Lx6&Lx0&LC86L&LG77L AGNSS Application Note

GNSS Module Series

Version: 1.1

Date: 2021-11-22

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

Document Information	
Title	**Lx6&Lx0&LC86L&LG77L AGNSS Application Note**
Subtitle	GNSS Module Series
Document Type	Application Note
Document Status	Released

Revision History

Revision	Date	Description
-	2017-04-28	Creation of the document
1.0	2017-04-28	First official release
1.1	2021-11-22	1. Modified the structure of the document. 2. Added information on Host EPO including the difference between Host EPO and Flash EPO, AGNSS implementation with Host EPO and relevant example, and Host EPO testing with QGNSS. 3. Added applicable modules (L26, L76, L76-L, L86, L96, L26-LB, L76-LB, L70, L80, L70-R, L80-R, LC86L and LG77L). 4. Added AGNSS requirements (Chapter 1.3). 5. Added the structure of Binary Protocol (Figure 5). 6. Added the description of Binary Protocol fields (Table 6). 7. Added AGNSS procedure with Flash EPO and description of the procedure (Chapter 3.1.3). 8. Added the following messages PMTK127, PMTK713, PMTK721, PMTK740 and PMTK741 (Chapter 4). 9. Added the content of Flash EPO testing with QGNSS (Chapter 5.1).

	10. Added the example on Flash EPO implementation (Chapter 6.1). 11. Deleted the description of error handing in data transfer procedure.

Contents

Table Index

Figure Index

1 Introduction

EPO (Extended Prediction Orbit) is an AGNSS feature provided by the chipset supplier, which enables the receiver to minimize TTFF and improve accuracy in weak signal conditions. This document mainly describes the download of EPO files, AGNSS implementation, EPO related PMTK commands and how to evaluate the EPO functionality through QGNSS tool.

1.1. Differences Between Host EPO and Flash EPO

Both Flash EPO and Host EPO allow the GNSS receiver to achieve a shorter TTFF, but their differences make each of them suitable for different applications.

Host EPO (also called Real Time AGNSS) allows the receiver to store in RAM up to 6 hours of assistance data which are sent to the receiver through NMEA PMTK commands listed in ***Chapter 4***. For Host EPO, there is no data retention after the GNSS receiver reboots and the data should be re-downloaded.

Flash EPO, on the other hand, allows the receiver to store in Flash 7 or 14 days' assistance data which are sent to the receiver through Binary Protocol defined by the chipset supplier. Flash EPO enables the receiver to reuse all assistance information stored in flash before the information expires. See ***Chapter 2.5*** for the validity period of EPO files.

Table 1: Differences Between Flash EPO and Host EPO

Item	Flash EPO	Host EPO
Storage Space	Flash	RAM
Storage Capacity	7 or 14 days' assistance data	6 hours' assistance data
Protocol	Binary	NMEA

NOTE

The maximum period that EPO data can be stored in Flash is 14 days for GPS-only EPO files, and 7 days for GPS+GLONASS EPO files. If a 30-day GPS-only EPO file is sent, only the first 14 days of EPO data will be stored. If a 30-day GPS+GLONASS EPO file is sent, only the first 7 days of EPO data will be stored.

1.2. Applicable Modules

Not all the applicable modules support both Flash EPO and Host EPO. See the following table for the applicable modules of this document and the type of EPO supported on each module.

Table 2: The Type of EPO Supported on Applicable Modules

Series	Module	Flash EPO	Host EPO
Lx6	L26	●	●
	L76	●	●
	L76-L	●	●
	L86	●	●
	L96	●	●
	L26-LB	●	●
	L76-LB	●	●
Lx0	L70	●	●
	L80	●	●
	L70-R	-	●
	L80-R	-	●
LC86L	LC86L	●	●
LG77L	LG77L	●	●

1.3. AGNSS Requirements

The host needs to provide the Reference Time, Reference Position and EPO data to the GNSS receiver. The information provided by the host needs to meet the following requirements so that the GNSS receiver can make better use of EPO:

- The **Reference Time** should be accurate within 3 s and must be specified in UTC time.
- The **Reference Position** should be accurate within 30 km around the position of the receiver. Keep in mind that if the receiver's view of the sky is limited, the accuracy of the Reference Position needs to be increased.
- The **EPO data** should be valid.

The receiver can benefit from any of the assistance data to improve the TTFF. All assistance data (Reference Time, Reference Position and EPO data) are useful but none of them are mandatory. If some of them are not available or have expired, it is recommended to avoid using them.

The host can send the Reference Time, Reference Position and EPO data to the GNSS receiver through the messages listed in following table. See ***Chapter 4*** for a detailed description of these messages.

Table 3: AGNSS Related Commands

Packet Type	Data Content
PMTK713	Reference Position.
PMTK721	GPS/GLONASS EPO data for a single satellite.
PMTK740	Reference UTC Time.
PMTK741	Reference Time and Position.

2 Download of EPO Files

Quectel does not provide any Service Level Agreement for EPO files. It is recommended to download EPO data to your own server and send them to devices so as to ensure the availability of EPO data.

2.1. Get EPO Files from Server

Table 4: Download URL of EPO Files

EPO Type	GNSS Type	EPO File URL	File Name
Unified QEPO	GPS only	http://wpepodownload.mediatek.com/QGPS.DAT?*vendorinfo*	Single name: QGPS.DAT
Unified QEPO	GPS + GLONASS	http://wpepodownload.mediatek.com/QG_R.DAT?*vendorinfo*	Single name: QG_R.DAT
EPO	GPS only	http://wpepodownload.mediatek.com/EPO_GPS_3_***X***.DAT?*vendorinfo*	***X*** = 1~10 EPO_GPS_3_1.DAT to EPO_GPS_3_10.DAT
EPO	GPS + GLONASS	http://wpepodownload.mediatek.com/EPO_GR_3_***X***.DAT?*vendorinfo*	***X*** = 1~10. EPO_GR_3_1.DAT to EPO_GR_3_10.DAT

The following shows a complete URL sample:
http://wpepodownload.mediatek.com/QGPS.DAT?*vendor=AAA&project=BBB&device_id=CCC*

- The query string starts with "*?*" and is separated by "*&*".
- The values of "*vendor*" and "*project*" (*AAA, BBB* in the example) are issued by Quectel, contact Quectel Technical Supports to get the value.
  The value of "*device_id*" (*CCC* in the example) contains two parts – one is assigned by Quectel and the other assigned by the customer. For example: if *CCC* = *XXX_YYY*, the value *XXX* is provided by Quectel and you can contact Quectel Technical Supports to get the value, while *YYY* can be assigned by yourself and it must be a unique value, such as IMEI. Each device must have a unique ID.

NOTE

As there can be a maximum of 30 days' predictions, there will be up to 10 files.
Slices of 30-day EPO:
_1 for days 1 to 3,
_2 for days 4 to 6,
...
_10 for days 28 to 30.

2.2. EPO Files Format

This part mainly illustrates the format of EPO files for GPS only and for GPS + GLONASS.

2.2.1. EPO Files Format – GPS Only

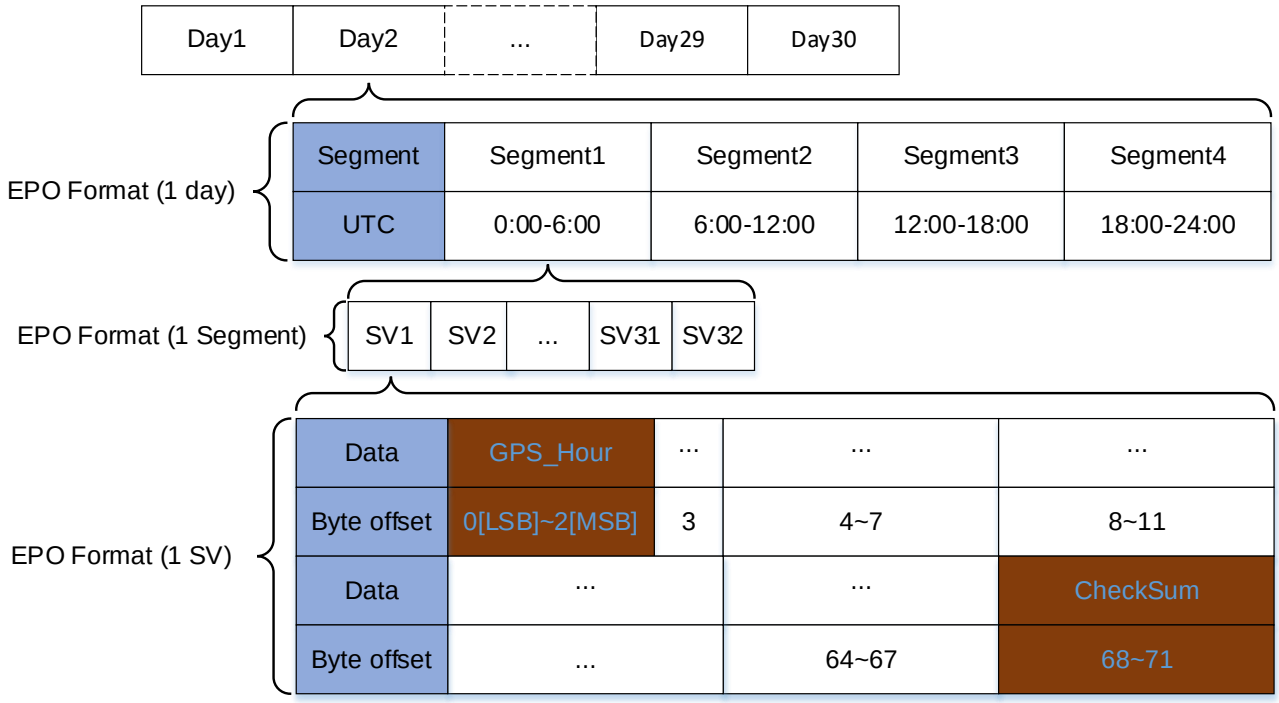

Figure 1: EPO Files Format – GPS Only

GPS_Secs = GPS_Hour * 3600
GPS_Week Number = GPS_Secs / 604800
GPS TOW = GPS_Secs % 604800

An EPO file contain GPS Time (GPS_Week, GPS_Hour and GPS_Secs). The maximum unit in GPS Time is GPS week which starts at approximately midnight of January 5th to 6th, 1980.

