_cfg()	Enables or disables the GNSS auto-start function.
ql_gnss_apflash_cycle_update()	Enables the periodic update function of AP flash data.
ql_gnss_apflash_getpvdata()	Gets the GNSS positioning data pointer.

ql_gnss_apflash_retry_enable()	Enables or disables the retry mechanism for getting AP flash data.
ql_gnss_apflash_data_parse()	Parses the AP flash data.
ql_gnss_apflash_set_recv_status()	Enables or disables the AP flash data reception.
ql_gnss_apflash_get_recv_status()	Gets the current state of AP flash data reception.
ql_gnss_apflash_enable()	Enables or disables AP flash.
ql_gnss_apflash_is_enable()	Gets AP flash status.

2.3. API Description

2.3.1. ql_gnss_switch

This function enables or disables GNSS on the modules that support GNSS. Modules without GNSS function will return an error code indicating the lack of GNSS support.

- **Prototype**

```
ql_errcode_gnss ql_gnss_switch(ql_GnssSW gnss_sw)
```

- **Parameter**

gnss_sw:
[In] Enabling/disabling GNSS. See ***Chapter 2.3.1.1*** for details.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.1.1. ql_GnssSW

Enumeration of enabling/disabling GNSS:

```
typedef enum
{
   GNSS_DISABLE,
   GNSS_ENABLE,
   GNSS_RESET
}ql_GnssSW;
```

- **Member**

Member	Description
GNSS_ENABLE	Enable GNSS.
GNSS_DISABLE	Disable GNSS.
GNSS_RESET	GNSS software reset.

2.3.1.2. ql_errcode_gnss

Enumeration of GNSS result codes:

```
typedef enum
{
QL_GNSS_SUCCESS  = QL_SUCCESS,

QL_GNSS_EXECUTE_ERR              = 1|QL_GNSS_ERRCODE_BASE,
QL_GNSS_INVALID_PARAM_ERR,
QL_GNSS_NOT_SUPPORT_ERR,
QL_GNSS_UART_SET_ERR,
QL_GNSS_CB_NULL_ERR,
QL_GNSS_ALREADY_OPEN,
QL_GNSS_NVM_WRITE_ERR
QL_GNSS_NEMA_PARSE_ERR
QL_GNSS_APFLASH_NO_ERR           = 50|QL_GNSS_ERRCODE_BASE,
QL_GNSS_APFLASH_MALLOC_ERR,
QL_GNSS_APFLASH_OVERFLOW_ERR,
QL_GNSS_APFLASH_HEAD_ERR,
QL_GNSS_APFLASH_RX_GOING,
QL_GNSS_APFLASH_RX_ERR,
QL_GNSS_APFLASH_RX_PASS
}ql_errcode_gnss;
```

- **Member**

Member	Description
QL_GNSS_SUCCESS	Successful execution.
QL_GNSS_EXECUTE_ERR	Failed execution.
QL_GNSS_INVALID_PARAM_ERR	Parameter error.

QL_GNSS_NOT_SUPPORT_ERR	The current firmware version does not support GNSS.
QL_GNSS_UART_SET_ERR	UART configuration error.
QL_GNSS_CB_NULL_ERR	The passed callback function is NULL.
QL_GNSS_ALREADY_OPEN	GNSS is already enabled.
QL_GNSS_NVM_WRITE_ERR	NVM writing error.
QL_GNSS_NEMA_PARSE_ERR	Failed to parse positioning information.
QL_GNSS_APFLASH_NO_ERR	Successful execution of AP flash function.
QL_GNSS_APFLASH_MALLOC_ERR	Failed to allocate AP flash memory.
QL_GNSS_APFLASH_OVERFLOW_ERR	AP flash data overflow error.
QL_GNSS_APFLASH_HEAD_ERR	Failed to parse AP flash data header.
QL_GNSS_APFLASH_RX_GOING	AP flash is receiving data.
QL_GNSS_APFLASH_RX_ERR	Failed AP flash data reception.
QL_GNSS_APFLASH_RX_PASS	Successful AP flash data reception.

2.3.2. ql_gnss_callback_register

This function registers a callback function to read NMEA data.

