

LC26G&LC26G-T&LC76G&LC86G Series

AGNSS Application Note

GNSS Module Series

Version: 1.1

Date: 2024-12-25

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.

Copyright © Quectel Wireless Solutions Co., Ltd. 2024. All rights reserved.

About the Document

Document Information

Title	LC26G&LC26G-T&LC76G&LC86G Series AGNSS Application Note
Subtitle	GNSS Module Series
Document Type	Application Note
Document Status	Released

Revision History

Revision	Date	Description
-	2022-07-11	Creation of the document
1.0	2022-10-10	First official release
1.1	2024-12-25	- Added applicable module LC26G-T (AA). - Added the AGNSS implementation methods EASY and EPOC: - Introduction (Chapter 1); - AGNSS with EASY (Chapter 2.4); - AGNSS with EPOC (Chapter 2.5); - EASY related messages (Chapter 3.2); - EPOC related messages (Chapter 3.3); - EASY & EPOC related messages (Chapter 3.4); - EASY implementation (Chapter 5.3); - EPOC implementation (Chapter 5.4). - Added a note on the requirement for using the same "device_id" (Chapter 2.1.1). - Added a note on purposes of different types of EPO files (Chapter 2.1.3). - Added the information on acknowledge EPO binary format and the note on EPO_GR_3_X.DAT file transmission

sequence ([Chapter 2.2.1](#)).

6. Added EPO related message \$PAIR010 ([Chapter 3.1.1](#)).
 7. Updated the PRN code range for Galileo and BDS, and added notes on the execution requirement and result of \$PAIR471 ([Chapter 3.1.3](#)).
 8. Added the requirement on reference position accuracy ([Chapter 3.1.6](#)).
-

Contents

About the Document	3
Contents	5
Table Index	7
Figure Index	8
1 Introduction	9
1.1. Network-based AGNSS	10
1.2. Self-based AGNSS	11
1.2.1. EASY	11
1.2.2. EPOC	11
1.3. Differences Between AGNSS Methods	11
1.4. AGNSS Related Messages	12
2 AGNSS Implementation	14
2.1. Download EPO Files	14
2.1.1. Get EPO Files from Server	14
2.1.2. EPO File Format	15
2.1.2.1. EPO File Format – GPS Only	16
2.1.2.2. EPO File Format – BDS/Galileo Only	17
2.1.2.3. EPO File Format – GPS + GLONASS	19
2.1.3. EPO File Types	20
2.1.4. Recommended Download Procedures of EPO Files	21
2.1.5. EPO File Validity Period	21
2.2. AGNSS with Flash EPO	22
2.2.1. Binary Protocol	23
2.2.2. EPO Data Transfer Protocol	25
2.2.3. AGNSS Procedure with Flash EPO	29
2.3. AGNSS with Host EPO	30
2.3.1. Recommended Sequence for Host EPO	30
2.3.2. Sample Code to Send EPO	31
2.4. AGNSS with EASY	33
2.5. AGNSS with EPOC	35
3 AGNSS Related Messages	37
3.1. EPO Related Messages	37
3.1.1. PAIR010: PAIR_REQUEST_AIDING	37
3.1.2. PAIR470: PAIR_EPO_GET_STATUS	38
3.1.3. PAIR471: PAIR_EPO_SET_DATA	39
3.1.4. PAIR472: PAIR_EPO_ERASE_FLASH_DATA	40
3.1.5. PAIR590: PAIR_TIME_SET_REF_UTC	41
3.1.6. PAIR600: PAIR_LOC_SET_REF	42
3.2. EASY Related Messages	43
3.2.1. PAIR490: PAIR_EASY_ENABLE	43

3.2.2.	PAIR491: PAIR_EASY_GET_STATUS	44
3.3.	EPOC Related Messages	45
3.3.1.	PAIR496: PAIR_EPOC_ENABLE.....	45
3.3.2.	PAIR498: PAIR_EPOC_SET_CONFIG	46
3.3.3.	PAIR507: PAIR_EPOC_CLEAR_DATA	46
3.3.4.	PAIR508: PAIR_EPOC_GET_STATUS	47
3.3.5.	PAIR509: PAIR_EPOC_GET_PREDICTION_STATUS	48
3.4.	EASY & EPOC Related Messages	49
3.4.1.	PAIR511: PAIR_NVRAM_SAVE_NAVIGATION_DATA	49
3.4.2.	PAIR512: PAIR_NVRAM_CLEAR_NAVIGATION_DATA.....	50
3.4.3.	PAIR513: PAIR_NVRAM_SAVE_SETTING	51
4	EPO Implementation with QGNSS	53
4.1.	Flash EPO Implementation with QGNSS	53
4.2.	Host EPO Implementation with QGNSS	54
5	AGNSS Implementation Example	56
5.1.	Flash EPO Implementation	56
5.2.	Host EPO Implementation.....	57
5.3.	EASY Implementation	58
5.4.	EPOC Implementation	59
6	Appendix A References	61
7	Appendix B Special Characters	63

Table Index

Table 1: Applicable Modules	9
Table 2: Differences Between AGNSS Methods	12
Table 3: AGNSS Related Messages	12
Table 4: Download URL of EPO Files	14
Table 5: EPO Data SVID Range	16
Table 6: EPO File Types	20
Table 7: Description of Binary Protocol Fields.....	23
Table 8: Start of EPO Binary Format.....	24
Table 9: EPO Data Binary Format.....	24
Table 10: End of EPO Binary Format	24
Table 11: Acknowledge EPO Binary Format	24
Table 12: Payload of Acknowledgement	24
Table 13: Related Document.....	61
Table 14: Terms and Abbreviations.....	61
Table 15: Special Characters	63

Figure Index

Figure 1: EPO File Format – GPS Only.....	16
Figure 2: Format of Several Segments of EPO Files.....	17
Figure 3: EPO File Format – BDS/Galileo Only.....	17
Figure 4: Galileo EPO Header.....	18
Figure 5: BDS EPO Header	19
Figure 6: EPO File Format – GPS + GLONASS.....	19
Figure 7: Recommended Download Procedures of EPO Files.....	21
Figure 8: Binary Protocol Structure	23
Figure 9: AGNSS Procedure with Flash EPO	29
Figure 10: Suggested Sequence for Host EPO	31
Figure 11: Mechanism of EASY Function.....	34
Figure 12: Recommended Process for Using EASY	34
Figure 13: Mechanism of EPOC Function	35
Figure 14: Recommended Process for Using EPOC.....	36
Figure 15: QGNSS Interface for Setting Flash EPO.....	53
Figure 16: Download Flash EPO File	54
Figure 17: QGNSS Interface for Setting Host EPO	55
Figure 18: Download Host EPO File.....	55

1 Introduction

Assisted GNSS (AGNSS) is an enhanced GNSS technology that addresses the limitations of standard GNSS receivers. Typically, GNSS receivers must wait for slow satellite transmissions to receive navigation messages, which can result in lengthy TTFF duration of tens of seconds after power on (cold start). This delay is exacerbated in areas with weak signals. The AGNSS technology mitigates this issue by enabling receivers to receive assistance data through auxiliary ways, such as a network connection. The assistance data generally refers to ephemeris, almanac, accurate time and approximate position. With this information, the receivers can calculate a position within a few seconds, even under weak signal conditions.

This document outlines two types of AGNSS technologies supported by the module: Network-based and Self-based AGNSS. The Network-based AGNSS, comprising Host EPO and Flash EPO methods, necessitates your device's capability to access the network and retrieve EPO (Extended Prediction Orbit) data from an AGNSS server. The Self-based AGNSS, consisting of EASY (Embedded Assist System) and EPOC (Enhanced Prediction Orbit on Chip) methods, is an embedded function within the module which works without network connectivity.

This document mainly introduces the implementation of AGNSS feature, AGNSS related messages, EPO implementation with QGNSS, and AGNSS implementation example.

Table 1: Applicable Modules

Module Series	Model
LC26G	LC26G (AB)
LC26G-T	LC26G-T (AA)
LC76G	LC76G (AB)
	LC76G (PA)
	LC76G (PB)
LC86G	LC86G (AA)
	LC86G (AB)
	LC86G (LA)
	LC86G (PA)

NOTE

EASY and EPOC cannot operate simultaneously within a single firmware. If you require details on which firmware versions support the EPOC function, contact Quectel Technical Support (support@quectel.com).

1.1. Network-based AGNSS

EPO (Extended Prediction Orbit) is an AGNSS feature implemented by the chipset supplier, which should improve the TTFF duration of the GNSS receivers. Be aware that AGNSS cannot improve the TTFF duration in cases where the signal strength is too low (due to deficiencies in RF design or poor signal reception quality).

The Network-based AGNSS requires the host to access the network, download the EPO file from AGNSS server, and inject the EPO data into module. There are two ways to inject EPO files into the module: Host EPO and Flash EPO.

Host EPO (also called real-time AGNSS) allows the host to download EPO files into its own memory (some storage space needs to be allocated to save EPO files, and the size depends on the number of EPO files you want), and then the host reads out one EPO data set and composes a NMEA PAIR message and sends it to the module. For details, refer to [Chapter 2 AGNSS Implementation](#). For Host EPO, the module does not save the EPO data to its NVM, which requires the host to inject EPO data every time it restarts.

With Flash EPO, the host downloads EPO files from the AGNSS server and injects them into the module's flash (NVM) using a Binary Protocol, allowing the module to store up to 14 days of EPO data in the flash. In this way, the host does not need to inject the EPO data into the module upon each reboot. Instead, the module reads the EPO data from its flash and uses it to compute the position if the data is valid. The host must check whether the EPO data in the module's flash has expired or not. If it has, the host should download an updated EPO file from the AGNSS server.

Requirements to implement AGNSS:

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

- The **Reference Time** should be accurate within 3 seconds and must be specified in UTC time.
- The **Reference Position** should be accurate within 30 km from the receiver's actual position. 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 [Table 3: AGNSS Related Messages](#). See [Chapter 3.1 EPO Related Messages](#) for a detailed description of these messages.

1.2. Self-based AGNSS

1.2.1. EASY

EASY is a solution designed to enhance the TTFF performance. It is an internal module application that predicts satellite orbits based on the received ephemeris data decoded from the ephemeris data broadcast by the satellites themselves. This way, the module does not need external network connectivity. Upon receiving the updated broadcast ephemeris data, the EASY function is automatically activated to predict satellite orbits for the following 3 days. EASY enters a standby state upon completion of the 3-day extension of satellite orbit prediction for all available SVs.

1.2.2. EPOC

EPOC is another internal module application designed to improve the TTFF performance by predicting GNSS constellation orbits using the received broadcast ephemeris data. EPOC aiding data serves as an alternative AGNSS method aimed at speeding up TTFF when the loss of EPO aiding data is caused by unavailability of external network connectivity.

On Day 1, TTFF is approximately 30 seconds without EPOC aiding data. Once the broadcast ephemerides are received, EPOC automatically will activate the 3-day satellite orbit prediction process. Over the subsequent 72 hours, EPOC accelerates TTFF and ensures precise positioning. After completing the orbit prediction process for all available broadcast ephemerides, EPOC transitions to standby state until new broadcast ephemeris data becomes available.

1.3. Differences Between AGNSS Methods

EASY, EPO and EPOC can shorten the time to first fix. The differences between them are shown below.

Table 2: Differences Between AGNSS Methods

Item	EASY	EPOC	EPO	
			Flash EPO	Host EPO
Storage Space	RTC RAM/Flash	RTC RAM/Flash	Flash	RAM
Storage Capacity	1, 2 or 3 days' assistance data	3 days' assistance data	Up to 14 days' assistance data	6-hour assistance data
Protocol	-	-	Binary	NMEA (ASCII)
Supported Satellites Constellations	GPS Only	- GPS only - GPS + Galileo - GPS + BDS (MEO and IGSO)	- GPS only - GPS + GLONASS - BDS only - Galileo only	- GPS only - GPS + GLONASS - BDS only - Galileo only
Supported Fix Rate	1 Hz	1 Hz	1 Hz 2 Hz 5 Hz 10 Hz	1 Hz 2 Hz 5 Hz 10 Hz

NOTE

- The maximum flash memory EPO data retention period is 14 days for GPS-only and GPS + GLONASS EPO files, 7 days for Galileo EPO files and 3 days for BDS EPO files. If 30-day GPS-only or GPS + GLONASS EPO files are sent, only the first 14 days of EPO data will be stored.
- The data generated by the EASY or EPOC feature is saved in the RTC RAM, and the data in the RTC RAM is synchronized to flash only after **\$PAIR511** is executed.

1.4. AGNSS Related Messages

AGNSS related messages are listed below.

Table 3: AGNSS Related Messages

Message	Description
\$PAIR010	Notifies the expiration of GNSS assistance data stored in the module.
\$PAIR470	Queries the status of EPO data stored in the module.
\$PAIR471	Sends the packet containing EPO data for GPS/GLONASS/Galileo/BDS satellite.

