About This Guide

This document takes ESP-LAUNCHER and ESP-WROOM-02 as examples to introduce how to use the ESP8266 SDK V2.X and earlier versions. The contents include preparations before compilation, SDK compilation and firmware download.

For the new ESP8266_RTOS_SDK (ESP-IDF style) V3.0 and later versions, please see ESP8266_RTOS_SDK/docs.

Release Notes

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Release notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016.04</td>
<td>V2.0</td>
<td>First release.</td>
</tr>
<tr>
<td>2016.07</td>
<td>V2.1</td>
<td>Added MXIC Flash QIO mode; Modified the default value of byte 112 to 0.</td>
</tr>
<tr>
<td>2016.07</td>
<td>V2.2</td>
<td>Updated Section 3.3.1.</td>
</tr>
<tr>
<td>2016.08</td>
<td>V2.3</td>
<td>Updated the Baidu link in Section 3.3.1.</td>
</tr>
<tr>
<td>2016.10</td>
<td>V2.4</td>
<td>Updated the flash address of eagle.irom0.text.bin in Section 4.1.1.</td>
</tr>
<tr>
<td>2016.11</td>
<td>V2.5</td>
<td>Added Appendix B—Learning Resources.</td>
</tr>
<tr>
<td>2017.01</td>
<td>V2.6</td>
<td>Modified the default value of byte 113 to 0 in Table 6-6.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added two Github links of RTOS and non-OS SDK sample code in Appendix B.2—Must-Have Resources.</td>
</tr>
<tr>
<td>2017.02</td>
<td>V2.7</td>
<td>Updated sections 3.1 and 3.2; Updated the link for the OVA image file in Section 3.3.1; Updated Section 5.1.2.</td>
</tr>
<tr>
<td>2017.05</td>
<td>V2.8</td>
<td>Updated Chapter 4 for 8 MB and 16 MB flash support.</td>
</tr>
<tr>
<td>2017.11</td>
<td>V2.9</td>
<td>Updated Table 1-1 in Chapter 1; Updated Figure 4-1 and the parameter descriptions in Chapter 4; Updated Table 4-1, Table 4-2, Table 4-3 and Table 4-4 in Chapter 4.</td>
</tr>
<tr>
<td>2018.03</td>
<td>V3.0</td>
<td>Updated section 4.1.1; Updated Table 4-1 in Chapter 4.</td>
</tr>
<tr>
<td>2018.06</td>
<td>V3.1</td>
<td>Updated Table 4-3 in Chapter 4.2.1.</td>
</tr>
<tr>
<td>2019.02</td>
<td>V3.2</td>
<td>Updated the description in Appendix A; Updated the document format.</td>
</tr>
<tr>
<td>2019.12</td>
<td>V3.3</td>
<td>Updated the description of ESP8266_RTOS_SDK (ESP-IDF style) V3.0 and later versions in About this Guide.</td>
</tr>
<tr>
<td>2020.09</td>
<td>V3.4</td>
<td>Removed section 6.3; Removed some obsolete links.</td>
</tr>
</tbody>
</table>
Documentation Change Notification


Certification

# Table of Contents

1. Overview .........................................................................................................................1
   1.1. Procedure Overview ..................................................................................................1
   1.2. ESP8266 HDK ..........................................................................................................1
   1.3. ESP8266 SDK ...........................................................................................................2
       1.3.1. Non-OS SDK ....................................................................................................2
       1.3.2. RTOS SDK ......................................................................................................2
   1.4. ESP8266 FW ............................................................................................................2
   1.5. ESP8266 Toolkit .......................................................................................................3
       1.5.1. Compiler ...........................................................................................................3
       1.5.2. Firmware Download Tool .................................................................................3
       1.5.3. Serial Port Debug Tool .....................................................................................3

2. Preparing the Hardware ....................................................................................................5
   2.1. ESP-LAUNCHER .......................................................................................................5
   2.2. ESP-WROOM-02 .....................................................................................................6

3. Preparing the Software .....................................................................................................8
   3.1. Non-OS SDK ...........................................................................................................8
   3.2. RTOS SDK .............................................................................................................8
   3.3. ESP8266 Toolkit ....................................................................................................10
       3.3.1. Compiler .........................................................................................................10
       3.3.2. Firmware Download Tool ..............................................................................12

