GSM QuecFOTA Application Note

GSM/GPRS/GNSS Module Series

Version: 3.4

Date: 2022-03-24

Status: Released

At Quectel, our aim is to provide timely and comprehensive services to our customers. If you require any assistance, please contact our headquarters:

Quectel Wireless Solutions Co., Ltd.
Building 5, Shanghai Business Park Phase III (Area B), No.1016 Tianlin Road, Minhang District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local offices. For more information, please visit:
http://www.quectel.com/support/sales.htm.

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/technical.htm.
Or email us at: support@quectel.com.

Legal Notices

We offer information as a service to you. The provided information is based on your requirements and we make every effort to ensure its quality. You agree that you are responsible for using independent analysis and evaluation in designing intended products, and we provide reference designs for illustrative purposes only. Before using any hardware, software or service guided by this document, please read this notice carefully. Even though we employ commercially reasonable efforts to provide the best possible experience, you hereby acknowledge and agree that this document and related services hereunder are provided to you on an “as available” basis. We may revise or restate this document from time to time at our sole discretion without any prior notice to you.

Use and Disclosure Restrictions

License Agreements

Documents and information provided by us shall be kept confidential, unless specific permission is granted. They shall not be accessed or used for any purpose except as expressly provided herein.

Copyright

Our and third-party products hereunder may contain copyrighted material. Such copyrighted material shall not be copied, reproduced, distributed, merged, published, translated, or modified without prior written consent. We and the third party have exclusive rights over copyrighted material. No license shall be granted or conveyed under any patents, copyrights, trademarks, or service mark rights. To avoid ambiguities, purchasing in any form cannot be deemed as granting a license other than the normal non-exclusive, royalty-free license to use the material. We reserve the right to take legal action for noncompliance with abovementioned requirements, unauthorized use, or other illegal or malicious use of the material.

Trademarks

Except as otherwise set forth herein, nothing in this document shall be construed as conferring any rights to use any trademark, trade name or name, abbreviation, or counterfeit product thereof owned by Quectel or any third party in advertising, publicity, or other aspects.

Third-Party Rights

This document may refer to hardware, software and/or documentation owned by one or more third parties ("third-party materials"). Use of such third-party materials shall be governed by all restrictions and obligations applicable thereto.

We make no warranty or representation, either express or implied, regarding the third-party materials, including but not limited to any implied or statutory, warranties of merchantability or fitness for a particular purpose, quiet enjoyment, system integration, information accuracy, and non-infringement of any third-party intellectual property rights with regard to the licensed technology or use thereof. Nothing herein constitutes a representation or warranty by us to either develop, enhance, modify, distribute, market, sell, offer for sale, or otherwise maintain production of any our products or any other hardware, software, device, tool, information, or product. We moreover disclaim any and all warranties arising from the course of dealing or usage of trade.

Privacy Policy

To implement module functionality, certain device data are uploaded to Quectel's or third-party's servers, including carriers, chipset suppliers or customer-designated servers. Quectel, strictly abiding by the relevant laws and regulations, shall retain, use, disclose or otherwise process relevant data for the purpose of performing the service only or as permitted by applicable laws. Before data interaction with third parties, please be informed of their privacy and data security policy.

Disclaimer

a) We acknowledge no liability for any injury or damage arising from the reliance upon the information.
b) We shall bear no liability resulting from any inaccuracies or omissions, or from the use of the information contained herein.
c) While we have made every effort to ensure that the functions and features under development are free from errors, it is possible that they could contain errors, inaccuracies, and omissions. Unless otherwise provided by valid agreement, we make no warranties of any kind, either implied or express, and exclude all liability for any loss or damage suffered in connection with the use of features and functions under development, to the maximum extent permitted by law, regardless of whether such loss or damage may have been foreseeable.
d) We are not responsible for the accessibility, safety, accuracy, availability, legality, or completeness of information, advertising, commercial offers, products, services, and materials on third-party websites and third-party resources.

About the Document

Revision History

Version	Date	Author	Description
3.0	2012-11-29	Bob DENG	Initial
3.1	2015-05-08	Bob DENG	Added applicable modules.
3.2	2015-11-18	Ablaze LU	Added the function that download the APP bin file and download the bin file to the specified address.
3.3	2020-08-12	Freddy LI	Added switching baud rate protocol and flash data read back protocol. (Chapter 4.3.11–4.3.14)
3.4	2022-03-24	Simon HU	1. Added applicable module M95-R. 2. Deleted M66-DS, M72, M89 and MC90. 3. Deleted CMD_RD_DATA and CMD_RD_DATA_RSP. 4. Added GNSS bin file download function supported by specific firmware version of MC60 module (Chapter 4.3.3). 5. Added the module data length requirement when downloading the GNSS bin file (Chapter 4.3.5).

Contents

Table Index

Figure Index

1 Introduction

Quectel GSM/GPRS/GNSS module series support QuecFOTA® (Firmware Over-The-Air) function, which allows you to upgrade or downgrade the firmware wirelessly. This document mainly describes how to upgrade the firmware of Quectel GSM/GPRS/GNSS module series via QuecFOTA®.

1.1. Applicable Modules

Table 1 : Applicable Modules

Module Series	Module
Mxx	M08-R
	M95-R
	M65
	M66
	M95
MCxx	MC60
	MC65

2 Overview of QuecFOTA®

The following chart illustrates the QuecFOTA procedure.

