Espressif IOT SDK User Manual

Version 1.3.0

Espressif Systems IOT Team

Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice.

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The Wi-Fi Alliance Member Logo is a trademark of the Wi-Fi Alliance.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

Table of Contents

1. Preambles....5
2. Development Tools....6
   2.1. Serial Port Tool – SecureCRT....6
   2.2. Download Tools: FLASH_DOWNLOAD_TOOLS....6
3. SDK Software Package....8
4. Compilation....10
   4.1. Compilation for Version 0.9.5 SDK and After....10
   4.2. Compilation for Version 0.9.5 SDK and After....12
5. Flash Map....14
   5.1. none boot - can's support upgrade through WiFi....14
      1. 512KB flash....14
      2. 1024KB flash....15
      3. 2048KB flash....16
      4. 4096KB flash....17
   5.2. with boot - support upgrade through WiFi (FOTA)....18
      1. 512KB flash....18
      2. 1024KB flash....18
      3. 2048KB flash....19
      4. 4096KB flash....20
6. Writing Image Into Flash....21
   6.1. Without Support For Cloud Update (FOTA) 21
      1. 512KB Flash....21
      2. 1024KB Flash....22
      3. 2048KB Flash....22
      4. 4096KB Flash....22
   6.2. Version that support Cloud Update (FOTA) 23
      1. 512KB Flash....23
      2. 1024KB Flash....23
      3. 2048KB Flash....24
      4. 4096KB Flash....24

7. Appendix 25

1. Preambles

This manual introduces the setting up of toolchain, and codes for ESP8266-based SDK for Internet of Things.

More information can be found at Espressif's BBS: http://bbs.espressif.com/

The user starter guide can be found at: http://bbs.espressif.com/viewforum.php?f=21

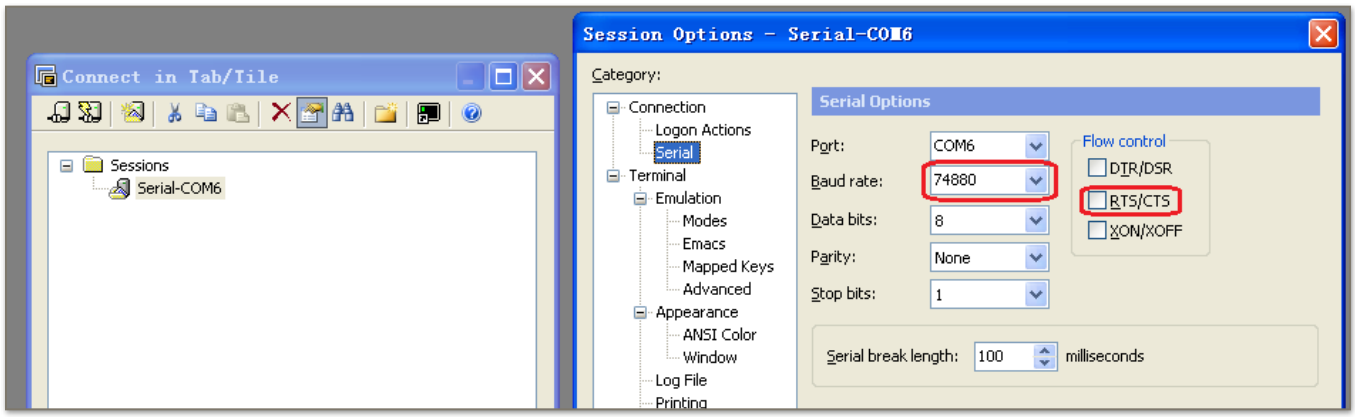
2. Development Tools

Use download tool to download the firmware to flash, use serial port tool to print logs to debug.

2.1. Serial Port Tool - SecureCRT

Here use SecureCRT as an example of serial port tool, in fact, you can use any other serial port tool to debug.

ESP8266 module adopts 74880 baud rate which can be set in SecureCRT.



2.2. Download Tools: FLASH_DOWNLOAD_TOOLS

Espressif Systems official flash download tool "ESP_FLASH_DOWNLOAD_TOOL" can be downloaded from BBS : http://bbs.espressif.com/viewtopic.php?f=57&t=433

Users can burn several bin files altogether at once, and download several complied *.bin files at a time into the SPI Flash on the ESP8266 motherboard.

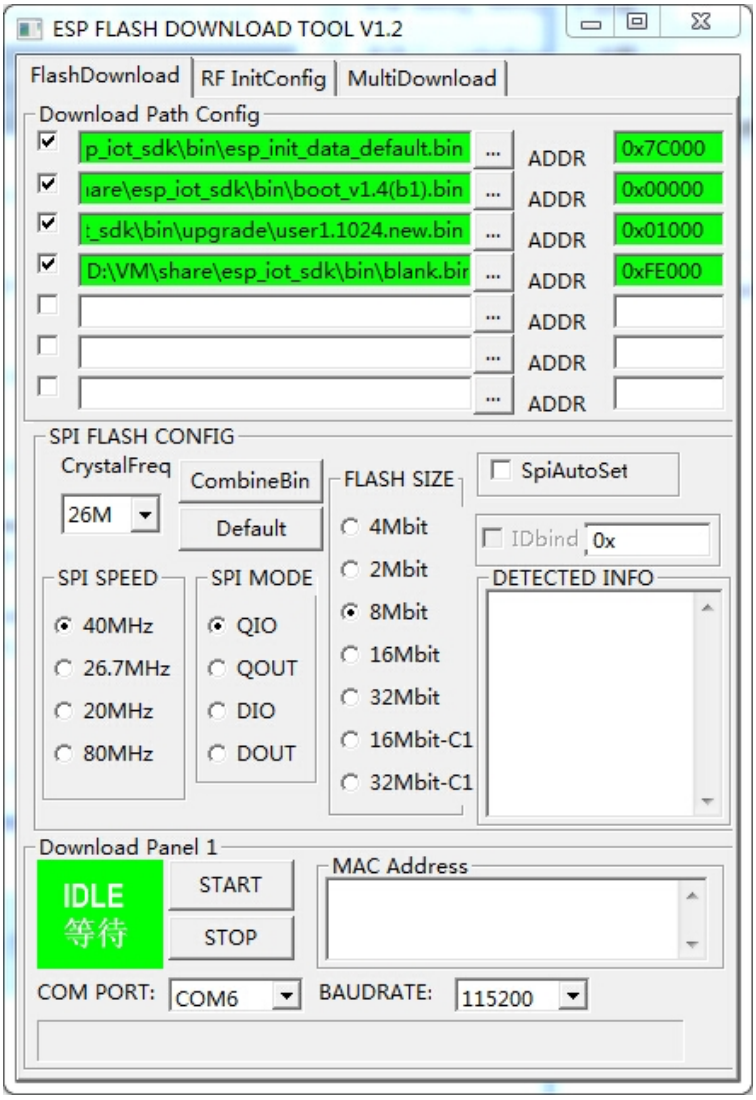
Using **ESP_FLASH_DOWNLOAD_TOOL**:

