Telit Jupiter SGEE-EPO Application Note

80433NT11332A r2 – 2015-03-06

APPLICABILITY TABLE

PRODUCT
SL871
SE868-A
SL869-V2
SC872-A

Notice

While reasonable efforts have been made to assure the accuracy of this document, Telit assumes no liability resulting from any inaccuracies or omissions in this document, or from use of the information obtained herein. The information in this document has been carefully checked and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies or omissions. Telit reserves the right to make changes to any products described herein and reserves the right to revise this document and to make changes from time to time in content hereof with no obligation to notify any person of revisions or changes. Telit does not assume any liability arising out of the application or use of any product, software, or circuit described herein; neither does it convey license under its patent rights or the rights of others.

It is possible that this publication may contain references to, or information about Telit products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that Telit intends to announce such Telit products, programming, or services in your country.

Copyrights

This instruction manual and the Telit products described in this instruction manual may be, include or describe copyrighted Telit material, such as computer programs stored in semiconductor memories or other media. Laws in the Italy and other countries preserve for Telit and its licensors certain exclusive rights for copyrighted material, including the exclusive right to copy, reproduce in any form, distribute and make derivative works of the copyrighted material. Accordingly, any copyrighted material of Telit and its licensors contained herein or in the Telit products described in this instruction manual may not be copied, reproduced, distributed, merged or modified in any manner without the express written permission of Telit. Furthermore, the purchase of Telit products shall not be deemed to grant either directly or by implication, estoppels, or otherwise, any license under the copyrights, patents or patent applications of Telit, as arises by operation of law in the sale of a product.

Computer Software Copyrights

The Telit and 3rd Party supplied Software (SW) products described in this instruction manual may include copyrighted Telit and other 3rd Party supplied computer programs stored in semiconductor memories or other media. Laws in the Italy and other countries preserve for Telit and other 3rd Party supplied SW certain exclusive rights for copyrighted computer programs, including the exclusive right to copy or reproduce in any form the copyrighted computer program. Accordingly, any copyrighted Telit or other 3rd Party supplied SW computer programs contained in the Telit products described in this instruction manual may not be copied (reverse engineered) or reproduced in any manner without the express written permission of Telit or the 3rd Party SW supplier. Furthermore, the purchase of Telit products shall not be deemed to grant either directly or by implication, estoppels, or otherwise, any license under the copyrights, patents or patent applications of Telit or other 3rd Party supplied SW, except for the normal non-exclusive, royalty free license to use that arises by operation of law in the sale of a product.

Usage and Disclosure Restrictions

License Agreements

The software described in this document is the property of Telit and its licensors. It is furnished by express license agreement only and may be used only in accordance with the terms of such an agreement.

Copyrighted Materials

Software and documentation are copyrighted materials. Making unauthorized copies is prohibited by law. No part of the software or documentation may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, without prior written permission of Telit

High Risk Materials

Components, units, or third-party products used in the product described herein are NOT fault-tolerant and are NOT designed, manufactured, or intended for use as on-line control equipment in the following hazardous environments requiring fail-safe controls: the operation of Nuclear Facilities, Aircraft Navigation or Aircraft Communication Systems, Air Traffic Control, Life Support, or Weapons Systems (High Risk Activities"). Telit and its supplier(s) specifically disclaim any expressed or implied warranty of fitness for such High Risk Activities.

Trademarks

TELIT and the Stylized T Logo are registered in Trademark Office. All other product or service names are the property of their respective owners.

Contents

Table of Figures

1. Introduction

1.1. Scope

This document describes the GNSS feature known as Extended Ephemeris (EE), with particular focus on the application support of Telit's MT3333 based GNSS modules with the context of Server-Generated Extended Ephemeris (SGEE) feature.

- SL871
- SE868-A
- SL869-V2
- SC872-A

To be specific, the SGEE feature that is found in the above family modules is supplied by Mediatek® in their Extended Prediction Orbit (EPO) data. With reference to SGEE feature in those modules, Telit SGEE-EPO is hereinafter referring to the Mediatek's EPO.

1.2. Audience

This document is intended for software engineers and developers who are interested in implementing the SGEE feature within their devices.

- SL871
- SE868-A
- SL869-V2
- SC872-A

It is also useful for those who are interested in learning more about the Assisted GNSS capability that is found in the Telit's GNSS module families that support the features.

1.3. Contact Information, Support

For general contact, technical support, to report documentation errors and to order manuals, contact Telit Technical Support Center (TTSC) at:

TS-EMEA@telit.com

TS-AMERICAS@telit.com

TS-LATINAMERICA@telit.com

TS-APAC@telit.com

Alternatively, use:

http://www.telit.com/en/products/technical-support-center/contact.php

For detailed information about where you can buy the Telit modules or for recommendations on accessories and components visit:

http://www.telit.com

To register for product news and announcements or for product questions contact Telit Technical Support Center (TTSC).

Our aim is to make this guide as helpful as possible. Keep us informed of your comments and suggestions for improvements.

Telit appreciates feedback from the users of our information.

1.4. Document Organization

This document contains the following chapters (sample):

"Chapter 1: "Introduction" provides a scope for this document, target audience, contact and support information, and text conventions.

"Chapter 2: "Overview" gives an overview of the Telit SGEE feature.

"Chapter 3: "Considerations for Using SGEE" discusses aspects of the customer application to be considered when assessing the Telit SGEE feature for inclusion in the customer's end product.

"Chapter 4: "SGEE Detailed Description" provides details regarding the SGEE feature.

1.5. Text Conventions

Danger – This information MUST be followed or catastrophic equipment failure or bodily injury may occur.

Caution or Warning – Alerts the user to important points about integrating the module, if these points are not followed, the module and end user equipment may fail or malfunction.

Tip or Information – Provides advice and suggestions that may be useful when integrating the module.