The following figure illustrates the format for several segments of EPO files.

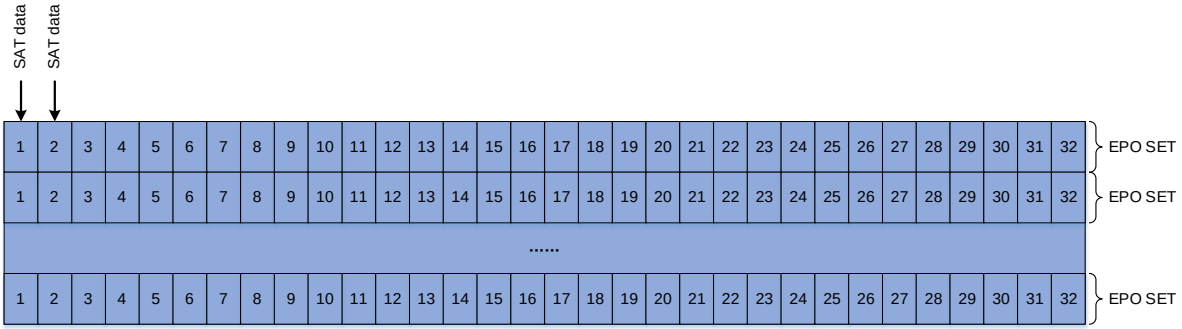

Figure 2: Format for Several Segments of EPO Files

The basic unit of an EPO file is SAT Data and the size of each SAT Data is 72 bytes. One EPO SET contains 32 SAT Data, so the data size of an EPO SET is 2304 bytes. Each EPO file contains several EPO SETs so the file size must be a multiple of 2304 bytes. An EPO SET is valid for 6 hours. Therefore, there will be 4 EPO SETs for one day.

2.2.2. EPO Files Format – GPS + GLONASS

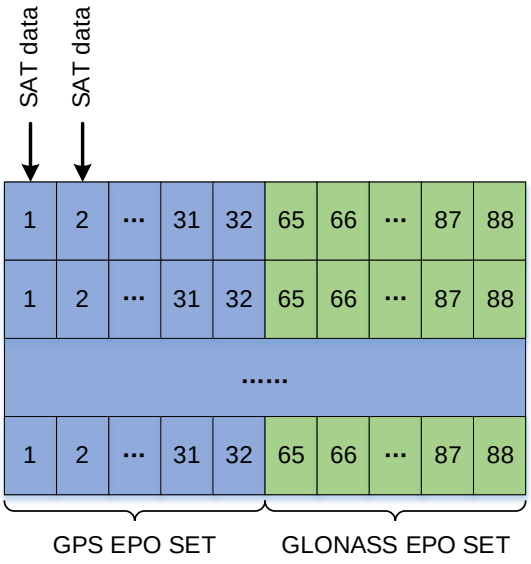

Figure 3: EPO Files Format – GPS + GLONASS

The basic unit of an EPO file is SAT Data, and the size of a SAT Data is 72 bytes. In GPS + GLONASS EPO files, one EPO SET contains 56 SAT Data, so the data size for an EPO SET is 4032 bytes. Each EPO file contains several EPO SETs. The file size must be a multiple of 4032 bytes. An EPO SET is valid

for 6 hours. Therefore, there will be 4 EPO SETs for one day.

2.3. Types of EPO Files

The EPO data can be downloaded in the form of files. You can select the most suitable file to download based on the availability of a data connection and storage space of your application. See *Table 4: Download URL of EPO Files* and *Table 5: Types of EPO Files* to decide on the file type to be downloaded.

Table 5: Types of EPO Files

EPO Type	GNSS Type	Description
Unified QEPO	GPS only	6-hour prediction orbit (ephemeris) Single file containing the latest GPS EPO data available.
Unified QEPO	GPS + GLONASS	6-hour prediction orbit (ephemeris) Single file containing the latest GPS + GLONASS EPO data available.
EPO	GPS only	3-30 days' prediction orbit (ephemeris) Split in 10 files and each contains 3-day information.
EPO	GPS + GLONASS	3-30 days' prediction orbit (ephemeris) Split in 10 files and each contains 3-day information.

2.4. Recommended Download Procedures of EPO Files

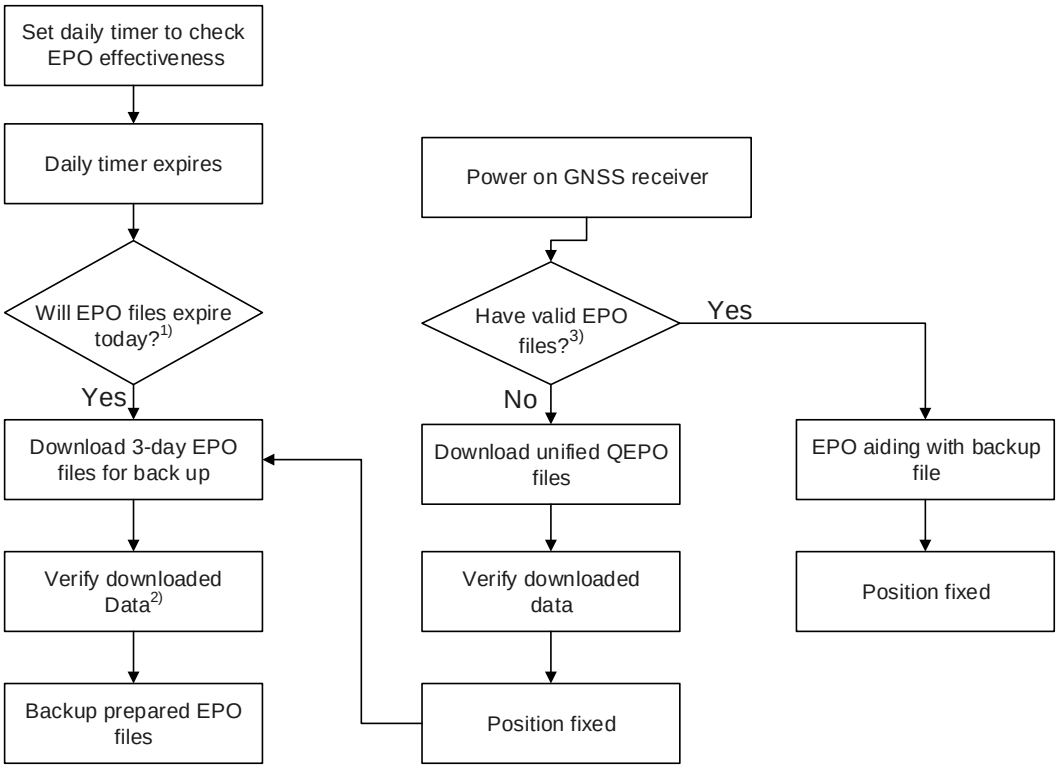

Figure 4: Recommended Download Procedures of EPO Files

NOTE

1. Users must know the current UTC time so as to download the valid EPO files.
2. Download MD5 checksum file by replacing file extension “DAT” with “MD5” for checking whether the data are correct.
3. If the device is powered off for a long time, EPO files stored in flash may expire.

2.5. The Validity Period of EPO Files

EPO validity period is related to the current UTC time. The EPO validity period can be obtained from the last segment of the EPO file. See *Figure 1: EPO Files Format – GPS Only* for the sample of how to calculate EPO validity period (GPS_Hour + 6). It is necessary to download the EPO file 12 hours in advance. The following codes show the conversion between UTC time and GPS time.

```
void utc_to_gpstime(kal_uint32  year,       //Input year
                    kal_uint8   mon,        //Input month: 1~12
                    kal_uint8   day,        //Input day: 1~31
                    kal_uint8   hour,       //Input hour: 0~23
                    kal_uint8   min,        //Input Minute: 0~59
                    kal_uint8   sec,        //Input second: 0~59
                    kal_int32*  wn,         //Output GPS week number
                    double*    tow)         //Output GPS time of week
{
    kal_int32 iYearsElapsed;                //Elapsed years since 1980
    kal_int32 iDaysElapsed;                 //Elapsed days since Jan 5/Jan 6, 1980
    kal_int32 iLeapDays;                    //Leap days since Jan 5/Jan 6, 1980
    kal_int32 i;
    //Number of days at the start of each month (ignore leap years).
    kal_uint16 doy[12] = {0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334};

    iYearsElapsed = year - 1980;
    i = 0;
    iLeapDays = 0;
    while (i <= iYearsElapsed)
    {
        if ((i % 100) == 20)
        {
            if ((i % 400) == 20)
            {
                iLeapDays++;
            }
        }
        else if ((i % 4) == 0)
        {
            iLeapDays++;
        }
        i++;
    }
    /*  iLeapDays = iYearsElapsed / 4 + 1; */.
    if ((iYearsElapsed % 100) == 20)
    {
        if (((iYearsElapsed % 400) == 20) && (mon <= 2))
        {
            iLeapDays--;
        }
    }
    else if (((iYearsElapsed % 4) == 0) && (mon <= 2))
    {
        iLeapDays--;
    }
    iDaysElapsed = iYearsElapsed * 365 + doy[mon - 1] + day + iLeapDays - 6;
    //Convert time to GPS weeks and seconds.
    *wn = iDaysElapsed / 7;
    *tow = (double)(iDaysElapsed % 7) * 86400 + hour * 3600 + min * 60 + sec;
}
```

3 AGNSS Implementation

This chapter describes two AGNSS implementation methods: Host EPO and Flash EPO.