1. Bin-Select Area: Choose bins to burn, and burn them in corresponding address.
2. SPI FLASH CONFIG: Set config of SPI flash. "`CombineBin`" merges all bins selected above to one (`target.bin`). "`Default`" reset to the default config.
3. Mac Address: MAC address of ESP8266.

Also set the jumper on the motherboard as **MTDO:0**, **GPIO0:0**, **GPIO2:1**; this causes the chip to enter the download mode. Steps are as follows:

- See the red boxes in the picture above, select the bin file to be written → fill in the path → check burning options.

- Set COM port and baud rate.
- Click "START" to start downloading.
- After the downloading, disconnect the power for the motherboard, and change the jumper into operation mode. Re-connect the power for operation.

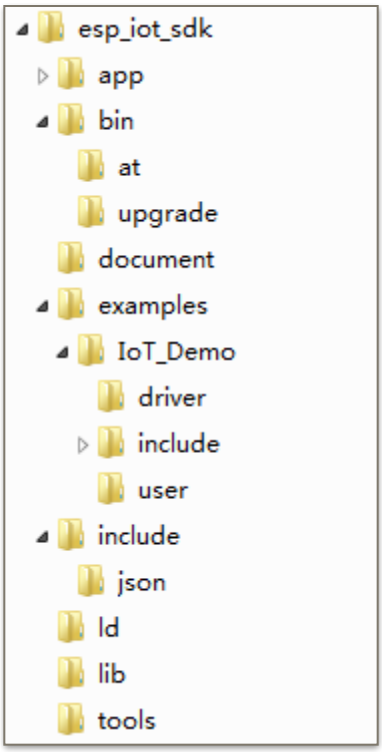


Set the jumper on the motherboard as **MTDO:0**, **GPIO0:1**, **GPIO2:1** for operating mode.

PS: Please disconnect the power when setting the jumper.

3. SDK Software Package

All header files, library files and compilation files needed for secondary development are included in the SDK software package. See the picture below for directory structure:



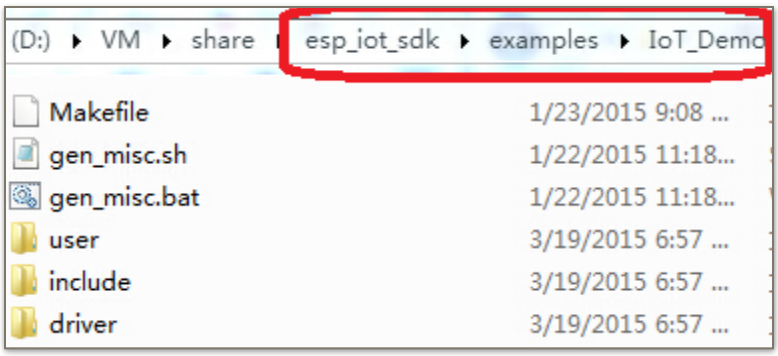
Detailed description:

- The "app" folder is the main working folder, we need to copy source codes to this folder to compile.
- "bin" folder stores the bin files downloaded into the Flash:
  - "at" folder :stores the bin files that support AT+ instructions, provided by Espressif;
  - "upgrade" folder :stores the bin files that support cloud update, generate by compilation;
  - "bin" folder root:stores the bin files that don't support cloud update, generate by compilation, and other bin files provided by Espressif.
- "examples" folder stores SDK examples, we need to copy the source code here (all files in the IoT_Demo folder) to "app" folder;
- "include" folder stores the header files pre-installed in the SDK, which may include relevant API functions and other definitions. Users can use them directly and do not need to change anything;

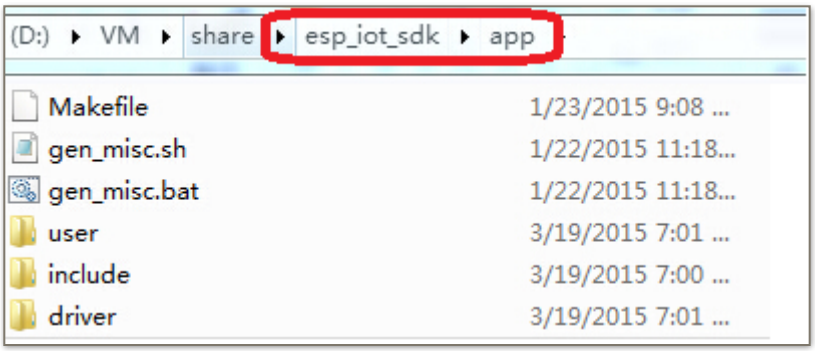
- "ld" folder stores the files needed for SDK software link. Users can use them directly and do not need to change anything;
- "lib" folder stores the library files needed for SDK compilation; "tools" folder stores the tools needed for generating bin files. Users can use them directly and do not need to change anything.

4. Compilation

When compiling, please remember to copy the sub-folders in the `esp_iot_sdk/examples/IOT_Demo` to `esp_iot_sdk/app`.



Copy all files in the picture above to `esp_iot_sdk/app` to compile.



4.1. Compilation for Version 0.9.5 SDK and After

With the release of `esp_iot_sdk_v0.9.5`, the compile process was simplified with a script in the APP folder.

Compile : `./gen_misc.sh`

```
esp8266@esp8266-VirtualBox:~/Share/esp_iot_sdk/app$ ./gen_misc.sh
Please follow below steps(1-5) to generate specific bin(s):
STEP 1: choose boot version(0=boot_v1.1, 1=boot_v1.2+, 2=none)
enter(0/1/2, default 2):
```

Then follow the tips and steps.