All dates are in ISO 8601 format, i.e. YYYY-MM-DD.

1.6. Related Documents

SL871 and SL869-V2 Families Software User Guide

2. Overview

Extended Ephemeris (EE) is referring to a technology developed by GNSS chip vendors to reduce Time To First Fix (TTFF), particularly in challenging RF environments affecting satellite signal reception.

2.1. Background

Telit and GNSS device vendors collaborate to furnish to customers who use Telit's GNSS modules to achieve the goal of improving TTFF performance.Background

In order for a GNSS receiver to provide a navigation solution, it must have satellite orbits data so that it can accurately determine the expected location of each visible GPS satellite. If the receiver does not have this data, or if the data it has is too old, it must collect the data from the satellite after its signal has been acquired. Even under the best circumstances, where initial position and time are known, this process can take up to 35 seconds, which in turn can mean a Time To First Fix (TTFF) of 35 seconds or more. In typical circumstances the TTFF can be even longer, as the receiver must collect ephemeris for three or four satellites in order to navigate.

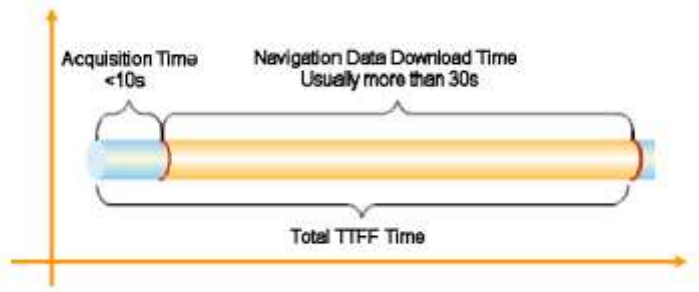


Figure 1: Collecting sat orbits data and TTFF

SGEE-EPO technology provides pre-calculated synthetic Extended Ephemeris (EE) data to the GNSS receiver to reduce TTFF. The presence of Extended Ephemeris allows the receiver to substitute it for

broadcast ephemeris downloaded from the GNSS satellite and to skip the navigation data collection requirement.

2.2. Summary of SGEE-EPO

SGEE-EPO refers to the calculation of synthetic ephemeris data, also known as Extended Ephemeris, delivery through various path, and use of the synthetic ephemeris data by the GNSS module in getting position solution in good TTFF performance.

Telit SGEE-EPO is calculated, at a remote source (i.e. at a vendor's AGPS server) – other than the GNSS module itself, by applying predictive models of satellite motion to actual broadcast ephemeris data, and is structured as a set of data segments on combination of satellite constellations; they can be GPS only, or GPS + GLONASS.

For Telit SGEE-EPO that is utilized in the GNSS modules described in this document, each segment contains synthetic ephemeris for a 6-hour time period. Thus, the extended ephemeris data for one day consists of 4 segments.

2.2.1. SGEE-EPO at Server

The SGEE-EPO server obtains ephemeris information from world-wide GNSS tracking stations and calculates EE data for all healthy GNSS satellites and store the data at on-demand basis.

The SGEE-EPO data has following characteristics:

- The EE data calculation and update is done at a predetermined schedule that is adequate to provide valid data at any time.
- The server calculates a sufficient number of data blocks to produce EE and the data is stored in files which can be accessed over a network connection.
- The EE data blocks are packaged into files according to on different prediction intervals. Each of the files represents a different prediction interval such as one-day, three-day, seven day, 14 days, etc which is the applicable length of time (in days) over which the EE within the file can be used by the receiver.

2.2.2. Transfer SGEE-EPO to GNSS Device

In order to use SGEE-EPO data, an OEM device (either a hosting device or functions like a host), must be capable of establishing an HTTP connection and retrieving data files from a SGEE-EPO server.

The OEM device must also transfer the file to the OEM GNSS receiver module (operating as a client) over the communication link between the host and the GNSS device (i.e. a serial port).

2.2.3. Conditions of Using SGEE-EPO

The use of SGEE-EPO by the receiver module requires that precise time is first derived and verified from live satellites. Thus the use of EE to reduce TTFF is most effective for applications in which the time on a GPS receiver module is maintained in the Hibernate state. Once the module has navigated,

time will have been verified and updated in the Real-Time Clock (RTC), where it is maintained on the module. Thus at start-up the module leaves its initial state with an accurate initial time, as well as an initial estimate of position from memory. The initial time eliminates the need to derive time from a live satellite in order to navigate, and if broadcast ephemeris has expired or is invalid (i.e. Warm Start conditions), the module uses EE instead. Under Warm Start conditions the typical TTFF will be approximately 10 seconds or less in good RF environments (minimal blockage and higher signal-to-noise ratios).

If however the receiver module starts up from a powered off state, i.e. Cold Start conditions, the receiver module does not have an estimate of time. Therefore it must derive and verify date and time and download ephemeris from the satellites, which can take up to 35 seconds, even in good conditions. Thus the use of SGEE-EPO is generally not very effective in reducing Cold Start TTFF.

3. Telit SGEE-EPO Detailed Description

The subsections below describe Telit SGEE-EPO feature in more detail.

3.1. Infrastructure and Overview

The overall data flow for Telit SGEE-EPO is depicted in Figure 3 below.

