GNSS Flash EPO

Application Note

GNSS Module Series

Rev. GNSS_Flash_EPO_Application_Note_V1.0

Date: 2017-04-28

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

Quectel Wireless Solutions Co., Ltd.
Office 501, Building 13, No.99, Tianzhou Road, Shanghai, China, 200233
Tel: +86 21 5108 6236
Email: info@quectel.com

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

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/techsupport.aspx
Or email to: Support@quectel.com

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT

About the Document

History

Revision	Date	Author	Description
1.0	2017-04-28	Matt YE	Initial

Contents

Table Index

Figure Index

1 Introduction

EPO (Extended Prediction Orbit) is a free service provided by MediaTek, which can achieve fast TTFF and improve accuracy in weak signal conditions. This document mainly describes EPO file format and EPO data transfer protocol for EPO Management Tool development.

This document is applicable to the following Quectel modules:

- L26
- L70
- L76
- L76B
- L76-L
- L80
- L86

2 Download EPO Files

2.1. Types of EPO Files

Table 1: Types of EPO Files

EPO Type	GNSS Type	Description
Unified QEPO URL	GPS only	6-hour prediction orbit (ephemeris) Single file name, always get the latest QEPO file The server auto selects the EPO files valid at the current time, no need to specify the file number
Unified QEPO URL	GPS+GLONASS	6-hour prediction orbit (ephemeris) Single file name, always get the latest QEPO file The server auto selects the EPO files valid at the current time, no need to specify the file number
EPO	GPS only	3-30 days prediction orbit (ephemeris)
EPO	GPS+GLONASS	3-30 days prediction orbit (ephemeris)

2.2. Download URL of EPO Files

Table 2: Download URL of EPO Files

EPO Type	GNSS Type	EPO File URL	File Name
Unified QEPO URL	GPS only	http://wpepodownload.mediatek.com/QGPS.DAT?*vendorinfo*	Single name: QGPS.DAT
Unified QEPO URL	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 ~ 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 ~ 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 separated by "*&*".
- The value of "*vendor*" and "*project*" (*AAA, BBB* in the example) are issued by Quectel, please contact Quectel technical support to get the value.
- The value of "*device_id*" (*CCC* in the example) is assigned by the vendor, each device must have its own unique ID. For example: *CCC=XXX_YYY*. *XXX*: need to contact Quectel technical support to get the value. *YYY*: can be assigned by customers, but must make sure it is unique, such as IMEI.

NOTE

Slices of 30 days EPO:
3 day for each file and up to 30 days.
_1 for day 1~3,
_2 for day 4~6,
...
_10 for day 28~30.

