ESP8266 SDK
Getting Started Guide

Version 2.1

About This Guide

This document takes ESP-LAUNCHER and ESP-WROOM-02 as examples to introduce how to use ESP8266 SDK. The contents include preparations before compilation, SDK compilation and firmware download.

The document is structured as follows.

Chapter	Title	Content
Chapter 1	Overview	Introduction to the overall procedure of using the SDK and familiarization with the HDK, FW and toolkit for the ESP8266.
Chapter 2	Preparing the Hardware	Introduction to hardware configuration and setup for programming. Illustrated with two examples, ESP-LAUNCHER and ESP-WROOM-02.
Chapter 3	Preparing the Software	Introduction to the non-OS SDK and RTOS SDK. Information on the tools for compiling the SDK and downloading the firmware.
Chapter 4	Flash Map	Address and layout specifications for downloading the firmware to flash memory. Introduction to the FOTA and non-FOTA firmware.
Chapter 5	Compiling the SDK	Introduction to compiling the SDK with compiling tools.
Chapter 6	Downloading the Firmware	Introduction to downloading firmware with download tools.
Appendix I	Configuring ISSI & MXIC Flash QIO Mode	Introduction to configuring ISSI & MXIC Flash QIO mode.

Release Notes

Date	Version	Release Notes
2016.04	V2.0	First release.
2016.07	V2.1	Added MXIC Flash QIO mode; Modified 112 byte default value to 0

Related Documents

Please download related documents at:

Espressif's official website: *http://www.espressif.com/support/download/documents*

Category	Documents
HDK Guides	***ESP8266 System Description***
	ESP-WROOM-02 Datasheet
SDK Guides	***ESP8266 FOTA Upgrade Guide***
	ESP8266 Non-OS SDK AT Instruction Set

Table of Contents

1. Overview

1.1. Procedure Overview

Figure 1-1 shows the overall procedure of SDK compilation.

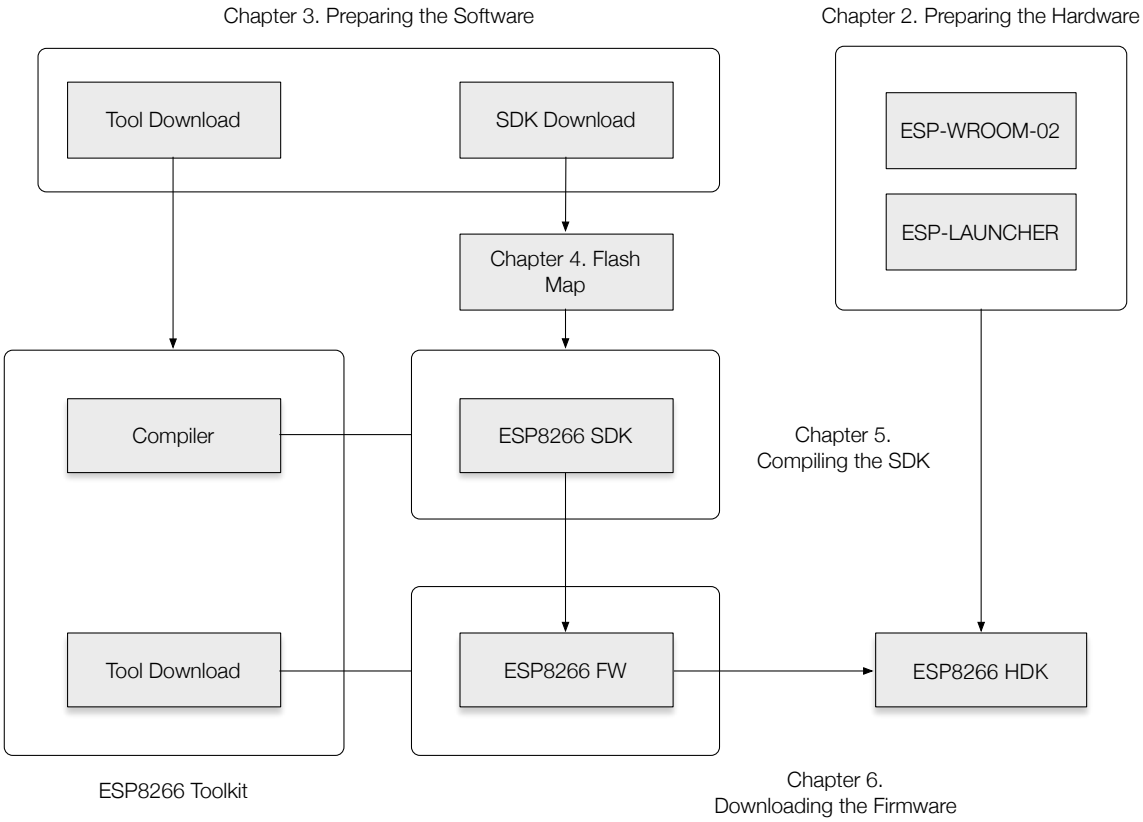


Figure 1-1 Procedure Overview

1.2. ESP8266 HDK

ESP8266 HDK (Hardware Development Kits) includes the chip - ESP8266EX, the module - ESP-WROOM-02 and the development board - ESP-LAUNCHER. Users can download the pre-compiled firmware using ESP-WROOM-02 or ESP-LAUNCHER.

> 📖 ***Notes:***
>
> - *If you are using other development boards or modules that integrate ESP8266EX, please use the development firmware provided by the corresponding manufacturers.*
> - *If you would like to purchase ESP-WROOM-02 or ESP-LAUNCHER, please visit Espressif's official online store at: https://espressif.taobao.com,*

1.3. ESP8266 SDK

ESP8266 SDK (Software Development Kits) is an IOT application development platform developed by Espressif for developers that includes basic platform and upper application development examples, such as Smart Light and Smart Plug.

Depending on whether it is based on an operating system (OS), SDK can be categorized into two versions: Non-OS SDK and RTOS SDK.

1.3.1. Non-OS SDK

Non-OS SDK is not based on an operating system. It supports the compilation of IOT_Demo and AT commands. Non-OS SDK uses timers and callbacks as the main way to perform various functions - nested events, functions triggered by certain conditions. Non-OS SDK uses the espconn network interface; users need to develop their software according to the usage rules of the espconn interface.

1.3.2. RTOS SDK

RTOS SDK is based on FreeRTOS and open source on Github.

- FreeRTOS SDK is based on FreeRTOS , a multi-tasking OS. You can use standard interfaces to realize resource management, recycling operations, execution delay, inter-task messaging and synchronization, and other task-oriented process design approaches. For the specifics of interface methods, please refer to the official website of FreeRTOS or USING THE FreeRTOS REAL TIME KERNEL--A Practical Guide
- The network operation interface in RTOS SDK is the standard lwIP API. RTOS SDK provides a package which enables BSD Socket API interface. Users can directly use the socket API to develop software applications; and port other applications from other platforms using socket API to ESP8266, effectively reducing the learning costs arising from platform switch.
- RTOS SDK introduces cJSON library whose functions make it easier to parse JSON packets.
- RTOS is compatible with non-OS SDK in Wi-Fi interfaces, Smart Config interfaces, Sniffer related interfaces, system interfaces, timer interface, FOTA interfaces and peripheral driver interfaces, but does not support the AT implementation.

1.4. ESP8266 FW

ESP8266 FW (Firmware) has been provided in binary format (.BIN) files that can be downloaded directly to the HDK. Users can choose between FOTA (Firmware Over-The-Air) and Non-FOTA binaries. For detailed information, please refer to Table 1-1.

Table 1-1. ESP8266 FW

Binaries	Compulsory or optional	Description	Non-FOTA	FOTA
master_device_key.bin	Optional	Users can apply for it from Espressif Cloud to get Espressif Cloud service.	☑	☑
esp_init_data_default.bin	Compulsory	Default system parameters, provided in SDK.	☑	☑
blank.bin	Compulsory	Default system parameters, provided in SDK.	☑	☑
eagle.flash.bin	Compulsory	Main program, compiled from SDK.	☑	✗
eagle.irom0text.bin	Compulsory	Main program, compiled from SDK.	☑	✗
user1.bin	Compulsory for first-time usage.	Main program, compiled from SDK.	✗	☑
user2.bin	Used in upgrade.	Main program, compiled from SDK.	✗	☑

📖 ***Notes:***

- *For the contents of SDK, please refer to **Chapter 3 Preparing the Software**.*
- *For SDK compilation, please refer to **Chapter 5 Compiling the SDK**.*
- *For the addresses of binaries in the flash, please refer to **Chapter 4 Flash Map**.*

1.5. ESP8266 Toolkit

1.5.1. Compiler

Linux OS is required to compile the ESP8266 SDK. If you are using Windows OS, we recommend VirtualBox as the virtual machine for ESP8266. In order to simplify the compilation procedure, we have installed the compiling tools on the virtual machine. Users can directly compile ESP8266 SDK by importing ESP8266 compiler (OVA image) into the virtual machine.