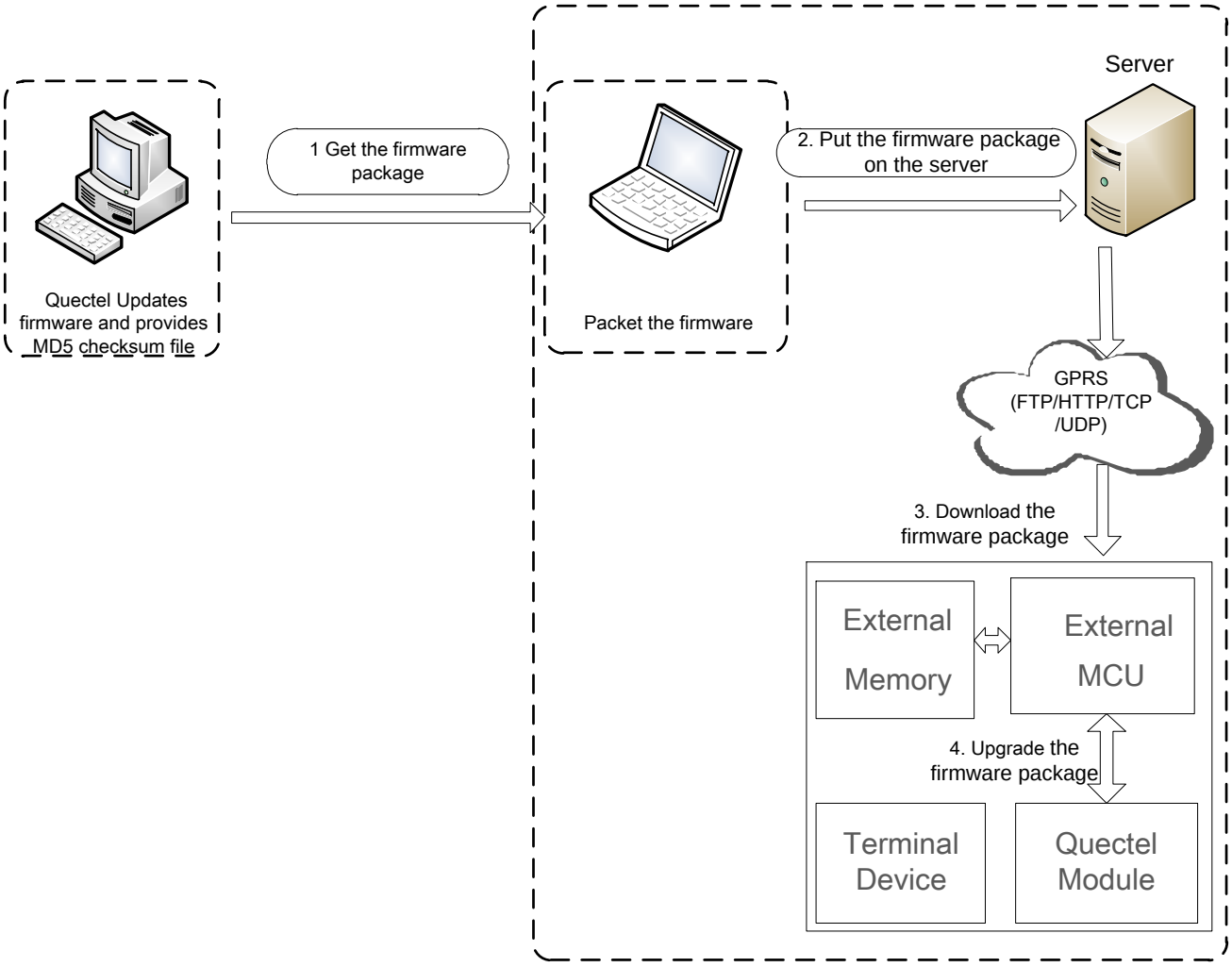

Figure 1: QuecFOTA® Procedure Overview

As shown in the above figure, the following steps need to be performed to upgrade the firmware:

Step 1: Get the firmware package from Quectel.
Step 2: Put the firmware package on the server.
Step 3: Download the firmware package from the server.
Step 4: Upgrade the firmware.

2.1. Get Firmware Package

Before upgrading the module's firmware, send the current firmware version number and the required firmware version number to Quectel. Then Quectel will provide users with the firmware.

Checking the completeness and correctness of the new firmware before upgrading. Package the new firmware by some algorithms. Quectel also provides users with MD5 checksum file and QuceFOTA package tool. See ***Chapter 6*** for QuceFOTA package tool.

2.2. Put Firmware Package on Server

A server needs to be established before using QuecFOTA function. Then put the firmware package on the server, and record the storage path.

2.3. Download Firmware Package

The typical downloading procedures are shown as below:

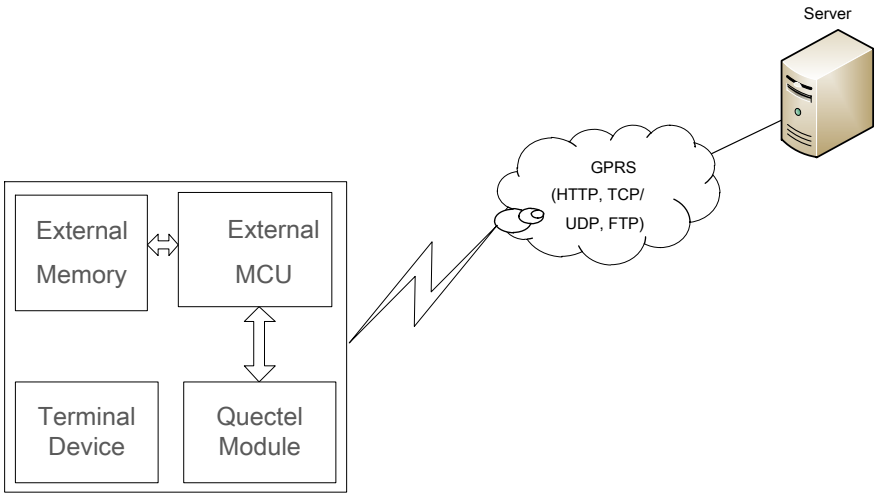

Figure 2: Package Download Procedure

Step 1: The MCU will establish a connection between the Quectel module and the server.
Step 2: The firmware package will be downloaded from servers via TCP/UDP, HTTP or FTP.
Step 3: The firmware package will be stored in external memory of the MCU.
Step 4: The MCU checks the completeness and correctness of the firmware package with checksum and version in the package file, see ***Chapter 6.1***.

2.4. Upgrade Firmware Package

In the upgrade procedure, the MCU will transfer the new firmware from the external memory to the flash of the module. The following figure shows the data road map of the QuecFOTA. See ***Chapter 3*** for the detailed information of QuecFOTA procedure.

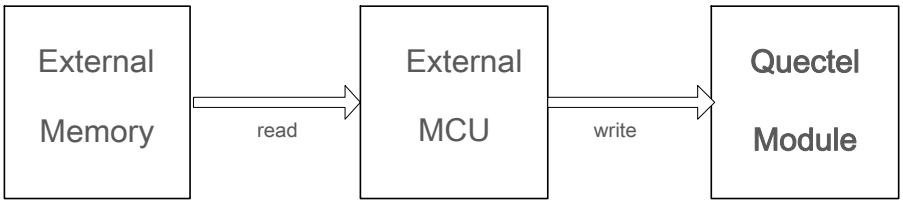

Figure 3: QuecFOTA® Upgrade Data Road Map

The following figure illustrates the procedure of QuecFOTA.

