Espressif IOT SDK User Manual

Version 1.0.1

Espressif Systems IOT Team

Disclaimer and Copyright Notice

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

Table of Contents

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

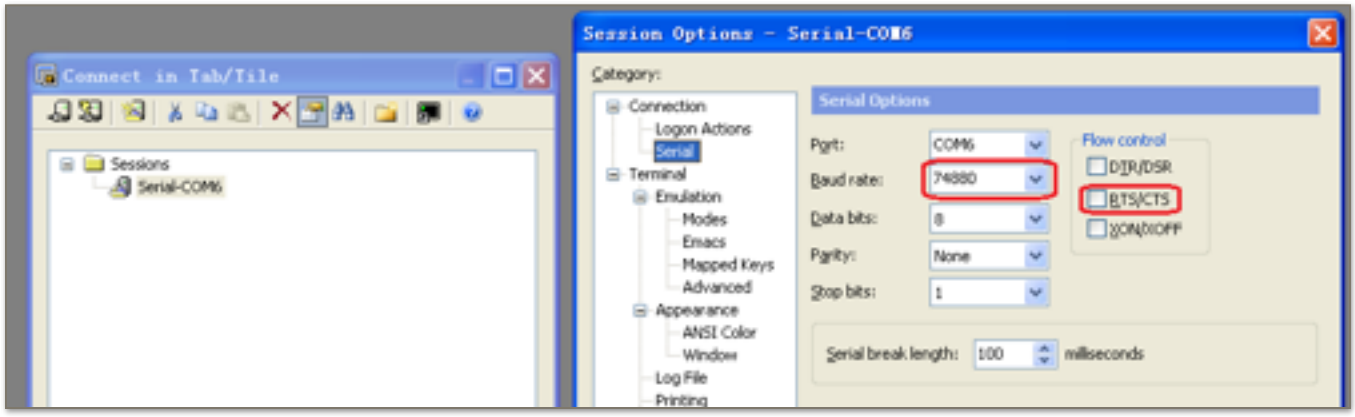
2. Development Tools

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

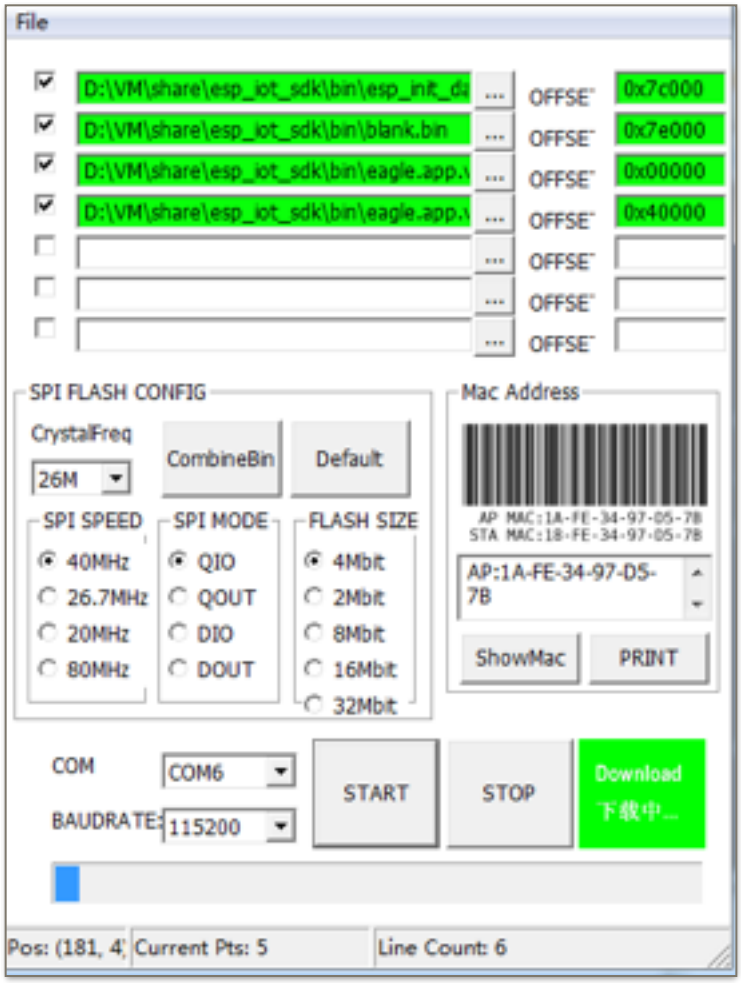
Using **ESP_FLASH_DOWNLOAD**:

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.

