

# ESP8266 SDK

## 入门指南



版本 2.0

版权 © 2016



# 关于本手册

本文以 ESP-LAUNCHER 和 ESP-WROOM-02 为例，介绍 ESP8266 SDK 相关使用方法，包括编译前的准备、SDK 的编译和固件的下载。

本手册结构如下：

章	标题	内容
第 1 章	概述	介绍 SDK 使用的整体流程，并给出 ESP8266 HDK、SDK、FW 和工具集的简单介绍。
第 2 章	硬件准备	介绍 SDK 使用需要的硬件，并给出开发板和模组两种方案。
第 3 章	软件准备	介绍 Non-OS 和 RTOS 两种版本的 SDK 及编译 SDK 和下载固件的相关工具。
第 4 章	Flash 布局	介绍固件下载到 Flash 中的布局和地址，区分 FOTA 和 Non-FOTA 固件。
第 5 章	编译 SDK	介绍如何使用编译工具来编译 SDK。
第 6 章	下载固件	介绍如何使用下载工具来下载固件。
附录 I	配置 ISSI Flash QIO 模式	介绍对 ISSI Flash 的支持和配置方式。

## 发布说明

日期	版本	发布说明
2016.04	V2.0	首次发布。

## 相关手册

请在如下链接下载相关手册。

官网：<http://www.espressif.com/support/download/documents>

官方论坛：<http://bbs.espressif.com/viewtopic.php?f=67&t=225>

文档类型	文档名称
硬件指南	《ESP8266 硬件描述》
	《ESP-WROOM-02 技术规格表》
开发指南	《ESP8266 Non-OS SDK 升级指南》
	《ESP8266 Non-OS SDK AT 指令集》



# 目录

---

1. 概述 .....	1
1.1. 流程概览 .....	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 工具集 .....	3
1.5.1. 编译器 .....	3
1.5.2. 固件下载工具 .....	3
1.5.3. 串口调试工具 .....	3
2. 硬件准备 .....	5
2.1. 开发板方案 .....	5
2.2. 模组方案 .....	6
3. 软件准备 .....	8
3.1. Non-OS SDK .....	8
3.2. RTOS SDK .....	9
3.3. ESP8266 工具集 .....	10
3.3.1. 编译器 .....	10
3.3.2. 固件下载工具 .....	13
4. Flash 布局 .....	14
4.1. Non-FOTA .....	15
4.1.1. 布局说明 .....	15
4.1.2. 下载地址 .....	16
4.2. FOTA .....	16
4.2.1. 布局说明 .....	16

4.2.2. 下载地址.....	17
5. 编译 SDK .....	18
5.1. 编译准备 .....	18
5.1.1. 修改 SDK 文件 .....	18
5.1.2. 加载 SDK 文件 .....	19
5.2. 开始编译.....	20
5.2.1. ESP8266_NONOS_SDK_v0.9.5 及之后版本.....	20
5.2.2. ESP8266_NONOS_SDK_v0.9.4 及之前版本.....	22
6. 下载固件 .....	23
6.1. 下载步骤.....	23
6.2. 查看打印信息.....	25
6.2.1. ESP8266 IOT Demo.....	25
6.2.2. ESP8266 AT .....	26
6.3. 射频初始化设置（可选） .....	26
6.3.1. RF InitConfig 选项 .....	27
6.3.2. RF InitConfig 参数 .....	27
6.3.3. 设置举例.....	29
I. 附录-配置 ISSI Flash QIO 模式 .....	31



