RTL8762D OTA User Manual

V1.1

2021/02/22

Revision History

Date	Version	Comments	Author	Reviewer
2020/06/29	V1.0	First release version	Grace	
2021/02/22	V1.1	Add upperstck image sdk ota details	Grace	

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 RTL8762D Flash.

1.2 Related Emphasis

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

2 Flash layout

Flash layout of RTL8762D 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.

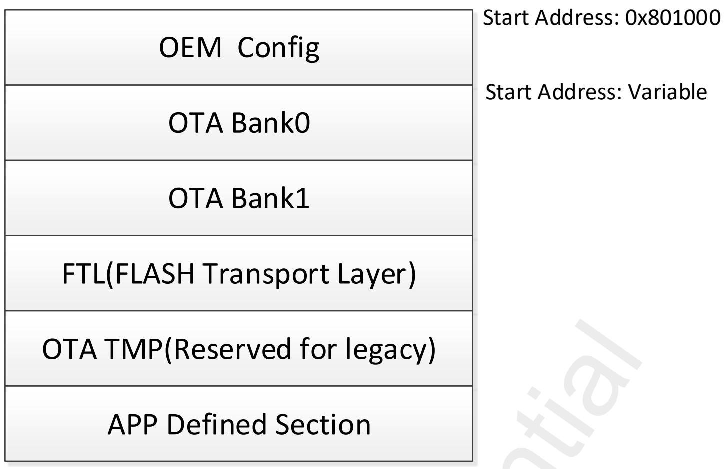


Figure 2.1: Flash Layout

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.

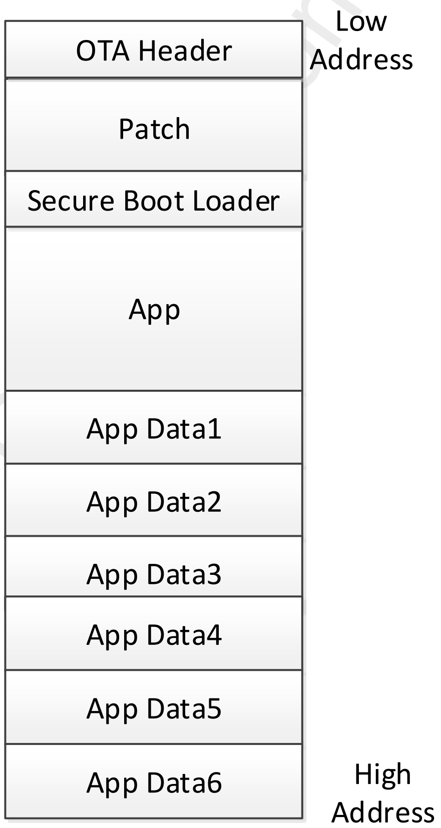


Figure 2.2 Layout of Bank 0/1 in Upperstack Lib SDK

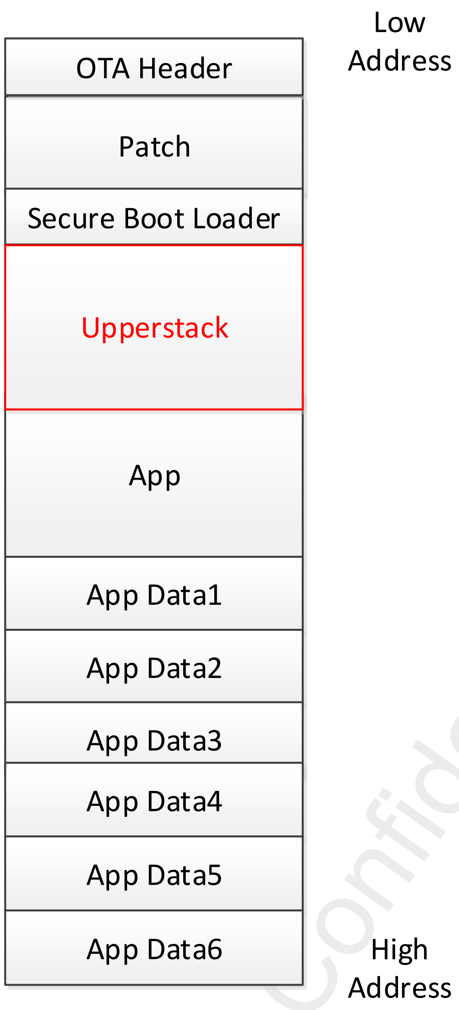


Figure 2.3 Layout of Bank 0/1 in Upperstack Image SDK

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

Upperstack	Determined in the OTA Header region	Variable	This region contains the upperstack code.
App	Determined in the OTA Header region	Variable	This region contains project code.
App Data1	Determined in the OTA Header region	Variable	Data region used in project.
App Data2	Determined in the OTA Header region	Variable	Data region used in project.
App Data3	Determined in the OTA Header region	Variable	Data region used in project.
App Data4	Determined in the OTA Header region	Variable	Data region used in project.
App Data5	Determined in the OTA Header region	Variable	Data region used in project.
App Data6	Determined in the OTA Header region	Variable	Data region used in project.

Table 2.2 Flash Segmentation

3 Image Format

All images that may need to be upgraded (OTA Header, Secure boot, Patch, APP, APP Data1, APP Data2, APP Data3, APP Data4, APP Data5, APP Data6) are composed of a 1KB header and payload. Among them, the header part in OTA Header image and other images are slightly different. In this article, the headers of all upgradeable images except the OTA Header image are called Image Headers.

