Lx6&LC86L&LG77L

Firmware Upgrade Guide

GNSS Module Series

Version: 1.0

Date: 2021-06-21

Status: Released

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

Disclaimer

While Quectel has made efforts to ensure that the functions and features under development are free from errors, it is possible that these functions and features could contain errors, inaccuracies and omissions. Unless otherwise provided by valid agreement, Quectel makes no warranties of any kind, implied or express, with respect to the use of features and functions under development. To the maximum extent permitted by law, Quectel excludes all liability for any loss or damage suffered in connection with the use of the functions and features under development, regardless of whether such loss or damage may have been foreseeable.

Duty of Confidentiality

The Receiving Party shall keep confidential all documentation and information provided by Quectel, except when the specific permission has been granted by Quectel. The Receiving Party shall not access or use Quectel's documentation and information for any purpose except as expressly provided herein. Furthermore, the Receiving Party shall not disclose any of the Quectel's documentation and information to any third party without the prior written consent by Quectel. For any noncompliance to the above requirements, unauthorized use, or other illegal or malicious use of the documentation and information, Quectel will reserve the right to take legal action.

Copyright

About the Document

Document Information	
Title	**Lx6&LC86L&LG77L Firmware Upgrade Guide**
Subtitle	GNSS Module Series
Document Type	Firmware Upgrade Guide
Document Status	Released

Revision History

Version	Date	Description
1.0	2018-08-23	Initial
1.1	2021-06-21	Added applicable modules L26-LB, L76-LB, LC86L and LG77L.

Contents

Figure Index

1 Introduction

Quectel GNSS modules support FOTA function, which allows you to upgrade the firmware of the modules over the air. This document introduces the detailed firmware upgrade procedures, through which you can download the built image to the module via UART.

The document is applicable to the following Quectel GNSS modules:

- L26
- L26-LB
- L76
- L76-LB
- L76-L
- L86
- L96
- LC86L
- LG77L

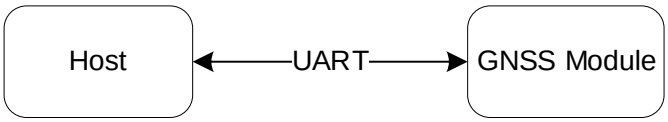

Figure 1: Firmware Upgrade Connection

2 Firmware Upgrade Procedure

The whole procedure has two stages:

Stage 1: Download DA to the internal SRAM, and then jump to DA start address to execute DA in the internal SRAM.

Stage 2: Download BIN file to the flash, then jump to the flash start address to execute the file.

2.1. CMD_Start (Start Upgrade Procedure)

After the GNSS module is powered on, the host sequentially sets 7 baud rates for serial communication and sends the NMEA_START_CMD **$PMTK180*3B\r\n** at each baud rate to force the module to reset. Then the host settles at the right baud rate to establish communication and the module starts the upgrade procedure.