1.5.2. Firmware Download Tool

ESP8266 DOWNLOAD TOOL is the official firmware download tool developed by Espressif. Users can download multiple binaries to the SPI Flash of ESP8266 mother board (ESP-LAUNCHER or ESP-WROOM-02) at one time according to the actual compilation mode and flash size.

1.5.3. Serial Port Debug Tool

The serial port debug tool can be used to directly communicate with the ESP8266 module over a standard RS-232 port. For PCs that do not have a physical serial port, a virtual comm port (USB-to-serial converter) can be used.

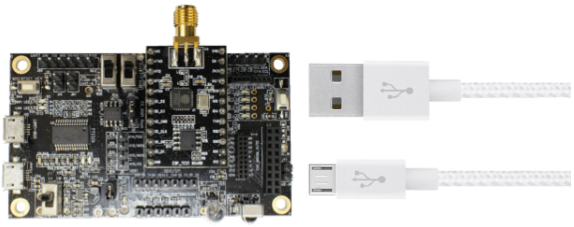
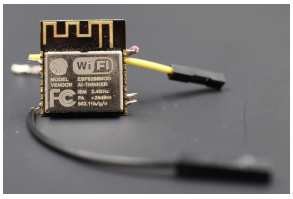
Users may directly input commands on the terminal and view or record responses in real time.

> 📖 ***Note:***
> *We recommend CoolTerm (for Windows and Mac OS) and Minicom (for Linux OS) as the serial port debug tool.*

2. Preparing the Hardware

Depending on whether you are using the ESP-LAUNCHER or the ESP-WROOM-02, you will need the hardware mentioned in Table 2-1 below:

Table 2-1. Hardware Preparations

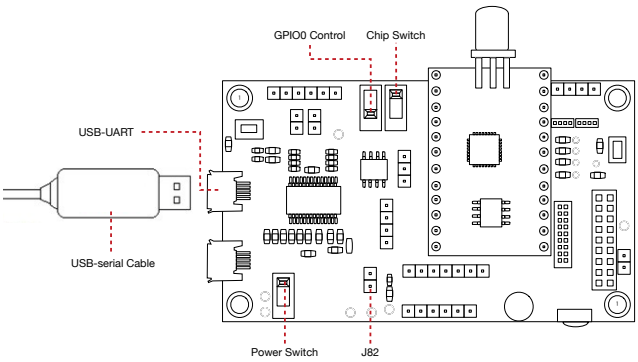
ESP-LAUNCHER	ESP-WROOM-02
• 1 ESP-LAUNCHER • 1 USB cable	• 1 ESP-WROOM-02 • 1 USB-to-TTL converter (FT232R recommended) • 6 Dupont lines • 1 slodering tool suite
	OR  
1 PC with pre-installed Windows OS	

⚠ **Notice:**

The ESP8266 Wi-Fi module needs 3.3 V power supply and a minimum current of 500 mA.

2.1. ESP-LAUNCHER

1. Connect PC to the USB-UART interface of ESP-LAUNCHER using the USB cable.
2. Set ESP-LAUNCHER to download mode.

Steps	Result
• Turn Power Switch to the outer side as the figure on the right 👉 shows. • Turn `GPIO0` Control to the inner side and enable download mode for ESP-LAUNCHER. ⚠ ***Notice:*** *J82 must be shorted by a jumper, otherwise code cannot be downloaded to the board.*	

3. Connect the USB-to-TTL converter to the PC.

> 📖 ***Note:***
>
> *Make sure that the proper driver for the USB-to-TTL converter is installed and it is recognized by the computer.*

4. Power on ESP-LAUNCHER by turning Power Switch to the inner side.
5. Power on the chip by turning Chip Switch to the outer side.
6. Download firmware to flash with ESP8266 DOWNLOAD TOOL.

> 📖 ***Note:***
>
> *On how to download firmware, please refer to **Chapter 4 Flash Map** and **Chapter 6 Downloading the Firmware**.*

7. After downloading, turn `GPIO0` Control to the outer side to enable working mode for ESP-LAUNCHER.
8. Power on the chip again with Chip Switch and the chip will read and run programs from the flash.

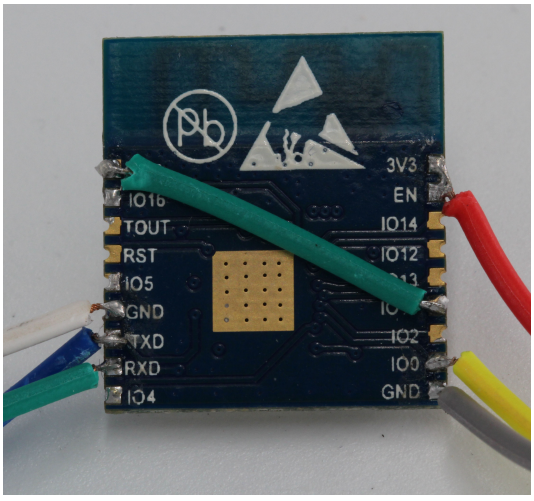
END

For more information on ESP-LAUNCHER hardware, please refer to ***ESP8266 Hardware Description.***

2.2. ESP-WROOM-02

1. Lead out the pins of ESP-WROOM-02 as shown in Table 2-2.

Table 2-2. ESP-WROOM-02 Pins

Pin	Pin status	Figure
EN	Pull up	
3V3	3.3 V power supply (VDD)	
IO15	Pull down	
IO0	UART download: pull down; Flash boot: floating / pull up	
GND	GND	
RXD	Receive end in UART download	
TXD	Transmit end in UART download; floating / pull up	

2. Connect ESP-WROOM-02 to USB-to-TTL converter using Dupont lines as shown in Figure 2-1.

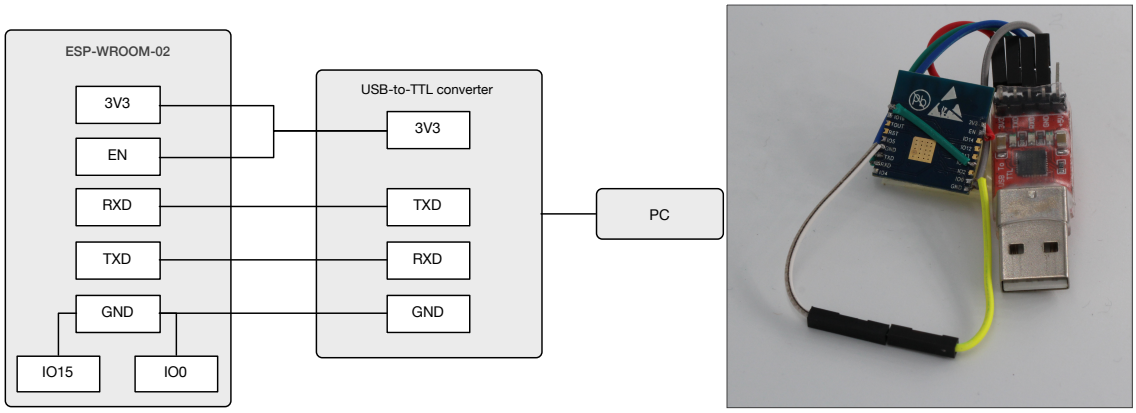


Figure 2-1. ESP-WROOM-02 Download Mode

3. Connect the USB-to-TTL converter to the PC.
4. Download firmware to flash with ESP8266 DOWNLOAD TOOL.

> 📖 ***Note:***
>
> *On how to download firmware, please refer to* ***Chapter 4 Flash Map*** *and* ***Chapter 6 Downloading the Firmware****.*

5. After downloading, switch ESP-WROOM-02 to working mode.
   Set `IO0` as floating or pull up.
6. Power on ESP-LAUNCHER again and the chip will read and run programs from the flash.

— — END

> 📖 ***Notes:***
>
> - `IO0` *is an internal pull up pin.*
> - *For more information on ESP-WROOM-02 hardware, please refer to* ***ESP8266 System Description*** *and* ***ESP-WROOM-02 Datasheet****.*

3. Preparing the Software

3.1. Non-OS SDK

Users can download Non-OS SDK (including application examples) at:

- Espressif's official website
  http://www.espressif.com/en/support/download/sdks-demos

Table 3-1 shows the directory structure of Non-OS SDK.