- Implement AGNSS with Host EPO
  The host sends EPO data to the GNSS receiver through NMEA PMTK command, such as **PMTK721**.

- Implement AGNSS with Flash EPO
  The EPO data are downloaded to the flash of GNSS receiver through Binary Protocol.

Flash EPO keeps data for a longer time than Host EPO.

3.1. AGNSS with Flash EPO

Flash EPO can store 7 or 14 days' EPO assistance data on flash, which enables the receiver to make use of the available data since boot time. The communication protocol of Flash EPO is Binary Protocol. In order to download assistance data to the receiver, you first need to set the device to Binary mode through **PMTK253** so that the assistance data are downloaded in the binary format specified in this document. See ***Chapter 3.1.2*** and ***Chapter 3.1.3*** for details.

3.1.1. Binary Protocol

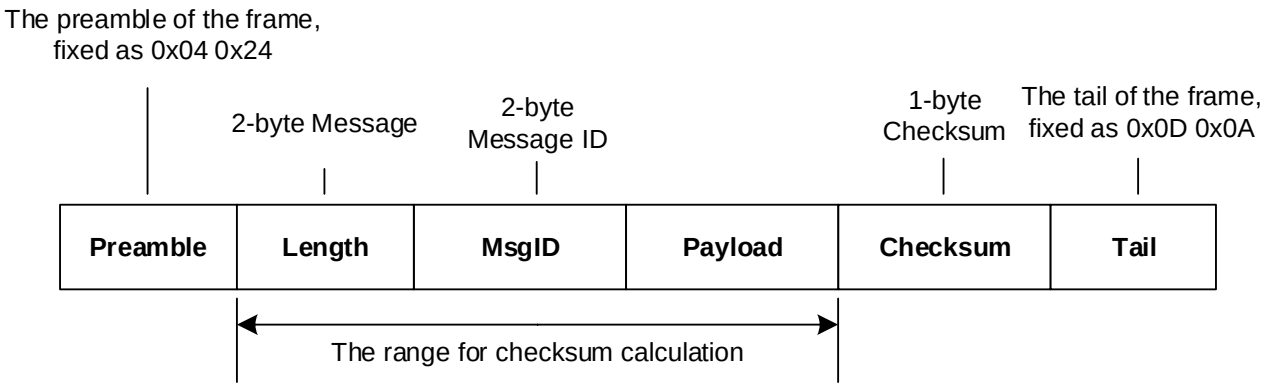

Figure 5: Structure of Binary Protocol

Table 6: Description of Binary Protocol Fields

Field	Length (Byte)	Description
Preamble	2	Fixed as 0x04 0x24. Use little endian.
Length	2	Total length of the messages from **Preamble** to **Tail**. Unit: byte. Maximum packet size: 256 bytes. Use little endian.
MsgID	2	Message ID.
Payload	Variable	Payload data to be transferred.
Checksum	1	The checksum is the 8-bit exclusive OR of all bytes in the message between (but not including) the **Preamble** and the **Checksum**
Tail	2	Fixed as 0x0D 0x0A. Use little endian.

3.1.1.1. MTK_BIN_EPO (MsgID = 723)

EPO binary packet is named as **MTK_BIN_EPO (MsgID = 723)** for convenience.

Table 7: MTK_BIN_EPO Format

Preamble	Length	MsgID	Payload				Checksum	Tail
0x04 0x24	0x00E3	0x02D3	EPO SEQ	SAT Data	SAT Data	SAT Data	0x**	0x0D 0x0A
2 Bytes	2 Bytes	2 Bytes	2 Bytes	72 Bytes	72 Bytes	72 Bytes	1 Byte	2 Bytes

An EPO file contains several SAT Data which are encapsulated in several **MTK_BIN_EPO** packets to be transferred to GNSS receiver. Each **MTK_BIN_EPO** packet contains a 2-byte EPO SEQ and 3 SAT Data fields. The length of **MTK_BIN_EPO** is 227 bytes. The EPO SEQ is used for synchronizing **MTK_BIN_EPO** packets in transfer protocol.

Sometimes, there is no enough EPO data to complete the three SAT Data fields. Some of the three fields can be left as blank, that is, to be filled with 0x00. An **MTK_BIN_EPO** packet that only contains 0–2 SAT Data is possible and acceptable. The following three **MTK_BIN_EPO** packets are examples:

Table 8: Format for MTK_BIN_EPO with 2 SAT Data

Preamble	Length	MsgID	Payload				Checksum	Tail
0x04 0x24	0x00E3	0x02D3	EPO SEQ	SAT Data	SAT Data	0x00	0x**	0x0D 0x0A
2 Bytes	2 Bytes	2 Bytes	2 Bytes	72 Bytes	72 Bytes	72 Bytes	1 Byte	2 Bytes

Table 9: Format for MTK_BIN_EPO with 1 SAT Data

Preamble	Length	MsgID	Payload				Checksum	Tail
0x04 0x24	0x00E3	0x02D3	EPO SEQ	SAT Data	0x00	0x00	0x**	0x0D 0x0A
2 Bytes	2 Bytes	2 Bytes	2 Bytes	72 Bytes	72 Bytes	72 Bytes	1 Byte	2 Bytes

Table 10: Format for MTK_BIN_EPO with No SAT Data

Preamble	Length	MsgID	Payload				Checksum	Tail
0x04 0x24	0x00E3	0x02D3	EPO SEQ	0x00	0x00	0x00	0x**	0x0D 0x0A
2 Bytes	2 Bytes	2 Bytes	2 Bytes	72 Bytes	72 Bytes	72 Bytes	1 Byte	2 Bytes

Table 11: Description of MTK_BIN_EPO Fields

Field	Length (Byte)	Description
EPO SEQ	2	Sequence number to indicate the corresponding received **MTK_BIN_EPO**.
SAT Data	72	Satellite EPO data.

GNSS receiver will return an ACK message for each received **MTK_BIN_EPO**. The message is named as **MTK_BIN_ACK_EPO** (**MsgID = 2**) for convenience. See ***Chapter 3.1.1.2*** for details.

3.1.1.2. MTK_BIN_ACK_EPO (Msg = 2)

This packet is usually returned after receiving an **MTK_BIN_EPO** packet.

Table 12: MTK_BIN_ACK_EPO Format

Preamble	Length	MsgID	Payload		Checksum	Tail
0x04 0x24	0x000C	0x0002	EPO SEQ	Result	0x**	0x0D 0x0A
2 Bytes	2 Bytes	2 Bytes	2 Bytes	1 Byte	1 Byte	2 Bytes

Table 13: Description of MTK_BIN_ACK_EPO Fields

Field	Length (Byte)	Description
EPO SEQ	2	Sequence number to indicate the corresponding received **MTK_BIN_EPO**.
Result	1	0: The received **MTK_BIN_EPO** is invalid 1: The received **MTK_BIN_EPO** is valid

Example:

```
//Received a valid EPO packet whose sequence number is 0x56:
0x04 0x24 0x0C 0x00 0x02 0x00 0x56 0x00 0x01 0x59 0x0D 0x0A

//Received an invalid EPO packet whose sequence number is 0x56:
0x04 0x24 0x0C 0x00 0x02 0x00 0x56 0x00 0x00 0x58 0x0D 0x0A
```

3.1.1.3. Change UART Format Packet (MsgID = 253)

Changes UART communication protocol and sets baud rate.

Table 14: Format for Change UART Format Packet

Preamble	Length	MsgID	Payload		Checksum	Tail
0x04 0x24	0x000E	0x00FD	Protocol	Baudrate	0x**	0x0D 0x0A
2 Bytes	2 Bytes	2 Bytes	1 Byte	4 Bytes	1 Byte	2 Bytes

Table 15: Description of Change UART Format Packet Fields

Field	Length (Byte)	Description
Protocol	1	0x00 = NMEA PMTK protocol 0x01 = Binary Protocol
Baudrate	4	UART baud rate. 0x00000000 = Default baud rate 00002580 = 9600 00004B00 = 19200 00009600 = 38400 0000E100 = 57600 0001C200 = 115200 00038400 = 230400 00070800 = 460800 000E1000 = 921600 Use little endian.

Example:

```
//Change UART to Binary Protocol and use baudrate 115200:
0x04 0x24 0x0E 0x00 0xFD 0x00 0x01 0x00 0xC2 0x01 0x00 0x31 0x0D 0x0A
//Change UART to PMTK protocol and use default baudrate:
0x04 0x24 0x0E 0x00 0xFD 0x00 0x00 0x00 0x00 0x00 0x00 0xF3 0x0D 0x0A
```

3.1.1.4. ACK Packet (MsgID = 1)

This packet is usually returned after receiving **Change UART Format Packet (MsgID = 253)**.

Table 16: ACK Packet Format

Preamble	Length	MsgID	Payload		Checksum	Tail
0x04 0x24	0x000C	0x0001	Responding MsgID	Flag	0x**	0x0D 0x0A
2 Bytes	2 Bytes	2 Bytes	2 Bytes	1 Byte	1 Byte	2 Bytes

Table 17: Description of ACK Packet Fields

Field	Length (Byte)	Description
Responding MsgID	2	Responding message ID.
Flag	1	0x00, 0x01: Invalid: the baud rate is invalid. 0x02: Failure: failed to set baud rate. 0x03: Success: succeeded to set baud rate.

Example:

```
//Received a valid binary packet and returned a success flag:
0x04 0x24 0x0C 0x00 0x01 0x00 0xFD 0x00 0x03 0xF3 0x0D 0x0A
```

3.1.2. EPO Data Transfer Protocol

EPO data are packed in **MTK_BIN_EPO** packets and then transferred to GNSS receiver. At the beginning of the transfer procedure, the host should split the EPO files and encapsulate them into several **MTK_BIN_EPO** packets, then assign a sequence number starting from zero for each **MTK_BIN_EPO** packet to make sure the **MTK_BIN_EPO** packets are transferred in correct order and not missed. The host should follow the EPO Data Transfer Protocol when transferring EPO data to GNSS receiver.

