ESP8266

SDK

Getting Started Guide

Version 1.5

About This Guide

This document gives instructions on building ESP8266 SDK compiling environment and compiling procedures. Programming reference such as binaries and flash map are also provided.

The document is structured as follow.

Chapter	Title	Subject
Chapter 1	Setup Compiling Environment	Provides instructions to setup a Linux environment for SDK compilation.
Chapter 2	Compile SDK	Provides introductions to SDK software package SDK compilation procedures..
Chapter 3	Download Binaries to Flash	Provides instructions for download SDK files to flash according to the flash size.
Chapter 4	Flash Map	Provides file allocations information in flash of different sizes.

Note :

Go to online Tutorial for Beginners for instructions to use the software development kit.

Note :

Go to Espressif BBS for more product information.

Release Notes

Date	Version	Release notes
2015.12	V1.5	

Table of Contents

1. Setup Compiling Environment 4
   1.1. Download Virtual Machine Software and Image 4
   1.2. Install and Import Virtual Computer 4
   1.3. Create Share Folder 7

2. Compile SDK 8
   2.1. SDK Software Package 8
   2.1. Prepare Compilation 9
   2.2. Compile SDK of v0.9.5 and Later Version 11
   2.3. Compile SDK of v0.9.4 and Previous Version 13

3. Download Binaries to Flash 15
   3.1. Tools 15
      3.1.1. Minicom 15
      3.1.2. ESP8266 FLASH TOOL 16
   3.2. Binaries and Addresses 16
      3.2.1. No Cloud Update 16
      3.2.2. Support Cloud Update 17

4. Flash Map 20
   4.1. None boot - No cloud Update 20
      4.1.1. 512 KB Flash 20
      4.1.2. 1024 KB Flash 21
      4.1.3. 2048 KB Flash 21
      4.1.4. 4096 KB Flash 22
   4.2. With boot - Support Cloud Update 23
      4.2.1. 512 KB Flash 23
      4.2.2. 1024KB Flash 24
      4.2.3. 2048 KB Flash 24
      4.2.4. 4096 KB Flash 25

Appendix 27

1. Setup Compiling Environment

All development tools for ESP8266 Internet of Things (IOT) module have been installed on a virtual machine. Users need to install the virtual machine to setup an Linux compiling environment for SDK compilation.

1.1. Download Virtual Machine Software and Image

VirtualBox is used for Linux virtualization. Version 4.3.12 is recommended.

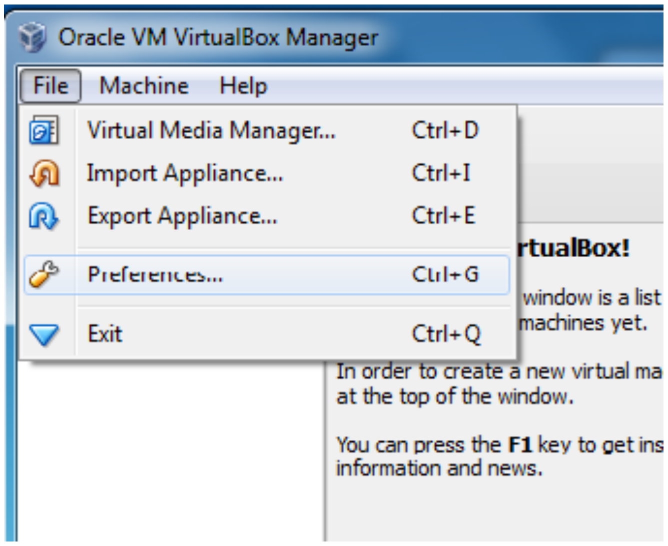
Espressif provides VirtualBox OVA image which can be imported into other virtual machine software.

Password: **qudl**

1.2. Install and Import Virtual Computer

Double-click **Virtualbox-4.3.12-xxx.exe** to install VirtualBox. By default, VirtualBox imports virtual computer on system disk which takes a lot of system space. It is suggested to import virtual computer on non-system disk. Follow the instructions to import and configure OVA image as below:

1. Open Virtualbox and select **Preference** from the **File** context menu.


Oracle VM VirtualBox Manager
File
Machine
Help
Virtual Media Manager... Ctrl+D
Import Appliance... Ctrl+I
Export Appliance... Ctrl+E
Preferences... Ctrl+G
Exit Ctrl+Q

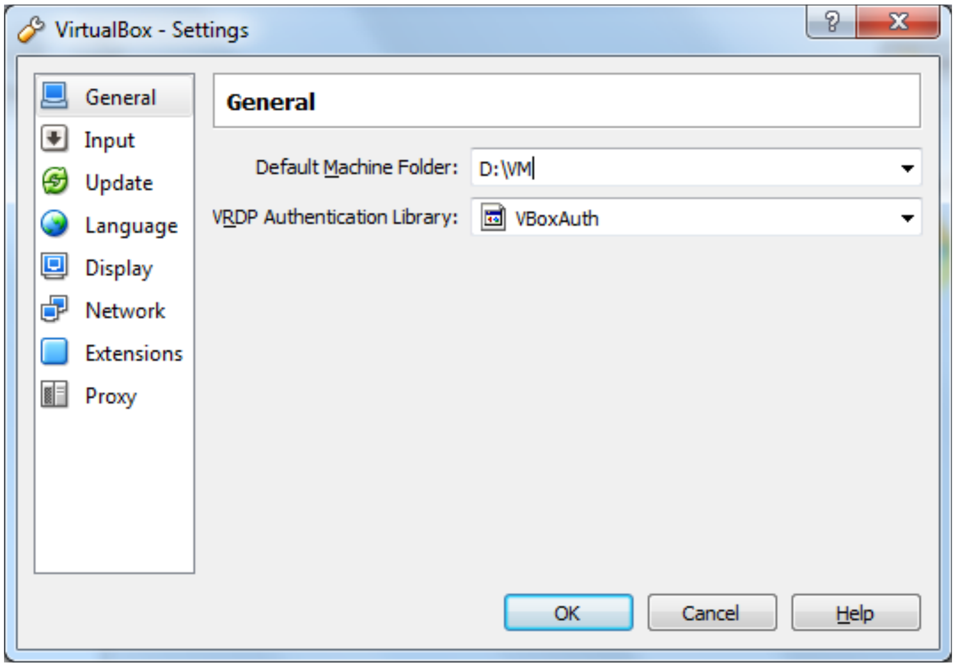
rtualBox!

window is a list
machines yet.

In order to create a new virtual ma
at the top of the window.

You can press the **F1** key to get ins
information and news.

2. Select **General** and set the default machine folder where the virtual computer locates. For example: D:\VM.

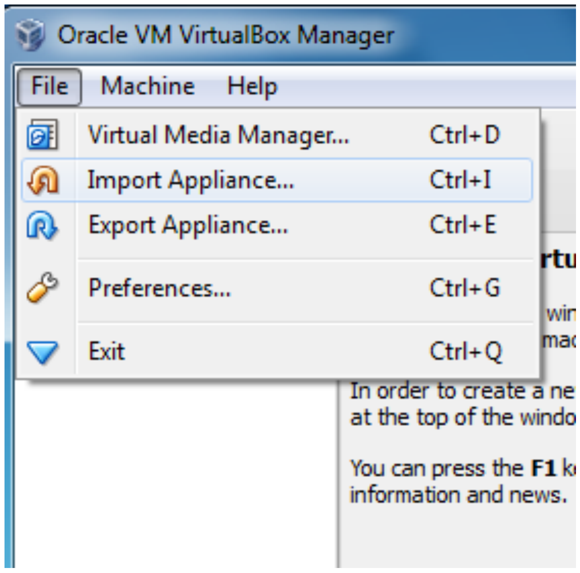


General

Default Machine Folder: D:\VM

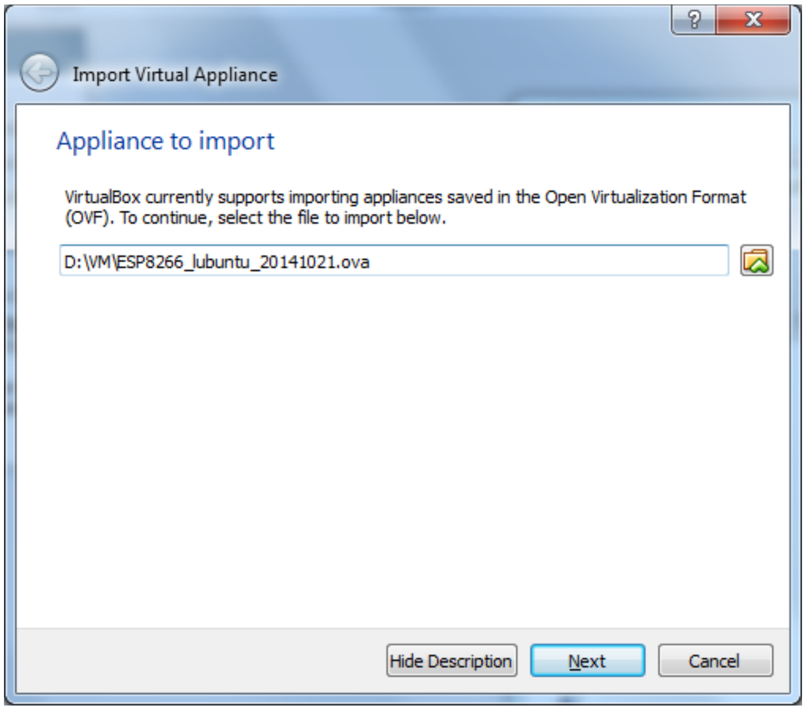
VRDP Authentication Library: VBoxAuth

3. Select **Import Appliance** from File context menu. Set the path for the OVA image to be imported. For example: D:\vm\ESP8266_lubuntu_20141021.ova.