3.1 OTA Header Image 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.

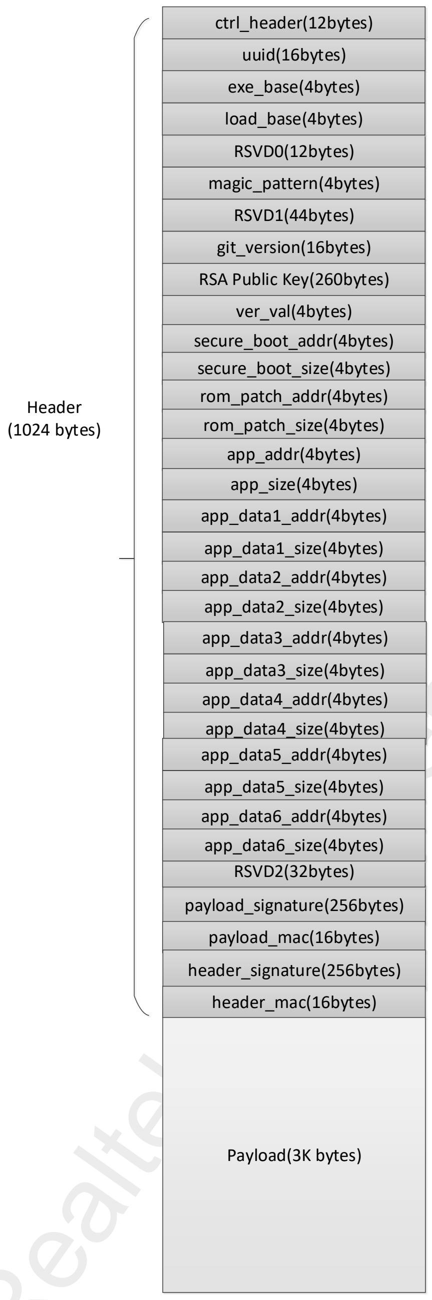


Figure 3.1: OTA Header Format

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
app_data3_addr	4	Start address of application data3
app_data3_size	4	Size of application data3
app_data4_addr	4	Start address of application data4
app_data4_size	4	Size of application data4
app_data5_addr	4	Start address of application data5
app_data5_size	4	Size of application data5
app_data6_addr	4	Start address of application data6
app_data6_size	4	Size of application data6

Table 3.3: Fields of OTA Header

3.2 Other Image Format

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

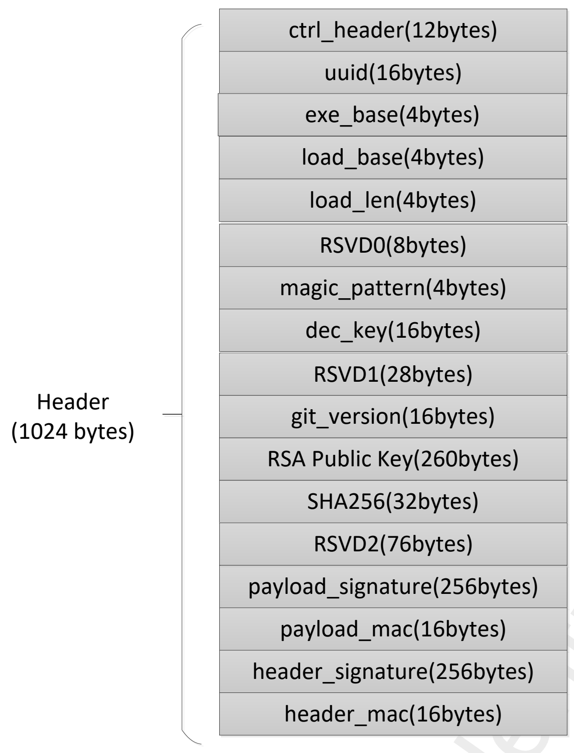


Figure 3.4: Image Header Layout

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 9 when RTL8762D 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 T_IMG_ID.

```
    typedef enum
    {
        SCCD           = 0x278D,
        OCCD           = 0x278E,
        FactoryCode = 0x278F,
        OTA            = 0x2790, /**< OTA header */
        SecureBoot   = 0x2791,
        RomPatch      = 0x2792,
        AppPatch      = 0x2793,
        AppData1      = 0x2794,
        AppData2      = 0x2795,
        AppData3      = 0x2796,
        AppData4      = 0x2797,
        AppData5      = 0x2798,
        AppData6       = 0x2799,
    #ifdef SUPPORT_ALONE_UPPERSTACK_IMG
        UpperStack    = 0x279a,
        IMAGE_MAX     = 0x279b,
    #else
        IMAGE_MAX     = 0x279a,
    #endif
        IMAGE_USER_DATA = 0xFFFE,    /**<the image only support unsafe single bank ota*/
    } T_IMG_ID;
```

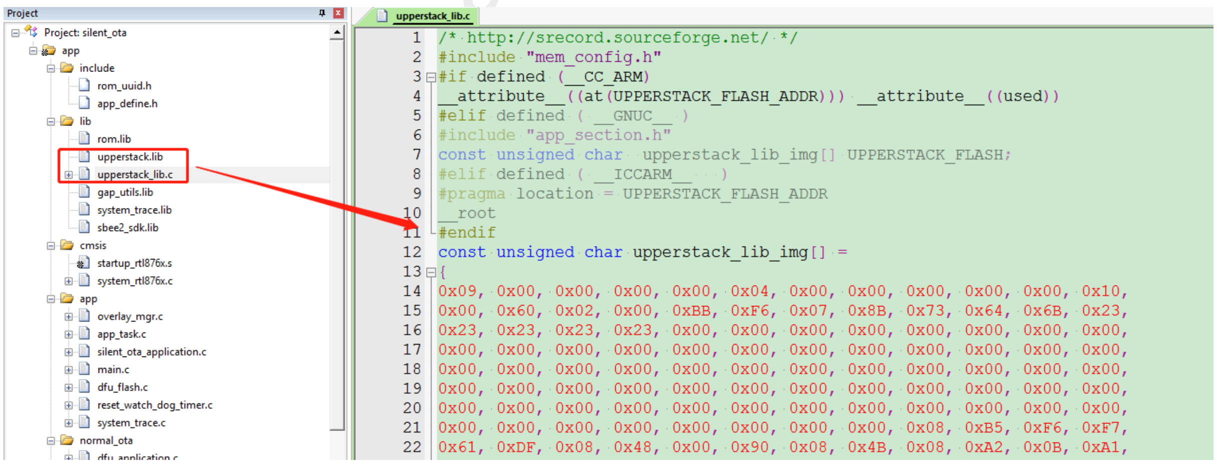
payload_length represents the size of image in byte, excluding 1KB image header.

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.

The part of Image Payload is the implement in SecureBoot/Patch/APP and the raw data part in APPData. Because the RTL8762D ROM only integrates the BT Lowerstack part, and the BT Upperstack part (the files in the path 'SBEE2-SDK-Vx.x.x\bin\upperstack') is compiled in the form of bin with the real APP code.



