

Espressif IOT SDK: USER MANUAL

Status	Released
Current version	V0.9.5
Author	Fei Yu
Completion Date	2015.01.23
Reviewer	JG Wu
Completion Date	2015.01.23

☒ CONFIDENTIAL
☐ INTERNAL
☐ PUBLIC

Version Info

Date	Version	Author	Comments / Changes
2013.12.24	0.1	JG Wu	Draft
2014.1.15 ~ 2014.6.19	0.2 ~ 0.6	JG Wu / Han Liu / Fei Yu	Internal modification
2014.7.10	0.7	Fei Yu	Support Cloud Upgrade (OTA)
2014.8.14	0.8	Fei Yu	Add Flash Download Tool
2014.11.07	0.9	Fei Yu	Revised compilation
2015.01.23	0.9.5	Fei Yu	Revised compiling and downloading method

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, NONINFRINGEMENT, 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.

Copyright © 2013 Espressif Systems Inc. All rights reserved.

Table of Contents

Version Info	2
Table of Contents	3
1. Foreword	4
2. Development Tools	5
2.1. Serial Port Tool – SecureCRT	5
2.2. Download Tools - FLASH_DOWNLOAD_TOOLS	5
2.3. NetAssist	7
2.4. Postman	7
2.5. Tomcat	7
3. SDK Software Package	8
3.1. Directory Structure	8
3.2. Compile method	9
3.2.1. After esp_iot_sdk_v0.9.5	10
3.2.2. Before esp_iot_sdk_v0.9.4	11
3.3. Burning into Flash	12
3.3.1. Version that does not support Cloud Update (OTA)	12
3.3.2. Version that support Cloud Update (OTA)	13

1. Foreword

This manual mainly introduces how to use ESP8266-based SDK for Internet of Things, including virtual machine installation, development tool usage, SDK software development kit etc.