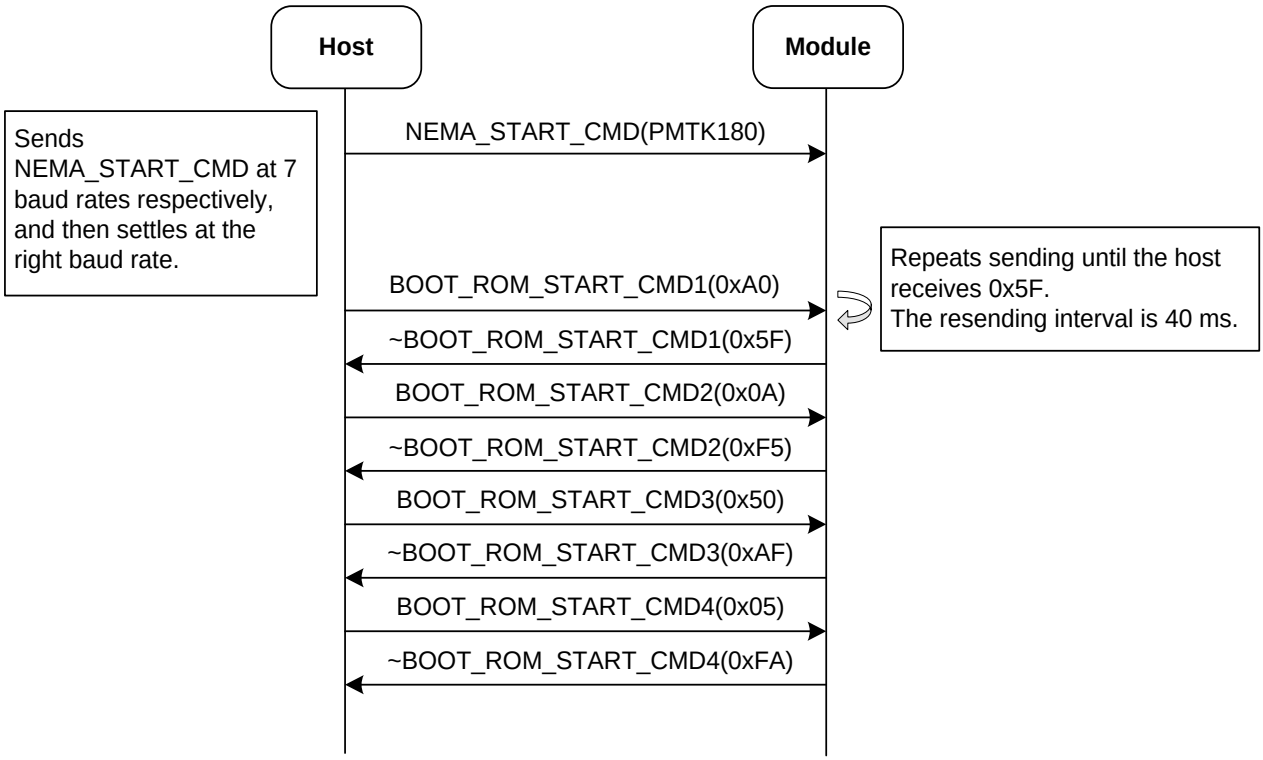

Figure 2: Flow of CMD_Start

NOTE

The 7 baud rates are 4800, 9600, 14400, 19200, 38400, 57600 and 115200 bps.

2.2. CMD_Write (Write DA to SRAM)

During this procedure, DA will be written to SRAM.

The Download Agent (DA) is an application that allows to perform the flash download via serial interface. After DA is successfully downloaded and running on target modules, it will be standby and passively waiting for the host to issue firmware upgrade commands via UART.