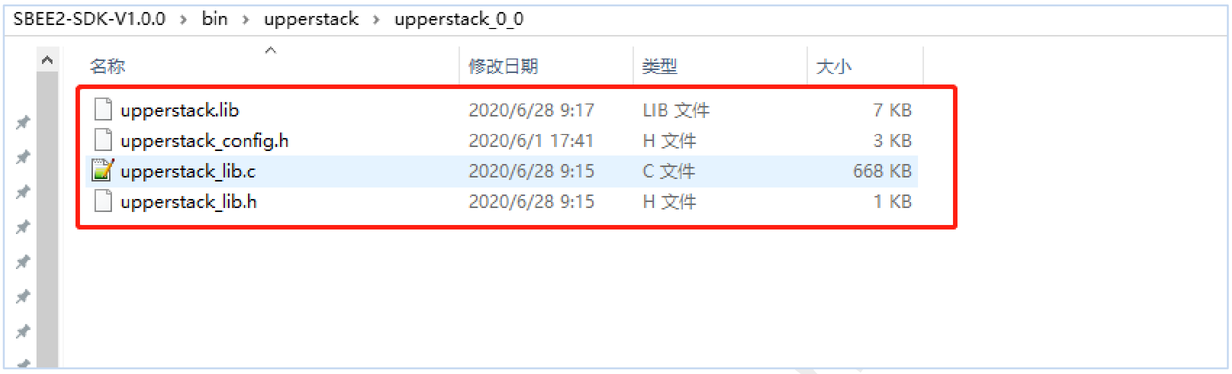
```
/* http://srecord.sourceforge.net/ */
#include "mem_config.h"
#if defined (__CC_ARM)
__attribute__((at(UPPERSTACK_FLASH_ADDR))) __attribute__((used))
#elif defined ( __GNUC__ )
#include "app_section.h"
const unsigned char  upperstack_lib_img[] UPPERSTACK_FLASH;
#elif defined ( __ICCARM__   )
#pragma location = UPPERSTACK_FLASH_ADDR
__root
#endif
const unsigned char upperstack_lib_img[] =
{
0x09, 0x00, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10,
0x00, 0x60, 0x02, 0x00, 0xBB, 0xF6, 0x07, 0x8B, 0x73, 0x64, 0x6B, 0x23,
0x23, 0x23, 0x23, 0x23, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0xB5, 0xF6, 0xF7,
0x61, 0xDF, 0x08, 0x48, 0x00, 0x90, 0x08, 0x4B, 0x08, 0xA2, 0x0B, 0xA1,
```

Figure 3-3 APP Image includes Upperstack co-compilation

4 Package and flash layout sample

4.1 Upperstack Lib SDK

In the Upperstack lib SDK, the code of the upperstack part is compiled together with the app in the form of static lib, and this part is published through the following files.


SBEE2-SDK-V1.0.0 › bin › upperstack › upperstack_0_0

名称	修改日期	类型	大小
upperstack.lib	2020/6/28 9:17	LIB 文件	7 KB
upperstack_config.h	2020/6/1 17:41	H 文件	3 KB
upperstack_lib.c	2020/6/28 9:15	C 文件	668 KB
upperstack_lib.h	2020/6/28 9:15	H 文件	1 KB

Figure 4 1 Upperstack Realeased Files

As the Upperstack lib SDK is the earliest maintenance version, it is not recommended to customers now. The OTA part of this SDK will not be described in detail in this paper.

4.2 Upperstack Image SDK

The biggest difference about OTA between Upperstack image SDK and Upperstack lib SDK is an additional Upperstack image. When the bank switch is not supported, the space of ota temp area can be saved and the utilization of flash can be improved using Upperstack image SDK.

■ 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.

4.1.1.1 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(bytes)	start address
1) Reserved	4K	0x00800000
2) OEM Header	4K	0x00801000
3) OTA Bank0	400K	0x00802000
a) OTA Header	4K	0x00802000
b) Secure boot loader	4K	0x0080D000
c) Patch code	40K	0x00803000
d) Upperstack code	120K	0x0080E000
e) APP code	232K	0x0082C000
f) APP data1	0K	0x00866000
g) APP data2	0K	0x00866000
h) APP data3	0K	0x00866000
i) APP data4	0K	0x00866000
j) APP data5	0K	0x00866000
k) APP data6	0K	0x00866000
4) OTA Bank1 (size must be same with OTA Bank0)	400K	0x00866000
a) OTA Header	4K	0x00866000
b) Secure boot loader	0K	0x00871000
c) Patch code	40K	0x00867000
d) Upperstack code	120K	0x00872000
e) APP code	232K	0x00890000
f) APP data1	0K	0x008CA000
g) APP data2	0K	0x008CA000
h) APP data3	0K	0x008CA000
i) APP data4	0K	0x008CA000

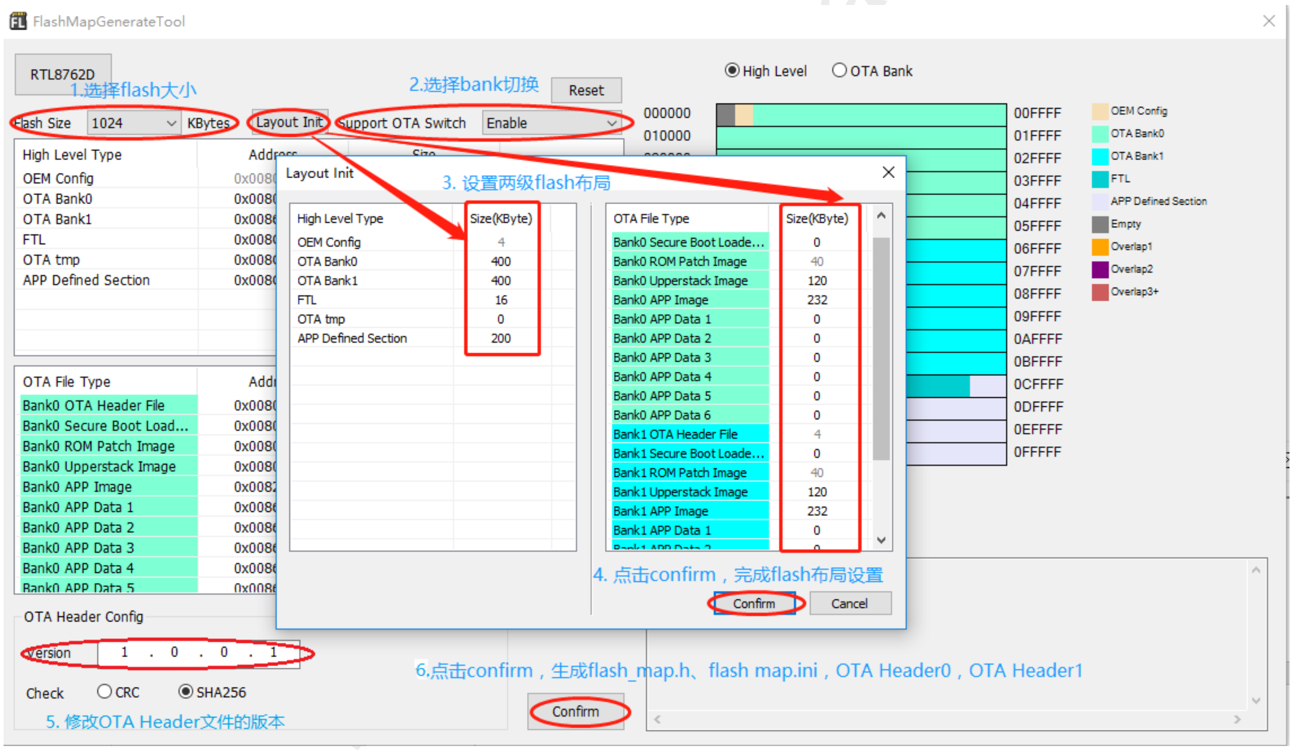
j) APP data5	0K	0x008CA000
k) APP data6	0K	0x008CA000
5) FTL	16K	0x008CA000
6) OTA Temp	0K	0x008CE000
7) APP Defined Section	200K	0x008CE000