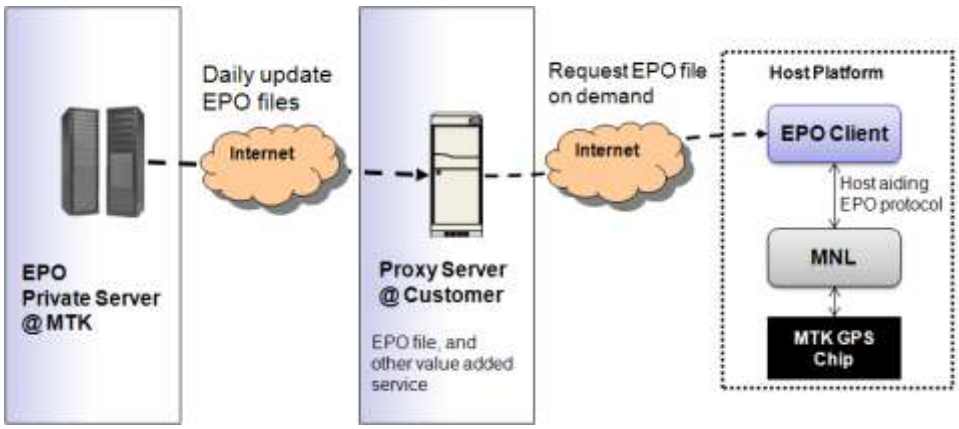


Figure 2: Infrastructure of SGEE-EPO

Refer to the above illustration, the overview of Telit SGEE-EPO workflow can be summarized in the following sections.

3.1.1. Mediatek EPO Server

The Mediatek's Extended Ephemeris (EE) data is generated at an EPO private server using one of their proprietary predictive models with respect to the predictive interval (i.e. 30 day EE). The time frame for each new EE data is each UTC day.

For a 30-day EE interval, a number of EE segments covering the next 30 days are generated. The EE data is compressed, formatted and published as a set of EE files. All of the satellites most recent

ephemerides are used to calculate the EE data. Thus all of the EE within the files is new (less than one hour old) when they are published at the beginning of the UTC day.

3.1.2. Extended Ephemeris Files

Currently the SGEE-EPO data file that is generated at the Mediatek server, publishes an EPO file for a 30-day prediction interval. However, MT3333 does not support 30-day EPO files and has a maximum storage capacity for 14 days.

For GPS constellation (32 GPS satellites), the EPO file size is 2304 bytes per segment; each segment covers a six-hour period of a day. The following table illustrates the SGEE-EPO data validity period and the SGEE-EPO file size.

SGEE-EPO Validity Period	SGEE-EPO File Size (bytes)
One six-hour	2304 (2KB+)
One day	9216 (9KB)
Three day	27648 (27KB)
Seven day	64512 (63KB)
14 day	129024 (126KB)
30 day	276480 (270KB)

This information provides the essential data on some fundamental considerations of use of SGEE-EPO in positioning applications, such as the frequency of download from the SGEE-EPO server to GNSS module, the data file size of the download, and the memory requirement (storage type and the size) in the custom device.

3.1.3. Telit SGEE-EPO Server

The Telit SGEE-EPO server retrieves EPO data files from the Mediatek's EPO server several times a day to ensure that the most recently updated files are available to Telit customers. Although Telit provides no specifications with regard to availability and quality of service, it maintains redundant servers for added reliability of service.

The server URL is http://epo.telit.com. A username/password is required for access, which is provided by Telit when the customer has signed a Telit Service Agreement.

3.1.4. Customer Mirror Server

The customer mirror server retrieves SGEE-EPO files from the Telit SGEE-EPO server and makes them available for the customer's deployed devices. Telit provides an *EE Mirror Server Setup* Application Note containing instructions for setting up this server. The customer may set up one or more additional mirror servers for purposes of redundancy and/or load balance.

3.1.5. Customer Device

The Host application within the customer device is responsible for downloading the SGEE-EPO file from the customer mirror server and transfers the file to the GNSS receiver module using the communication link between the host and the GNSS device (i.e. a serial port).

The GNSS device receives the file over the communication link and unpacks the EE data. The module saves the blocks of EE and stores in NVM, or saved in memory, from where they can be retrieved as needed by the GNSS receiver.

4. SGEE-EPO File Format

4.1. EPO-II Format

The basic unit of an EPO file is SAT Data, which is corresponding to the predicated satellite orbits data for one satellite in the GPS constellation, or any other constellation in the GNSS system.

As a reference, the following information is for GPS only EPO.

- Data size of a SAT Data is 72 bytes.
- One EPO segment contains the SAT Data for whole satellites in a constellation (GPS: 32), the data size for an EPO segment is 2304 bytes.
- Each EPO file contains several EPO segments, therefore the file size must be a multiple of 2304. That means no partial segment is allowed in a EPO file.
- An EPO segment is valid for 6 hours. Therefore, there will be four (4) EPO segments contained in a EPO file to provide a predictive satellite orbits data for one day period.

For more information about the covering periods and file sizes, please refer to section 3.1.2 Extended Ephemeris Files.

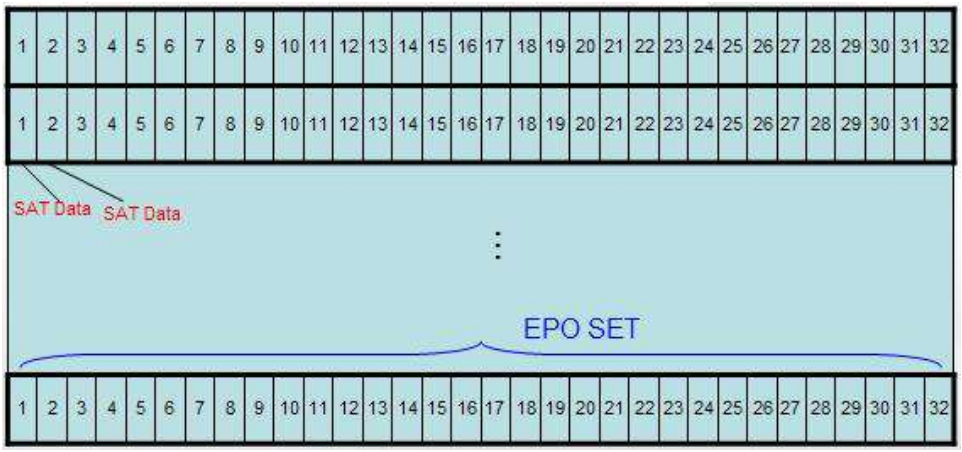


Figure 3: EPO file format and segments

An EPO SET = an EPO segment.