3.1.2.1. Pseudo Code for EPO Data Transfer Protocol

The following shows pseudo codes for EPO data transfer procedure, which are for reference only.

```
#define MTKBIN_3EPO_PKT_LNG 227

//At first, the protocol of the communication UART is supposed to be PMTK Protocol. Since EPO data are transferred by using binary packet, the protocol should be changed to Binary Protocol.
//Before starting EPO data transfer procedure, change the UART protocol setting by PMTK253. See Chapter 4.3 for the details of PMTK253.
//The SendPmtkCmd() function must be implemented by the programmer.
//It is recommended to explicitly specify a baud rate when changing UART packet protocol, for example, $PMTK253,1,115200*00<CR><LF>.

SendPmtkCmd ("$PMTK253,1,0*37\r\n");

//Now the data transferred via the UART port will be regarded as binary packet format. Please create a thread to transmit/receive binary packets for the UART. And the thread TMtkBinCmdThread() must be implemented by the programmer.

pMtkBinCmdThread() =new TMtkBinCmdThread();
```

```
//Read data in the EPO file, and then verify the validity of EPO data. If the inputted EPO file is not a valid EPO format, the programmer shall terminate the procedure.
//Please check whether the file size is a multiple of 2304 bytes or 4032 bytes.
//The fgEPO_Verify_File() function must be implemented by the programmer.

if (!fgEPO_Verify_File (pEpoFile))
return;

//Get total length of MTK_BIN_EPO packets that will be sent.
//Total number=ceil ((file size/72)/3)
//The i2EPO_Get_Num_Pkt function must be implemented by the programmer.

i4NumSvEpoPkt = i2EPO_Get_Num_Pkt(pEpoFile);

//Start EPO data transfer procedure to send EPO data.
u2EpoSeq=0;
u2LastEpoSeq=0;
for (i = 0; i<i4NumSvEpoPkt; i++)
{

//The fgEPO_Get_One_Pkt function takes out 3 SAT Data from the EPO file and encapsulates them in an MTK_BIN_EPO packet with appropriate EPO SEQ number.
//In order to save the total transfer time, it is recommended to generate a current EPO packet first, and then wait for MTK_BIN_ACK_EPO acknowledgement of the previous MTK_BIN_EPO packet from the GNSS receiver.
//The fgEPO_Get_One_Pkt function must be implemented by the programmer.

    if (fgEPO_Get_One_Pkt(u2EpoSeq, pEpoFile, szPktData))
    {

//Wait for EPO acknowledgment. The GNSS receiver will return an MTK_BIN_ACK_EPO acknowledgement packet after receiving and processing previous MTK_BIN_EPO packet. See Chapter 3.1.1.2 for the details.
//If the acknowledgment indicates failure, the process shall be terminated.
//The fgWait_Epo_Ack function must be implemented by the programmer
        if (!fgWait_Epo_Ack(u2LastEpoSeq))
        {
            return;
        }

//Send current MTK_BIN_EPO packet. The packet size of MTK_BIN_EPO is MTKBIN_3EPO_PKT_LNG.
//The function SendData must be implemented by the programmer.
```

```
        pPortMtkBinThread->SendData(szPktData, MTKBIN_3EPO_PKT_LNG);

//Update sequence number.

            u2LastEpoSeq = u2EpoSeq;
            u2EpoSeq++;
        }
}

//Generate final MTK_BIN_EPO packet to indicate the GNSS receiver that the process is finished.
//The fgEPO_Get_Final_Pkt function must be implemented by the programmer.

vEPO_Get_Final_Pkt(szPktData);

//Send final MTK_BIN_EPO packet to the GNSS receiver. The packet size of MTK_BIN_EPO is
MTKBIN_3EPO_PKT_LNG.
//Then the process is finished.
//The SendData function must be implemented by the programmer.

pPortMtkBinThread->SendData(szPktData, MTKBIN_3EPO_PKT_LNG);

//Switch UART protocol setting to PMTK packet format and change the baudrate to 115200 for the
communication UART. See Chapter 3.1.1.3 for the details.
//The SendMtkBinCmd function must be implemented by the programmer.

SendMtkBinCmd(0x04 0x24 0x0E 0x00 0xFD 0x00 0x00 0x00 0xC2 0x01 0x00 0x30 0x0D 0x0A);
```

3.1.3. AGNSS Procedure with Flash EPO

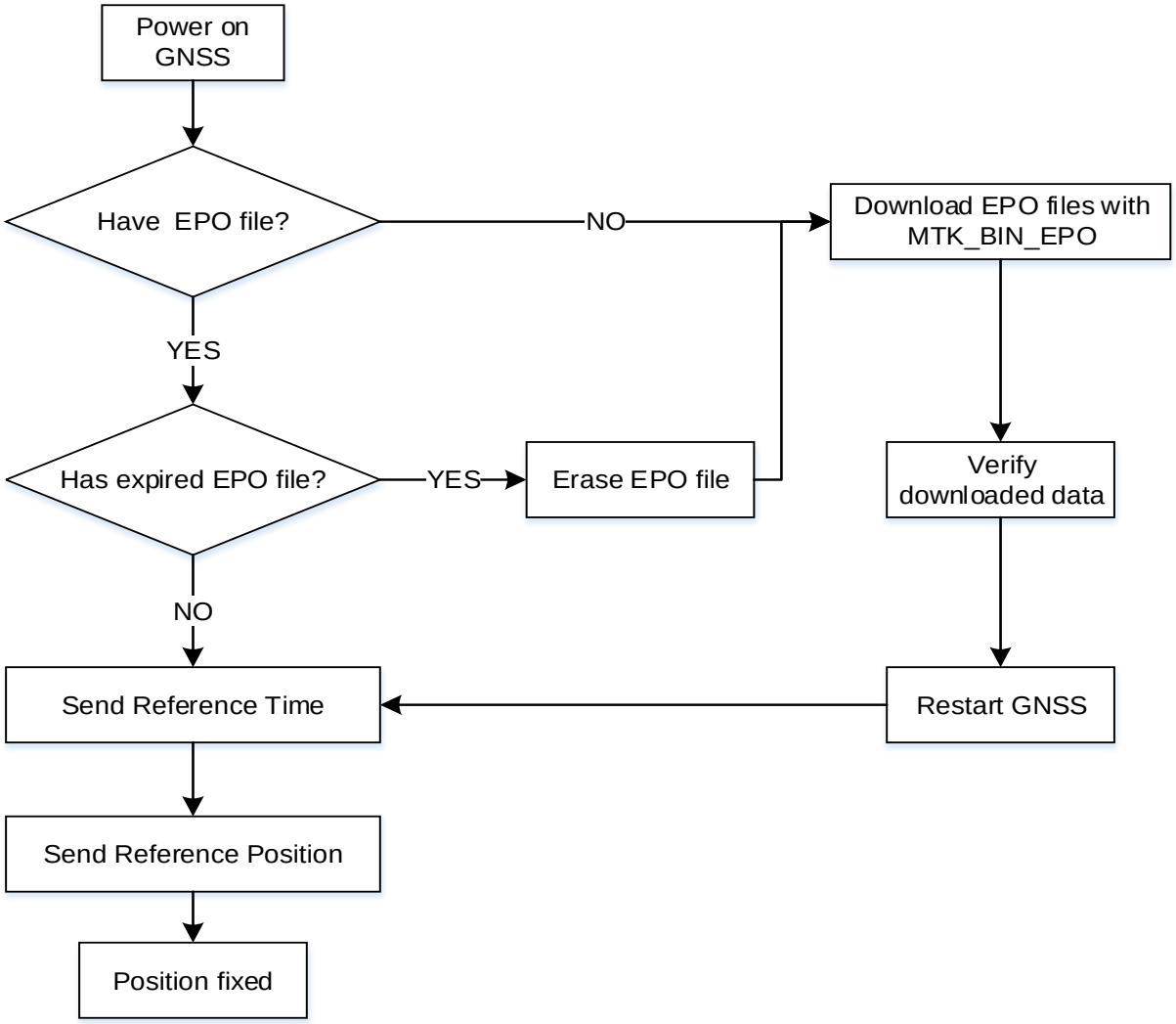

Figure 6: AGNSS Procedure with Flash EPO

1. Power on the GNSS module.
2. Check whether there are EPO data in GNSS module through **PMTK607**.
3. If there exists EPO data, go to the next step to check the data validity. Otherwise, download EPO data to GNSS module and verify downloaded data, then restart GNSS module and directly go to step 6.
4. Check whether the EPO file in GNSS module has expired.
5. If not, go to next step. Otherwise, erase expired EPO file through **PMTK127** and download a new EPO file.
6. Send reference time to GNSS module with **PMTK740**.
7. Send reference location to GNSS module with **PMTK741** or **PMTK713**.
8. Wait for GNSS module to fix position.

3.2. AGNSS with Host EPO

Host EPO allows for a simpler text-based implementation which enables the receiver to perform a fast start up where assistance data must be sent to the receiver each time it boots. When using Host EPO, the receiver can only receive one block of assistance data valid for 6 hours.

Implementing Host EPO only requires a few PMTK sentences and all data transfer can be done in NMEA mode. See ***Chapter 4*** for a detailed description of **PMTK713**, **PMTK721 PMTK740** and **PMTK741**. Both assistance time and position can be sent through **PMTK741**.

3.2.1. Recommended Sequence for Host EPO

After the GNSS receiver is powered on, it will output a start-up message **$PMTK010,001*2E** to notify the host that it has finished initialization and is capable of receiving PMTK commands. After the host receives the system startup message, it can send the assistance data in sequence shown in the *Figure 7: Suggested Sequence for Host EPO*. The sequence of assistance data is Reference Time, Reference Position and EPO data.

The Host EPO procedure consists of the following steps:

1. GNSS module starts up
2. The host sends Reference Time
3. The host sends Reference Position
4. The host sends EPO data.