Table 4.1: FLASH Layout Sample

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

4.1.1.2 Usage of package tool with bank switching

1． Use FlashMapGenerateTool to ‘flash map.ini’, ‘flash map.h’,’ OTA Header0’ and ‘OTA Header1’.



FlashMapGenerateTool

RTL8762D

1.选择flash大小

Flash Size 1024 KBytes

Layout Init

2.选择bank切换

Support OTA Switch: Enable

High Level Type	Address
OEM Config	0x008
OTA Bank0	0x008
OTA Bank1	0x008
FTL	0x008
OTA tmp	0x008
APP Defined Section	0x008

OTA File Type	Addr
Bank0 OTA Header File	0x008
Bank0 Secure Boot Load...	0x008
Bank0 ROM Patch Image	0x008
Bank0 Upperstack Image	0x008
Bank0 APP Image	0x008
Bank0 APP Data 1	0x008
Bank0 APP Data 2	0x008
Bank0 APP Data 3	0x008
Bank0 APP Data 4	0x008
Bank0 APP Data 5	0x008

OTA Header Config

Version 1 . 0 . 0 . 1

Check ○ CRC [✓ SHA256]

5. 修改OTA Header文件的版本

Layout Init

3. 设置两级flash布局

High Level Type	Size(KByte)
OEM Config	4
OTA Bank0	400
OTA Bank1	400
FTL	16
OTA tmp	0
APP Defined Section	200

OTA File Type	Size(KByte)
Bank0 Secure Boot Loade...	0
Bank0 ROM Patch Image	40
Bank0 Upperstack Image	120
Bank0 APP Image	232
Bank0 APP Data 1	0
Bank0 APP Data 2	0
Bank0 APP Data 3	0
Bank0 APP Data 4	0
Bank0 APP Data 5	0
Bank0 APP Data 6	0
Bank1 OTA Header File	4
Bank1 Secure Boot Loade...	0
Bank1 ROM Patch Image	40
Bank1 Upperstack Image	120
Bank1 APP Image	232
Bank1 APP Data 1	0

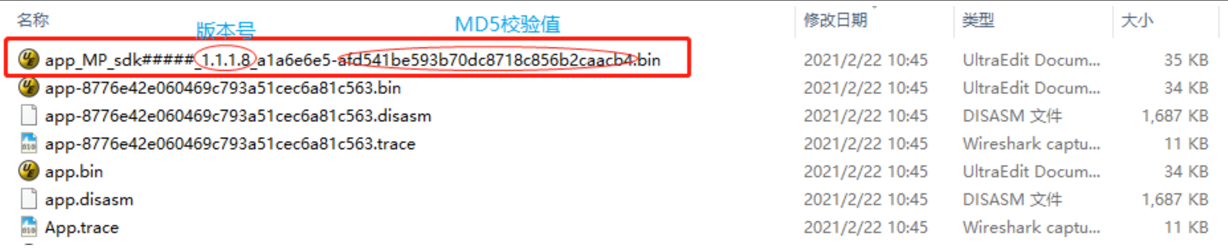
4. 点击confirm，完成flash布局设置

6.点击confirm，生成flash_map.h、flash map.ini，OTA Header0，OTA Header1

Figure 4-2 flash layout config and generate ota header image

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

2． 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. Bank switching method needs to compile the app images of OTA BANK0 and OTA BANK1.



名称	修改日期	类型	大小
app_MP_sdk#####_1.1.1.8_a1a6e6e5-afd541be593b70dc8718c856b2caacb4.bin	2021/2/22 10:45	UltraEdit Docum...	35 KB
app-8776e42e060469c793a51cec6a81c563.bin	2021/2/22 10:45	UltraEdit Docum...	34 KB
app-8776e42e060469c793a51cec6a81c563.disasm	2021/2/22 10:45	DISASM 文件	1,687 KB
app-8776e42e060469c793a51cec6a81c563.trace	2021/2/22 10:45	Wireshark captu...	11 KB
app.bin	2021/2/22 10:45	UltraEdit Docum...	34 KB
app.disasm	2021/2/22 10:45	DISASM 文件	1,687 KB
App.trace	2021/2/22 10:45	Wireshark captu...	11 KB

Figure 4-3 builded APP image

The demo app project in the Realtek released SDK can only compile the app image of OTA BANK0 by default. Compiling app image of OTA BANK1 need modify the macro 'APP_BANK' in mem_config.h in the same directory as the project file to 1.