# 1.

# 概述

## 1.1. 流程概览

SDK 的使用流程如图 1-1 所示。

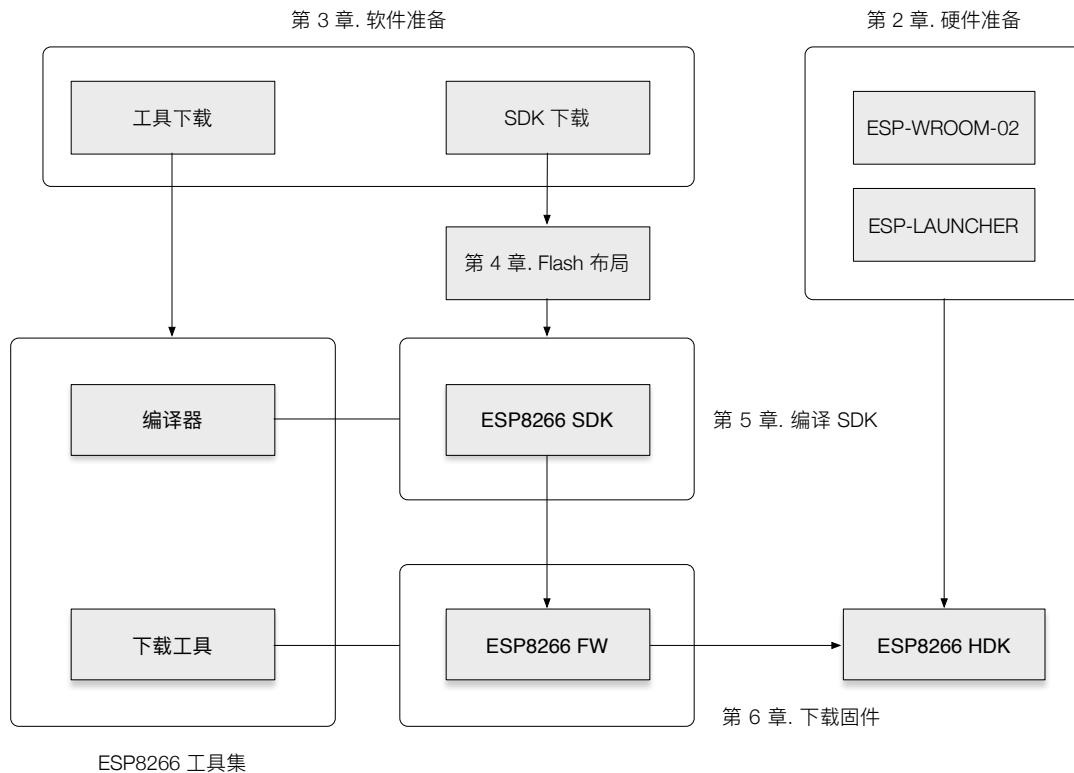


图 1-1 流程概览

## 1.2. ESP8266 HDK

ESP8266 HDK (Hardware Development Kits) 包括芯片 ESP8266EX、模组 ESP-WROOM-02 和开发板 ESP-LAUNCHER 等。您可以使用乐鑫官方的 ESP-WROOM-02 或 ESP-LAUNCHER 下载编译好的固件。

### 说明:

- 如要使用其他集成 ESP8266EX 的开发板或者模组，请使用相应厂商提供的开发固件。
- 如需购买 ESP-WROOM-02 或 ESP-LAUNCHER，请访问乐鑫网上商店：  
<https://espressif.taobao.com>。
- 关于更多硬件信息，请登录：
  - 乐鑫官网：<http://www.espressif.com/support/download/documents>
  - 乐鑫官方论坛：<http://bbs.espressif.com/viewtopic.php?f=21&t=412&p=1545#p1545>。



## 1.3. ESP8266 SDK

ESP8266 SDK (Software Development Kits) 是乐鑫为开发者提供的 IOT 应用开发平台，包括基础平台以及上层应用开发示例，如智能灯、智能开关等。

SDK 的基础平台按照是否基于操作系统可分为：Non-OS 和 RTOS 两种版本。

### 1.3.1. Non-OS SDK

Non-OS SDK 是不基于操作系统的 SDK，提供 IOT\_Demo 和 AT 的编译。Non-OS SDK 主要使用定时器和回调函数的方式实现各个功能事件的嵌套，达到特定条件下触发特定功能函数的目的。Non-OS SDK 使用 espconn 接口实现网络操作，用户需要按照 espconn 接口的使用规则进行软件开发。

### 1.3.2. RTOS SDK

RTOS SDK 基于 FreeRTOS，在 Github 上开源。

- RTOS 版本 SDK 使用 FreeRTOS 系统，引入 OS 多任务处理的机制，用户可以使用 FreeRTOS 的标准接口实现资源管理、循环操作、任务内延时、任务间信息传递和同步等面向任务流程的设计方式。具体接口使用方法参考 FreeRTOS 官方网站的使用说明或者 USING THE FreeRTOS REAL TIME KERNEL--A Practical Guide 介绍。
- RTOS 版本 SDK 的网络操作接口是标准 lwIP API，同时提供了 BSD Socket API 接口的封装实现，用户可以直接按照 Socket API 的使用方式来开发软件应用，也可以直接编译运行其他平台的标准 Socket 应用，有效降低平台切换的学习成本。
- RTOS 版本 SDK 引入了 cJSON 库，使用该库函数可以更加方便的实现对 JSON 数据包的解析。
- RTOS 版本兼容 Non-OS SDK 中的 Wi-Fi 接口、Smart Config 接口、Sniffer 相关接口、系统接口、定时器接口、FOTA 接口和外围驱动接口，不支持 AT 实现。

## 1.4. ESP8266 FW

ESP8266 FW (Firmware) 是一些可直接下载到 ESP8266 HDK 中的 BIN 文件，您可以选择下载 FOTA (Firmware Over-The-Air 支持云端升级) 和 Non-FOTA (不支持云端升级) 的 BIN 文件，具体如表 1-1 所示。

表 1-1. ESP8266 FW

文件列表	是否必选	说明	Non-FOTA	FOTA
master_device_key.bin	可选	用户在 Espressif Cloud 申请，依此享受乐鑫云端服务。	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>



文件列表	是否必选	说明	Non-FOTA	FOTA
esp_init_data_default.bin	必选	初始化射频参数，在 SDK 根目录中提供。	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
blank.bin	必选	初始化系统参数，在 SDK 根目录中提供。	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
eagle.flash.bin	必选	主程序，编译代码生成。	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
eagle.irom0text.bin	必选	主程序，编译代码生成。	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
user1.bin	初次使用必选	主程序，编译代码生成。	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
user2.bin	升级时使用	主程序，编译代码生成。	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

#### 说明：

- 关于 SDK 的内容介绍，请参考“第 3 章 软件准备”。
- 关于 SDK 的编译，请参考“第 5 章 编译 SDK”。
- 关于 BIN 文件在 Flash 中的地址，请参考“第 4 章 Flash 布局”。

## 1.5. ESP8266 工具集

### 1.5.1. 编译器

编译 ESP8266 SDK 需要使用 Linux 操作系统，若您使用 Windows 操作系统，建议您使用 VirtualBox 作为 ESP8266 虚拟机。为了简化您的编译操作，乐鑫已将编译 SDK 所需要的工具安装到虚拟机中。您只需安装虚拟机，并导入 ESP8266 编译器（OVA 镜像文件）即可直接编译 ESP8266 SDK。