- **Prototype**

```
ql_errcode_gnss ql_gnss_callback_register(ql_gnss_callback gnss_cb)
```

- **Parameter**

gnss_cb:
[In] The callback function to be registered. See ***Chapter 2.3.2.1*** for details.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.2.1. ql_gnss_callback

- **Prototype**

```
typedef void (*ql_gnss_callback)(uint32 ind_type, ql_uart_port_number_e port, uint32 size);
```

- **Parameter**

ind_type:
[In] Event type.

port:
[In] UART communicating with GNSS. See ***Chapter 2.3.2.2*** for details.

size:
[In] Size of received data. Unit: byte.

2.3.2.2. ql_uart_port_number_e

Enumeration of UARTs communicated with GNSS:

```
typedef enum
{
    QL_UART_PORT_1,
    QL_UART_PORT_2,
    QL_UART_PORT_3,
    QL_USB_PORT_AT,
    QL_USB_PORT_MODEM,
    QL_USB_PORT_NMEA
    QL_PORT_MAX,
}ql_uart_port_number_e;
```

- **Member**

Member	Description
QL_UART_PORT_1	UART 1.
QL_UART_PORT_2	UART 2.
QL_UART_PORT_3	UART 3.
QL_USB_PORT_AT	AT port.
QL_USB_PORT_MODEM	Modem port.

QL_USB_PORT_NMEA	NMEA port.
QL_PORT_MAX	Reserved.

2.3.3. ql_gnss_agps_cfg

This function enables or disables AGPS.

- **Prototype**

```
ql_errcode_gnss ql_gnss_agps_cfg(ql_AGPSGnssSW gnssagpsflag);
```

- **Parameter**

gnssagpsflag:
[In] Enabling/disabling AGPS. See ***Chapter 2.3.3.1*** for details.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.3.1. ql_AGPSGnssSW

Enumeration of enabling/disabling AGPS:

```
typedef enum
{
   AGPS_GNSS_DISABLE,
   AGPS_GNSS_ENABLE
}ql_AGPSGnssSW;
```

- **Member**

Member	Description
AGPS_GNSS_ENABLE	Enable AGPS.
AGPS_GNSS_DISABLE	Disable AGPS.

2.3.4. ql_gnss_state_info_get

This function gets GNSS state.

- **Prototype**

```
ql_GnssState ql_gnss_state_info_get(void);
```

- **Parameter**

None

- **Return Value**

GNSS state. See ***Chapter 2.3.4.1*** for details.

2.3.4.1. ql_GnssState

Enumeration of GNSS states:

```
typedef enum
{
  GNSS_CLOSE,
  GNSS_FIRMWARE_UPDATE,
  GNSS_POSITIONING,
  GNSS_FIX
}ql_GnssState;
```

- **Member**

Member	Description
GNSS_CLOSE	GNSS is disabled.
GNSS_FIRMWARE_UPDATE	GNSS is loading firmware.
GNSS_POSITIONING	GNSS is positioning.
GNSS_FIX	GNSS positioning is successful.

2.3.5. ql_gnss_nmea_get

This function gets GNSS NMEA sentences.

- **Prototype**

```
ql_errcode_gnss ql_gnss_nmea_get(ql_uart_port_number_e uart_port, unsigned char *pbuff, uint32 len);
```

- **Parameter**

uart_port:
[In] The UART communicating with GNSS. Default: UART 3. Default baud rate: 115200 bps. See ***Chapter 2.3.2.2*** for details.

pbuff:
[In] Pointer of NMEA sentences, which is used to receive NMEA sentences of GNSS.

len:
[In] Length of NMEA sentences.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.6. ql_gnss_sys_type_cfg

This function configures GNSS satellite constellation. The configuration takes effect after the module is rebooted.

- **Prototype**

```
ql_errcode_gnss ql_gnss_sys_type_cfg(uint32 sys_type);
```

- **Parameter**

sys_type:
[In] GNSS satellite constellation. Default: 5.

0 GPS only
3 GPS + GLONASS + Galileo hybrid positioning
4 GPS + GLONASS hybrid positioning
5 GPS + BDS hybrid positioning (only supported by EC200U-CN)
GPS + BDS + Galileo hybrid positioning (only supported by EC200U-AU and EC200U-EU)
6 GPS + Galileo hybrid positioning
7 BDS only

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

NOTE

GNSS function is optional. Please contact Quectel Technical Support to confirm whether the model in use supports GNSS, if necessary.