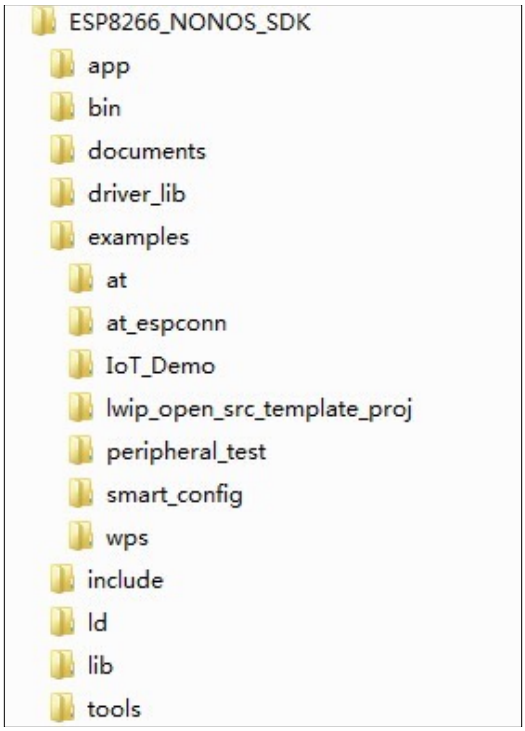


Figure 3-1. Non-OS SDK Directory Structure

- ***app***: the main working directory that contains source codes and header files to be compiled.
- *bin*: compiled binaries to be downloaded directly into the flash.
- ***documents***: SDK-related documents or links.
- ***driver_lib***: library files that drive peripherals such as UART, I2C and GPIO.
- ***examples***: sample codes for secondary development, for example, IoT Demo.
- ***include***: header files pre-installed in SDK. The files contain relevant API functions and other macro definitions. Users do not need to modify it.
- ***ld:*** files for SDK software link. We recommend users not modify ing it without specific requirements.
- ***lib:*** library files provided in SDK.
- ***tools:*** tools needed for compiling binaries. Users do not need to modify it.

3.2. RTOS SDK

Users can download RTOS SDK and its application examples (ESP8266_IOT_PLATFORM) at:

- RTOS SDK
  https://github.com/espressif/ESP8266_RTOS_SDK
- ESP8266_IOT_PLATFORM
  https://github.com/espressif/ESP8266_IOT_PLATFORM

Table 3-2 shows the directory structure of RTOS SDK.

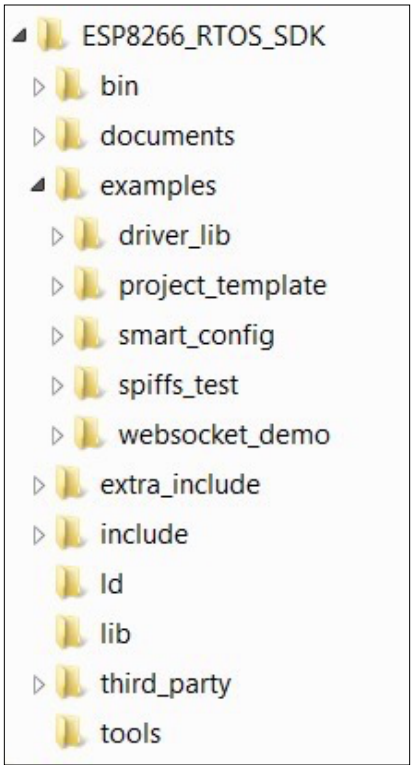


Figure 3-2. RTOS SDK Directory Structure

- ***bin:*** compiled binaries to be downloaded directly into the flash.
- ***documents:*** SDK-related documents or links.
- ***examples:*** sample codes for secondary development.
  - ***examples/driver_lib:*** library files that drive peripherals such as UART, I2C and GPIO.
  - ***examples/project_template:*** project directory template.

> 📖 ***Note:***
>
> *Users can copy **project_template** to any directory, for example, ~/workspace.*

  - ***examples/smart_config:*** Smart Config-related sample codes.
  - ***examples/spiffs_test:*** SPIFFS-related sample codes.

  - ***examples/websocket_demo:*** WebSocket-related sample codes.
- ***extra_include:*** header files provided by Xtensa.
- ***include:*** header files pre-installed in the SDK. The files contain relevant API functions and other macro definitions. Users do not need to modify it.
- ***ld:*** files for SDK software link. We recommend users not modifying it without specific requirements.
- ***lib:*** library files provided in SDK.
- ***third_party:*** third party open source library file.
- ***tools:*** tools needed for compiling binaries. Users do not need to modify it.

3.3. ESP8266 Toolkit

3.3.1. Compiler

Please download VirtualBox at:

https://www.virtualbox.org/wiki/Downloads

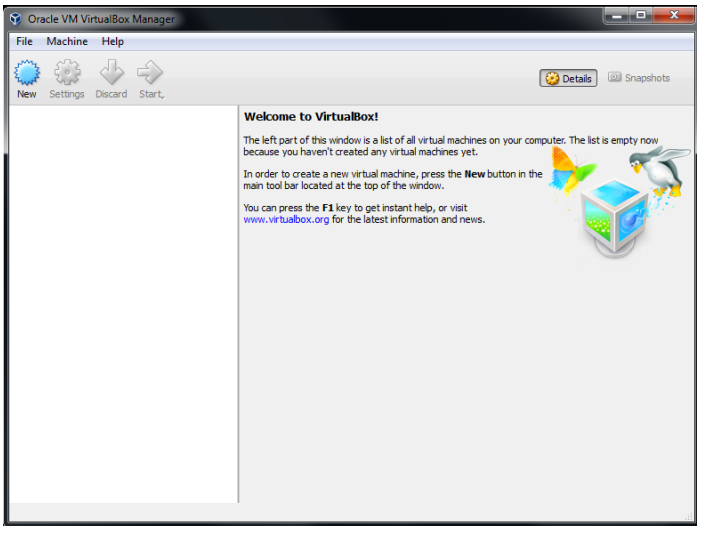
> 📖 ***Note:***
> *Please choose the right version of VirtualBox according to your host machine OS.*

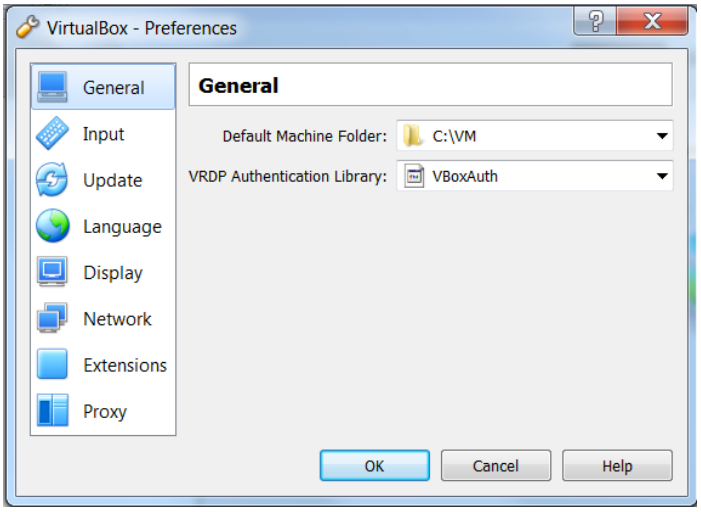
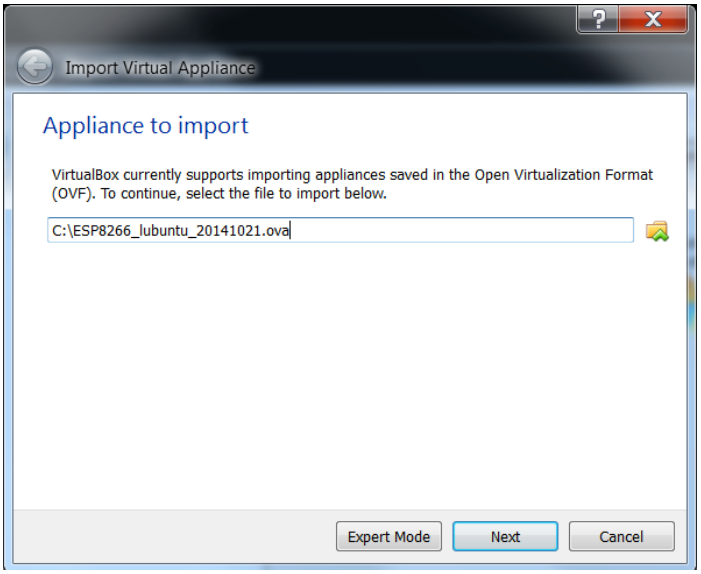
Please download the compiler ***ESP8266_lubuntu_20141021.ova*** at:

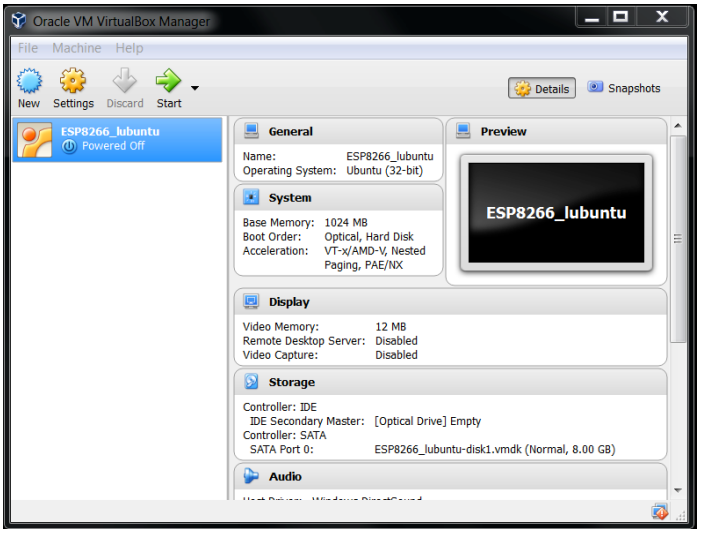
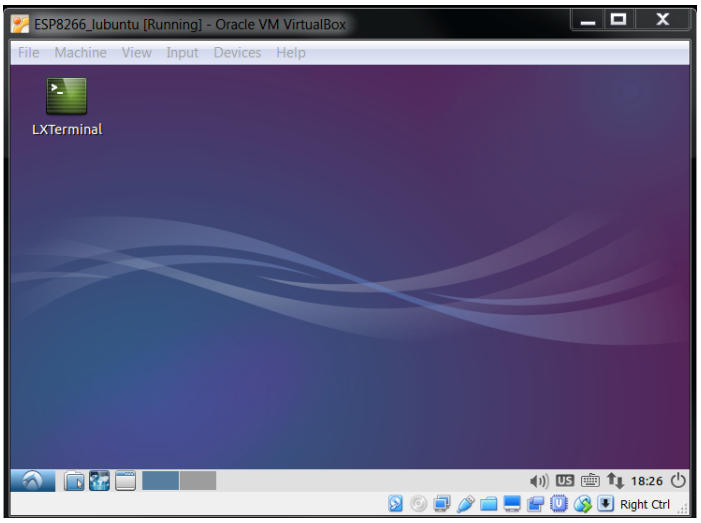
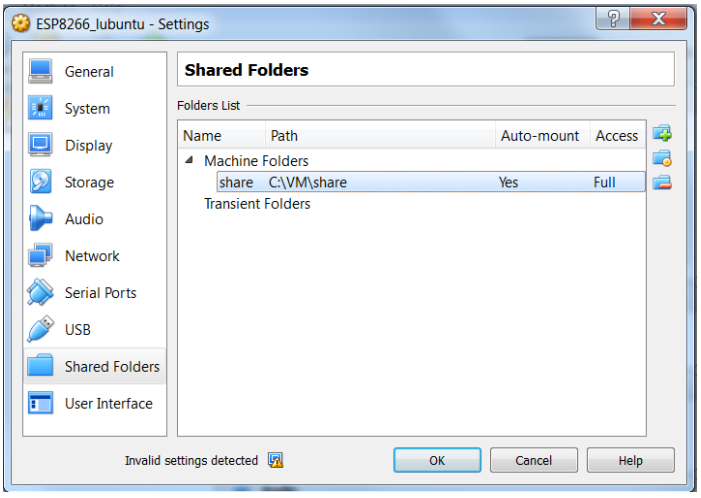
Baidu: *https://pan.baidu.com/share/init?shareid=3541602653&uk=190196792&third=15*

Password: qudl

Google: *https://drive.google.com/folderview?id=0B5bwBE9A5dBXaExvdDExVFNrUXM&usp=sharing*

Steps	Results
1. Start Windows OS and install the virtual machine.	
• Double-click ***VirtualBox-5.0.16-105871-Win.exe*** and install VirtualBox. 📖 ***Note:*** *VirtualBox has different versions. We are using Windows V.5.0.16 as an example.* • Double-click ***Oracle VM VirtualBox.exe*** to run the program and the system will show the main menu 👉. 💬 ***Tip:*** *ESP8266 virtual machine takes up a lot of space (memory). Please reserve enough space for it.*	

Steps	Results
2. Configure VirtualBox file folder.	
• Create a new folder, for example, ***C:\VM.*** • Select ***File > Preferences***, the system shows the dialog box 👉. • Select ***General***, set the location for the virtual machine in ***Default Machine Folder***, for example, ***C:\VM***.	
3. Import the image file.	
• Select ***File > Import Appliance***, a dialog box will show up 👉. • Select the demanded image file, for example, ***C:\ESP8266_lubuntu_20141021.ova***, and click ***Next***. • Click ***Import*** to confirm the settings.	
4. Run the virtual machine.	

Steps	Results
• After importing, a virtual machine named ***ESP8266_lubuntu*** shows up 👉. • Double-click ***ESP8266_lubuntu*** or ***Start*** to run the virtual machine.	
• The system shows ESP8266 virtual machine 👉. • If a dialog box as below 👇 shows up, please enter the password: ***espressif***. 	
5. Create a shared folder.	
• Create a new folder named ***C:\VM \share***. • Select ***Machine > Settings > Shared Folders…***, a dialog box will show up 👉. • Select the shared folder in ***Machine Folders***, for example, ***C:\VM\share***.	

3.3.2. Firmware Download Tool

Please download ESP8266 DOWNLOAD TOOL at:

http://www.espressif.com/support/download/other-tools.

4. Flash Map

This chapter provides the flash maps for FOTA and Non-FOTA firmware in flash of different capacities. Users can modify the map as needed.

Figure 4-1 shows the flash maps.

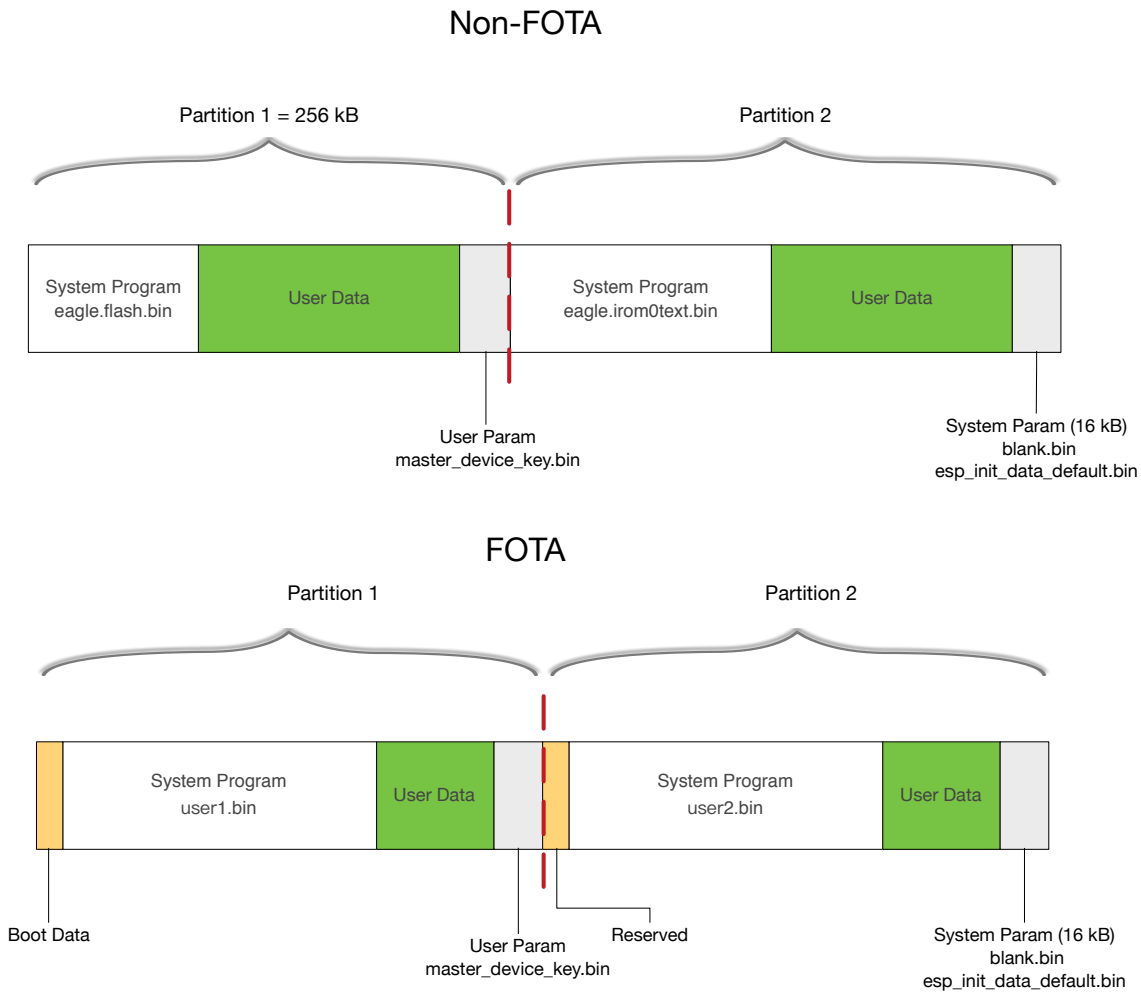


Figure 4-1. Flash Map

> 📖 ***Note:***
>
> *For ESP8266 firmware, please refer to **1.3 ESP8266 FW**.*

- ***System Program***: this area stores the firmware necessary for system running.
- ***User Data***: If system data does not take up all the flash memory, the remaining area can be used to store user data.
- ***User Param***: Users can define the address. In IOT_Demo, the four sectors starting from 0x3C000 are defined as the user parameter area. Users can define any available address for this area.