### 1.5.2. 固件下载工具

ESP8266 DOWNLOAD TOOL 工具是由乐鑫官方开发的固件下载工具，用户可根据实际的编译方式和 Flash 的容量，将多个 BIN 文件一键下载到 ESP8266 母板（开发板或者模组）的 SPI Flash 中。

### 1.5.3. 串口调试工具

串口调试工具可以通过标准 RS-232 端口直接与 ESP8266 建立通信。对于不带有物理串口的 PC，可以使用 USB 转串口模块来虚拟出一个串口设备。

用户可以直接在串口终端输入命令和实时查看相关打印信息。



## 说明:

建议使用 *CoolTerm* (*Windows* 和 *Mac* 系统) 和 *Minicom* (*Linux* 系统) 作为串口调试工具。



## 2.

# 硬件准备

您可以选择使用如表 2-1 所示的两种方案中的任意一种。

表 2-1. 硬件准备

开发板方案	模组方案
<ul style="list-style-type: none"><li>• 1 个 ESP-LAUNCHER</li><li>• 1 根 USB 数据线</li></ul>	<ul style="list-style-type: none"><li>• 1 个 ESP-WROOM-02</li><li>• 1 个 USB 转 TTL 串口模块（推荐 FT232R）</li><li>• 6 根杜邦线</li><li>• 1 套焊接工具</li></ul>
或者	
预装 Windows 系统的 PC 机	

**⚠ 注意：**

ESP8266 Wi-Fi 模块需要保证 3.3 V 电源和最少 500 mA 的电流。

## 2.1. 开发板方案

1. 用 USB 数据线将 PC 机与 ESP-LAUNCHER 的 USB-UART 接口相连。
2. 将开发板置为下载模式。

步骤	结果
<ul style="list-style-type: none"><li>• 如右图  所示，将电源开关 (Power Switch) 拨到外侧。</li><li>• 将 GPIO0 开关 (GPIO0 Control) 拨到内侧将 ESP-LAUNCHER 开发板置为下载模式。</li></ul> <p><b>⚠ 注意：</b> ESP-LAUNCHER 上的 J82 跳针需要用跳线帽短接，否则无法下载。</p>	

3. 将 USB 转 TTL 串口模块与 PC 机连接。

**说明：**

请安装正确的、可被 PC 识别的 USB 转 TTL 串口模块驱动。

4. 将电源开关 (Power Switch) 拨到内侧给开发板上电。
5. 将芯片开关 (Chip Switch) 拨到外侧给芯片上电。
6. 通过下载工具 (ESP8266 DOWNLOAD TOOL) 将固件下载到 Flash 中。

**说明：**

关于如何下载固件，请参考“第 4 章 Flash 布局”和“第 6 章 下载固件”。

7. 下载完毕后将 GPIO0 开关 (GPIO0 Control) 拨到外侧将 ESP-LAUNCHER 开发板置为工作模式。
8. 使用芯片开关 (Chip Switch) 给芯片重新上电，芯片初始化时会从 Flash 中读取程序运行。

— — END

更多 ESP-LAUNCHER 的硬件信息，请参考《ESP8266 硬件描述》。

## 2.2. 模组方案

1. 将 ESP-WROOM-02 的如表 2-2 所示的管脚引出。

表 2-2. ESP-WROOM-02 的管脚

管脚名称	管脚状态	实物图
EN	上拉	
3V3	上拉	
IO15	下拉	
IO0	下载模式：下拉； 工作模式：悬空 / 上拉	
GND	GND	
RXD	上拉	
TXD	上拉	

