RTL8762C OTA User Manual

V1.01

2018/06/11

Revision History

Date	Version	Comments	Author	Reviewer
2018/06/09	V1.0	First release version	Ken	
2018/06/11	V1.0.1	Modify Indications for image version.		
2018/09/12	V1.1	Correction, formatting		Astor

Contents

Table List

1 Overview

1.1 Function Description

OTA (Over The Air) represents the technology that apply Bluetooth to update image (code and data) that runs in RTL8762C Flash.

Note: This document is also applied to RLT8752 series.

1.2 Related Emphasis

1. Flash Layout
2. IMG format
3. Package
4. Ota Protocol

2 Flash layout

Flash layout of RTL8762C consists of OEM Config, OTA Bank0, OTA Bank1, FLASH Transport Layer(FTL), OTA TMP and APP defined section, as shown in Figure 2.1. Start address for accessing Flash is 0x800000.

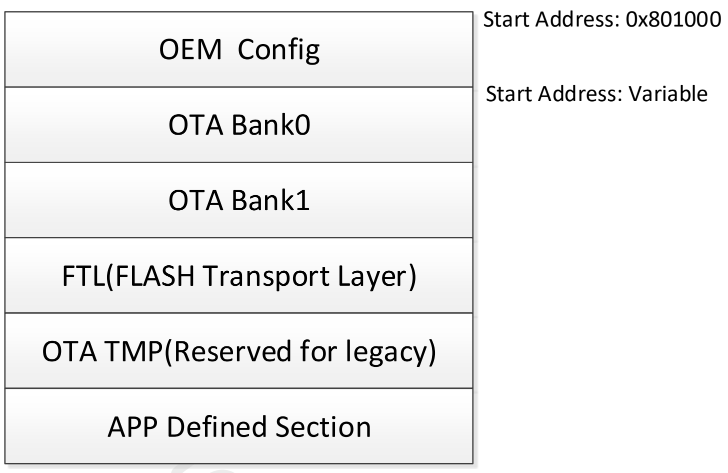


Figure 2.1: Flash Layout

Memory and corresponding function of Flash is shown in Table 2.1

Memory Segment	Starting Address	Size (Bytes)	Functions
OEM Config	0x801000	0x1000	Storage of Config information, including Bluetooth address, AES Key and Customizable Flash Layout.
OTA Bank 0	Variable (defined in OEM Config)	Variable length (defined in OEM Config)	If not in bank switching mode, this region contains the project data and codes to be executed, including OTA Header, Secure boot, Patch, APP, Data1, Data2. OTA_TMP is the backup region of this OTA. In bank switching mode, OTA Bank 0 and OTA Bank 1 is backup region of each other. Suppose OTA Bank 0 is execution region, then OTA Bank 1 is backup region.

OTA Bank 1	Variable (defined in OEM Config)	Variable length (defined in OEM Config)	This region only exists when bank switching method is applied. It has same functions and same size with Bank 0 in bank switching mode
FTL	Variable (defined in OEM Config)	Variable length (defined in OEM Config)	A software technology that access Flash with logical address. Customer no longer needs to focus on operations on Flash physical layer. This region also balances consumption.
OTA_TMP	Variable (defined in OEM Config)	Variable length (defined in OEM Config)	Used as backup region of OTA if not in bank switching mode. Its size should be no less than largest image in OTA Bank 0.
APP Defined Section	Variable (defined in OEM Config)	Variable length (defined in OEM Config)	The rest of Flash that can be customized. This region cannot be managed by OTA scheme.

Table 2.1 FLASH Memory and Function Description

OTA bank layout is shown in Figure 2.2, and description for each part is shown in Table 2.2.

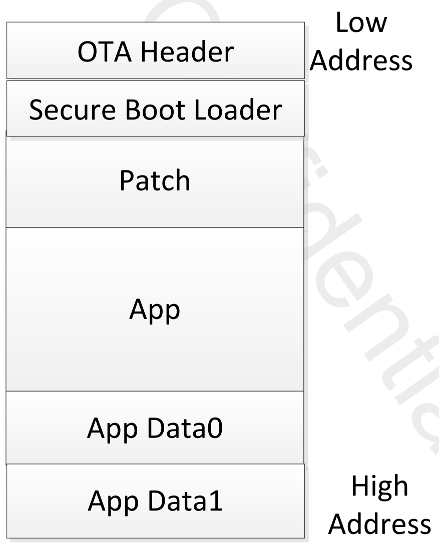


Figure 2.2 Layout of Bank 0/1

Memory Segment	**Starting Address**	**Size**	**Functions**
OTA Header	Determined in the OEM Config	4KB	This region contains the OTA Header version and start address and size of the images in the bank

	region		
Secure Boot Loader	Determined in the OTA Header region	Variable	This region contains secure boot loader.
Patch	Determined in the OTA Header region	Variable	This region contains the code that optimize and extend the protocol stack and system in ROM.
App	Determined in the OTA Header region	Variable	This region contains project code.
App Data0	Determined in the OTA Header region	Variable	Data region used in project.
App Data1	Determined in the OTA Header region	Variable	Data region used in project.

Table 2.2 Flash Segmentation

3 Image Header Format

OTA Header image is made up of header (1KB) and dummy payload (3KB). OTA Header is generated by MPPackTool. Different fields of header are shown in Figure 2.3.

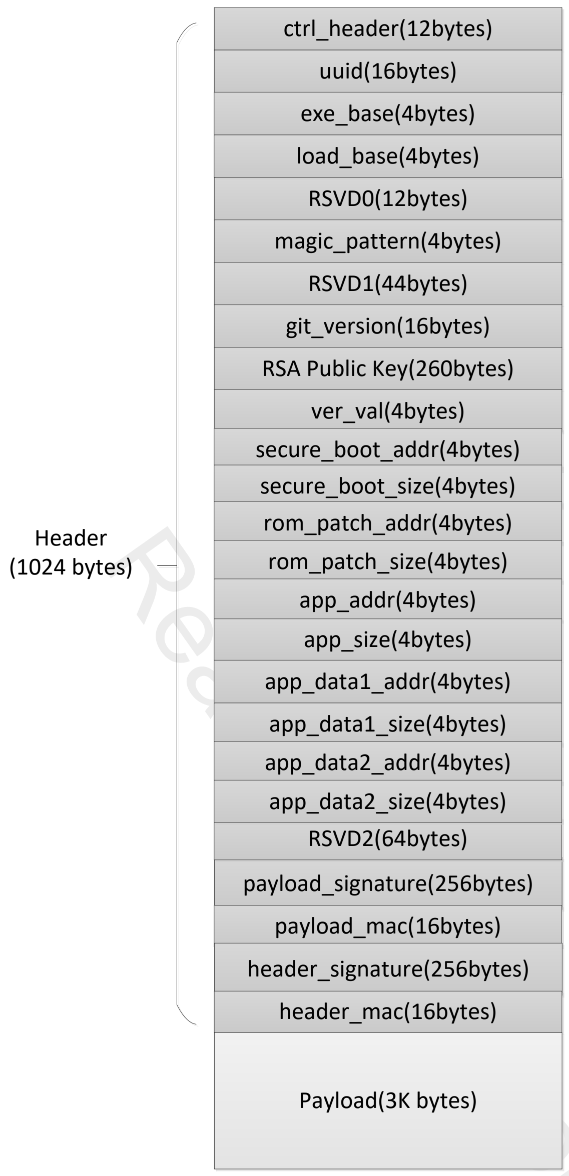


Figure 3.1: OTA Header Format

Header fields and corresponding functions are shown in Table 2.3

Fields	Length (Byte)	Functions
ctrl_header	12	Control message of OTA Header
secure_boot_addr	4	Start address of secure boot image
secure_boot_size	4	Size of secure boot image
rom_patch_addr	4	Start address of ROM patch image

rom_patch_size	4	Size of ROM patch image
app_addr	4	Start address of application image
app_size	4	Size of application image
app_data1_addr	4	Start address of application data1
app_data1_size	4	Size of application data1
app_data2_addr	4	Start address of application data2
app_data2_size	4	Size of application data2

Table 3.3: Fields of OTA Header

Image of patch, APP and App data is made up of image header (1KB) and corresponding payload. Image header of patch and APP is generated while compiling and linking, and that of App data is added by APP DATA Tool. Header fields are shown in Figure 2.4, and corresponding functions are shown in Table 2.4