The supplied Reference Time, Reference Position and EPO data must comply with the requirements listed in ***Chapter 1.3***.

NOTE

In the current implementation, the host needs to wait for a **PMTK001** packet to be returned before sending another segment of EPO data.

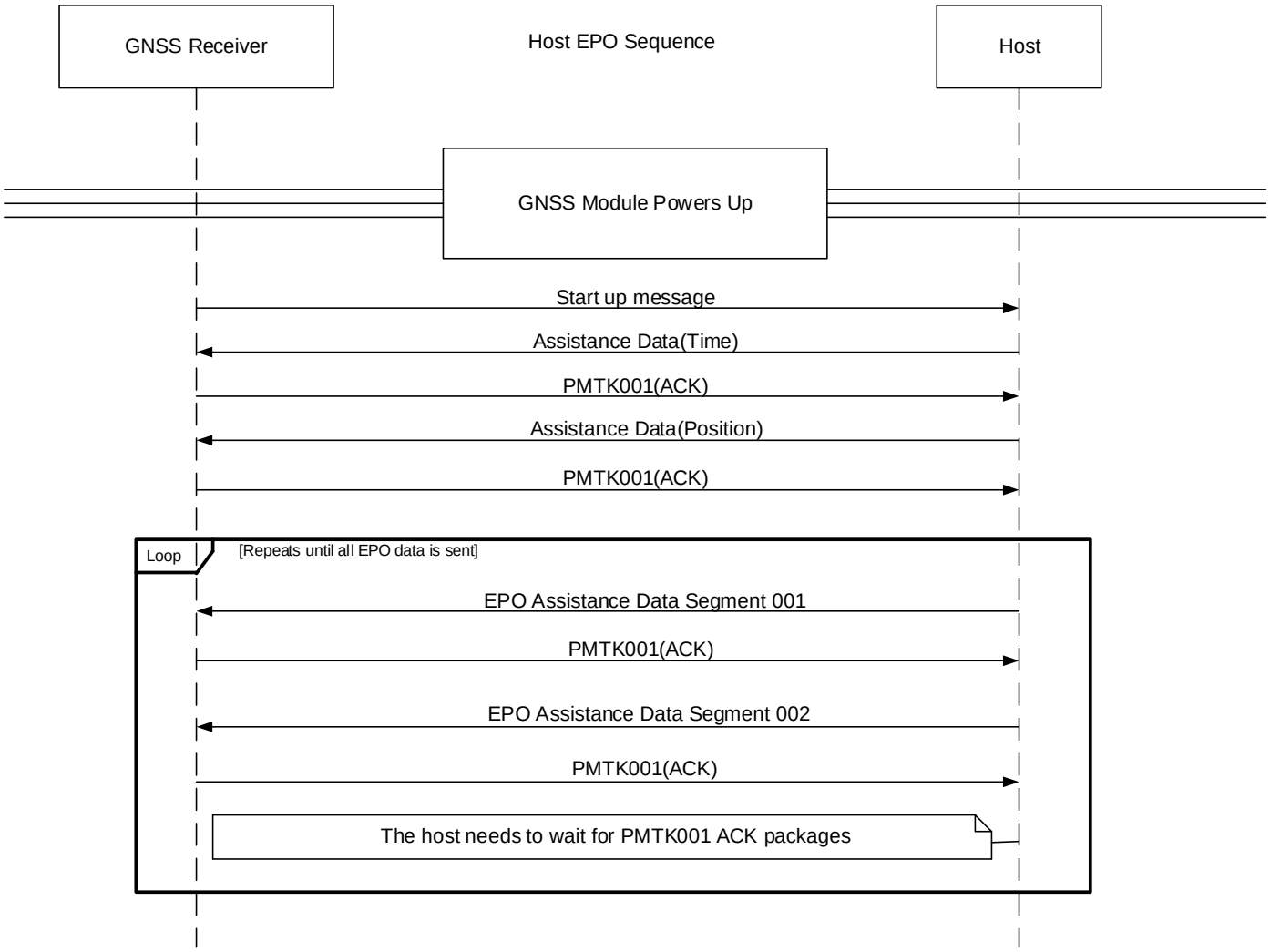

Figure 7: Suggested Sequence for Host EPO

3.2.2. Sample Code to Send EPO

The following is the reference code to send one segment of EPO data to GNSS chip. It reveals how to extract correct EPO segment from an EPO file and how to construct PMTK messages for GNSS receiver. PMTK messages for Reference Time, Reference Position, and second EPO segment are not included in this example.

```
#include <stdio.h>
#include <string.h>
#define MTKEPO_SV_NUMBER 32
#define MTKEPO_RECORD_SIZE 72
#define MTKEPO_SEGMENT_NUM (30 * 4)

unsigned char calc_nmea_checksum(const char* sentence)
{
    unsigned char checksum = 0;
    while (*sentence)
    {
```

```
        checksum ^= (unsigned char)*sentence++;
    }
    return checksum;
}

// translate UTC to GPS_Hour
int utc_to_gps_hour(int iYr, int iMo, int iDay, int iHr)
{
    int iYearsElapsed; // Years since 1980
    int iDaysElapsed; // Days elapsed since Jan 6, 1980
    int iLeapDays; // Leap days since Jan 6, 1980
    int i;
    // Number of days into the year at the start of each month (ignoring leap years)
    const unsigned short doy[12] = {0,31,59,90,120,151,181,212,243,273,304,334};
    iYearsElapsed = iYr - 1980;
    i = 0;
    iLeapDays = 0;
    while (i <= iYearsElapsed)
    {
        if ((i % 100) == 20)
        {
            if ((i % 400) == 20)
            {
                iLeapDays++;
            }
        }
        else if ((i % 4) == 0)
        {
            iLeapDays++;
        }
        i++;
    }
    if ((iYearsElapsed % 100) == 20)
    {
        if (((iYearsElapsed % 400) == 20) && (iMo <= 2))
        {
            iLeapDays--;
        }
    }
    else if (((iYearsElapsed % 4) == 0) && (iMo <= 2))
    {
        iLeapDays--;
    }
    iDaysElapsed = iYearsElapsed * 365 + (int)doy[iMo - 1] + iDay + iLeapDays - 6;
    // Convert time to GPS weeks and seconds
    return (iDaysElapsed * 24 + iHr);
}

void send_assistance_data(int iYr, int iMo, int iDay, int iHr)
{
    FILE* fp;
    int i, segment, epo_gps_hour, current_gps_hour;
    unsigned epobuf[MTKEPO_RECORD_SIZE/sizeof(unsigned)];
    char strbuf[200], outbuf[200];
    // open EPO file and read the header (assume EPO file has passed integrity check)
    if (NULL == (fp = fopen("MTKEPO.bin", "rb")))
```

```
    {
        return;
    }
    fread(&epo_gps_hour, 4, 1, fp);
    epo_gps_hour &= 0x00FFFFFF;
    // determine the segment to use
    current_gps_hour = utc_to_gps_hour(iYr, iMo, iDay, iHr);
    segment = (current_gps_hour - epo_gps_hour) / 6;
    if ((segment < 0) || (segment >= MTKEPO_SEGMENT_NUM))
    {
        return;
    }
    // read binary EPO data and sent it to MT3339
    fseek(fp, segment*(MTKEPO_RECORD_SIZE)*(MTKEPO_SV_NUMBER), SEEK_SET);
    for (i = 0; i < MTKEPO_SV_NUMBER; i++)
    {
        fread(epobuf, MTKEPO_RECORD_SIZE, 1, fp);
        // assume host system is little-endian

sprintf(strbuf,"PMTK721,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X",
        i+1,
        epobuf[0], epobuf[1], epobuf[2], epobuf[3], epobuf[4], epobuf[5],
        epobuf[6], epobuf[7], epobuf[8], epobuf[9], epobuf[10], epobuf[11],
        epobuf[12], epobuf[13], epobuf[14], epobuf[15], epobuf[16], epobuf[17]);
        sprintf(outbuf, "$%s*%02X\r\n", strbuf, calc_nmea_checksum(strbuf));
        // send them by UART
        // uart_send(outbuf, strlen(outbuf));
        printf("%s", outbuf); // for demo
    }
    fclose(fp);
}

int main (void)
{
    // get current system time
    // ... time();
    // send assistance data of current time
    send_assistance_data(2009, 8, 3, 22);
    return 0;
}
```

4 AGNSS Related Messages

4.1. PMTK001 PMTK_ACK

Acknowledges a PMTK command. This message is returned to inform the sender that the receiver has received the packet.

Type:

Output.

Synopsis:

```
$PMTK001,<CMD>,<Flag>*<Checksum>
```

Parameter:

Field	Format	Unit	Description
<CMD>	Numeric	-	The command types the ACK message responds.
<Flag>	Numeric	-	0 = Invalid command 1 = Unsupported command type 2 = Valid command, but action fails 3 = Valid command, and action succeeds

Example:

```
$PMTK001,0,3*30
```

4.2. PMTK127 PMTK_CMD_CLEAR_EPO

Erases the EPO data stored in the GNSS receiver.

Type:

Command.

Synopsis:

```
$PMTK127*<Checksum><CR><LF>
```

Parameter:

None.

Result:

Returns a **PMTK001** message.

Example:

```
$PMTK127*36
$PMTK001,127,3*34
```

4.3. PMTK253 PMTK_SET_OUTPUT_FMT

Sets data output format and baud rate for current port.

Type:

Set.

Synopsis:

```
$PMTK253,<Flag>,<Baudrate>*<Checksum><CR><LF>
```

Parameter:

Field	Unit	Default	Description
<Flag>	-	0	0 = NMEA mode 1 = Binary mode
<Baudrate>	bps	115200	Baud rate for the new output mode 0: use default baud rate (not recommended) It is highly recommended to specify an explicit baud rate value. possible values will be: 4800 9600 14400 19200 38400 57600

115200

Result:

Returns a **PMTK001** message.