2.3. Recommended Download Strategy of EPO Files

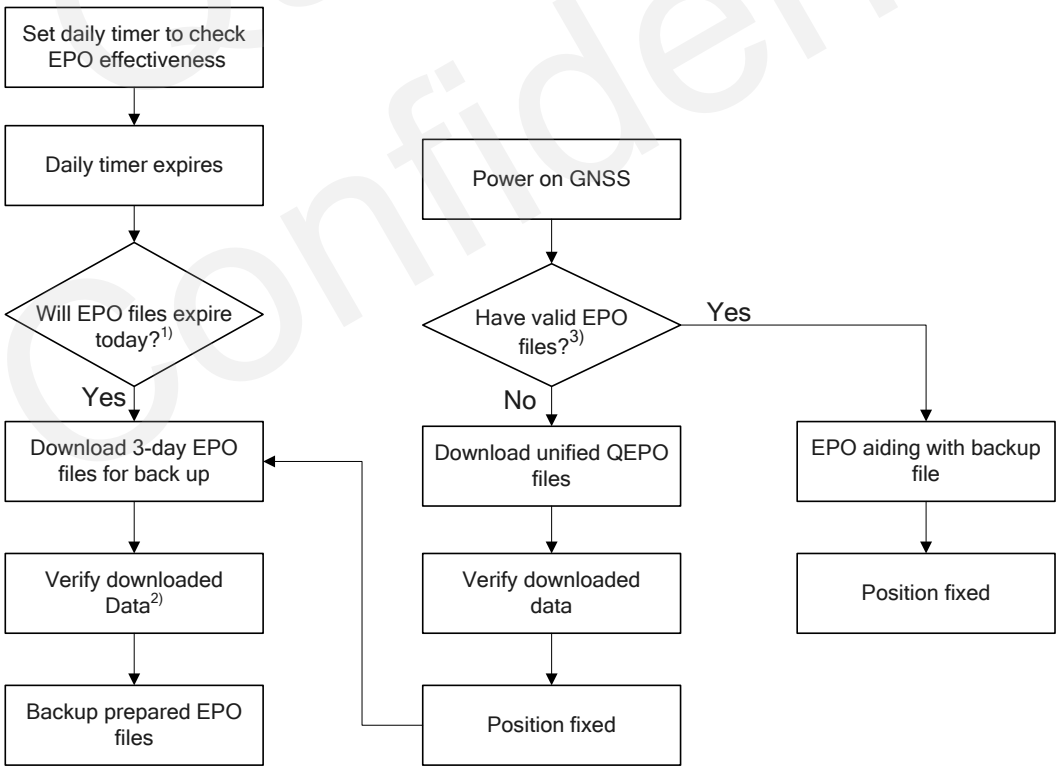

Figure 1: Recommended Download Strategy of EPO Files

NOTES

1. [1)] Customers must know the current UTC time to download the current valid QEPO files.
2. [2)] Download MD5 checksum file by replacing file extension “DAT” with “MD5” to get the MD5 file for checking data correctness.
3. [3)] If the device is powered off for a long time, EPO files stored in flash will be expired.