4. Flash Maps ....................................................................................................................13
   4.1. Non-OTA ................................................................................................................14
       4.1.1. Flash Map .......................................................................................................14
       4.1.2. Download Addresses ......................................................................................15
   4.2. OTA Firmware .......................................................................................................15
       4.2.1. Flash Map .......................................................................................................15
       4.2.2. Download Addresses ......................................................................................16

5. Compiling the SDK .........................................................................................................17
   5.1. Preparations ............................................................................................................17
       5.1.1. Modifying SDK Files ......................................................................................17
1. Overview

1.1. Procedure Overview

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

1.2. ESP8266 HDK

The ESP8266 HDK (Hardware Development Kit) 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 users use other development boards or modules that integrate ESP8266EX, please use the development firmware provided by the corresponding manufacturers.
- If users would like to purchase ESP-WROOM-02 or ESP-LAUNCHER, please visit Espressif's official online store at: [https://espressif.taobao.com](https://espressif.taobao.com).
1.3. **ESP8266 SDK**

The ESP8266 Software Development Kit (SDK) is an Internet of Things (IoT) application development platform developed by Espressif for developers. It includes such examples of application development as Smart Lights and Smart Plugs.

Depending on whether they are based on an operating system (OS), SDKs can be categorized into two types: 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 such as nested events and functions triggered by certain conditions. Non-OS SDK uses the espconn network interface; users need to develop their software according to usage rules of the espconn interface.

1.3.2. **RTOS SDK**

RTOS SDK is based on FreeRTOS, open-source software development on Github.

- The FreeRTOS SDK is based on FreeRTOS, a multi-tasking OS. Users can use standard interfaces to realize resource management, recycling operations, execution delays, 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 a BSD Socket API interface. Users can directly use the socket API to develop software applications; and port to ESP8266 other applications from other platforms using the socket API, effectively reducing the learning costs arising from switching platforms.

- 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, SmartConfig interfaces, Sniffer related interfaces, system interfaces, timer interfaces, FOTA interfaces and peripheral driver interfaces, but does not support AT implementation.

1.4. **ESP8266 FW**

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

1.5. ESP8266 Toolkit

1.5.1. Compiler

Linux OS is required to compile the ESP8266 SDK. When 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 the ESP8266 SDK by importing the ESP8266 compiler (OVA image) into the virtual machine.

1.5.2. Firmware Download Tool

The ESP8266 DOWNLOAD TOOL is the official firmware download tool developed by Espressif. Users can download multiple binaries to the SPI Flash of the ESP8266 motherboard (ESP-LAUNCHER or ESP-WROOM-02) at the same 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 com port (USB-to-serial converter) can be used.

---

Table 1-1. ESP8266 FW

<table>
<thead>
<tr>
<th>Binaries</th>
<th>Compulsory or optional</th>
<th>Description</th>
<th>Non-OTA</th>
<th>OTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>esp_init_data_default.bin</td>
<td>Compulsory</td>
<td>Default system parameters provided in SDK.</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>blank.bin</td>
<td>Compulsory</td>
<td>Default system parameters provided in SDK.</td>
<td>☑</td>
<td>☑</td>
</tr>
<tr>
<td>eagle.flash.bin</td>
<td>Compulsory</td>
<td>Main program compiled from SDK.</td>
<td>☑</td>
<td>☒</td>
</tr>
<tr>
<td>eagle.irom0text.bin</td>
<td>Compulsory</td>
<td>Main program compiled from SDK.</td>
<td>☑</td>
<td>☒</td>
</tr>
<tr>
<td>boot.bin</td>
<td>Compulsory</td>
<td>Bootloader provided in SDK.</td>
<td>☒</td>
<td>☑</td>
</tr>
<tr>
<td>user1.bin</td>
<td>Compulsory for first usage</td>
<td>Main program compiled from SDK.</td>
<td>☒</td>
<td>☑</td>
</tr>
<tr>
<td>user2.bin</td>
<td>Used in firmware upgrade</td>
<td>Main program compiled from SDK.</td>
<td>☒</td>
<td>☑</td>
</tr>
</tbody>
</table>

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 Maps".
Users may directly input commands into 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 the ESP-LAUNCHER or the ESP-WROOM-02 is used, users will need either of the hardware mentioned in Table 2-1 below:

### Table 2-1. Hardware Preparations

<table>
<thead>
<tr>
<th>ESP-LAUNCHER</th>
<th>ESP-WROOM-02</th>
</tr>
</thead>
<tbody>
<tr>
<td>- 1 × ESP-LAUNCHER</td>
<td>- 1 × ESP-WROOM-02</td>
</tr>
<tr>
<td>- 1 × USB cable</td>
<td>- 1 × USB-to-TTL converter (FT232R recommended)</td>
</tr>
<tr>
<td>OR</td>
<td>- 6 × Dupont lines</td>
</tr>
<tr>
<td>- 1 × soldering tool suite</td>
<td>- 1 × soldering tool suite</td>
</tr>
</tbody>
</table>

1 × PC with pre-installed Windows OS

⚠ Notice:
The ESP8266 Wi-Fi module needs a 3.3V power supply and may draw 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.