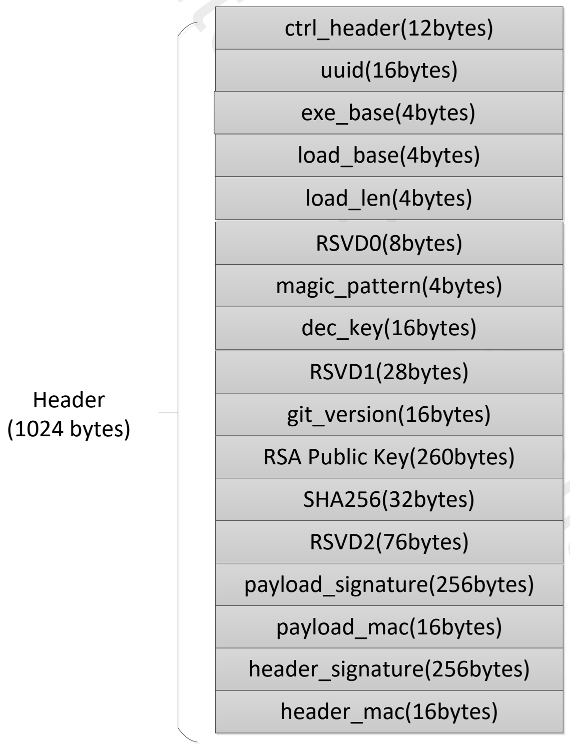


Figure 3.4: Image Header Layout

Fields	Length(Byte)	Functions
ctrl_header	12	Control information field of Image Header
git_version	16	Information field of version control

Table 3.4: Image Header Field

ctrl_header format in Image Header is shown as follows:

```
    typedef struct _IMG_CTRL_HEADER_FORMAT
    {
        uint8_t ic_type;
        uint8_t secure_version;
        union
        {
            uint16_t value;
            struct
            {
                uint16_t xip: 1; // payload is executed on flash
                uint16_t enc: 1; // all the payload is encrypted
                uint16_t load_when_boot: 1; // load image when boot
                uint16_t enc_load: 1; // encrypt load part or not
                uint16_t enc_key_select: 3; // referenced to ENC_KEY_SELECT
                uint16_t not_ready : 1; //for copy image in ota
                uint16_t not_obsolete : 1; //for copy image in ota
                uint16_t integrity_check_en_in_boot : 1; // enable image integrity
check in boot flow
                uint16_t rsvd: 6;
            };
        } ctrl_flag;
        uint16_t image_id;
        uint16_t crc16;
        uint32_t payload_len;
    } T_IMG_CTRL_HEADER_FORMAT;
```

ic_type represents IC type, which has the value of 5 when RTL8762C/RTL8752 chip is used. secure_version indicates version of secure boot image.

image_id identifies different types of image, among which SCCD, OCCD and FactoryCode cannot be updated through OTA. The types are enumerated in IMG_ID.

```
typedef enum _IMG_ID
{
    SCCD        = 0x278D,
    OCCD       = 0x278E,
    FactoryCode   = 0x278F,
    OTA         = 0x2790, /* OTA header */
    SecureBoot    = 0x2791,
    RomPatch     = 0x2792,
    AppPatch     = 0x2793,
    AppData1     = 0x2794,
    AppData2     = 0x2795,
    IMAGE_MAX   = 0x2796,
} IMG_ID;
```

payload_length represents the size of image in byte, excluding 1KB image header.

crc16 indicates check method, which can be crc check and SHA256 check. 0 represents crc check and 1 represents SHA256 check.

ctrl_flag and OTA related bit field can be not_ready and not_obsolete. not_ready indicates whether OTA write is successfully completed and its default value is 0. When image is about to be written into backup region, not_ready will be set to 1 at first. Not until update transmission is completed and integrity check is passed will the not_ready flag be set to 0 to indicate that image is ready.

not_obsolete indicates if the image should be abandoned and its default value is 1. This parameter is invalid in bank switching mode. When not in bank switching mode, if not_ready is read 0 and not_obsolete is read 1, image will be moved from OTA_TMP region to specified region (APP region, Patch region or App data region). not_obsolete flag will be written 0 after transfer completed.

4 Package and flash layout sample

4.1 Support bank switching

Related tool and their functions:

- FlashMapGenerateTool: Generate flash map.ini and flash_map.h, flash_map.h should be put in same directory with project and generate APP Image. flash map.ini is the input file of MPPackTool and MPTool to ensure image has the same address with the address in settings.
- MPPackTool: Package OTA files.
- MPTool: Perform address conversion operation on Patch.

4.1.1 FLASH Layout

Bank switching method needs 2 OTA Banks that are completely same to become the backup of each other. Its's advantage is that program can directly jump to new bank when reboot. However, OTA update in bank switching mode takes more flash memory to speed up update, so the size of flash memory should be larger if bank switching method is applied.

If flash size is comparatively large, user can update firmware by applying bank switching method. Take 1 MB Flash as example, the suggested Flash layout is shown below:

sample layout for flash(total size = 1MB)	size	start address
1) SOCV Header	4K	0x800000
2) OEM Header	4K	0x801000
3) OTA Bank0	400K	0x802000
a) OTA Header	4K	
b) Secure boot loader	16K	0x803000
c) Patch code	40K	0x807000
d) APP code	160K	0x811000
e) APP data1	180K	0x839000
f) APP data2	0K	
4) OTA Bank1 (same as OTA Bank0)	400K	0x866000
a) OTA Header	4K	

b) Secure boot loader	16K	0x867000
c) Patch code	40K	0x86B000
d) APP code	160K	0x875000
e) APP data1	180K	0x89D000
f) APP data2	0K	
5) FTL	16K	0x8A1000
6) OTA Temp	0K	
7) APP Defined Section	200K	

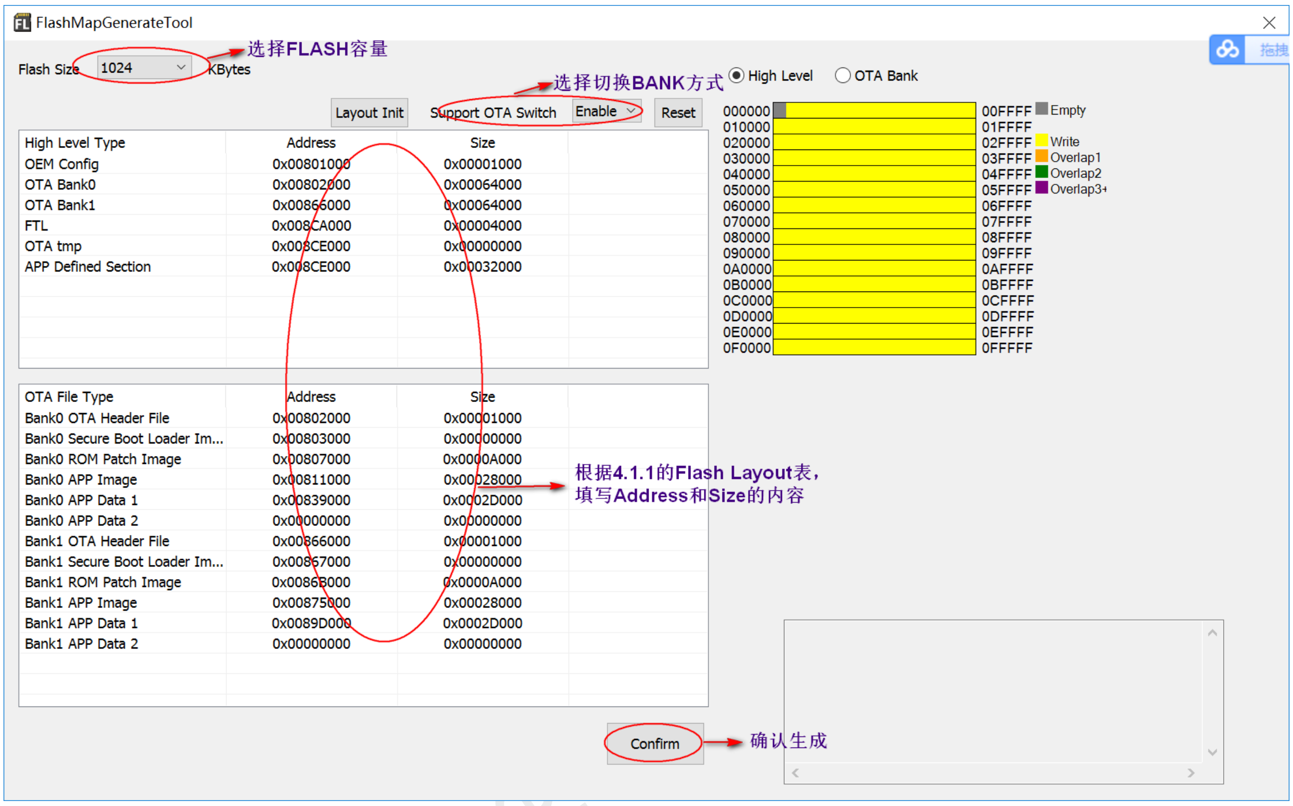
Table 4.1: FLASH Layout Sample

Note: Flash Layout should be determined based on actual size of image and data.