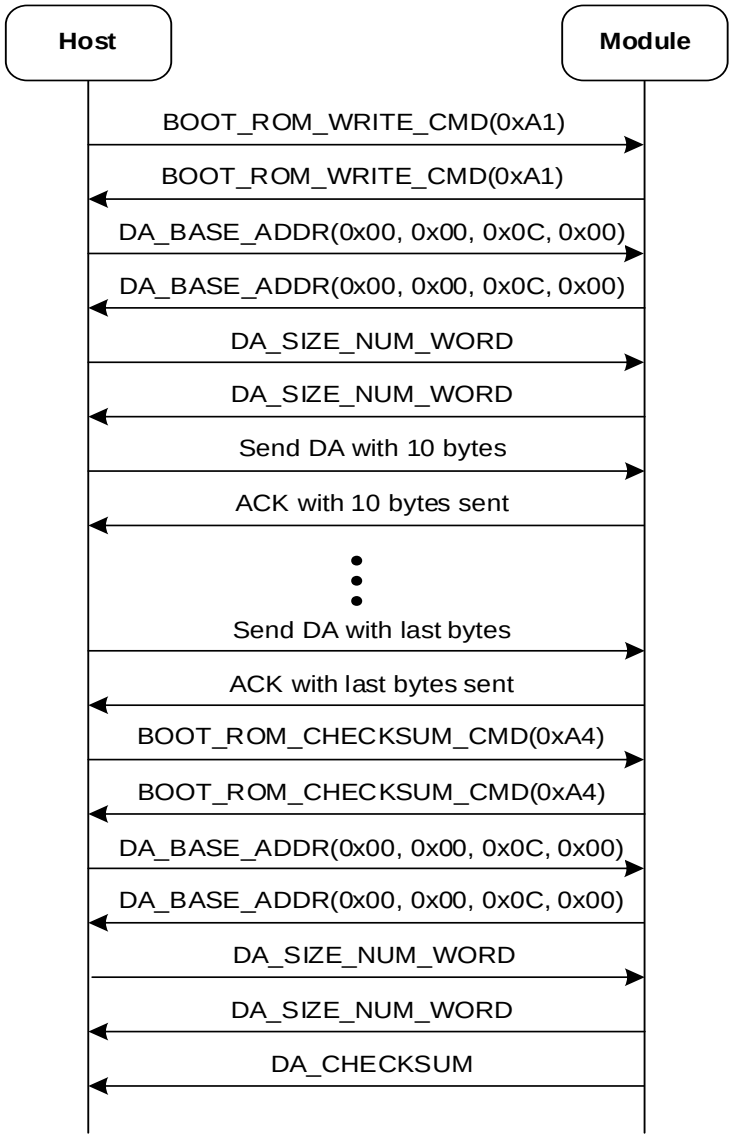

Figure 3: Flow of CMD_Write

NOTE

1. DA_SIZE_NUM_WORD is four bytes in big-endian mode.
2. DA_CHECKSUM is two bytes.

Sample Code for DA File Checksum

```
uint16_t DA_CHECKSUM (uint8_t *buf, uint32_t buf_len)
{
    uint16_t checksum = 0;
    if (buf == NULL || buf_len == 0)
    {
        return 0;
    }
    int i = 0;
    for (i = 0; i < buf_len / 2; i++)
    {
        checksum ^= *(uint16_t *) (buf + i * 2);
    }
    if ((buf_len % 2) == 1)
    {
        checksum ^= buf[i * 2];
    }
    return checksum;
}
```

2.3. CMD_Jump (Jump to DA Start Address)

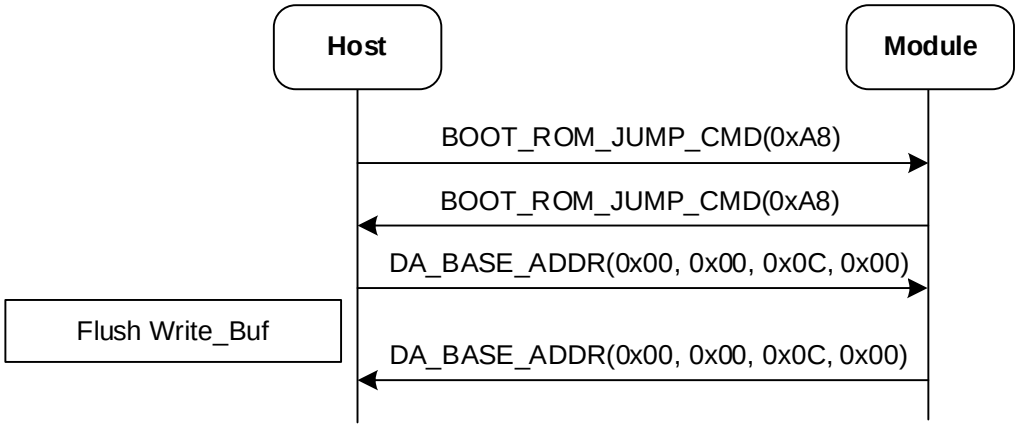

Figure 4: Flow of CMD_Jump

2.4. CMD_Sync (Report Flash Information)

When DA is downloaded and executed, it will actively report one byte **SYNC_CHAR**, two-byte **DA_VERSION**, one byte **FLASH_DEVICE_ID**, four-byte **FLASH_SIZE**, eight-byte **FLASH_HW_ID** and four-byte **EXT_SRAM_SIZE**.

- **SYNC_CHAR**

  When DA is executed, it returns **SYNC_CHAR** (0xC0). If the returned byte is not **SYNC_CHAR**, then a wrong DA is downloaded probably.

- **DA_VERSION**

  After **SYNC_CHAR** is returned, DA will return its version number. The version number contains two bytes, which are major version and minor version, respectively. The relevant program in the host should check whether it supports this DA.