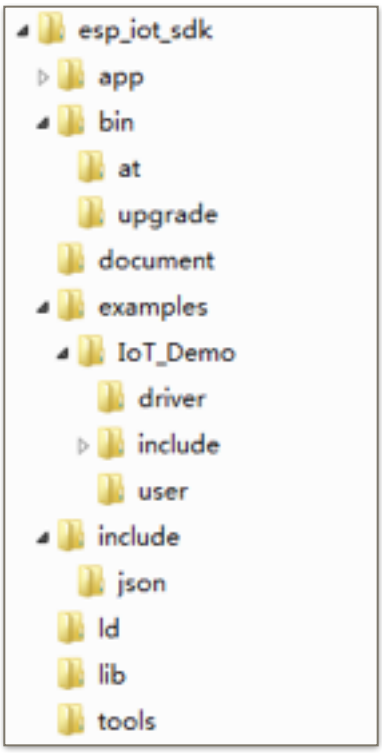
PS: Please disconnect the power when setting the jumper.



3. SDK Software Package

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:



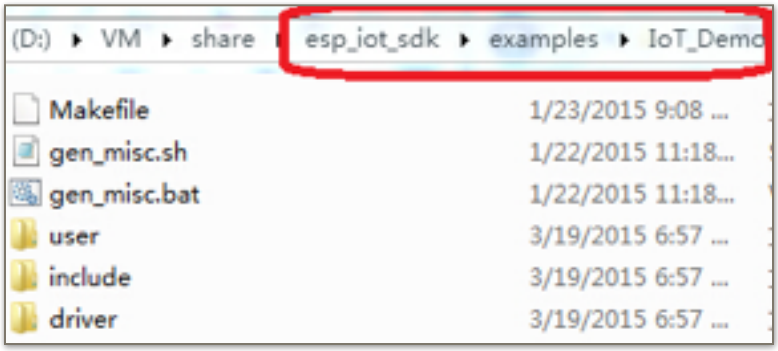
Detailed description:

- 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;

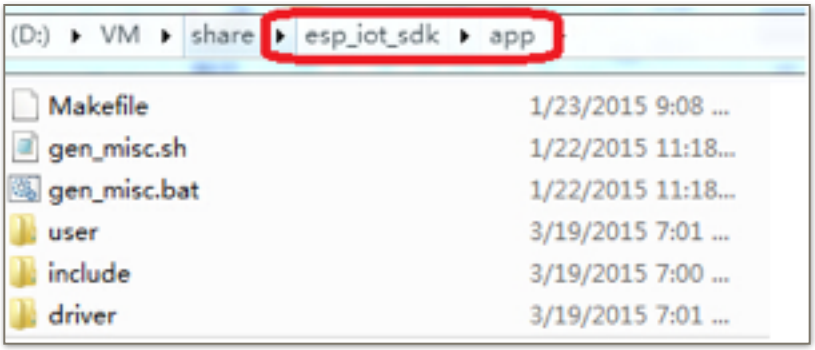
- "ld" folder stores the files needed for SDK software link. Users can use them directly and do not need to change anything;
- "lib" folder stores the library files needed for SDK compilation; "tools" folder stores the tools needed for generating bin files. Users can use them directly and do not need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

2. 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.



2.1. 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.

Notice,

- Enter different parameter, it may support different version of `boot.bin`; we recommend using the latest boot.
- 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$
```

2.2. 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.

3. Writing Image Into Flash

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

3.1. Without Support For Cloud Update (OTA)

Note:

- 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.
- It is usually only necessary to burn these 2 bins: eagle.app.v6.flash.bin and eagle.app.v6.irom0text.bin. You can also burn blank.bin as initialization.

1. 512KB Flash

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

2. 1MB Flash or larger

- blank.bin: provided in SDK; to be written to the last sector but one in Flash，as 0xFE000 in 1MB Flash，0x3FE000 in 4MB Flash
- eagle.app.v6.flash.bin: compiled by the steps above; to be written to 0x00000
- master_device_key.bin: obtained from Espressif Cloud Server; to be written to 0x7E000
- eagle.app.v6.irom0text.bin: compiled by the steps said above; to be written to 0x80000
- esp_init_data_default.bin: provided by Espressif; stores default parameter values and to be written to the forth sector from the end of Flash，as 0xFC000 in 1MB Flash，0x3FC000 in 4MB Flash.

3.2. Version that support Cloud Update (OTA)

Note:

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

1. 512KB Flash

- blank.bin: provided in SDK and to be written to both 0x3E000 and 0x7E000;

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

2. 1MB Flash or larger

- blank.bin: provided in SDK and to be written to the last sector but one in Flash，as 0xFE000 in 1MB Flash，0x3FE000 in 4MB Flash;
- esp_init_data_default.bin: provided by Espressif; stores default parameters and be written to the forth sector from the end of Flash，as 0xFC000 in 1MB Flash，0x3FC000 in 4MB Flash.
- boot.bin: provided in SDK and to be written to 0x00000;
- user1.bin: compiled by the steps said above and to be written to 0x01000;
- user2.bin: compiled by the steps said above and to be written to 0x81000;
- master_device_key.bin: applied for through Espressif server and to be written to 0x7E000;