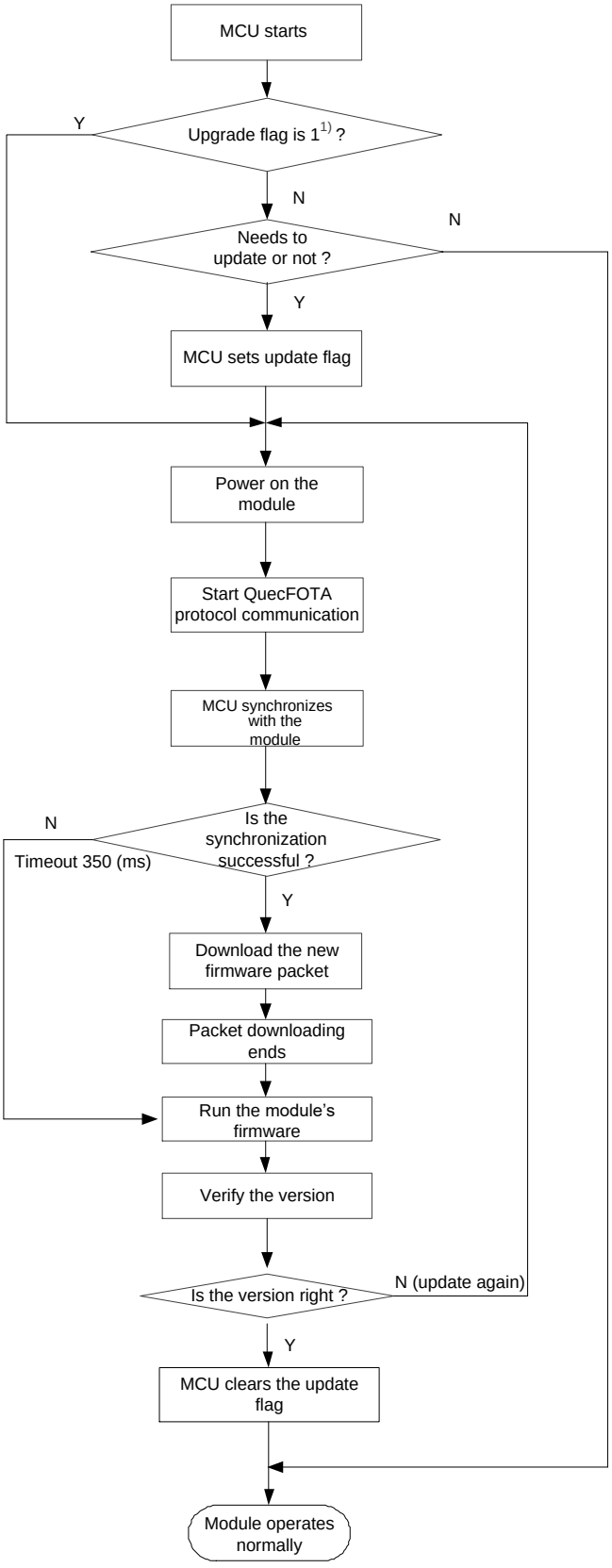

Figure 4: QuecFOTA® Procedure Chart

As shown in the above figure, the following detailed steps need to be performed to update the firmware.

Step 1: The MCU sets upgrade flag.
Step 2: The MCU sends synchronization word.
Step 3: Power on the module and keep the PWRKEY at low level during the upgrading procedure.
Step 4: The module will be in upgrade procedure after receiving the synchronization word sent by MCU.
Step 5: Data exchanging will end once the firmware download is completed.
Step 6: The module runs with new firmware.
Step 7: The MCU checks the module's firmware version.
Step 8: If the firmware version is right, the MCU clears the upgrade flag.
Step 9: The module operates normally.

NOTE

When the upgrade procedure is aborted, the MCU needs to perform the upgrade from 1) in the figure above. See ***Chapter 3.3*** for the abnormal situation during QuecFOTA procedure.

3 QuecFOTA® Procedure

The upgrade procedure includes two steps: The MCU synchronizes with module; download and upgrade the firmware package. The MCU needs to be synchronized with the module and let the module enter command mode. Then the MCU packets the new firmware and sends the package to the module for upgrading.

3.1. MCU Synchronizes with Module

As shown below, the new firmware is stored in the flash of the MCU. The MCU upgrades the module's firmware based on the download protocol via the module's UART interface.

The parameters of UART interface should be configured as bellow:

- Default baud rate: 115200 bps
- Data bit: 8 bits
- Stop bit: 1 bit
- Parity bite: None
- Flow control: None

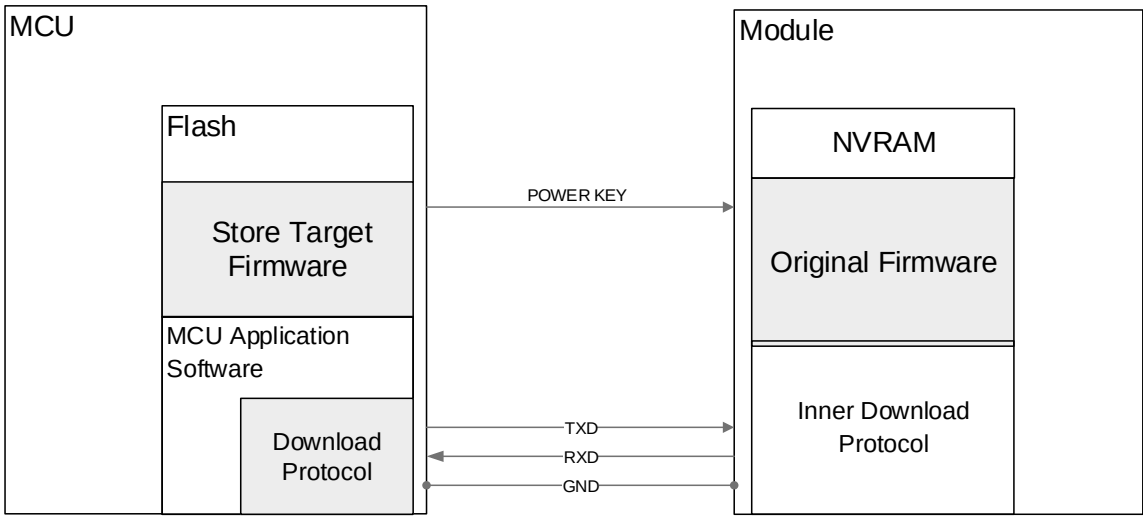

Figure 5: Synchronization Framework of the MCU and Module

The following figure shows the QuecFOTA synchronization procedure. Firstly, the MCU continuously sends SYNC_WORD1 to the module via UART interface at interval of 20 ms. The MCU synchronizes with the module by sending SYNC_WORD signals to the module after module's successful startup. In the meantime, the MCU discards the unexpected data (e.g. 0xB6) sent by the module. After the module responds with SYNC_WORD_RSP1, the MCU sends SYNC_WORD2 to the module. The module responds with SYNC_WORD_RSP2 at the same time. Then the baud rate of the interface should be set as 115200. After that, the module enters into "Command Mode". If the module cannot receive the "synchronous sequence" or finish synchronous negotiation, it runs the firmware that has been stored.