2. 按照图 2-1 用杜邦线将 ESP-WROOM-02 和 USB 转 TTL 串口模块连接。

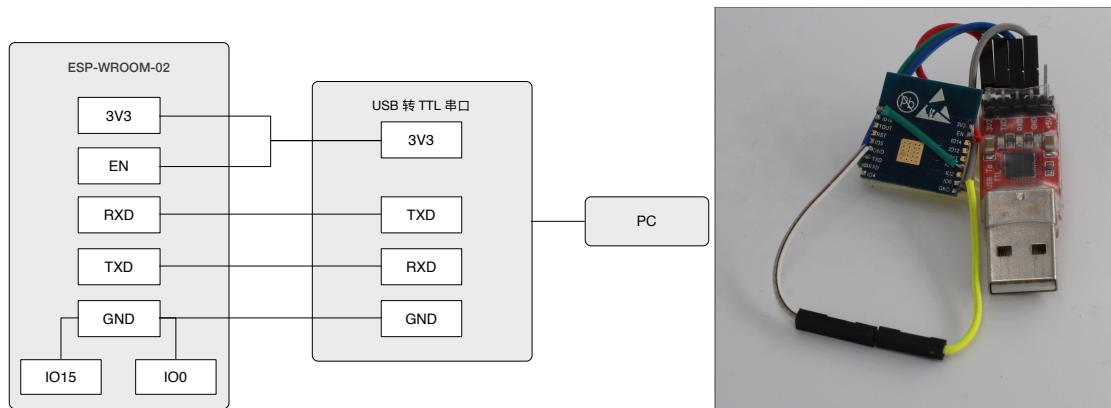


图 2-1. ESP-WROOM-02 下载模式

3. 将 USB 转 TTL 串口模块与 PC 机连接。
4. 通过下载软件 (ESP8266 DOWNLOAD TOOL) 将固件下载到 Flash 中。

**说明:**

关于如何下载固件, 请参考“第 4 章 Flash 布局”和“第 6 章 下载固件”。

5. 下载完毕后将 ESP-WROOM-02 切换为工作模式。  
将 IO0 悬空或者上拉。
6. 重新上电, 芯片初始化时会从 Flash 中读取程序运行。

— — END

**说明:**

I00 管脚为内置高电平, 更多 ESP-WROOM-02 的硬件信息, 请参考《ESP8266 硬件描述》和《ESP-WROOM-02 技术规格表》。



# 3.

# 软件准备

## 3.1. Non-OS SDK

您可以在如下路径下载 Non-OS SDK（包括应用示例）。

- 官网  
<http://www.espressif.com/en/support/download/sdks-demos>
- 官方论坛  
<http://bbs.espressif.com/viewtopic.php?f=46&t=851>

Non-OS SDK 软件包中的内容如图 3-1 所示。

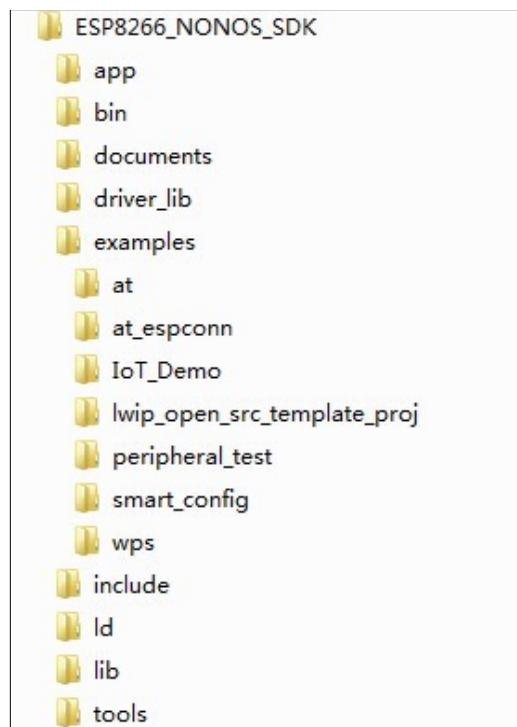


图 3-1. Non-OS SDK 内容

- **app**: 用户工作主目录，用户级代码及头文件均放在此目录下编译。
- **bin**: 编译生成的 BIN 文件，可直接下载到 Flash 中。
- **documents**: SDK 相关的文档或链接。
- **driver\_lib**: 外设驱动的库文件，如：UART、I2C 和 GPIO 等。
- **examples**: 可供用户二次开发的示例代码，如 IoT Demo 等。
- **include**: SDK 自带头文件，包含了用户可使用的相关 API 函数及其他宏定义，用户无需修改。



- *Id*: 链接时所需的脚本文件，若无特殊需求，用户无需修改。
- *lib*: SDK 提供的库文件。
- *tools*: 编译 BIN 文件所需的工具，用户无需修改。

## 3.2. RTOS SDK

您可以在如下路径下载 SDK 及其应用示例（ESP8266\_IOT\_PLATFORM）。

- RTOS SDK  
[https://github.com/espressif/ESP8266\\_RTOS\\_SDK](https://github.com/espressif/ESP8266_RTOS_SDK)
- ESP8266\_IOT\_PLATFORM  
[https://github.com/espressif/ESP8266\\_IOT\\_PLATFORM](https://github.com/espressif/ESP8266_IOT_PLATFORM)

RTOS SDK 软件包中的内容如图 3-2 所示。

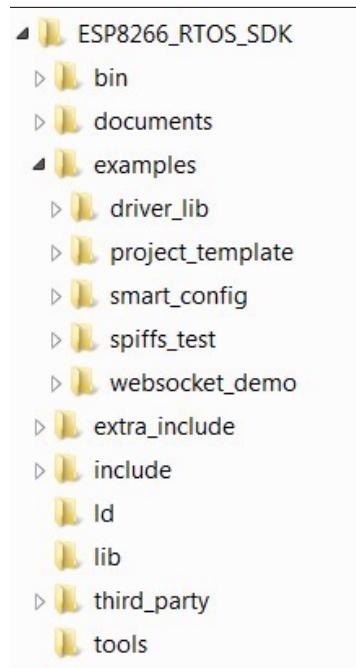


图 3-2. RTOS SDK 内容

- *bin*: 编译生成的 BIN 文件，可直接下载到 Flash 中。
- *documents*: SDK 相关的文档或链接。
- *examples*: 可供用户二次开发的示例代码。
  - *examples/driver\_lib*: 外设驱动的库文件，如：UART、I2C 和 GPIO 等。
  - *examples/project\_template*: 用户工程目录的模版。

### 说明：

您可以将 *project\_template* 文件夹复制到任意目录，如：`~/workspace`。



- **examples/smart\_config**: Smart Config 相关的代码示例。
- **examples/spiffs\_test**: SPIFFS 相关的代码示例。
- **examples/websocket\_demo**: WebSocket 相关的代码示例。
- **extra\_include**: Xtensa 提供的相关头文件。
- **include**: SDK 自带头文件，包含了用户可使用的相关 API 函数及其他宏定义，用户无需修改。
- **ld**: 链接时所需的脚本文件，如无特殊需求，用户无需修改。
- **lib**: SDK 提供的库文件。
- **third\_party**: 第三方开源的库文件。
- **tools**: 编译 BIN 文件所需的工具，用户无需修改。