4.1.2 Usage of package tool with bank switching

1. Use FlashMapGenerateTool to ‘flash map.ini’ and ‘flash map.h’. Copy ‘flash map.h’ to project directory and open project with Keil. Link and compile the project to generate “app_MP_sdk#####+version+MD5.bin” file for packaging. To apply Bank switching method, “mem_config.h” in project directory should be modified.

```
/** @brief set app bank to support OTA: 1 is ota bank1, 0 is ota bank0 */
    #define APP_BANK                                        0
```



Flash Size 1024 KBytes

[✓ High Level] OTA Bank

Support OTA Switch: Enable

High Level Type	Address	Size
OEM Config	0x00801000	0x00001000
OTA Bank0	0x00802000	0x00064000
OTA Bank1	0x00866000	0x00064000
FTL	0x008CA000	0x00004000
OTA tmp	0x008CE000	0x00000000
APP Defined Section	0x008CE000	0x00032000

OTA File Type	Address	Size
Bank0 OTA Header File	0x00802000	0x00001000
Bank0 Secure Boot Loader Im...	0x00803000	0x00000000
Bank0 ROM Patch Image	0x00807000	0x0000A000
Bank0 APP Image	0x00811000	0x00028000
Bank0 APP Data 1	0x00839000	0x0002D000
Bank0 APP Data 2	0x00000000	0x00000000
Bank1 OTA Header File	0x00866000	0x00001000
Bank1 Secure Boot Loader Im...	0x00867000	0x00000000
Bank1 ROM Patch Image	0x0086B000	0x0000A000
Bank1 APP Image	0x00875000	0x00028000
Bank1 APP Data 1	0x0089D000	0x0002D000
Bank1 APP Data 2	0x00000000	0x00000000

根据4.1.1的Flash Layout表，填写Address和Size的内容

000000	00FFFF
010000	01FFFF
020000	02FFFF
030000	03FFFF
040000	04FFFF
050000	05FFFF
060000	06FFFF
070000	07FFFF
080000	08FFFF
090000	09FFFF
0A0000	0AFFFF
0B0000	0BFFFF
0C0000	0CFFFF
0D0000	0DFFFF
0E0000	0EFFFF
0F0000	0FFFFF

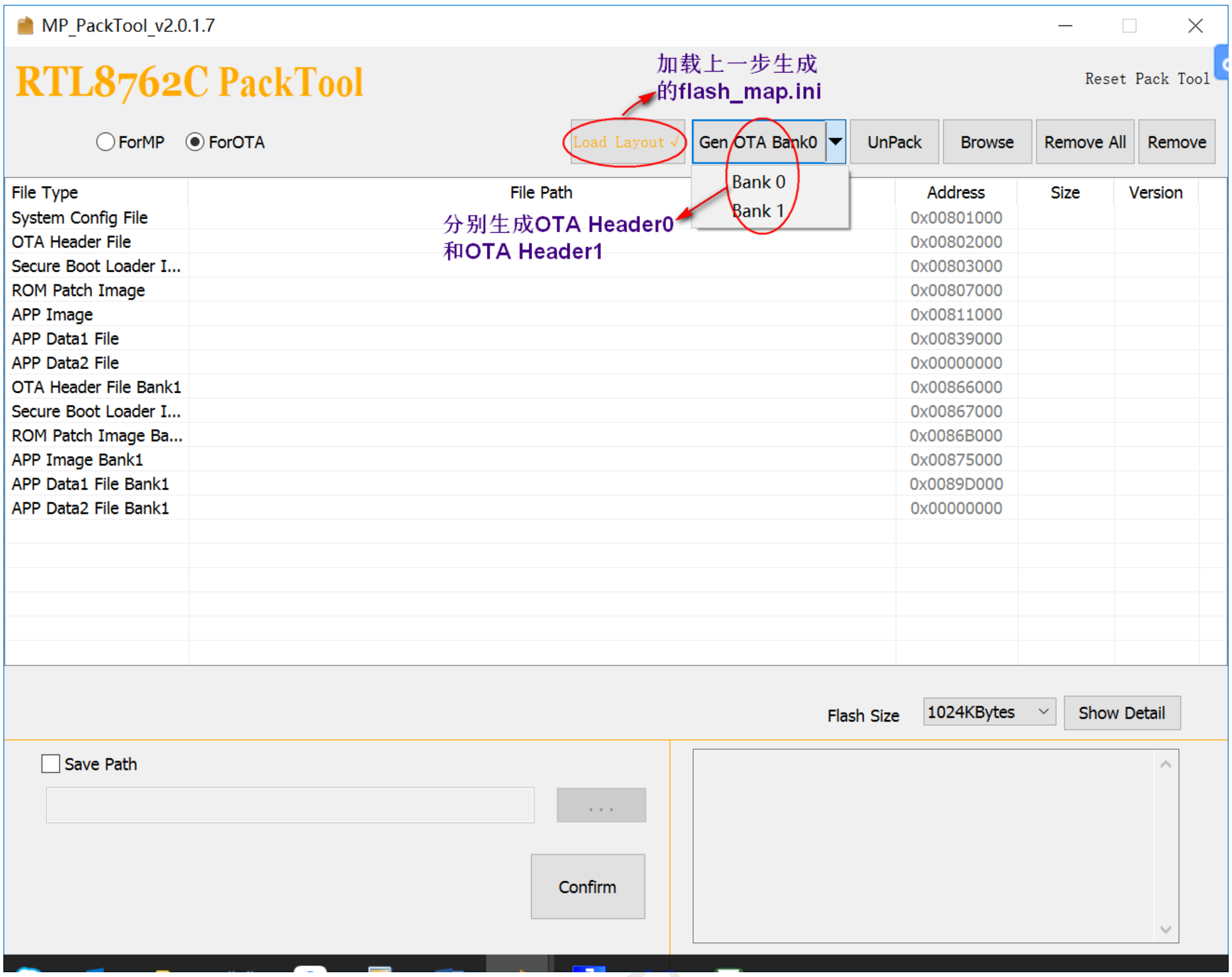
⬜ Empty
🟨 Write
🟧 Overlap1
🟩 Overlap2
🟪 Overlap3+

Confirm

Figure 4.1 Generate Flash Layout

Note: The 'flash map.ini' generated should keep consistent with the one used in mass production.