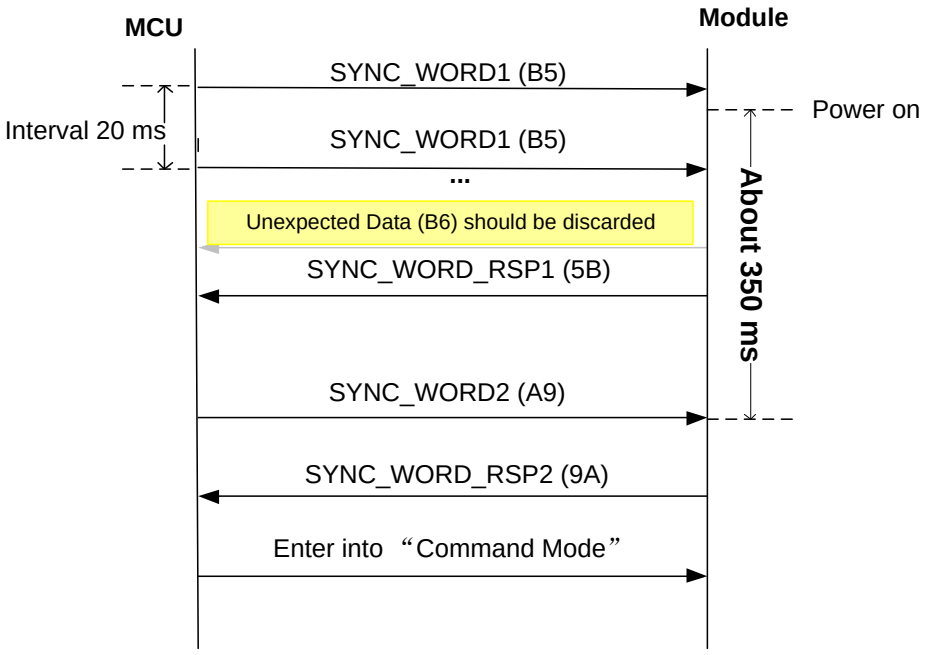

Figure 6: QuecFOTA® Synchronization Sequence

3.2. Download and Upgrade Firmware Package

After the module enters the "Command mode", the MCU packets the new firmware and download it to the module. QuecFOTA package is generated by the MCU and the data frame should follow the package type and format. The ***Chapter 4*** describes the detailed definition of QuecFOTA package.

In the "Command mode", the MCU can send package to upgrade the firmware. If the MCU does not send any commands, the module will be in the "Command mode" all the time. The QuecFOTA sequence is as follows.

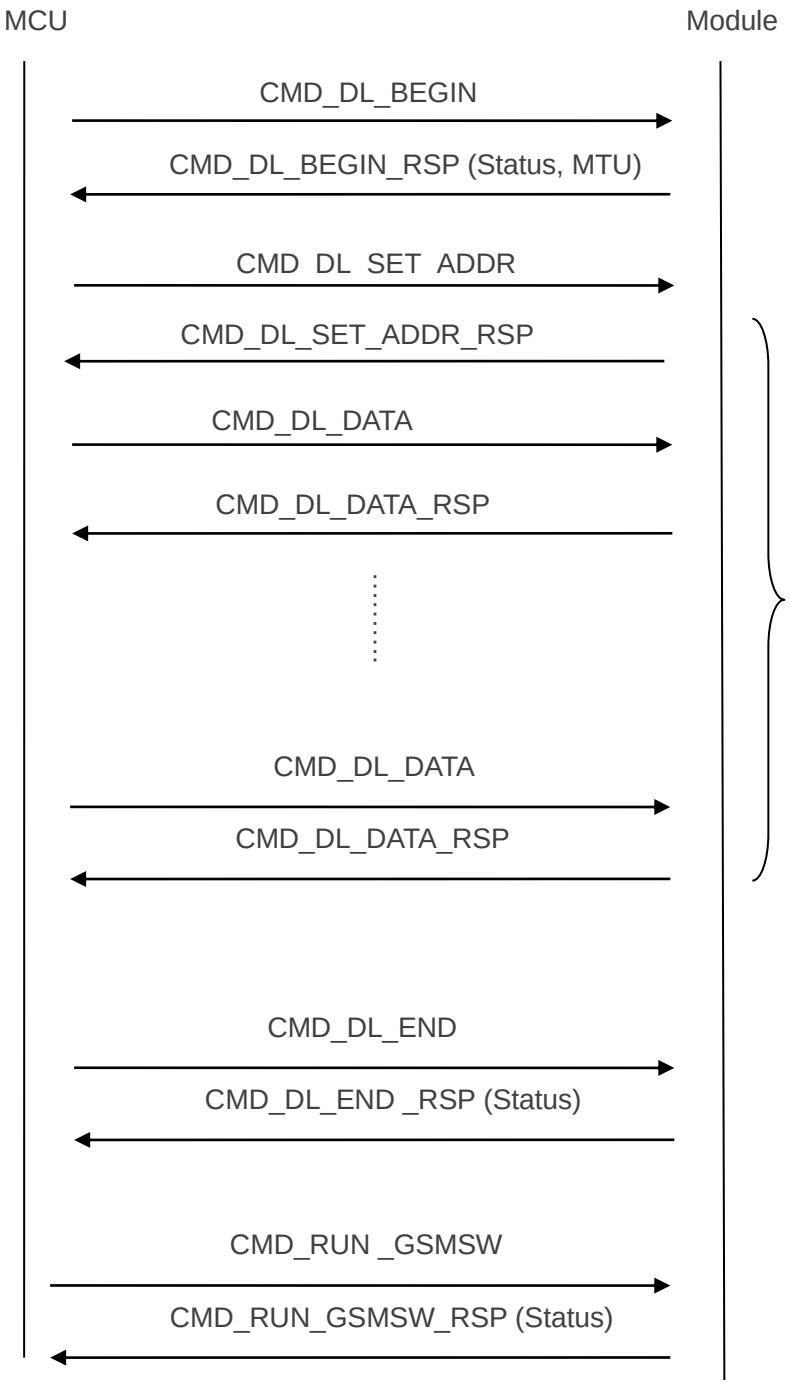

Figure 7: QuecFOTA® Sequence

Step 1: The MCU sends CMD_DL_BEGIN to the module and then module returns the CMD_DL_BEGIN_RSP.