4.2. Binary Protocol

This section provides a brief introduction to the Mediatek's binary packet format as it relates to the data format used in EPO file transfer.

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

Figure 4: Binary format

1. Preamble (2-Byte word): 0x2404

2. Length: Total number bytes in the packet from Preamble to End Word.

Maximum packet size: 256 bytes

3. Use little endian

4. Use one byte alignment

5. Command ID:

(1) 0 ~ 999: conform to ASCII Protocol

(2) 1000 ~ 65535: designated for Mediatek Binary Protocol

6. Checksum:

The checksum is the 8-bit exclusive OR of all bytes in the packet between but not including the "Preamble" and the "Checksum"

7. End Word (2-Byte word): 0x0A0D

4.3. Binary Packet For EPO

The Mediatek binary packet that is used for EPO data transfer is called MTK_BIN_EPO packet (packet type 723, 0x02D3).

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 byte	2 bytes	

An EPO file will be divided into several SAT Data and encapsulated in several MTK_BIN_EPO packets to be transferred to Telit GNSS receivers that supports the format. Each MTK_BIN_EPO packet contains a 2-byte EPO SEQ and 3 SAT Data. The packet length of MTK_BIN_EPO is 227 bytes. The EPO SEQ is used for synchronization of MTK_BIN_EPO packets in transferring protocol.

Sometimes, there's no enough EPO data to full fill the three SAT Data fields. You can leave some of the three fields as blank, that is, to fill them with 0x00. A MTK_BIN_EPO packet that only contains 0 ~ 2 SAT Data is possible and acceptable.

4.4. SGEE-EPO File Transfer To Receiver

This section provides further details regarding the transfer of an SGEE-EPO file from the Host application to the receiver module.

4.4.1. NVM Storage Requirement

As mentioned above, SGEE-EPO data can be stored in non-volatile memory (NVM) if it is supported. It is required that the NVM storage and/or adequate memory resource is provided in order for SGEE-EPO to be operational. Currently MT3333 supports 14 day SGEE-EPO.

4.4.2. Full-power Requirement

Even though SGEE-EPO is also used by the receiver module when it is operating in a low power mode, it is required that the receiver is at Full Power mode for an SGEE-EPO file transfer. If the receiver is operating in a low power mode, the Host application must command the module to enter full power mode prior to initiating an SGEE-EPO file transfer.

4.4.3. File Transfer Process

The process for transferring an SGEE-EPO file to the receiver module is performed using serial data messages to transport the file over the serial host port.

At initial, or in normal operating state, the communication protocol at a module is set at ASCII mode in that the data input/out is on a NMEA string basis.

One of the protocols for SGEE-EPO file transfer is the Binary packets.

In this protocol the Host application sends a command to the module, to change the serial communication protocol from the ASCII mode to Mediatek Binary mode.

Host application initiates and carries out a SGEE-EPO file transfer by by sending a series of MTK_BIN_EPO packet which contains 1 ~ 3 satellite data to the module. All satellite data contained in the MTK_BIN_EPO packets are sourced from the SGEE-EPO file.

Initially the SGEE-EPO transfer sequence requires wait-for-ack between the sequential MTK_BIN_EPO packets. The Host application must wait for the acknowledgement before sending the next MTK_BIN_EPO packet. But as a result of improvements in the receiving side of the GPS firmware, the wait-for-ack is no longer required in the protocol.

After all SGEE-EPO data based on the validity period (one-day, three-day, 14-day, etc) is transferred, Host application sends a final MTK_BIN_EPO packet which contains the sequence number of 0xFFFF to indicate the completion of the EPO file transfer. The three satellite data fields in the MTK_BIN_EPO packet are filled with blank (0x00). If this packet is never received by the module, it has 10-seconds timeout and switch back to the default communication protocol, the ASCII mode.

On completion of the SGEE-EPO file transfer (indicated by the final MTK_BIN_EPO packet to the module), Host application sends a Mediatek binary command to the module, to change its serial communication protocol back to the default ASCII mode.

4.4.4. Error Handling

If there is any problem occurs in the transferring protocol, you shall stop the process and restart the transferring protocol again. Every time when the protocol starts, the EPO sequence number should be reset to zero to indicate the GPS receiver that a new transferring process has begun. The GPS receiver then needs to do preparation for the new process.

The interval of time between two continuous MTK_BIN_EPO packets shall not be longer than 10 seconds. Otherwise, the GPS receiver will determine to have problem occurred and terminate the process.

4.5. Check SGEE-EPO data in GPS chip

It needs to ensure that the EPO data were successfully updated into the GPS chip. After finishing the EPO transfer protocol, make sure current data port packet format is ASCII mode.

You can issue the PMTK_Q_EPO_INFO command:

$PMTK607*33<CR><LF>

to query the EPO data status.