### 3.3. ESP8266 工具集

#### 3.3.1. 编译器

请在如下链接中下载 VirtualBox。

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



说明：

请根据您的主机配置选择合适的 VirtualBox 版本。

请在如下链接中下载编译器 **ESP8266\_Lubuntu\_20141021.ova**。

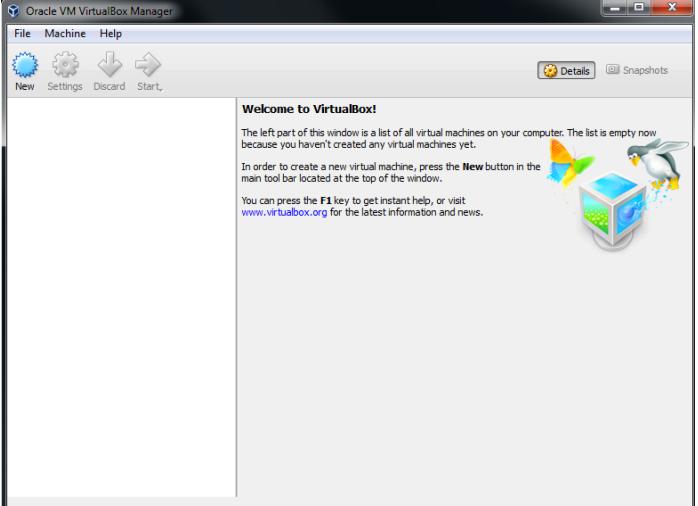
百度：<https://pan.baidu.com/share/init?shareid=3541602653&uk=190196792&third=15>

下载密码：qudl

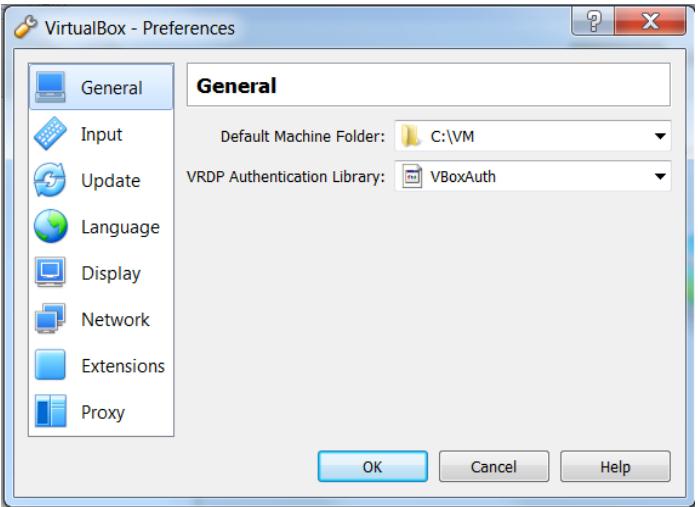
Google：<https://drive.google.com/folderview?id=0B5bwBE9A5dBXaExvdDExVFNrUXM&usp=sharing>

步骤	结果
1. 进入 Windows 系统安装虚拟机。	



步骤	结果
<ul style="list-style-type: none"><li>双击 <b>VirtualBox-5.0.16-105871-Win.exe</b> 文件并按照提示安装虚拟机。</li></ul> <p><b>说明：</b> VirtualBox 有不同的版本，本手册以 Windows V.5.0.16 版本为例。</p> <ul style="list-style-type: none"><li>双击 <b>Oracle VM VirtualBox.exe</b> 运行虚拟机程序，系统显示如右图 手所示主菜单。</li></ul> <p><b>提示：</b> ESP8266 虚拟机会占用较大空间，请预留足够的空间。</p>	

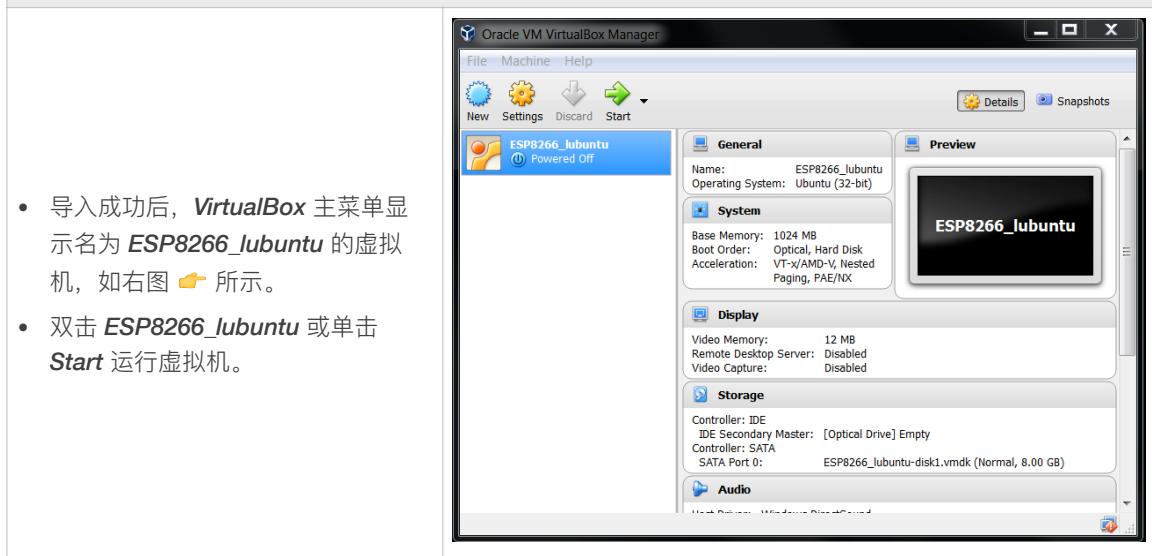
## 2. 设置虚拟机文件夹。

<ul style="list-style-type: none"><li>新建文件夹，如: C:\VM。</li><li>在主菜单中选择 <b>File &gt; Preferences</b>，系统进入如右图 手所示对话框。</li><li>选择 <b>General</b>，在 <b>Default Machine Folder</b> 中设置虚拟机的位置，如: C:\VM。</li></ul>	
--	---