Message	Description
\$PAIR472	Erases the EPO data stored in the flash memory.
\$PAIR490	Enables/disables EASY function.
\$PAIR491	Gets the status of EASY function and prediction status (EASY data extension status).
\$PAIR496	Enables/disables EPOC function.
\$PAIR498	Sets EPOC constellation configuration.
\$PAIR507	Clears all EPOC data from NVM.
\$PAIR508	Gets the status of EPOC function and constellation configuration.
\$PAIR509	Gets EPOC prediction status.
\$PAIR511	Saves current navigation data from RTC RAM to NVM.
\$PAIR512	Clears navigation data in both RTC RAM and NVM.
\$PAIR513	Saves current configuration from RTC RAM to NVM.
\$PAIR590	Sends reference UTC time to the GNSS receiver.
\$PAIR600	Sends reference position to the GNSS receiver.

2 AGNSS Implementation

This chapter describes the four AGNSS implementation methods: Host EPO, Flash EPO, EASY and EPOC.

- Implement AGNSS with Host EPO
The host sends EPO data to the GNSS receiver through NMEA (ASCII) PAIR commands, such as **\$PAIR471**.
- Implement AGNSS with Flash EPO
The EPO data are downloaded to the flash of GNSS receiver through Binary Protocol.

Flash EPO retains data longer than Host EPO.
- Implement AGNSS with EASY or EPOC
The EASY or EPOC predicts and generates ephemeris information based on the currently received satellite information. The data predicted by EASY or EPOC is less effective at reducing positioning time compared to the data in EPO files.

2.1. Download EPO Files

Quectel does not provide any Service Level Agreement for EPO files. Download EPO data to your own server and send them to devices to ensure EPO data availability.

2.1.1. 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
Unified QEPO	BDS only	http://wpepodownload.mediatek.com/QBD2.DAT?vendorinfo	Single name: QBD2.DAT
Unified QEPO	Galileo only	http://wpepodownload.mediatek.com/QGA.DAT?vendorinfo	Single name: QGA.DAT

EPO Type	GNSS Type	EPO File URL	File Name
EPO	GPS only	http://wpepdownload.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://wpepdownload.mediatek.com/EPO_GR_3_ X .DAT?vendorinfo	X = 1–10 EPO_GR_3_1.DAT to EPO_GR_3_10.DAT
EPO	BDS only	http://wpepdownload.mediatek.com/EPO_BDS_3.DAT?vendorinfo	EPO_BDS_3.DAT
EPO	Galileo only	http://wpepdownload.mediatek.com/EPO_GAL_ X .DAT?vendorinfo	X = 3 or 7 EPO_GAL_3.DAT or EPO_GAL_7.DAT

The following is a complete URL sample:

http://wpepdownload.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 and BBB in the example) are issued by Quectel. Contact Quectel Technical Support to get the value.
- The value of “device_id” (CCC in the example) contains two parts – one is assigned by Quectel and the other by the customer. For example: if CCC = XXX_YYY, the value XXX is provided by Quectel and you can contact Quectel Technical Support 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

1. There will be up to 10 files as the GPS-only or GPS + GLONASS EPO files may include a maximum of 30 days of predictions. Slices of 30-day EPO:
_1 for days 1 to 3,
_2 for days 4 to 6,
...
_10 for days 28 to 30.
2. The host using the same “device_id” should not access the AGNSS server more than 20 times a day, otherwise the access will fail.

2.1.2. EPO File Format

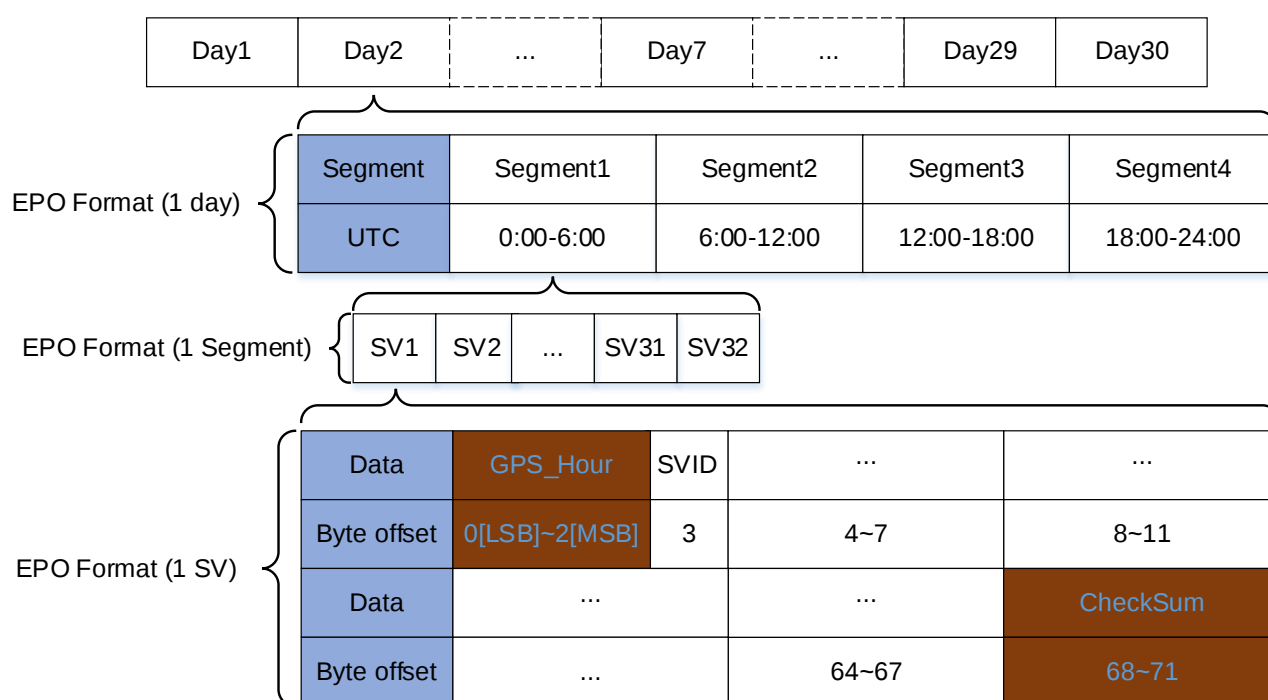
This chapter mainly illustrates the format of the EPO files.

The SVID numbers of EPO files for different constellations are shown below.

Table 5: EPO Data SVID Range

GNSS Type	PRN	EPO Data SVID
GPS	1–32	1–32
GLONASS	1–24	65–88
Galileo	1–36	101–136
BDS	1–54, 55–63	201–254, 190–198

2.1.2.1.EPO File Format – GPS Only


Figure 1: EPO File Format – GPS Only

$GPS_Secs = GPS_Hour \times 3600$

$GPS_Week\ Number = GPS_Secs / 604800$

$GPS\ TOW = GPS_Secs \% 604800$

An EPO file contains 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 of several segments of EPO files.

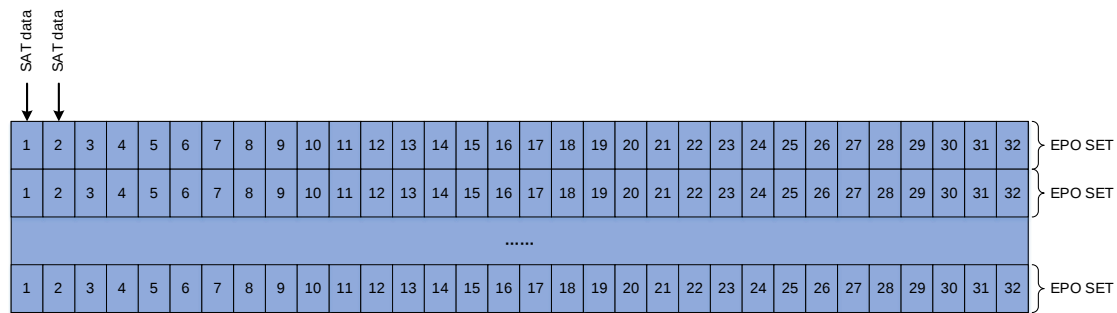


Figure 2: Format of Several Segments of EPO Files

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 and thus 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.

2.1.2.2.EPO File Format – BDS/Galileo Only

Galileo EPO data consist of the 72-byte header and 3- or 7- day fundamental EPO data. BDS EPO data consist of the 72-byte header and 3-day EPO data only. The EPO format for both has no fixed size.

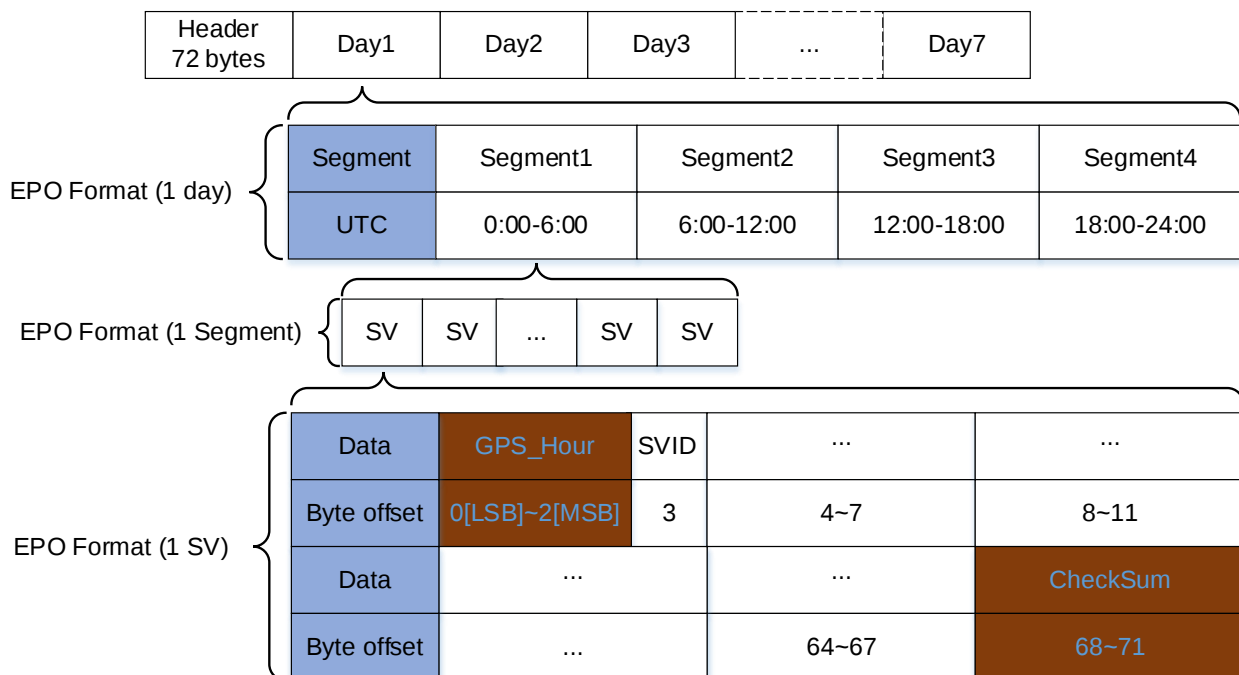


Figure 3: EPO File Format – BDS/Galileo Only

The 72-byte header contains the SV available bitmask that can be used for calculating the available satellite ID. When the SV available bitmask position is 1, it indicates that the satellite is available, and the number of the bit indicates the satellite ID.

For example, you can parse the data from [Figure 4: Galileo EPO Header](#) as follows.

- SYSID is FE for Galileo.
- SV available bitmask: 09 67 94 5D DF.
- Total available SVs: 22.
- Available SVs: 1, 2, 3, 4, 5, 7, 8, 9, 11, 12, 13, 15, 19, 21, 24, 25, 26, 27, 30, 31, 33, 36.

Data	***	SYSID	SV available bitmask(Lbytes)	***	***
Byte offset	0~2	3	4[LSB]~7[MSB]	8~11	12~15
Example		FE	DF 5D 94 67		

Data	SV available bitmask(Hbytes)	***	***	***
Byte offset	16[LSB]~19[MSB]	20~23	24~27	18~31
Example	09 00 00 00			

Figure 4: Galileo EPO Header

You can parse the data from [Figure 5: BDS EPO Header](#) as follows.

- SYSID is FF for BDS.
- SV available bitmask: 3F FF BF FC 3F FF.
- Total available SVs: 41.
- Available SVs: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46.