Oracle VM VirtualBox Manager
File
Machine
Help
Virtual Media Manager... Ctrl+D
Import Appliance... Ctrl+I
Export Appliance... Ctrl+E
Preferences... Ctrl+G
Exit Ctrl+Q

In order to create a ne
at the top of the windo

You can press the **F1** k
information and news.

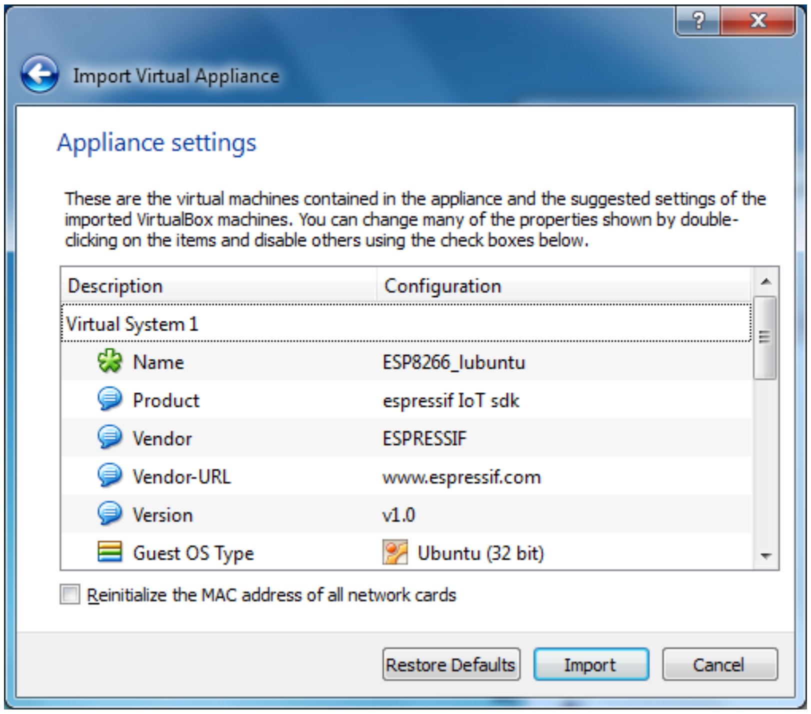


Appliance to import

VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

D:\VM\ESP8266_lubuntu_20141021.ova

4. Click **Import** to confirm the settings.



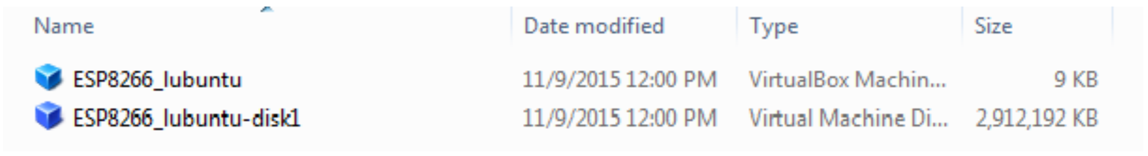
Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

Description	Configuration
Virtual System 1	
Name	ESP8266_lubuntu
Product	espressif IoT sdk
Vendor	ESPRESSIF
Vendor-URL	www.espressif.com
Version	v1.0
Guest OS Type	Ubuntu (32 bit)

☐ Reinitialize the MAC address of all network cards

5. Following files can be found in the directory where the virtual machine locates. For example, D:\VM\ESP_IOT_SDK.



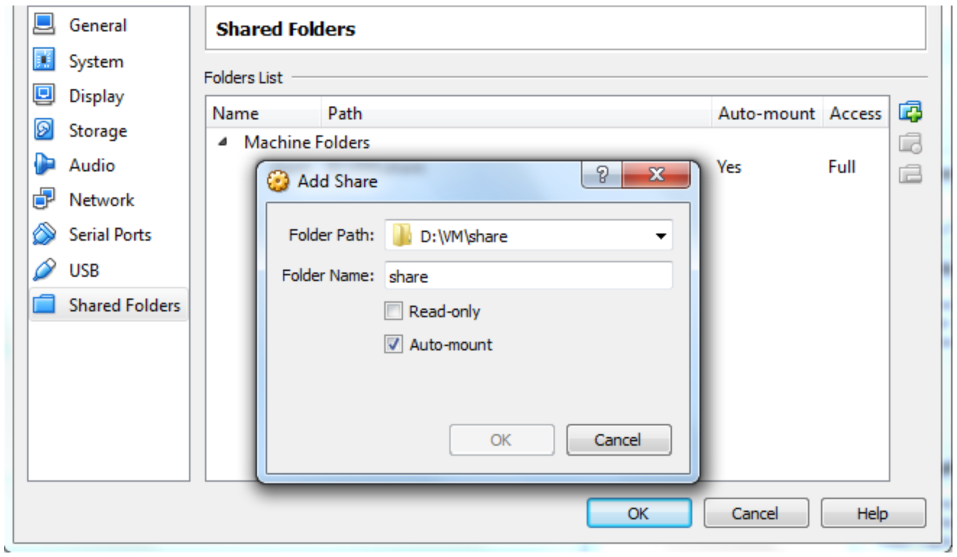
Name	Date modified	Type	Size
ESP8266_lubuntu	11/9/2015 12:00 PM	VirtualBox Machin...	9 KB
ESP8266_lubuntu-disk1	11/9/2015 12:00 PM	Virtual Machine Di...	2,912,192 KB

—END

1.3. Create Share Folder

Share the folders in the hosting machine with the virtual machine before using the virtual machine, for example, create a file holder named "share", which is dedicated to mapping the virtual computer, and the hosting machine can share files with the virtual machine by copying files into this file holder.

In VM VirtualBox Manager, click **Settings**, select **Machine Folders** in **Shared Folders**, and then click on the right side to set up a new share folder.



Shared Folders

Folders List

Name	Path	Auto-mount	Access
Machine Folders			
		Yes	Full

Add Share

Folder Path: D:\VM\share

Folder Name: share

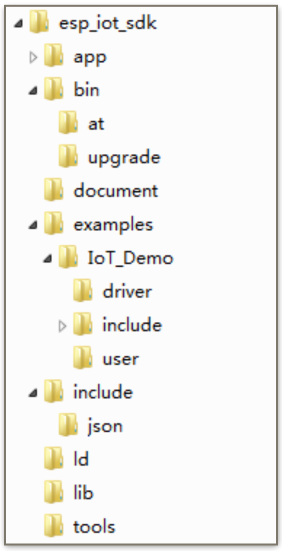
☐ Read-only

☑ Auto-mount

2. Compile SDK

2.1. 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:



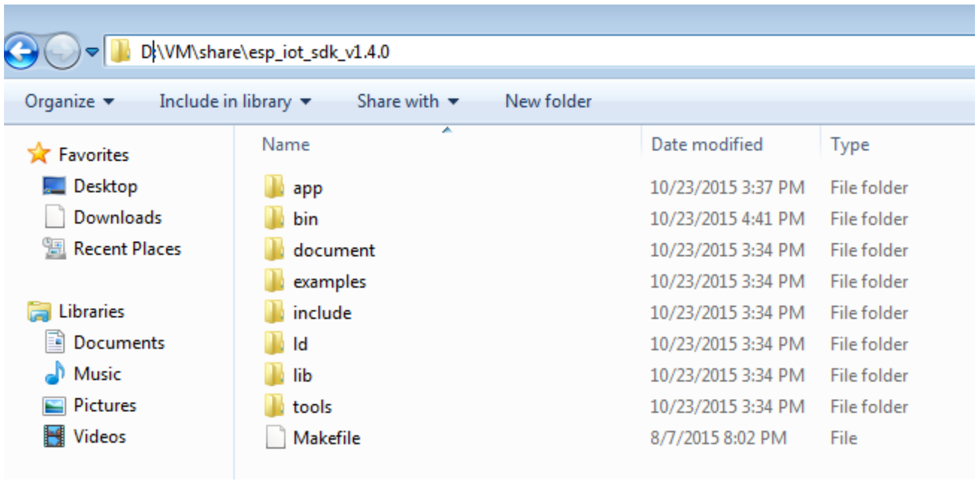
- **app** folder is the main working directory which contains source codes to be compiled.
- **bin** folder stores the bin files to be downloaded into the Flash:
  - **at** folder stores the bin files that support AT instructions provided by Espressif Systems.
  - **upgrade** folder stores the bin files generated by compilation which support FOTA (**user1.bin** or **user2.bin**).
  - The root directory stores the bin files generated by compilation which do not support FOTA and other bin files provided by Espressif Systems.
- **examples** folder stores example code (in IoT_Demo folder for example) which need to be duplicated to app folder if applicable.
- **include** folder stores the header files pre-installed in the SDK, which may include relevant API functions and other macro definitions. No modifications are needed.
- **ld** folder stores the files needed for SDK software link. No modifications are needed.
- **lib** folder stores the library files needed for SDK compilation.
- **tools** folder stores the tools needed for generating bin files. No modifications are needed.