STEP 1 : boot version	
0	boot_v1.1, old version boot, support FOTA (firmware upgrade through Wi-Fi)
1	boot_v1.2+, new version boot, always recommend to use the latest boot.bin, support FOTA
2	none boot, generate `eagle.flash.bin` and `eagle.irom0text.bin`, can't FOTA
STEP 2 : bin generated	
0	input 2 in STEP 1, generate `eagle.flash.bin` and `eagle.irom0text.bin`, can't FOTA
1	input 0 or 1 in STEP 1, generate `user1.bin`, support FOTA
2	input 0 or 1 in STEP 1, generate `user2.bin`, support FOTA
STEP 3 : SPI flash configuration（ SPI speed ）	
0	SPI speed 20MHz, according to your actual SPI flash, set same configuration while downloading by ESP flash download tool.
1	SPI speed 26.7MHz, according to your actual SPI flash, set same configuration while downloading by ESP flash download tool.
2	SPI speed 40MHz, according to your actual SPI flash, set same configuration while downloading by ESP flash download tool.
3	SPI speed 80MHz, according to your actual SPI flash, set same configuration while downloading by ESP flash download tool.
STEP 4 : SPI flash configuration（ SPI mode ）	
0	QIO, according to your actual SPI flash, set same configuration while downloading by ESP flash download tool.
1	QOUT, according to your actual SPI flash, set same configuration while downloading by ESP flash download tool.
2	DIO, according to your actual SPI flash, set same configuration while downloading by ESP flash download tool.
3	DOUT, according to your actual SPI flash, set same configuration while downloading by ESP flash download tool.
STEP 5 : SPI flash configuration（ SPI flash size & map ）	
0	flash size 512KB, flash map of program area is 256KB + 256KB
2	flash size 1024KB, flash map of program area is 512KB + 512KB
3	flash size 2048KB, only the first 1024KB to be program area, flash map of program area is 512KB + 512KB

4	flash size 4096KB, only the first 1024KB to be program area, flash map of program area is 512KB + 512KB
5	flash size 2048KB, flash map of program area is 1024KB + 1024KB Only be supported since sdk_v1.1.0 + boot 1.4 + flash download tool_v1.2 and later version
6	flash size 4096KB, only the first 2048KB to be program area, flash map of program area is 1024KB + 1024KB Only be supported since sdk_v1.1.0 + boot 1.4 + flash download tool_v1.2 and later version

Notice,

- `none boot` : generate `eagle.flash.bin` and `eagle.irom0text.bin` which do not support FOTA.
- `boot_v1.1` & `boot_v1.2` : we recommend using the latest boot; choosing boot in compilation will get `user1.bin` or `user2.bin` which support FOTA (firmware upgrade through WiFi)
- After compiled `user1.bin`, please call `make clean` to clean up the temporary files generated by last compilation first, then compile `user2.bin`.
- Compile succeeds: it shows the address for the bins to be written to. For example:

```
eagle.app.v6.flash.bin---------->addr:0x00000
eagle.app.v6.irom0text.bin------>addr:0x40000
!!!
esp8266@esp8266-VirtualBox:~/Share/esp_iot_sdk/app$
```

Or,

```
Generate user1.512.old.bin successully in folder bin/upgrade.
Support boot_v1.1 and +
user1.512.old.bin------>addr:0x1000
!!!
esp8266@esp8266-VirtualBox:~/Share/esp_iot_sdk/app$
```

4.2. Compilation for Version 0.9.5 SDK and After

For `esp_iot_sdk_v0.9.4` and before, FW does not support upgrade through WiFi compiled by `./gen_misc.sh`.

FW support upgrade through WiFi (FOTA) compiled as:

(1) Run `./gen_misc_plus.sh 1` to generate `user1.bin` at `/esp_iot_sdk/bin/upgrade`

(2) Run `make clean` to clean up all previous compilation

(3) Run `./gen_misc_plus.sh 2` to generate `user2.bin` at `/esp_iot_sdk/bin/upgrade`

Note:

1) Please refer to document "**Firmware update through cloud server**" for details about FOTA.

2) `esp_iot_sdk_v0.7` and previous versions do not support FOTA.
3) `esp_iot_sdk_v0.8` and later versions support cloud update and are compatible with previous compilation and burning methods.

5. Flash Map

Different settings of STEP 1 and STEP 5 in compilation leads to different flash size and flash map.

Note

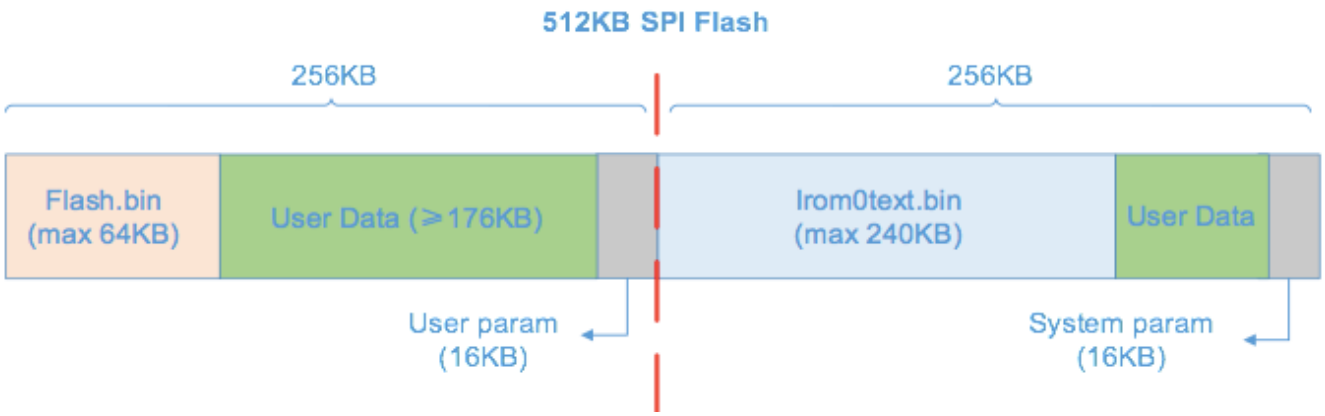
- System param (system parameter area) is the last 16KB of flash.
- User param is the user parameter area used by Espressif demo code (IOT_Demo or AT). If users develop their own application, user data can be saved in any available flash area.
- User Data area (green area in pictures below) means the flash area that may be available, if program area doesn't reach the maximum size, remaining area can be used to save user data.