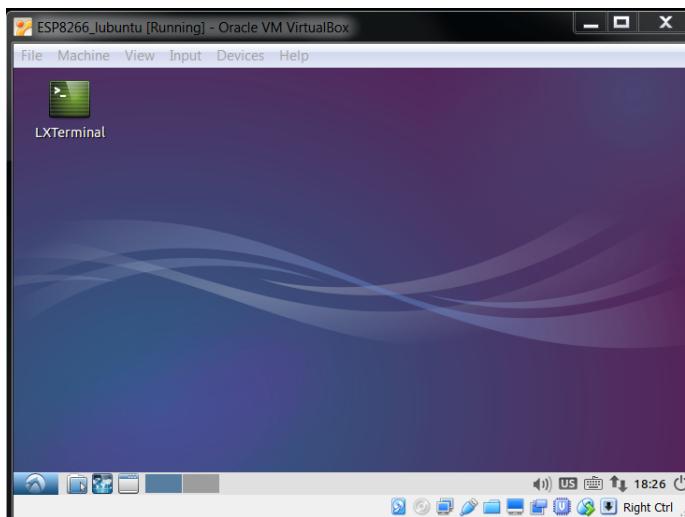
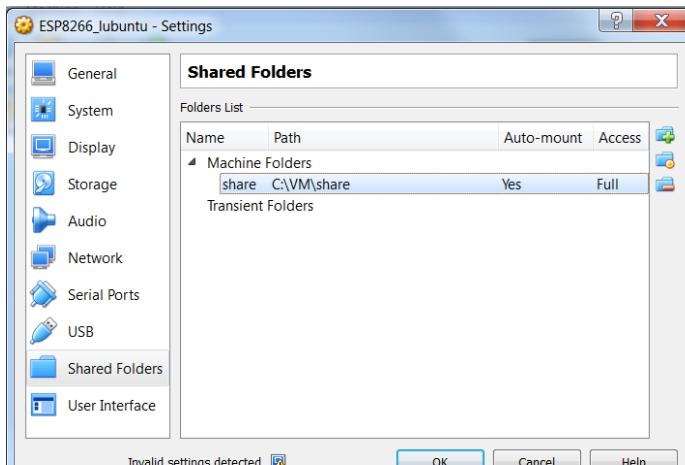
## 3. 导入虚拟机镜像文件。



#### 4. 运行虚拟机。





步骤	结果								
<ul style="list-style-type: none"><li>系统显示 ESP8266 虚拟机，如右图  所示。</li><li>若系统显示如下图  所示锁定对话框，请输入解锁密码：<b>espressif</b>。</li></ul> 									
<p>5. 设置虚拟机共享文件夹。</p>  <p>The screenshot shows the 'Shared Folders' tab in the 'Machine' settings of Oracle VM VirtualBox. It displays a table with one entry:</p> <table border="1"><thead><tr><th>Name</th><th>Path</th><th>Auto-mount</th><th>Access</th></tr></thead><tbody><tr><td>share</td><td>C:\VM\share</td><td>Yes</td><td>Full</td></tr></tbody></table>		Name	Path	Auto-mount	Access	share	C:\VM\share	Yes	Full
Name	Path	Auto-mount	Access						
share	C:\VM\share	Yes	Full						