- **FLASH_DEVICE_ID**

  After the DA version is reported, DA will automatically detect the flash type on the module.

- **FLASH_SIZE**

  Four bytes flash size. For example, 128 Mbits flash will be 0x01000000 bytes, and DA will send 0x01, 0x00, 0x00, and 0x00.

- **FLASH_MANUFACTURE_CODE**

  Two bytes flash manufacture code.

- **FLASH_DEVICE_CODE**

  Two bytes flash device code.

- **FLASH_EXT_DEVICE_CODE1**

  Two bytes flash extended device code 1.

- **FLASH_EXT_DEVICE_CODE2**

  Two bytes flash extended device code 2.

- **EXT_SRAM_SIZE**

  Four bytes external SRAM size. For example, 64 Mbits external SRAM will be 0x00800000 bytes, and DA will send 0x00, 0x80, 0x00 and 0x00.

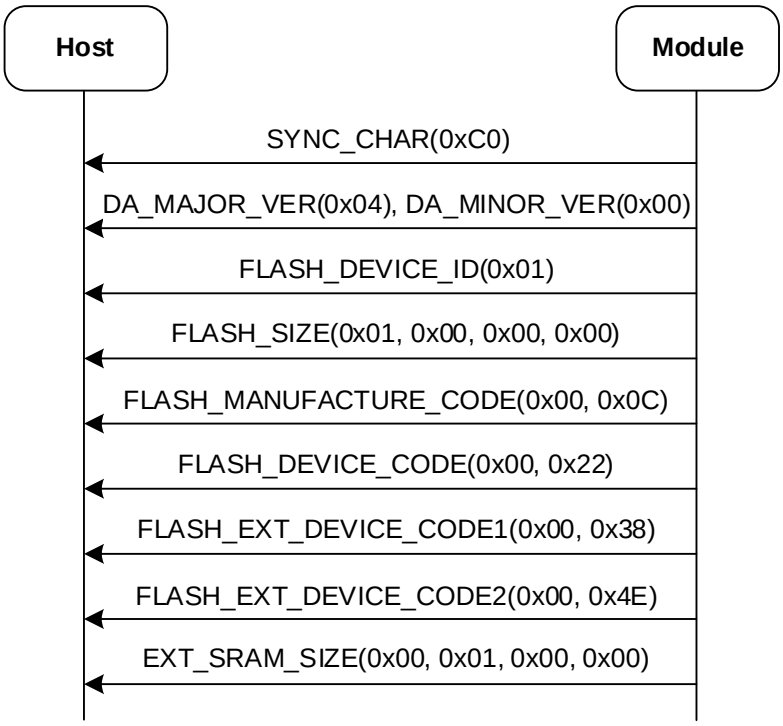

Figure 5: Flow of CMD_Sync

NOTE

The flash related information is fixed for handshake authorization, not real flash information.

2.5. CMD_SetMemBlock (Set Memory Block)

The commands are used to notify DA the total memory block count and the range for each block. The block information indicates how many BIN files will be downloaded and the range of each BIN file.

If any memory block exceeds the flash size, DA will return **NACK** (0xA5) to indicate the **DA_MEM_CMD** command is failed. If all the download memory blocks are valid, DA will return **ACK** (0x5A) and **UNCHANGED_BLOCK_COUNT**.