The GPS chip will return you PMTK_DT_EPO_INFO like below

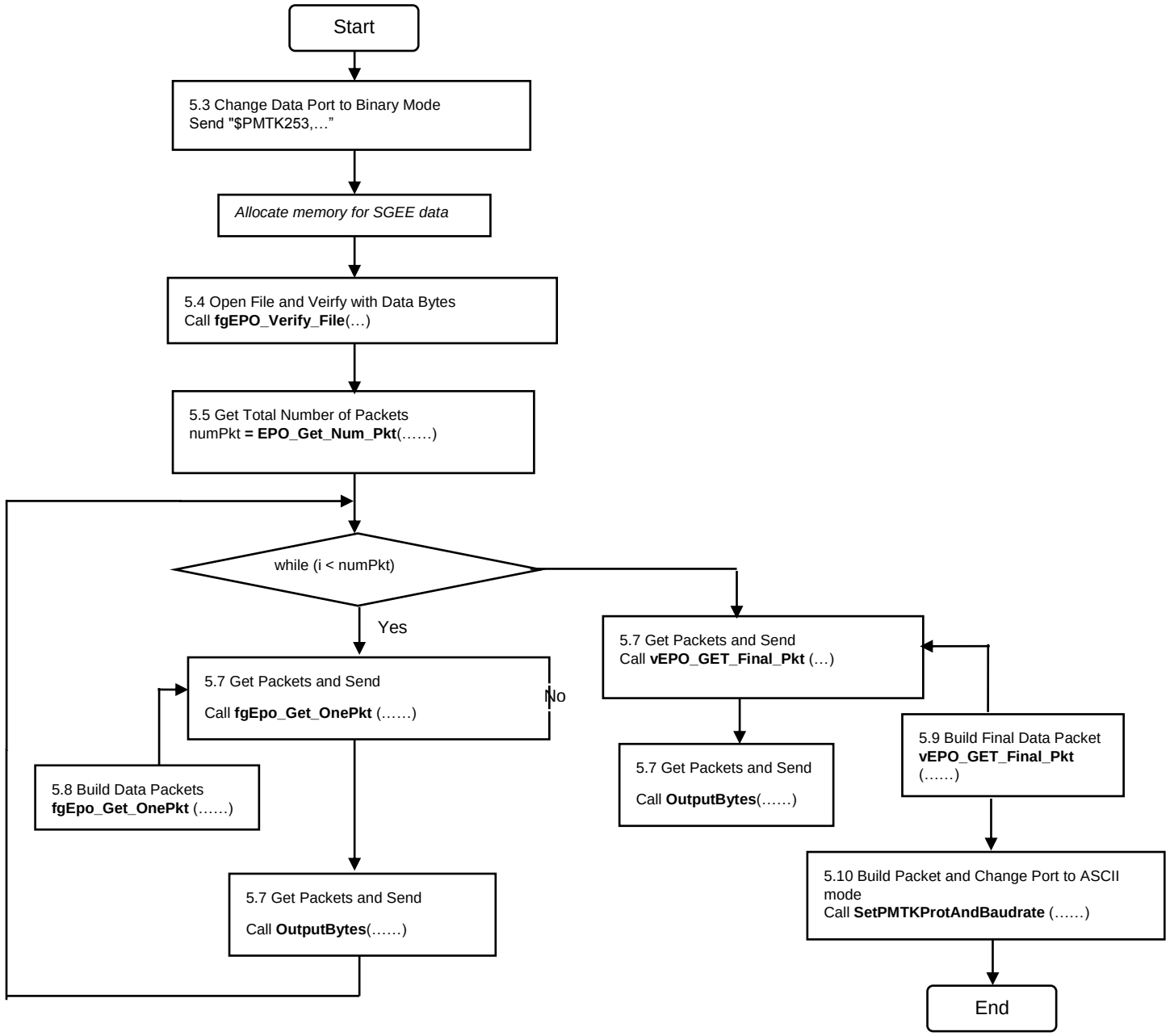
$PMTK707,56,1468,172800,1470,151200,1468,259200,1468,259200*1F<CR><LF>

This packet shows you the information of EPO data that stored inside GPS chip. For 14-day EPO file, the first argument following PMTK707 will be 56; for 7-day EPO file, it will be 28; (1468, 172800) means the starting GPS time (GPS week, GPS TOW) of the EPO data, and (1470, 151200) means the ending GPS time (GPS week, GPS TOW) of the EPO data. You have to convert (GPS week, GPS TOW) into UTC time format, so as to compare the UTC time to verify that the EPO data stored in the flash matches that of the EPO file.

Please refer to section 7.1.3 Query EPO Info and 7.1.4 Response EPO Info for the details of EPO related ASCII commands.

5. Pseudo Code for SGEE-EPO Update Process

5.1. Flow Chart of the SGEE-EPO Update Process



The above flowchart illustrates the core flow of the SGEE-EPO update process as demonstrated in the pseudo code, with following notes:

- The names with bold fonts indicate they are functions
- Each block contains subsection number and its self-explanatory subtitles for what it is attempting to accomplish
- The text with Italicized fonts indicates they are general descriptions for the intended task but no reference code is provided

Please also note that the pseudo code is, in the format, based on the Windows platform implementation. For applications on any other platforms or environment, it is programmer's responsibility to adopt, design and implement the functionalities for their specific running environment.

This section lists pseudo code for reference purpose.

5.2. Definition of Constants

```
////////////////////////////////////////////////////////////////////
/** \brief Definitions of constants
 *
 * #define constants
 */
////////////////////////////////////////////////////////////////////
#define SUCCESS   1
#define FAIL      0

#define PREAMBLE_MTK_STX0   0x04
#define PREAMBLE_MTK_STX1   0x24
#define PREAMBLE_MTK_ETX0   0x0D
#define PREAMBLE_MTK_ETX1   0x0A

#define BYTES_PER_EPO_SEG   2304
#define NUMB_SAT_PER_PACKET    3
#define BYTES_PER_SAT         72
#define BYTES_3SATS_PER_PACKT   NUMB_SAT_PER_PACKET*BYTES_PER_SAT
#define NUMB_SAT_PER_EPO_SEG    32
#define MTKBIN_3EPO_PKT_LNG   227
```

5.3. Change Data Port to Binary Mode

At initial, the protocol setting of the communication data port is supposed to be ASCII protocol. Since EPO data are transferred using Binary Packet, you have to change the protocol setting to Binary Protocol before starting EPO Transfer Protocol.

```
////////////////////////////////////////////////////////////////////
```

```
/** \fn void SetBinProt(UINT32 baud)
This function builds the PMTK command string to set the UART to the
binary packet protocol
*/
////////////////////////////////////////////////////////////
void SetBinProt(UINT32 baud)
{
   char* szCmdBuf= "$PMTK253,1,115200*00";

   OutputText(szCmdBuf);
}
```

The function OutputText(char*) must be implemented by the programmer.