Data	***	SYSID	SV available bitmask(Lbytes)	***	***
Byte offset	0~2	3	4[LSB]~7[MSB]	8~11	12~15
Example		FF	FF 3F FC BF		

Data	SV available bitmask(Hbytes)	***	***	***
Byte offset	16[LSB]~19[MSB]	20~23	24~27	18~31
Example	FF 3F 00 00			

Figure 5: BDS EPO Header

2.1.2.3.EPO File Format – GPS + GLONASS

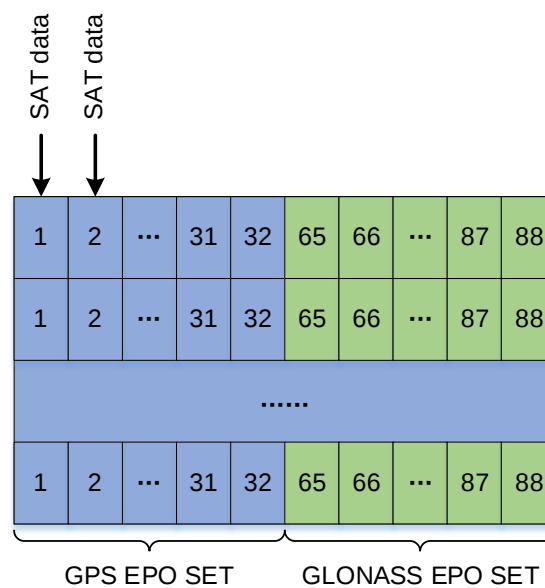


Figure 6: EPO File Format – GPS + GLONASS

The basic unit of an EPO file is SAT Data, and the size of SAT Data is 72 bytes. In GPS + GLONASS EPO files, one EPO SET contains 56 SAT Data, so the EPO SET data size 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.

2.1.3. EPO File Types

The EPO data can be downloaded in the form of files. You can select the most suitable file type 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 6: EPO File Types](#) to decide on the file type to be downloaded.

Table 6: EPO File Types

EPO Type	GNSS Type	Description
<i>Unified QEPO</i>	GPS only	6-hour prediction orbit (ephemeris). Single file containing the latest available GPS EPO data.
<i>Unified QEPO</i>	GPS + GLONASS	6-hour prediction orbit (ephemeris). Single file containing the latest available GPS + GLONASS EPO data.
<i>Unified QEPO</i>	BDS/Galileo only	6-hour prediction orbit (ephemeris). Single file containing the latest available BDS/Galileo EPO data.
<i>EPO</i>	GPS only	3–14 days of prediction orbit (ephemeris). Split into 5 files, each containing 3-day information.
<i>EPO</i>	Galileo only	3- or 7-day prediction orbit (ephemeris).
<i>EPO</i>	BDS only	3-day prediction orbit (ephemeris).
<i>EPO</i>	GPS + GLONASS	3–14 days of prediction orbit (ephemeris). Split in 5 files, each containing 3-day information.

NOTE

GPS/GLONASS EPO accelerates the TTFF in positioning, whereas BDS/Galileo EPO does not speed up TTFF but is used to shorten the positioning convergence time.

2.1.4. Recommended Download Procedures of EPO Files

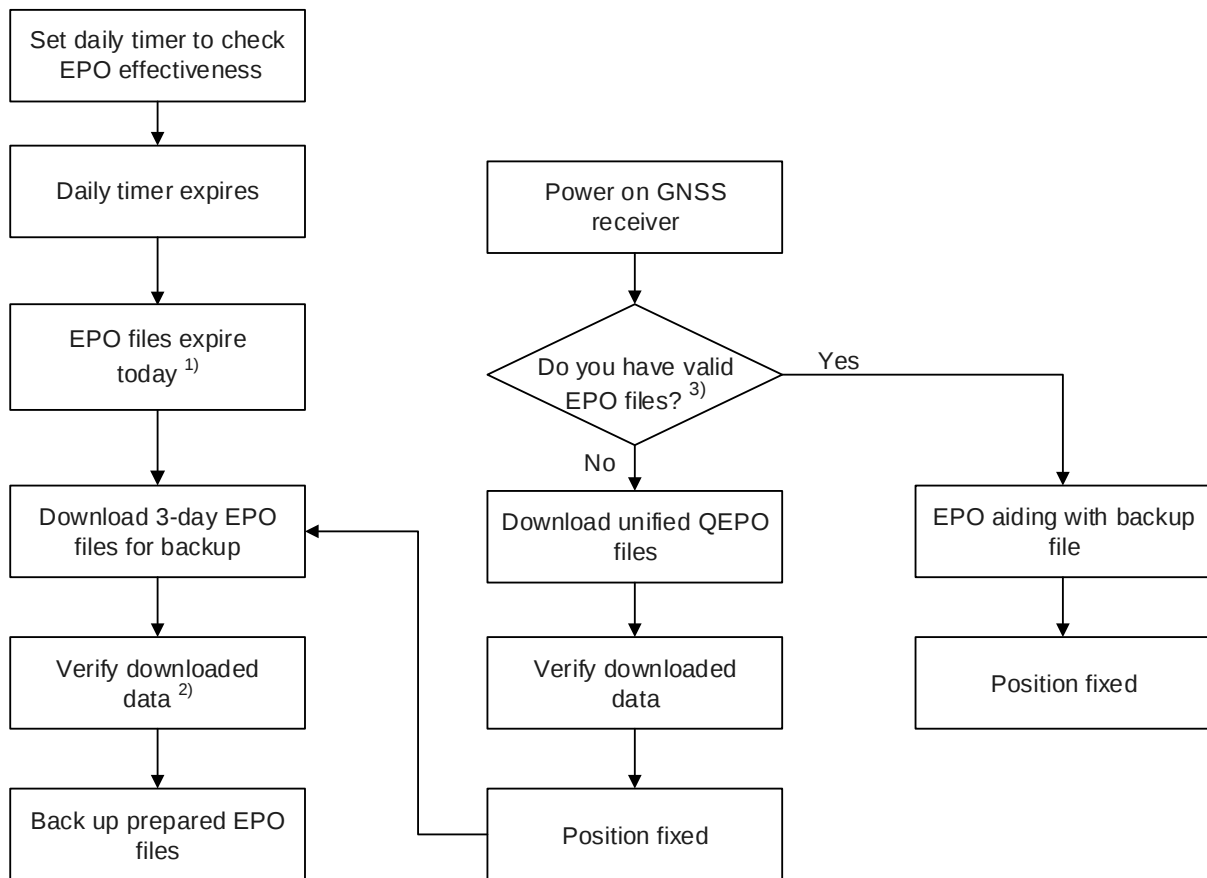


Figure 7: Recommended Download Procedures of EPO Files

NOTE

1. ¹⁾ You must know the current UTC time to download valid EPO files.
2. ²⁾ Send **\$PAIR470** to check if the data are correct.
3. ³⁾ If the device is powered off for a long time, EPO files stored in flash may expire.

2.1.5. EPO File Validity Period

EPO file validity period is related to the current UTC time. The EPO file validity period can be obtained from the last segment of the EPO file. See [Figure 1: EPO File Format – GPS Only](#) or the sample of how to calculate EPO file 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;
}

```

2.2. AGNSS with Flash EPO

Flash EPO can store up to 14 days of EPO assistance data in flash, which enables the receiver to use the available data since boot time. The communication protocol of Flash EPO is Binary Protocol. Thus, you

need to download assistance data to the GNSS receiver in the binary format specified in this document. See [Chapter 2.2.2 EPO Data Transfer Protocol](#) and [Chapter 2.2.3 AGNSS Procedure with Flash EPO](#) for details.

2.2.1. Binary Protocol

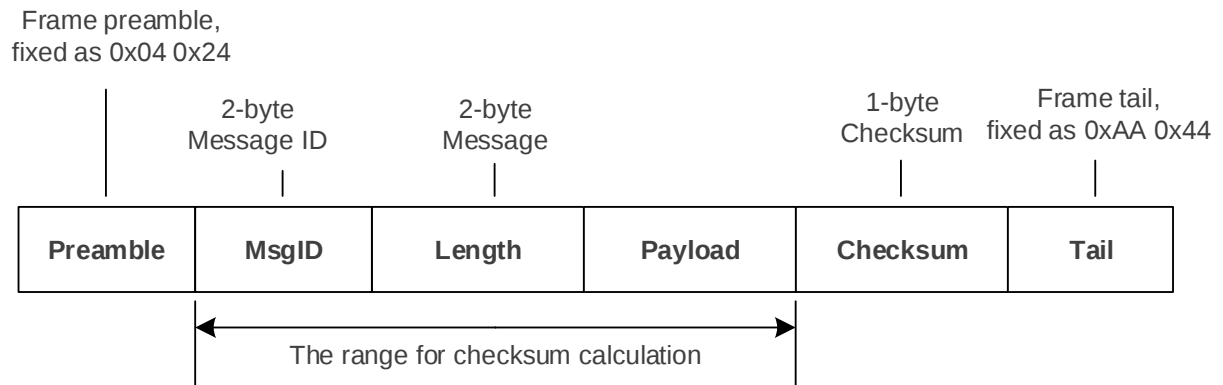


Figure 8: Binary Protocol Structure

Table 7: Description of Binary Protocol Fields

Field	Length (Byte)	Description
Preamble	2	Fixed as 0x2404. Little-endian.
MsgID	2	Message ID.
Length	2	Payload length. Unit: byte. Default packet size: 72 bytes. Little-endian.
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 0x44AA. Little-endian.

The EPO binary format is divided into start message, EPO data message and end message. The module responds with an acknowledgment message after receiving each message from the host.

Table 8: Start of EPO Binary Format

Preamble	MsgID	Length	Payload	Checksum	Tail
0x04 0x24	0xB0 0x04	1	'G' – GPS 'R' – GLONASS 'E' – Galileo 'C' – BDS	0x**	0xAA 0x44
2 Bytes	2 Bytes	2 Bytes	1 Byte	1 Byte	2 Bytes

Table 9: EPO Data Binary Format

Preamble	MsgID	Length	Payload	Checksum	Tail
0x04 0x24	0xB1 0x04	72	EPO Data	0x**	0xAA 0x44
2 Bytes	2 Bytes	2 Bytes	72 Bytes	1 Byte	2 Bytes

Table 10: End of EPO Binary Format

Preamble	MsgID	Length	Payload	Checksum	Tail
0x04 0x24	0xB2 0x04	1	'G' – GPS 'R' – GLONASS 'E' – Galileo 'C' – BDS	0x**	0xAA 0x44
2 Bytes	2 Bytes	2 Bytes	1 Byte	1 Byte	2 Bytes

Table 11: Acknowledge EPO Binary Format

Preamble	MsgID	Length	Payload	Checksum	Tail
0x04 0x24	0xE8 0x03	4	See Table 12: Payload of Acknowledgement .	0x**	0xAA 0x44
2 Bytes	2 Bytes	2 Bytes	4 Bytes	1 Byte	2 Bytes

Table 12: Payload of Acknowledgement

Byte Offset	Length (Byte)	Name	Unit	Description
0	2	MsgID_ACK	-	MsgID of the acknowledged message.

2	2	Reserved	-	Fixed data: 0x00 0x00.
---	---	----------	---	------------------------

NOTE

The *EPO_GR_3_X.DAT* file transmission process: First, send the GPS related information in the *EPO_GR_3_X.DAT* file according to the GPS process. After the GPS process is completed, send the GLONASS related information in the *EPO_GR_3_X.DAT* file according to the GLONASS process. The *EPO_GR_3_X.DAT* file transmission sequence is as follows:

1. GPS related EPO data transmission
 - 1) Start of EPO (GPS)
 - 2) EPO data (GPS)
 - 3) End of EPO (GPS)
2. GLONASS related EPO data transmission
 - 1) Start of EPO (GLONASS)
 - 2) EPO data (GLONASS)
 - 3) End of EPO (GLONASS)

2.2.2. EPO Data Transfer Protocol

When transmitting assistance data, the host first sends the start message packet, then splits the EPO data into data packets and sends them, and finally sends the end message packet. The host should follow the EPO Data Transfer Protocol when transferring EPO data to the GNSS receiver.

Pseudo code for the EPO data transfer procedure of GPS + GLONASS, for reference only:

```
#define GNSS_APP_BINARY_BINARY_PREAMBLE1      (0x04)
#define GNSS_APP_BINARY_BINARY_PREAMBLE2      (0x24)
#define GNSS_APP_BINARY_BINARY_ENDWORD1       (0xAA)
#define GNSS_APP_BINARY_BINARY_ENDWORD2       (0x44)

#define GNSS_APP_BINARY_BINARY_PREAMBLE_SIZE   (2)
#define GNSS_APP_BINARY_BINARY_CHECKSUM_SIZE   (1)
#define GNSS_APP_BINARY_BINARY_ENDWORD_SIZE   (2)
#define GNSS_APP_BINARY_BINARY_CONTROL_SIZE
    (GNSS_APP_BINARY_BINARY_PREAMBLE_SIZE + \
    GNSS_APP_BINARY_BINARY_CHECKSUM_SIZE + \
    GNSS_APP_BINARY_BINARY_ENDWORD_SIZE)