2.3.7. ql_gnss_agps_param_cfg

This function configures AGPS parameters.

- **Prototype**

```
ql_errcode_gnss ql_gnss_agps_param_cfg(uint8 profile, const char *URL, const char *vendorID, const char *modelID, const char *password)
```

- **Parameter**

profile:
[In] PDP index.

URL:
[In] AGPS server address. Maximum length: 100 bytes.

vendorID:
[In] Username for logging into the AGPS server. Maximum length: 30 bytes.

modelID:
[In] User identification. Maximum length: 30 bytes.

password:
[In] Password for logging into the AGPS server. Maximum length: 30 bytes.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

NOTE

Once AGPS is enabled, the PDP context specified by *profile* is deactivated. Therefore, the PDP context used by AGPS cannot be shared simultaneously with other applications. Attempting to do so may result in abnormal disconnection of network applications, requiring a redial to reestablish the connection.

2.3.8. ql_gnss_agps_param_cfg_ex

This function configures AGPS extended parameters.

- **Prototype**

```
ql_errcode_gnss ql_gnss_agps_param_cfg_ex(quec_agps_cfg_ex *agpscfg_ex)
```

- **Parameter**

agpscfg_ex:
[In] AGPS extended parameters. See ***Chapter 2.3.8.1*** for details.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.8.1. quec_agps_cfg_ex

Structure of AGPS extended parameters:

```
typedef struct
{
   bool agpsflag;
    uint8 profile_inx;
    char agps_url[100];
    char agpsvendorID[30];
    char agpsmodelID[30];
    char agpspassWord[30];
   quec_agps_apn_cfg agps_apn_cfg;
}quec_agps_cfg_ex;
```

- **Parameter**

Type	Parameter	Description
bool	*agpsflag*	Enable or disable AGPS. See ***Chapter 2.3.3.1*** for details.
uint8	*profile_inx*	PDP index. Range: 1–7.
char	*agps_url*	AGPS server address. Maximum length: 100 bytes.
char	*agpsvendorID*	Username for logging into the AGPS server. Maximum length: 30 bytes.
char	*agpsmodelID*	User identification. Maximum length: 30 bytes.
char	*agpspassWord*	Password for logging into the AGPS server. Maximum length: 30 bytes.
quec_agps_apn_cfg	*agps_apn_cfg*	APN used by the PDP context. See ***Chapter 2.3.8.2*** for details.

NOTE

Once AGPS is enabled, the PDP context specified by *profile* is deactivated. Therefore, the PDP context used by AGPS cannot be shared simultaneously with other applications. Attempting to do so may result in abnormal disconnection of network applications, requiring a redial to reestablish the connection..

2.3.8.2. quec_agps_apn_cfg

Structure of APN used by the PDP context in AGPS:

```
typedef struct
{
    uint8 ip_version;
    char agps_apn[APN_LEN_MAX];
}quec_agps_apn_cfg;
```

- **Parameter**

Type	Parameter	Description
uint8	*ip_version*	Protocol type. *0* Specify the IP type set in the last activation of the PDP context *1* IPv4 *2* IPv6 *3* IPv4v6

		Note: If the specified PDP context has never been activated, the default value is 1.
char	*agps_apn*	Access point name. Maximum length: 99 bytes.

2.3.9. ql_gnss_agps_get_cfg

This function gets the AGPS extended parameter configuration.

- **Prototype**

```
ql_errcode_gnss ql_gnss_agps_get_cfg(quec_agps_cfg_ex *agpscfg_ex);
```

- **Parameter**

agpscfg_ex:
[In] AGPS extended parameters. See ***Chapter 2.3.8.1*** for details.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.10. ql_auto_gnss_cfg

This function enables or disables the GNSS auto-start function.