5.4. Open File and Veirfy with Data Bytes

This function to read the EPO file, and then verify the validity of EPO data. If the input EPO file is not in valid file length, the programmer shall terminate the process.

```
int fgEPO_Verify_File(char* epoFileName)
{
   /**
   Open the EPO file and read the file into the allocated buffer.
   This function shall return the data bytes read from the file.
   */
   // [CODE]

   /**
    * A valid data file length must be multiples of segment length.
    * GPS: the segment length is 2304 bytes
   if ((bytesRead % BYTES_PER_EPO_SEG) != 0)
   {
      bytesRead = 0;
   }

   return bytesRead;
}
```

5.5. Get Total Number of Packets

This function to get total number of MTK_BIN_EPO packets that will be sent in EPO_Get_Num_Pkt function.

```
int EPO_Get_Num_Pkt(int recordLen)
{
   int retVal = 0;

   if (recordLen != 0)
   {
```

```
      int numbOfSet = recordLen/BYTES_PER_EPO_SEG;
      retVal = (numbOfSet*NUMB_SAT_PER_EPO_SEG)/NUMB_SAT_PER_PACKET;
   }
   return retVal;
}
```

5.6. Start EPO Download Function

Now the data in the data port will be viewed as Binary Packet format. Please create a thread to transmit / receive binary packets for the data port.

```
{
   ……

   /** Allocate buffer for the EPO data
   // This must be implemented by the programmer.
   // ……


   /**
     * Call EPO download engine, with passing parameters
       - pEPOBuffer: EPO data buffer
       - fSize: EPO data size
       - numPkts: number og packets to send
       - nBaudRate: the current baud rate
    */
    EMOD_DLEng(nBaudRate, numPkts, fSize, pEPOBuff);
}
```

Please note that the code here is for reference and is it based on the Windows platform only, as it is depending on the programmer's design as how to use a thread or task to run a SGEE-EPO update process, or if to use one after all.

5.7. Get Packets and Send

This function is to start SGEE-EPO data transfer protocol to send SGEE-EPO data.

```
bool EMOD_DLEng(UINT32 baud, int numPkt, DWORD epoSize, const BYTE*
pEPOBuff)
{
   for (int i = 0; i < numPkt; i++)
   {
      if (fgEpo_Get_OnePkt(dataBuf, sizeof(dataBuf), epoSeq,
pEPOBuff) == SUCCESS)
      {
         //* Send the current packet
         OutputBytes(dataBuf , MTKBIN_3EPO_PKT_LNG);
```


```
            //* Update the last Epo SEQ number
            lastEpoSeq = epoSeq;
            epoSeq++;
            ::Sleep(100);
         }
      }
   }
```

5.8. Build Data Packets

This function, fgEPO_Get_One_Pkt(…), takes out three SAT data from the SGEE-EPO data file and encapsulated them in a MTK_BIN_EPO packet with appropriate EPO SEQ number.

```
int fgEpo_Get_OnePkt(unsigned int seq, BYTE* pEpoRecord, BYTE* pData,
int size)
{
   int retVal = FAIL;

   /**
Sanity check the input argument
*/
   // [CODE]

   if ((seq+1)*BYTES_3SATS_PER_PACKT <= nEpoRecordSize)
   {
      int indx = 0;

      pData[indx++] = PREAMBLE_MTK_STX0;
      pData[indx++] = PREAMBLE_MTK_STX1;

      pData[indx++] = 0xE3;
      pData[indx++] = 0x0;

      pData[indx++] = 0xD3;
      pData[indx++] = 0x02;

      pData[indx++] = seq & 0xFF;
      pData[indx++] = (seq >> 8) & 0xFF;

      memcpy((BYTE*)&(pData[indx]),
(BYTE*)&(pEpoRecord[seq*BYTES_3SATS_PER_PACKT]),
BYTES_3SATS_PER_PACKT);

      indx += BYTES_3SATS_PER_PACKT;

      BYTE chksum = ComputeChkSum(pData+2, 6 +
BYTES_3SATS_PER_PACKT); //* len byte + Cmd byte + Seq byte = 6
      pData[indx++] = chksum;

      pData[indx++] = PREAMBLE_MTK_ETX0;
      pData[indx++] = PREAMBLE_MTK_ETX1;
```

```
      retVal = SUCCESS;
   }
```

fgEPO_Get_One_Pkt also contains the code to fill the MTK_BIN_EPO packet when there are less than three (3) sat data left from the data buffer.

```
int fgEpo_Get_OnePkt(unsigned int seq, BYTE* pEpoRecord, BYTE* pData,
int size)
{
   /**
    * Sanity check the input argument
*/
   // [CODE]

   if ((seq+1)*BYTES_3SATS_PER_PACKT <= nEpoRecordSize)
   {
   }
   else
   {
      int trailBytes = nEpoRecordSize - seq*BYTES_3SATS_PER_PACKT;

      if ((trailBytes % 72) == 0)
      {
         int indx = 0;
         BYTE fillBytes[BYTES_3SATS_PER_PACKT];   // To prefill the
bytes with numerial 0
         memset (fillBytes, 0x30, sizeof(fillBytes));
         memcpy((BYTE*)&(fillBytes[0]),
(BYTE*)&(pEpoRecord[seq*BYTES_3SATS_PER_PACKT]), trailBytes);

         pData[indx++] = PREAMBLE_MTK_STX0;
         pData[indx++] = PREAMBLE_MTK_STX1;

         pData[indx++] = 0xE3;
         pData[indx++] = 0x0;

         pData[indx++] = 0xD3;
         pData[indx++] = 0x02;

         pData[indx++] = seq & 0xFF;
         pData[indx++] = (seq >> 8) & 0xFF;

         memcpy((BYTE*)&(pData[indx]), (BYTE*)&fillBytes[0],
BYTES_3SATS_PER_PACKT);

         indx += BYTES_3SATS_PER_PACKT;

         BYTE chksum = ComputeChkSum(pData+2, 6 +
BYTES_3SATS_PER_PACKT); //* len byte + Cmd byte + Seq byte = 6
         pData[indx++] = chksum;

         pData[indx++] = PREAMBLE_MTK_ETX0;
         pData[indx++] = PREAMBLE_MTK_ETX1;
```

```
            retVal = SUCCESS;
        }
    }

    return retVal;
}
```