2.4. The Validity Period of EPO Files

If EPO validity period is less than current UTC time, the EPO file will expire. The EPO validity period can be obtained from the last segment of the EPO file. Please refer to ***Figure 2*** 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;
}
```

2.5. One SV Format of EPO Files

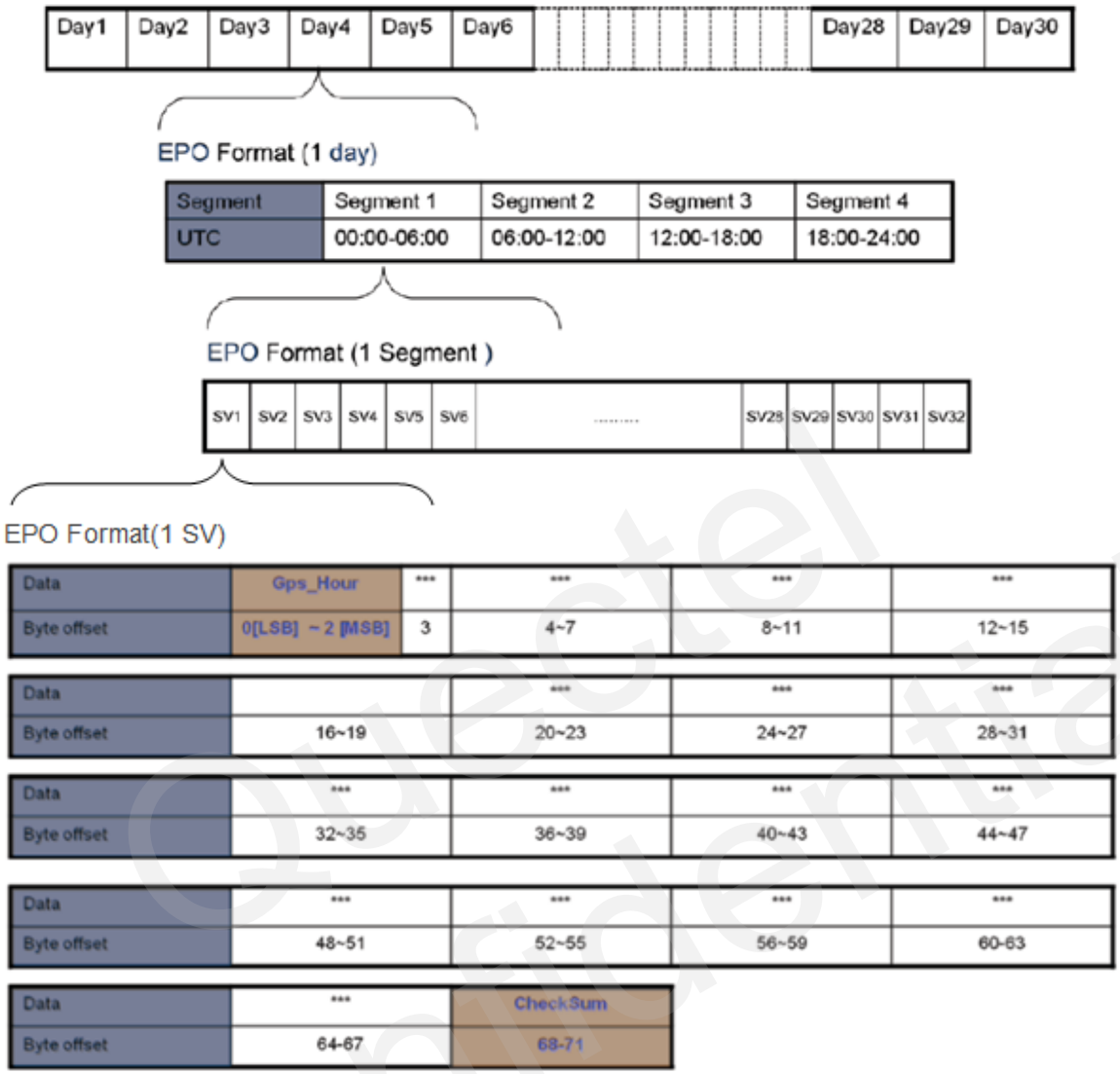

Figure 2: One SV Format of EPO Files

- Gps_Secs = Gps_Hour * 3600
- GPS_Week Number = Gps_Secs / 604800
- GPS TOW = Gps_Secs % 604800

GPS week number starts from 0 at approximately midnight on the evening of 05 January, 1980 or morning of 06 January, 1980 and is incremented by 1 each week.

3 Format of EPO Files

3.1. GPS Only EPO File Format

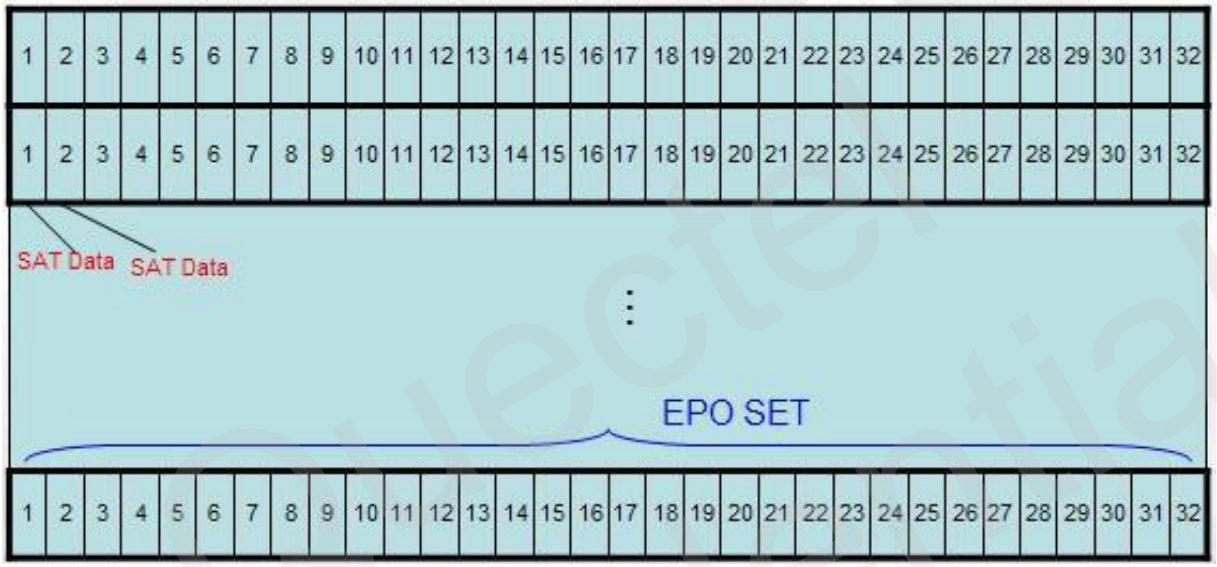

Figure 3: GPS Only EPO File Format

The basic unit of an EPO file is SAT Data and the data size of a SAT Data is 72 bytes. One EPO SET contains 32 SAT Data, so the data size for an EPO SET is 2304 bytes. Each EPO file contains several EPO SETs. 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.

3.2. GPS+GLONASS EPO File Format

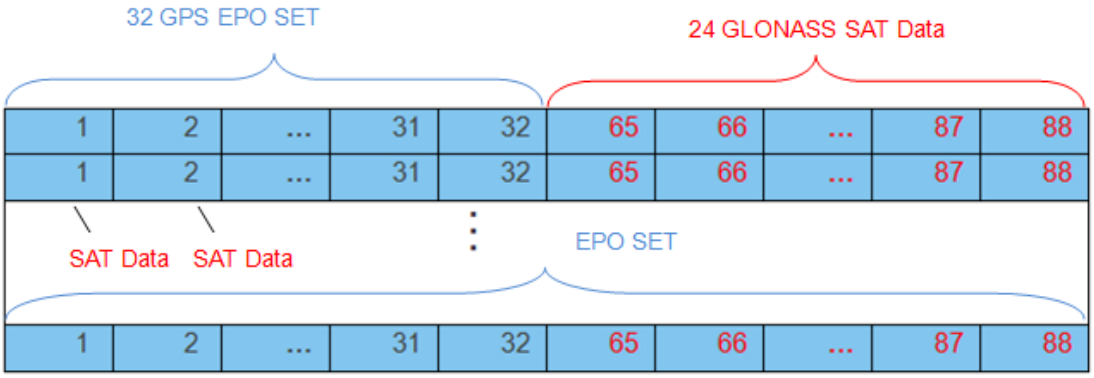

Figure 4: GPS+GLONASS EPO File Format

The basic unit of an EPO file is SAT Data, and the data 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.

4 MTK Binary Protocol

Table 3: MTK Binary Protocol Format

Preamble	Length	Command ID	Data	Checksum	End Word
0x04 0x24				0xHH	0x0D 0x0A
2 Bytes	2 Bytes	2 Bytes	Variable	1 Bytes	2 Bytes

Table 4: Description of MTK Binary Protocol Parameters

Parameters	Length (Byte)	Description
Preamble	2	0x2404
Length	2	Total number in the packet from Preamble to End Word. Maximum packet size: 256 bytes Use little endian Use one byte alignment
Command ID	2	0~999: conform to PMTK ASCII protocol 1000~65535: designated for MTK binary protocol
Data	Variable	Data to be transferred
Checksum	1	The checksum is the 8-bit exclusive OR of all bytes in the packet between (but not including) the “Preamble” and the “Checksum”
End Word	2	0x0A0D

Please refer to ***Chapter 8.4*** for the samples of MTK binary packet types.

5 EPO Binary Packet Format

EPO binary packet is named as MTK_BIN_EPO packet (packet type 723) for convenience.

Table 5: EPO Binary Packet Format

Preamble		Length	Command ID	Data				Checksum	End Word	
0x04	0x24	0x00E3	0x02D3	EPO SEQ	SAT Data	SAT Data	SAT Data	0xHH	0x0D	0x0A
2 Bytes		2 Bytes	2 Bytes	2 Bytes	72 Bytes	72 Bytes	72 Bytes	1 Bytes	2 Bytes	