#define GNSS_APP_BINARY_BINARY_MESSAGE_ID_SIZE (2)
#define GNSS_APP_BINARY_BINARY_PAYLOAD_LENGTH_SIZE (2)
#define GNSS_APP_BINARY_BINARY_PAYLOAD_HEADER_SIZE
    (GNSS_APP_BINARY_BINARY_MESSAGE_ID_SIZE + \
    GNSS_APP_BINARY_BINARY_PAYLOAD_LENGTH_SIZE)
```

```

#define GNSS_APP_BINARY_BINARY_MAX_DATA_SIZE (512)
#define GNSS_APP_BINARY_BINARY_MAX_PAYLOAD_DATA_SIZE
    (GNSS_APP_BINARY_BINARY_MAX_DATA_SIZE - \
    GNSS_APP_BINARY_BINARY_CONTROL_SIZE - \
    GNSS_APP_BINARY_BINARY_PAYLOAD_HEADER_SIZE)

typedef enum gnss_app_binary_binary_decode_results {
    GNSS_APP_BINARY_BINARY_DECODE_SUCCESS = 0,
    GNSS_APP_BINARY_BINARY_DECODE_WRONG_PARAMETER = -1,
    GNSS_APP_BINARY_BINARY_DECODE_WRONG_PREAMBLE = -2,
    GNSS_APP_BINARY_BINARY_DECODE_WRONG_CHECKSUM = -3,
    GNSS_APP_BINARY_BINARY_DECODE_WRONG_ENDWORD = -4,
}gnss_app_binary_binary_decode_results_t;
typedef struct gnss_app_binary_binary_payload {
    uint16_t message_id;
    uint16_t data_size; /* actual size of data in payload data buffer */
    uint8_t data[GNSS_APP_BINARY_BINARY_MAX_PAYLOAD_DATA_SIZE];
}gnss_app_binary_binary_payload_t;

uint8_t gnss_app_binary_calculate_binary_checksum(const
    gnss_app_binary_binary_payload_t* const payload)
{
    uint8_t checksum = 0;
    uint8_t* pheader = NULL;
    uint8_t* pdata = NULL;
    uint16_t i;
    if (NULL == payload) {
        return 0;
    }
    /* The checksum is the 8-bit exclusive OR of all bytes in the payload. */
    pheader = (uint8_t*)payload;
    for (i = 0; i < GNSS_APP_BINARY_BINARY_PAYLOAD_HEADER_SIZE; i++) {
        checksum ^= *pheader;
        pheader++;
    }
    pdata = (uint8_t*)payload->data;
    for (i = 0; i < payload->data_size; i++) {
        checksum ^= *pdata;
        pdata++;
    }
    return checksum;
}

```

```

int16_t gnss_app_binary_encode_binary_packet(uint8_t* const buffer, uint16_t
    max_buffer_size, const gnss_app_binary_binary_payload_t* const payload)
{
    uint8_t* pbyte;
    uint16_t required_length;
    if (NULL == buffer || payload == NULL) {
        return -1;
    }
    required_length = payload->data_size + GNSS_APP_BINARY_BINARY_CONTROL_SIZE +
        GNSS_APP_BINARY_BINARY_PAYLOAD_HEADER_SIZE;
    if (max_buffer_size < required_length) {
        return -1;
    }
    memset((void*)buffer, 0, max_buffer_size);
    buffer[0] = GNSS_APP_BINARY_BINARY_PREAMBLE1;
    buffer[1] = GNSS_APP_BINARY_BINARY_PREAMBLE2;
    pbyte = &buffer[2];
    memcpy(pbyte, payload, GNSS_APP_BINARY_BINARY_PAYLOAD_HEADER_SIZE);
    pbyte += GNSS_APP_BINARY_BINARY_PAYLOAD_HEADER_SIZE;
    memcpy(pbyte, payload->data, payload->data_size);
    pbyte += payload->data_size;
    *pbyte++ = gnss_app_binary_calculate_binary_checksum(payload);
    *pbyte++ = GNSS_APP_BINARY_BINARY_ENDWORD1;
    *pbyte = GNSS_APP_BINARY_BINARY_ENDWORD2;
    return required_length;
}

FILE* gnss_epo_file = NULL;
#define GNSS_MAX_EPO_NUMBER (37)
#define GNSS_MAX_RECORD_SIZE (72)
static uint32_t gnss_epo_sv_buf[(GNSS_MAX_EPO_NUMBER *
    GNSS_MAX_RECORD_SIZE) / sizeof(uint32_t)];
int16_t gnss_epo_encode_binary(uint16_t msg_id, char* buffer, uint16_t buffer_size,
    char* data_input, int32_t data_length) {
    gnss_app_binary_binary_payload_t payload;
    int16_t binary_message_length;
    memset((void*)&payload, 0, sizeof(gnss_app_binary_binary_payload_t));
    payload.message_id = msg_id;
    payload.data_size = (uint16_t)data_length;
    memcpy(payload.data, data_input, sizeof(uint8_t) * data_length);
    binary_message_length = gnss_app_binary_encode_binary_packet(buffer,
        buffer_size, &payload);
    return binary_message_length;
}

```

```

void gnss_epo_binary_demo() {
    gnss_app_binary_data_t data;
    gnss_app_binary_data_result_t result = { 0 };
    uint32_t* epobuf;
    int32_t i;
    char buffer[500];
    uint16_t length = 0;
    int32_t buffer_size = 0;
    uint8_t segment = 0;

    uint8_t curr_sys_type = 'G'; //type is the GPS
    gnss_epo_file = fopen("EPO_GR_3_1.DAT", "rb");
    length = gnss_epo_encode_binary(1200, buffer, 512, &curr_sys_type, 1);
    gnss_app_uart_send_data(buffer, length);
    memset(&gnss_epo_sv_buf, 0, sizeof(gnss_epo_sv_buf));
    while (gnss_epo_read_data(&gnss_epo_sv_buf, 32 * GNSS_MAX_RECORD_SIZE, segment *
        (32 + 24) * GNSS_MAX_RECORD_SIZE)) {
        segment++;
        for (i = 0; i < 32; i++) {
            epobuf = (uint32_t*)(gnss_epo_sv_buf + ((i * GNSS_MAX_RECORD_SIZE) /
                4));
            length = gnss_epo_encode_binary(1201, buffer, 512, (char*)epobuf,
                GNSS_MAX_RECORD_SIZE);
            gnss_app_uart_send_data(buffer, length);
        }
        memset(&gnss_epo_sv_buf, 0, sizeof(gnss_epo_sv_buf));
    }
    length = gnss_epo_encode_binary(1202, buffer, 512, &curr_sys_type, 1);
    gnss_app_uart_send_data(buffer, length);

    curr_sys_type = 'R'; //type is the GLONASS
    length = gnss_epo_encode_binary(1200, buffer, 512, &curr_sys_type, 1);
    gnss_app_uart_send_data(&data, &result);
    memset(&gnss_epo_sv_buf, 0, sizeof(gnss_epo_sv_buf));
    segment = 0;
    while (gnss_epo_read_data(&gnss_epo_sv_buf, 37 * GNSS_MAX_RECORD_SIZE, (segment
        * (32 + 24) * GNSS_MAX_RECORD_SIZE) + (32 * GNSS_MAX_RECORD_SIZE))) {
        segment++;
        for (i = 0; i < 24; i++) {
            epobuf = (uint32_t*)(gnss_epo_sv_buf + ((i * GNSS_MAX_RECORD_SIZE) /
                4));
            length = gnss_epo_encode_binary(1201, buffer, 512, (char*)epobuf,
                GNSS_MAX_RECORD_SIZE);
        }
    }
}

```

```

        gnss_app_uart_send_data(buffer, length);
    }
    memset(&gnss_epo_sv_buf, 0, sizeof(gnss_epo_sv_buf));
}
length = gnss_epo_encode_binary(1202, buffer, 512, &curr_sys_type, 1);
gnss_app_uart_send_data(buffer, length);
fclose(gnss_epo_file);
}

```

2.2.3. AGNSS Procedure with Flash EPO

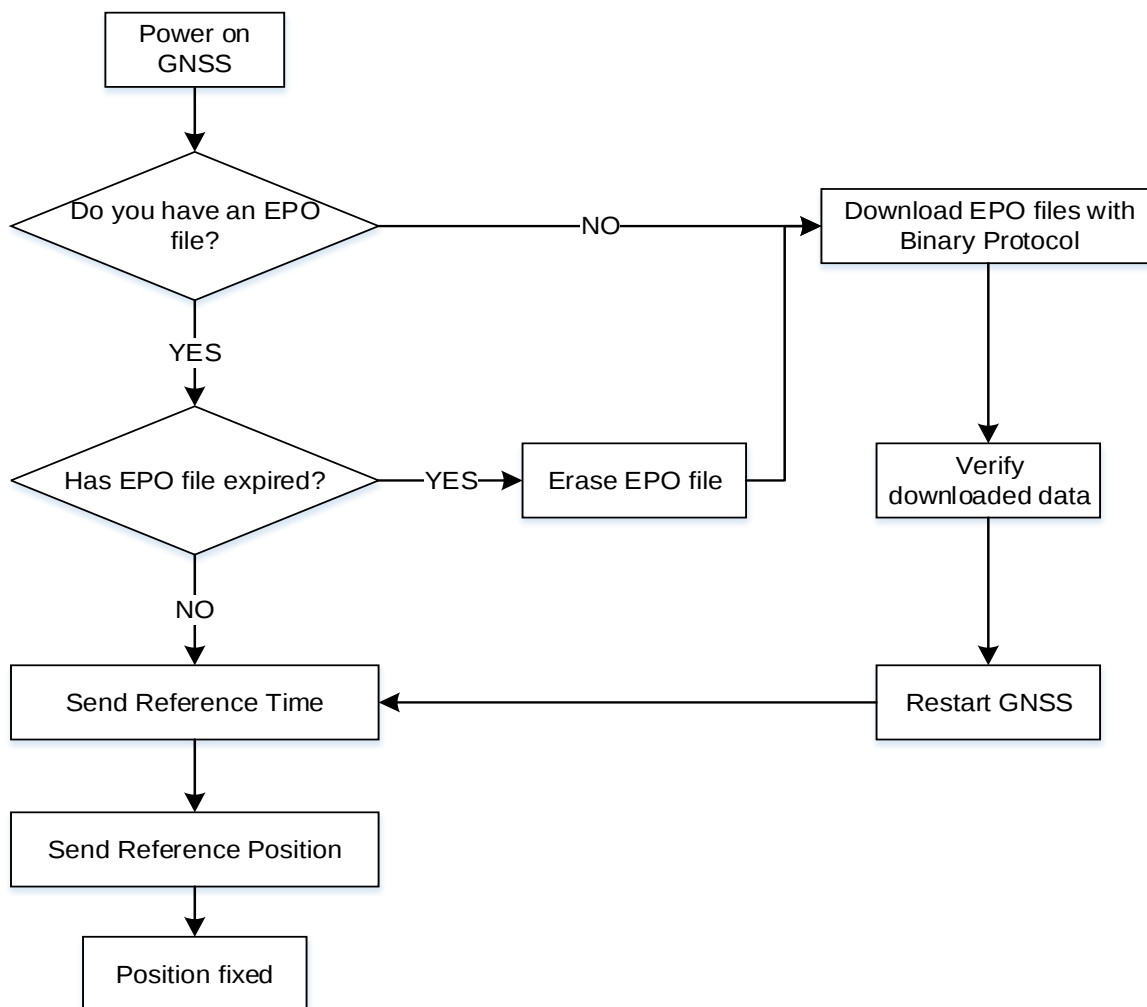


Figure 9: AGNSS Procedure with Flash EPO

- Step 1** Power on the GNSS module.
- Step 2** Check if there are EPO data in the GNSS module flash memory with **\$PAIR470**.
- Step 3** If the flash memory contains EPO data, go to the next step to check data validity. Otherwise, download EPO data to the GNSS module and verify the downloaded data, then restart the GNSS

module and go to **Step 5**.

Step 4 Check whether the EPO file in the GNSS module flash memory has expired.

Step 5 If the EPO file is still valid, go to **Step 6**. Otherwise, erase the expired EPO file with **\$PAIR472** and download a new EPO file.

Step 6 Send reference time to the GNSS module with **\$PAIR590**.

Step 7 Send reference position to GNSS module with **\$PAIR600**.

Step 8 Wait for the GNSS module to fix position.

2.3. AGNSS with Host EPO

Host EPO allows for a simpler text-based implementation that 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 PAIR sentences and the whole data transfer can be performed in NMEA mode. See [Chapter 3.1 EPO Related Messages](#) for detailed description of **\$PAIR471**, **\$PAIR590** and **\$PAIR600**.

2.3.1. Recommended Sequence for Host EPO

After the module is powered on, it sends an aiding request ¹⁾ to notify the expiration of the stored GNSS assistance data when both assistance data and ephemeris are invalid. Therefore, if the module does not report an aiding request, it is recommended to determine whether the assistance data have expired by sending **\$PAIR470** after the host receives the system startup message. The host sends the assistance data in the sequence shown in [Figure 10: Suggested Sequence for Host EPO](#).

Host EPO procedure:

1. GNSS module starts up;
2. Host sends Reference Time;
3. Host sends Reference Position;
4. Host sends EPO data.

The supplied Reference Time, Reference Position and EPO data must comply with the requirements listed in [Chapter 1.1 Network-based AGNSS](#).

NOTE

In the current implementation, the host needs to wait for a **\$PAIR001** packet to be returned before sending another segment of EPO data. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

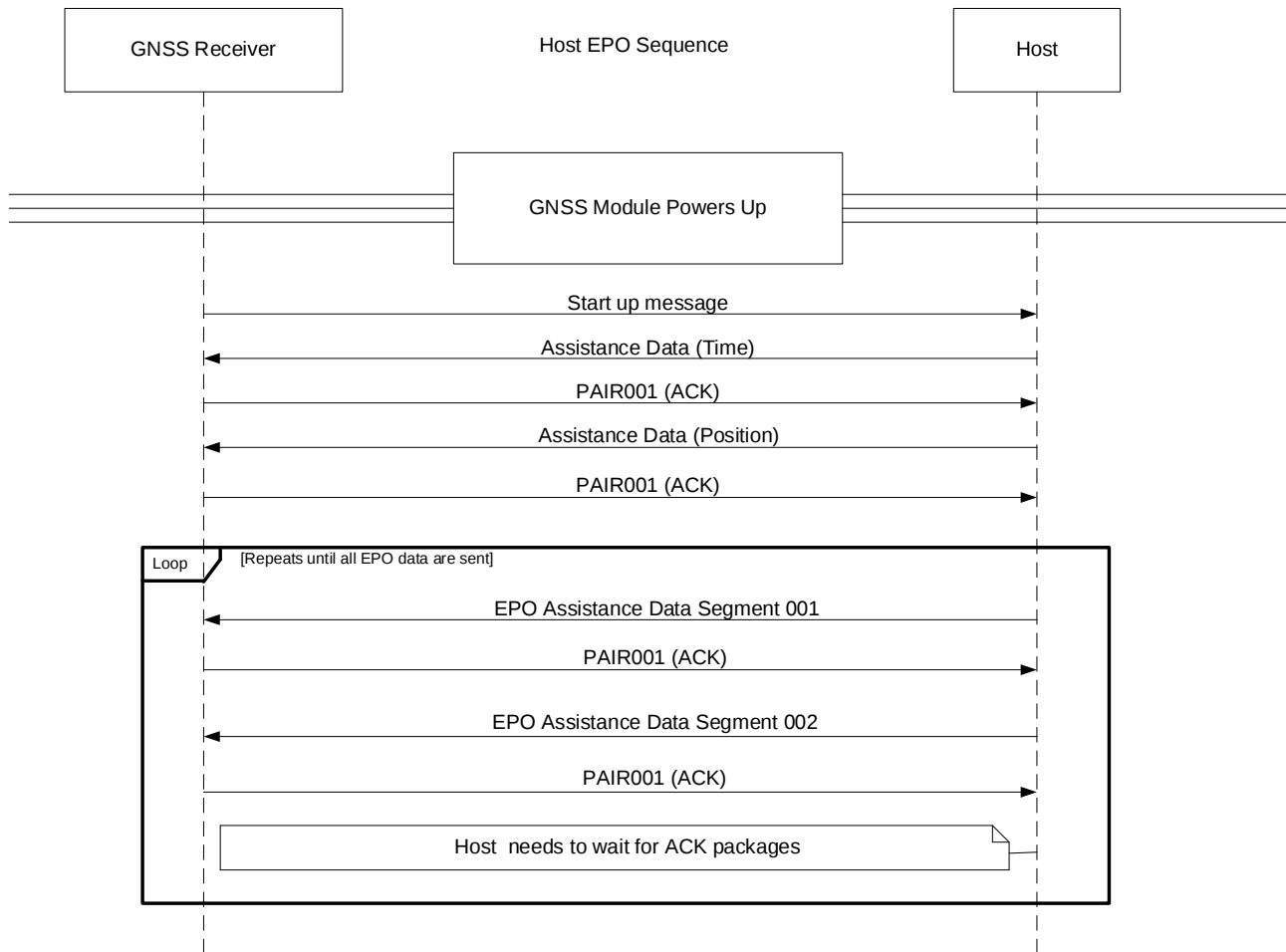


Figure 10: Suggested Sequence for Host EPO

2.3.2. Sample Code to Send EPO

The following is the reference code to send one segment EPO data to GNSS module. It indicates how to construct PAIR messages for GNSS receiver. PAIR messages for Reference Time and Reference Position are not included in this example.