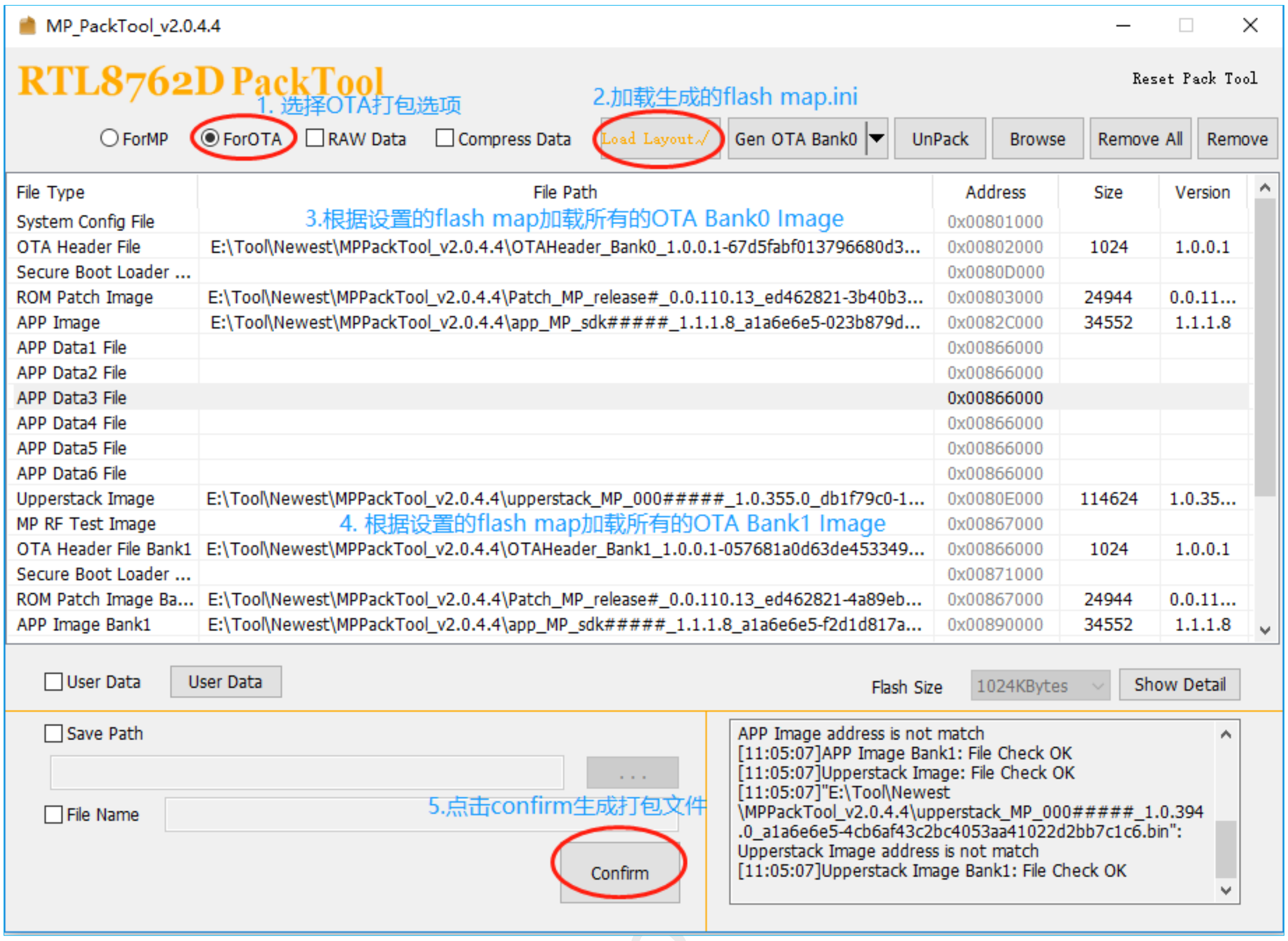
```
/** @brief set app bank to support OTA: 1 is ota bank1, 0 is ota bank0 */
    #define APP_BANK                              1


    #if (APP_BANK == 0)
    #define UPPERSTACK_FLASH_ADDR          (BANK0_APP_ADDR + 1024)
    #else
    #define UPPERSTACK_FLASH_ADDR          (BANK1_APP_ADDR + 1024)
    #endif
```

3. Get patch and Upperstack image that runs runs in OTA Bank1.

Note: The default released patch and Upperstack image can only run in OTA Bank0. Please consult Realtek to get patch and Upperstack image that runs in OTA Bank1, when you choose bank swithing method.