<table>
<thead>
<tr>
<th>Steps</th>
<th>Result</th>
</tr>
</thead>
</table>
| • Slide Power Switch towards the outer side as the figure on the right 🅰️ shows.  
• Slide GPIO0 Control towards the inner side to enable ESP-LAUNCHER’s download mode.  
⚠ Notice:  
J82 must be shorted by a jumper, otherwise code cannot be downloaded to the board. |  

⚠ Notice:
J82 must be shorted by a jumper, otherwise code cannot be downloaded to the board.
2. Preparing the Hardware

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 recognized by the PC.

4. Power on ESP-LAUNCHER by sliding the Power Switch towards the inner side.
5. Power on the chip by sliding the Chip Switch towards the outer side.
6. Download firmware to flash with the 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, slide the GPIO0 Control towards the outer side to enable ESP-LAUNCHER’s working mode.
8. Power on the chip again with the Chip Switch and the chip will read and run programs from the flash.

2.2. ESP-WROOM-02

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

   Table 2-2. ESP-WROOM-02 Pins

<table>
<thead>
<tr>
<th>Pin</th>
<th>Pin status</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>Pull up</td>
<td></td>
</tr>
<tr>
<td>3V3</td>
<td>3.3V power supply (VDD)</td>
<td></td>
</tr>
<tr>
<td>IO15</td>
<td>Pull down</td>
<td></td>
</tr>
<tr>
<td>IO0</td>
<td>UART download: pull down; Flash boot: floating/pull up</td>
<td></td>
</tr>
<tr>
<td>GND</td>
<td>GND</td>
<td></td>
</tr>
<tr>
<td>RXD</td>
<td>Receive-end in UART download</td>
<td></td>
</tr>
<tr>
<td>TXD</td>
<td>Transmit-end in UART download; floating/pull up</td>
<td></td>
</tr>
</tbody>
</table>

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

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

4. Download firmware to flash with the ESP8266 DOWNLOAD TOOL.

**Note:**
On how to download firmware, please refer to Chapter 4, "Flash Maps" 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.

---

**Notes:**
- IO0 is an internal pull-up pin.
- For more information on ESP-WROOM-02 hardware, please refer to [ESP-WROOM-02 Datasheet](#).
3. Preparing the Software

3.1. Non-OS SDK


Figure 3-1 shows the directory structure of the non-OS SDK.

![Figure 3-1. Non-OS SDK Directory Structure](image)

- **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 them.
- **ld**: linker scripts. We suggest users not modifying them without any specific reasons.
- **lib**: library files provided in SDK.
- **tools**: tools needed for compiling binaries. Users do not need to modify them.

3.2. RTOS SDK

Users can download RTOS SDK and its application examples from:

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

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

Figure 3-2. RTOS SDK Directory Structure

- **bin**: boot and initialization firmware.
- **documents**: ESP8266_RTOS_SDK files.
- **driver_lib**: sample codes of drivers.
- **examples**: sample codes for Espressif's application programs.
  - **openssl_demo**: sample codes of the openssl API function.
  - **project_template**: sample codes of project templates.
  - **smart_config**: sample codes of SmartConfig.
  - **spiffs_test**: sample codes of the spiffs file system function.
  - **websocket_demo**: sample codes of web socket.
- **include**: header files of ESP8266_RTOS_SDK, including software interfaces and macro functions for users to use.
- **ld**: link files used when compiling; users do not need to modify them.
- **lib**: library file of ESP8266_RTOS_SDK.
- **third_party**: third-party library of Espressif’s open-source codes, currently including free RTOS, JSON, lwIP, mbedTLS, noPoll, OpenSSL, spiffs, and SSL.
- **tools**: tools needed for compiling binaries; users do not need to modify them.
3. Preparing the Software

3.3. ESP8266 Toolkit

3.3.1. Compiler

Please download VirtualBox from: [https://www.virtualbox.org/wiki/Downloads](https://www.virtualbox.org/wiki/Downloads).