Note :

The latest SDK software package is provided on Espressif BBS.

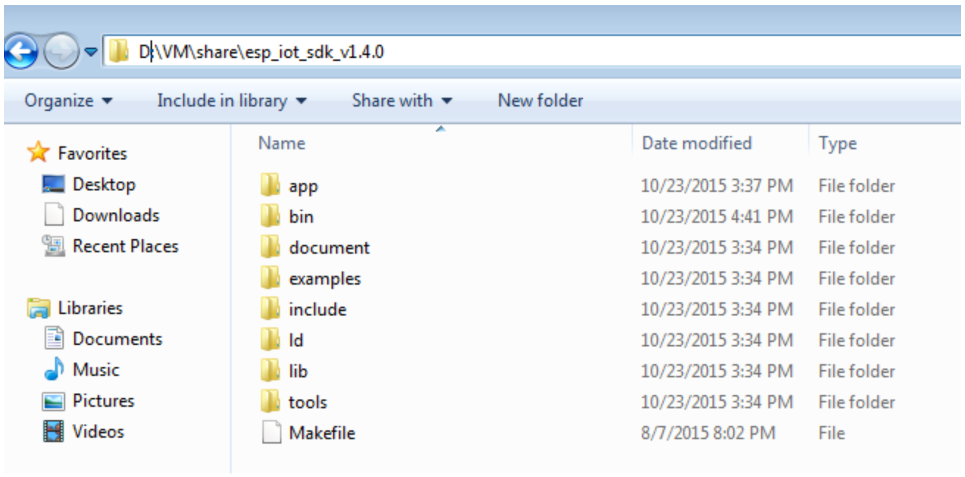
2.1. Prepare Compilation

1. Copy **esp_iot_sdk** folder (including sub-folders and files) to **D:\VM\share**.



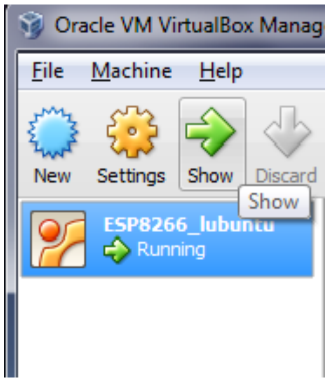
Name	Date modified	Type
app	10/23/2015 3:37 PM	File folder
bin	10/23/2015 4:41 PM	File folder
document	10/23/2015 3:34 PM	File folder
examples	10/23/2015 3:34 PM	File folder
include	10/23/2015 3:34 PM	File folder
ld	10/23/2015 3:34 PM	File folder
lib	10/23/2015 3:34 PM	File folder
tools	10/23/2015 3:34 PM	File folder
Makefile	8/7/2015 8:02 PM	File

2. Copy all the files and sub-folders in **\esp_iot_sdk\examples\IoT_Demo** to **D:\VM\share \esp_iot_sdk\app**.

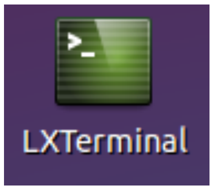


Name	Date modified	Type
app	10/23/2015 3:37 PM	File folder
bin	10/23/2015 4:41 PM	File folder
document	10/23/2015 3:34 PM	File folder
examples	10/23/2015 3:34 PM	File folder
include	10/23/2015 3:34 PM	File folder
ld	10/23/2015 3:34 PM	File folder
lib	10/23/2015 3:34 PM	File folder
tools	10/23/2015 3:34 PM	File folder
Makefile	8/7/2015 8:02 PM	File

3. Run VirtualBox.

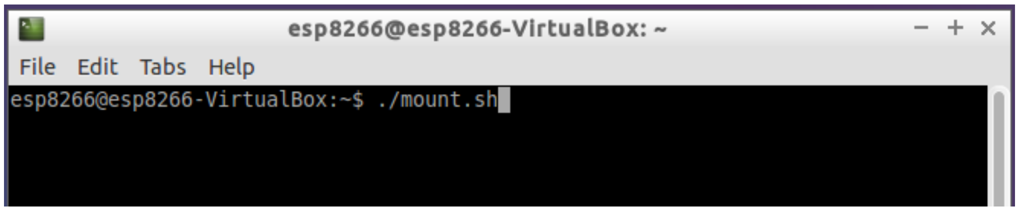


4. Run LXTerminal on the desktop of virtual machine.



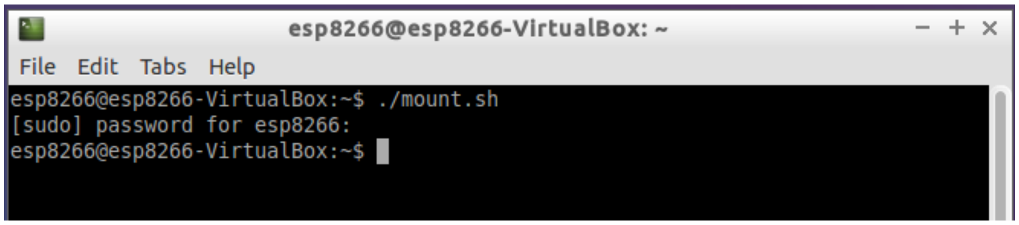
5. Input following command and press Enter to mount the share folder.

```
./mount.sh
```



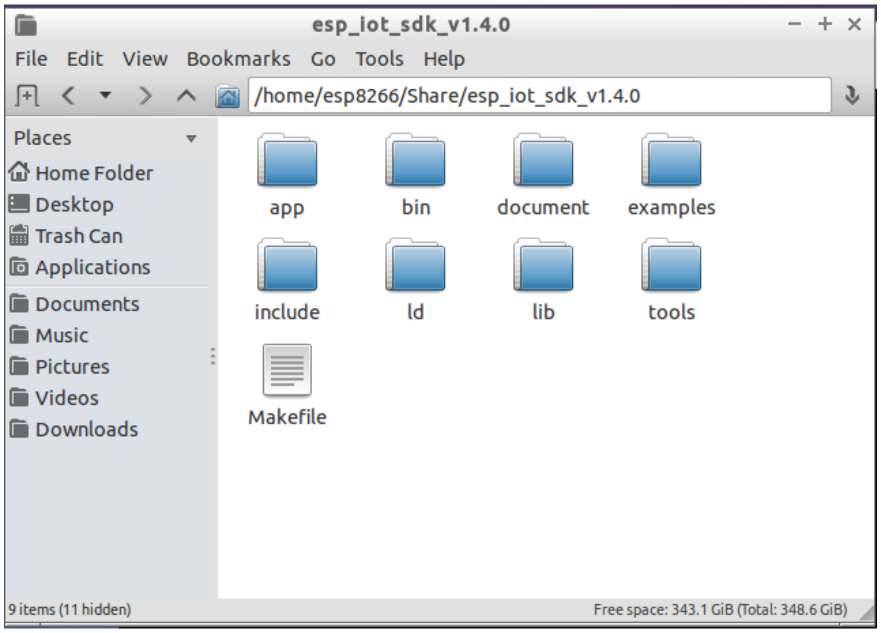
```
esp8266@esp8266-VirtualBox:~$ ./mount.sh
```