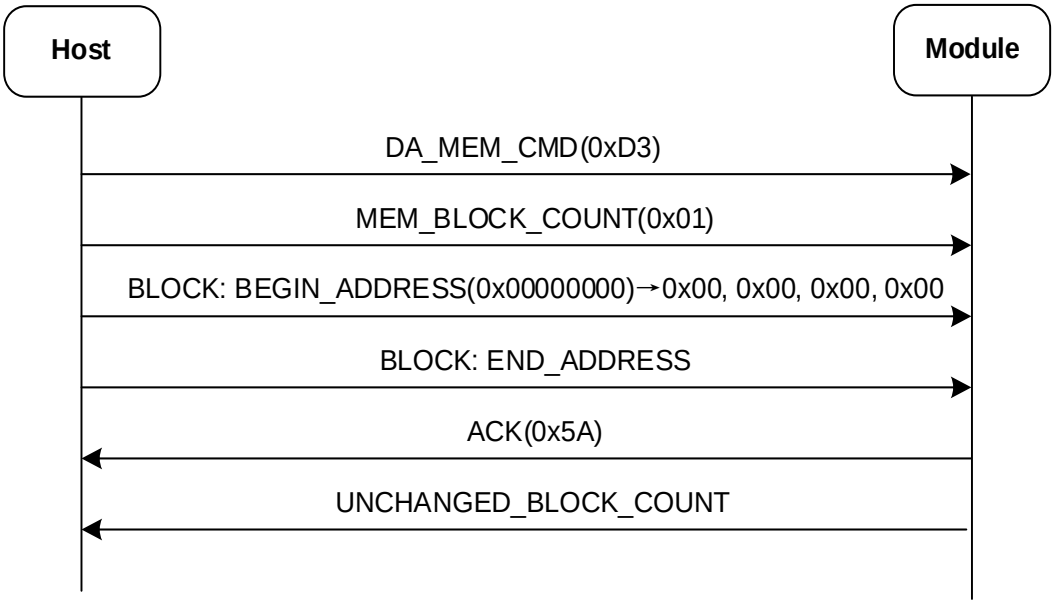

Figure 6: Flow of CMD_SetMemBlock

NOTE

END_ADDRESS = BEGIN_ADDRESS + GNSS BIN File Size - 1

2.6. CMD_WriteData (Write BIN File to Module Flash)

The commands are used to write all the data of BIN file to module flash. Every packet has a fixed length, that is **PACKET_LENGTH** plus two bytes checksum. As the last packet is usually not enough for **PACKET_LENGTH**, the relevant program in the host should send the actual data length plus two bytes checksum.