Send current MTK_BIN_EPO packet. The packet size of MTK_BIN_EPO is MTKBIN_3EPO_PKT_LNG.

The call to OutputBytes() must be made by the programmer.

```
void SendData(BYTE* pData, int dataLen)
{
    writeChars(pData, dataLen);
}
```

The function writeChars must be implemented by the programmer.

5.9. Build Final Data Packet

Generate final MTK_BIN_EPO packet to indicate the GPS receiver that the process is finish.

```
int vEPO_GET_Final_Pkt(BYTE* pData, int size)
{
    int retVal = FAIL;
    BYTE val = 0;

    /**
     * Sanity check the input argument
    */
    // [CODE]

    int indx = 0;

    pData[indx++] = PREAMBLE_MTK_STX0;
    pData[indx++] = PREAMBLE_MTK_STX1;

    pData[indx++] = 0xE3;
    pData[indx++] = 0x0;

    pData[indx++] = 0xD3;
    pData[indx++] = 0x02;

    pData[indx++] = 0xFF;
    pData[indx++] = 0xFF;

    memcpy((BYTE*)&(pData[indx]), &val, BYTES_3SATS_PER_PACKT);

    indx += BYTES_3SATS_PER_PACKT;

    BYTE chksum = ComputeChkSum(pData+2, 6 + BYTES_3SATS_PER_PACKT);
```

```
pData[indx++] = chksum;

pData[indx++] = PREAMBLE_MTK_ETX0;
pData[indx++] = PREAMBLE_MTK_ETX1;
}
```

Like before, call the SendData function to send the final MTK_BIN_EPO packet to the serial port.

The call to SendData() must be made by the programmer.

5.10. Build Packet and Change Port to ASCII mode

Switch UART protocol setting to ASCII mode and baudrate 115200.

```
void SetTextProt(UINT32 baud)
{
   int indx = 0;
   BYTE cmd[200] = { 0 };

   cmd[indx++] = 0x04; //* Preamble
   cmd[indx++] = 0x24;

   cmd[indx++] = 0x0E; //* Len 2 bytes for MTK_BIN_EPO packet
   cmd[indx++] = 0x00;

   cmd[indx++] = 0xFD; //* Command ID for MTK_BIN_EPO packet
   cmd[indx++] = 0x00;

   cmd[indx++] = PCKT253_SETPMTKPROTO_FLAG;  //* PMTK protocol

   cmd[indx++] = (baud >> 24) & 0x000000FF;
   cmd[indx++] = (baud >> 8) & 0x000000FF;
   cmd[indx++] = (baud >> 16) & 0x000000FF;
   cmd[indx++] = baud & 0x000000FF;
   cmd[indx++] = ComputeChkSum(&cmd[2], indx-2);

   cmd[indx++] = 0x0D;
   cmd[indx++] = 0x0A;

   OutputBytes(cmd, indx);
}
```

The call to OutputBytes() must be made by the programmer.

6. Sample Project for EPO Update: TDepo

TDepo is a console application of Windows, a command line tool that illustrates an sample implementation of the Mediatek's EPO update protocol.

TDepo is developed and intended to run in the following enviornment:

- Windows 7 or later
- Visual Studio 2010
- .NET 4.0

6.1. TDepo Usage

6.1.1. Directory and Files

In a simplest setup, create a folder directory called TDepo. Then copy the files into the directory

1) TDepo.exe
2) TDepoCfg.txt
3) MTK30.EPO

- TDepo.exe: the executable
- TDepoCfg.txt: configuration file to provide parameters for running the application. This file is highly recommended, unless user choose to provide the parameters through command line. This file is assumed to reside in the same directory as the executable.
- MTK30.EPO: the EPO data file for Mediatek's GPS only EPO. This file can be stored in any user's directory – provided that its full path is included wither from the command line argument or in the configuration file TDepoCfg.txt.

Contents of the TDepoCfg.txt file

```
-pCOM26
-b9600
-eC:\TDepo\MTK30.EPO
```

The parameters contained in the configutration file can also be entered from command line.

6.1.2. EPO File Source

There are significant information available pertaining where to get and how to use the EPO file. But for the purpose of this application note and the nature that is to introduce the EPO download sample application, it is suffice to say that the EPO file is available for download at Telit's EPO server.

There are credentials (user name, password, etc) that are required in loggin. Please Telit customer support for more details about the information.

The EPO file can be downloaded through internet browser, or, in the case of TelitView application is available, there is a EPO Host Manager window in the application that also be used to provides the access and functionality to download the EPO file.

Please take a note about the directory where the EPO file is stored at after download from the server, and provide the full path either in the TDepo command line or use the configuration file.

6.1.3. Application Launch

From command line type “TDepo –h”, the application launch will bring up the following printout on screen.