6. Input the password **espressif** and press Enter.

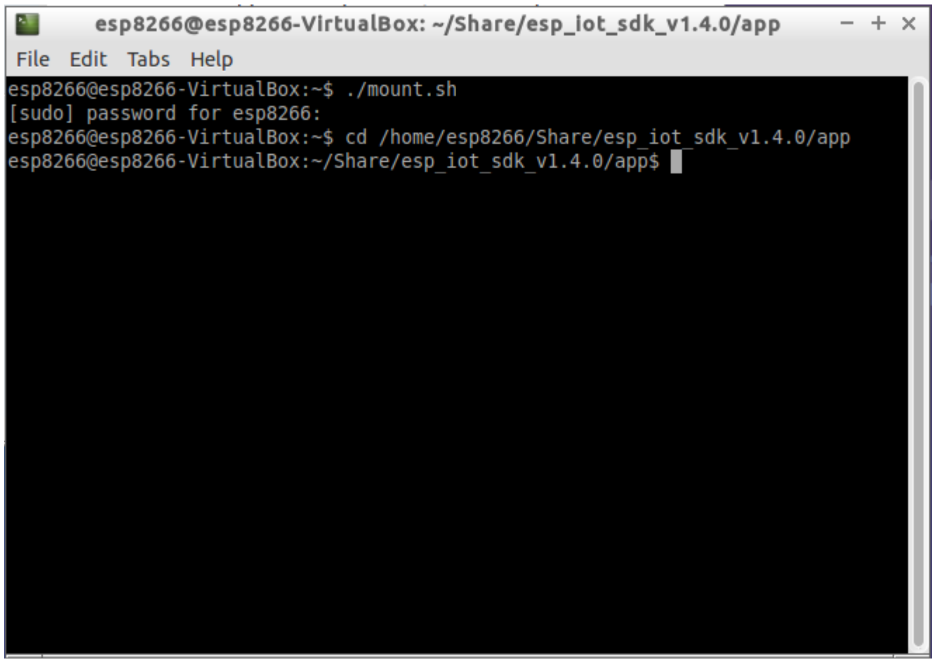


```
esp8266@esp8266-VirtualBox:~$ ./mount.sh
[sudo] password for esp8266:
esp8266@esp8266-VirtualBox:~$
```

7. Check if the shared folder can be find in the VirtualBox.



8. On the terminal, change to **/share/esp_iot_sdk/app t**o start compiling.



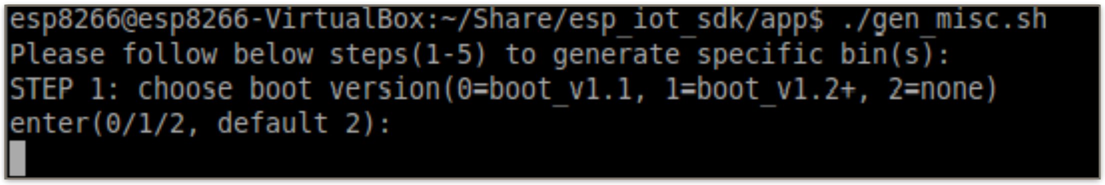
```
esp8266@esp8266-VirtualBox:~$ ./mount.sh
[sudo] password for esp8266:
esp8266@esp8266-VirtualBox:~$ cd /home/esp8266/Share/esp_iot_sdk_v1.4.0/app
esp8266@esp8266-VirtualBox:~/Share/esp_iot_sdk_v1.4.0/app$ 
```

END

2.2. Compile SDK of v0.9.5 and Later Version

For the SDK of release of 0.9.5 and later, the compile process is simplified with a script in the APP folder. Execute the following command

```
./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):
```

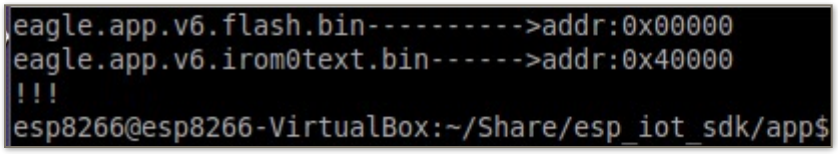
STEP 1: Select boot version	
0	boot_v1.1, old version boot, supports FOTA (firmware upgrade through Wi-Fi)
1	boot_v1.2+, new version boot, always recommend to use the latest boot.bin, supportsFOTA
2	none boot, generate **eagle.flash.bin** and **eagle.irom0text.bin**, does not support FOTA
STEP 2: Select bin to be generated	
0	input 2 in STEP 1, generate **eagle.flash.bin** and **eagle.irom0text.bin**, does not support FOTA
1	input 0 or 1 in STEP 1, generate **user1.bin**, supports FOTA
2	input 0 or 1 in STEP 1, generate **user2.bin**, supports FOTA
STEP 3: SPI flash configuration (SPI speed)	
0	SPI speed 20 MHz, according to your actual SPI flash, set same configuration while downloading by ESP flash download tool.
1	SPI speed 26.7 MHz, according to your actual SPI flash, set same configuration while downloading by ESP flash download tool.
2	SPI speed 40 MHz, according to your actual SPI flash, set same configuration while downloading by ESP flash download tool.
3	SPI speed 80 MHz, 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 512 KB, flash map of program area is 256 KB + 256 KB
2	flash size 1024 KB, flash map of program area is 512 KB + 512 KB when selecting 0 or 1 at STEP 1
3	flash size 2048 KB, only the first 1024 KB is program area, flash map of program area is 512KB + 512KB when selecting 0 or 1 at STEP1

4	flash size 4096 KB, only the first 1024 KB is program area, flash map of program area is 512 KB + 512 KB when selecting 0 or 1 at STEP 1
5	flash size 2048 KB, flash map of program area is 1024 KB + 1024 KB when selecting 0 or 1 at STEP 1 Only supported by sdk_v1.1.0 + boot 1.4 + flash download tool_v1.2 and later version
6	flash size 4096 KB, only the first 2048 KB is program area, flash map of program area is 1024 KB + 1024 KB when selecting 0 or 1 at STEP 1

Note :

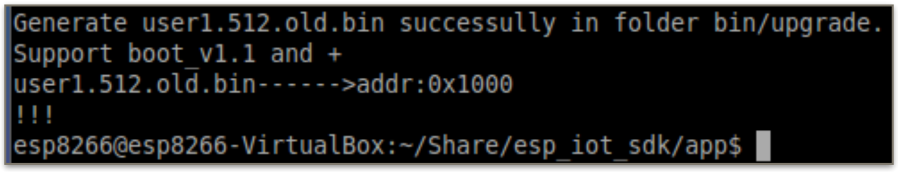
- After compiling user1.bin, execute `make clean` firstly to clean up the temporary files generated by last compilation, then compile **user2.bin**.
- For detailed flash map, refers to Chapter 4.

When compilation 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$
```

2.3. Compile SDK of v0.9.4 and Previous Version

For the SDK of v0.9.4 and previous version, follow the steps to enable cloud update (FOTA), as below:

1. Execute following command, **user1.bin** is generated in **/esp_iot_sdk/bin/upgrade.**

```
./gen_misc_plus.sh 1
```

2. Execute following command to clean up the previous data.

```
make clean
```

3. Execute following command, **user2.bin** is generated in **/esp_iot_sdk/bin/upgrade**.

```
./gen_misc_plus.sh 2
```

Note :

- Refer to document “Firmware update through cloud server" for details about FOTA.
- esp_iot_sdk_v0.7 and previous versions do not support FOTA.
- esp_iot_sdk_v0.8 and later versions support cloud update and are compatible with previous compilation and burning methods.

3. Download Binaries to Flash

3.1. Tools

It is recommended to use minicom for serial port to debug and ESP8266 FLASH TOOL for downloading binaries to flash.

3.1.1. Minicom

When ESP8266 module is initialised, the default baudrate of serial port is 74880 bps with 8 data bits,1 stop bit and no RTC/CTS. There is no 74880 bps option in minicom by default, follow the steps to configure the baudrate as below:

1. Install setserial using “yum” or “apt get” command. For example,

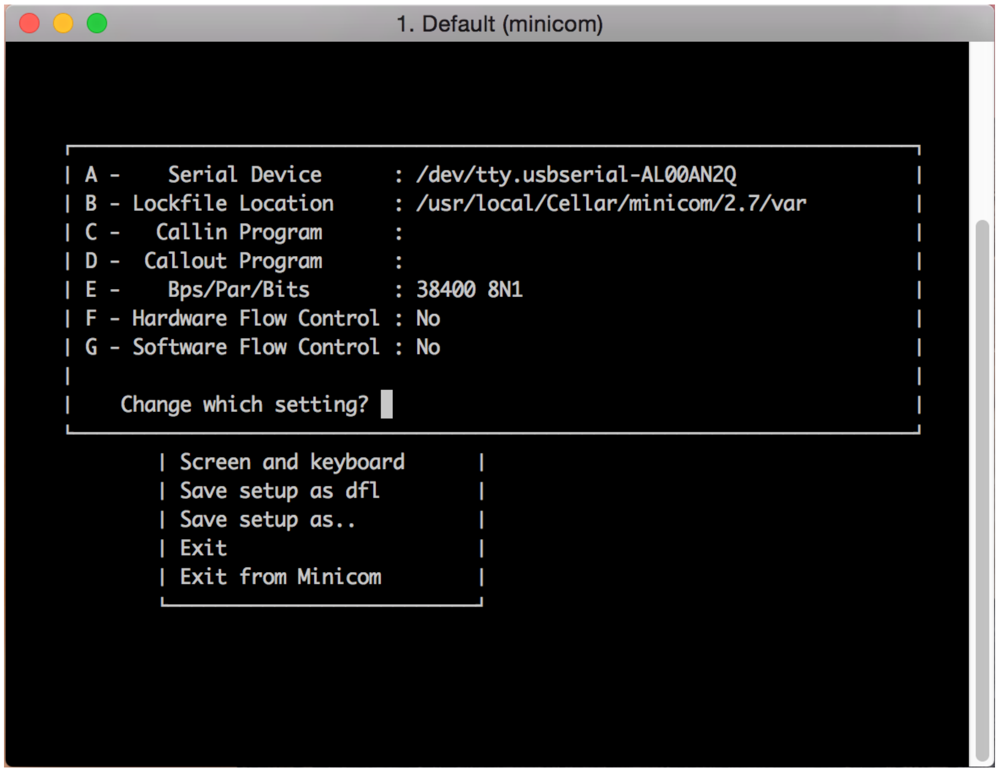
```
apt-get install setserial
```

2. Execute following command to set the divison ratio. `ttyUSBx` refers to the serial port which connects to the ESP8266 board.