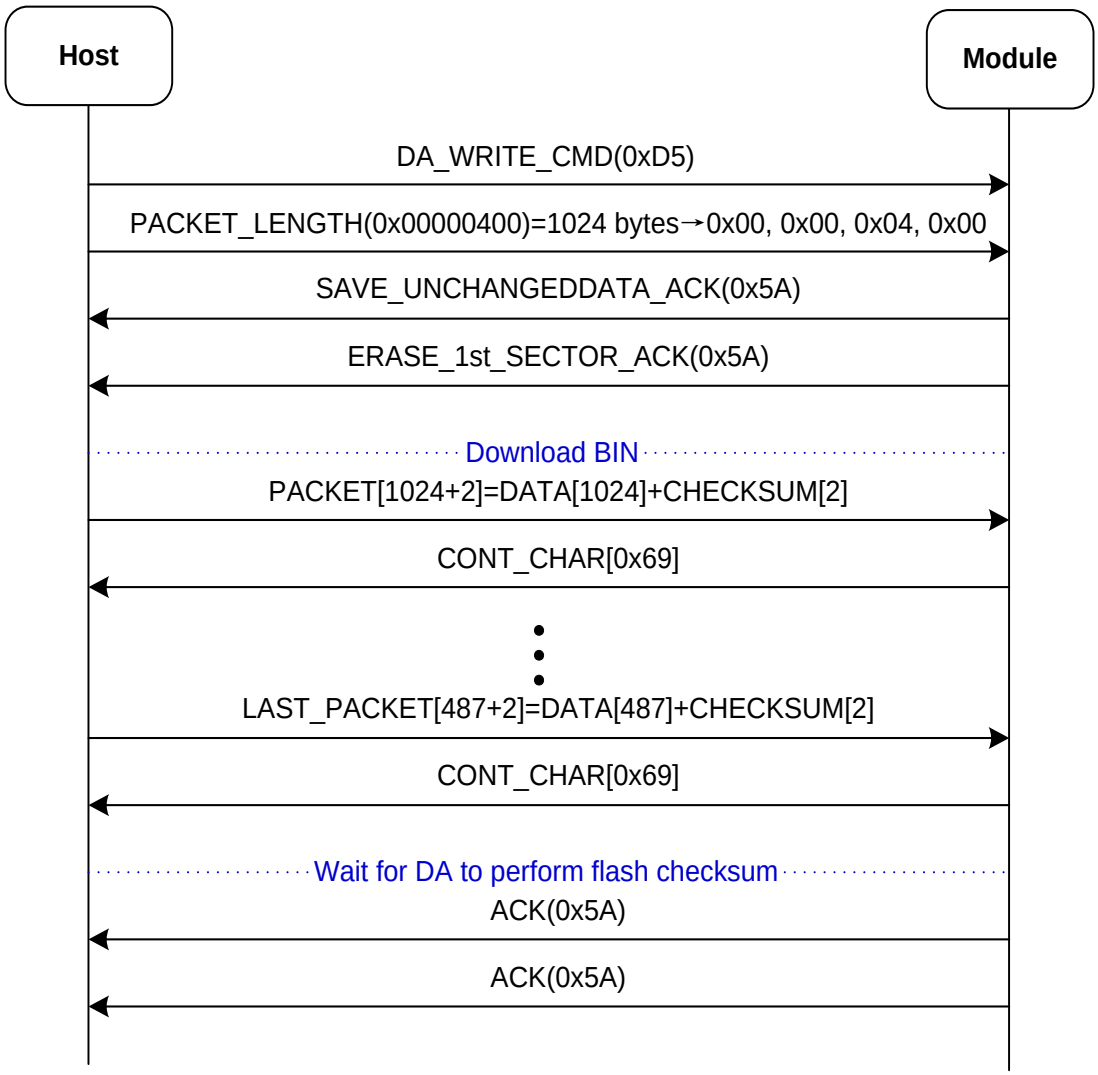

Figure 7: Flow of CMD_WriteData

The Sample Code for Checksum

```
uint16 DATA_CHECKSUM(uint8_t *buf, uint32_t buf_len)
{
    uint16 checksum = 0;
    int i;
    for(i=0; i< buf_len;i++)
    {
        checksum += *(buf + i);
    }
    return checksum;
}
```

2.7. CMD_Finish (Finish Upgrade Procedure)

The command is used to make the module restart and output NMEA sentences. The firmware upgrade procedure is thus finished.

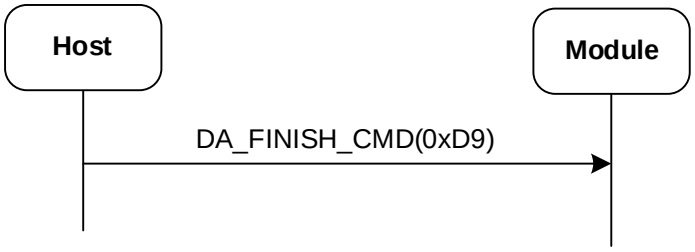

Figure 8: Flow of CMD_Finish

3 Upgrade Implementation Example

In this chapter you can find the example of firmware upgrade procedure.

```
//Figure 2: Flow of CMD_Start
//Host sends $PMTK180*3B for seven times.
24 50 4D 54 4B 31 38 30 2A 33 42 0D 0A
//Host continuously sends 0xA0
A0
A0
A0
…
A0
A0
A0
A0
//Module responds with 0x5F
5F
//Host sends 0x0A
0A
//Module responds with 0xF5
F5
//Host sends 0x50
50
//Module responds with 0xAF
AF
//Host sends 0x05
05
//Module responds with 0xFA
FA

//Figure 3: Flow of CMD_Write
//Host sends 0xA1
A1
//Module responds with 0xA1
A1
//Host sends 0x00000C00
00 00 0C 00
//Module responds with 0x00000C00
```