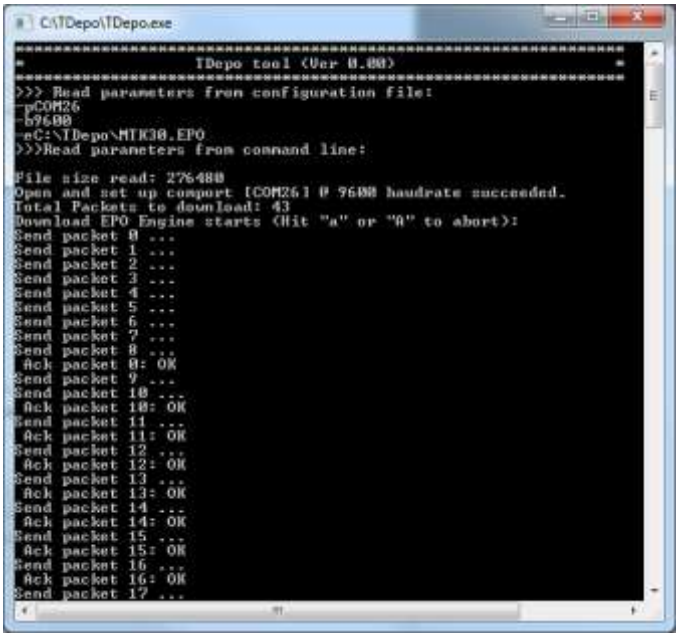
```
*****************************************************************
*                          TDepo tool (Ver 0.00)                *
*****************************************************************
>>> Help menu:
TDepoCfg -p<port name> -b<baud rate> -e<EPO filepath>
  <port name>: PC com port name (COM2, COM39, etc)
  <baud rate>: Init baud rate (115200, 9600, etc).
          This is the baud rate that communication is established on
          before the EPO download, and return to after the EPO download.
  <EPO filepath>: complete file path for the EPO file file
TDepoCfg -h<no parameter>
  : help menu (this menu)
  - Not parameter-order sensative
  - Command line inputs take priority over parameters from file

Press any key to continue . . .
```

User can use parameter “-h” from command line only and it will bring the menu printout on screen.

User is expected to provide other supported argument except “-h”, or no argument at all but provide the necessary configuration file for the application to run.

The following screenshot illustrates what is seen when the application is running to update EPO to the module.



6.1.4. TTFF with EPO Available in GPS

On completion of update EPO to the GNSS module, user may launch TelitView, or PowerGPS tool (a toll provided by Mediatek), and command a reset.

The most evident example would be the improvement of TTFF on a Warm start or Cold start. With the EPO now is available in the GNSS module, a warm start of GPS device would yield a nominal TTFF of 1 – 3 seconds.

6.2. TDepo Source Code

The table below lists and describes the TDepo source files

Files	Description
TDepo.h TDepo.cpp	Application's main file with the main function, command line parser, open com port, and other set-up functions.
Com_port.h Com_port.cpp	The module to handle the com port and serial communication.
HostEpoEng.h HostEpoEng.cpp	The Host EPO update engine that executes the EPO update process based on the protocol
Messages.h Messages.cpp	The functions that provide the implementation of some functions that are not a part of the EPO update, but nevertheless necessary to be included to support the Mediatek interface.

For more information, please refer to the source code listing for implementation details.

7. Appendix

7.1. ASCII Commands ($PMTK)

7.1.1. Set Data Output Format and Baudrate

This command sets data output format and baudrate for current port

[Format]

$PMTK253,Flag,Baudrate

Flag : 0 - NMEA mode; 1 - binary mode

Baudrate: baudrate for the new output mode

It is recommended that an explicit baudrate value is used in the command.

All valid values are:

- 4800, 9600, 14400, 19200, 38400, 57600, 115200.

[Example]

Switch the UART protocol format to BINARY mode, and use default baudrate 115200

$PMTK253,1,0*37<CR><LF>

7.1.2. Change UART Format Packet (Packet Type 253)

This binary packet sets UART communication protocol and baudrate to ASCII mode.

[Format]

MTK Binary Packet

[Example]

Change UART to ASCII protocol and baudrate 115200

0x04 0x24 0x0E 0x00 0xFD 0x00 0x00 0x00 0xC2 0x01 0x00 0x30 0x0D 0x0A

Preamble: 0x2404

Packet Length: 0x000E <= Packet length: 14 bytes

Command ID: 0x00FD <= Change UART packet protocol

Data: 0x00 <= ASCII protocol

Data: 0x00 0xC2 0x01 0x00 <= UART baudrate 115200

Checksum: 0x30

End Word: 0x0A0D

7.1.3. Query EPO Info

This command queries the EPO data status stored in the GPS chip

[Format]

$PMTK607

[Return]

PMTK_DT_EPO_INFO

[Example]

Query the current EPO data status

$PMTK607*33<CR><LF>

7.1.4. Response EPO Info

This response contains the EPO data status stored in the GPS chip

[Format]

PMTK705,Set,FWN,FTOW,LWN,LTOW,FCWN,FCTOW,LCWN,LCTOW

Set: Total number sets of EPO data stored in chip

FWN, FTOW: GPS week number & TOW of the first set of EPO data stored in chip respectively

LWN, LTOW: GPS week number & TOW of the last set of EPO data stored in chip respectively

FCWN, FCTOW: GPS week number & TOW of the first set of EPO data that are currently used respectively

LCWN, LCTOW: GPS week number & TOW of the last set of EPO data that are currently used respectively

[Example]

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

8. Document History

Revision	Date	Changes
0	2014-12-10	Telit MT3333 SGEE-EPO Application Note preliminary release
1	2015-02-09	Update info. Added SC872-A
2	2015-03-06	Document file name changed. Telit Jupiter SGEE-EPO Application Note Rev2 release. Added sample project of TDepo, a Windows comman line program to illustrate the implementation of the EPO update flow and provide the full operational EPO update tool. - Sample run of the program - Directory and files - Information about how to get EPO file from server - Components of the source code