5.1. none boot - can's support upgrade through WiFi

Choose `2 none boot` in `STEP 1` of compilation to generate `eagle.flash.bin` (hereinafter called `flash.bin`) and `eagle.irom0text.bin` (hereinafter called `irom0text.bin`). Then choose different flash map in `STEP 5` according to your actual SPI flash.

1. 512KB flash

If choose `2 none boot` in `STEP 1`, choose `0 512KB` in `STEP 5`, flash map will be as below

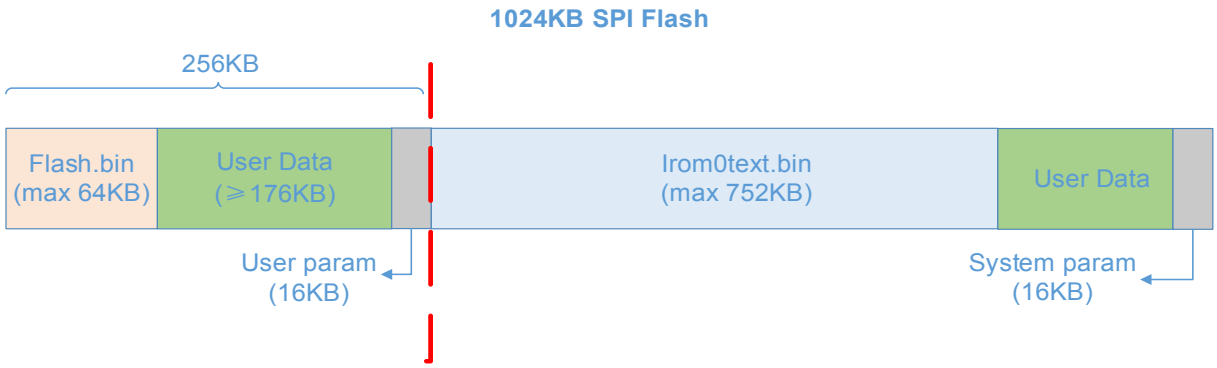


- User Data area: if program area (`flash.bin` and `irom0text.bin`) doesn't fill flash, the remaining area can be used to store user data.
- `irom0text.bin` default to be less than 200KB ; for 512KB flash, user can revise ld file which for compilation, to make the maximum size of `irom0text.bin` to be 256 – 16 = 240 KB
- In "`eagle.app.v6.ld`" (`\esp_iot_sdk\ld`), "`len`" of "`irom0_0_seg` " means the maximum size of `irom0text.bin` . For 512KB flash, it can be revised to `0x3C000` at most, `irom0text.bin` can be 240 KB.

```
MEMORY
{
  dport0_0_seg :                  org = 0x3FF00000, len = 0x10
  dram0_0_seg :                   org = 0x3FFE8000, len = 0x14000
  iram1_0_seg :                   org = 0x40100000, len = 0x8000
  irom0_0_seg :                   org = 0x40240000, len = 0x32000
}
```

2. 1024KB flash

If choose `2 none boot` in `STEP 1`, choose `2 1024KB` in `STEP 5`, flash map will be as below

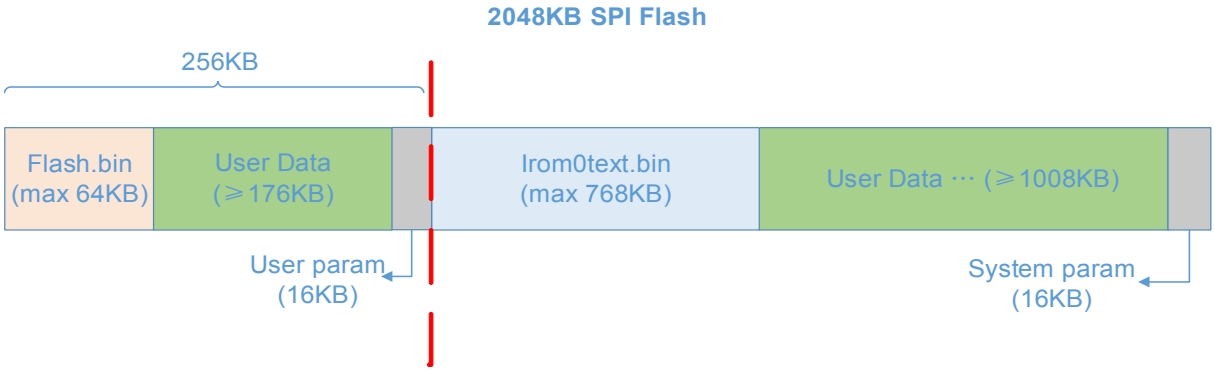


- User Data area: if program area (`flash.bin` and `irom0text.bin`) doesn't fill flash, the remaining area can be used to store user data.
- `irom0text.bin` default to be less than 200KB ; for 1024KB flash, user can revise ld file which for compilation, to make the maximum size of `irom0text.bin` to be 1024 - 256 – 16 = 752 KB
- In "`eagle.app.v6.ld`" (`\esp_iot_sdk\ld`), "`len`" of "`irom0_0_seg` " means the maximum size of `irom0text.bin`. For 1024KB flash, it can be revised to `0xBC000` at most, `irom0text.bin` can be 752 KB.

```
MEMORY
{
  dport0_0_seg :                  org = 0x3FF00000, len = 0x10
  dram0_0_seg :                   org = 0x3FFE8000, len = 0x14000
  iram1_0_seg :                   org = 0x40100000, len = 0x8000
  irom0_0_seg :                   org = 0x40240000, len = 0x32000
}
```

3. 2048KB flash

If choose `2 none boot` in `STEP 1`, choose `3 2048KB` in `STEP 5`, flash map will be as below