00 00 0C 00
//Host sends 0x0000173A
00 00 17 3A
//Module responds with 0x0000173A
00 00 17 3A
//Host sends 10 bytes
00 00 E1 A0 00 00 E1 0F 10 C0
//Module responds with 10 bytes
00 00 E1 A0 00 00 E1 0F 10 C0
//Host sends 10 bytes
E3 A0 00 01 E1 80 F0 00 E1 2F
//Module responds with 10 bytes
E3 A0 00 01 E1 80 F0 00 E1 2F
//Host sends 10 bytes
10 00 E3 A0 19 02 E2 81 10 04
//Module responds with 10 bytes
E3 A0 00 01 E1 80 F0 00 E1 2F
……
//Host sends 10 bytes
00 00 00 00 00 00 00 00 00 00
//Module responds with 10 bytes
00 00 00 00 00 00 00 00 00 00
//Host sends 10 bytes
00 00 00 00 00 00 00 00 00 00
//Module responds with 10 bytes
00 00 00 00 00 00 00 00 00 00
//Host sends 0x0000
00 00
//Module responds with 0x0000
00 00
//Host sends 0xA4
A4
//Module responds with 0xA4
A4
//Host sends 0x00000C00
00 00 0C 00
//Module responds with 0x00000C00
00 00 0C 00
//Host sends 0x0000173A
00 00 17 3A
//Module responds with 0x0000173A
00 00 17 3A
//Module responds with DA_CHECKSUM
67 8D (*Sample Code for DA File*)

//*Figure 4: Flow of CMD_Jump*
//Host sends 0xA8
A8
//Module responds with 0xA8
A8
//Host sends 0x00000C00
00 00 0C 00
//Module responds with 0x00000C00
00 00 0C 00

//*Figure 5: Flow of CMD_Sync*
//Module responds with Report_Flash_Information
C0 04 00 01 00 10 00 00 00 0C 00 22 00 38 00 4E 00 01 00 00

//*Figure 6: Flow of CMD_SetMemBlock*
//Host sends 0xD301
D3 01
//Host sends 0x00000000
00 00 00 00
//Host sends 0x000A31C7
00 0A 31 C7
//Module responds with 0x5A00
5A 00

//*Figure 7: Flow of CMD_WriteData*
//Host sends 0xD500000400
D5 00 00 04 00
//Module responds with 0x5A5A
5A 5A
//Host sends firmware packet (1024 Bytes) and checksum (2 Bytes)
18 F0 9F E5 18 F0 9F E5 18 F0 9F E5 18 F0 9F E5 18 F0 9F E5 CF D4 1B 91 14 F0 9F E5 ……
95 75 (*The Sample Code for Checksum*)
//Module responds with 0x69
69
//Host sends firmware packet (1024 Bytes) and checksum (2 Bytes)
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ……
FC 00 (*The Sample Code for Checksum*)
//Module responds with 0x69
69
//Host sends firmware packet (1024 Bytes) and checksum (2 Bytes)
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ……
E2 A6 (*The Sample Code for Checksum*)
//Module responds with 0x69
69

//Host sends firmware packet (1024 Bytes) and checksum (2 Bytes)
90 00 00 EB FE FF FF EA 0E 60 A0 E1 00 40 0F E1 80 40 84 E3 40 40 84 E3 1F 4- C4 E3 ……
A9 2D (*The Sample Code for Checksum*)
//Module responds with 0x69
69
……
//Host sends firmware packet (1024 Bytes) and checksum (2 Bytes)
06 10 A0 E1 05 00 A0 E1 F9 F9 FF EB 07 10 04 E0 00 50 A0 E1 07 00 A0 E1 CF F9 FF ……
99 CE (*The Sample Code for Checksum*)
//Module responds with 0x69
69
//Host sends firmware packet (1024 Bytes) and checksum (2 Bytes)
1B C0 A0 E3 01 20 A0 D3 02 1A 81 E1 04 F9 FF EA 06 2C 42 E2 02 3B 42 E2 01 30 93 E2 ……
91 E8 (*The Sample Code for Checksum*)
//Module responds with 0x69
69
//Module responds with 0x5A5A
5A 5A

//*Figure 8: Flow of CMD_Finish*
//Host sends 0xD9
D9

4 Appendix A Reference

Table 1: Terms and Abbreviations

Abbreviation	Description
ACK	Acknowledge
DA	Download Agent
FOTA	Firmware Over the Air
GNSS	Global Navigation Satellite System
NACK	Negative Acknowledge
SRAM	Static Random-access Memory
UART	Universal Asynchronous Receiver/Transmitter