**Note:**
Please choose the right version of VirtualBox according to the host machine's OS.

Please download the compiler *ESP8266_lubuntu_20141021.ova* from: [http://downloads.espressif.com/FB/ESP8266_GCC.zip](http://downloads.espressif.com/FB/ESP8266_GCC.zip)

<table>
<thead>
<tr>
<th>Steps</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.</strong> Start Windows OS and install the virtual machine.</td>
<td><img src="image" alt="VirtualBox" /></td>
</tr>
<tr>
<td>• Double-click <em>VirtualBox-5.0.16-105871-Win.exe</em> and install VirtualBox.</td>
<td><strong>Note:</strong> VirtualBox has different versions. We are using Windows V.5.0.16 as an example.</td>
</tr>
<tr>
<td>• Double-click <em>Oracle VM VirtualBox.exe</em> to run the program, and the system will show the main menu 📚.</td>
<td><strong>Tip:</strong> The ESP8266 virtual machine takes up much space (memory). Please reserve enough space for it.</td>
</tr>
<tr>
<td><strong>2.</strong> Import the image file.</td>
<td><img src="image" alt="Image File" /></td>
</tr>
</tbody>
</table>
### 3. Preparing the Software

<table>
<thead>
<tr>
<th>Steps</th>
<th>Results</th>
</tr>
</thead>
</table>
| • Select **File > Import Appliance**, and a dialog box will show up 🚀.  
• Select the image file to import, for example, C:\ESP8266_lubuntu_20141021.ova, and click **Next**.  
• Click **Import** to confirm the settings. | ![Import Virtual Appliance](image) |

3. Create a shared folder.

<table>
<thead>
<tr>
<th>Steps</th>
<th>Results</th>
</tr>
</thead>
</table>
| • Create a new folder named D:\VM\share.  
• Select **Machine > Settings > Shared Folders...**, and a dialog box will show up 🚀.  
• Select the shared folder in **Machine Folders**, for example, D:\VM\share. | ![Shared Folders](image) |

4. Run the virtual machine.
3. Preparing the Software

3.3.2. Firmware Download Tool

Please download the ESP8266 DOWN LOAD TOOL from:


<table>
<thead>
<tr>
<th>Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>After importing, a virtual machine named <strong>ESP8266_lubuntu</strong> shows up 👉.</td>
</tr>
<tr>
<td>Double-click <strong>ESP8266_lubuntu</strong> or <strong>Start</strong> to run the virtual machine.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system shows the ESP8266 virtual machine 👉.</td>
</tr>
<tr>
<td>If a dialog box like the one below 👇 shows up, please enter the password: <strong>espressif</strong>.</td>
</tr>
</tbody>
</table>
This chapter provides the flash maps for OTA firmware and non-OTA firmware in flash memories with a different capacity. Users can modify the map as needed.

Figure 4-1 shows the flash maps for the two different types of firmware.

**Non-FOTA**

![Non-FOTA Flash Map Diagram]

- **System Program**: This area stores the firmware necessary for the system to run.
- **User Data**: If the system data do not take up all the flash memory, the remaining area can be used to store user data. It is recommended that the user reserve at least 12 KB in the user data area to store user parameters.
- **RF_CAL Parameter**: The system automatically stores the calibrated RF parameters in this area.
- **Default RF Parameter**: Download esp_int_data_default.bin in this area to store the default RF parameters.
- **System Parameter Area**: This area stores the system parameters.
- **Boot Data**: It is located in Partition 1 of the FOTA firmware, and stores boot data.

**FOTA**

![FOTA Flash Map Diagram]

**Note:**

For ESP8266 firmware, please refer to Section 1.3, "ESP8266 FW".
4. Flash Maps

4.1. Non-OTA

4.1.1. Flash Map

Users can change the limit by modifying `ESP8266_NONOS_SDK/ld/eagle.app.v6.ld`.

Users can modify the `len` field in `irom0_0_seg`, as shown in Figure 4-2 (red rectangle).

The location of `irom0.text` varies across different versions of SDK. Users must consult the `eagle.app.v6.id` file and ensure that they are downloading `eagle.irom0.text.bin` to the correct offset in the flash. The address in the blue rectangle marks the location of `eagle.irom0.text.bin` in the flash.