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

Sometimes, there is no enough EPO data to full fill the three SAT Data fields. Some of the three fields can be left as blank, that is, to be filled with 0x00. A 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 6: MTK_BIN_EPO (Contains 2 SAT Data)

Preamble		Length	Command ID	Data				Checksum	End Word	
0x04	0x24	0x00E3	0x02D3	EPO SEQ	SAT Data	SAT Data	0x00	0xHH	0x0D	0x0A
2 Bytes		2 Bytes	2 Bytes	2 Bytes	72 Bytes	72 Bytes	72 Bytes	1 Bytes	2 Bytes	

Table 7: MTK_BIN_EPO (Contains 1 SAT Data)

Preamble		Length	Command ID	Data				Checksum	End Word	
0x04	0x24	0x00E3	0x02D3	EPO SEQ	SAT Data	0x00	0x00	0xHH	0x0D	0x0A
2 Bytes		2 Bytes	2 Bytes	2 Bytes	72 Bytes	72 Bytes	72 Bytes	1 Bytes	2 Bytes	

Table 8: MTK_BIN_EPO (Contains no SAT Data)

Preamble		Length	Command ID	Data				Checksum	End Word	
0x04	0x24	0x00E3	0x02D3	EPO SEQ	0x00	0x00	0x00	0xHH	0x0D	0x0A
2 Bytes		2 Bytes	2 Bytes	2 Bytes	72 Bytes	72 Bytes	72 Bytes	1 Bytes	2 Bytes	

MTK GNSS receiver will return an acknowledge packet for each received MTK_BIN_EPO. The acknowledge packet is named as MTK_BIN_ACK_EPO (packet type 2) for convenience.

Table 9: Acknowledge Packet of MTK _BIN_EPO

Preamble		Length	Command ID	Data		Checksum	End Word	
0x04	0x24	0x000C	0x0002	EPO SEQ	Result	0xHH	0x0D	0x0A
2 Bytes		2 Bytes	2 Bytes	2 Bytes	1 Bytes	1 Bytes	2 Bytes	

- **EPO SEQ:** sequence number to indicate the corresponding received MTK_BIN_EPO
- **Result:** '0'-- the received MTK_BIN_EPO is invalid, which means it is failed to receive the packet.
  '1'-- the received MTK_BIN_EPO is valid, which means the packet is successfully received.

6 EPO Data Transfer Protocol

EPO data are packed in MTK_BIN_EPO packets by using MTK binary protocol, and then transferred from EPO Management Tool to EPO Transfer Agent in a MTK GNSS receiver. At the beginning of the transfer procedure, EPO Management Tool splits the EPO files and encapsulates them into several MTK_BIN_EPO packets, then assigns 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. EPO Management Tool and EPO Transfer Agent follow the “EPO Data Transfer Protocol” to transfer EPO data into MTK GNSS receiver.

6.1. EPO Data Transfer Procedure

1. EPO Management Tool: Send one MTK_BIN_EPO packet which contains 1~3 SAT Data to MTK GNSS receiver. The sequence number in the packet starts from zero and will add one for each of the following MTK_BIN_EPO packets.