CONFIDENTIAL

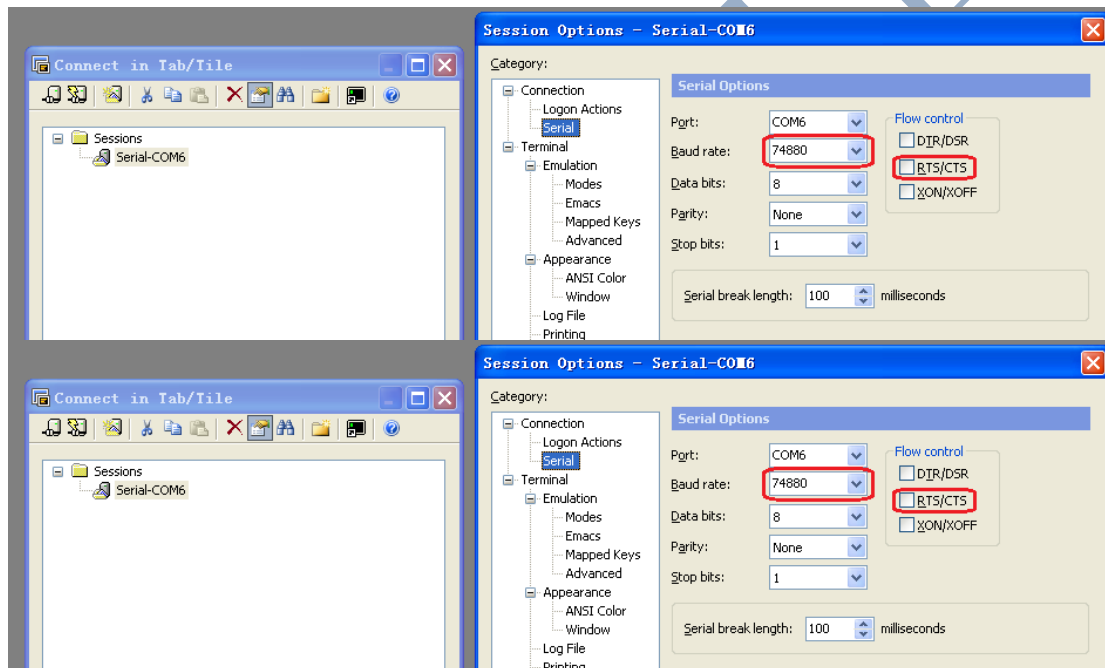
2. Development Tools

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

2.1. Serial Port Tool – SecureCRT

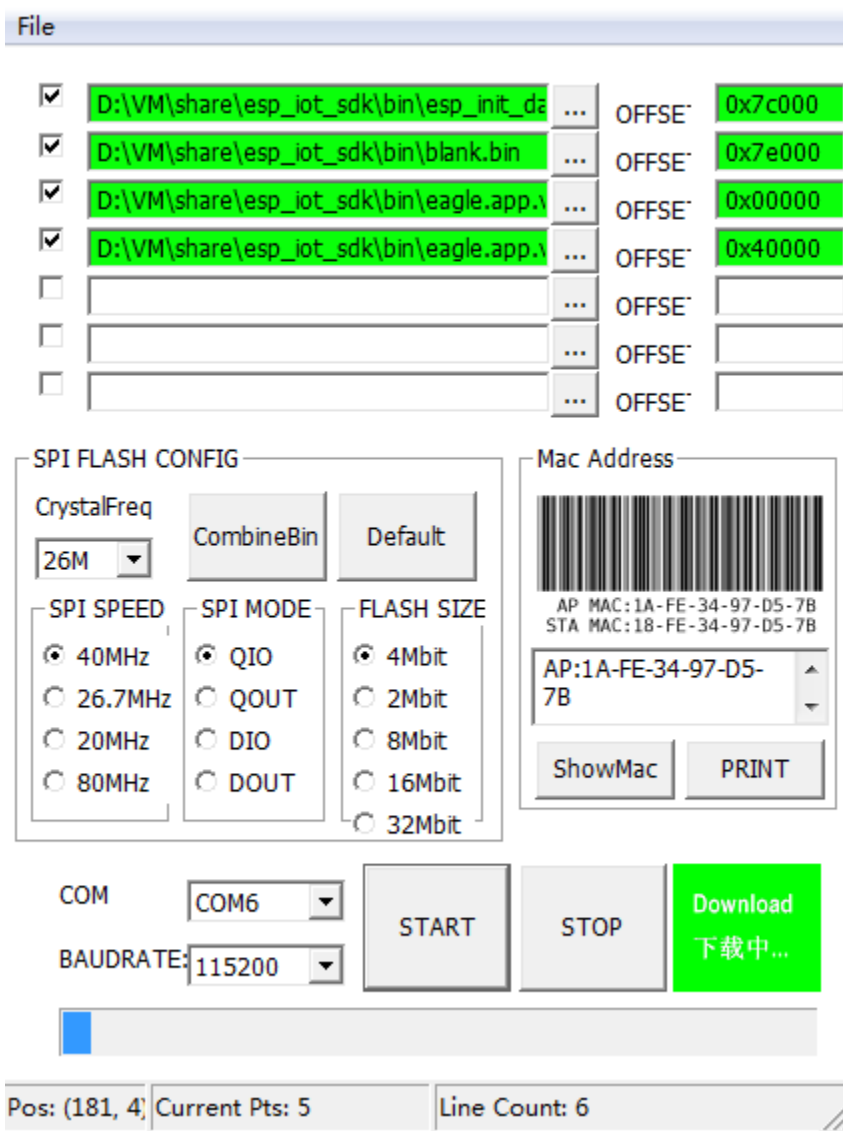
Here use SecureCRT as an example of serial port tool, in fact, you can use any other serial port tool to debug.

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



2.2. Download Tools - FLASH_DOWNLOAD_TOOLS

Espressif provides the tool "ESP_FLASH_DOWNLOAD" for users to burn several bin files altogether at once, and download several compiled *.bin files at a time into the SPI Flash on the ESP8266 motherboard.



“ESP_FLASH_DOWNLOAD” introduction:

- (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**, then it will enter download mode. Steps are as follows:

- (1) See the red boxes in the picture above, select the bin file to be burned ->fill in the path ->check burning options.
- (2) Set COM port and baud rate.
- (3) Click "START" to start downloading.
- (4) 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.

Note: please disconnect the power when setting the jumper.

2.3. NetAssist

NetAssist is used to test TCP and UDP.

2.4. Postman

Chrome plug-in is used to test REST-structured web service.