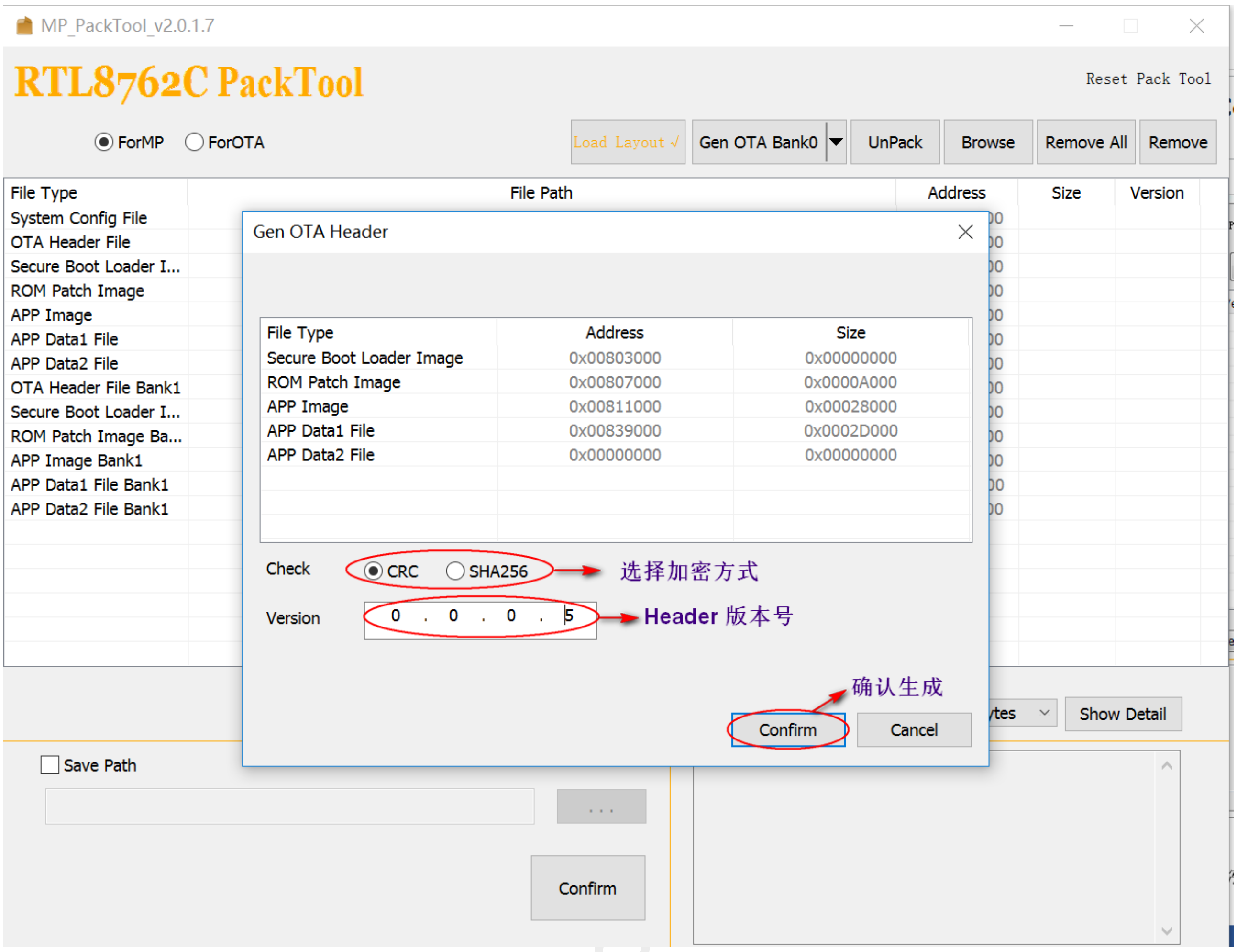
2. Open MP_PackTool, load the 'flash map.ini' file generated in step 1 and generate OTA Header0, OTA Header1 separately.



File Type	File Path	Address	Size	Version
System Config File		0x00801000		
OTA Header File		0x00802000		
Secure Boot Loader I...		0x00803000		
ROM Patch Image		0x00807000		
APP Image		0x00811000		
APP Data1 File		0x00839000		
APP Data2 File		0x00000000		
OTA Header File Bank1		0x00866000		
Secure Boot Loader I...		0x00867000		
ROM Patch Image Ba...		0x0086B000		
APP Image Bank1		0x00875000		
APP Data1 File Bank1		0x0089D000		
APP Data2 File Bank1		0x00000000		

Figure 4.2 MP PACK Tool Load Flash Layout

3. Generate OTA Header0 and OTA Header1. Take OTA Header0 as an example, as shown below. Generating OTA Header1 follows the same procedure.



MP_PackTool_v2.0.1.7

RTL8762C PackTool

Reset Pack Tool

(● ForMP) ○ ForOTA

File Type	File Path	Address	Size	Version
System Config File				
OTA Header File				
Secure Boot Loader I...				
ROM Patch Image				
APP Image				
APP Data1 File				
APP Data2 File				
OTA Header File Bank1				
Secure Boot Loader I...				
ROM Patch Image Ba...				
APP Image Bank1				
APP Data1 File Bank1				
APP Data2 File Bank1				

Gen OTA Header

File Type	Address	Size
Secure Boot Loader Image	0x00803000	0x00000000
ROM Patch Image	0x00807000	0x0000A000
APP Image	0x00811000	0x00028000
APP Data1 File	0x00839000	0x0002D000
APP Data2 File	0x00000000	0x00000000

Check [✓ CRC] SHA256 → 选择加密方式

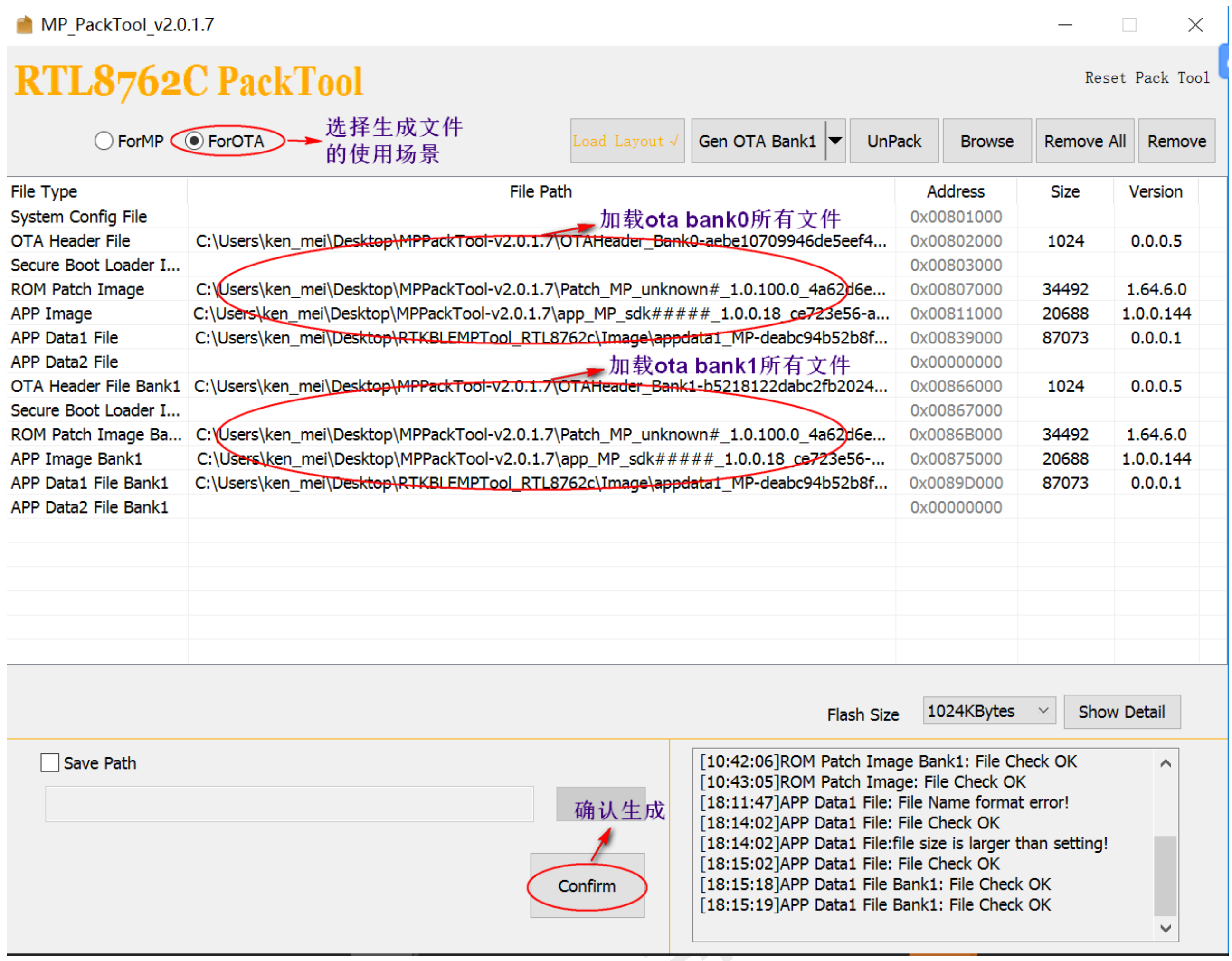
Version 0 . 0 . 0 . 5 → Header 版本号

确认生成

Figure 4.3 Generate OTA Header

Note: Only if the version number of OTA Header to be packaged is higher than that of current version, can new bank be valid after OTA.

4． Generate packet file of ImgPacketFile-xxxxxx.bin in current directory, which is used for updating.