2. EPO Management Tool: Wait for the MTK_BIN_ACK_EPO which has the same sequence number with the transmitted MTK_BIN_EPO.

3. EPO Transfer Agent in MTK GNSS receiver: Receive MTK_BIN_EPO packet from EPO Management Tool, then verify the validity of EPO data in the packet. If it is a correct packet, EPO Transfer Agent will return a MTK_BIN_ACK_EPO packet to indicate success; otherwise, it will return a MTK_ BIN_ACK_EPO packet to indicate failure.

4. EPO Transfer Agent in MTK GNSS receiver: Wait for the next EPO packet.

5. EPO Management Tool: Receive MTK_ BIN_ACK_EPO packet. If the acknowledge character indicates success, Management Tool prepares to send the next MTK_BIN_EPO packet; otherwise, any error occurs and it needs to exit from the procedure.

6. Repeat Steps 1~5 until all the EPO data are transferred.

7. EPO Management Tool: Send a final MTK_BIN_EPO packet which contains sequence number of 0xFFFF to indicate the finish of the procedure. The 3 SAT Data fields in the final MTK_BIN_EPO packet are all blank.

NOTES

For GPS+GLONASS EPO files, please send all the SAT Data of one SET before sending another SET. Otherwise GNSS receiver may receive data incorrectly.

- Right way: Send SAT1~SAT32 of SET1 → Send SAT33~SAT56 of SET1 → Send SAT1~SAT32 of SET2 → Send SAT33~SAT56 of SET2 → Send SAT1~SAT32 of SET3…
- Wrong way: Send SAT1~SAT32 of SET1 → Send SAT1~SAT32 of SET2 → Send SAT1~SAT32 of SET3... → Send SAT33~SAT56 of SET1 → Send SAT33~SAT56 of SET2...

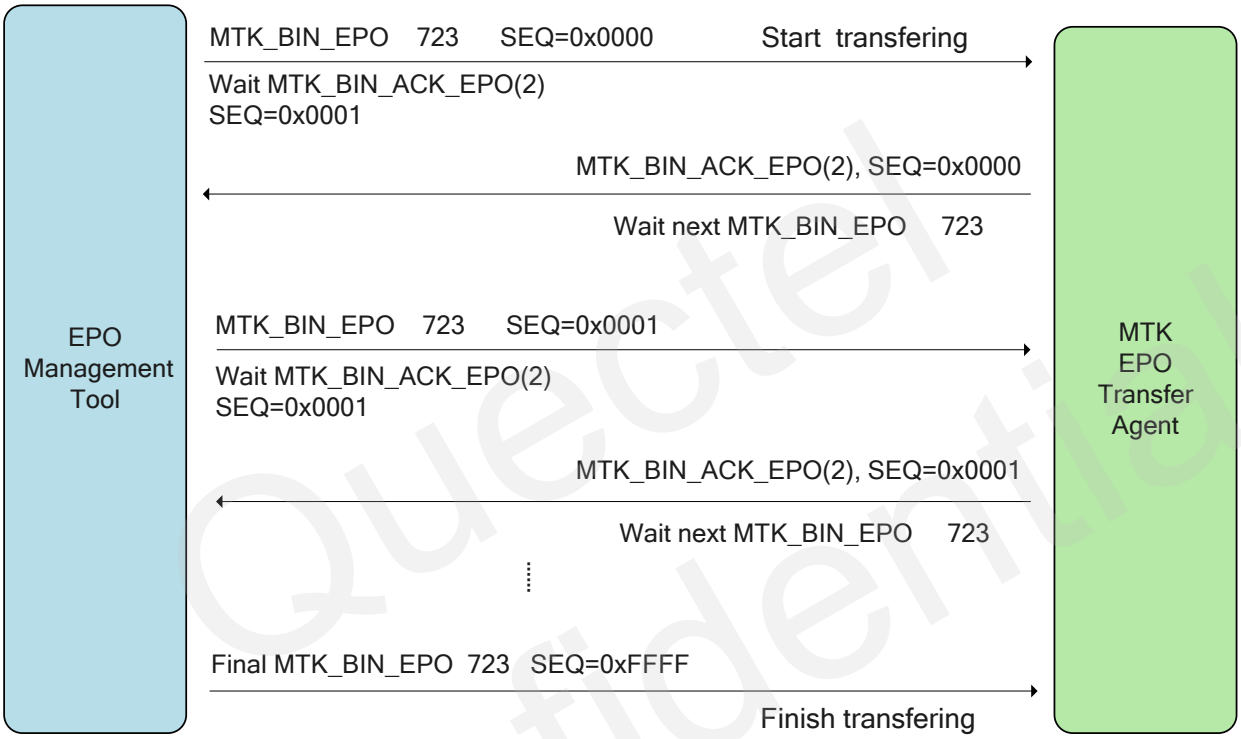

Figure 5: EPO Data Transfer Procedure

6.2. Error Handling

If there is any problem occurs in the transfer procedure, please stop the process and restart the transfer procedure again. Every time when the procedure starts, the EPO sequence number should be reset to zero to indicate the GNSS receiver that a new transfer procedure has begun. Then the GNSS receiver prepares for the new procedure.

The interval of time between two continuous MTK_BIN_EPO packets shall not be longer than 10 seconds. Otherwise, the GNSS receiver will consider that there is any problem occurs and then terminate the procedure.

6.3. Check EPO Data in GNSS Chip

It needs to ensure that the EPO data were successfully updated into GNSS chip. After finishing the EPO transfer procedure, please make sure the current UART packet format is NMEA mode. Then customers can execute "**$PMTK607*33<CR><LF>**" PMTK_Q_EPO_INFO command to query the EPO data status. The GNSS chip will return PMTK_DT_EPO_INFO as below:

*$PMTK707,56,1468,172800,1470,151200,1468,259200,1468,259200*1F<CR><LF>*

This packet shows the information of EPO data stored in the GNSS chip. For 14-day EPO files, the first argument following PMTK707 will be 56; for 7-day EPO files, it will be 28; 1468 and 172800 means the starting of GPS time (GPS week, GPS TOW) for the EPO data; 1470 and 151200 means the ending of GPS time (GPS week, GPS TOW) for the EPO data. GPS week and GPS TOW need to be converted into UTC time format to ensure that the format of EPO data stored in the flash matches that of the EPO file.

The maximum period of EPO data can be stored in MT3333 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 in MT3333. If a 30-day GPS+GLONASS EPO file is sent, only the first 7 days of EPO data will be stored in MT3333.

Please refer to ***Chapter 8*** for the details of PMTK_Q_EPO_INFO and PMTK_DT_EPO_INFO.

7 Pseudo Codes 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 MTK binary packet, the protocol should be changed to MTK binary protocol.
//Before starting EPO data transfer procedure, change the UART protocol setting by PMTK command 253. Please refer to ***Chapter 8.1*** for the details of PMTK command 253.
//The SendPmtkCmd function must be implemented by the programmer.
//It is recommended to explicitly specify a baudrate when changing UART packet protocol, for example, $PMTK253,1,115200*00<CR><LF>.

SendPmtkCmd ($PMTK253,1,0*37<CR><LF>);

//Now the data transferred via the UART port will be regarded as MTK binary packet format. Please create a thread to transmit/receive MTK 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 MTK 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 number 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 a 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 acknowledge 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 acknowledge. The GNSS receiver will return a MTK_BIN_ACK_EPO acknowledge packet after receiving and process previous MTK_BIN_EPO packet. Please refer to Chapter 8.2 for the details.
//If the acknowledge 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 baudrate 115200 for the communication UART. Please refer to Chapter 8.2 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);
```