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



File Type	File Path	Address	Size	Version
System Config File		0x00801000		
OTA Header File	E:\Tool\Newest\MPPackTool_v2.0.4.4\OTAHeader_Bank0_1.0.0.1-67d5fabf013796680d3...	0x00802000	1024	1.0.0.1
Secure Boot Loader ...		0x0080D000		
ROM Patch Image	E:\Tool\Newest\MPPackTool_v2.0.4.4\Patch_MP_release#_0.0.110.13_ed462821-3b40b3...	0x00803000	24944	0.0.11...
APP Image	E:\Tool\Newest\MPPackTool_v2.0.4.4\app_MP_sdk#####_1.1.1.8_a1a6e6e5-023b879d...	0x0082C000	34552	1.1.1.8
APP Data1 File		0x00866000		
APP Data2 File		0x00866000		
APP Data3 File		0x00866000		
APP Data4 File		0x00866000		
APP Data5 File		0x00866000		
APP Data6 File		0x00866000		
Upperstack Image	E:\Tool\Newest\MPPackTool_v2.0.4.4\upperstack_MP_000#####_1.0.355.0_db1f79c0-1...	0x0080E000	114624	1.0.35...
MP RF Test Image		0x00867000		
OTA Header File Bank1	E:\Tool\Newest\MPPackTool_v2.0.4.4\OTAHeader_Bank1_1.0.0.1-057681a0d63de453349...	0x00866000	1024	1.0.0.1
Secure Boot Loader ...		0x00871000		
ROM Patch Image Ba...	E:\Tool\Newest\MPPackTool_v2.0.4.4\Patch_MP_release#_0.0.110.13_ed462821-4a89eb...	0x00867000	24944	0.0.11...
APP Image Bank1	E:\Tool\Newest\MPPackTool_v2.0.4.4\app_MP_sdk#####_1.1.1.8_a1a6e6e5-f2d1d817a...	0x00890000	34552	1.1.1.8

```
APP Image address is not match
[11:05:07]APP Image Bank1: File Check OK
[11:05:07]Upperstack Image: File Check OK
[11:05:07]"E:\Tool\Newest
\MPPackTool_v2.0.4.4\upperstack_MP_000#####_1.0.394
.0_a1a6e6e5-4cb6af43c2bc4053aa41022d2bb7c1c6.bin":
Upperstack Image address is not match
[11:05:07]Upperstack Image 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.

*5. APP DATA file is generated with APP DATA generation script, for detailed information refer to **SBee2 Tools User Guide**.*

Do not support bank switching

4.1.2.1 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:

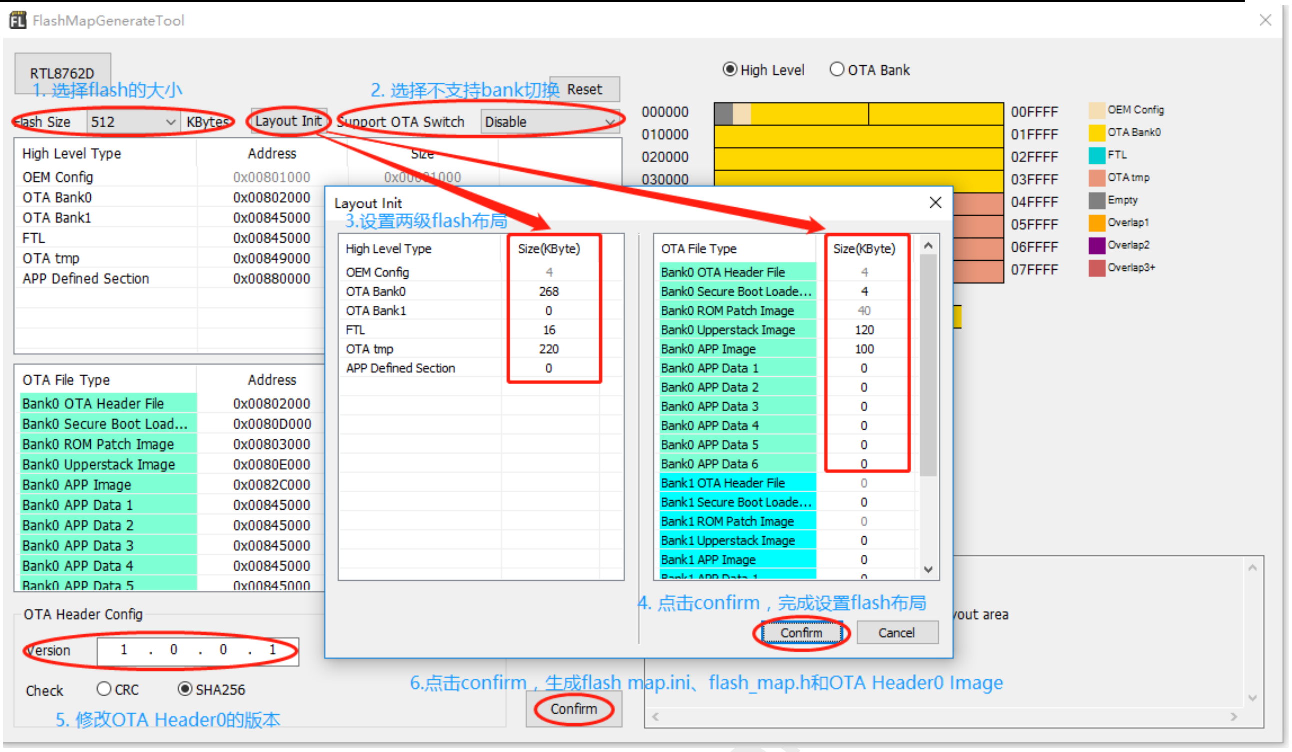
Table 4-2: FLASH Layout Sample

Sample flash layout(total size is 512KB)	Size(bytes)	Start address
1) Reserved	4K	0x800000
2) OEM Header	4K	0x801000
3) OTA Bank0	140K	0x802000
a) OTA Header	4K	0x802000
b) Secure boot loader	4K	0x80D000
c) Patch code	40K	0x803000
d) Upperstack code	120K	0x80E000
e) APP code	100K	0x82C000
f) APP data1	0K	0x00845000
g) APP data2	0K	0x00845000
h) APP data3	0K	0x00845000
i) APP data4	0K	0x00845000
j) APP data5	0K	0x00845000
k) APP data6	0K	0x00845000
4) OTA Bank1	0K	0x00845000
5) FTL	16K	0x00845000
6) OTA Temp	220K	0x00849000
7) APP Defined Section	0K	0x00880000

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.

4.1.2.2 Usage of package tool without bank switching

1. Use FlashMapGenerateTool to 'flash map.ini' and 'flash map.h'.



Flash Size: 512 KBytes
Support OTA Switch: Disable

High Level Type	Address	Size
OEM Config	0x00801000	0x00001000
OTA Bank0	0x00802000	
OTA Bank1	0x00845000	
FTL	0x00845000	
OTA tmp	0x00849000	
APP Defined Section	0x00880000	

OTA File Type	Address
Bank0 OTA Header File	0x00802000
Bank0 Secure Boot Load...	0x0080D000
Bank0 ROM Patch Image	0x00803000
Bank0 Upperstack Image	0x0080E000
Bank0 APP Image	0x0082C000
Bank0 APP Data 1	0x00845000
Bank0 APP Data 2	0x00845000
Bank0 APP Data 3	0x00845000
Bank0 APP Data 4	0x00845000
Bank0 APP Data 5	0x00845000

OTA Header Config
Version 1 . 0 . 0 . 1
Check CRC [✓ SHA256]

Layout Init

High Level Type	Size(KByte)
OEM Config	4
OTA Bank0	268
OTA Bank1	0
FTL	16
OTA tmp	220
APP Defined Section	0

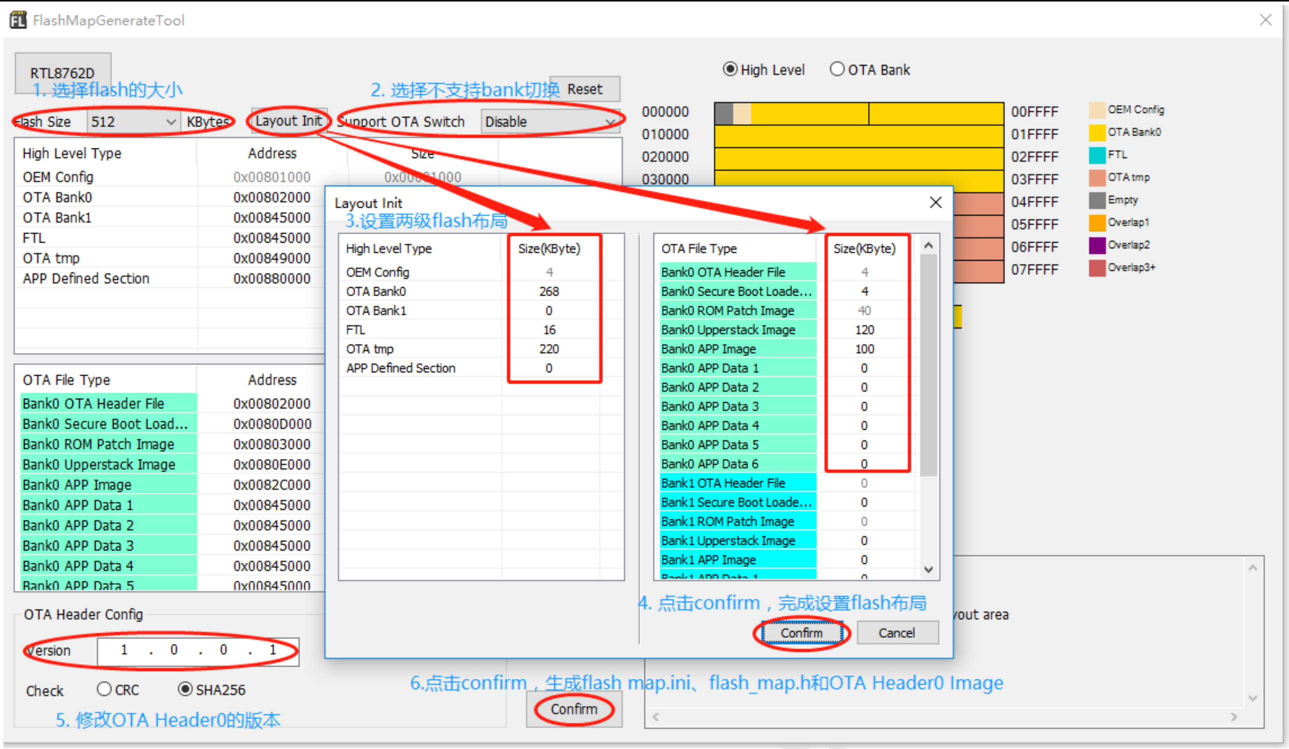
OTA File Type	Size(KByte)
Bank0 OTA Header File	4
Bank0 Secure Boot Loade...	4
Bank0 ROM Patch Image	40
Bank0 Upperstack Image	120
Bank0 APP Image	100
Bank0 APP Data 1	0
Bank0 APP Data 2	0
Bank0 APP Data 3	0
Bank0 APP Data 4	0
Bank0 APP Data 5	0
Bank0 APP Data 6	0
Bank1 OTA Header File	0
Bank1 Secure Boot Loade...	0
Bank1 ROM Patch Image	0
Bank1 Upperstack Image	0
Bank1 APP Image	0

Figure 4-5 flash layout config and generate ota header image

2． 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
```