- **Prototype**

```
ql_errcode_gnss ql_auto_gnss_cfg(ql_AutoGnssSW autoflag);
```

- **Parameter**

autoflag:
[In] Enabling/disabling the GNSS auto-start function. See ***Chapter 2.3.10.1*** for details.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.10.1. ql_AutoGnssSW

Enumeration of enabling/disabling the GNSS auto-start function:

```
typedef enum
{
AUTO_GNSS_DISABLE,
AUTO_GNSS_ENABLE
}ql_AutoGnssSW;
```

- **Member**

Member	Description
AUTO_GNSS_DISABLE	Disable the GNSS auto-start function.
AUTO_GNSS_ENABLE	Enable the GNSS auto-start function.

2.3.11. ql_gnss_apflash_cycle_update

This function enables the periodic update function of AP flash data. After successful GNSS positioning, the AP flash data stored in the module will be automatically updated every hour.

- **Prototype**

```
ql_errcode_gnss ql_gnss_apflash_cycle_update(ql_gnss_apflashupdate_e status);
```

- **Parameter**

status:
[In] GNSS positioning status of AP flash. See ***Chapter 2.3.11.1*** for details.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.11.1. ql_gnss_apflashupdate_e

Enumeration of GNSS positioning status of AP flash:

```
typedef enum
{
APFLASH_POSITIONING = 0,
APFLASH_FIXOK
```

```
}ql_gnss_apflashupdate_e;
```

- **Member**

Member	Description
APFLASH_POSITIONING	GNSS is currently positioning.
APFLASH_FIXOK	Successful GNSS positioning.

2.3.12. ql_gnss_apflash_getpvdata

This function gets the GNSS positioning data pointer.

- **Prototype**

```
ql_errcode_gnss ql_gnss_apflash_getpvdata(ql_gnss_data_t *data);
```

- **Parameter**

data:
[In] Pointer to GNSS positioning data. See ***Chapter 2.3.12.1*** for details.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.12.1. ql_gnss_data_t

Structure of GNSS positioning data:

```
typedef struct ql_gnss_data_struct {
    unsigned char valid;
    double longitude;
    unsigned char longitude_cardinal;
    double latitude;
    unsigned char latitude_cardinal;
    float  hdop;
    float pdop;
    double heading;
    float gps_speed;
    unsigned char gps_signal;
    unsigned int max_cnr;
    unsigned int max_cnr2;
```

```
    unsigned int min_cnr;
    unsigned int avg_cnr;
    unsigned int cnrs[QL_GSV_MAX_SATS];
    unsigned int cnrs_index;
    unsigned int satellites_num;
    float altitude;
    char fwver[32];
    struct tm time;
    unsigned int UTC;
    unsigned char quality;
    unsigned char navmode;
}ql_gnss_data_t;
```

- **Parameter**

Type	Parameter	Description
unsigned char	*valid*	Valid positioning or not.
double	*longitude*	Longitude.
unsigned char	*longitude_cardinal*	Longitude E (East) or W (West).
double	*latitude*	Latitude.
unsigned char	*latitude_cardinal*	Latitude N (North) or S (South).
float	*hdop*	Horizontal dilution of precision.
float	*pdop*	Position dilution of precision.
double	*heading*	Course (Range: 0–360°).
float	*gps_speed*	Speed (Unit: km/h).
unsigned char	*gps_signal*	Signal strength (Maximum: 5).
unsigned int	*max_cnr*	Maximum signal value (Unit: dB).
unsigned int	*max_cnr2*	The second maximum signal value (Unit: dB). (Not supported currently)
unsigned int	*min_cnr*	Minimum signal value (Unit: dB).
unsigned int	*avg_cnr*	Average signal value (Unit: dB).
unsigned int	*cnrs*	Signal value data (Unit: dB).
unsigned int	*cnrs_index*	Signal value index.

unsigned int	*satellites_num*	The number of positioning satellites.
float	*altitude*	Positioning altitude.
char	*fwver*	Firmware version.
struct tm	*time*	Positioning date.
unsigned int	*UTC*	Positioning time.
unsigned char	*quality*	Current positioning quality.
unsigned char	*navmode*	Positioning mode.

2.3.13. ql_gnss_apflash_retry_enable

This function enables or disables the retry mechanism for getting AP flash data. After the first failure to get AP flash data, you can initiate another request to get AP flash data by calling this function.