ForMP [✓ ForOTA]

File Type	File Path	Address	Size	Version
System Config File		0x00801000		
OTA Header File	C:\Users\ken_mei\Desktop\MPPackTool-v2.0.1.7\OTAHeader_Bank0-aebe10709946de5eef4...	0x00802000	1024	0.0.0.5
Secure Boot Loader I...		0x00803000		
ROM Patch Image	C:\Users\ken_mei\Desktop\MPPackTool-v2.0.1.7\Patch_MP_unknown#_1.0.100.0_4a62d6e...	0x00807000	34492	1.64.6.0
APP Image	C:\Users\ken_mei\Desktop\MPPackTool-v2.0.1.7\app_MP_sdk#####_1.0.0.18_ce723e56-a...	0x00811000	20688	1.0.0.144
APP Data1 File	C:\Users\ken_mei\Desktop\RTKBLEMPTool_RTL8762c\Image\appdata1_MP-deabc94b52b8f...	0x00839000	87073	0.0.0.1
APP Data2 File		0x00000000		
OTA Header File Bank1	C:\Users\ken_mei\Desktop\MPPackTool-v2.0.1.7\OTAHeader_Bank1-b5218122dabc2fb2024...	0x00866000	1024	0.0.0.5
Secure Boot Loader I...		0x00867000		
ROM Patch Image Ba...	C:\Users\ken_mei\Desktop\MPPackTool-v2.0.1.7\Patch_MP_unknown#_1.0.100.0_4a62d6e...	0x0086B000	34492	1.64.6.0
APP Image Bank1	C:\Users\ken_mei\Desktop\MPPackTool-v2.0.1.7\app_MP_sdk#####_1.0.0.18_ce723e56-...	0x00875000	20688	1.0.0.144
APP Data1 File Bank1	C:\Users\ken_mei\Desktop\RTKBLEMPTool_RTL8762c\Image\appdata1_MP-deabc94b52b8f...	0x0089D000	87073	0.0.0.1
APP Data2 File Bank1		0x00000000		

Flash Size 1024KBytes

```
[10:42:06]ROM Patch Image Bank1: File Check OK
[10:43:05]ROM Patch Image: File Check OK
[18:11:47]APP Data1 File: File Name format error!
[18:14:02]APP Data1 File: File Check OK
[18:14:02]APP Data1 File:file size is larger than setting!
[18:15:02]APP Data1 File: File Check OK
[18:15:18]APP Data1 File Bank1: File Check OK
[18:15:19]APP Data1 File Bank1: File Check OK
```

Figure 4.4 Package to generate PACK

Note: 1. Both OTA Head0 and OTA Header1 need to be packaged to PACK, different from the mode without bank switching.

2. All the contents defined in Flash layout need to be packaged, especially Header, Patch and APP of OTA.

3. It is recommended that package both bank0 and bank1 in PACK.

4. Patch image address needs to be converted through RTKBLEMPTool.

*5. APP DATA file is generated with APP DATA generation script, for detailed information refer to **Bee2 Tools User Guide**.*

4.2 Do not support bank switching

4.2.1 FLASH Layout

The differences between the method without bank switching and the one with bank switching are:

1. OTA Bank1 region needn't be allocated.
2. OTA Temp region needs to be allocated and its size should be no less than the largest image in OTA Bank0.

Thus, the method without bank switching saves more flash. After OTA transmission is completed and program is rebooted, the data in OTA Temp region will be moved to the image region specified by OTA Bank0. The data won't be valid until program is rebooted, which increase the duration of update.

The suggested Flash layout is shown below:

sample layout for flash(total size = 256KB)	size	start addr
1) OEM Header	4K	0x801000
2) OTA Bank0	140K	0x802000
a) OTA Header	4K	
b) Secure boot loader	4K	0x80D000
c) Patch code	40K	0x803000
d) APP code	92K	0x80E000
e) APP data1	0K	0x825000
f) APP data2	0K	
3) OTA Bank1 (same as OTA Bank0)	0K	0x825000
4) FTL	16K	0x825000
5) OTA Temp	92K	0x829000
6) APP Defined Section	0K	

Table 4-2：FLASH Layout Sample

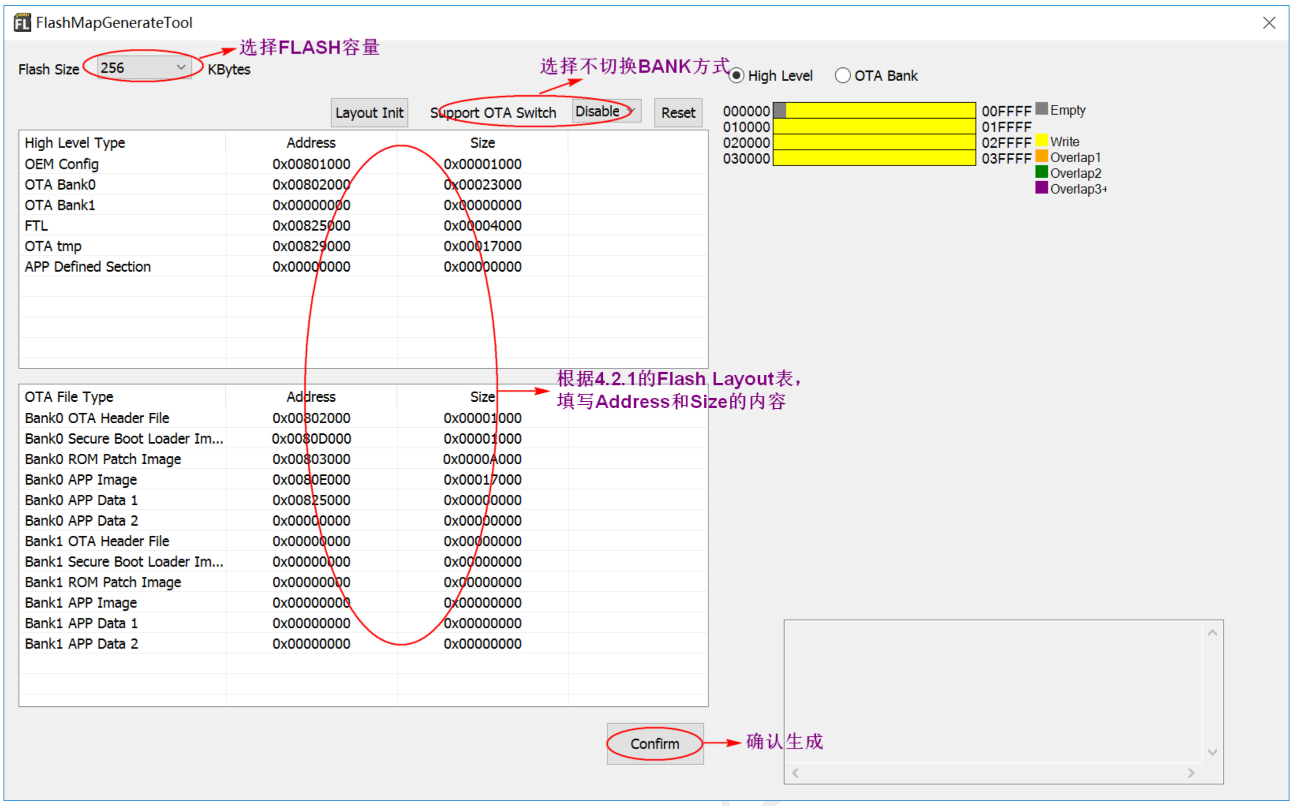
Note: The space for APP data is not allocated in this sample; FLASH Layout should be distributed based on actual size of image and data.