  - ***master_device_key.bin***: In IOT_Demo, it is located in the third sector of user parameter area.
- ***System Param***: this area is the last four sectors of flash.
  - ***blank.bin***: the download address is the second last sector in the flash.
  - ***esp_init_data_default.bin***: the download address is the fourth last sector of flash.
- ***Boot Data***: located in Partition 1 of FOTA firmware, and stores FOTA related data.
- ***Reserved***: a reserved area in Partition 2 of FOTA firmware, corresponding to ***Boot data*** area in Partition 1 of FOTA firmware.

> 📖 ***Notes:***
>
> - *Each sector of the flash is 4 KB.*
> - *For detailed download addresses, please refer to **4.1.1 Flash Map** and **4.2.2 Flash Map**.*

4.1. Non-FOTA

4.1.1. Flash Map

For flash of different capacities, the storage limit for ***eagle.irom0text.bin*** is 200 kB. Users can change the limit by modifying ***ESP8266_NONOS_SDK/ld/eagle.app.v6.ld***.

You can modify the `len` field in `irom0_0_seg` as shown in Figure 4-2.

```
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
}
```

Figure 4-2. Location for `len`

Table 4-1 shows the storage limits for *eagle.irom0text.bin* with different `len` values.

Table 4-1. Non-FOTA Flash Map (unit: kB)

Flash capacity	eagle.flash.bin	eagle.irom0text.bin	User data	len	User/System Param
512	≤ 64	≤ 240	≥ 176	0x3C000	16
1024	≤ 64	≤ 752	≥ 176	0xBC000	16
2048	≤ 64	≤ 768	≥ 176	0xC0000	16
4096	≤ 64	≤ 768	≥ 176	0xC0000	16

> 📖 ***Note:***
>
> *ESP8266 presently only supports System Param area up to 1024 kB.*

4.1.2. Download Addresses

Table 4-2 lists the download addresses for Non-FOTA firmware.

Table 4-2. Download Address for Non-FOTA Firmware (unit: kB)

Binaries	Download addresses in flash of different capacities			
	512	1024	2048	4096
master_device_key.bin	0x3E000			
esp_init_data_default.bin	0x7C000	0xFC000	0x1FC000	0x3FC000
blank.bin	0x7E000	0xFE000	0x1FE000	0x3FE000
eagle.flash.bin	0x00000			
eagle.irom0text.bin	0x40000			

4.2. FOTA

4.2.1. Flash Map

Table 4-3 lists the download addresses for FOTA firmware.

Table 4-3. FOTA Flash Map (unit: kB)

Flash capacity	boot	user1.bin	user2.bin	User/System Param	User data
512	4	≤ 236	≤ 236	16	≥ 0
1024	4	≤ 492	≤ 492	16	≥ 0
2048 (Partition 1 = 512)	4	≤ 492	≤ 492	16	≥ 1024
2048 (Partition 1 = 1024)	4	≤ 1004	≤ 1004	16	≥ 0
4096 (Partition 1 = 512)	4	≤ 492	≤ 492	16	≥ 3072
4096 (Partition 1 = 1024)	4	≤ 1004	≤ 1004	16	≥ 2048

4.2.2. Download Addresses

Table 4-4 lists the download addresses for FOTA firmware.

Table 4-4. Download Addresses for FOTA Firmware (unit: kB)

Binaries	Download addresses in flash of different capacities					
	512	1024	2048		4096	
			512+512	1024+1024	512+512	1024+1024
master_device_key.bin	0x3E000	0x7E000	0x7E000	0xFE000	0x7E000	0xFE000
esp_init_data_default.bin	0x7C000	0xFC000	0x1FC000		0x3FC000	
blank.bin	0x7E000	0xFE000	0x1FE000		0x3FE000	
boot.bin	0x00000					
user1.bin	0x01000					
user2.bin	0x41000	0x81000	0x81000	0x101000	0x81000	0x101000

📖 ***Notes:***

- *For FOTA firmware, you don't need to download* ***user2.bin,*** *but upgrade the firmware through the cloud server.*
- *For details on FOTA functional description, please refer to* ***ESP8266 FOTA Upgrade Guide****.*

5. Compiling the SDK

> 📖 ***Notes:***
>
> - *This chapter demonstrates how to compile SDK by taking **ESP8266_NONOS_SDK/examples/IoT_Demo** as an example.*
> - ***IoT_Demo** provides three devices, i.e., Smart Light, Smart Plug and Sensor that are defined in **examples>IoT_Demo/include/user_config.h**. Users can only commission one device at a time. The default device for commissioning is Smart Light.*

5.1. Preparations

5.1.1. Modify SDK Files

> 📖 ***Note:***
>
> *Users need to modify SDK files if using FOTA firmware.*

1. Start Windows OS.
2. Modify files in ***ESP8266_NONOS_SDK/examples/IoT_Demo/include*** according to different Flash maps.
   - Modify `#define PRIV_PARAM_START_SEC` in ***user_light.h*** and ***user_plug.h***.

```
/* NOTICE !!! ---this is for 512KB spi flash.*/
/* You can change to other sector if you use other size spi flash. */
/* Refer to the documentation about OTA support and flash mapping*/
#define PRIV_PARAM_START_SEC          0x3C
#define PRIV_PARAM_SAVE     0
```

   - Modify `#define ESP_PARAM_START_SEC` in ***user_esp_platform.h***.

```
/* NOTICE---this is for 512KB spi flash.
 * you can change to other sector if you use other size spi flash. */
#define ESP_PARAM_START_SEC           0x3D
```

Table 5-1 lists the modified values.

Table 5-1. Modify the Field Values in include File (unit: kB)

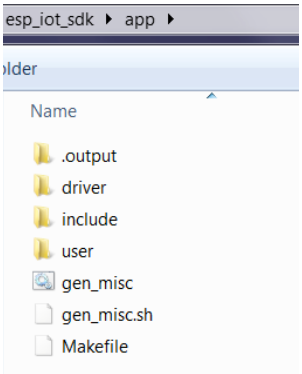
Default value (512)	Modified values					
	512	1024	2048 (512+512)	2048 (1024+1024)	4096 (512+512)	4096 (1024+1024)
0x3C	-	0x7C	0x7C	0xFC	0x7C	0xFC
0x3D	-	0x7D	0x7D	0xFD	0x7D	0xFD

> 📖 ***Note:***
>
> *Users need not to modify SDK files if using 512 kB flash.*

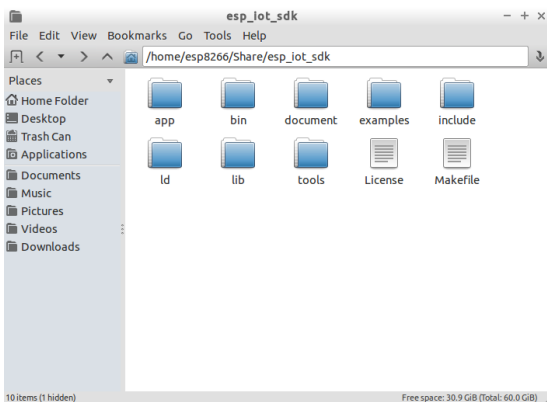
— — END

5.1.2. Download SDK Files

1. Start Linux OS.
2. Run LXTerminal on the desktop of virtual machine.
3. Copy the files to be compiled to the shared folder.

Steps	Result
• Copy ***ESP8266_NONOS_SDK*** folder to the shared directory, for example, ***C:\VM\share***. • Copy ***IoT_Demo*** folder to ***C:\VM\share \ESP8266_NONOS_SDK***, as shown in the figure on the right 👉.	

4. Download shared directory.

步骤	结果
• Execute `./mount.sh`. • Input the password: ***espressif***. Shared files download is completed. • Open the shared directory ***ESP8266_NONOS_SDK*** in the virtual machine and confirm if the download is successful. - If successful, the directory contains files as the figure on the right 👉 shows. - If not, the directory will be empty, you will need to do this step again.	