```

#define GNSS_GLONASS_EPO_BASE_ID (64)
#define GNSS_GALILEO_EPO_BASE_ID (100)
#define GNSS_BDS_EPO_BASE_ID (200)

#define MNL_SERVICE_MAX_COMMAND_LEN (352)
#define EPO_DEMO_RECORD_SIZE (72)

typedef enum{
    EPO_DEMO_MODE_GPS,
    EPO_DEMO_MODE_GLONASS,

```

```

    EPO_DEMO_MODE_GALILEO,
    EPO_DEMO_MODE_BEIDOU
}epo_demo_mode_t;

int32_t epo_demo_get_sv_prn(int32_t type, uint8_t *data)
{
    int32_t sv_id, sv_prn = 0;
    sv_id = data[3];

    switch(type) {
        case EPO_DEMO_MODE_GPS:
            sv_prn = sv_id;
            break;
        case EPO_DEMO_MODE_GLONASS:
            sv_prn = sv_id - GNSS_GLONASS_EPO_BASE_ID;
            break;
        case EPO_DEMO_MODE_GALILEO:
            if(sv_id == 255) {
                sv_prn = 255;
            } else {
                sv_prn = sv_id - GNSS_GALILEO_EPO_BASE_ID;
            }
            break;
        case EPO_DEMO_MODE_BDS:
            if(sv_id == 255) {
                sv_prn = 255;
            } else {
                sv_prn = sv_id - GNSS_BDS_EPO_BASE_ID;
            }
            break;
        default:
            sv_prn = 0;
    }
    return sv_prn;
}

void epo_demo_send_data(epo_demo_epo_data_t *data_p, int32_t data_num, int32_t
type){

    char temp_buffer[MNL_SERVICE_MAX_COMMAND_LEN] = {0};
    uint8_t data_buffer[EPO_DEMO_RECORD_SIZE] = {0};
    int32_t i;
    int32_t sv_prn = 0;

```

```

for(i = 0; i < data_num; i++) {
    unsigned int *epobuf = (unsigned int *)data_buffer;
    epo_demo_epo_fread(data_p, data_buffer, EPO_DEMO_RECORD_SIZE);
    sv_prn = epo_demo_get_sv_prn(type, data_buffer);

    sprintf((char *) temp_buffer,
        "471,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X,%X",
        (unsigned int)type,
        (unsigned int)sv_prn,
        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]);