```
sudo setserial -v /dev/ttyUSBx spd_cust divisor $((24000000/74880))
```

3. Execute following command to configure the baudrate.

```
minicom-s
```



```
 +-----------------------------------------------------------------------+
 | A -    Serial Device      : /dev/tty.usbserial-AL00AN2Q               |
 | B - Lockfile Location     : /usr/local/Cellar/minicom/2.7/var         |
 | C -   Callin Program      :                                           |
 | D -  Callout Program      :                                           |
 | E -    Bps/Par/Bits       : 38400 8N1                                 |
 | F - Hardware Flow Control : No                                        |
 | G - Software Flow Control : No                                        |
 |                                                                       |
 |    Change which setting?                                              |
 +-----------------------------------------------------------------------+
         | Screen and keyboard      |
         | Save setup as dfl        |
         | Save setup as..          |
         | Exit                     |
         | Exit from Minicom        |
         +--------------------------+
```

3.1.2. ESP8266 FLASH TOOL

Espressif provides ESP8266 FLASH TOOL to download multiple SDK binaries to a SPI flash of ESP8266 board by simple configuration.

> Note :
>
> Download ESP8266 FLASH TOOL from Espressif BBS - Flash Download Tool.
>
> For detailed information on the tool , refer to “ESP8266 FLASH TOOL User Manual”.

3.2. Binaries and Addresses

Once compilation succeeds, it shows the address for the binary to be written to the flash. The binaries and addresses vary by the compilation mode and flash size. This chapter lists the required binaries and responding addresses for flashes of different capacities (512/1024/2048/4096 KB) in both supporting cloud update (FOTA) and no cloud update scenarios.

3.2.1. No Cloud Update

512 KB Flash

Bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users to get Espressif Cloud service
esp_init_data_default.bin	0x7C000	Contains default RF parameters provided in SDK
blank.bin	0x7E000	Contains default system parameters provided in SDK
eagle.flash.bin	0x00000	Compiled from SDK
eagle.irom0text.bin	0x40000	Compiled from SDK

1024 KB Flash

Bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users to get Espressif Cloud service
esp_init_data_default.bin	0xFC000	Contains default RF parameters provided in SDK
blank.bin	0xFE000	Contains default system parameters provided in SDK
eagle.flash.bin	0x00000	Compiled from SDK
eagle.irom0text.bin	0x40000	Compiled from SDK

2048 KB Flash

bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users to get Espressif Cloud service
esp_init_data_default.bin	0x1FC000	Contains default RF parameters provided in SDK
blank.bin	0x1FE000	Contains default system parameters provided in SDK
eagle.flash.bin	0x00000	Compiled from SDK
eagle.irom0text.bin	0x40000	Compiled from SDK

4096 KB Flash

bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users to get Espressif Cloud service
esp_init_data_default.bin	0x3FC000	Contains default RF parameters provided in SDK
blank.bin	0x3FE000	Contains default system parameters provided in SDK
eagle.flash.bin	0x00000	Compiled from SDK
eagle.irom0text.bin	0x40000	Compiled from SDK

3.2.2. Support Cloud Update

Note :

User2.bin does not need to be burned into Flash, it can be downloaded through Wi-Fi (FOTA).

512 KB Flash

bin	Address	Description
master_device_key.bin	0x3E000	Obtained from Espressif Cloud by users to get Espressif Cloud service
esp_init_data_default.bin	0x7C000	Contains default RF parameters provided in SDK
blank.bin	0x7E000	Contains default system parameters provided in SDK
boot.bin	0x00000	Boot loader provided in SDK. Latest version is recommended.
user1.bin	0x01000	Compiled from SDK
user2.bin	0x41000	Compiled from SDK, does not need to be burned into Flash

1024 KB Flash

Bin	Address	Description
master_device_key.bin	0x3E000 (suggest to revise)	Obtained from Espressif Cloud by users to get Espressif Cloud service. Located in user parameter area. The default address in IOT_Demo is 0x3E000, which can be changed by the user. When 1 MB Flash is used, it is suggested to change the address to 0x7E000. Refer to Espressif BBS for details
esp_init_data_default.bin	0xFC000	Contains default RF parameters provided in SDK
blank.bin	0xFE000	Contains default system parameters provided in SDK
boot.bin	0x00000	Boot loader provided in SDK. Latest version is recommended.
user1.bin	0x01000	Compiled from SDK
user2.bin	0x81000	Compiled from SDK, does not need to be burned into Flash

2048 KB Flash

bin	Address	Description
master_device_key.bin	0x3E000 (suggest to revise)	Obtained from Espressif Cloud by users to get Espressif Cloud service. Located in user parameter area. The default address in IOT_Demo is 0x3E000, which can be changed by the user. If selecting 3 in STEP 5 during the compilation, it is suggested to change the address to 0x7E000. If selecting 5 in STEP 5 during the compilation, it is suggested to change the address to 0xFE000 Refer to Espressif BBS for details
esp_init_data_default.bin	0x1FC000	Contains default RF parameters provided in SDK
blank.bin	0x1FE000	Contains default system parameters provided in SDK
boot.bin	0x00000	Boot loader provided in SDK. Latest version is recommended.
user1.bin	0x01000	Compiled from SDK
user2.bin	0x81000	Compiled from SDK, does not need to be burned into Flash

4096 KB Flash

bin	Address	Description
master_device_key.bin	0x3E000 (suggest to revise)	Obtained from Espressif Cloud by users to get Espressif Cloud service. Located in user parameter area. The default address in IOT_Demo is 0x3E000, which can be changed by the user. If selecting 4 in STEP 5 during the compilation, it is suggested to change the address to 0x7E000. If selecting 6 in STEP 5 during the compilation, it is suggested to change the address to 0xFE000 Refer to Espressif BBS for details