- **Prototype**

```
ql_errcode_gnss ql_gnss_apflash_retry_enable(ql_gnss_apflashretry_e status);
```

- **Parameter**

status:
[In] Enabling/disabling the retry mechanism for getting the AP flash data. See ***Chapter 2.3.13.1*** for details.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.13.1. ql_gnss_apflashretry_e

Enumeration of enabling/disabling the retry mechanism for getting the AP flash data:

```
typedef enum
{
APFLASH_RETRY_DISABLE = 0,
APFLASH_RETRY_ENABLE
}ql_gnss_apflashretry_e;
```

- **Member**

Member	Description
APFLASH_RETRY_DISABLE	Disable the retry mechanism for getting the AP flash data.
APFLASH_RETRY_ENABLE	Enable the retry mechanism for getting the AP flash data.

2.3.14. ql_gnss_apflash_data_parse

This function parses the AP flash data.

- **Prototype**

```
ql_errcode_gnss ql_gnss_apflash_data_parse(unsigned char *pbuff, uint32 len);
```

- **Parameter**

pbuff:
[In] Pointer to AP flash data.

len:
[In] Length of AP flash data. Unit: Byte.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.15. ql_gnss_apflash_set_recv_status

This function enables or disables the AP flash data reception.

- **Prototype**

```
ql_errcode_gnss ql_gnss_apflash_set_recv_status(ql_gnss_apflashdatarecv_e status);
```

- **Parameter**

status:
[In] Enabling/disabling the AP flash data reception. See ***Chapter 2.3.15.1*** for details.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.15.1. ql_gnss_apflashdatarecv_e

Enumeration of enabling/disabling the AP flash data reception:

```
typedef enum
{
APFLASH_DATA_NOT_RECV = 0,
APFLASH_DATA_SET_RECV
}ql_gnss_apflashdatarecv_e;
```

- **Member**

Member	Description
APFLASH_DATA_NOT_RECV	Disable the AP flash data reception.
APFLASH_DATA_SET_RECV	Enable the AP flash data reception.

2.3.16. ql_gnss_apflash_get_recv_status

This function gets the current status of the AP flash data reception.

- **Prototype**

```
ql_errcode_gnss ql_gnss_apflash_get_recv_status(ql_gnss_apflashdatarecv_e *status);
```

- **Parameter**

status:
[Out] Current status indicating the AP flash data reception is enabled or disabled.
APFLASH_DATA_NOT_RECV The AP flash data reception is disabled
APFLASH_DATA_SET_RECV The AP flash data reception is enabled

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.17. ql_gnss_apflash_enable

This function enables or disables AP flash.

- **Prototype**

```
ql_errcode_gnss ql_gnss_apflash_enable(ql_gnss_apflash_enable_e status);
```

- **Parameter**

status:
[In] Enabling/disabling AP flash. See ***Chapter 2.3.17.1*** for details.

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

2.3.17.1. ql_gnss_apflash_enable_e

Enumeration of enabling/disabling AP flash:

```
typedef enum
{
APFLASH_DISABLE = 0,
APFLASH_ENABLE
}ql_gnss_apflash_enable_e
```

- **Member**

Member	Description
APFLASH_DISABLE	Disable AP flash.
APFLASH_ENABLE	Enable AP flash.

2.3.18. ql_gnss_apflash_is_enable

This function gets AP flash status.

- **Prototype**

```
ql_errcode_gnss ql_gnss_apflash_is_enable(ql_gnss_apflash_enable_e *status);
```

- **Parameter**

status:
[In] Status indicating whether the AP flash is enabled or disabled.
APFLASH_DISABLE The AP flash is disabled
APFLASH_ENABLE The AP flash is enabled

- **Return Value**

See ***Chapter 2.3.1.2*** for details.

3 Development Example and Debugging

This chapter explains how to use the above GNSS API in the application layer for development and debugging. The example demonstrates the process of enabling GNSS with related API, as well as outputting the NMEA sentences through UART3, with the default baud rate of 115200 bps.