    gnss_app_send_command_ex(temp_buffer);
    memset(temp_buffer, 0, MNL_SERVICE_MAX_COMMAND_LEN);
}
}

```

2.4. AGNSS with EASY

EASY only supports GPS constellation. The figure below illustrates how the EASY function is activated and utilized by the module for orbit prediction during free time when it is not processing internal tasks. The EASY aiding data supplements the lack of the EPO aiding data in scenarios where the module may not have access to an external network and server derived predictions.

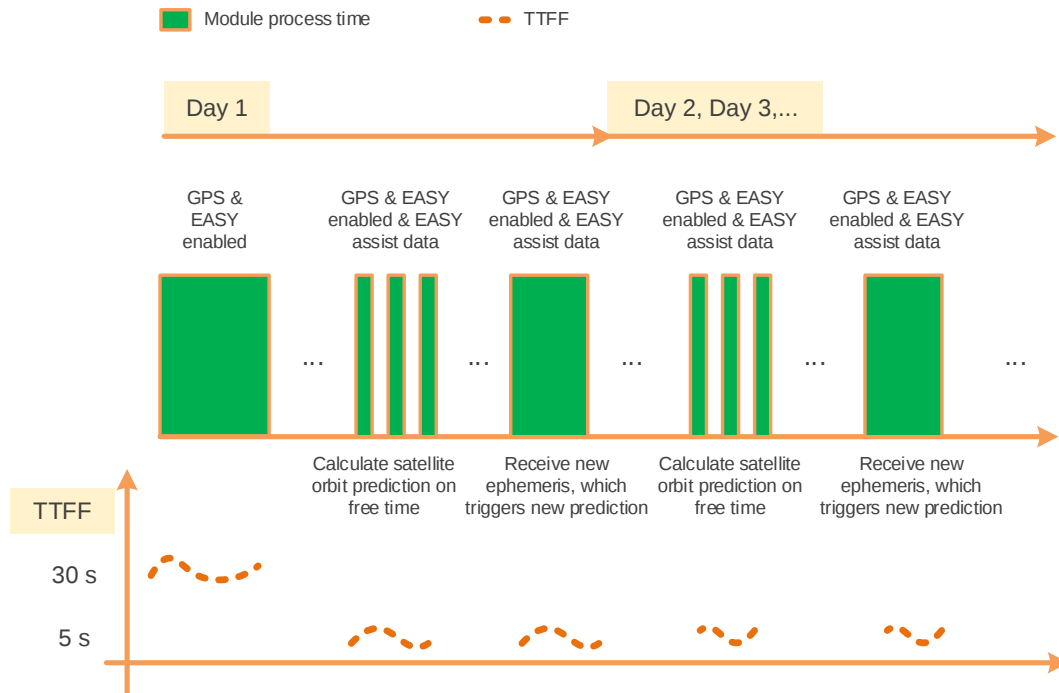


Figure 11: Mechanism of EASY Function

The recommended process for utilizing the EASY function is illustrated below:

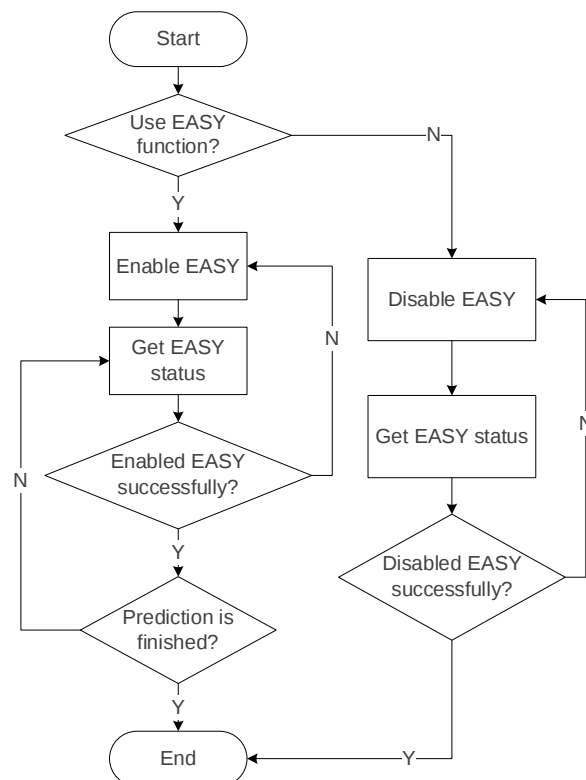


Figure 12: Recommended Process for Using EASY

NOTE

Performing “Enable EASY” is optional since EASY is enabled by default.

2.5. AGNSS with EPOC

EPOC in contrast to EASY supports three options of constellation configurations: GPS only, GPS + Galileo, and GPS + BDS (MEO and IGSO). In each configuration, the module can receive and process data from up to 40 satellites. If a particular constellation search is not turned on while EPOC prediction is enabled, the orbit will not be predicted for that constellation.

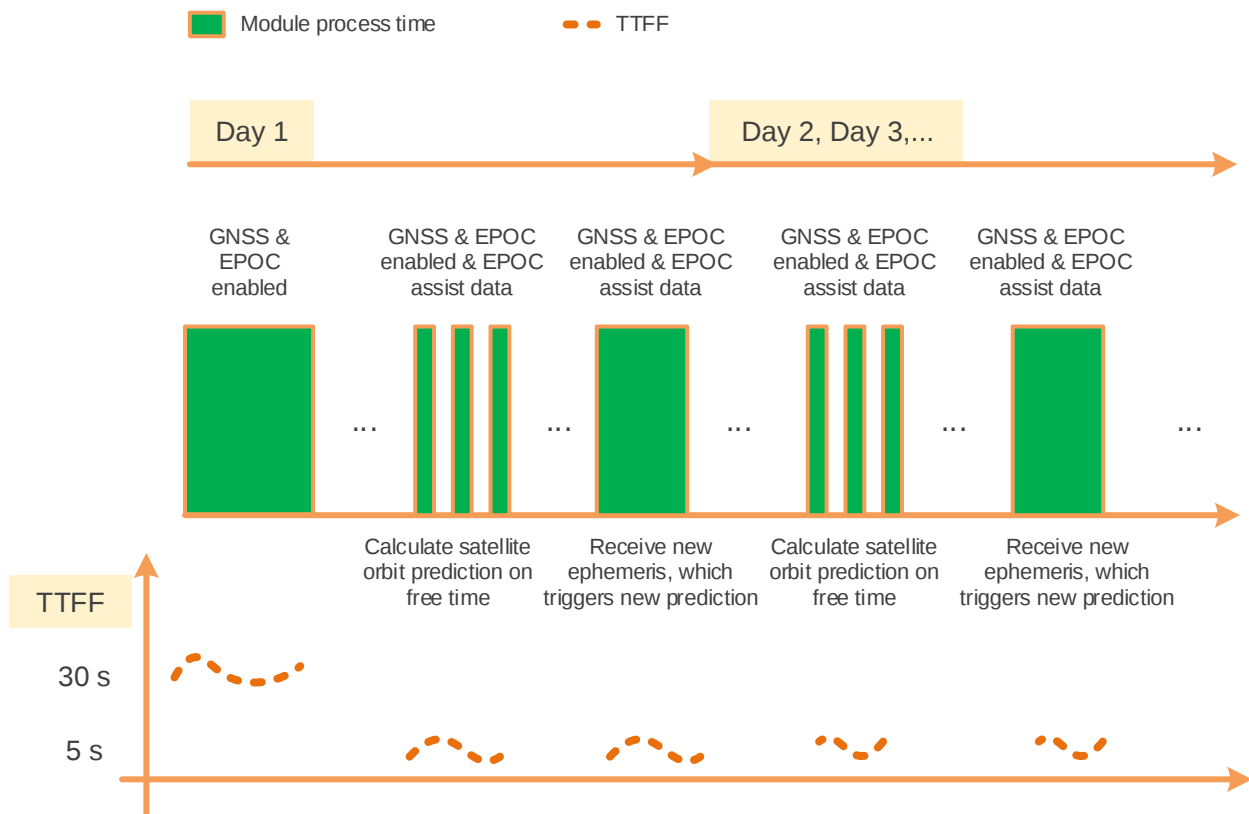


Figure 13: Mechanism of EPOC Function

The recommended process for utilising the EPOC function is illustrated below:

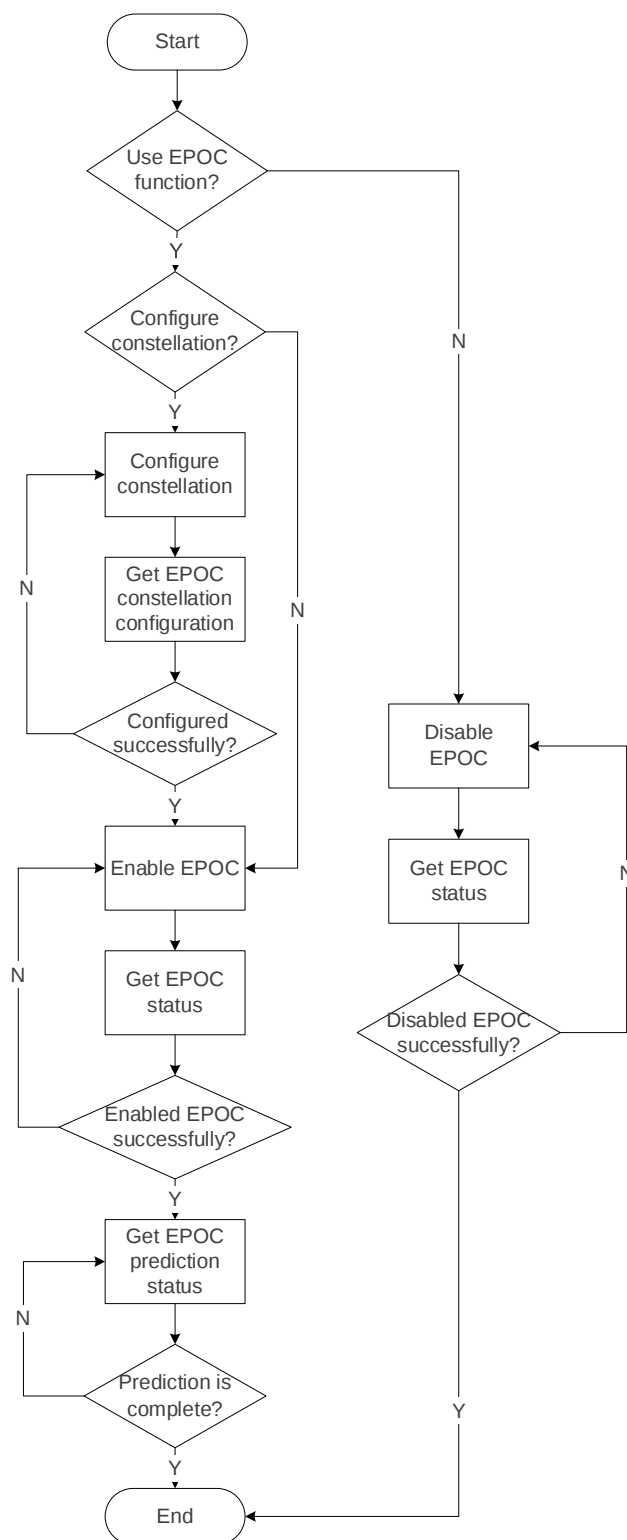


Figure 14: Recommended Process for Using EPOC

NOTE

Performing “Enable EPOC” is optional since EPOC is enabled by default.

3 AGNSS Related Messages

This chapter outlines AGNSS related PAIR messages (proprietary NMEA messages defined by the chipset supplier). “P” means proprietary message, “AIR” means the command defined by the chipset supplier.

3.1. EPO Related Messages

3.1.1. PAIR010: PAIR_REQUEST_AIDING

Notifies the expiration of GNSS assistance data stored in the module. This message is automatically output when the module powers up.

Type:

Output

Synopsis:

```
$PAIR010,<Type>,<GNSS_System>,<WN>,<TOW>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Type>	Numeric	-	Type of data to be updated. 0 = EPO data 1 = Time 2 = Position Note that when <Type> = 1 or 2, the fields following <Type> are replaced with a “-1”.
<GNSS_System>	Numeric	-	Type of GNSS data needed. 0 = GPS data 1 = GLONASS data 2 = Galileo data 3 = BDS data
<WN>	Numeric	Week	Week Number (accommodating roll-over).
<TOW>	Numeric	Second	Time of Week.

Example:

```
//Send GPS EPO data when this sentence is received:
```

```
$PAIR010,0,0,2044,369413*33
```

```
//Send reference time when this sentence is received:
```

```
$PAIR010,1,-1*16
```

```
//Send reference position when this sentence is received:
```

```
$PAIR010,2,-1*15
```

NOTE

The GNSS system outputs this command automatically. Do not send it to the GNSS system.

3.1.2. PAIR470: PAIR_EPO_GET_STATUS

Queries the status of EPO data stored in the module.

Type:

Query

Synopsis:

```
$PAIR470,<System_ID>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<System_ID>	Numeric	-	GNSS system ID. 0 = GPS 1 = GLONASS 2 = Galileo 3 = BDS

Result:

Returns **\$PAIR001** message and the query result. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Query result message format:

```
$PAIR470,<System_ID>,<Set>,<FWN>,<FTOW>,<LWN>,<LTOW>,<FCWN>,<FCTOW>,<LCWN>,<LC
```

```
TOW>*<Checksum><CR><LF>
```

Parameters included in the result:

Field	Format	Unit	Description
<System_ID>	Numeric	-	GNSS system ID. 0 = GPS 1 = GLONASS 2 = Galileo 3 = BDS
<Set>	Numeric	-	Total number of EPO data sets stored in the module.
<FWN>	Numeric	-	GPS week number of the first set of EPO data stored in flash.
<FTOW>	Numeric	-	GPS TOW of the first set of EPO data stored in flash.
<LWN>	Numeric	-	GPS week number of the last set of EPO data stored in flash.
<LTOW>	Numeric	-	GPS TOW of the last set of EPO data stored in flash.
<FCWN>	Numeric	-	GPS week number of the first set of EPO data that are currently being used.
<FCTOW>	Numeric	-	GPS TOW of the first set of EPO data that are currently being used.
<LCWN>	Numeric	-	GPS week number of the last set of EPO data that are currently being used.
<LCTOW>	Numeric	-	GPS TOW of the last set of EPO data that are currently being used.

Example:

```
$PAIR470,0*25
$PAIR001,470,0*38
$PAIR470,0,1,2098,194400,2098,216000,2098,194400,2098,216000*38
```

3.1.3. PAIR471: PAIR_EPO_SET_DATA

Sends the packet containing EPO data for a single satellite.

Type:

Input

Synopsis:

```
$PAIR471,<System_ID>,<SV_ID>,<W[0]>,...,<W[17]>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<System_ID>	Numeric	-	GNSS system ID. 0 = GPS 1 = GLONASS 2 = Galileo 3 = BDS
<SV_ID>	Hexadecimal	-	PRN number of the satellite whose EPO data are being sent. GPS range: 1–32. GLONASS range: 1–24. Galileo range: 1–36. BDS range: 1–54. Special 255: BDS IONO data. Special 254: Galileo IONO data.
<W[0]>–<W[17]>	-	-	18 words (LSB-first) of one EPO segment data (total 72 bytes).

Result:

Returns **\$PAIR001** message. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Example:

```
$PAIR471,1,16,56056272,F2BC0244,4F19AE34,F95C534D,FAE67014,4F19AF6B,F96749BD,9F341F2
D,6F4EA9F,77DB4710,66ADAC2,9ADF3B01,8CC8B19C,29D2D20C,FC5B2E94,1000001C,11005000,
748B45F4*0A
$PAIR001,471,0*39
$PAIR471,56056272*13
```

NOTE

1. This command needs to be sent every time after GNSS module reboots.
2. In addition to a **\$PAIR001** message, a **\$PAIR471,<Debug_Info>*<Checksum>** message is also returned for debug purpose.

3.1.4. PAIR472: PAIR_EPO_ERASE_FLASH_DATA

Erases the EPO data stored in the flash memory.

Type:

Command

Synopsis:

```
$PAIR472*<Checksum><CR><LF>
```

Parameter:

None

Result:

Returns **\$PAIR001** message. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Example:

```
$PAIR472*3B
$PAIR001,472,0*3A
```

3.1.5. PAIR590: PAIR_TIME_SET_REF.UTC

Sends reference UTC time to the GNSS receiver for faster TTFF. Local time should be avoided due to time-zone offset. To achieve a faster TTFF, the reference time should be accurate within 3 seconds and must be specified in UTC time.

Type:

Input

Synopsis:

```
$PAIR590,<YYYY>,<MM>,<DD>,<hh>,<mm>,<ss>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<YYYY>	Numeric	-	UTC year. Minimum value: 1980.
<MM>	Numeric	-	UTC month. Range: 1–12.
<DD>	Numeric	-	UTC day. Range: 1–31.
<hh>	Numeric	-	UTC hours. Range: 0–23.
<mm>	Numeric	-	UTC minutes. Range: 0–59.
<ss>	Numeric	-	UTC seconds. Range: 0–59.

Result:

Returns **\$PAIR001** message. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Example:

```
$PAIR590,2019,2,10,9,0,58*0B
$PAIR001,590,0*37
```

NOTE

This command needs to be sent every time after GNSS module reboots.

3.1.6. PAIR600: PAIR_LOC_SET_REF

Sends reference position to the GNSS receiver for faster TTFF. The reference position should be accurate within 30 km from the receiver's actual position. Keep in mind that if the receiver's view of the sky is limited, the accuracy of the reference position needs to be increased.

Type:

Input

Synopsis:

```
$PAIR600,<Lat>,<Lon>,<Height>,<AccMaj>,<AccMin>,<Bear>,<AccVert>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Lat>	Numeric	Degree	Reference latitude. Range: -90 to 90. Minus: south; plus: north. It is recommended to express this value in floating point with 6 decimal points.
<Lon>	Numeric	Degree	Reference longitude. Range: -180 to 180. Minus: west; plus: east. It is recommended to express this value in floating point with 6 decimal points.
<Height>	Numeric	Meter	Reference height.
<AccMaj>	Numeric	Meter	Semi-major RMS accuracy.
<AccMin>	Numeric	Meter	Semi-minor RMS accuracy.
<Bear>	Numeric	Degree	Bearing.
<AccVert>	Numeric	Meter	Vertical RMS accuracy.

Result:

Returns **\$PAIR001** message. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Example:

```
$PAIR600,24.772816,121.022636,175.0,50.0,50.0,0.0,100.0*06
$PAIR001,600,0*3D
```

NOTE

This command needs to be sent every time after GNSS module reboots.