![Figure 4-2. Location for irom0.text](image)

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

<table>
<thead>
<tr>
<th>Flash capacity</th>
<th>eagle.flash.bin</th>
<th>eagle.irom0text.bin</th>
<th>User data</th>
<th>len</th>
<th>RF_CAL parameter</th>
<th>Default parameter</th>
<th>System parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>≤ 64</td>
<td>≤ 368</td>
<td>≥ 60</td>
<td>0x5C000</td>
<td>4</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>1024</td>
<td>≤ 64</td>
<td>≤ 752</td>
<td>≥ 176</td>
<td>0xBC000</td>
<td>4</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>2048</td>
<td>≤ 64</td>
<td>≤ 768</td>
<td>≥ 176</td>
<td>0xC0000</td>
<td>4</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>4096</td>
<td>≤ 64</td>
<td>≤ 768</td>
<td>≥ 176</td>
<td>0xC0000</td>
<td>4</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>8192</td>
<td>≤ 64</td>
<td>≤ 768</td>
<td>≥ 176</td>
<td>0xC0000</td>
<td>4</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>16*1024</td>
<td>≤ 64</td>
<td>≤ 768</td>
<td>≥ 176</td>
<td>0xC0000</td>
<td>4</td>
<td>4</td>
<td>12</td>
</tr>
</tbody>
</table>

**Note:**

ESP8266 presently only supports a System Param area of up to 1024 KB.
4. Flash Maps

4.1.2. Download Addresses

Table 4-2 lists the download addresses for non-OTA firmware.

<table>
<thead>
<tr>
<th>Binaries</th>
<th>Download addresses in flash with different capacities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>512</td>
</tr>
<tr>
<td>blank.bin</td>
<td>0x7B000</td>
</tr>
<tr>
<td>esp_init_data_default.bin</td>
<td>0x7C000</td>
</tr>
<tr>
<td>blank.bin</td>
<td>0x7E000</td>
</tr>
<tr>
<td>eagle.flash.bin</td>
<td></td>
</tr>
<tr>
<td>eagle.irom0text.bin</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- In general, ESP Flash Download Tool can be used to download firmware into flash. It is recommended that the user start from the lower bits when downloading the firmware.
- But for 8 MB or 16 MB flash, please use esptool instead.

4.2. OTA Firmware

4.2.1. Flash Map

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

<table>
<thead>
<tr>
<th>Flash capacity</th>
<th>boot</th>
<th>user1.bin</th>
<th>user2.bin</th>
<th>RF_CAL parameter</th>
<th>Default parameter</th>
<th>System parameter</th>
<th>User data</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>4</td>
<td>≤ 232</td>
<td>≤ 232</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>≥ 0</td>
</tr>
<tr>
<td>1024</td>
<td>4</td>
<td>≤ 488</td>
<td>≤ 488</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>≥ 0</td>
</tr>
<tr>
<td>2048 (Partition 1 = 512)</td>
<td>4</td>
<td>≤ 488</td>
<td>≤ 488</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>≥ 1024</td>
</tr>
<tr>
<td>2048 (Partition 1 = 1024)</td>
<td>4</td>
<td>≤ 1000</td>
<td>≤ 1000</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>≥ 0</td>
</tr>
<tr>
<td>4096 (Partition 1 = 512)</td>
<td>4</td>
<td>≤ 488</td>
<td>≤ 488</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>≥ 3072</td>
</tr>
<tr>
<td>4096 (Partition 1 = 1024)</td>
<td>4</td>
<td>≤ 1000</td>
<td>≤ 1000</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>≥ 2048</td>
</tr>
</tbody>
</table>
# 4. Flash Maps

## 4.2.2. Download Addresses

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

<table>
<thead>
<tr>
<th>Flash capacity</th>
<th>boot</th>
<th>user1.bin</th>
<th>user2.bin</th>
<th>RF_CAL parameter</th>
<th>Default parameter</th>
<th>System parameter</th>
<th>User data</th>
</tr>
</thead>
<tbody>
<tr>
<td>8192 (Partition 1 = 1024)</td>
<td>4</td>
<td>≤ 1000</td>
<td>≤ 1000</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>≥ 6144</td>
</tr>
<tr>
<td>16384 (Partition 1 = 1024)</td>
<td>4</td>
<td>≤ 1000</td>
<td>≤ 1000</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>≥ 14336</td>
</tr>
</tbody>
</table>