Step 2: The MCU sends CMD_DL_SET_ADDR to the module to specify the download address and then module returns the CMD_DL_SET_ADDR_RSP.

Step 3: The MCU packets the firmware data (including sequence number and data block) into a data package, and then sends the data package to the module. The sequence number starts from 0. After the module receives the data package and writes it to flash successfully, it will return CMD_DL_DATA_RSP (Status = 0). Then MCU can send the next data package.

If the module returns CMD_DL_DATA_RSP (Status = 1 or Status = 4), which indicates the module fails to write the data to flash, MCU shall resend the data package.

If the module returns CMD_DL_DATA_RSP (Status = 2), which means the flash memory has an error, MCU must power on the module and re-upgrade the firmware. See ***Chapter 4.4*** for the status value, and see ***Chapter 3.3*** for failure and error handling.

MCU must read and send the data block of the application firmware in turn. The length of the other data block must be aligned in even-type, except the last data block.

After MCU sends all firmware data package, it needs to send CMD_DL_END to module, which means the firmware download is finished. The module will return CMD_DL_END_RSP and exit from download mode.

After MCU finishes the download process, MCU needs to inform the module to run the new firmware by sending CMD_RUN_GSMSW to module. Then module will return CMD_RUN_GSMSW_RSP and run the firmware.

NOTE

If MCU fails to receive the response message from module in 3 seconds after sending a firmware data package, the MCU should resend it. If MCU does not receive the right response message from the module after sending the same firmware data package for 3 times, MCU must power off the module and restart the upgrade procedure.

3.3. Failure and Error Handling

The firmware upgrade procedure will be interrupted and the module's firmware will be invalid if the following situations occur during QuecFOTA procedure. Moreover, the module cannot work normally and MCU should restart the firmware upgrade procedure illustrated in ***Figure 4***.

- Power supply is interrupted.
- PWRKEY is released during the upgrading procedure.

If the following cases occurred during the upgrading procedure, MCU must perform QuecFOTA error handling flowchart according to the status values. See ***Chapter 4.4*** for the status value.

- CRC16 error
- Flash error
- Module is in download mode
- Data package error
- Command execution failed or invalid command

The following figure is the detailed QuecFOTA error handling flowchart.

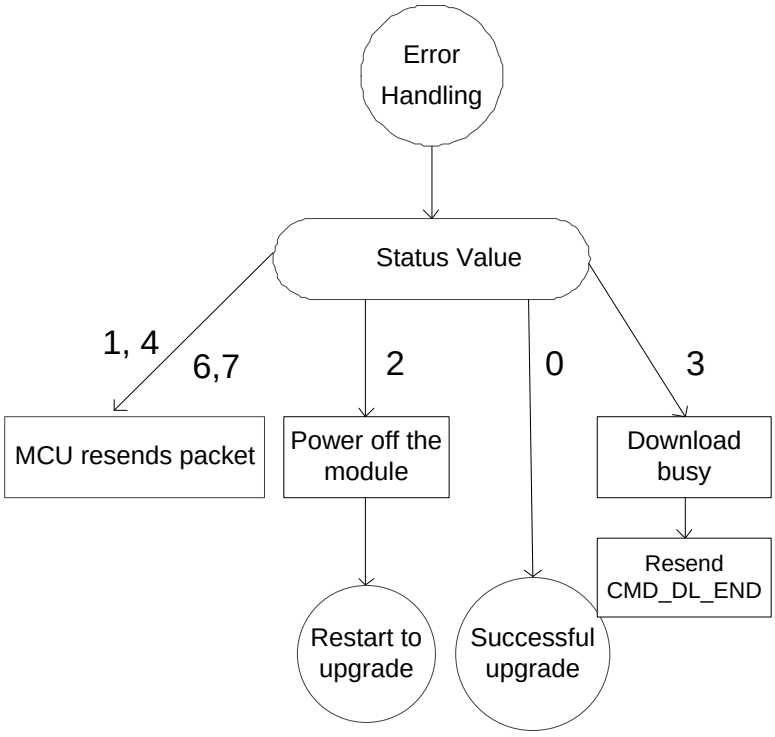

Figure 8: QuecFOTA® Error Handling Flowchart

4 Appendix A Definition of QuecFOTA®

4.1. Format of Package

Table 2: The Format of Package

Head	Type	Length	Data	CRC16
1 byte (0xAA)	2 bytes	2 bytes	N bytes	2 bytes

The value of "length" means the length of the data field, which does not include the length of CRC16 whose length is two bytes. The checksum range consists of "Type" field, "Length" field and "Data" field.

NOTE

CRC16 Polynomial: CRC-16-CCITT x16 + x12 + x5 + 1.

4.2. Frame Protocol Command List

Table 3: Frame Protocol Command List

Type	CMD ID	Description	Direction
CMD_DL_BEGIN	0x0001	Begins to download	MCU to Module
CMD_DL_BEGIN_RSP	0x0002	Responses to CMD_DL_BEGIN	Module to MCU
CMD_DL_SET_ADDR	0x0012	Sets the download address	MCU to Module
CMD_DL_SET_ADDR_RSP	0x0013	Responses to CMD_DL_SET_ADDR	Module to MCU
CMD_DL_DATA	0x0003	Downloads data	MCU to Module
CMD_DL_DATA_RSP	0x0004	Responses to CMD_DL_DATA	Module to MCU

CMD_DL_END	0x0005	Ends Downloading	MCU to Module
CMD_DL_END_RSP	0x0006	Responses to CMD_DL_END	Module to MCU
CMD_RUN_GSMSW	0x0007	Requires to run application firmware	MCU to Module
CMD_RUN_GSMSW_RSP	0x0008	Responses to CMD_RUN_GSMSW	Module to MCU
CMD_CH_BAUDRATE	0x0010	Sets the baud rate	MCU to Module
CMD_CH_BAUDRATE_RSP	0x0011	Responses to CMD_CH_BAUDRATE	Module to MCU

4.3. Description of Frame Protocol Command

4.3.1. CMD_DL_BEGIN

The length of the data field is defined by the data length field. The content of the data field is defined as below:

Content	Bytes	Description
Application firmware version data	4	Reserve

Example:
Application firmware version is 1, and the data package of the command CMD_DL_BEGIN is shown as below:
0xAA 0x00 0x01 0x00 0x04 0x00 0x00 0x00 0x01 0xXX 0xXX

Content	Description
0xAA	Package head
0x00 0x01	CMD_DL_BEGIN command ID
0x00 0x04	Data length
0x00 0x00 0x00 0x01	Application firmware version data. Recommend using the default value
0xXX 0xXX	CRC16 value

4.3.2. CMD_DL_BEGIN_RSP

The length of the data field is defined by the data length field. The content of the data field is defined as below:

Content	Bytes	Description
Status	2	See ***Chapter 4.4***
MTU	2	The maximum length of command package that module can receive

NOTE

1. Status means whether the module receives the download request.
2. MTU means the maximum length of command package that the module received at a time (the length value consists of package head field, command ID field, length field, data field and CRC field).