### 3.3.2. 固件下载工具

请在如下链接下载 ESP8266 DOWNLOAD TOOL：

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



# 4.

# Flash 布局

本章分别介绍 Non-FOTA 与 FOTA 固件在不同容量 Flash 中的布局。您可以根据实际情况修改。

Flash 布局如图 4-1 所示。

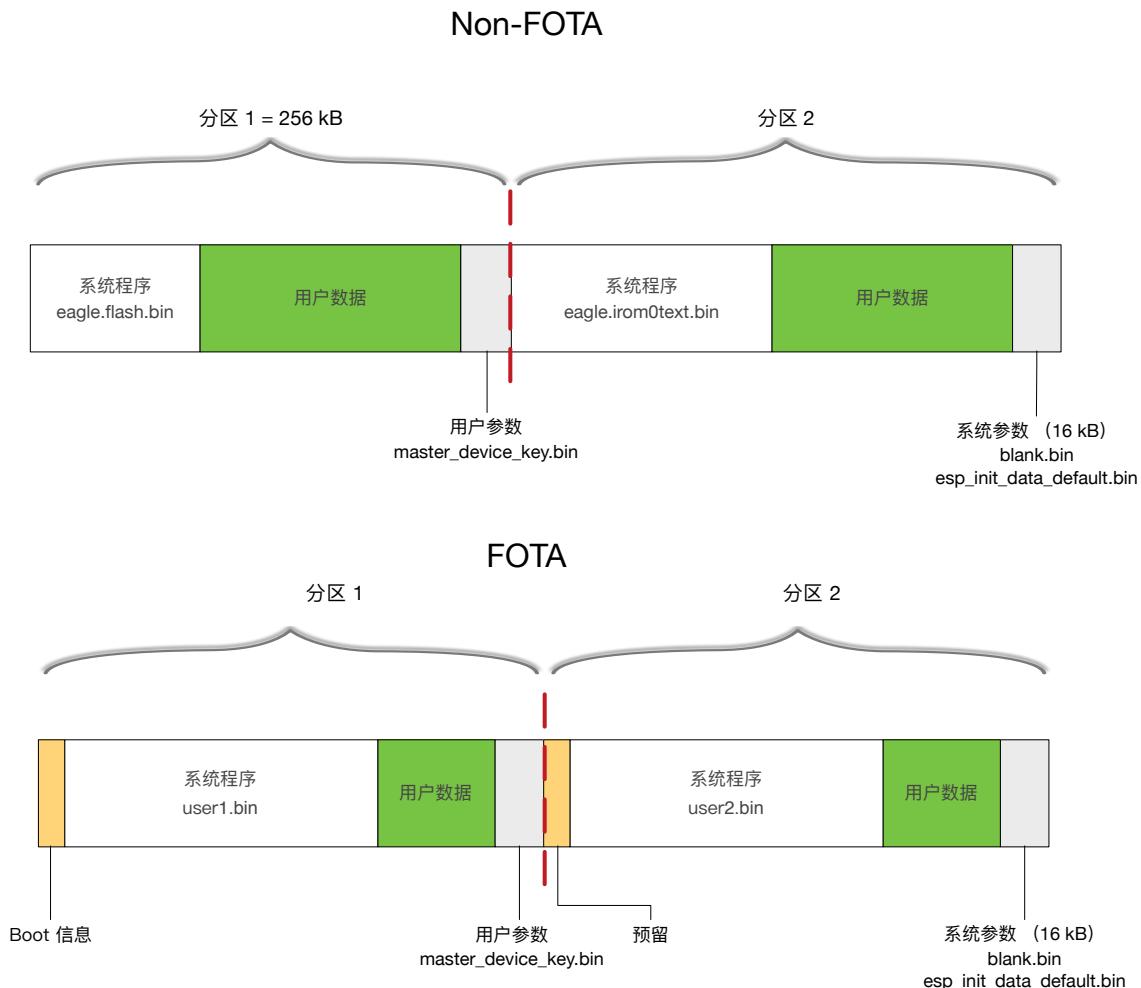


图 4-1. Flash 布局图

### 说明:

关于 ESP8266 的固件请参考“1.3 节 ESP8266 FW”。

- **系统程序**: 用于存放运行系统必要的固件。
- **用户数据**: 当系统数据未占满整个 Flash 空间时, 空闲区域均可用于存放用户数据。



- **用户参数**: 地址由用户自定义, IOT\_Demo 中设置为 0x3C000 开始的 4 个扇区, 用户可以设置为任意未占用的地址。
  - **master\_device\_key.bin** 下载地址在 IOT\_Demo 中设置为用户参数的第 3 个扇区。
- **系统参数**: 固定为 Flash 的最后 4 个扇区。
  - **blank.bin** 下载地址为 Flash 的倒数第 2 个扇区。
  - **esp\_init\_data\_default.bin** 下载地址为 Flash 的倒数第 4 个扇区。
- **Boot 信息**: 位于 FOTA 固件的分区 1, 存放 FOTA 升级相关信息。
- **预留**: 位于 FOTA 固件的分区 2, 与分区 1 Boot 信息区对应的预留区域。

说明:

- Flash 中每扇区为 4 kB。
- 具体下载地址请参考“4.1.2 节 下载地址”和“4.2.2 节 下载地址”。

## 4.1. Non-FOTA

### 4.1.1. 布局说明

对于不同容量的 Flash, **eagle.irom0text.bin** 的存储上限值均为 200 kB。您可以通过修改 **ESP8266\_NONOS\_SDK/ld/eagle.app.v6.ld** 文件来改变 **eagle.irom0text.bin** 的上限值。

即修改文件中 **irom0\_0\_seg** 参数的 **len** 字段, 如图 4-2 所示。

```
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
}
```

图 4-2. len 的位置

不同的 Flash 容量, **len** 的值和修改后 **eagle.irom0text.bin** 的存储上限值如表 4-1 所示。

表 4-1. Non-FOTA Flash 布局 (单位: kB)

Flash 容量	eagle.flash.bin	eagle.irom0text.bin	用户数据	len	用户/系统参数
512	≤ 64	≤ 240	≥ 176	0x3C000	16
1024	≤ 64	≤ 752	≥ 176	0xBC000	16



Flash 容量	eagle.flash.bin	eagle.irom0text.bin	用户数据	len	用户/系统参数
2048	$\leq 64$	$\leq 768$	$\geq 176$	0xC0000	16
4096	$\leq 64$	$\leq 768$	$\geq 176$	0xC0000	16

**说明:**

ESP8266 目前系统程序区最大支持 1024 kB。

#### 4.1.2. 下载地址

Non-FOTA 固件的下载地址如表 4-2 所示。

表 4-2. Non-FOTA 的下载地址 (单位: kB)

BIN	各个 Flash 容量对应的下载地址			
	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	0x10000			

## 4.2. FOTA

#### 4.2.1. 布局说明

FOTA 的固件布局如表 4-3 所示。

表 4-3. FOTA Flash 布局 (单位: kB)