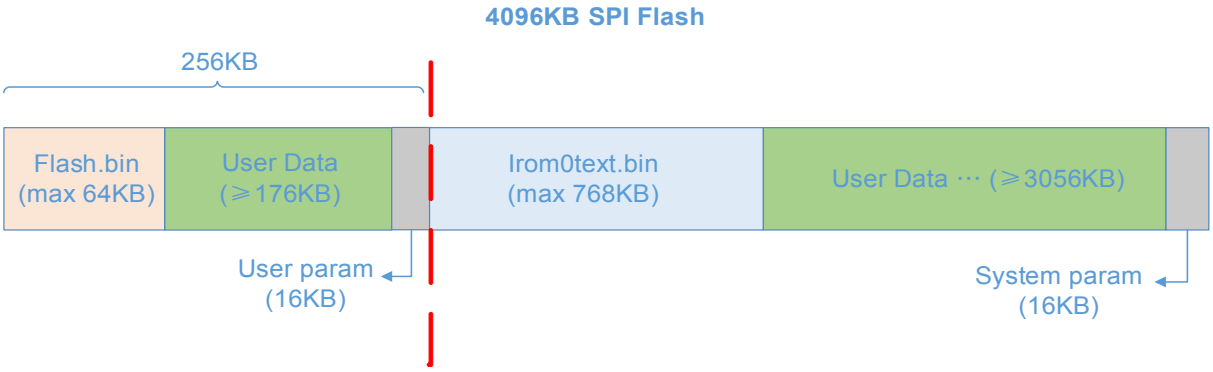


- User Data area: if program area (`flash.bin` and `irom0text.bin`) doesn't fill flash, the remaining area can be used to store user data.
- `irom0text.bin` default to be less than 200KB ; Only the first 1024KB can be program area now, so for 2048KB flash, user can revise ld file which for compilation, to make the maximum size of `irom0text.bin` to be 1024 - 256 = 768 KB
- In "`eagle.app.v6.ld`" (`\esp_iot_sdk\ld`), "`len`" of "`irom0_0_seg` " means the maximum size of `irom0text.bin`. For 2048KB flash, it can be revised to `0xC0000` at most,`irom0text.bin` can be 768 KB.

```
MEMORY
{
  dport0_0_seg :                  org = 0x3FF00000, len = 0x10
  dram0_0_seg :                   org = 0x3FFE8000, len = 0x14000
  iram1_0_seg :                   org = 0x40100000, len = 0x8000
  irom0_0_seg :                   org = 0x40240000, len = 0x32000
}
```

4. 4096KB flash

If choose `2 none boot` in `STEP 1`, choose `4 4096KB` in `STEP 5`, flash map will be as below



- User Data area: if program area (`flash.bin` and `irom0text.bin`) doesn't fill flash, the remaining area can be used to store user data.
- `irom0text.bin` default to be less than 200KB ; Only the first 1024KB can be program area now, so for 4096KB flash, user can revise ld file which for compilation, to make the maximum size of `irom0text.bin` to be 1024 - 256 = 768 KB
- In "`eagle.app.v6.ld`" (`\esp_iot_sdk\ld`), "`len`" of "`irom0_0_seg` " means the maximum size of `irom0text.bin`. For 4096KB flash, it can be revised to `0xC0000` at most,`irom0text.bin` can be 768 KB.

```
MEMORY
{
  dport0_0_seg :                  org = 0x3FF00000, len = 0x10
  dram0_0_seg :                   org = 0x3FFE8000, len = 0x14000
  iram1_0_seg :                   org = 0x40100000, len = 0x8000
  irom0_0_seg :                   org = 0x40240000, len = 0x32000
}
```

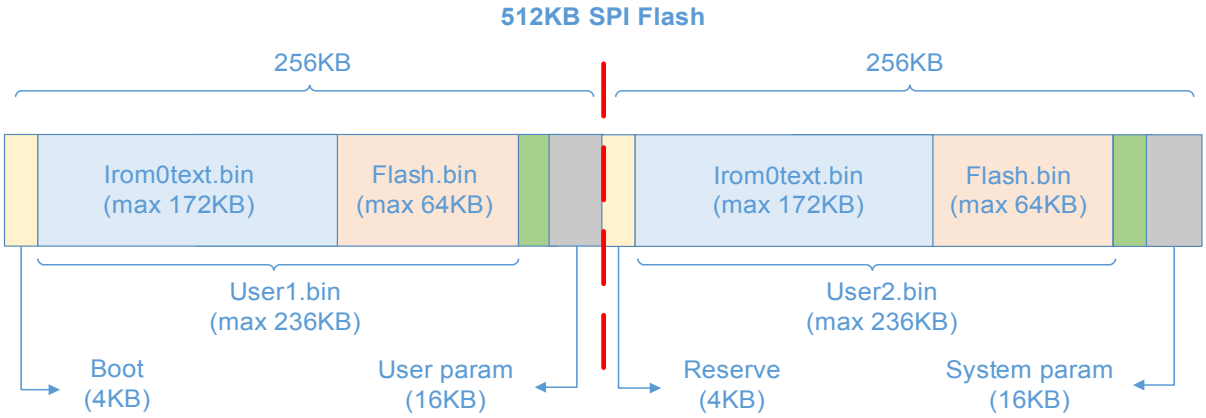
5.2. with boot - support upgrade through WiFi (FOTA)

Choose `1 boot_v1.2+` in `STEP 1` to support FOTA, generate `user1.bin` and `user2.bin`. Then choose different flash map in `STEP 5` according to your actual SPI flash.

- After generated `user1.bin` , call `make clean` first to clean the temporary files, then compile again to generate `user2.bin`
- `boot_v1.1` is a old version boot, compilation and downloading are same as `boot_v1.2+`, we recommended to use the latest version of `boot.bin`.
- User Data area (green area in pictures below) means the flash area that may be available, if program area (`user1.bin` and `user2.bin`) doesn't reach the maximum size, remaining area can be used to save user data.