esp_init_data_default.bin	0x3FC000	Contains default RF parameters provided in SDK
blank.bin	0x3FE000	Contains default system parameters provided in SDK
boot.bin	0x00000	Boot loader provided in SDK. Latest version is recommended.
user1.bin	0x01000	Compiled from SDK
user2.bin	0x81000	Compiled from SDK, does not need to be burned into Flash

Note :

- System parameter area is the last four sectors of flash, 4 KB per sector, that is the last 16 KB of flash.
- User parameter area depends on user's definition. In IOT_Demo the four sectors starting from 0x3C000 are defined as the user parameter area.
- master_device_key.bin is needed if Espressif Cloud is used, and it is only necessary for the initial download. In IOT _Demo, it is located in the third sector of user parameter area.
- blank.bin should be downloaded to the second last sector in the flash as the initialisation parameter.
- esp_init_data_default.bin stores default RF values and which should be written to the forth sector from the end of flash.
- The flash size applied in IOT_Demo is 512 KB.

4. Flash Map

Different settings of STEP 1 and STEP 5 in compilation (see Chapter 2.2) leads to different flash map.

Note :

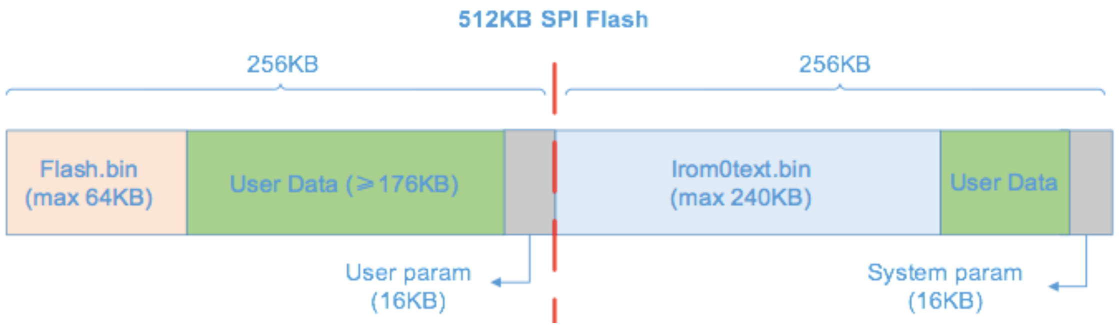
- System param (system parameter area) is always the last 16 KB of flash.
- User param is the user parameter area used by Espressif demo code (IOT_Demo or AT). If users develop their own application, user data can be saved in any flash area available.
- User Data area (green area in figures below) represents 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.

4.1. None boot - No cloud Update

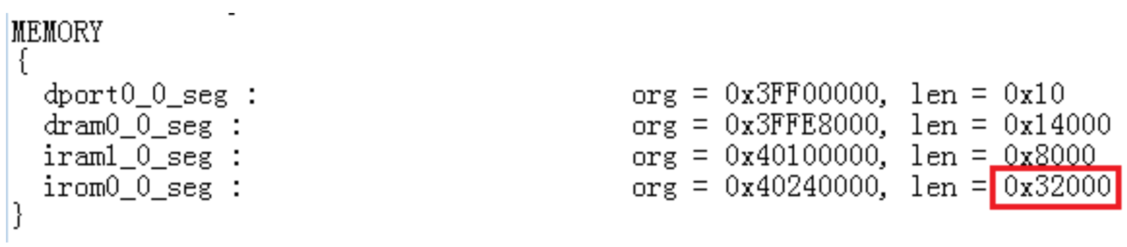
Select 2 none boot in STEP 1 of compilation to generate **eagle.flash.bin** (hereinafter called **flash.bin**) and **eagle.irom0text.bin** (hereinafter called **irom0text.bin**). Then select different flash map in STEP 5 according to the actual SPI flash used.

4.1.1. 512 KB Flash

Select 2 none boot in STEP 1, and then select 0 in STEP 5, the flash map will be as below.



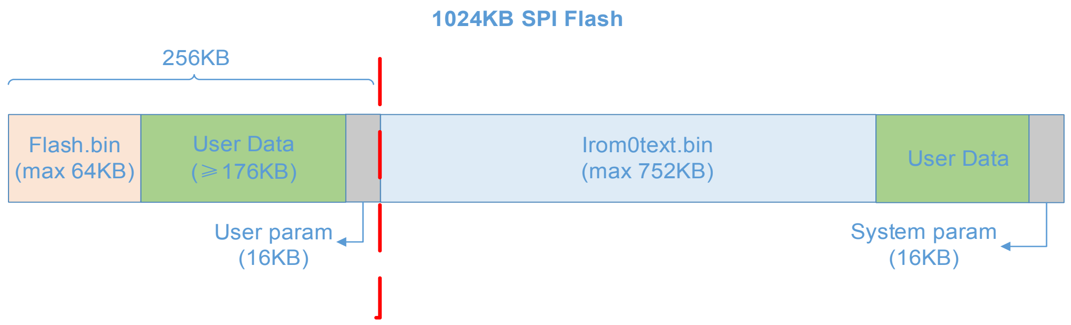
- User Data area: If program area (**flash.bin** and **irom0text.bin**) does not fill up the flash, the remaining area can be used to store user data.
- **irom0text.bin** is defined less than 200 KB by default. For 512 KB flash, user can revise ld file for compilation to make the maximum size to be 256 - 16 = 240 KB.
- In **eagle.app.v6.ld** (\esp_iot_sdk\ld), **len** of **irom0_0_seg** refers to the maximum size of **irom0text.bin**. For 512KB flash, it can be revised to 0x3C000 at most with the maximum size 240 KB.



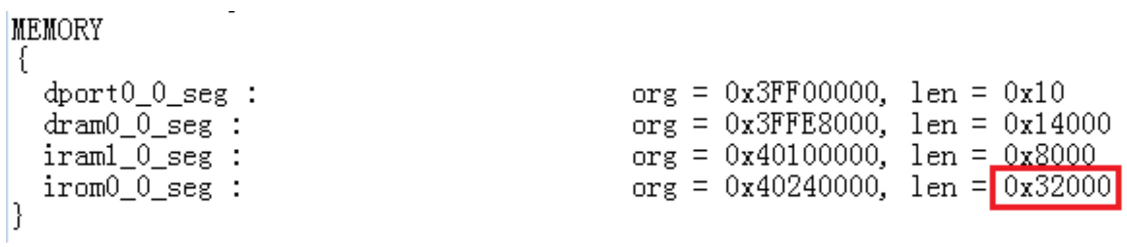
```
MEMORY
{
  dport0_0_seg :                        org = 0x3FF00000, len = 0x10
  dram0_0_seg :                         org = 0x3FFE8000, len = 0x14000
  iram1_0_seg :                         org = 0x40100000, len = 0x8000
  irom0_0_seg :                         org = 0x40240000, len = 0x32000
}
```

4.1.2. 1024 KB Flash

Select 2 none boot in STEP 1, and then select 2 in STEP 5, the flash map will be as below.



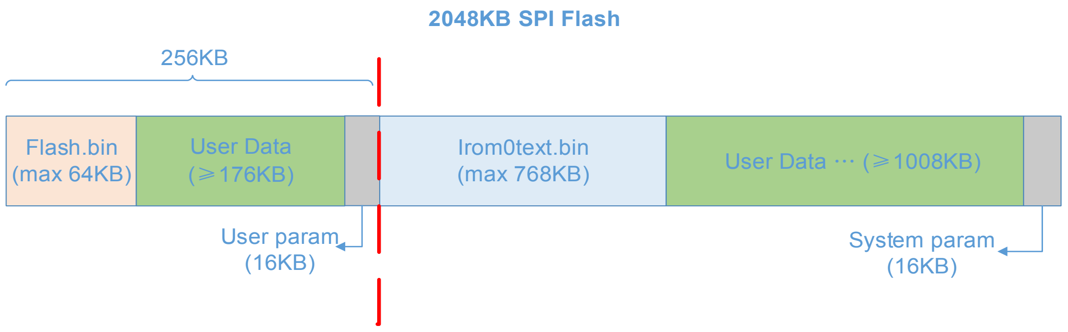
- User Data area: If program area (**flash.bin** and **irom0text.bin**) does not fill up the flash, the remaining area can be used to store user data.
- **irom0text.bin** is defined less than 200 KB by default. For 1024 KB flash, user can revise ld file for compilation to make the maximum size of to be 1024 – 256 - 16 = 752 KB.
- In **eagle.app.v6.ld** (\esp_iot_sdk\ld), **len** of **irom0_0_seg** refers to the maximum size of **irom0text.bin**. For 1024KB flash, it can be revised to 0xBC000 at most with the maximum size 752 KB.



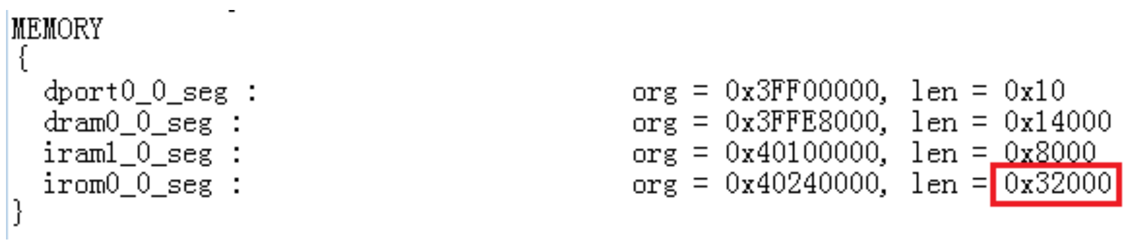
```
MEMORY
{
  dport0_0_seg :                        org = 0x3FF00000, len = 0x10
  dram0_0_seg :                         org = 0x3FFE8000, len = 0x14000
  iram1_0_seg :                         org = 0x40100000, len = 0x8000
  irom0_0_seg :                         org = 0x40240000, len = 0x32000
}
```

4.1.3. 2048 KB Flash

Select none boot in STEP 1, and then select 3 in STEP 5, the flash map will be as below.



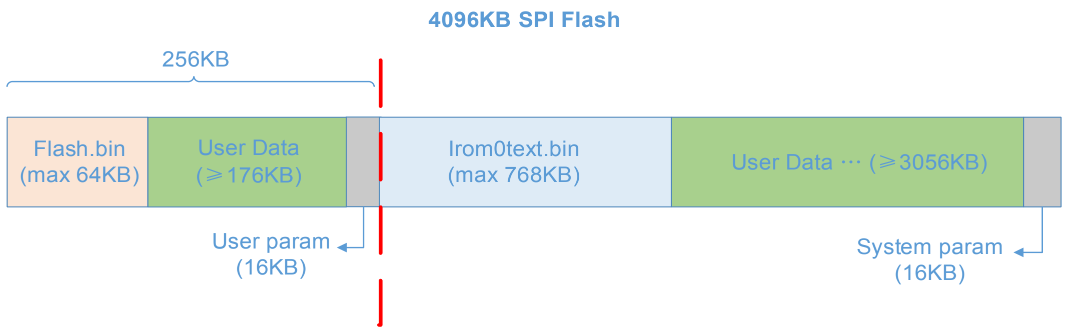
- User Data area: If program area (**flash.bin** and **irom0text.bin**) does not fill up the flash, the remaining area can be used to store user data.
- **irom0text.bin** is defined less than 200 KB by default. Only the first 1024 KB can be program area now, so for 2048 KB flash, user can revise ld file for compilation to make the maximum size to be 1024 - 256 = 768 KB.
- In **eagle.app.v6.ld** (**\esp_iot_sdk\ld**), **len** of i**rom0_0_seg** refers to the maximum size of irom0text.bin. For 2048 KB flash, it can be revised to 0xC0000 at most with the maximum size 768 KB.



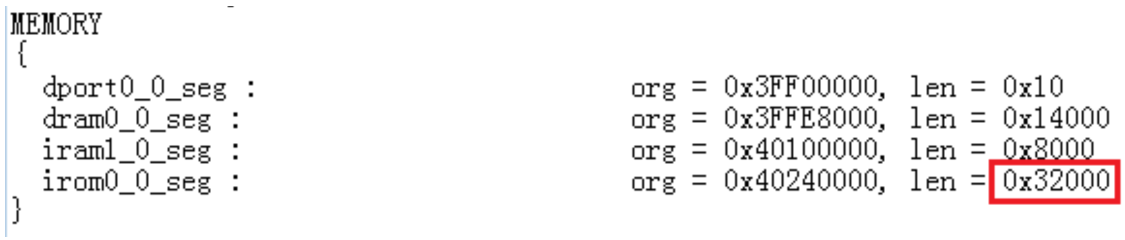
```
MEMORY
{
  dport0_0_seg :                        org = 0x3FF00000, len = 0x10
  dram0_0_seg :                         org = 0x3FFE8000, len = 0x14000
  iram1_0_seg :                         org = 0x40100000, len = 0x8000
  irom0_0_seg :                         org = 0x40240000, len = 0x32000
}
```

4.1.4. 4096 KB Flash

Select 2 none boot in STEP 1, choose 4 in STEP 5, the flash map will be as below.



- User Data area: If program area (**flash.bin** and **irom0text.bin**) does not fill up the flash, the remaining area can be used to store user data.
- **irom0text.bin** is defined less than 200 KB by default. Only the first 1024 KB can be program area now, so for 4096 KB flash, user can revise ld file for compilation to make the maximum size to be 1024 - 256 = 768 KB.
- In **eagle.app.v6.ld** (**\esp_iot_sdk\ld**), **len** of **irom0_0_seg** refers to the maximum size of irom0text.bin. For 2048 KB flash, it can be revised to 0xC0000 at most with the maximum size 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
}
```