> ⚠ ***Notice:***
>
> *If you are using RTOS SDK, please continue the following steps; if you are using Non-OS SDK, please skip Step 5.*

5. Set variable PATH to point to SDK and binaries.

```
export SDK_PATH=~/share/ESP8266_RTOS_SDK
```

```
export BIN_PATH=~/share/ESP8266_RTOS_SDK/bin
```

> 📖 ***Note:***
>
> *You can add it to* ***.bashrc*** *file, otherwise you need to repeat Step 5 each time restarting the compiler.*

– – END

5.2. Compilation

5.2.1. Compile ESP8266_NONOS_SDK_v0.9.5 and Later

1. Switch to ***/share/ESP8266_NONOS_SDK/app*** directory on the terminal.

```
cd /home/esp8266/Share/ESP8266_NONOS_SDK/app
./gen_misc.sh
```

The system shows the following information.

```
gen_misc.sh version 20150511
Please follow below steps(1-5) to generate specific bin(s):
```

2. Select the required options as shown in Figure 5-1.

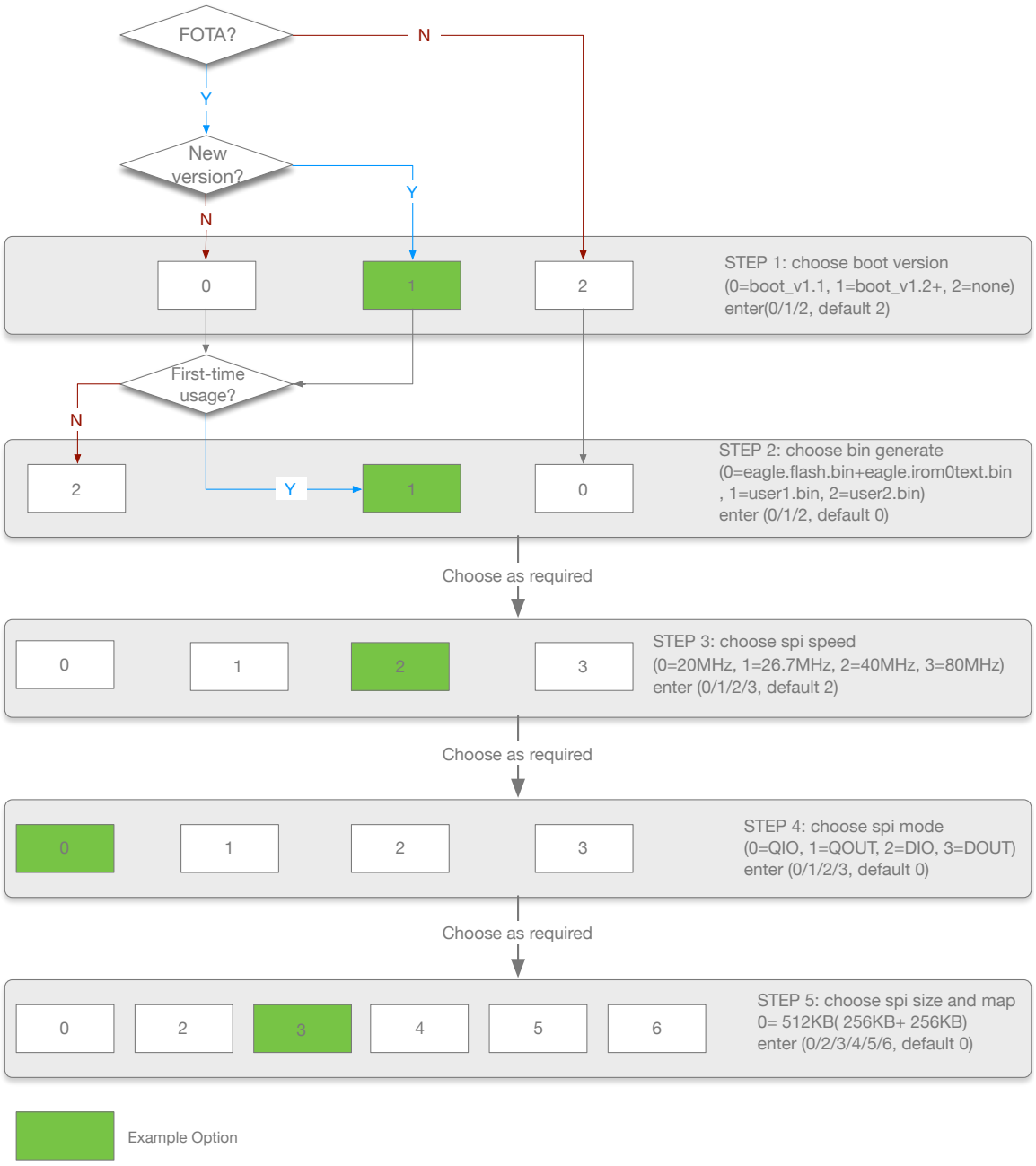


Figure 5-1. Compile SDK

📖 ***Notes:***

- *The sample options are marked in green. Users can select the right options as needed.*
- *For FOTA and Non-FOTA binaries, please refer to **1.4 ESP8266 FW**.*
- *Only sdk_v1.1.0 + boot 1.4 + flash download tool_v1.2 and higher versions support options 5 and 6 in Step 5.*
- *After compiling **user1.bin**, execute `make clean` first to clean up the temporary files generated by last compilation, then compile **user2.bin**.*
- *For Flash map in Step 5, please refer to **Chapter 4 Flash Map**.*

3. After compilation, the binaries generated and the addresses in flash are shown as follows.

```
Generate user1.2048.new.3.bin successfully in folder bin/upgrade.
boot.bin------------>0x00000
user1.2048.new.3.bin--->0xSupport boot_v1.2 and +
01000
!!!
```

> 📖 ***Note:***
>
> *You can open **/home/esp8266/Share/ESP8266_NONOS_SDK/bin** directory and check the binaries compiled.*

– – END

5.2.2. ESP8266_NONOS_SDK_v0.9.4 and Earlier

For ESP8266_NONOS_SDK_v0.9.4 and previous versions, the compilation process is as follows.

1. Execute `./gen_misc_plus.sh 1` to generate ***user1.bin*** under the ***/ESP8266_NONOS_SDK/bin/upgrade*** path.
2. Execute `make clean` to clean up previous compilation data.
3. Execute `./gen_misc_plus.sh 2` to generate ***user2.bin*** under the ***/ESP8266_NONOS_SDK/bin/upgrade*** path.

> 📖 ***Note:***
>
> *ESP8266_NONOS_SDK_v0.7 and earlier are Non-FOTA.*

– – END

6. Downloading the Firmware

6.1. Download Procedure

1. Start Windows OS.
2. Double-click ***ESP_DOWNLOAD_TOOL.exe*** to open Flash tool.

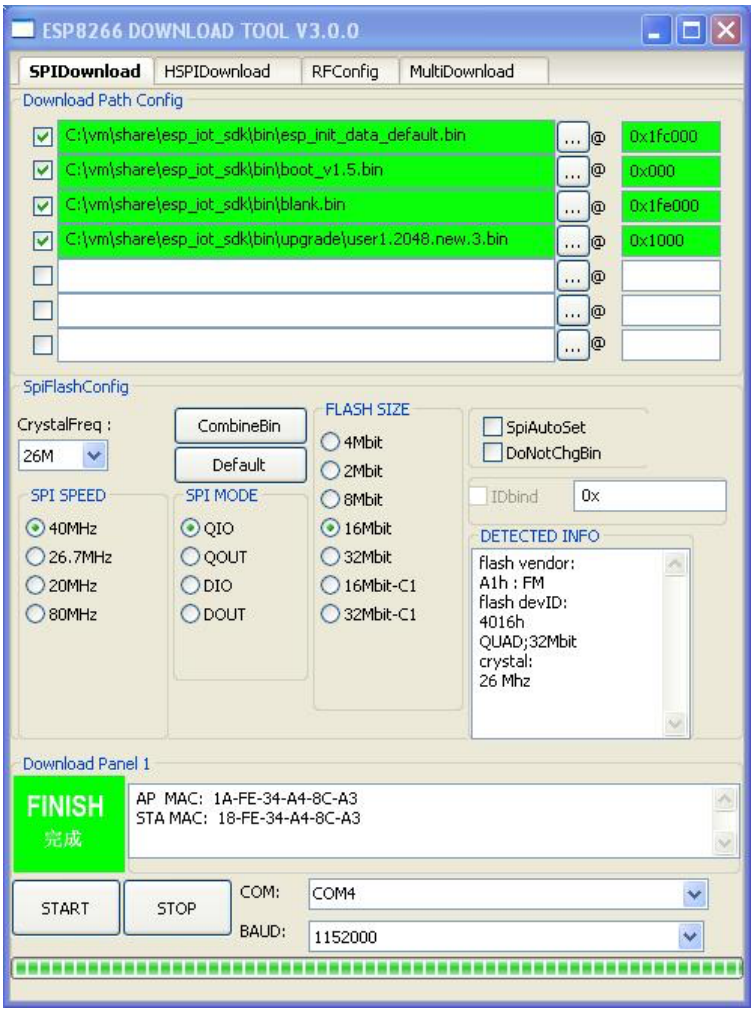


Figure 6-1. ESP8266 DOWNLOAD TOOL - SPIDownload

SPIDownload	For SPI Flash download.
HSPIDownload	For HSPI Flash download.
RFConfig	RF initialization Configuration.
MutiDownload	For multi-mother boards download.