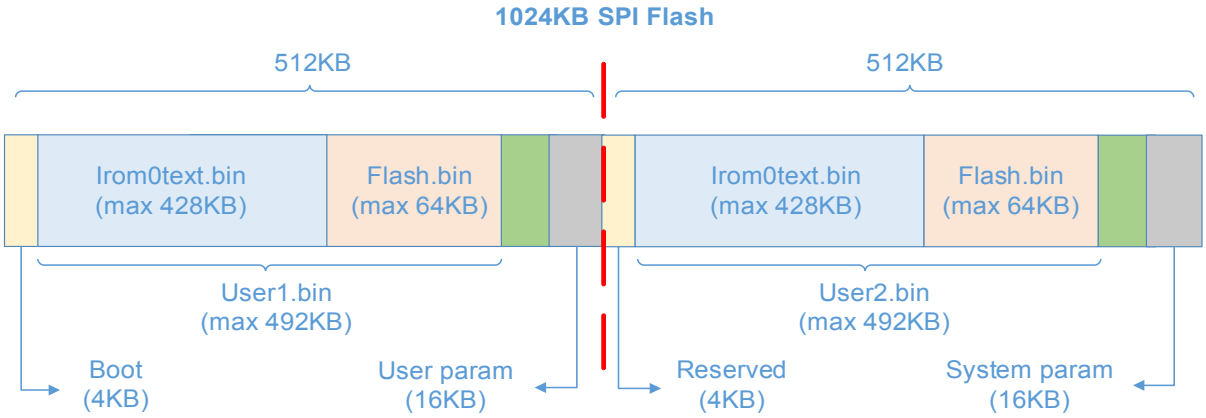
1. 512KB flash

If choose `1 boot_v1.2+` in `STEP 1`, choose `0 512KB` in `STEP 5`, flash map will be as below



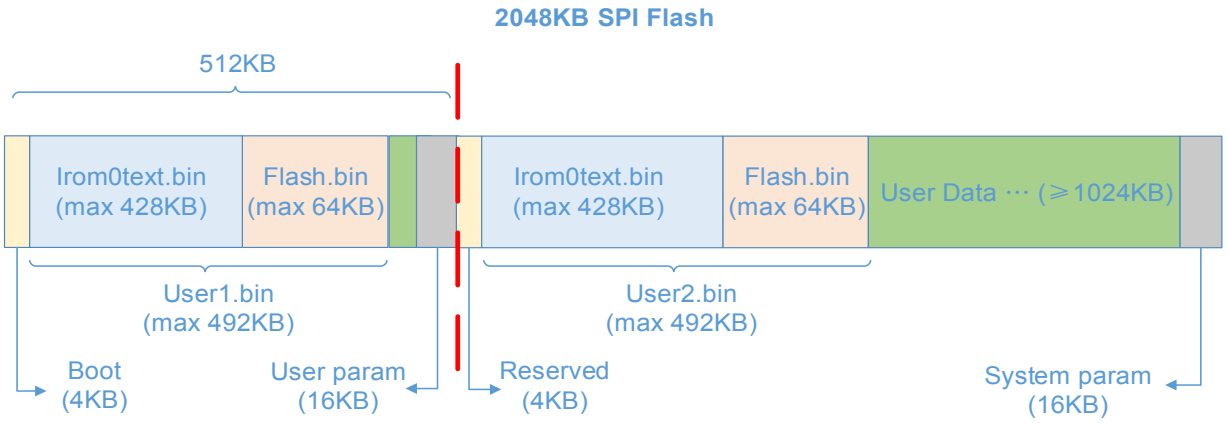
2. 1024KB flash

If choose `1 boot_v1.2+` in `STEP 1`, choose `2 1024KB` in `STEP 5`, flash map will be as below

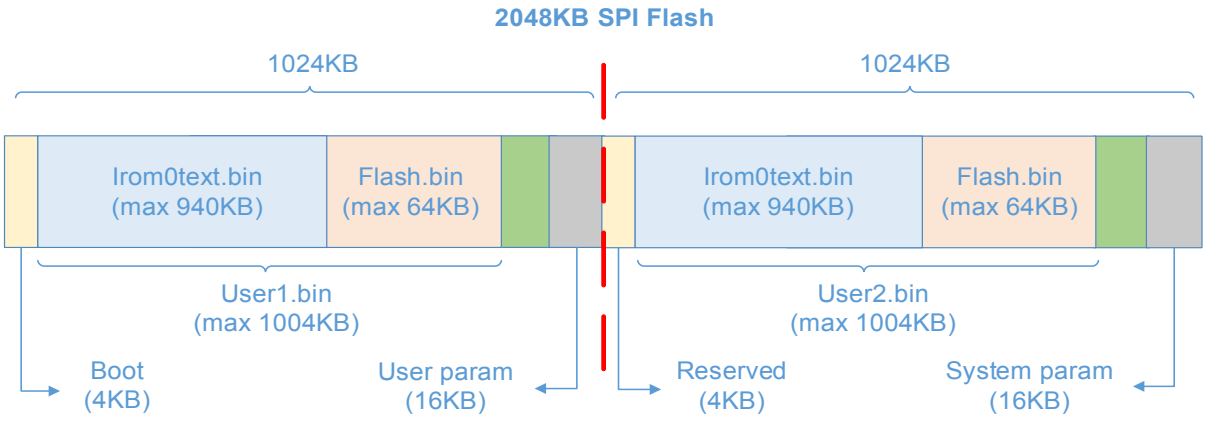


3. 2048KB flash

If choose `1 boot_v1.2+` in `STEP 1`, choose `3 2048KB` in `STEP 5`, only the first 1024KB to be the program area (512KB + 512KB), flash map will be as below

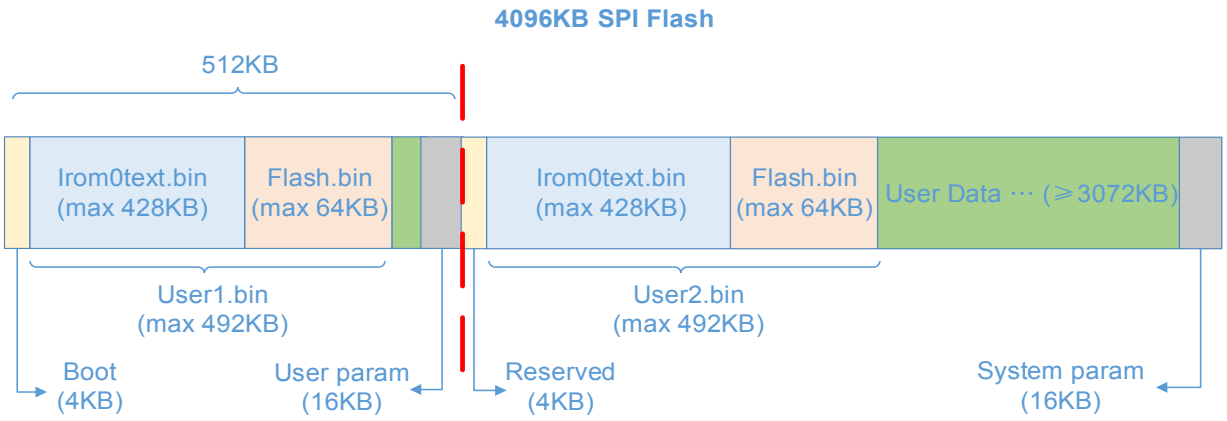


If choose `1 boot_v1.2+` in `STEP 1`, choose `5 2048KB` in `STEP 5` (only be supported since sdk_v1.1.0 + boot v1.4 + flash download tool v1.2 and later version), the program area is 1024KB + 1024KB, flash map will be as below