Example:

```
//Switch from NMEA mode to Binary mode, and use default baud rate 115200 bps:
$PMTK253,1,0*37
//Switch from Binary mode to NMEA mode, and use baud rate 9600 bps:
$PMTK253,0,9600*09
$PMTK001,253,3*34
```

NOTE

When switching from Binary mode to NMEA mode, a binary ACK packet (**PMTK001**) will be returned after this message is executed. When switching from NMEA mode to Binary mode, no ACK packet is returned.

4.4. PMTK607 PMTK_Q_EPO_INFO

Queries the EPO data stored in the GNSS chip.

Type:

Query.

Synopsis:

```
$PMTK607*<Checksum><CR><LF>
```

Parameter:

None.

Result:

Returns a **PMTK707** message.

Example:

```
$PMTK607*33
$PMTK707,56,1468,172800,1470,151200,1468,259200,1468,259200*1F
```

4.5. PMTK707 PMTK_DT_EPO_INFO

This message is returned after receiving **PMTK607** and it contains EPO data stored in GNSS chip.

Type:

Output.

Synopsis:

```
$PMTK707,<Set>,<FWN>,<FTOW>,<LWN>,<LTOW>,<FCWN>,<FCTOW>,<LCWN>,<LCTOW>*<Checksum>
```

Parameter:

Field	Unit	Default	Description
<Set>	-	-	Total number of EPO SET stored in GNSS chip
<FWN>	-	-	GPS week number of the first EPO SET stored in GNSS chip
<FTOW>	-	-	GPS TOW of the first EPO SET stored in GNSS chip
<LWN>	-	-	GPS week number of the last EPO SET stored in GNSS chip
<LTOW>	-	-	GPS TOW of the last EPO SET stored in GNSS chip
<FCWN>	-	-	GPS week number of the first EPO SET currently used
<FCTOW>	-	-	GPS TOW of the first EPO SET currently used
<LCWN>	-	-	GPS week number of the last EPO SET currently used
<LCTOW>	-	-	GPS TOW of the last EPO SET currently used

Result:

None.

Example:

```
$PMTK707,56,1468,172800,1470,151200,1468,259200,1468,259200*1F
```

4.6. PMTK713 PMTK_DT_LOC

Sends Reference Position to the GNSS receiver. To have a quick TTFF, the Reference Position shall be accurate within 30 km.

Type:

Input.

Synopsis:

```
$PMTK713,<Lat>,<Lon>,<Alt>,<Unc_SMaj>,<Unc_SMin>,<Maj_Bear>,<Unc_Vert><Conf>*<Checksum>
```

Parameter:

Field	Unit	Range	Description
<Lat>	Degree	-90.0–90.0	WGS84 geodetic latitude. It is recommended to express this value in floating-point with 6 decimal points.
<Lon>	Degree	-180.0–180.0	WGS84 geodetic longitude. It is recommended to express this value in floating-point with 6 decimal points.
<Alt>	m	-	WGS84 ellipsoidal altitude.
<Unc_SMaj>	m	> 0	Standard deviation of semi-major axis of error ellipse.
<Unc_SMin>	m	> 0	Standard deviation of semi-minor axis of error ellipse.
<Maj_Bear>	Degree	0–179	Orientation of semi-major axis of error ellipse .
<Unc_Vert>	m	> 0	Vertical uncertainty.
<Conf>	%	0–100	The confidence by which the position of a target entity is known to be within the shape description.

Result:

Returns a **PMTK001** message.

Example:

```
$PMTK713,24.772816,121.022636,160,333,333,6,50,67*08
$PMTK001,713,3,24.772816,121.022636,160.000000,333.000000,333.000000,6,50.000000,67*0A
```

NOTE

We use Ellipsoid Point with altitude and uncertainty ellipsoid to describe poosition error shape.

4.7. PMTK721 PMTK_DT_SV_EPO

Sends GPS/GLONASS EPO data for a single satellite to GNSS receiver.

GPS satellites are identified by their PRN number, which ranges from 1 to 32.

GLONASS satellites are identified by their slot number which ranges from 1 to 24 for the full constellation of 24 satellites. <SatID> for GLONASS is assigned by slot number plus 64, which ranges from 65 to 88.

Type:

Input.

Synopsis:

```
$PMTK721,<SatID>,<W[0]>,...,<W[17]>*<Checksum><CR><LF>
```

Parameter:

Field	Unit	Range	Description
<SatID>	-	GPS: 1–32 GLONASS:65–88	Satellite PRN number [represented in HEX characters] for the EPO data to follow.
<W[0]–W[17]>	-	-	18 words [LSB first] of one EPO segment data (total 72 bytes).

Result:

Returns a **PMTK001** message.

Example:

```
$PMTK721,4,40568B0,D6C83E84,70E8E10,FA414370,F913650C,70E71F8,F8D266CF,8732D6FD,7F1
D9CE,E59383DF,76BFB93,A60319DF,C36C3289,20173F8D,959E4561,1000001C,40000,8B94CFB6*2
C
$PMTK001,721,17,00000000*32
```

4.8. PMTK740 PMTK_DT_UTC

Sends current Reference UTC time to GNSS receiver. Local time should be avoided due to time-zone offset. To have a faster TTFF, the Reference Time should be accurate with 3 s and must be specified in UTC time.

Type:

Input.

Synopsis:

```
$PMTK740,<Year>,<Month>,<Day>,<Hour>,<Minute>,<Second>*<Checksum><CR><LF>
```

Parameter:

Field	Unit	Range	Description
<Year>	Year	> 1980	UTC time: year in 4 digits.
<Month>	Month	1–12	UTC time: month.
<Day>	Day	1–31	UTC time: day.
<Hour>	Hour	0–23	UTC time: hour.
<Minute>	Minute	0–59	UTC time: minute.
<Second>	Second	0–59	UTC time: second.

Result:

Returns a **PMTK001** message.

Example:

```
$PMTK740,2010,2,10,9,0,58*05
$PMTK001,740,3,2010,2,10,9,0,58*07
```

4.9. PMTK741 PMTK_DT_POS

Injects Reference Position and Reference UTC time into GNSS device to have a faster TTFF.

Type:

Input.

Synopsis:

```
$PMTK741,<Lat>,<Lon>,<Alt>,<Year>,<Month>,<Day>,<Hour>,<Minute>,<Second>*<Checksum><CR><LF>
```

Parameter:

Field	Unit	Range	Description
<Lat>	Degree	-90.0–0.0	WGS84 geodetic latitude. It is recommended to express this value in floating-point with 6 decimal points. Minus: south; Plus: north.
<Lon>	Degree	-180.0–180.0	WGS84 geodetic longitude. It's recommended to express this value in floating-point with 6 decimal points. Minus: west; Plus: east.
<Alt>	m	-	WGS84 ellipsoidal altitude.
<Year>	Year	> 1980	UTC time: year in 4 digits.
<Month>	Month	01–12	UTC time: month.
<Day>	Day	01–31	UTC time: day.
<Hour>	Hour	00–23	UTC time: hour.
<Minute>	Minute	00–59	UTC time: minute.
<Second>	Second	00–59	UTC time: second.

Result:

Returns a **PMTK001** message.

Example:

```
$PMTK741,24.772816,121.022636,160,2016,01,01,12,00,00*17
$PMTK001,741,3,24.772816,121.022636,160.000000,2016,1,1,12,0,0*3B
```

5 EPO Usage Through QGNSS

QGNSS is a Quectel official tool which allows users to evaluate the receiver performance as well as to perform different measurements on the receiver. Visit http://220.180.239.212:8177/ for details on QGNSS. This chapter describes how to evaluate the EPO functionality through QGNSS.

5.1. Testing Flash EPO with QGNSS

Steps to test Flash EPO with the QGNSS tool:

1. Run the QGNSS tool.

2. In the main interface, click "**AGNSS**" → "**Assistant GNSS Offline**" as shown below.

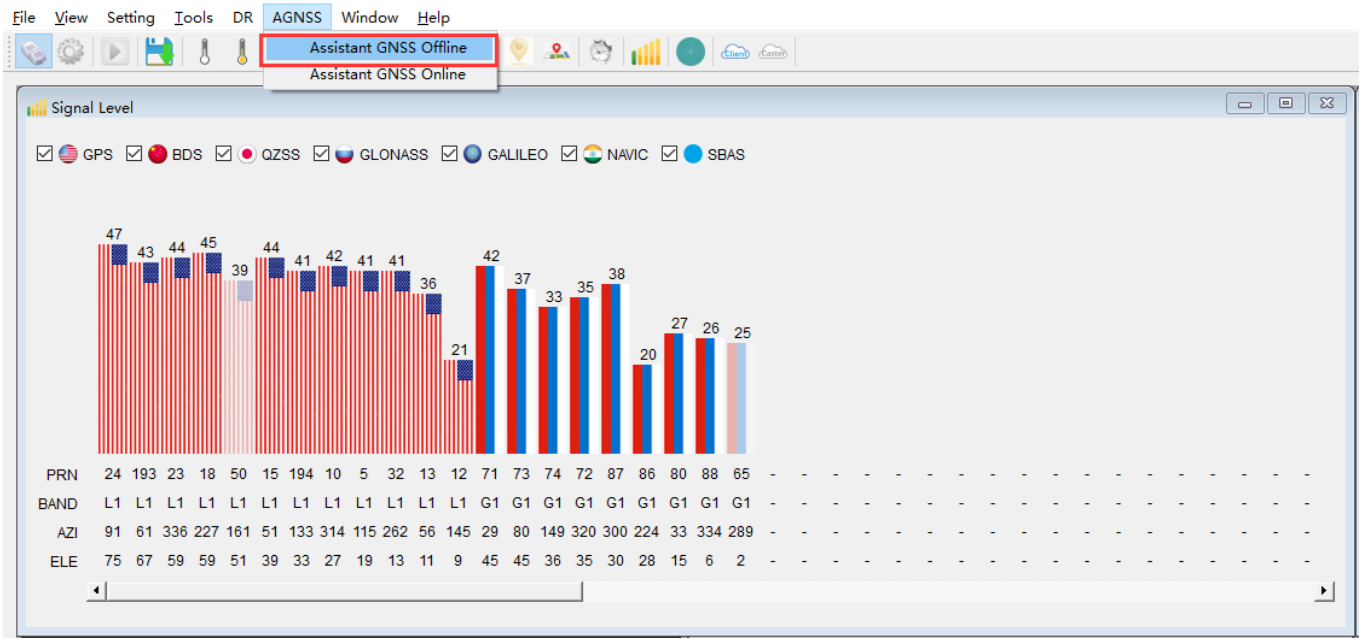

Figure 8: AGNSS Setting Interface of QGNSS

3. Download EPO file to the module.
   a) Click the "**Connect**" button to connect to the FTP server.
   b) Select EPO file.
   c) Click the "**Download selected file**" button to download the EPO file to computer.
   d) Click the "**Download to module**" to download the EPO file to module.