4.2.2 Usage of package tool without bank switching

1. Use FlashMapGenerateTool to 'flash map.ini' and 'flash map.h'. Copy 'flash map.h' to project directory and open project with Keil. Link and compile the project to generate "app_MP_sdk#####+version+MD5.bin" file

for packaging. To apply Without Bank switching method, “mem_config.h” in project directory should be modified.

```
/** @brief set app bank to support OTA: 1 is ota bank1, 0 is ota bank0 */
#define APP_BANK                                    0
```



Flash Size 256 KBytes

[● High Level] OTA Bank

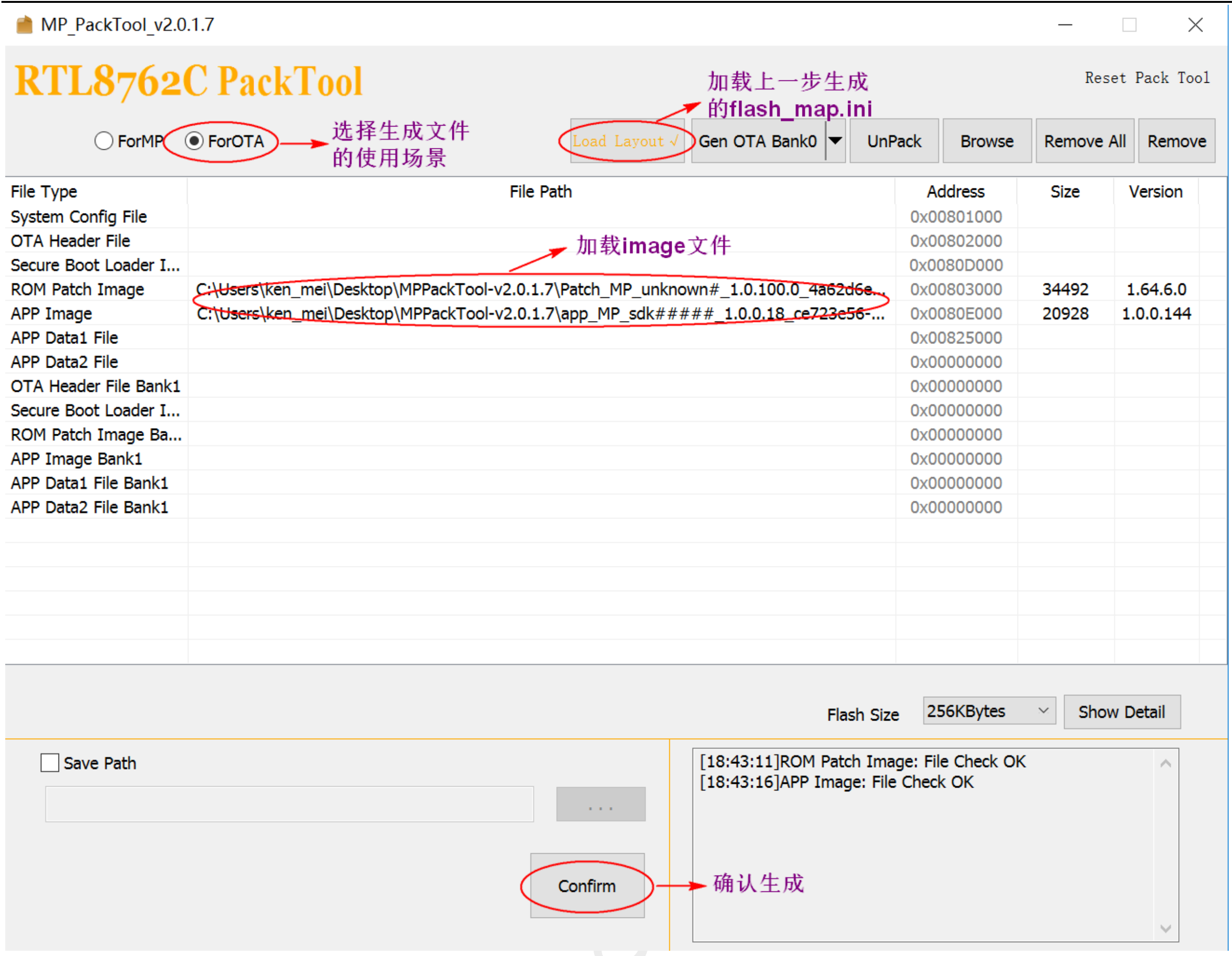
Support OTA Switch Disable

High Level Type	Address	Size
OEM Config	0x00801000	0x00001000
OTA Bank0	0x00802000	0x00023000
OTA Bank1	0x00000000	0x00000000
FTL	0x00825000	0x00004000
OTA tmp	0x00829000	0x00017000
APP Defined Section	0x00000000	0x00000000

OTA File Type	Address	Size
Bank0 OTA Header File	0x00802000	0x00001000
Bank0 Secure Boot Loader Im...	0x0080D000	0x00001000
Bank0 ROM Patch Image	0x00803000	0x0000A000
Bank0 APP Image	0x0080E000	0x00017000
Bank0 APP Data 1	0x00825000	0x00000000
Bank0 APP Data 2	0x00000000	0x00000000
Bank1 OTA Header File	0x00000000	0x00000000
Bank1 Secure Boot Loader Im...	0x00000000	0x00000000
Bank1 ROM Patch Image	0x00000000	0x00000000
Bank1 APP Image	0x00000000	0x00000000
Bank1 APP Data 1	0x00000000	0x00000000
Bank1 APP Data 2	0x00000000	0x00000000

Figure 4.5 flash layout generation

2. Open MP_PackTool to load flash_map.ini generated in previous step and load corresponding image.



File Type	File Path	Address	Size	Version
System Config File		0x00801000		
OTA Header File		0x00802000		
Secure Boot Loader I...		0x0080D000		
ROM Patch Image	C:\Users\ken_mei\Desktop\MPPackTool-v2.0.1.7\Patch_MP_unknown#_1.0.100.0_4a62d6e...	0x00803000	34492	1.64.6.0
APP Image	C:\Users\ken_mei\Desktop\MPPackTool-v2.0.1.7\app_MP_sdk#####_1.0.0.18_ce723e56-...	0x0080E000	20928	1.0.0.144
APP Data1 File		0x00825000		
APP Data2 File		0x00000000		
OTA Header File Bank1		0x00000000		
Secure Boot Loader I...		0x00000000		
ROM Patch Image Ba...		0x00000000		
APP Image Bank1		0x00000000		
APP Data1 File Bank1		0x00000000		
APP Data2 File Bank1		0x00000000		

```
[18:43:11]ROM Patch Image: File Check OK
[18:43:16]APP Image: File Check OK
```

Figure 4.6 MP PACK Tool Load flash layout

Note: 1. OTA Head0 doesn't need to be packaged to PACK, different from the mode with bank switching.

2. Content of Secure boot loader Image is defined in Flash Layout, but it's not recommended to package if there isn't any new version of Secure boot loader Image.

3. If only ROM Patch Image or APP Image, either of them can be packaged.

4. Patch image address needs to be converted through RTKBLEMPTool generally. Patch address here is the same with the default address (0x803000) and it can be used without conversion.

5 OTA Protocol

5.1 DFU Service

DFU Service uuid: { 0x12, 0xA2, 0x4D, 0x2E, 0xFE, 0x14, 0x48, 0x8e, 0x93, 0xD2, 0x17, 0x3C, ***0x87, 0x62,*** *0x00, 0x00*}.

DFU Service defines two Characteristics:

Data Characteristic accepts img data (write no response);

Control Point Characteristic accepts control commands (write/notification);

Control points supported by DFU Service:

Procedure	Requirement	Properties	Parameter Description	Applicable Response Value(s)	Response Parameter

Start DFU①	M	Write	ic_type(UINT8)
			secure_version (UINT8)
			ctrl_flag.value(UINT16)
			image_id (UINT16)
			crc16((UINT16)
			payload_len (UINT32)
				ARV	None

Receive FW image	M	Write	image_id (2byte-UINT16) nImageLength (4Byte-UINT32)	ARV	None
Validate FW	M	Write	image_id (2byte-UINT16)	ARV	None
Activate Image and Reset	M	Write	None	None	None
Reset System	M	Write	None	None	None

Report Received Image Information	M	Write	image_id(UINT16)	ARV	origin_image_version (UINT32) cur_offset (UINT32)
Connection parameter update	M	Write	connIntervalMin(UINT16) connIntervalMax (UINT16) connLatency(UINT16) supervisionTimeout (UINT16)	ARV	None
Buffer check enable	M	Write	None	ARV	Max buffer size(UINT16) Mtu size(UINT16)
Buffer check size&crc	M	Write	mBufferSize(UINT16) mCrc(UINT16)	ARV	Next send offset(UINT32)
IC type	O	Write	None	ARV	ic_type(UINT8)
Copy Img②	M	Write	image_id(UINT16) destination_addr(UINT32) copysize(UINT32)	ARV	None

Table 5.1: Dfu opcode

Note:

1. Parameter of "Start DFU" is ctrlheader of image. It will be written into flash as a part of update file after receiving ctrlheader. The 12 bytes received parameter of Start DFU will be decrypted first, then resolved to be written into Flash.