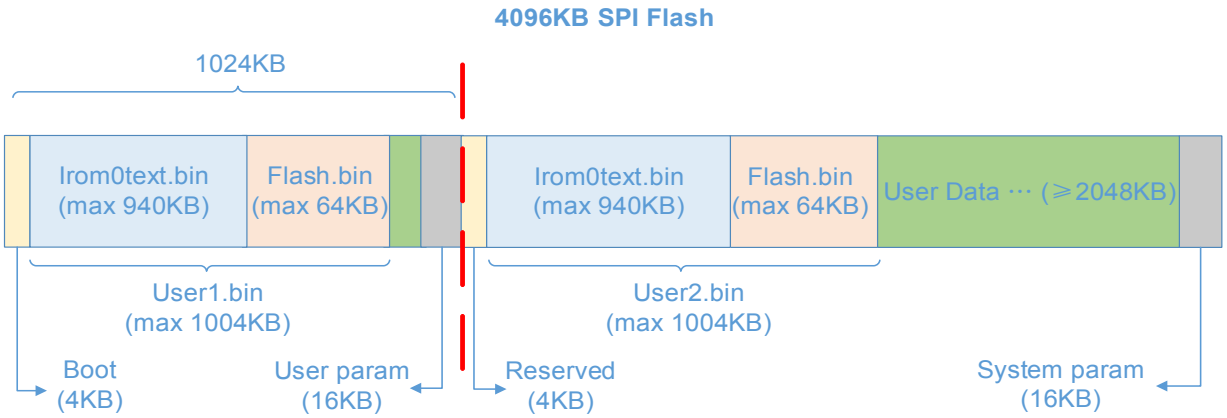


4. 4096KB flash

If choose `1 boot_v1.2+` in `STEP 1`, choose `4 4096KB` in `STEP 5`, only the first 1024KB to be the program area (512KB + 512KB), flash map will be as below



If choose `1 boot_v1.2+` in `STEP 1`, choose `6 4096KB` in `STEP 5` (only be supported since sdk_v1.1.0 + boot 1.4 + flash download tool v1.2 and later version), the first 2048KB to be the program area (1024KB + 1024KB), flash map will be as below



6. Writing Image Into Flash

According to compiling method and flash size, we can choose one of the following ways to write the image to the flash device.

Note:

- Flash system parameter area is the last 4 sectors of flash, 4KBytes per sector, so it's the last 16KB of flash;
- Flash user parameter area depends on user-defined application; in IOT_Demo which is using 512KB flash as example, user parameter area is 4 sectors start from `0x3C000`
- `master_device_key.bin` is needed if you are using Espressif Cloud， otherwise it need not to burn into Flash; it is only necessary for initial write-in and revision of `master_device_key`; in IOT_Demo, `master_device_key` is in the third sector of user parameter area.
- `blank.bin` as initialization, to be written to the last but one in flash;
- `esp_init_data_default.bin` stores default RF parameter values and to be written to the forth sector from the end of flash;
- How to use 1MB or larger flash can refer to BBS : http://bbs.espressif.com/viewtopic.php?f=10&t=305

6.1. Without Support For Cloud Update (FOTA)

1. 512KB Flash

bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users themselves to get Espressif Cloud service
esp_init_data_default.bin	0x7C000	Stores default RF parameter values, provided in SDK
blank.bin	0x7E000	Stores default system parameter values, provided in SDK
eagle.flash.bin	0x00000	Compiled by the steps said above
eagle.irom0text.bin	0x40000	Compiled by the steps said above

2. 1024KB Flash

bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users themselves to get Espressif Cloud service
esp_init_data_default.bin	0xFC000	Stores default RF parameter values, provided in SDK
blank.bin	0xFE000	Stores default system parameter values, provided in SDK
eagle.flash.bin	0x00000	Compiled by the steps said above
eagle.irom0text.bin	0x40000	Compiled by the steps said above

3. 2048KB Flash

bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users themselves to get Espressif Cloud service
esp_init_data_default.bin	0x1FC000	Stores default RF parameter values, provided in SDK
blank.bin	0x1FE000	Stores default system parameter values, provided in SDK
eagle.flash.bin	0x00000	Compiled by the steps said above
eagle.irom0text.bin	0x40000	Compiled by the steps said above

4. 4096KB Flash

bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users themselves to get Espressif Cloud service
esp_init_data_default.bin	0x3FC000	Stores default RF parameter values, provided in SDK
blank.bin	0x3FE000	Stores default system parameter values, provided in SDK
eagle.flash.bin	0x00000	Compiled by the steps said above
eagle.irom0text.bin	0x40000	Compiled by the steps said above

6.2. Version that support Cloud Update (FOTA)

Note:

- `User2.bin` need not to burn into Flash, it can be download through WiFi (FOTA)

1. 512KB Flash

bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users themselves to get Espressif Cloud service
esp_init_data_default.bin	0x7C000	Stores default RF parameter values, provided in SDK
blank.bin	0x7E000	Stores default system parameter values, provided in SDK
boot.bin	0x00000	Boot loader, provided in SDK, recommend to use the latest version
user1.bin	0x01000	Compiled by the steps said above
user2.bin	0x41000	Compiled by the steps said above, need not to be burned into flash, can be download through WiFi as upgrade

2. 1024KB Flash

bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users themselves to get Espressif Cloud service; to be written into the third sector of flash user parameter area which is 0x3E000 in IOT_Demo, can be changed by user. If using 1024KB flash, recommend to change it to 0x7E000, refer to BBS http://bbs.espressif.com/viewtopic.php?f=10&t=305
esp_init_data_default.bin	0xFC000	Stores default RF parameter values, provided in SDK
blank.bin	0xFE000	Stores default system parameter values, provided in SDK
boot.bin	0x00000	Boot loader, provided in SDK, recommend to use the latest version
user1.bin	0x01000	Compiled by the steps said above
user2.bin	0x81000	Compiled by the steps said above, need not to be burned into flash, can be download through WiFi as upgrade