Example:
The following is the data package of CMD_DL_BEGIN_RSP. Its status is 0 and MTU is 8224 bytes (8 K + 32).
0xAA 0x00 0x02 0x00 0x04 0x00 0x00 0x20 0x20 0xXX 0xXX

Content	Description
0xAA	Package head
0x00 0x02	CMD_DL_BEGIN_RSP command ID
0x00 0x04	Data length
0x00 0x00	Status value, see ***Chapter 4.4***
0x20 0x20	MTU is 8224 in decimal
0xXX 0xXX	CRC16 value

4.3.3. CMD_DL_SET_ADDR

The length of the data field is defined by the data length field. The content of the data field is defined as below:

Content	Bytes	Description
Set the download address	4	0x10000000: Download core bin file 0x20000000: Download app bin file 0x30000000 + [GNSS bin file size]: Download GNSS bin file 0x00xxxxxx: Download the bin file to the specified address

NOTE

GNSS bin file download is supported on specific firmware version of MC60, see the firmware release notes of different modules for details.

Example:
The following is the data package of the command CMD_DL_SET_ADDR which means that the core bin file is downloaded.
0xAA 0x00 0x12 0x00 0x04 0x10 0x00 0x00 0x00 0xXX 0xXX

Content	Description
0xAA	Package head
0x00 0x12	CMD_DL_SET_ADDR command ID
0x00 0x04	Data length
0x10 0x00 0x00 0x00	0x10000000: Download core bin file 0x20000000: Download app bin file 0x30000000 + [GNSS bin file size]: Download GNSS bin file 0x00xxxxxx: Download the bin file to the specified address
0xXX 0xXX	CRC16 value

4.3.4. CMD_DL_SET_ADDR_RSP

The length of the data field is defined by the data length field. The content of the data field is defined as below:

Content	Bytes	Description
Status	2	See ***Chapter 4.4***

Example:
The following is the data package of CMD_DL_SET_ADDR_RSP. Its status is 0.
0xAA 0x00 0x13 0x00 0x02 0x00 0x00 0xXX 0xXX

Content	Description
0xAA	Package head
0x00 0x13	CMD_DL_SET_ADDR_RSP command ID
0x00 0x02	Data length
0x00 0x00	Status value, see ***Chapter 4.4***
0xXX 0xXX	CRC16 value

4.3.5. CMD_DL_DATA

The length of the data field is defined by the data length field. The content of the data field is defined as below:

Content	Bytes	Description
Sequence Number	4	The sequence number of module's data package and begins from 0
Module Data	N - 4 bytes	Module data; N is the data length of the command package

Example:
The following is the data package of the command CMD_DL_DATA which means that the sequence number is 244 and the length of module data is 1024 bytes.
0xAA 0x00 0x03 0x04 0x04 0x00 0x00 0x00 0xF4 0xXX 0xXX 0xXX... (1024 bytes) 0xXX 0xXX

Content	Description
0xAA	Package head
0x00 0x03	CMD_DL_DATA command ID
0x04 0x04	Data length: 1028 bytes. Note that the length includes 4-byte "Sequence number"
0x00 0x00 0x00 0xF4	Sequence number 244
0xXX 0xXX 0xXX...(1024 bytes)	1024 bytes data
0xXX 0xXX	CRC16 value

NOTE

1. Sequence number: 0x00 0x00 0x00 0xF4.
2. Module Data: 0xXX 0xXX 0xXX... (1024 bytes).
3. Module data Length: (Length) N - (Sequence number Length) 4.
4. Module data requires two-byte alignment.
5. If it is not a two-byte alignment, fill *0xff.6*. The total length of package head field, command ID field, length field, sequence number field, data field and CRC16 field does not exceed the MTU.
6. When a GNSS bin file is downloaded, the data length of the module must be 1 K (1024) bytes.

4.3.6. CMD_DL_DATA_RSP

The length of the data field is defined by the data length field. The content of the data field is defined as below:

Content	Bytes	Description
Status	2	See definition list in ***Chapter 4.4***
Next sequence number	4	-

Example:
The data package of CMD_DL_DATA_RSP is shown as below. Its status is 0 and the next sequence number is 245.
0xAA 0x00 0x04 0x00 0x06 0x00 0x00 0x00 0x00 0x00 0xF5 0xXX 0xXX

Content	Description
0xAA	Package head
0x00 0x04	CMD_DL_DATA_RSP command ID
0x00 0x06	Data length
0x00 0x00	Status value, see ***Chapter 4.4***
0x00 0x00 0x00 0xF5	Next sequence number
0xXX 0xXX	CRC16 value

4.3.7. CMD_DL_END

The command does not have data field. The length of data field is 0.

Example:
The data package of the command CMD_DL_END is shown as below:
0xAA 0x00 0x05 0x00 0x00 0xXX 0xXX

Content	Description
0xAA	Package head
0x00 0x05	CMD_DL_END command ID
0x00 0x00	Data length is 0
0xXX 0xXX	CRC16 value

4.3.8. CMD_DL_END_RSP

The length of the data field is defined by the data length field. The content of the data field is defined as below:

Content	Bytes	Description
Status	2	See ***Chapter 4.4***

Example:
The following is the data package of CMD_DL_END_RSP and its status is 0.
0xAA 0x00 0x06 0x00 0x02 0x00 0x00 0xXX 0xXX

Content	Description
0xAA	Package head
0x00 0x06	CMD_DL_END_RSP command ID
0x00 0x02	Data length is 2 bytes
0x00 0x00	Status value, see ***Chapter 4.4***
0xXX 0xXX	CRC16 value

4.3.9. CMD_RUN_GSMSW

The command has no data field. The length of data field is 0.