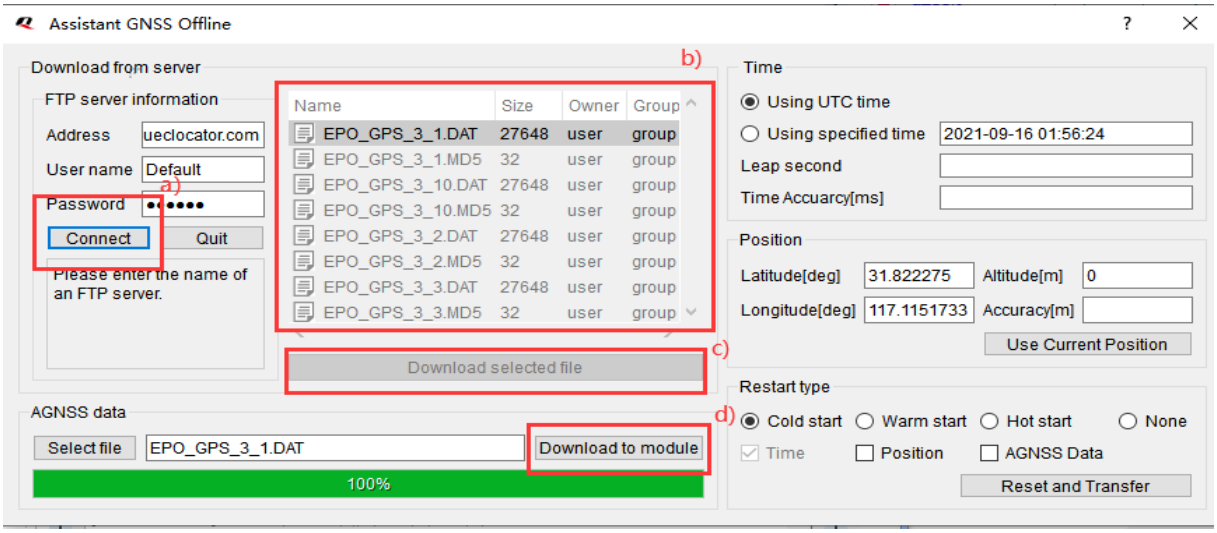

Figure 9: EPO File Downloading

4. Click "**Tools**" → "**Static TTFF Testing**" to enter TTFF testing window.

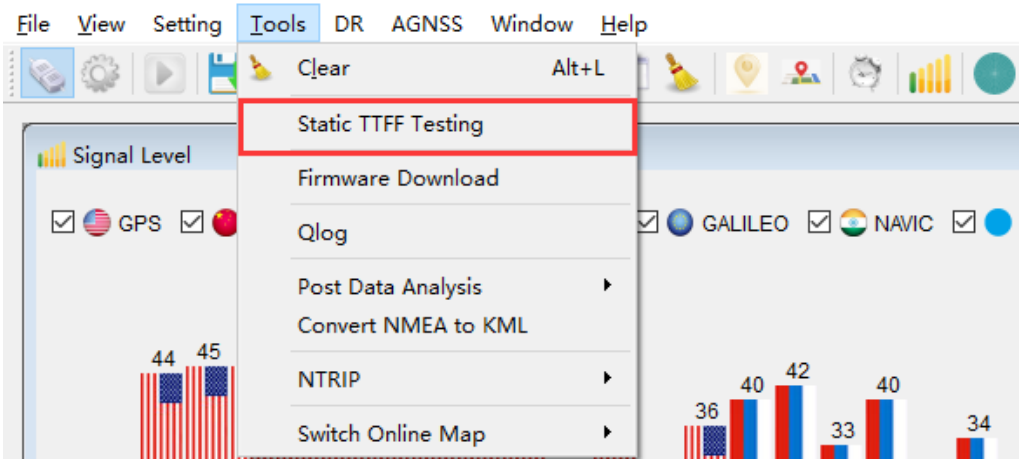

Figure 10: Static TTFF Testing

5. Check "**Use AGNSS**" and click "**Run**" to run the TTFF testing.

Figure 11: TTFF Setting

5.2. Testing Host EPO with QGNSS

The steps to test Host EPO with the QGNSS tool:

1. Run QGNSS tool.

2. In the main interface, click "**Tools**" **->** "**Static TTFF Testing**" to enter the TTFF testing window.

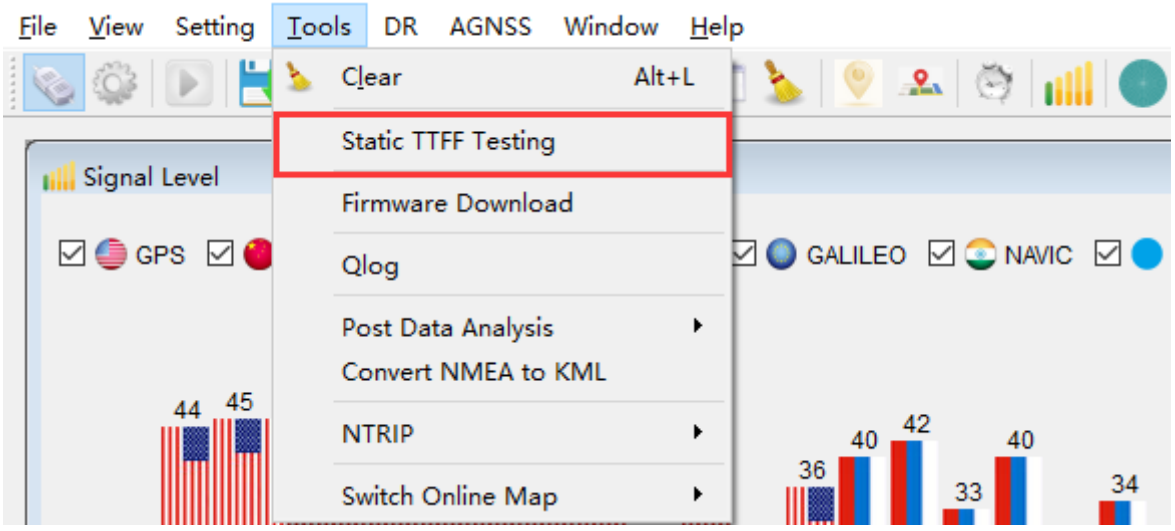

Figure 12: Static TTFF Testing

3. Configure parameter:
   a) Click "**Set AGNSS**" and check "**Use AGNSS**" to enter the Assistant GNSS Online window**.**
   b) Click "**Connect**" button to connect to FTP server.
   c) Select EPO file and click **"Download selected file**" button to download the EPO file to computer.
   d) Check **"Using UTC time**", "**Cold start**", "**Position**" and "**AGNSS Data**".
   e) Enter the latitude and longitude or click "**Use Current Position**" to automatically select the last fixed position.
   f) Click "**Run**" to test TTFF.

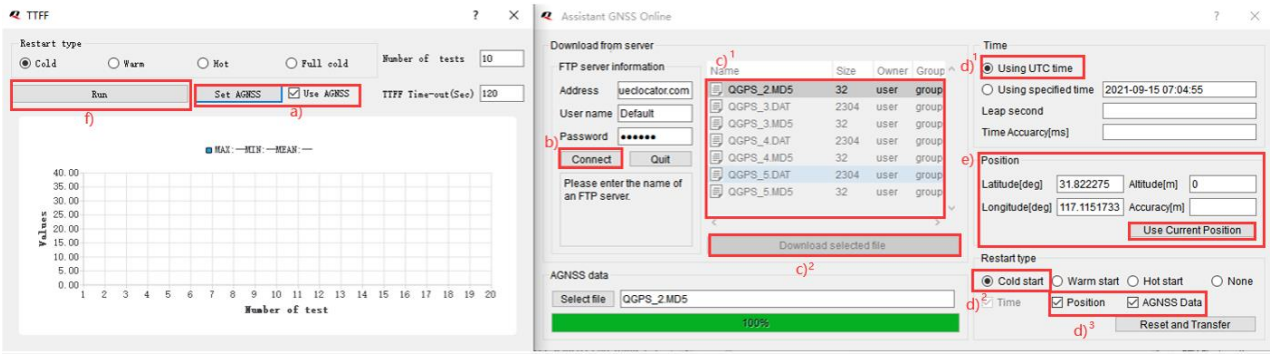

Figure 13: TTFF Setting

NOTE

1. If the Flash EPO has been tested first, use $PMTK127*36 to clear the Flash EPO data before you perform the Host EPO testing. A Host EPO file can be selected according to UTC time:
   0:00 to 6:00: EPO file 1
   6:00 to 12:00: EPO file 2
   12:00 to 18:00: EPO file 3
   18:00 to 24:00: EPO file 4

6 AGNSS Implementation Example

This chapter gives examples to show how EPO files can be downloaded to the module.