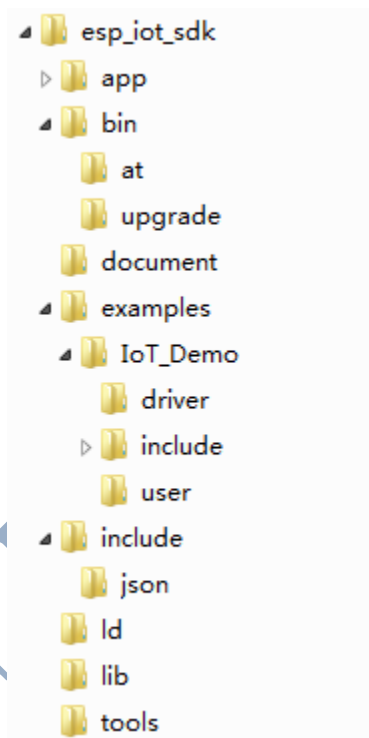
2.5. Tomcat

Web application server, used to store updates.

3. SDK Software Package

3.1. Directory Structure

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:



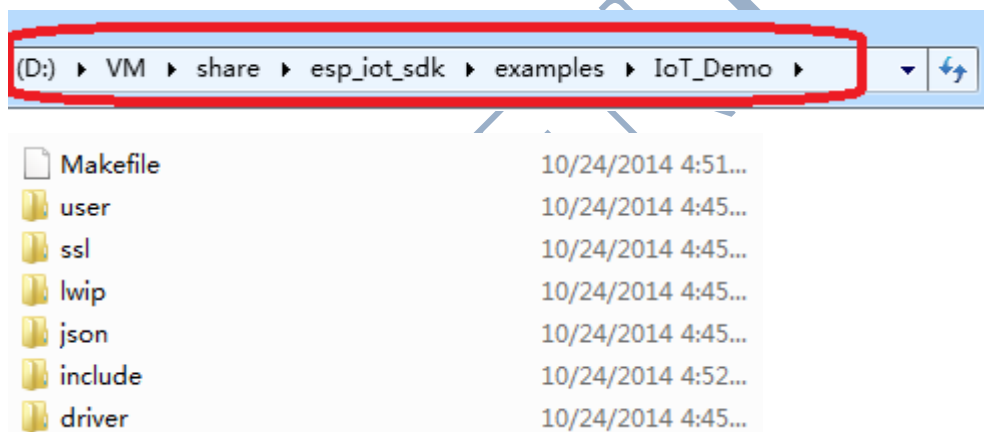
Detailed description:

1. The "app" folder is the main working folder, we need to copy source codes to this folder to compile.
2. "bin" folder stores the bin files downloaded into the Flash, "at" folder stores the bin files that support AT+ instructions and "upgrade" folder stores the bin files that support cloud update.
3. "examples" folder stores SDK examples, we need to copy the source code here (all files in the IoT_Demo folder) to "app" folder;
4. "include" folder stores the header files pre-installed in the SDK, which may include relevant API functions and other definitions. Users can

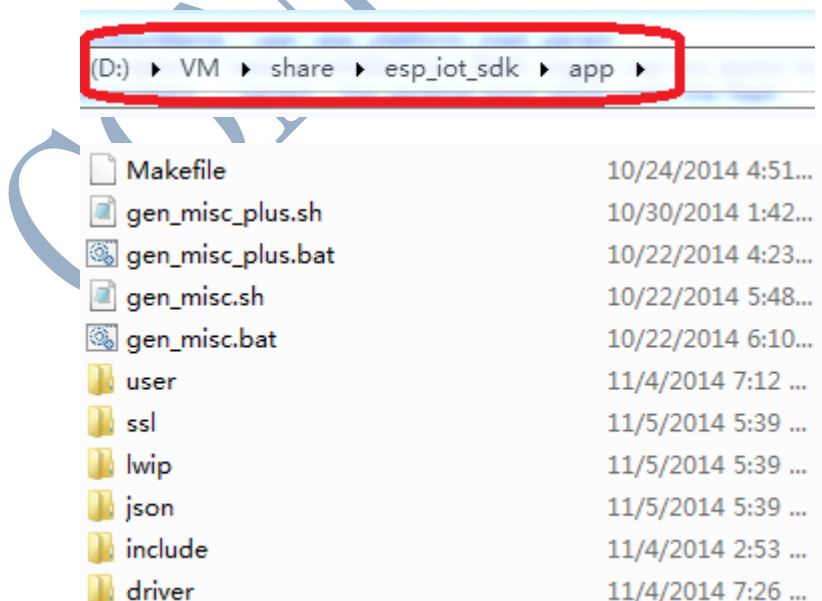
- use them directly and do not need to change anything;
- "ld" folder stores the files needed for SDK software link. Users can use them directly and do not need to change anything;
 - "lib" folder stores the library files needed for SDK compilation;"tools" folder stores the tools needed for generating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