4.2. With boot - Support Cloud Update

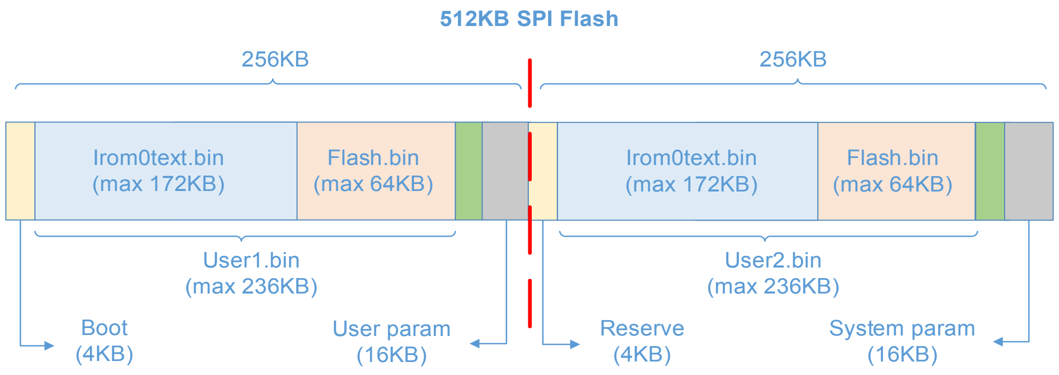
Select 1 boot_v1.2+ in STEP 1 to support FOTA to generate **user1.bin** and **user2.bin**. Then select different flash map in STEP 5 according to the actual SPI flash used.

Note :

- After compiling **user1.bin**, execute `make clean` firstly to clean up the temporary files generated by last compilation, then compile **user2.bin**.
- boot_v1.1 is an old version boot, compilation and downloading are the same as boot_v1.2+, it is recommended to use the latest version of **boot.bin**.
- User Data area (green area in figures below) represents 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.

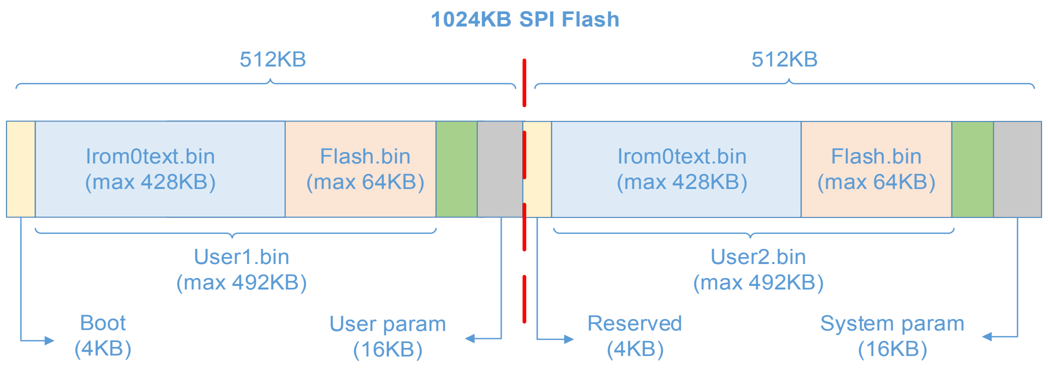
4.2.1. 512 KB Flash

Select 1 boot_v1.2+ in STEP 1, and then select 0 in STEP 5, flash map will be as below.



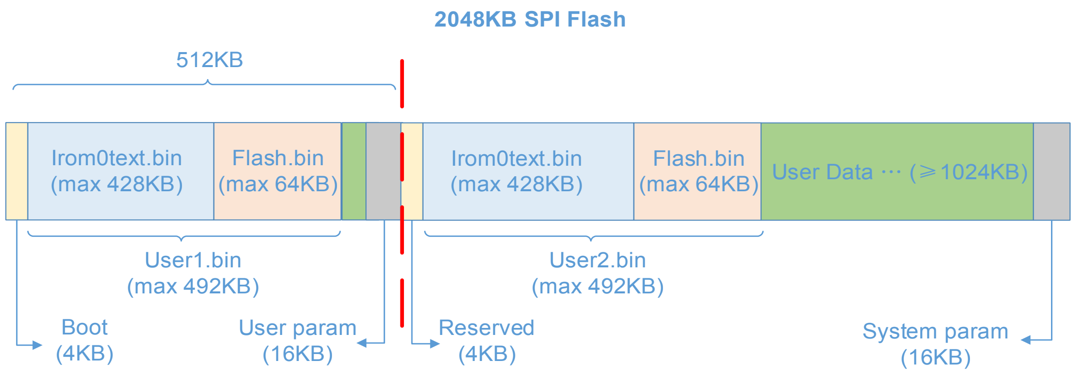
4.2.2. 1024KB Flash

Select 1 boot_v1.2+ in STEP 1, and then select 2 in STEP 5, flash map will be as below.

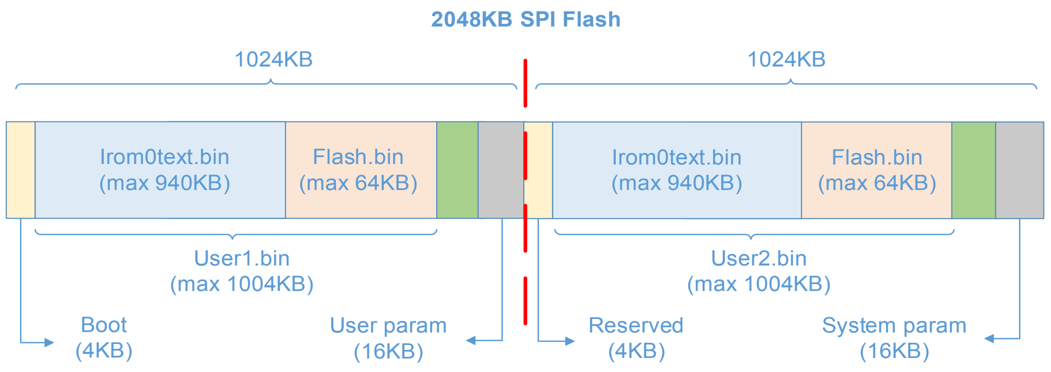


4.2.3. 2048 KB Flash

- Select 1 boot_v1.2+ in STEP 1, and select 3 in STEP 5, only the first 1024KB is the program area (512KB + 512KB), the flash map will be as below.