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



Flash Size 512 KBytes

Support OTA Switch Disable

High Level Type	Address	Size
OEM Config	0x00801000	0x00001000
OTA Bank0	0x00802000	
OTA Bank1	0x00845000	
FTL	0x00845000	
OTA tmp	0x00849000	
APP Defined Section	0x00880000	

OTA File Type	Address
Bank0 OTA Header File	0x00802000
Bank0 Secure Boot Load...	0x0080D000
Bank0 ROM Patch Image	0x00803000
Bank0 Upperstack Image	0x0080E000
Bank0 APP Image	0x0082C000
Bank0 APP Data 1	0x00845000
Bank0 APP Data 2	0x00845000
Bank0 APP Data 3	0x00845000
Bank0 APP Data 4	0x00845000
Bank0 APP Data 5	0x00845000

OTA Header Config

Version 1 . 0 . 0 . 1

Check CRC [✓ SHA256]

Layout Init

High Level Type	Size(KByte)
OEM Config	4
OTA Bank0	268
OTA Bank1	0
FTL	16
OTA tmp	220
APP Defined Section	0

OTA File Type	Size(KByte)
Bank0 OTA Header File	4
Bank0 Secure Boot Loade...	4
Bank0 ROM Patch Image	40
Bank0 Upperstack Image	120
Bank0 APP Image	100
Bank0 APP Data 1	0
Bank0 APP Data 2	0
Bank0 APP Data 3	0
Bank0 APP Data 4	0
Bank0 APP Data 5	0
Bank0 APP Data 6	0
Bank1 OTA Header File	0
Bank1 Secure Boot Loade...	0
Bank1 ROM Patch Image	0
Bank1 Upperstack Image	0
Bank1 APP Image	0

Figure 4-6 pack OTA images using MP PACK Tool

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.

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	M	Write	connIntervalMin(UINT16)	ARV	None

parameter update			connIntervalMax (UINT16) connLatency(UINT16) supervisionTimeout (UINT16)		
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 RTL8762D receives 2000 bytes of data.

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

$$\boldsymbol{size = 16 \times q + r,}$$

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

■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

Device Mac

UUID: 0xFFD2

This characteristic is used to read BDA (Bluetooth Device Address) of RTL8762D 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

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/SBee2.

Name	Field Requirement	Format	Value
Patch Version	Mandatory	Uint32	0xNNNNNNNN

Table 5.5: Patch Version characteristic for Bee2/SBee2 (not recommend, described in image version)

APP Version

UUID: 0xFFD4

This characteristic is used to read APP version and compatible with Bee1 (not recommended in Bee2/SBee2). APP version information is described in "Image version" in Bee2/SBee2.

Name	Field Requirement	Format	Value
APP Version	Mandatory	Uint32	0xNNNNNNNN

Table 5.6: APP Version characteristic for Bee2/SBee2 (not recommend, described in image version)

Patch Extension Version

UUID: 0xFFD5

This characteristic is used to read patch extension version. It is only used for Bee1 but not for Bee2/SBee2.

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

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.

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/SBee2.

Format	ICType	Version	Secure Version	MODE		Max Buffer Size	Reserved
	8bit	8bit	8bit	8bit		16bit	16bit
Value	BBpro: 4 Bee2:5 SBee2: 9	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/SBee2 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/SBee2 (OTA version = 1)