3.2. Compile method

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



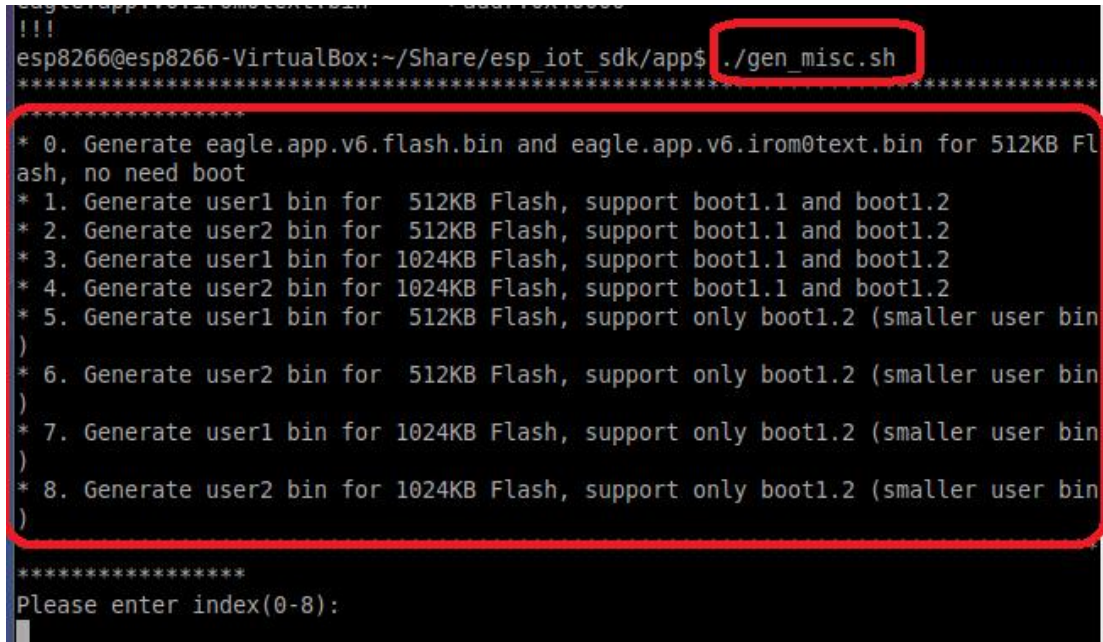
Copy all files in the picture above to `esp_iot_sdk\app` to compile.



3.2.1. After esp_iot_sdk_v0.9.5

Compile : `./gen_misc.sh`

Then follow the tips and steps.



```

!!!
esp8266@esp8266-VirtualBox:~/Share/esp_iot_sdk/app$ ./gen_misc.sh
*****
* 0. Generate eagle.app.v6.flash.bin and eagle.app.v6.irom0text.bin for 512KB Flash, no need boot
* 1. Generate user1 bin for 512KB Flash, support boot1.1 and boot1.2
* 2. Generate user2 bin for 512KB Flash, support boot1.1 and boot1.2
* 3. Generate user1 bin for 1024KB Flash, support boot1.1 and boot1.2
* 4. Generate user2 bin for 1024KB Flash, support boot1.1 and boot1.2
* 5. Generate user1 bin for 512KB Flash, support only boot1.2 (smaller user bin)
* 6. Generate user2 bin for 512KB Flash, support only boot1.2 (smaller user bin)
* 7. Generate user1 bin for 1024KB Flash, support only boot1.2 (smaller user bin)
* 8. Generate user2 bin for 1024KB Flash, support only boot1.2 (smaller user bin)
*****
Please enter index(0-8):

```

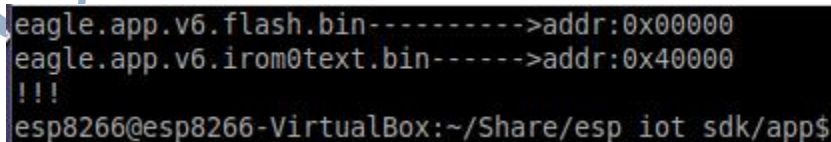
Enter 0, generate bin files do **not** support upgrading through WIFI (FOTA): eagle.app.v6.flash.bin and eagle.app.v6.irom0text.bin.