3.1. Development Example

3.1.1. Example Description

The demo file for the GNSS API, *gnss_demo.c*, is available in the QuecOpen SDK at *components\ql-application\gnss*. The example covers the following aspects:

- GNSS enablement
- GNSS firmware update
- NMEA sentence retrieval
- Positioning information retrieval

In the example, GNSS is enabled first, then UART3 is configured and enabled (GNSS is connected to UART3 by default), and the GNSS NMEA sentences are printed when data is received. The example entry function is *ql_gnss_app_init()*, as shown below:

```
void ql_gnss_app_init(void)
{
    QlOSStatus err = QL_OSI_SUCCESS;

    err = ql_rtos_task_create(&gnss_task, 1024, APP_PRIORITY_NORMAL, "ql_gnssdemo", ql_gnss_demo_thread, NULL, 5);
    if( err != QL_OSI_SUCCESS )
    {
        QL_GNSSDEMO_LOG("gnss demo task created failed");
    }
}
```

3.1.2. Example Automatic Startup

The above example is not started by default in *ql_init_demo_thread*, and but it can be set to automatically start by uncommenting *ql_gnss_app_init()* when the module boots.

```
static void ql_init_demo_thread(void *param)
{
    QL_INIT_LOG("init demo thread enter, param 0x%x", param);
#if 0
    ql_gpio_app_init();
    ql_gpioint_app_init();
#endif

#ifndef QL_APP_FEATURE_CAMERA
    ql_ledcfg_app_init();
#endif

#ifdef QL_APP_FEATURE_AUDIO
    ql_audio_app_init();
#endif
#ifdef QL_APP_FEATURE_LCD
    //ql_lcd_app_init();
#endif
    ql_nw_app_init();
    //ql_datacall_app_init();
    ql_osi_demo_init();

#ifdef QL_APP_FEATURE_GNSS
    //ql_gnss_app_init();
#endif
```

3.1.3. Hibernation

After the module is powered up and GNSS is enabled, the example calls *ql_gnss_switch(GNSS_DISABLE)* to disable GNSS and the GNSS engine enters hibernate mode without interruption of power supply or reboot. The module stops positioning, the current consumption is reduced and the ephemeris data is saved. If *ql_gnss_switch(GNSS_ENABLE)* is called within 2 hours to re-enable GNSS, the GNSS engine performs a hot start for fast positioning.

```
ret = ql_gnss_switch(GNSS_ENABLE);// Enable GNSS
if(ret == QL_GNSS_ALREADY_OPEN)
{
    QL_GNSSDEMO_LOG("GNSS demo already open");
}
if( ret == QL_GNSS_NOT_SUPPORT_ERR)
{
    goto exit;
}
while(ql_gnss_state_info_get()==GNSS_FIRMWARE_UPDATE)
{
  ql_rtos_task_sleep_ms(1000);
}
ql_gnss_callback_register(ql_gnss_notify_cb);
if(ret ==QL_GNSS_CB_NULL_ERR)
{
    goto exit;
}
ret = ql_gnss_switch(GNSS_DISABLE);// Go into hibernate mode
ql_rtos_task_sleep_s(1000);
ret = ql_gnss_switch(GNSS_ENABLE);// Wake up
```

3.1.4. AGPS

In the example, *ql_gnss_agps_param_cfg()* is called to configure the AGPS parameters, and then *ql_gnss_agps_cfg()* is called to enable AGPS. This allows for fast positioning in case of normal networking and correct parameters.

```
if(QL_GNSS_SUCCESS==ql_gnss_agps_param_cfg(1,\
"http://quectel-api1.rx-networks.cn/rxn-api/locationApi/rtcm",\
"wLgWwv6JQt",\
"Quectel",\
"aFltUERDZzZxeTY5cEp2eA=="))
{
    QL_GNSSDEMO_LOG("AGPS parameter config success\r\n");
}
if(ql_gnss_agps_cfg(AGPS_GNSS_ENABLE)==QL_GNSS_SUCCESS)
{
    QL_GNSSDEMO_LOG("AGPS open success\r\n");
}

/* open GNSS */
ret = ql_gnss_switch(GNSS_ENABLE);// Enable GNSS
```

3.2. Debugging

GNSS of EC200U series module can be debugged on Quectel LTE OPEN EVB.