2. To update APP data with bank switching when secure version and APPDATA version are the same, this command can be used to copy contents of source bank to the destination bank directly without OTA data transporting

To transmit data with buffer check enabled, the size of buffer check must be $(16 * 2^n)$ bytes and no more than max buffer size (returned by buffer check enable commands). If AES enabled, every 16-byte data will be encrypted with AES. When data is received, it needs to be decrypted first. For the last 16 bytes don't need encryption. When buffer is full, data in buffer will be written into Flash.

To transmit data with buffer check disabled, data of 20*n (n=1,2,4,5,10) bytes is sent each time. Data won't be written into Flash until RTL8762C receives 2000 bytes of data.

If AES enabled, the data with the size of 16× q will be encrypted, and the data with the size of r won't be encrypted. q and r follows the formula:

$$size = 16 \times q + r,$$

where q stands for 'quotient' and r stands for 'remainder'.

5.2 OTA Service

OTA Service uuid: { 0x12, 0xA2, 0x4D, 0x2E, 0xFE, 0x14, 0x48, 0x8e, 0x93, 0xD2, 0x17, 0x3C, ***0xFF, 0xD0***, *0x00, 0x00*}.

OTA Service defines the following Characteristics:

Characteristic Name	Requirement	Mandatory Properties	Description
OTA CMD	M/O	WriteWithoutResponse	Refer to OTA CMD
Device Mac	M	Read	Refer to Device Mac
Patch Version	M	Read	Refer to Patch Version
App Version	M	Read	Refer to App Version
Patch Extension Version	O	Read	Refer to Patch Extension Version
Test Mode	O	WriteWithoutResponse	Refer to Test Mode
Device Info	M	Read	Refer to Device Info
Image Counter	O	WriteResponse	Refer to Image Counter
Image Version	M	Read	Refer to Image Version

Table 5.2: OTA Characteristic

5.2.1 OTA CMD

UUID: 0xFFD1

This characteristic allows device to access control point of OTA. If DFU service runs in ROM code, it uses this command to enter DFU mode.

Names	Field Requirement	Format	Value
OTA CMD	Mandatory	Uint8	1

Table 5.3: OTA CMD characteristics

5.2.2 Device Mac

UUID: 0xFFD2

This characteristic is used to read BDA (Bluetooth Device Address) of RTL8762C to compare with the scanned BDA in OTA mode.

Name	Field Requirement	Format	Value
Device Mac	Mandatory	Uint8*6	XX:XX:XX:XX:XX:XX

Table 5.4: Device Mac characteristics

5.2.3 Patch Version

UUID: 0xFFD3

This characteristic is used to read patch version and compatible with Bee1. Patch version information is described in "Image version" in Bee2.

Name	Field Requirement	Format	Value
Patch Version	Mandatory	Uint32	0xNNNNNNNN

Table 5.5: Patch Version characteristic for Bee2 (not recommend, described in image version)

5.2.4 APP Version

UUID: 0xFFD4

This characteristic is used to read APP version and compatible with Bee1 (not recommended in Bee2). APP version information is described in "Image version" in Bee2.

Name	Field Requirement	Format	Value
APP Version	Mandatory	Uint32	0xNNNNNNNN

Table 5.6: APP Version characteristic for Bee2 (not recommend, described in image version)

5.2.5 Patch Extension Version

UUID: 0xFFD5

This characteristic is used to read patch extension version. It is only used for Bee1 but not for Bee2.

Name	Field Requirement	Format	Value
Patch extension Version	Optional	Uint16	0xNNNN

Table 5.7: **错误!未找到引用源。**Patch Extension Version characteristic

5.2.6 Test Mode

UUID: 0xFFD8

This characteristic allow device to exit control point in test mode and write '1' to clear test flag to quit MP mode.

Name	Field Requirement	Format	Value
Test mode	Optional	Uint8	1

Table 5.8: Test Mode characteristics

Note: This characteristic is not related to OTA.

5.2.7 Device Info

UUID: 0xFFF1

This characteristic is used to read device information, and its description is shown below:

For the other BT SoC chip, the characteristic is as below.

Name	Field Requirement	Format	Value
Device info	Mandatory	As Table 6.10	As Table 6.10

Table 5.9: Device info characteristic for Bee2.

Format	**ICType**	**Version**	**Secure Version**	**MODE**		**Max Buffer Size**	**Reserved**
	8bit	8bit	8bit	8bit		16bit	16bit
Value	BBpro: 4 BEE2:5	Bit3~0: OTA version = 0x1 Bit7~4: Reserved:0x0.		Bit 0	0:normal mode 1:Support buffer check	0xNNNN	0x00
				Bit 1	0:Aes flag not set 1:Aes flag Set		
				Bit 2	0: Only encrypt first 16 bytes of OTA data in normal mode. 1:Encrypt 16*N bytes of OTA date in normal mode		
				Bit3	0: Disable Copy Image. 1: Enable Copy Image.		
				Bit4	0: Update one Image at a time. 1: Update multiple Images at a time.		

Format (Attach to above table)	**Image Version Indicator**
	32bit

Value (Attach to above table)	0xNNNNNNNN Indications for each image version. Each indication uses 2 bits. 00: image does not exist. 01: image exists in bank0, OTA should update image for bank1. 10: image exists in bank1, OTA should update image for bank0. 11: image is standalone. OTA should update image for standalone. bit[1:0]: Image 0 … bit[2N+1:2N]:Image N Image indicator for bee2 is as below:

Image 0	SOCV Config File
Image 1	System Config File
Image 2	OTA Header File
Image 3	Secure Boot Loader Image
Image 4	ROM Patch Image
Image 5	APP Image
Image 6	APP Data1 File
Image 7	APP Data2 File

Table 5.10: Device info Format For Bee2 (OTA version = 1)

5.2.8 Image Counter

UUID: 0xFFF2

This characteristic is used to write response and inform device how many image files are about to be written.

Name	Field Requirement	Format	Value
Image Counter	Optional	Uint8	0xNN

Table 5.11: Image Counter characteristics

5.2.9 Image Version

UUID: 0xFFE0~FFEF

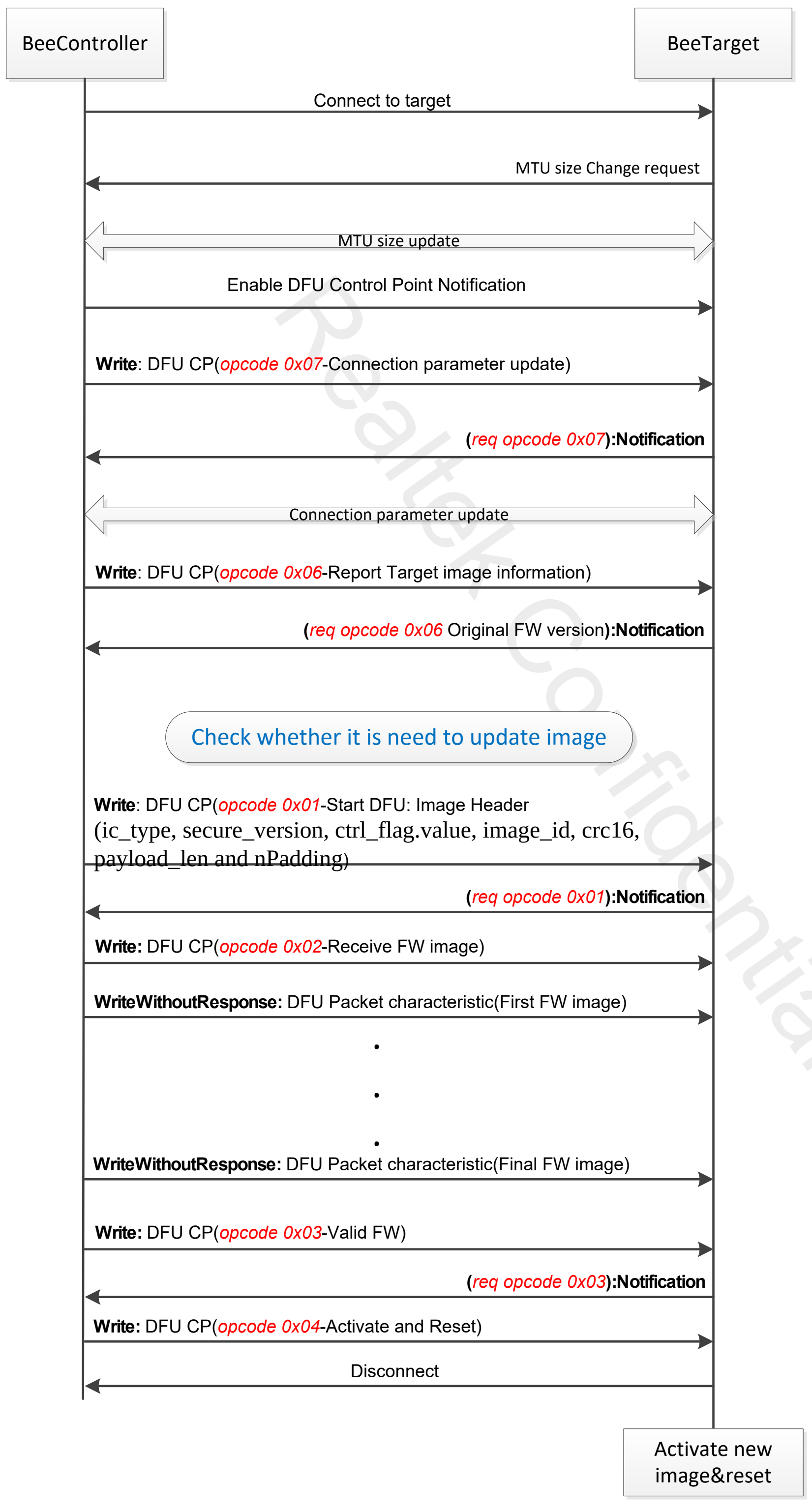
This characteristic is used to read image versions of device. Each image version occupies 4 bytes. Limited to MTU size (20 bytes), user needs to define another characteristic (**UUID: 0xFFE0~FFEF**) to read next image version when number of image is greater than 5. The number of device img versions is indicated by Image Version Indicator, which is defined in Device Info (0xfff1).