Example:
The data package of CMD_RUN_GSMSW is shown as below:
0xAA 0x00 0x07 0x00 0x00 0xXX 0xXX

Content	Description
0xAA	Package head
0x00 0x07	CMD_RUN_GSMSW command ID
0x00 0x00	Data length is 0

0xXX 0xXX	CRC16 value

4.3.10. CMD_RUN_GSMSW_RSP

The length of the data field is defined by the data length field. The content of the data field is defined as below:

Content	Bytes	Description
Status	2	See ***Chapter 4.4***

Example:
The following is the data package of CMD_RUN_GSMSW_RSP. Its status is 3.
0xAA 0x00 0x08 0x00 0x02 0x00 0x03 0xXX 0xXX

Content	Description
0xAA	Package head
0x00 0x08	CMD_RUN_GSMSW_RSP command ID
0x00 0x02	Data length is 2 bytes
0x00 0x03	Status value, see ***Chapter 4.4***
0xXX 0xXX	CRC16 value

4.3.11. CMD_CH_BAUDRATE

The length of the data field is defined by the data length field. The content of the data field is defined as below:

Content	Bytes	Description
Set the baud rate	4	0x000012C0: 4800 bps 0x00002580: 9600 bps 0x00004B00: 19200 bps 0x00009600: 38400 bps 0x0000E100: 57600 bps 0x0001C200: 115200 bps

	0x00038400: 230400 bps 0x00070800: 460800 bps 0x000E1000: 921600 bps

Example:
The following is the data package of the command CMD_CH_BAUDRATE which means that the baud rate is set to 921600.
0xAA 0x00 0x10 0x00 0x04 0x00 0x0E 0x10 0x00 0xXX 0xXX

Content	Description
0xAA	Package head
0x00 0x10	CMD_CH_BAUDRATE command ID
0x00 0x04	Data length
0x00 0x0E 0x10 0x00	Baud rate
0xXX 0xXX	CRC16 value

4.3.12. CMD_CH_BAUDRATE_RSP

The length of the data field is defined by the data length field. The content of the data field is defined as below:

Content	Bytes	Description
Status	2	See ***Chapter 4.4***

Example:
The following is the data package of CMD_CH_BAUDRATE_RSP. Its status is 0.
0xAA 0x00 0x11 0x00 0x02 0x00 0x00 0xXX 0xXX

Content	Description
0xAA	Package head
0x00 0x11	CMD_CH_BAUDRATE_RSP command ID
0x00 0x02	Data length

0x00 0x00	Status value, see ***Chapter 4.4***
0xXX 0xXX	CRC16 value

4.4. Definition List

Table 4: Status Value

Status Value	Description	Response
0	Success	-
1	CRC16 error	MCU retransmits the response sequence number.
2	Flash error	MCU restarts module, and downloads the application firmware again.
3	Module is in download mode	-
4	Data package error	MCU retransmits the response sequence number.
6	Command execution failed	MCU retransmits the response sequence number.
7	Invalid CMD	-

5 Appendix B CRC-16 Algorithms

5.1. CRC-16-CCITT Coding Table

```
__align (4) unsigned short CRC16_CCITT_tbl [256] = {
0x0,0x1021,0x2042,0x3063,0x4084,0x50a5,0x60c6,0x70e7,0x8108,0x9129,0xa14a,0xb16b,0xc18c,
0xd1ad,0xe1ce,0xf1ef,0x1231,0x210,0x3273,0x2252,0x52b5,0x4294,0x72f7,0x62d6,0x9339,0x8318,
0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff,
0xe3de,
0x2462,0x3443,0x420,0x1401,0x64e6,0x74c7,0x44a4,0x5485,0xa56a,0xb54b,0x8528,0x9509,0xe5ee,0
xf5cf,0xc5ac,0xd58d,
0x3653,0x2672,0x1611,0x630,0x76d7,0x66f6,0x5695,0x46b4,0xb75b,0xa77a,0x9719,0x8738,0xf7df,0xe
7fe,0xd79d,0xc7bc,
0x48c4,0x58e5,0x6886,0x78a7,0x840,0x1861,0x2802,0x3823,0xc9cc,0xd9ed,0xe98e,0xf9af,0x8948,0x
9969,0xa90a,0xb92b,
0x5af5,0x4ad4,0x7ab7,0x6a96,0x1a71,0xa50,0x3a33,0x2a12,0xdbfd,0xcbdc,0xfbbf,0xeb9e,0x9b79,0x8
b58,0xbb3b,0xab1a,
0x6ca6,0x7c87,0x4ce4,0x5cc5,0x2c22,0x3c03,0xc60,0x1c41,0xedae,0xfd8f,0xcdec,0xddcd,0xad2a,0xb
d0b,0x8d68,0x9d49,
0x7e97,0x6eb6,0x5ed5,0x4ef4,0x3e13,0x2e32,0x1e51,0xe70,0xff9f,0xefbe,0xdfdd,0xcffc,0xbf1b,0xaf3a,
0x9f59,0x8f78,
0x9188,0x81a9,0xb1ca,0xa1eb,0xd10c,0xc12d,0xf14e,0xe16f,0x1080,0xa1,0x30c2,0x20e3,0x5004,0x4
025,0x7046,0x6067,
0x83b9,0x9398,0xa3fb,0xb3da,0xc33d,0xd31c,0xe37f,0xf35e,0x2b1,0x1290,0x22f3,0x32d2,0x4235,0x5
214,0x6277,0x7256,
0xb5ea,0xa5cb,0x95a8,0x8589,0xf56e,0xe54f,0xd52c,0xc50d,0x34e2,0x24c3,0x14a0,0x481,0x7466,0x
6447,0x5424,0x4405,
0xa7db,0xb7fa,0x8799,0x97b8,0xe75f,0xf77e,0xc71d,0xd73c,0x26d3,0x36f2,0x691,0x16b0,0x6657,0x7
676,0x4615,0x5634,
0xd94c,0xc96d,0xf90e,0xe92f,0x99c8,0x89e9,0xb98a,0xa9ab,0x5844,0x4865,0x7806,0x6827,0x18c0,0
x8e1,0x3882,0x28a3,
0xcb7d,0xdb5c,0xeb3f,0xfb1e,0x8bf9,0x9bd8,0xabbb,0xbb9a,0x4a75,0x5a54,0x6a37,0x7a16,0xaf1,0x1
ad0,0x2ab3,0x3a92,
0xfd2e,0xed0f,0xdd6c,0xcd4d,0xbdaa,0xad8b,0x9de8,0x8dc9,0x7c26,0x6c07,0x5c64,0x4c45,0x3ca2,0x
2c83,0x1ce0,0xcc1,
0xef1f,0xff3e,0xcf5d,0xdf7c,0xaf9b,0xbfba,0x8fd9,0x9ff8,0x6e17,0x7e36,0x4e55,0x5e74,0x2e93,0x3eb2
,0xed1,0x1ef0};
```