The debugging steps are as follows:

1. Download the compiled firmware to the module.
2. Connect the USB port of LTE OPEN EVB to the PC with a USB cable.
3. Capture the log with the “cooltools” tool via USB AP Log Port. For details about log capture, see ***document [2]***.
4. The USB AP Log Port mainly display system debugging information. After you capture the log, you will be able to check the development example related information.

ql_gnss_app_init() starts automatically after the module boots and waits for the update to complete in case of a firmware update. NMEA sentences can be viewed printing through the log information.

The debugging logs printed by USB AP Log Port are shown below:

```
Description
[ql_GNSS][ql_gnss_nmea_get, 184] GNSS nmea data is $GNRMC,021643.000,V,0000.000000,N,00000.000000,E,0.000,0.000,,,E,N*2F$GNGGA,021643.000,0000.000000,N,00000.000
[ql_GNSSDEMO][ql_gnss_notify_cb, 56] gnss demo port is 2, size is 112
[ql_GNSS][ql_gnss_nmea_get, 184] GNSS nmea data is 000,E,0,00,127.000,0.000,M,0,M,,*6B$GNGLL,0000.000000,N,00000.000000,E,021643.000,V,N*5D
[ql_GNSSDEMO][ql_gnss_notify_cb, 56] gnss demo port is 2, size is 112
[ql_GNSS][ql_gnss_nmea_get, 184] GNSS nmea data is ,,,127.000,127.000,127.000*2A$GNGSA,A,1,,,,,,,,,,,,,127.000,127.000,127.000*2A
[ql_GNSSDEMO][ql_gnss_notify_cb, 56] gnss demo port is 2, size is 6
[ql_GNSS][ql_gnss_nmea_get, 184] GNSS nmea data is 0*68
[ql_GNSSDEMO][ql_gnss_notify_cb, 56] gnss demo port is 2, size is 112
[ql_GNSS][ql_gnss_nmea_get, 184] GNSS nmea data is $GNRMC,021644.000,V,0000.000000,N,00000.000000,E,0.000,0.000,,,E,N*28$GNGGA,021644.000,0000.000000,N,00000.000
[ql_GNSSDEMO][ql_gnss_notify_cb, 56] gnss demo port is 2, size is 112
[ql_GNSS][ql_gnss_nmea_get, 184] GNSS nmea data is 000,E,0,00,127.000,0.000,M,0,M,,*6C$GNGLL,0000.000000,N,00000.000000,E,021644.000,V,N*5A
[ql_GNSSDEMO][ql_gnss_notify_cb, 56] gnss demo port is 2, size is 112
[ql_GNSS][ql_gnss_nmea_get, 184] GNSS nmea data is ,,,127.000,127.000,127.000*2A$GNGSA,A,1,,,,,,,,,,,,,127.000,127.000,127.000*2A
[ql_GNSSDEMO][ql_gnss_notify_cb, 56] gnss demo port is 2, size is 6
[ql_GNSS][ql_gnss_nmea_get, 184] GNSS nmea data is 0*68
```

Figure 1: Debugging Logs Printed by USB AP Log Port

In addition, you can get the positioning information in the structure variable *ql_gnss_data_t g_gps_data*. *g_gps_data* is the variable name. For details about *ql_gnss_data_t*, see ***Chapter 2.3.12.1*** for details.

4 Appendix References

Table 2: Related Documents

Document Name
[1] Quectel_EC200U&EG91xU_Series_QuecOpen(SDK)_Quick_Start_Guide
[2] Quectel_EC200U&EG91xU_Series_QuecOpen(SDK)_Log_Capture_Guide

Table 3: Terms and Abbreviations

Abbreviation	Description
AGPS	Assisted GPS (Global Positioning System)
AP	Application Processor
API	Application Programming Interface
BDS	BeiDou Navigation Satellite System
EVB	Evaluation Board
Galileo	Galileo Satellite Navigation System (EU)
GLONASS	Global Navigation Satellite System (Russia)
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HDOP	Horizontal Dilution of Precision
NMEA	NMEA (National Marine Electronics Association) 0183 Interface Standard
PC	Personal Computer
PDOP	Position Dilution of Precision

RTOS	Real-Time Operating System
UART	Universal Asynchronous Receiver & Transmitter
USB	Universal Serial Bus
UTC	Coordinated Universal Time