### Table 4-4. Download Addresses for OTA Firmware (unit: KB)

<table>
<thead>
<tr>
<th>Binaries</th>
<th>Download addresses in flash with different capacities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>512</td>
</tr>
<tr>
<td>blank.bin</td>
<td>0x7B000</td>
</tr>
<tr>
<td>esp_init_data_default.bin</td>
<td>0x7C000</td>
</tr>
<tr>
<td>blank.bin</td>
<td>0x7E000</td>
</tr>
<tr>
<td>boot.bin</td>
<td>0x00000</td>
</tr>
<tr>
<td>user1.bin</td>
<td>0x01000</td>
</tr>
<tr>
<td>user2.bin</td>
<td>0x41000</td>
</tr>
</tbody>
</table>

### Notes:

- In general, **ESP Flash Download Tool** can be used to download firmware into flash. It is recommended that the user start from the lower bits when downloading the firmware.

- But for 8 MB or 16 MB flash, please use **esptool** instead.

- For OTA firmware, users do not need to download **user2.bin**, but upgrade the firmware via the cloud server.
5. Compiling the SDK

Notes:
- This chapter demonstrates how to compile the SDK by taking ESP8266_NONOS_SDK/examples/IoT_Demo as an example.
- IoT_Demo defines three types of devices, i.e., LIGHT_DEVICE, PLUG_DEVICE and SENSOR_DEVICE in examples/IoT_Demo/include/user_config.h. Users can only configure one device at a time. The default device for configuration is LIGHT_DEVICE.

5.1. Preparations

5.1.1. Modifying SDK Files

Note:
Users need to modify the SDK files if using the OTA firmware.

1. Start Windows OS.
2. Modify files in ESP8266_NONOS_SDK/examples/IoT_Demo/include according to the flash map.
   - Modify #define PRIV_PARAM_START_SEC in user_light.h and user_plug.h.

```c
/* 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.

```c
/* 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>
<thead>
<tr>
<th>Default value (512)</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>(512+512)</td>
<td>(1024+1024)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3C</td>
<td>-</td>
<td>0x7C</td>
<td>0x7C</td>
<td>0xFC</td>
<td>0x7C</td>
<td>0xFC</td>
<td>0x7C</td>
<td>0xFC</td>
<td>0x7C</td>
<td>0xFC</td>
<td>0x7C</td>
<td>0xFC</td>
<td>0x7C</td>
<td>0xFC</td>
<td>0x7C</td>
<td>0xFC</td>
<td>0x7C</td>
<td></td>
</tr>
<tr>
<td>0x3D</td>
<td>-</td>
<td>0x7D</td>
<td>0x7D</td>
<td>0xFD</td>
<td>0x7D</td>
<td>0xFD</td>
<td>0x7D</td>
<td>0xFD</td>
<td>0x7D</td>
<td>0xFD</td>
<td>0x7D</td>
<td>0xFD</td>
<td>0x7D</td>
<td>0xFD</td>
<td>0x7D</td>
<td>0xFD</td>
<td>0x7D</td>
<td></td>
</tr>
</tbody>
</table>

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

5.1.2. Downloading SDK Files

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

<table>
<thead>
<tr>
<th>Steps</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Copy ESP8266_NONOS_SDK folder to the shared directory, for example, C:VM\share.</td>
<td>![Image of ESP8266_NONOS_SDK directory]</td>
</tr>
<tr>
<td>• Copy IoT_Demo folder to C:VM\share\ESP8266_NONOS_SDK, as shown in the figure on the right.</td>
<td>![Image of IoT_Demo directory]</td>
</tr>
</tbody>
</table>

4. Download shared directory.

<table>
<thead>
<tr>
<th>Steps</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Execute ./mount.sh.</td>
<td>Downloading shared files is completed.</td>
</tr>
<tr>
<td>• Input the password: espressif.</td>
<td>![Image of terminal input]</td>
</tr>
<tr>
<td>• Open the shared directory ESP8266_NONOS_SDK in the virtual machine and confirm whether the download has been successful.</td>
<td>![Image of ESP8266_NONOS_SDK directory in terminal]</td>
</tr>
<tr>
<td>- If successful, the directory contains such files as those in the figure on the right.</td>
<td>![Image of ESP8266_NONOS_SDK directory]</td>
</tr>
<tr>
<td>- If not, the directory will be empty, and users will need to go over this step again.</td>
<td>![Image of empty directory]</td>
</tr>
</tbody>
</table>

⚠️ Notice:

If users use the RTOS SDK, please continue with the following steps; if use the non-OS SDK, please skip Step 5.