Enter 1 ~ 8, according to flash size, generate user1.bin and user2.bin.

Notice,

- 1) Enter different parameter, it may support different version of boot.bin;
- 2) Please compile user1.bin and user2.bin with the same configuration.
- 3) Compile succeed, it shows the address that bins need to be download.

For example,

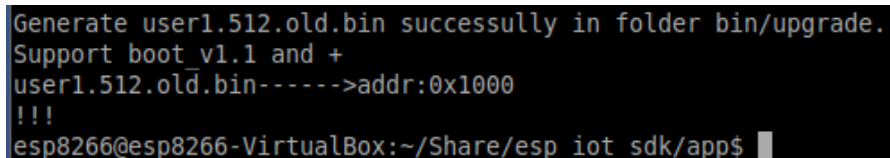


```

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

```

Example,

Run “./gen_misc.sh”, enter 1, to generate user1.bin for 512KB Flash. If succeed, it will generate “user1.512.old.bin” at “\esp_iot_sdk\bin\upgrade”;

Then run “./gen_misc.sh”, enter 2, to generate user2.bin with the same configuration. If succeed, it will generate “user2.512.old.bin” at “\esp_iot_sdk\bin\upgrade”

3.2.2. Before esp_iot_sdk_v0.9.4

FW do 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.

3.3. Burning into Flash

According to usage and compiling method, choose one way to burn bin files into flash.

3.3.1. Version that does not support Cloud Update (OTA)

Note:

- (1) It is not necessary to burn blank.bin every time and it is only necessary for SDK update or clearing of WIFI configuration
- (2) It is not necessary to burn master_device_key.bin every time and it is only necessary for initial write-in and revision of master_device_key
- (3) Normally, it is only necessary to burn these 2 bins: **eagle.app.v6.flash.bin** and **eagle.app.v6.irom0text.bin**.

3.3.1.1. 512KB Flash

- blank.bin, provided in SDK; to be burned to 0x7E000
- eagle.app.v6.flash.bin, compiled by the steps said above; to be burned to 0x0000
- master_device_key.bin, obtained from Espressif Cloud Server; to be burned to 0x3E000
- eagle.app.v6.irom0text.bin, compiled by the steps said above; to be burned to 0x40000
- esp_init_data_default.bin, provided by Espressif; stores default parameter values and to be burned to 0x7C000.

3.3.1.2. 1MB Flash or larger

- blank.bin, provided in SDK; to be burned to 0xFE000

- eagle.app.v6.flash.bin, compiled by the steps above; to be burned to 0x00000
- master_device_key.bin, obtained from Espressif Cloud Server; to be burned to 0x7E000
- eagle.app.v6.irom0text.bin, compiled by the steps said above; to be burned to 0x80000
- esp_init_data_default.bin, provided by Espressif; stores default parameter values and to be burned to 0xFC000.

3.3.2. Version that support Cloud Update (OTA)

Note:

For future updates, please upload user1.bin and user2.bin to the server and the server will send update information to users. If users choose to update, then the device will select and download user1.bin or user2.bin, whichever is necessary for cloud update.

3.3.2.1. 512KB Flash

- blank.bin; provided in SDK and to be burned to both 0x3E000 and 0x7E000;
- esp_init_data_default.bin, provided by Espressif; stores default parameter values and to be burned to 0x7C000.
- boot.bin, provided in SDK and to be burned to 0x00000;
- user1.bin, compiled by the steps said above and to be burned to 0x01000;
- user2.bin, compiled by the steps said above and to be burned to 0x41000;
- master_device_key.bin, applied for through Espressif server and to be burned to 0x3E000;

3.3.2.2. 1MB Flash or larger

- blank.bin; provided in SDK and to be burned to both 0x7E000 and 0xFE000;
- esp_init_data_default.bin, provided by Espressif; stores default parameter values and to be burned to 0xFC000.
- boot.bin, provided in SDK and to be burned to 0x00000;
- user1.bin, compiled by the steps said above and to be burned to 0x01000;
- user2.bin, compiled by the steps said above and to be burned to 0x81000;
- master_device_key.bin, applied for through Espressif server and to be burned to 0x7E000;

CONFIDENTIAL