3. 2048KB Flash

bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users themselves to get Espressif Cloud service; to be written into the third sector of flash user parameter area which is 0x3E000 in IOT_Demo, can be changed by user. If choose 3 in STEP 5, please change it to 0x7E000, refer to BBS http://bbs.espressif.com/viewtopic.php?f=10&t=305 If choose 5 in STEP 5, change it to 0xFE000, and download it to 0xFE000.
esp_init_data_default.bin	0x1FC000	Stores default RF parameter values, provided in SDK
blank.bin	0x1FE000	Stores default system parameter values, provided in SDK
boot.bin	0x00000	Boot loader, provided in SDK, recommend to use the latest version
user1.bin	0x01000	Compiled by the steps said above
user2.bin	0x81000	Compiled by the steps said above, need not to be burned into flash, can be download through WiFi as upgrade

4. 4096KB Flash

bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users themselves to get Espressif Cloud service; to be written into the third sector of flash user parameter area which is 0x3E000 in IOT_Demo, can be changed by user. If choose 4 in STEP 5, please change it to 0x7E000, refer to BBS http://bbs.espressif.com/viewtopic.php?f=10&t=305 If choose 6 in STEP 5, change it to 0xFE000, and download it to 0xFE000.
esp_init_data_default.bin	0x3FC000	Stores default RF parameter values, provided in SDK
blank.bin	0x3FE000	Stores default system parameter values, provided in SDK
boot.bin	0x00000	Boot loader, provided in SDK, recommend to use the latest version
user1.bin	0x01000	Compiled by the steps said above
user2.bin	0x81000	Compiled by the steps said above, need not to be burned into flash, can be download through WiFi as upgrade

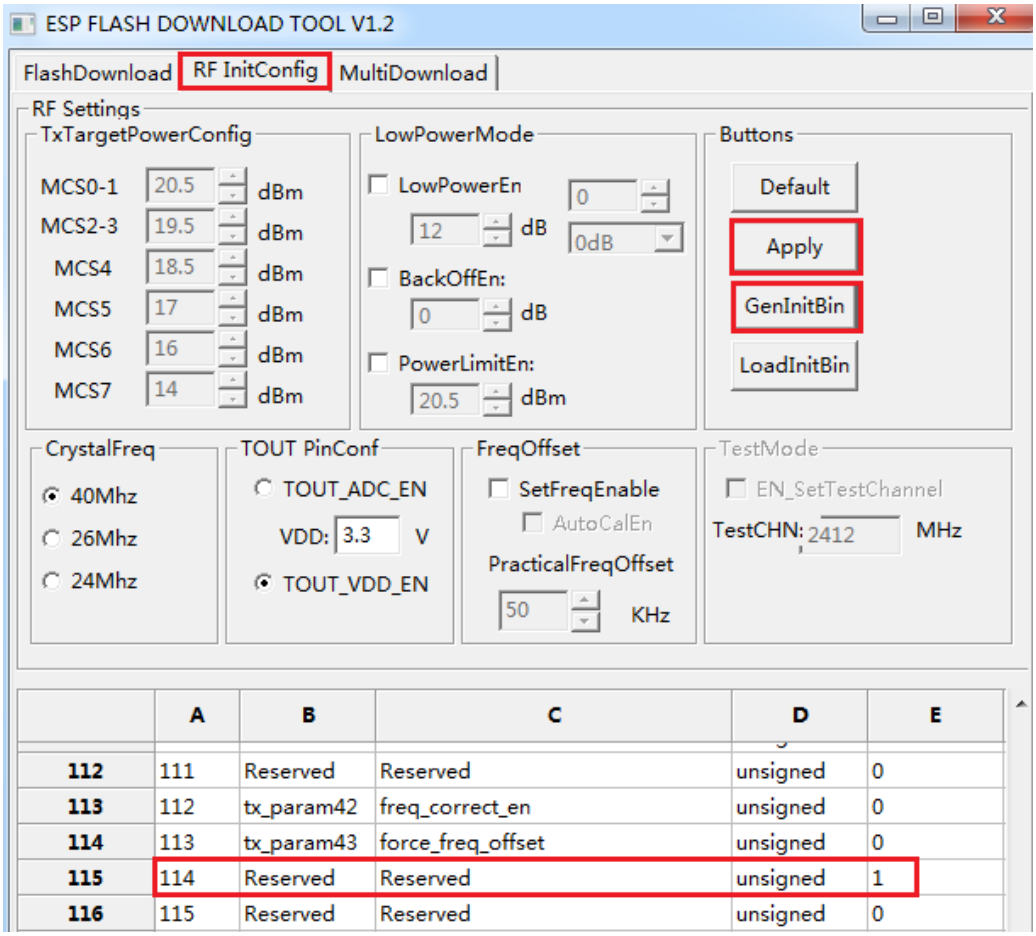
7. Appendix

Q : How to short the booting time (include awakened from deep sleep) ?

A : RF will do self-calibration by default during booting. But since **esp_iot_sdk_v1.3.0**, users can configure RF to do calibration only at the first-time power-on, and store the calibration parameters into the flash, so that they don't need to calibrate RF again when the chip is booting up.

⚠ **Notice**: In this way, the fifth sector from the bottom of the flash will have to be occupied to store RF calibration parameters, and therefore the system parameter area will increase from 16KB to 20KB. Due to that 4 KB's increase, the adjacent or related area, as shown in the diagram in previous Chapter of **Flash Map**, will correspondingly decrease by 4 KB.

(1) Click "RF InitConfig" in tool "ESP FLASH DOWNLOAD TOOL"



	A	B	C	D	E
112	111	Reserved	Reserved	unsigned	0
113	112	tx_param42	freq_correct_en	unsigned	0
114	113	tx_param43	force_freq_offset	unsigned	0
115	114	Reserved	Reserved	unsigned	1
116	115	Reserved	Reserved	unsigned	0

(2) The table at the bottom is the configuration of esp_init_data_default.bin (0~127 byte), change byte 114 to be 1 as picture above.

(3) Click "Apply" to validate the new configuration.

(4) Click "GenInitBin" to generate a new bin "esp_init_data_setting.bin", and download it into Flash instead of esp_init_data_default.bin.