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

```bash
export SDK_PATH=~/Share/ESP8266_RTOS_SDK
export BIN_PATH=~/Share/ESP8266_RTOS_SDK/bin
```
5. Compiling the SDK

5.2. Compilation

5.2.1. Compile ESP8266_NONOS_SDK_v0.9.5 and Later Versions

1. Switch to the `/Share/ESP8266_NONOS_SDK/IoT_Demo` directory in the terminal.

   ```
   cd /home/esp8266/Share/ESP8266_NONOS_SDK/IoT_Demo
   ./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](image)

   **Note:** Users can add it to `.bashrc` file, otherwise Step 5 needs to be repeated each time the compiler is restarted.
5. Compiling the SDK

### Notes:
- The sample options are marked in green. Users can select the right options as needed.
- For OTA and non-OTA firmware, please refer to Section 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 clear the temporary files generated by the last compilation, and then compile user2.bin.
- For the flash map in Step 5, please refer to Chapter 4, "Flash Maps".

3. After compilation, the generated binaries 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:**
Users can open the `/home/esp8266/Share/ESP8266_NONOS_SDK/bin` directory and check the compiled binaries.

5.2.2. ESP8266_NONOS_SDK_v0.9.4 and Earlier Versions

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 clear 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-OTA firmware.
6. Downloading the Firmware

6.1. Download Procedure

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

![ESP8266 DOWNLOAD TOOL—SPIDownload](image)

**Figure 6-1. ESP8266 DOWNLOAD TOOL—SPIDownload**

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPIDownload</td>
<td>For SPI Flash download.</td>
</tr>
<tr>
<td>HSPIDownload</td>
<td>For HSPI Flash download.</td>
</tr>
<tr>
<td>RFConfig</td>
<td>RF initialization Configuration.</td>
</tr>
<tr>
<td>MutiDownload</td>
<td>For multi-mother boards download.</td>
</tr>
</tbody>
</table>
3. Double-click ![ Download Path Config ] 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 downloaded and the corresponding addresses vary with different SPI Flash sizes and actual demands. For details, please refer to Chapter 4, “Flash Maps”.

### Table 6-1. SPIDownload Configuration

<table>
<thead>
<tr>
<th>Items</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI FLASH CONFIG</td>
<td></td>
</tr>
<tr>
<td>CrystalFreq</td>
<td>Select the crystal frequency according to the crystal oscillator used.</td>
</tr>
<tr>
<td>CombineBin</td>
<td>Combine the selected binaries into target.bin with the address 0x0000.</td>
</tr>
<tr>
<td>Default</td>
<td>Set the SPI Flash to the default value.</td>
</tr>
<tr>
<td>SPI SPEED</td>
<td>Select SPI read/write speed with the maximum value of 80 MHz.</td>
</tr>
<tr>
<td>SPI MODE</td>
<td>Select SPI mode according to the SPI Flash used. If the flash is Dual SPI, select <strong>DIO</strong> or <strong>DOUT</strong>. If the flash is Quad SPI, select <strong>DIO</strong> or <strong>DOUT</strong>.</td>
</tr>
<tr>
<td>![ Notice: ]</td>
<td>If ISSI Flash is used, please refer to Appendix, “Configure ISSI &amp; MXIC Flash QIO Mode”.</td>
</tr>
<tr>
<td>FLASH SIZE</td>
<td>Select the flash size according to the flash type.</td>
</tr>
<tr>
<td>![ Note: ]</td>
<td><strong>16Mbit-C1</strong> refers to 1024+1024 flash map and <strong>32Mbit-C1</strong> 1024+1024 flash map as well.</td>
</tr>
<tr>
<td>SpiAutoSet</td>
<td>We recommend not checking <strong>SpiAutoSet</strong>, but configuring the flash manually as needed. If users select <strong>SpiAutoSet</strong>, 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.</td>
</tr>
<tr>
<td>DoNotChgBin</td>
<td>• If users select <strong>DoNotChgBin</strong>, the flash working frequency, mode, and flash map will be based on the configuration when compiling. • If users do not select <strong>DoNotChgBin</strong>, the flash working frequency, mode, and flash map will be defined by the final configuration of the compiler.</td>
</tr>
<tr>
<td>Download Panel</td>
<td></td>
</tr>
<tr>
<td>START</td>
<td>Click <strong>START</strong> to start download. When the download completes, <strong>FINISH</strong> will appear in the green area on the left.</td>
</tr>
<tr>
<td>STOP</td>
<td>Click <strong>STOP</strong> to stop download.</td>
</tr>
<tr>
<td>MAC Address</td>
<td>If download is successful, the system will show the MAC addresses of ESP8266 STA and ESP8266 AP.</td>
</tr>
<tr>
<td>COM PORT</td>
<td>Select the actual COM port of ESP8266.</td>
</tr>
</tbody>
</table>
5. After downloading, turn GPIO0 Control on ESP-LAUNCHER to the outer side and power the board on to enable the working mode.

6.2. Check Log File

After downloading firmware, users can check the log printed in the terminal by using the serial port debug tool.

Users need to configure the settings of the serial port debug tool, as follows:

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

6.2.1. ESP8266 IOT Demo

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