5.2. Calculate the CRC Value

```
void calculate_crc16(unsigned char*      aData, unsigned short  aSize, unsigned char*      Higher,
unsigned char*    Lower)
{
      unsigned short    i;
      unsigned short nAccum = 0;

      for ( i = 0; i < aSize; i++ )
            nAccum  =  ( nAccum  <<  8  )  ^  ( unsigned  short  )CRC16_CCITT_tbl[(( nAccum  >>  8  )  ^
*aData++)&0xff];

      *Higher = (unsigned char)((nAccum>>8) & 0xff);
      *Lower = (unsigned char)((nAccum) & 0xff);
}
unsigned long CalculateCRC16(unsigned char*ptr,    unsigned long len)
{
    unsigned char Higher = 0;
    unsigned char Lower = 0;
    calculate_crc16(ptr,len,&Higher,&Lower);
    return (((0x00000000 | Higher) << 8) | Lower);
}
```

5.3. Example

```
      unsigned long new_CRC16_value = 0;
      new_CRC16_value = CalculateCRC16 ((unsigned char*)pData, nLength);
      if(org_CRC16_value != new_CRC16_value)
      {
            //CRC checksum failed
            return -1;
      }
      else
      {
            //CRC checksum successful
            return 0;
      }
```

6 Appendix C QuecFOTA® Package Tool

6.1. The Format of QuecFOTA Package Tool

In the process of download and copy, the firmware may be damaged accidentally. Hence, it is recommended to package the new firmware as the following format with QuecFOTA package tool before uploading the new firmware to the server.

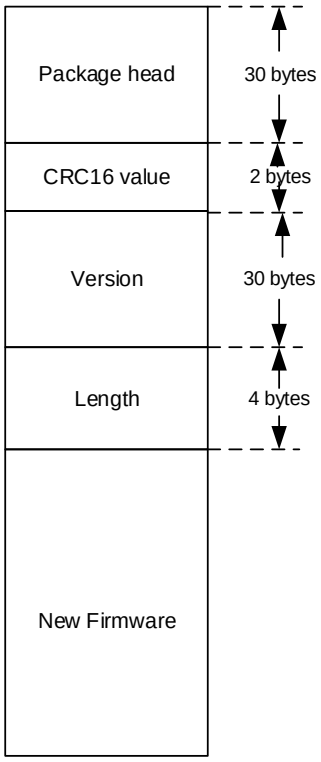

Figure 9: QuecFOTA® Package Format

It is necessary to check the completeness and correctness of the new firmware before downloading. The "Version" field can be used to check the correctness of the new firmware version. The "CRC16" field can be used to check the correctness and completeness of the downloaded package after downloading. The CRC16 is calculated with the CRC16 algorithm as ***Appendix B*** in ***Chapter 5***.

Content	Length (bytes)	Description	Example
Package Head	30	Package head	"QuectFOTAPackageV0.1\0" If it is empty, set it to 0.
CRC16	2	CRC16 checksum value	Including "Version", "Length" and "New Firmware".
Version	30	Version of the new firmware	M10ER01A08W32\0 If it is empty, set it to 0.
Length	4	Size of the new firmware	-
New Firmware	Length	Entire bin file	M10ER01A08W32.BIN.

6.2. The Usage of QuecFOTA Package Tool

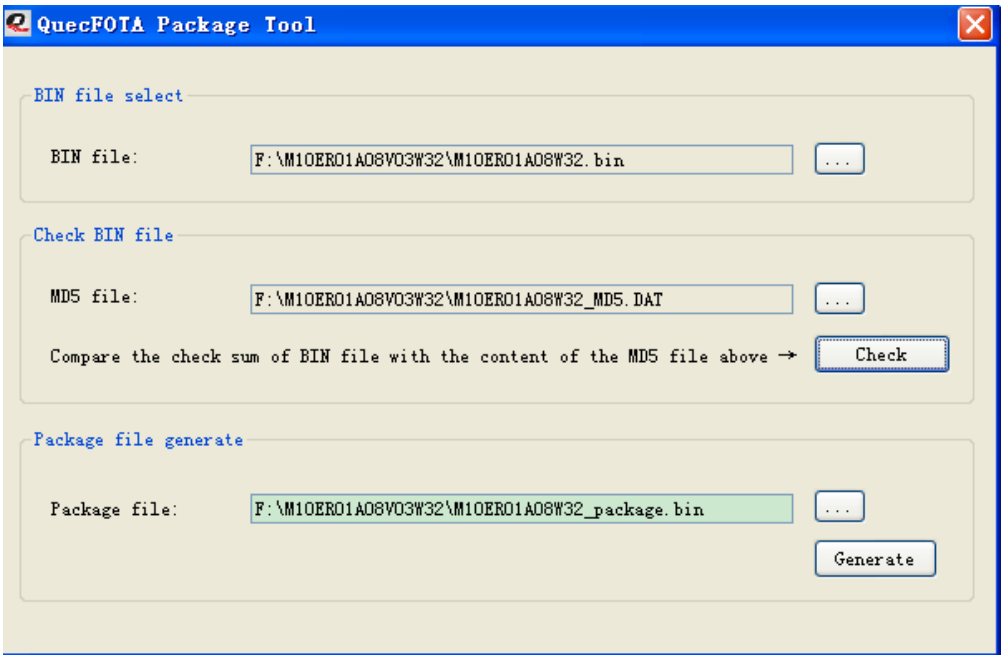

Figure 10: QuecFOTA® Package Tool

The usage of QuecFOTA package tool consists of three steps (take the *M10ER01A08W32.bin* file as an example):

Step 1: Select the bin file.

Step 2: Check the bin file.

Check whether the bin is correct or not with MD5 file. For example, select the "M10ER01A08W32_MD5.DAT", then click "Check" button to check whether the *M10ER01A08W32.bin* is correct or not.

Step 3: "Package file generation", namely, generate the QuecFOTA package file.

First, type "*M10ER01A08W32_package.bin*", then click "Generate" button to generate the *M10ER01A08W32_package.bin* file with the *M10ER01A08W32.bin* file.

7 Appendix D Reference

Table 5: Terms and Abbreviations

Abbreviation	Description
CRC	Cyclic Redundancy Check
MD5	Message-Digest Algorithm
FOTA	Firmware Over-The-Air
FTP	File Transfer Protocol
GSM	Global System for Mobile Communications
GNSS	Global Navigation Satellite System
HTTP	Hyper Text Transfer Protocol
ID	Mostly refers to Identifier in terms of software
MCU	Microcontroller Unit
MTU	Maximum Transmission Unit
NVRAM	Non-Volatile Random Access Memory
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UART	Universal Asynchronous Receiver/Transmitter