6.1. Flash EPO Implementation

Blue: Send data
Red: ACK information

//Host sends **$PMTK253,1,0*37** for switching from NMEA mode to binary mode, NO ACK will be sent from module:
24 50 4D 54 4B 32 35 33 2C 31 2C 30 2A 33 37 0D 0A

//Host sends EPO file:
04 24 E3 00 D3 02 00 00 DC 93 05 01 C6 03 F8 9F 59 DE 0A 73 39 0F 54 00 23 25 FD 02 F4 21 0A 73 ……0D 0A

//Module will erase the area where the EPO saved.
24 43 4C 52 2C 45 50 4F 2C 30 30 30 61 38 30 30 30 2A 35 45 0D 0A
24 43 4C 52 2C 45 50 4F 2C 30 30 30 61 39 0D 0A
24 43 4C 52 2C 45 50 4F 2C 30 30 30 61 39 30 30 30 2A 35 46 0D 0A
24 43 4C 52 2C 45 50 4F 2C 30 30 30 61 61 30 30 30 2A 30 37 0D 0A
24 43 4C 52 2C 45 50 4F 2C 30 30 30 61 62 30 30 30 2A 30 34 0D 0A
24 43 4C 52 2C 45 50 4F 2C 30 30 30 61 63 30 30 30 2A 30 35 0D 0A
……

//Module returns a **PMTK001** message:
04 24 0C 00 20 00 00 00 01 0F 0D 0A

//Host sends EPO data:
04 24 E3 00 D3 02 01 00 DC 93 05 04 B2 02 F8 01 C7 71 0A 73 A6 5F 49 FC A9 68 D9 FC C9 71 0A 73 ……. 0D 0A

//Module returns a **PMTK001** message:
04 24 0C 00 02 00 01 00 01 0E 0D 0A

//Host sends EPO data:
04 24 E3 00 D3 02 02 00 DC 93 05 07 1B 03 F8 3E F8 41 0A 73 2B 74 62 03 A8 4D 5F 04 C9 41 0A 73

……0D 0A

//Module returns a **PMTK001** message:
04 24 0C 00 02 00 02 00 01 0D 0D 0A

//Host send EPO data:
04 24 E3 00 D3 02 03 00 DC 93 05 ……. 0D 0A

//Module returns a **PMTK001** message:
04 24 0C 00 02 00 03 00 01 0C OD 0A
……

//Host send EPO data:
04 24 E3 00 D3 02 6A 00 DC 93 05 ……. 0D 0A

//Module returns a **PMTK001** message:
04 24 0C 00 02 00 6A 00 01 65 0D 0A

//End of host sending, no data any more:
04 24 E3 00 D3 02 FF FF 00 00 00 00 00 00 00 00……0D 0A

//End of module reception:
04 24 0C 00 02 00 FF FF 01 0F 0D 0A

//Host changes UART to PMTK protocol and use default baudrate:
04 24 0E 00 FD 00 00 00 00 00 00 F3 0D 0A

6.2. Host EPO Implementation

Blue: Send data
Red: ACK information

//Host Sends cold start command: **$PMTK103*30**:
24 50 4D 54 4B 31 30 33 2A 33 30 0D 0A

//Host sends the current UTC time command: **$PMTK740,2021,09,18,07,34,14*35**:
24 50 4D 54 4B 37 34 30 2C 32 30 32 31 2C 30 39 2C 31 38 2C 30 37 2C 33 34 2C 31 34 2A 33 35 0D 0A

//Module returns a **PMTK001** message: **$PMTK001,740,2,2021,9,18,7,34,14*36**:
24 50 4D 54 4B 30 30 31 2C 37 34 30 2C 32 2C 32 30 32 31 2C 39 2C 31 38 2C 37 2C 33 34 2C 31 34 2A 33 36 0D 0A

//Host sends the current location information command:

$PMTK741,31.822218,117.115247,49.200000,2021,09,18,07,34,14*0B
24 50 4D 54 4B 37 34 31 2C 33 31 2E 38 32 32 32 31 38 2C 31 31 37 2E 31 31 35 32 34 37 2C 34 39 2E 32 30 30 30 30 30 2C 32 30 32 31 2C 30 39 2C 31 38 2C 30 37 2C 33 34 2C 31 34 2A 30 42 0D 0A

//Module returns a **PMTK001** message:
$PMTK001,741,3,31.822218,117.115247,49.200000,2021,9,18,7,34,14*09
24 50 4D 54 4B 30 30 31 2C 37 34 31 2C 33 2C 33 31 2E 38 32 32 32 31 38 2C 31 31 37 2E 31 31 35 32 34 37 2C 34 39 2E 32 30 30 30 30 30 2C 32 30 32 31 2C 39 2C 31 38 2C 37 2C 33 34 2C 31 34 2A 30 39 0D 0A

//Host sends EPO data:
$PMTK721,1,10593EE,9FFC01C0,84D8DE59,E250EA6,F6326A6,84D8DE3E,75D098A,855C5F41,6F18520,CAD37EAE,2A797EC,A60346B9,25302423,2F28853F,24458AEA,1000001C,0000000,4B65EDB3*19
24 50 4D 54 4B 37 32 31 2C 31 2C 31 30 35 39 33 45 45 2C 39 46 46 43 30 31 43 30 2C 38 34 44 38 44 45 35 39 2C 45 32 35 30 45 41 36 2C 46 36 33 32 36 41 36 2C 38 34 44 38 44 45 33 45 2C 37 35 44 30 39 38 41 2C 38 35 35 43 35 46 34 31 2C 36 46 31 38 35 32 30 2C 43 41 44 33 37 45 41 45 2C 32 41 37 39 37 45 43 2C 41 36 30 33 34 36 42 39 2C 32 35 33 30 32 34 32 33 2C 32 46 32 38 38 35 33 46 2C 32 34 34 35 38 41 45 41 2C 31 30 30 30 30 30 31 43 2C 30 30 30 30 30 30 30 2C 34 42 36 35 45 44 42 33 2A 31 39 0D 0A

//Module returns a **PMTK001** message: **$PMTK001,721,3, 1,00000000*25**:
24 50 4D 54 4B 30 30 31 2C 37 32 31 2C 33 2C 20 31 2C 30 30 30 30 30 30 30 30 2A 32 35 0D 0A

//Host sends EPO data:
$PMTK721,2,20593EE,EEFC02B5,84D8EE28,DF12534,E55186E,84D8EEFF,F8B63471,83655362,6F18E3E,DFA23AA6,D6EB7E5,A6001FC7,19B59076,20492A75,C32F2503,1000001C,0000000,C604863*1C

24 50 4D 54 4B 37 32 31 2C 32 2C 32 30 35 39 33 45 45 2C 45 45 46 43 30 32 42 35 2C 38 34 44 38 45 45 32 38 2C 44 46 31 32 35 33 34 2C 45 35 35 31 38 36 45 2C 38 34 44 38 45 45 46 46 2C 46 38 42 36 33 34 37 31 2C 38 33 36 35 35 33 36 32 2C 36 46 31 38 45 33 45 2C 44 46 41 32 33 41 41 36 2C 44 36 45 42 37 45 35 2C 41 36 30 30 31 46 43 37 2C 31 39 42 35 39 30 37 36 2C 32 30 34 39 32 41 37 35 2C 43 33 32 46 32 35 30 33 2C 31 30 30 30 30 30 31 43 2C 30 30 30 30 30 30 30 2C 43 36 30 34 38 36 33 2A 31 43 0D 0A

//Module returns a **PMTK001** message: **$PMTK001,721,3, 2,00000000*26**:
24 50 4D 54 4B 30 30 31 2C 37 32 31 2C 33 2C 20 32 2C 30 30 30 30 30 30 30 30 2A 32 36 0D 0A
……
……

//Host sends EPO data:
$PMTK721,20,200593EE,E2FC0257,84DACE24,FE391CC7,FD6A2881,84DA31D2,F8FB23C7,82F3C284,6F39D87,4190A41C,590822F,A601FD64,71D66A59,2004BD1E,9F31A748,100

D1E,9F31A748,1000001C,1000000,80B38D89*53

35 41 35 37 32 42 41 2A 31 36 0D 0A 24 50 4D 54 4B 37 32 31 2C 32 30 2C 32 30 30 35 39 33 45 45 2C 45 32 46 43 30 32 35 37 2C 38 34 44 41 43 45 32 34 2C 46 45 33 39 31 43 43 37 2C 46 44 36 41 32 38 38 31 2C 38 34 44 41 33 31 44 32 2C 46 38 46 42 32 33 43 37 2C 38 32 46 33 43 32 38 34 2C 36 46 33 39 44 38 37 2C 34 31 39 30 41 34 31 43 2C 35 39 30 38 32 32 46 2C 41 36 30 31 46 44 36 34 2C 37 31 44 36 36 41 35 39 2C 32 30 30 34 42 44 31 45 2C 39 46 33 31 41 37 34 38 2C 31 30 30 30 30 30 31 43 2C 31 30 30 30 30 30 30 2C 38 30 42 33 38 44 38 39 2A 35 33 0D 0A

//Module returns a **PMTK001** message: **$PMTK001,721,3, 32,00000000*35**:
0A 24 50 4D 54 4B 30 30 31 2C 37 32 31 2C 33 2C 33 32 2C 30 30 30 30 30 30 30 30 2A 33 35 0D 0A

7 Appendix References

Table 18: Almanac, Ephemeris & EPO

Type	Source	Valid	Effect
Almanac	Satellites	Several weeks	Satellites searching
Ephemeris	Satellites	< 4 hours	Positioning
EPO	Chipset supplier’s server	6 hours to 30 days	Satellites searching & positioning

Table 19: Terms and Abbreviations

Abbreviation	Description
EPO	Extended Prediction Orbit
FTP	File Transfer Protocol
GLONASS	Global Navigation Satellite System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
TTFF	Time to First Fix
TOW	Time of week
UART	Universal Asynchronous Receiver/Transmitter
UTC	Coordinated Universal Time
URL	Uniform Resource Locator
ACK	Acknowledgement
SV	Satellite Vehicle