8 EPO Related PMTK Commands

8.1. Packet Type: 253 PMTK_SET_OUTPUT_FMT

This command is used to set data output format and baudrate for the current port.

Table 10: 253 PMTK_SET_OUTPUT_FMT

Example: Switch the UART protocol format to binary mode, and use default baudrate 115200bps: $PMTK253,1,0*37<CR><LF> Switch the UART protocol format to NMEA mode, and use baudrate 9600bps: $PMTK253,0,9600*09<CR><LF>			
Name	**Unit**	**Default**	**Description**
Flag	--	0	0 - NMEA mode 1 - Binary mode
Baudrate	bps	115200	Baudrate for the new output mode 0: use default baudrate (not recommended) UART1: default baudrate will be the value set in “Data Port UART1 Baudrate” of Corebuilder. UART0: default baudrate will be the value set in “NMEA Baudrate” of Corebuilder It is highly recommended to specify an explicit baudrate value, other possible values will be 4800, 9600, 14400, 19200, 38400, 57600 and 115200.

NOTES

1. When switching from binary mode to NMEA mode, a binary acknowledge will be received (Packet Type 1) after the command is executed. Please refer to ***Chapter 8.4*** for the details of acknowledge packet.
2. When switching from NMEA mode to binary mode, no acknowledge will be sent.

8.2. Packet Type: 607 PMTK_Q_EPO_INFO

This command is used to query the EPO data status stored in the GNSS chip.

Table 11: 607 PMTK_Q_EPO_INFO

Example: $PMTK607*33<CR><LF> Response: PMTK_DT_EPO_INFO			
Name	**Unit**	**Default**	**Description**
--	--	--	--

8.3. Packet Type: 707 PMTK_DT_EPO_INFO

This response packet contains EPO data status stored in GNSS chip.

Table 12: 707 PMTK_DT_EPO_INFO

Example: $PMTK707,56,1468,172800,1470,151200,1468,259200,1468,259200*1F<CR><LF>			
Name	**Unit**	**Default**	**Description**
Set	--	--	Total number sets of EPO data stored in GNSS chip
FWN	--	--	GPS week number of the first set of EPO data stored in GNSS chip
FTOW	--	--	GPS week TOW of the first set of EPO data stored in GNSS chip
LWN	--	--	GPS week number of the last set of EPO data stored in GNSS chip
LTOW	--	--	GPS week TOW of the last set of EPO data stored in GNSS chip
FCWN	--	--	GPS week number of the first set of EPO data that are currently used
FCTOW	--	--	GPS week TOW of the first set of EPO data that are currently used
LCWN	--	--	GPS week number of the last set of EPO data that are currently used
LCTOW	--	--	GPS week TOW of the last set of EPO data that are currently used

8.4. MTK Binary Packet Types

8.4.1. Acknowledge Packet (Packet Type 1)

This packet is usually returned after receiving a MTK binary packet.