3. Double-click  in ***Download Path Config*** panel to select the binaries to be downloaded. Set the corresponding download addresses in ***ADDR***.
4. Configure SPIDownload.

> 📖 ***Note:***
>
> *The binaries to be download and the corresponding addresses vary with SPI Flash size and actual demands. For details, please refer to* ***Chapter 4 Flash Map****.*

Table 6-1. SPIDownload Configuration

Items	Description
SPI FLASH CONFIG	
CrystalFreq	Select the crystal frequency according to the crystal oscillator used.
CombineBin	Combine the selected binaries into ***target.bin*** with the address 0x0000.
Default	Set the SPI Flash to the default value.
SPI SPEED	Select SPI read/write speed with the maximum value of 80 MHz.
SPI MODE	Select SPI mode according to the SPI Flash used. If the flash is Dual SPI, select ***DIO*** or ***DOUT***. If the flash is Quad SPI, select ***DIO*** or ***DOUT***. ⚠️ ***Notice:*** *If you are using ISSI Flash, please refer to* ***Appendix - Configure ISSI & MXIC Flash QIO Mode****.*
FLASH SIZE	Select the flash size according to the flash type used.
SpiAutoSet	We recommend not checking ***SpiAutoSet***, but configuring the flash manually as needed. If users select ***SpiAutoSet***, the binaries will be downloaded according to the default flash map. The flash map of 16 Mbit and 32 Mbit will be 512 KByte + 512 KByte.
DoNotChgBin	• If users select ***DoNotChgBin***, the flash working frequency, mode, and flash map will be based on the configuration when compiling. • If users do not select ***DoNotChgBin***, the flash working frequency, mode, and flash map will be defined by the final configuration of the compiler.
Download Panel	
START	Click ***START*** to start download. When the download completes, ***FINISH*** will appear in the green area on the left.
STOP	Click ***STOP*** to stop download.
MAC Address	If download is successful, the system will show the MAC addresses of ESP8266 STA and ESP8266 AP.
COM PORT	Select COM port of ESP8266.
BAUDRATE	Select the baud rate of downloading. The default value is 115200.

5. After downloading, turn `GPIO0` Control on ESP-LAUNCHER to the outer side and power the board on to enable the working mode.

——END

6.2. Check Log File

After downloading firmware, you can check the log printed on the terminal end with serial port debug tool.

You need to configure the following options of the serial port debug tool.

Table 6-2. Serial Port Debug Tool Configuration

Items	Configuration Description
Protocol	Serial port.
Port number	Set the port number according to the connected device.
Baud rate	Baud rate at which the device is running, related to the crystal oscillator. • 69120 (24 M crystal oscillator) • 74880 (26 M crystal oscillator) • 115200 (40 M crystal oscillator) The ESP8266 AT example supports the baud rate of 115200 by default. Users can not modify it. The ESP8266 IOT Demo example supports the baud rate of 74880. Users can modify it.
Data bit	8
Calibration	None.
Flow control	None.

6.2.1. ESP8266 IOT Demo

If users are downloading ESP8266 IOT Demo firmware, the system will show the initialization information including SDK version, etc in working mode. “Finish” means the firmware works properly.

```
SDK version:X.X.X(e67da894)
IOT VERSION = v1.0.5t45772(a)
reset reason: 0
PWM version: 00000003
mode: sta(18:fe:34:a4:8c:a3) + softAP(1a:fe:34:a4:8c:a3)
add if0
add if1
```

```
dhcp server start:(ip:192.168.4.1,mask:255.255.255.0,gw:192.168.4.1)
bcn 100
finish
```

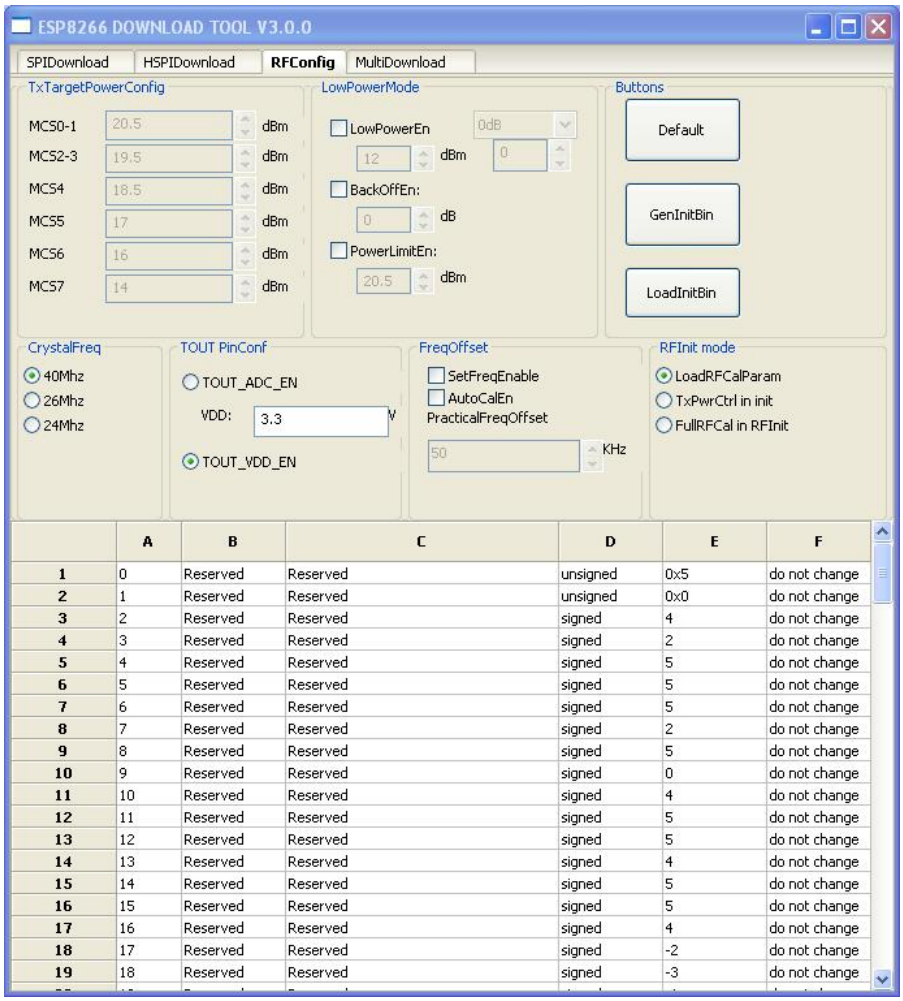
6.2.2. ESP8266 AT

If users are downloading ESP8266 AT firmware or the default firmware in ESP-LAUNCHER or ESP-WROOM-02, the system will display “Ready” at the end in the working mode. Input command “AT” on the terminal end and the system will return “OK”, meaning that the firmware is working properly.

> 📖 ***Notes:***
>
> - The baud rate in AT firmware is mandatorily configured as 115200, and the default baud rate of ESP8266 is 74880. Due to the discrepancy, the system initialization information will be displayed as Mojibake. It is a normal phenomenon as long as the system shows “Ready” in the end.
> - For more information on AT commands, please refer to ***ESP8266 AT Instruction Set***.

6.3. RF initialization Configuration (Optional)

Before downloading binaries to flash, users can modify RF initialization settings in the ***RF InitConfig*** tab. The newly generated ***esp_init_data_setting.bin*** can be downloaded to flash instead of ***esp_init_data_default.bin***. Users can configure both the options and the parameters for RF settings.



	A	B	C	D	E	F
1	0	Reserved	Reserved	unsigned	0x5	do not change
2	1	Reserved	Reserved	unsigned	0x0	do not change
3	2	Reserved	Reserved	signed	4	do not change
4	3	Reserved	Reserved	signed	2	do not change
5	4	Reserved	Reserved	signed	5	do not change
6	5	Reserved	Reserved	signed	5	do not change
7	6	Reserved	Reserved	signed	5	do not change
8	7	Reserved	Reserved	signed	2	do not change
9	8	Reserved	Reserved	signed	5	do not change
10	9	Reserved	Reserved	signed	0	do not change
11	10	Reserved	Reserved	signed	4	do not change
12	11	Reserved	Reserved	signed	5	do not change
13	12	Reserved	Reserved	signed	5	do not change
14	13	Reserved	Reserved	signed	4	do not change
15	14	Reserved	Reserved	signed	5	do not change
16	15	Reserved	Reserved	signed	5	do not change
17	16	Reserved	Reserved	signed	4	do not change
18	17	Reserved	Reserved	signed	-2	do not change
19	18	Reserved	Reserved	signed	-3	do not change

Figure 6-2. ESP8266 DOWNLOAD TOOL - RF InitConfig

6.3.1. RF InitConfig Options Configuration

RF InitConfig options are listed in the upper part in Figure 6-2. Please refer to Table 6-3 for configuration description.