```
SDK version: X.X.X(e67da894)
IOT VERSION = v1.0.5t45772(a)
reset reason: 0
PWM version: 00000003
```
6. Downloading the Firmware

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 download the ESP8266 AT firmware, or the default firmware in ESP-LAUNCHER or ESP-WROOM-02, the system in working mode will display “Ready” at the end. Input command “AT” in the terminal and the system will return “OK”, which means that the firmware works properly.

Notes:

- The baud rate in AT firmware is configured as 115200 manually, however, the default baud rate of ESP8266 is 74880, due to this discrepancy, the system initialization information will be displayed as mojibake. It is a normal phenomenon as long as the system shows “Ready” at the end.
- For more information on AT commands, please refer to ESP8266 AT Instruction Set.
A. Appendix—Configuring Flash DIO to 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.

For some 2-line flash (for example, the flash manufactured by ISSI or MXIC), users can configure the flash from the DIO mode to the QIO mode by modifying the first two bytes in `blank.bin`, as instructed in Table A-1. When booting up, ESP8266 will check the first two bytes of the `blank.bin` file and switch to the QIO mode automatically if the condition set forth in Table A-1 is met.

```c
struct 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>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without secondary boot loader</td>
<td>Modify <code>to_qio</code> to 0.</td>
</tr>
</tbody>
</table>
| With secondary boot loader      | Modify `user_bin` to 0 and `to_qio` to 0, as well. Modify `version` according to the current boot version. **Example:** If users use the secondary `boot_v1.5.bin`, please modify the first two bytes FF FF to F4 E5.

Table A-1. blank.bin Configuration
Appendix—Learning Resources

B.1. Must-Read Documents

- **ESP8266EX Datasheet**
  Description: This document introduces the specifications of ESP8266EX, including an overview of the features, protocols, technical parameters and applications. It also describes the pin layout, as well as major functional modules integrated in ESP8266EX (CPU, flash and memory, clock, radio, Wi-Fi, and low-power management). Additionally, it provides descriptions of peripheral interfaces integrated on ESP8266EX, lists the electrical data of ESP8266EX and illustrates the package details of ESP8266EX.

- **ESP8266 AT Command Examples**
  Description: This document introduces some specific examples of how to use Espressif AT commands, including single connection as a TCP client, UDP transmission and transparent transmission, and multiple connection as a TCP server.

- **ESP8266 AT Instruction Set**
  Description: This document provides lists of AT commands based on ESP8266_NONOS_SDK, including user-defined AT commands, basic AT commands, Wi-Fi AT commands and TCP/IP-related AT commands. It also introduces the downloading of AT firmware into flash.

- **ESP8266 Non-OS SDK API Reference**
  Description: This document lists ESP8266_NONOS_SDK APIs, provides an overview of ESP8266_NONOS_SDK and introduces the readers to system APIs, TCP/UDP APIs, mesh APIs, application specific APIs, definitions and data structures, and APIs for peripheral interfacing.

- **FAQ**

B.2. Must-Have Resources

- **ESP8266 SDKs**
  Description: This webpage provides links to the latest version of ESP8266 SDK and the older ones.

- **Non-OS Sample Code**
  Description: This webpage provides the sample code for the commonly used functions.

- **ESP8266 Tools**
  Description: This webpage provides links to the ESP8266 flash download tools and ESP8266 performance evaluation tools.
• ESP8266 Certification and Test Guide
• ESP8266 BBS
• ESP8266 Resources