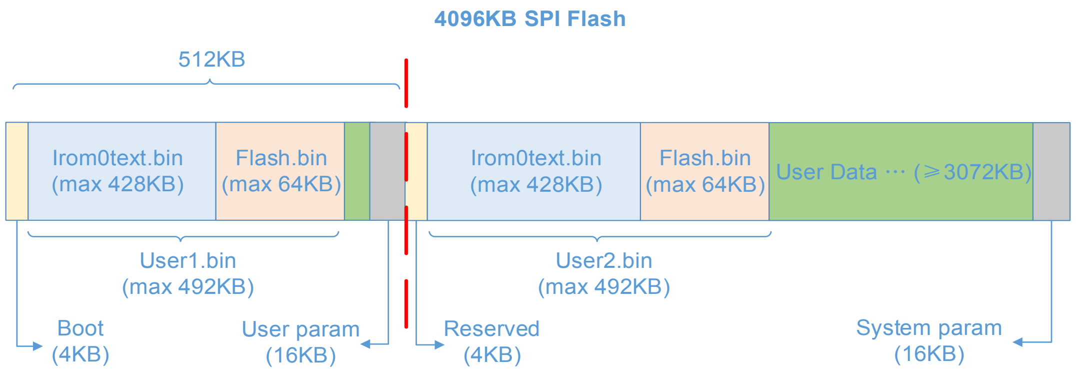


- Select 1 boot_v1.2+ in STEP 1, and then select 5 in STEP 5 (only supported by sdk_v1.1.0 + boot v1.4 + flash download tool v1.2 and later version), the program area is 1024KB + 1024KB, the flash map will be as below.

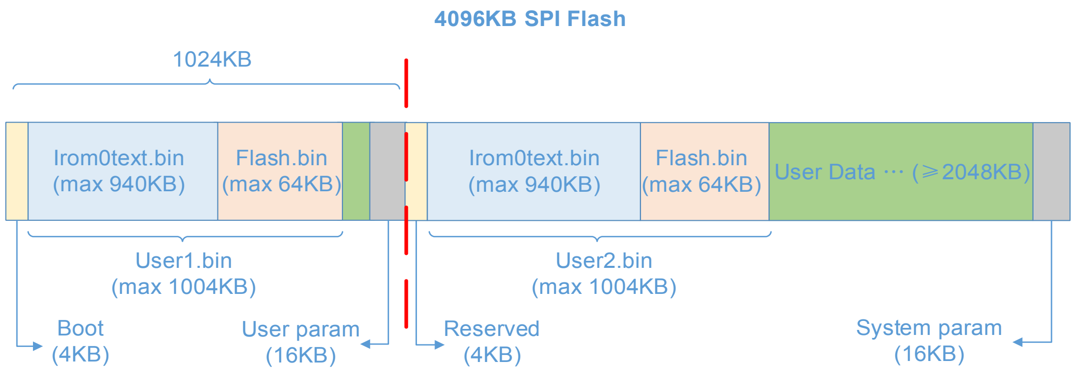


4.2.4. 4096 KB Flash

- Select 1 boot_v1.2+ in STEP 1, and then select 4 in STEP 5, only the first 1024KB is the program area (512KB + 512KB), the flash map will be as below

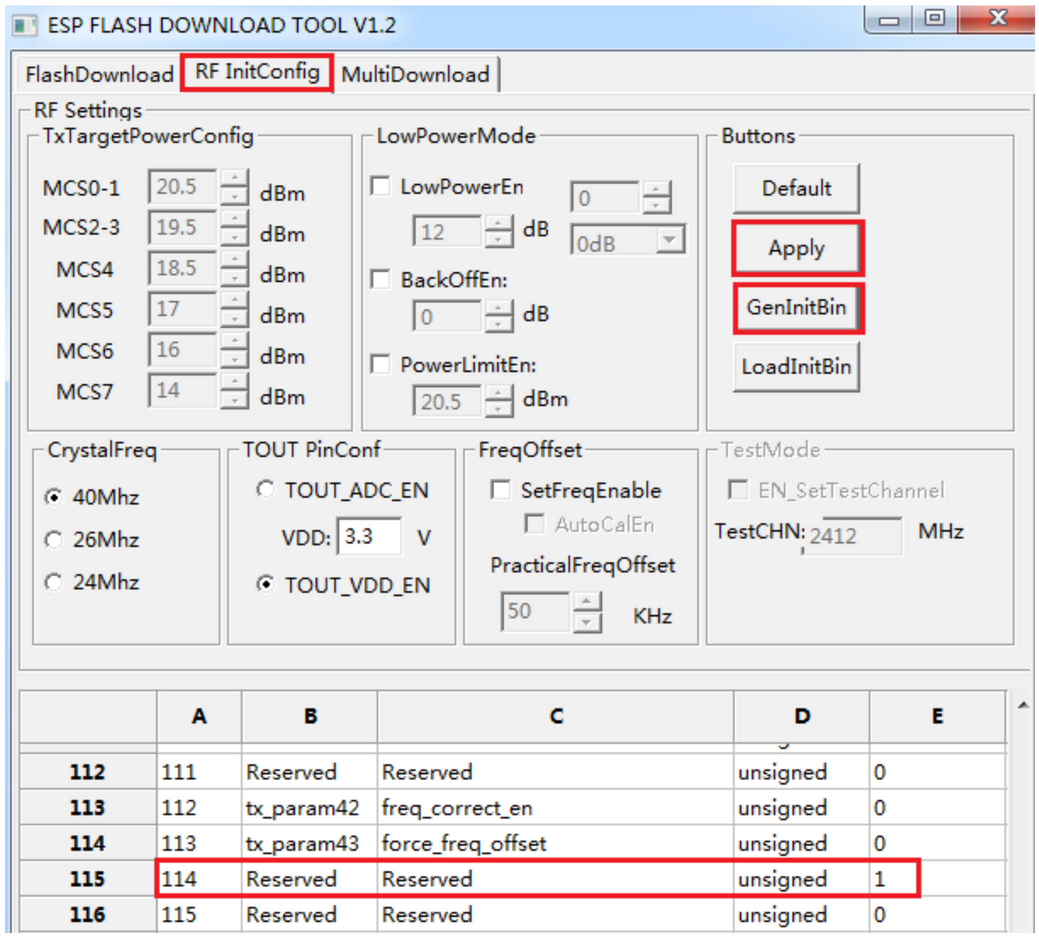


- Select 1 boot_v1.2+ in STEP 1, and then select 6 in STEP 5 (only supported by sdk_v1.1.0 + boot 1.4 + flash download tool v1.2 and later version), the first 2048KB is the program area (1024KB + 1024KB), the flash map will be as below.


4096KB SPI Flash
1024KB
Irom0text.bin (max 940KB)
Flash.bin (max 64KB)
Irom0text.bin (max 940KB)
Flash.bin (max 64KB)
User Data ··· (⩾2048KB)
User1.bin (max 1004KB)
User2.bin (max 1004KB)
Boot (4KB)
User param (16KB)
Reserved (4KB)
System param (16KB)

Appendix

By default, RF calibration will be performed every time the system is initialised which takes some time. From esp_iot_sdk_v1.3.0 and later releases, the user can set RF calibration only when the system is powered on for the first time. The calibration data will be saved in flash and no more calibrations are needed afterwards. Follow the steps to make the configuration as below:



RF Settings

TxTargetPowerConfig

MCS0-1	20.5	dBm
MCS2-3	19.5	dBm
MCS4	18.5	dBm
MCS5	17	dBm
MCS6	16	dBm
MCS7	14	dBm

LowPowerMode: ☐ LowPowerEn 0; 12 dB; 0dB; ☐ BackOffEn: 0 dB; ☐ PowerLimitEn: 20.5 dBm

CrystalFreq: [✓ 40Mhz] 26Mhz 24Mhz

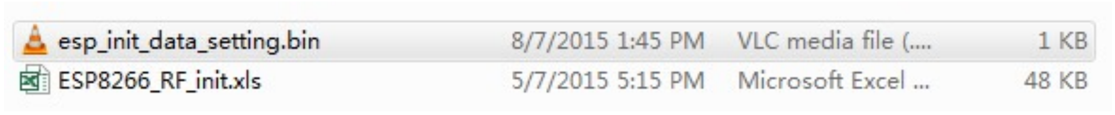
TOUT PinConf: TOUT_ADC_EN; VDD: 3.3 V; [✓ TOUT_VDD_EN]

FreqOffset: ☐ SetFreqEnable; ☐ AutoCalEn; PracticalFreqOffset 50 KHz

TestMode: ☐ EN_SetTestChannel; TestCHN: 2412 MHz

	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

1. Open ESP8266 FLASH TOOL and click **RF InitConfig** tab. The table at the bottom contains RF configuration parameters (0-127 byte) in **esp_init_data_default.bin**. Click byte 114, and change the parameter to 1.
2. Click **Apply** button to confirm the configuration.
3. Click **GenInitBin** to generate **esp_init_data_setting.bin** ,which will be downloaded to Flash instead of **esp_init_data_default.bin**.



esp_init_data_setting.bin	8/7/2015 1:45 PM	VLC media file (....	1 KB
ESP8266_RF_init.xls	5/7/2015 5:15 PM	Microsoft Excel ...	48 KB

END

Espressif System

IOT Team

http://bbs.espressif.com