3.2. EASY Related Messages

3.2.1. PAIR490: PAIR_EASY_ENABLE

Enables/disables EASY function.

Type:

Set

Synopsis:

```
$PAIR490,<Enabled>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Enabled>	Numeric	-	EASY function setting. 0 = Disable <u>1</u> = Enable

Result:

Returns **\$PAIR001** message. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Example:

```
$PAIR490,1*2A
$PAIR001,490,0*36
```

3.2.2. PAIR491: PAIR_EASY_GET_STATUS

Gets the status of EASY function and prediction status (EASY data extension status).

Type:

Get

Synopsis:

```
$PAIR491*<Checksum><CR><LF>
```

Parameter:

None

Result:

Returns **\$PAIR001** message and the query result. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Query result message format:

```
$PAIR491,<Enabled>[,<Status>]*<Checksum><CR><LF>
```

Parameters included in the result:

Field	Format	Unit	Description
<Enabled>	Numeric	-	EASY function setting. 0 = Disabled 1 = Enabled
<Status>	Numeric	-	EASY data extension status. 0 = Not finished 1 = 1-day extension finished 2 = 2-day extension finished 3 = 3-day extension finished

Example:

```
$PAIR491*36
$PAIR001,491,0*37
$PAIR491,1,0*37
```

NOTE

1. The status of EASY function and prediction status are dynamic and reset when the module receives new ephemeris data for a satellite or searches a new satellite. This reset is indicated by the query result **\$PAIR491,1,0*37**. The queried status will remain unchanged until the completion of orbit predictions for all visible satellites.
2. If the EASY status remains "0" after the module has fixed its position for 12.5 minutes, it does not mean that EASY is incapable of performing orbit predictions or lacks the ability to accelerate TTFF performance. The prerequisite for a change in the EASY status is that all visible satellites have completed their predictions. However, due to certain reasons (such as signal obstruction leading to incomplete reception of some satellite ephemerides), it is possible that some satellites have not met the conditions for completing predictions, which would result in the EASY status remaining "0".
3. If EASY function is disabled, only the value of **<Enabled>** will be returned after executing this command.

3.3. EPOC Related Messages

3.3.1. PAIR496: PAIR_EPOC_ENABLE

Enables/disables EPOC function.

Type:

Set

Synopsis:

```
$PAIR496,<Enabled>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Enabled>	Numeric	-	EPOC function setting. 0 = Disable <u>1</u> = Enable

Result:

Returns **\$PAIR001** message. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Example:

```
$PAIR496,1*2C
$PAIR001,496,0*30
```

3.3.2. PAIR498: PAIR_EPOCH_SET_CONFIG

Sets EPOCH constellation configuration.

Type:

Set

Synopsis:

```
$PAIR498,<Constellation>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Constellation>	Numeric	-	EPOCH constellation configuration. Default configuration: GPS. Bit 0 = GPS Bit 1 = Galileo Bit 2 = BDS

Result:

Returns **\$PAIR001** message. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Example:

```
$PAIR498,1*22
$PAIR001,498,0*3E
```

NOTE

<Constellation> is a decimal value. Only GPS, GPS + Galileo, and GPS + BDS are available.

3.3.3. PAIR507: PAIR_EPOCH_CLEAR_DATA

Clears all EPOCH data from NVM.

Type:

Command

Synopsis:

```
$PAIR507*<Checksum><CR><LF>
```

Parameter:

None

Result:

Returns **\$PAIR001** message. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Example:

```
$PAIR507*38
$PAIR001,507,0*39
```

3.3.4. PAIR508: PAIR_EPOCH_GET_STATUS

Gets the status of EPOCH function and constellation configuration.

Type:

Get

Synopsis:

```
$PAIR508*<Checksum><CR><LF>
```

Parameter:

None

Result:

Returns **\$PAIR001** message and the query result. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Query result message format:

```
$PAIR508,<Enabled>,<Constellation>*<Checksum><CR><LF>
```

Parameters included in the result:

Field	Format	Unit	Description
<Enabled>	Numeric	-	EPOCH function setting. 0 = Disabled 1 = Enabled
<Constellation>	Hexadecimal	-	EPOCH constellation configuration. Bit 0 = GPS Bit 1 = Galileo

Field	Format	Unit	Description
			Bit 2 = BDS

Example:

```
$PAIR508*37
$PAIR001,508,0*36
$PAIR508,1,1*37
```

3.3.5. PAIR509: PAIR_EPOCH_GET_PREDICTION_STATUS

Gets EPOCH prediction status.

Type:

Get

Synopsis:

```
$PAIR509,<Constellation>*<Checksum><CR><LF>
```

Parameter:

Field	Format	Unit	Description
<Constellation>	Hexadecimal	-	EPOCH constellation configuration. Bit 0 = GPS Bit 1 = Galileo Bit 2 = BDS You can only query one constellation at a time, otherwise an error will be returned.

Result:

Returns **\$PAIR001** message and the query result. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Query result message format:

```
$PAIR509,<Status>,<Constellation>,<SatelliteStatus>*<Checksum><CR><LF>
```

Parameters included in the result:

Field	Format	Unit	Description
<Status>	Numeric	-	Prediction status. 1 = Prediction in progress. 2 = All received ephemeris data have been calculated for prediction. 3 = No valid ephemeris data.
<Constellation>	Numeric	-	EPOC constellation configuration. Bit 0 = GPS Bit 1 = Galileo Bit 2 = BDS
<SatelliteStatus>	Hexadecimal	-	Satellite available bitmask. Bit number represents satellite ID and "1" in a specific bit position indicates that the orbit prediction for the corresponding satellite is complete.

Example:

```
$PAIR509,1*2B
$PAIR001,509,0*37
$PAIR509,2,1,000000000000D2D2C*6A
```

NOTE

<SatelliteStatus>: Satellite number in bitwise format. For example, "000000000000D2D2C" indicates that the 3-day predictions for satellites 3, 4, 6, 9, 11, 12, 14, 17, 19, and 20 have been completed.

3.4. EASY & EPOC Related Messages

3.4.1. PAIR511: PAIR_NVRAM_SAVE_NAVIGATION_DATA

Saves current navigation data from RTC RAM to NVM.

Type:

Command

Synopsis

```
$PAIR511*<Checksum><CR><LF>
```

Parameter:

None

Result:

Returns **\$PAIR001** message. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Example:

//If the position fix rate is 1 Hz.

\$PAIR511*3F
\$PAIR001,511,0*3E

//If the position fix rate exceeds 1 Hz.

\$PAIR382,1*2E
\$PAIR001,382,0*32
\$PAIR003*39
\$PAIR001,003,0*38
\$PAIR511*3F
\$PAIR001,511,0*3E
\$PAIR002*38
\$PAIR001,002,0*39

NOTE

1. If the backup domain cannot be powered after the power supply of the module is cut off, you must send **\$PAIR511** every time the parameters are modified.
1. If position fix rate exceeds 1 Hz, power off the GNSS system with **\$PAIR382,1*2E** and **\$PAIR003*39** in sequence before sending this command. After sending **\$PAIR511*3F**, send **\$PAIR002*38** to re-power the module. This limitation does not apply to fix rates below 1 Hz.

3.4.2. PAIR512: PAIR_NVRAM_CLEAR_NAVIGATION_DATA

Clears navigation data in both RTC RAM and NVM.

Type:

Command

Synopsis

\$PAIR512*<Checksum><CR><LF>

Parameter:

None

Result:

Returns **\$PAIR001** message. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Example:

```
$PAIR512*3C
$PAIR001,512,0*3D
```

3.4.3. PAIR513: PAIR_NVRAM_SAVE_SETTING

Saves current configuration from RTC RAM to NVM.

Type:

Command

Synopsis:

```
$PAIR513*<Checksum><CR><LF>
```

Parameter:

None

Result:

Returns **\$PAIR001** message. For details about **\$PAIR001**, see [document \[1\] protocol specification](#).

Example:

```
//If the position fix rate is 1 Hz.
$PAIR513*3D
$PAIR001,513,0*3C

//If the position fix rate exceeds 1 Hz.
$PAIR382,1*2E
$PAIR001,382,0*32
$PAIR003*39
$PAIR001,003,0*38
$PAIR513*3D
$PAIR001,513,0*3C
$PAIR002*38
$PAIR001,002,0*39
```

NOTE

1. If the backup domain cannot be powered after the power supply of the module is cut off, you must send **\$PAIR513** every time the parameters are modified.
2. If position fix rate exceeds 1 Hz, power off the GNSS system with **\$PAIR382,1*2E** and **\$PAIR003*39** in sequence before sending this command. After sending **\$PAIR513*3D**, send **\$PAIR002*38** to re-power the module. This limitation does not apply to fix rates below 1 Hz.

4 EPO Implementation with QGNSS

This chapter describes how to download EPO data through QGNSS. Contact Quectel Technical Support for details on QGNSS.

4.1. Flash EPO Implementation with QGNSS

Steps to download Flash EPO with the QGNSS tool:

1. Run the QGNSS tool.
2. In the main interface, click “**AGNSS**” → “**Assistant GNSS Offline**” as shown below.

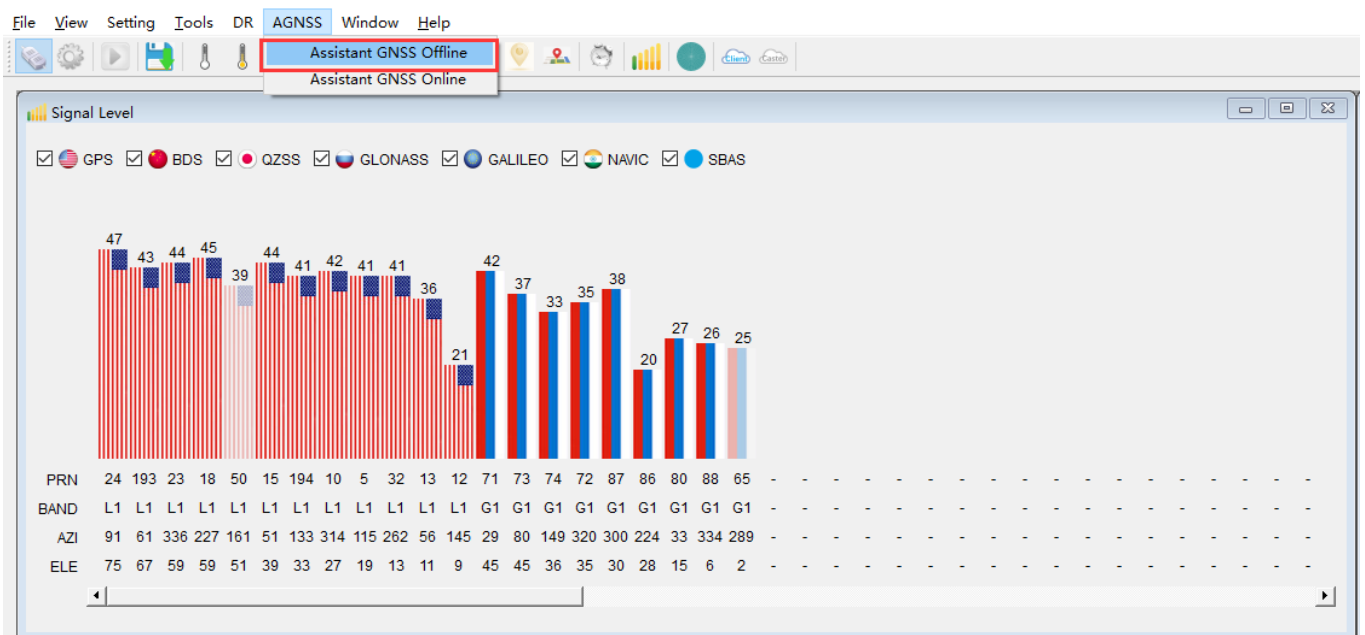


Figure 15: QGNSS Interface for Setting Flash EPO

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) Select **“Satellites type”**.

- e) Click the “...” button to select EPO file.
- f) Click “Download” button to download the EPO file to module.

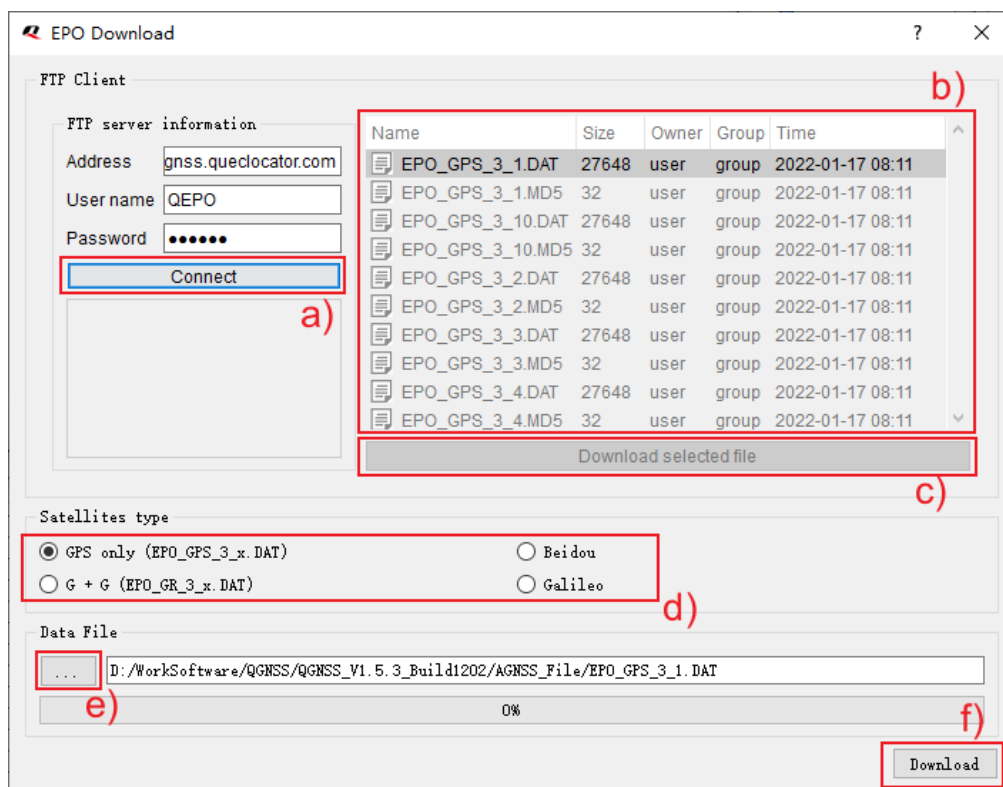


Figure 16: Download Flash EPO File

4.2. Host EPO Implementation with QGNSS

Steps to download Host EPO with the QGNSS tool:

1. Run the QGNSS tool.
2. In the main interface, click “AGNSS” → “Assistant GNSS Online” as shown below.

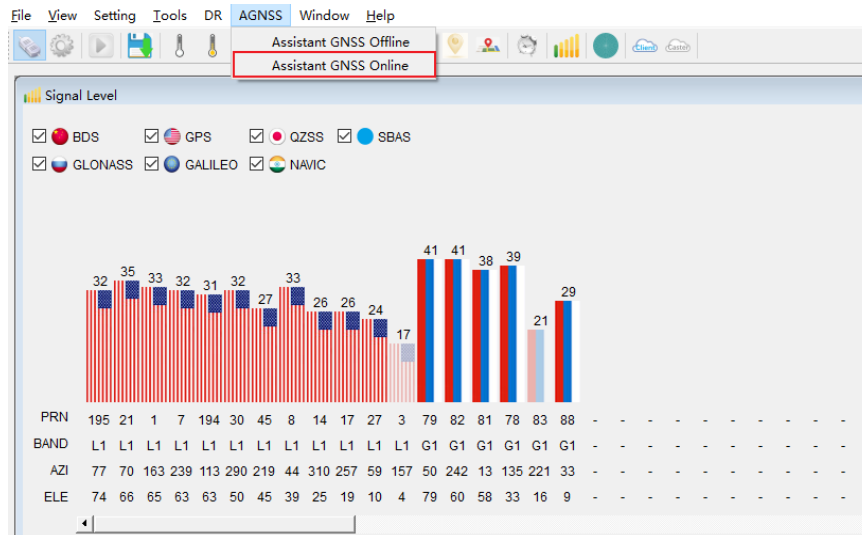


Figure 17: QGNSS Interface for Setting Host EPO

3. Configure parameters:
 - a) Check “Use Current Position” to use current position.
 - b) Check “Use Current UTC” to use current time.
 - c) Click “Transfer” to download host EPO file.

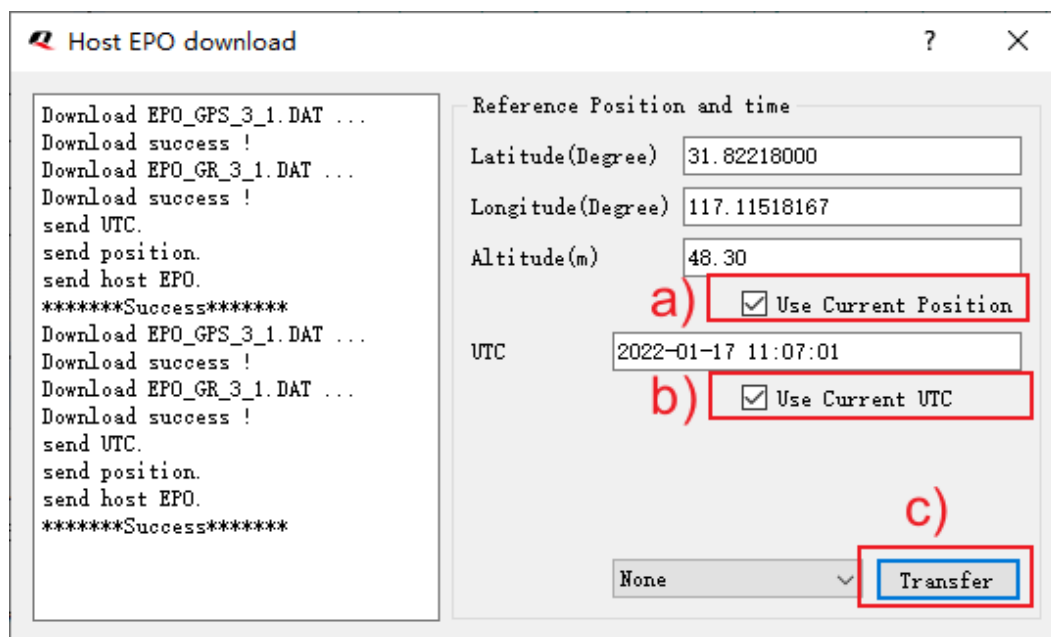


Figure 18: Download Host EPO File

5 AGNSS Implementation Example

This chapter gives examples of how to download EPO files to the module and how to use the EASY and EPOC functions.

5.1. Flash EPO Implementation

Blue bold: Sent data

Black bold: ACK information

//Host sends **\$PAIR472*3B** to erase the EPO data stored in the flash memory:

\$PAIR472*3B

//Module returns a **\$PAIR001** message:

\$PAIR001,472,0*3A

//Host sends EPO start message in hexadecimal format:

04 24 B0 04 01 00 47 F2 AA 44

//Module returns an ACK message:

04 24 E8 03 04 00 B0 04 00 00 5B AA 44

//Host sends EPO data in hexadecimal format:

04 24 B1 04 48 00 00 A9 05 01 CD 00 80 B6 73 DE FD AA 44

//Module returns an ACK message:

04 24 E8 03 04 00 B1 04 00 00 5A AA 44

.....

//Host sends EPO data in hexadecimal format:

04 24 B1 04 48 00 42 A9 05 20 AF 3F 80 B7 71 CE..... FD AA 44

//Module returns an ACK message:

04 24 E8 03 04 00 B1 04 00 00 5A AA 44

//Host sends EPO end data in hexadecimal format:

04 24 B2 04 01 00 52 E5 AA 44

//Module returns an ACK message:

04 24 E8 03 04 00 B2 04 00 00 59 AA 44

//Host queries the EPO data status stored in the module:

\$PAIR470,0*25

//Module returns **\$PAIR001** and **\$PAIR470** messages:

\$PAIR001,470,0*38

\$PAIR470,0,12,2208,86400,2208,345600,0,0,0,0*34

5.2. Host EPO Implementation

Blue bold: Sent data

Black bold: ACK information

//Host sends the power-on GNSS system command **\$PAIR002**:

\$PAIR002*38

//Module returns **\$PAIR001**:

\$PAIR001,002,0*39

//Module outputs **\$PAIR010** messages automatically:

\$PAIR010,1,-1*16

\$PAIR010,2,-1*15

//Host sends the reference UTC time command **\$PAIR590**:

\$PAIR590,2022,05,01,07,10,27*33

//Module returns a **\$PAIR001** message:

\$PAIR001,590,0*37

//Host sends the reference position information command **\$PAIR600**:

\$PAIR600,31.822203,117.115219,175.0,50.0,50.0,0.0,100.0*0F

//Module returns a **\$PAIR001** message:

\$PAIR001,600,0*3D

//Host sends EPO data:

\$PAIR471,0,1,105A906,B58000F9,248DE73,F3A00478,F2FD36BA,248DE5E,F844379B,84C21659,6F18FEA,124FCF96,103EC82,A6034274,7D15B64A,2F4E5561,24984FBC,1000001C,1100000,1D33F8D3*3F

```
//Module returns a $PAIR001 message:
```

```
$PAIR001,471,0*39
```

```
//Host sends EPO data:
```

```
$PAIR471,0,2,205A906,803F5B,24811C6,F67D3826,F4A80421,2481133,7110470,85A4863A,6F1BED  
3,1C59F758,D76D644,A60354C6,7166D3CC,2052969A,C611DBC7,1000001C,0,B41D7D61*08
```

```
//Module returns a $PAIR001 message:
```

```
$PAIR001,471,0*39
```

```
.....
```

```
//Host sends EPO data:
```

```
$PAIR471,0,E,E05A906,EC80019B,2482E2A,950E550,B55D902,248D1E5,708F5C8,83F256C6,6F175  
E1,E620A86A,7E544E8,A6039859,22C26DDF,21DDAAB5,7D32DCE7,1000001C,2210000,4941D2A5*  
7D
```

```
//Module returns a $PAIR001 message:
```

```
$PAIR001,471,0*39
```

5.3. EASY Implementation

Blue bold: Sent data

Black bold: ACK information

```
//1. Enable EASY function.
```

```
//Enable EASY:
```

```
$PAIR490,1*2A
```

```
$PAIR001,490,0*36
```

```
//Get EASY status:
```

```
$PAIR491*36
```

```
$PAIR001,491,0*37
```

```
$PAIR491,1,0*37
```

```
//2. Disable EASY function.
```

```
//Disable EASY:
```

```
$PAIR490,0*2B
```

```
$PAIR001,490,0*36
```

```
//Get EASY status:
```

```
$PAIR491*36
```

```
$PAIR001,491,0*37
```

\$PAIR491,0*2A

NOTE

Performing “Enable EASY” is optional since EASY is enabled by default.

5.4. EPOC Implementation

Blue bold: Sent data

Black bold: ACK information

//1. Enable EPOC function.

//Set EPOC constellation configuration:

\$PAIR498,1*22

\$PAIR001,498,0*3E

//Get EPOC constellation configuration:

\$PAIR508*37

\$PAIR001,508,0*36

\$PAIR508,0,1*36

//Enable EPOC:

\$PAIR496,1*2C

\$PAIR001,496,0*30

//Get EPOC status:

\$PAIR508*37

\$PAIR001,508,0*36

\$PAIR508,1,1*37

//Get EPOC prediction status:

\$PAIR509,1*2B

\$PAIR001,509,0*37

\$PAIR509,2,1,000000000000D2D2C*6A

//2. Disable EPOC function.

//Disable EPOC:

\$PAIR496,0*2D

\$PAIR001,496,0*30

```
//Get EPOC status:
```

```
$PAIR508*37
```

```
$PAIR001,508,0*36
```

```
$PAIR508,0,1*36
```

NOTE

Performing “Enable EPOC” is optional since EPOC is enabled by default.

6 Appendix A References

Table 13: Related Document

Document Name
[1] Quectel LC26G&LC26G-T&LC76G&LC86G Series Protocol Specification

Table 14: Terms and Abbreviations

Abbreviation	Description
ACK	Acknowledgement
AGNSS	Assisted GNSS (Global Navigation Satellite System)
EASY	Embedded Assist System
EPO	Extended Prediction Orbit
EPOC	Enhanced Prediction Orbit on Chip
GLONASS	Global Navigation Satellite System (Russia)
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IGSO	Inclined Geosynchronous Orbit
IMEI	International Mobile Equipment Identity
IONO	Ionospheric
MEO	Medium Earth Orbit
MNL	Middleware Navigation Library
PRN	Pseudo Random Noise
RAM	Random Access Memory

Abbreviation	Description
SV	Space Vehicle
SVID	Space Vehicle Identifier
SYSID	System Identification
TOW	Time of Week
TTFF	Time to First Fix
UART	Universal Asynchronous Receiver/Transmitter
UTC	Coordinated Universal Time
URL	Uniform Resource Locator

7 Appendix B Special Characters

Table 15: Special Characters

Special Character	Description
<CR>	Carriage return character.
<LF>	Line feed character.
<...>	Parameter name. Angle brackets do not appear in the message.
[...]	Optional field of a message. Square brackets do not appear in the message.
{...}	Repeated field of a message. Curly brackets do not appear in the message.
<u>Underline</u>	Default setting of a parameter.