Name	Field Requirement	Format	Value
Image Version	mandatory	Uint32*N	

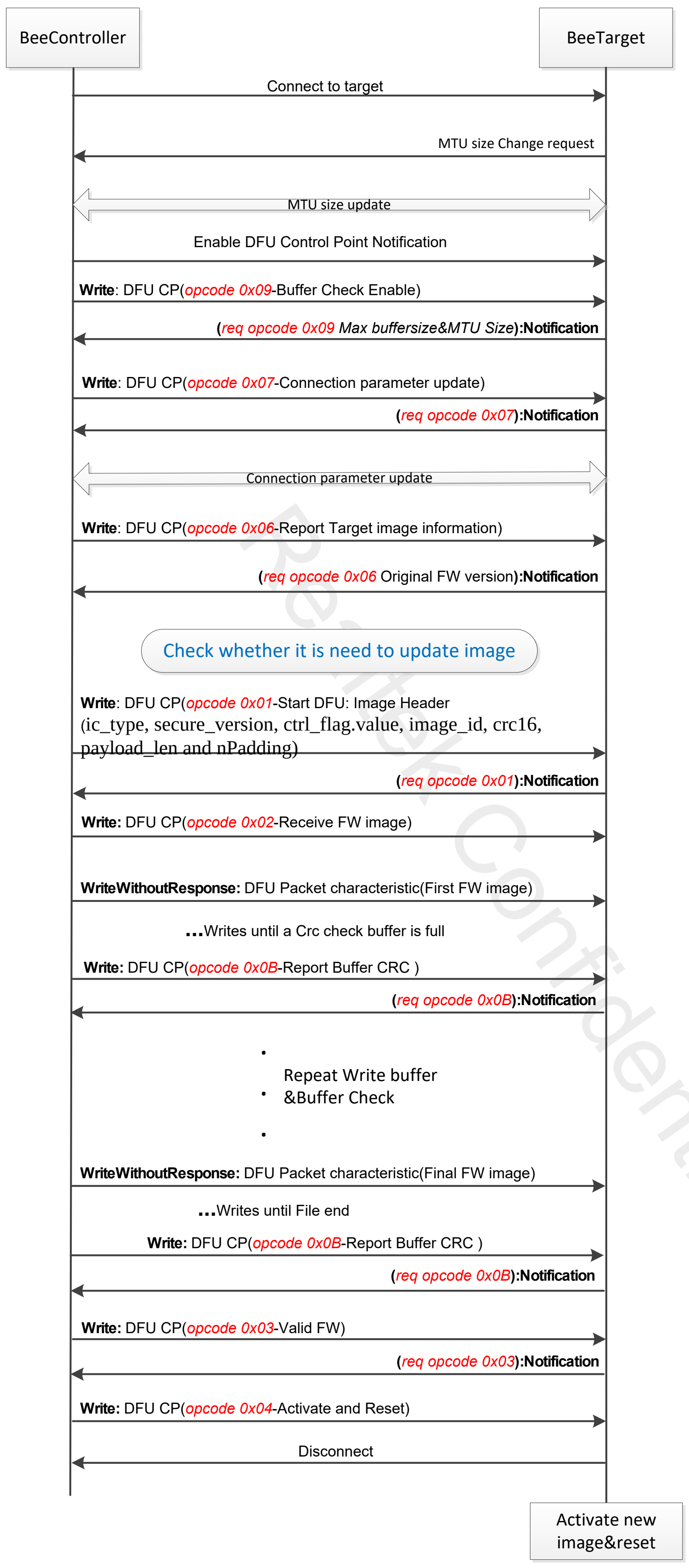
Table 5.12: Image Counter characteristics

5.3 OTA Procedure

5.3.1 OTA procedure without buffer check


BeeController
BeeTarget
Connect to target
MTU size Change request
MTU size update
Enable DFU Control Point Notification
Write: DFU CP(opcode 0x07-Connection parameter update)
(req opcode 0x07):Notification
Connection parameter update
Write: DFU CP(opcode 0x06-Report Target image information)
(req opcode 0x06 Original FW version):Notification
Check whether it is need to update image
Write: DFU CP(opcode 0x01-Start DFU: Image Header (ic_type, secure_version, ctrl_flag.value, image_id, crc16, payload_len and nPadding)
(req opcode 0x01):Notification
Write: DFU CP(opcode 0x02-Receive FW image)
WriteWithoutResponse: DFU Packet characteristic(First FW image)
WriteWithoutResponse: DFU Packet characteristic(Final FW image)
Write: DFU CP(opcode 0x03-Valid FW)
(req opcode 0x03):Notification
Write: DFU CP(opcode 0x04-Activate and Reset)
Disconnect
Activate new image&reset

5.3.2 OTA procedure with buffer check


BeeController
BeeTarget
Connect to target
MTU size Change request
MTU size update
Enable DFU Control Point Notification
Write: DFU CP(opcode 0x09-Buffer Check Enable)
(req opcode 0x09 Max buffersize&MTU Size):Notification
Write: DFU CP(opcode 0x07-Connection parameter update)
(req opcode 0x07):Notification
Connection parameter update
Write: DFU CP(opcode 0x06-Report Target image information)
(req opcode 0x06 Original FW version):Notification
Check whether it is need to update image
Write: DFU CP(opcode 0x01-Start DFU: Image Header (ic_type, secure_version, ctrl_flag.value, image_id, crc16, payload_len and nPadding)
(req opcode 0x01):Notification
Write: DFU CP(opcode 0x02-Receive FW image)
WriteWithoutResponse: DFU Packet characteristic(First FW image)
...Writes until a Crc check buffer is full
Write: DFU CP(opcode 0x0B-Report Buffer CRC)
(req opcode 0x0B):Notification
Repeat Write buffer &Buffer Check
WriteWithoutResponse: DFU Packet characteristic(Final FW image)
...Writes until File end
Write: DFU CP(opcode 0x0B-Report Buffer CRC)
(req opcode 0x0B):Notification
Write: DFU CP(opcode 0x03-Valid FW)
(req opcode 0x03):Notification
Write: DFU CP(opcode 0x04-Activate and Reset)
Disconnect
Activate new image&reset

5.3.3 Multiple File Update

1. Without bank switching, a new file cannot be updated until the previous file has been verified and program has been rebooted when packaged file includes Patch, APP or APPDATA.
2. With bank switching, program cannot be rebooted until all the files have been updated and verified when the packaged file includes OTA Header, Patch, APP or APPDATA. Otherwise, this update will be invalid for that all the files in bank region must come into effect to ensure the program is running properly with bank switching.

6 Usage of Master application

Omitted

7 Reference

[1] 《RTL8762C Device Firmware Update Profile Rom Version》

[2] 《RTL8762C Device Firmware Update Service Rom Version》