Table 6-3. RF InitConfig Options Configuration

Items	Description
TxTargetPowerConfig	Users need not configure it. It varies with the options in LowPowerMode.
LowPowerMode	Configure low power mode as required. • *LowPowerEn*: enable low power mode, set a power value for all data rates. • *PowerLimtEn*: set a limit for output power. • *BackOffEn*: set backoff value for each data rate. 📖 ***Note:*** *You cannot configure LowPowerEn and PowerLimtEn at the same time.*

Items	Description
CrystalFreq	Select the crystal oscillator frequency according to the crystal oscillator used. 📖 ***Note:*** *If a different option is selected when downloading, it will override the configuration.*
TOUT PinConf	Configure TOUT pin according to the actual TOUT pin status. We recommend the default value. • ***TOUT_ADC_EN***: When TOUT pin connects to external circuit, measure the external voltage (0 V - 1 V) through internal embedded ADC. • ***TOUT_VDD_EN***: When TOUT pin is dangled, measure VDD33 voltage through uint16 system_get_vdd33(void). ⚠️ ***Notice:*** • *You cannot configure **TOUT_ADC_EN** and **TOUT_VDD_EN** at the same time.* • *When using **TOUT_ADC_EN**, you need to input the actual voltage on VDD3P3 pin 3 and pin 4.*
FreqOffset	• ***SetFreqEnable***: Set the frequency offset manually. - ***PracticalFreqOffset***: the option is valid when selecting ***SetFreqEnable***. • ***AutoCalEn***: Set the frequency offset automatically.
RFInt mode	Users can select RF initialization mode: • ***LoadRFCalParam*:** During RF initialization, RF data is loaded directly from the flash without any calibration. It takes about 2 ms and the least initial current. • ***TxPwrCtrl in init***: During RF initialization, only Tx Power calibration will be made, and other data is loaded from flash. It takes about 20 ms and small initial current. • ***FullRFCal in RFInit***: All calibration are made during RF initialization. It takes 200 ms and large initial current.

6.3.2. RF InitConfig Parameters Configuration

RF InitConfig parameters are listed in the lower part of Figure 6-2. Parameter description is shown in Table 6-4.

Table 6-4. RF InitConfig Parameters Configuration

Items	Description
A	The byte in ***esp_init_data_setting.bin*** (0 ~ 127 byte). For example, A = 0 represents Byte 0 in ***esp_init_data_setting.bin***.
B	The item name. Users cannot modify it if marked as Reserved.
C	The item name. Users cannot modify it if marked as Reserved.
D	Data types of configuration items, including unsigned and signed data types.
E	The hexadecimal value of configuration item.

> ⚠ ***Notice:***
>
> Please do not modify the parameters marked as Reserved.

The following introduces how to modify the 112 ~ 114 byte parameters. Figure 6-3 lists the initial configuration.

A	B	C	D	E	F
112	tx_param42	freq_correct_en	unsigned	0	bit[0]:0->do not correct fre
113	tx_param43	force_freq_offset	unsigned	0	signed, unit is 8khz
114	tx_param44	rf_cal_use_flash	unsigned	0	0: RF init no RF CAL, using

Figure 6-3. 112 ~ 114 Byte Parameters

Modify RF Initialization Parameters

114 byte is used to control RF initialization when ESP8266 is powered on. Table 6-5 provides the parameter configuration.

> 📖 ***Note:***
>
> *Supported by ESP8266_NONOS_SDK_V1.5.3 and ESP8266_RTOS_SDK_V1.3.0 and higher.*

Table 6-5. Modify RF Initialization Parameters

Option	Description
114 byte = 0	Only VDD33 calibration is performed during RF initialization. It takes about 2 ms and the least initial current.
114 byte = 1	The default value is 1. VDD33 and TX power calibrations are performed during RF initialization. It takes about 18 ms and small initial current.
114 byte = 2	The same as “114 byte = 0”.
114 byte = 3	All calibrations are performed during RF initialization. It takes about 200 ms and large initial current.

Correct Frequency Offset

The 112 and 113 bytes relate to frequency offset correction. Table 6-6 provides the parameter configuration.

> 📖 ***Note:***
>
> *Supported by ESP8266_NONOS_SDK_V1.4.0 and ESP8266_RTOS_SDK_V1.3.0 and higher.*

Table 6-6. Frequency Offset Correction Option

Option	Description
112 byte, the default value is 0.	
bit 0	the highest priority. • bit 0 = 0: frequency offset cannot be corrected. • bit 0 = 1: frequency offset can be corrected.
bit 1	Value = 0 means that the bbpll is 168 M. Both positive and negative frequency offset can be corrected. It may effect the digital peripheral performance, therefore, it is not recommended. Value = 1 means that the bbpll is 160 M. Only positive frequency offset can be corrected.
{bit 3, bit 2}	Value = 0 means the chip will track and correct the frequency offset automatically. The initial correction value is 0. Value = 1 means the chip will compulsively correct the frequency offset to be the value of the 113 byte, and will not track and correct the frequency offset automatically. Value = 2 means the chip will track and correct the frequency offset automatically. The initial correction value is the value of the 113 byte.
113 byte, the default value is 3.	
113 byte	Mandatory frequency offset correction value or the initial correction value in frequency tracking. The data type is sign int8, in multiples of 8 kHz.

6.3.3. Configuration Examples

The configuration of the 112 byte and 113 byte depends on users' specific needs. We provide some examples below for reference:

1. **The module works at ambient temperature and needs not correct frequency offset.**
   - Set 112 byte = 0, 113 byte = 0.
2. **The module works at ambient temperature and needs not track and correct the frequency offset automatically, but the frequency offset is large. In this case, mandatory frequency offset correction is recommended.**
   - If the frequency offset is +160 KHz (at ambient temperature), users can set 112 byte = 0x07, 113 byte = (256 - 160/8) = 236 = 0xEC.
   - If the frequency offset is -160 KHz (at ambient temperature), users can set 112 byte = 0x05, 113 byte = 160/8 = 20 = 0x14. This may effect the digital peripheral performance, so we don't recommend it.
3. **Applications work at a wide temperature range of -40 °C to 125 °C (such as smart light) and need to track and correct the frequency offset automatically. The frequency offset at ambient temperature is small, so the initial offset correction value is not needed.**
   - Set 112 byte = 0x03, 113 byte = 0.

4. **Applications work at a wide temperature range of -40 °C to 125 °C (such as smart light) and need to track and correct the frequency offset automatically. The frequency offset at ambient temperature is large, so the initial offset correction value is needed.**
    - If the frequency offset is +160 kHz (at ambient temperature), users can set 112 byte = 0x0B, 113 byte = (256 - 160/8) = 236 = 0xEC.
    - If the frequency offset is -160 kHz (at ambient temperature), users can set 112 byte = 0x09, 113 byte = 160/8 = 20 = 0x14. But this may effect the digital peripheral performance and needs substantive tests, so we don't recommend it.

We recommend Example 3.

When RF initialization configuration is done, click ***GenInitBin*** button to generate

esp_init_data_setting.bin.

In addition, users can click ***Default*** button to set the value to default, or click ***LoadInitBin***

button to import a binary file for configuration.

I. Appendix - Configuring ISSI & MXIC Flash QIO Mode

> ⚠ ***Notice:***
> *Choose DIO or DOUT mode when downloading, otherwise errors may occur. There is no need to modify binaries in DIO or DOUT mode.*

When using QIO mode of ISSI flash or MXIC flash, you need to modify the first two bytes in ***blank.bin*** as shown in Table I-I. During initialization, ESP8266 will automatically check the first two bytes and switch to QIO mode to read ISSI_FLASH or MXIC_FLASH. The structure of ***blank.bin*** is as below.

```
  strcut boot_hdr{
    char user_bin:2;     //low_bit
    char boot_status:1;
    char to_qio:1;
    char reverse:4;
    char version:5;      //low bit
    char test_pass_flag:1;
    char test_start_flag:1;
    char enhance_boot_flag:1;
}
```

Table I-I. blank.bin Configuration

Option	Description
Without secondary boot loader	Modify `to_qio` to 0.
With secondary boot loader	Modify `use_bin` to 0 and `to_qio` to 0. Modify version to the current boot version. ***Example:*** If you are using secondary ***boot_v1.5.bin***, modify the first two bytes FF FF to F4 E5.

Espressif IOT Team

www.espressif.com