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

Image Version

UUID: 0xFFE0~FFEF

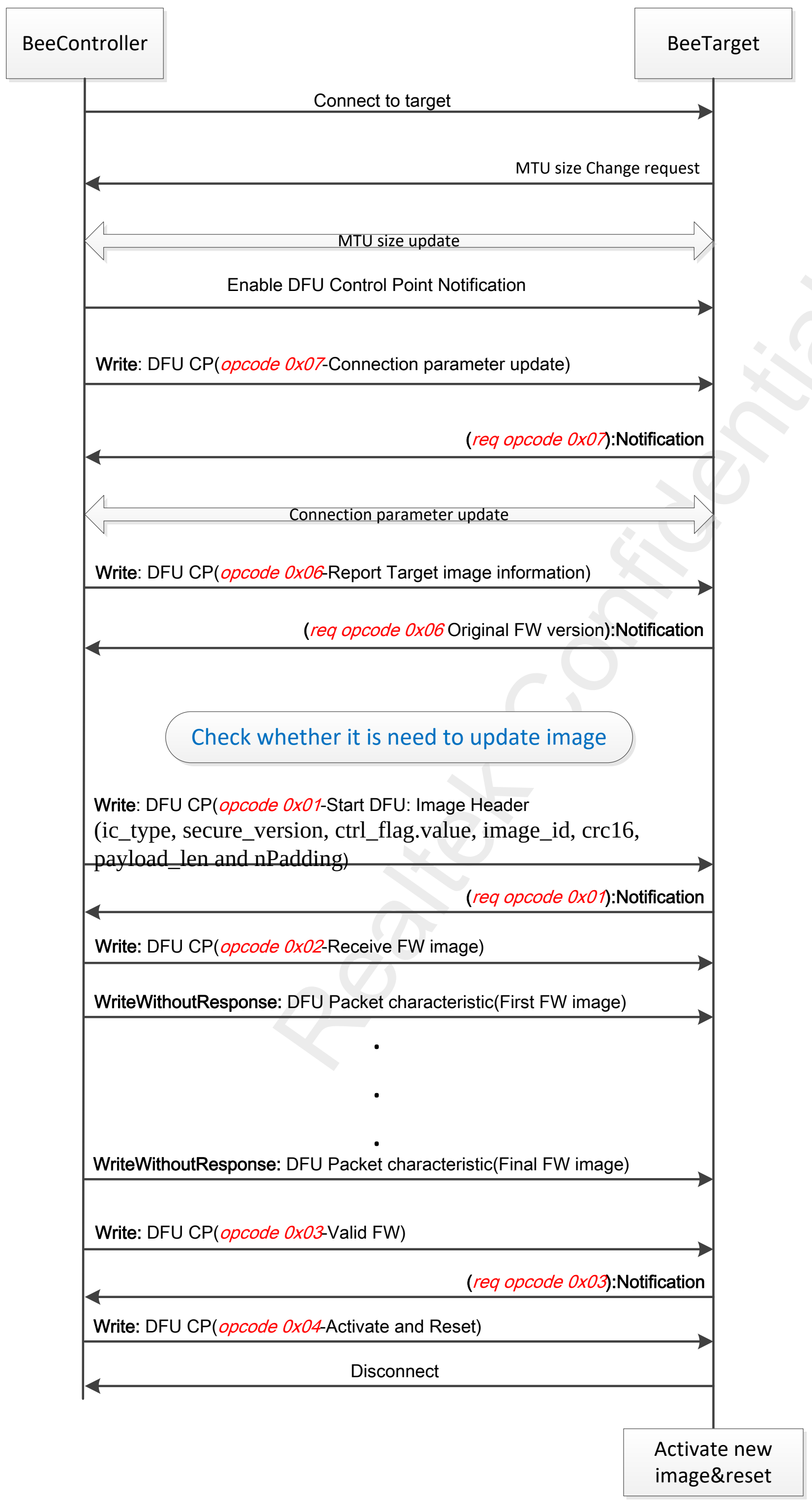
This characteristic is used to read image versions of device. Each image version occupies 4 bytes. Limited to MTU size (20 bytes), user needs to define another characteristic (**UUID: 0xFFE0~FFEF**) to read next image version when number of image is greater than 5. The number of device img versions is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

Name	**Field Requirement**	**Format**	**Value**
Image Version	mandatory	Uint32*N	

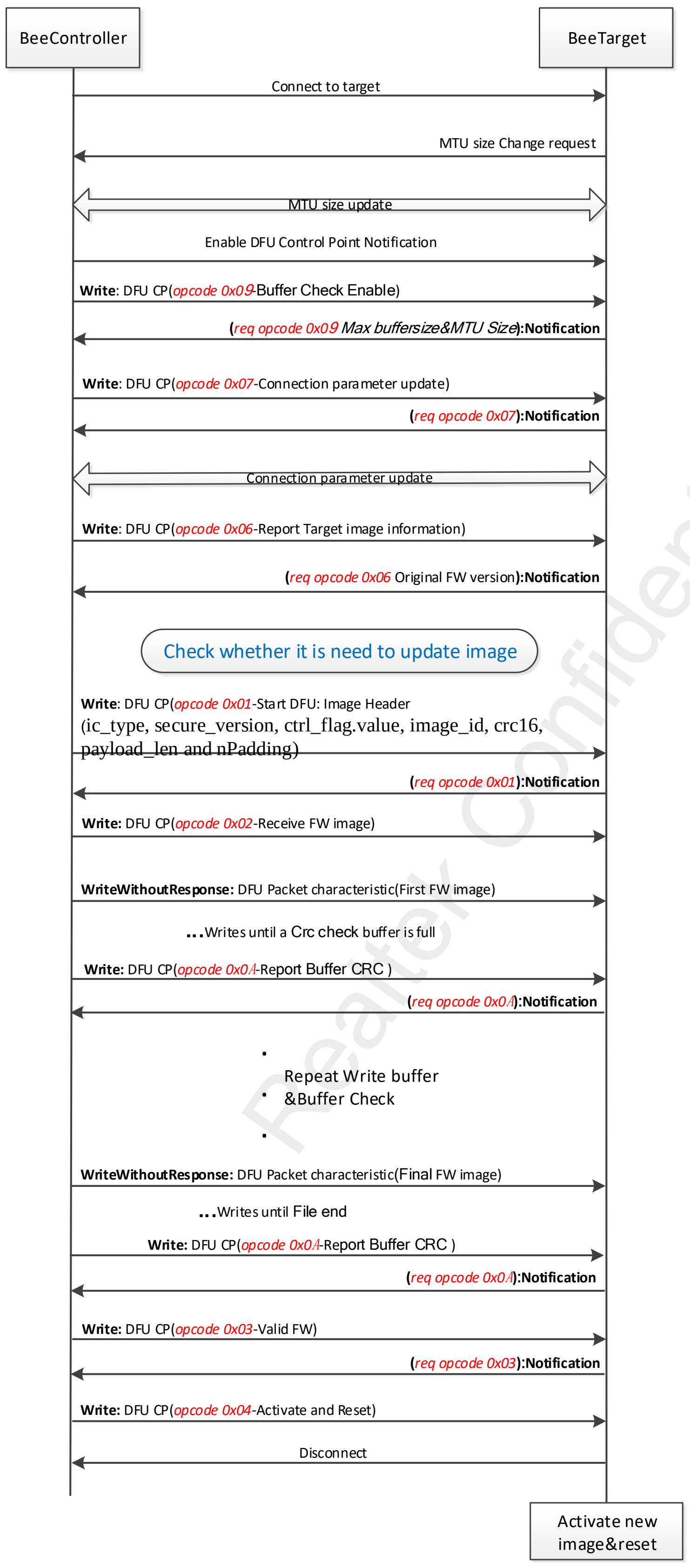
Table 5.12: Image Counter characteristics

5.3 OTA Procedure

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

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 0x0A-Report Buffer CRC)
(req opcode 0x0A):Notification
Repeat Write buffer &Buffer Check
WriteWithoutResponse: DFU Packet characteristic(Final FW image)
...Writes until File end
Write: DFU CP(opcode 0x0A-Report Buffer CRC)
(req opcode 0x0A):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

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