Table 13: Acknowledge Packet (Packet Type 1)

Example: Receive a valid MTK binary packet and return a success flag. 0x04 0x24 0x0C 0x00 0x01 0x00 0xFD 0x00 0x03 0xF3 0x0D 0x0A			
Parameter	**Length (Bytes)**	**Example**	**Description**
Preamble	2	0x2404	
Length	2	0x000C	Packet length: 12 bytes
Command ID	2	0x0001	Acknowledge packet
Data	2	0x00FD	The acknowledge is responding to packet type 253
Flag	1	0x03	0x00, 0x01: invalid packet 0x02: failure Flag: 0x03: success
Checksum	1	0xF3	The checksum is the 8-bit exclusive OR of all bytes in the packet between (but not including) the “Preamble” and the “Checksum”
End Word	2	0x0A0D	

8.4.2. EPO Acknowledge Packet (Packet Type 2)

This packet is usually returned after receiving an EPO binary packet.

Table 14: EPO Acknowledge Packet (Packet Type 2) Example 1

Example: Receive a valid EPO packet for sequence number 0x56: 0x04 0x24 0x0C 0x00 0x02 0x00 0x56 0x00 0x01 0x59 0x0D 0x0A			
Parameter	**Length (Bytes)**	**Example**	**Description**
Preamble	2	0x2404	
Length	2	0x000C	Packet length: 12 bytes

Command ID	2	0x0002	Acknowledge packet
Data	3	0x056 0x00 0x01	Valid EPO packet for sequence: 0x56
Checksum	1	0x59	The checksum is the 8-bit exclusive OR of all bytes in the packet between (but not including) the “Preamble” and the “Checksum”
End Word	2	0x0A0D	

Table 15: EPO Acknowledge Packet (Packet Type 2) Example 2

Example: Receive an invalid EPO packet for sequence number 0x56: 0x04 0x24 0x0C 0x00 0x02 0x00 0x56 0x00 0x00 0x58 0x0D 0x0A			
Parameter	**Length (Bytes)**	**Example**	**Description**
Preamble	2	0x2404	
Length	2	0x000C	Packet length: 12 bytes
Command ID	2	0x0002	Acknowledge packet
Data	3	0x056 0x00 0x00	Invalid EPO packet for sequence: 0x56
Checksum	1	0x58	The checksum is the 8-bit exclusive OR of all bytes in the packet between (but not including) the “Preamble” and the “Checksum”
End Word	2	0x0A0D	

8.4.3. Change UART Format Packet (Packet Type 253)

It is used to change UART communication protocol and baudrate.

Table 16: Change UART Format Packet (Packet Type 253) Example 1

Example: Change UART to PMTK protocol and baudrate is 115200: 0x04 0x24 0x0E 0x00 0xFD 0x00 0x00 0x00 0xC2 0x01 0x00 0x30 0x0D 0x0A			
Parameter	**Length (Bytes)**	**Example**	**Description**
Preamble	2	0x2404	

Length	2	0x000E	Packet length: 14 bytes
Command ID	2	0x00FD	Change UART packet protocol
Protocol	1	0x00	PMTK protocol
UART Baudrate	4	0x00 0xC2 0x01 0x00	UART baudrate: 115200bps
Checksum	1	0x30	The checksum is the 8-bit exclusive OR of all bytes in the packet between (but not including) the "Preamble" and the "Checksum"
End Word	2	0x0A0D	

Table 17: Change UART Format Packet (Packet Type 253) Example 2

Example: Change UART to PMTK protocol and use default baudrate: 0x04 0x24 0x0E 0x00 0xFD 0x00 0x00 0x00 0x00 0x00 0x00 0xF3 0x0D 0x0A			
Parameter	**Length (Bytes)**	**Example**	**Description**
Preamble	2	0x2404	
Length	2	0x000E	Packet length: 14 bytes
Command ID	2	0x00FD	Change UART packet protocol
Protocol	1	0x00	PMTK protocol
UART Baudrate	4	0x00 0x00 0x00 0x00	UART baudrate: default value
Checksum	1	0xF3	
End Word	2	0x0A0D	

9 Appendix A References

Table 18: Almanac, Ephemeris & EPO

Type	Source	Valid	Effect
Almanac	Satellites	Several weeks	Search satellites
Ephemeris	Satellites	< 4 hours	Positioning
MediaTek EPO	MediaTek server	6 hours to 30 days	Search satellites & positioning

Table 19: Terms and Abbreviations

Abbreviation	Description
ASCII	American Standard Code for Information Interchange,
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
UART	Universal Asynchronous Receiver/Transmitter
UTC	Coordinated Universal Time