Flash 容量	boot	user1.bin	user2.bin	用户/系统参数	用户数据
512	4	$\leq 236$	$\leq 236$	16	$\geq 0$
1024	4	$\leq 492$	$\leq 492$	16	$\geq 0$
2048 (分区 1 = 512)	4	$\leq 492$	$\leq 492$	16	$\geq 1024$
2048 (分区 1 = 1024)	4	$\leq 1004$	$\leq 1004$	16	$\geq 0$
4096 (分区 1 = 512)	4	$\leq 492$	$\leq 492$	16	$\geq 3072$
4096 (分区 1 = 1024)	4	$\leq 1004$	$\leq 1004$	16	$\geq 2048$



### 4.2.2. 下载地址

FOTA 固件的下载地址如表 4-4 所示。

表 4-4. FOTA 的下载地址（单位：kB）

BIN 文件	各个 Flash 容量对应的下载地址					
	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

说明：

- 支持 FOTA 的固件无需下载 **user2.bin**，您可以从云端服务器升级固件。
- 详细的 FOTA 功能说明，请参考文档《ESP8266 Non-OS SDK 升级指南》。



# 5.

# 编译 SDK

## 说明:

- 本章使用 *ESP8266\_NONOS\_SDK/examples/IoT\_Demo* 中的内容为例介绍如何编译 SDK。
- IoT\_Demo* 提供 Smart Light、Smart Plug 和 Sensor 三种设备，在 *examples/IoT\_Demo/include/user\_config.h* 中定义，请每次只使能一种设备调试，默认为 Smart Light。

## 5.1. 编译准备

### 5.1.1. 修改 SDK 文件

## 说明:

若选择下载 FOTA 固件，则需要修改。

- 进入 Windows 系统。
- 根据不同的 Flash 布局修改 *ESP8266\_NONOS\_SDK/examples/IoT\_Demo/include* 中的文件。
  - user\_light.h* 和 *user\_plug.h* 需要修改 `#define PRIV_PARAM_START_SEC`。

```
/* 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
```

- user\_esp\_platform.h* 需要修改 `#define ESP_PARAM_START_SEC`。

```
/* 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
```

修改的值如表 5-1 所示。

表 5-1. 修改 include 文件中的字段 (单位: kB)

默认值 (512)	修改后的值					
	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

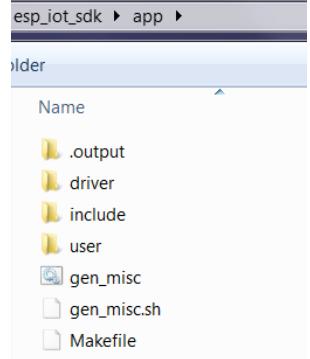
**说明:**

若 Flash 容量为 512 kB, 则不需要修改。

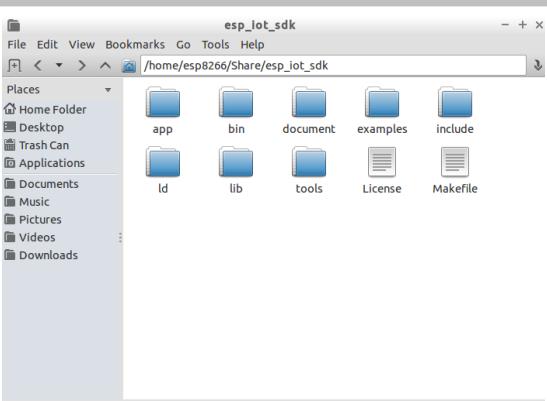
--<  
END

### 5.1.2. 加载 SDK 文件

1. 进入 Linux 系统。
2. 运行虚拟机桌面上的终端工具 LXTerminal。
3. 复制待编译文件至虚拟机共享目录。

步骤	结果
<ul style="list-style-type: none"><li>• 复制 <i>ESP8266_NONOS_SDK</i> 文件夹到您的虚拟机共享目录, 如: <b>C:\VM\share</b> 目录。</li><li>• 将 <i>IoT_Demo</i> 文件夹复制到 <b>C:\VM\share\ESP8266_NONOS_SDK</b> 目录下, 如右图所示。</li></ul>	

4. 加载共享目录。

步骤	结果
<ul style="list-style-type: none"><li>• 执行 <code>./mount.sh</code>。</li><li>• 根据提示输入密码: <b>espressif</b>。 系统完成共享文件加载。</li><li>• 在虚拟机中进入共享目录 <b>ESP8266_NONOS_SDK</b> 下查看文件内容, 确认共享目录是否加载成功。<ul style="list-style-type: none"><li>- 若加载成功目录如右图所示。</li><li>- 若加载不成功, 目录为空, 您需要再次执行本步骤。</li></ul></li></ul>	

**注意:**

若您使用 RTOS SDK, 请继续执行步骤 5; 若您使用 Non-OS SDK, 请跳过步骤 5。

5. 设置路径变量, 指向 SDK 和 BIN 文件。

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



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

说明:

您可以将其添加在 `.bashrc` 文件中，否则每次重启编译器都需要重复步骤 5。

-- END

## 5.2. 开始编译

### 5.2.1. ESP8266\_NONOS\_SDK\_v0.9.5 及之后版本

1. 在终端切换到 `/share/ESP8266_NONOS_SDK/app` 目录。

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

系统显示如下提示信息。

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

2. 如图 5-1 所示，按系统提示根据实际情况选择相应选项。

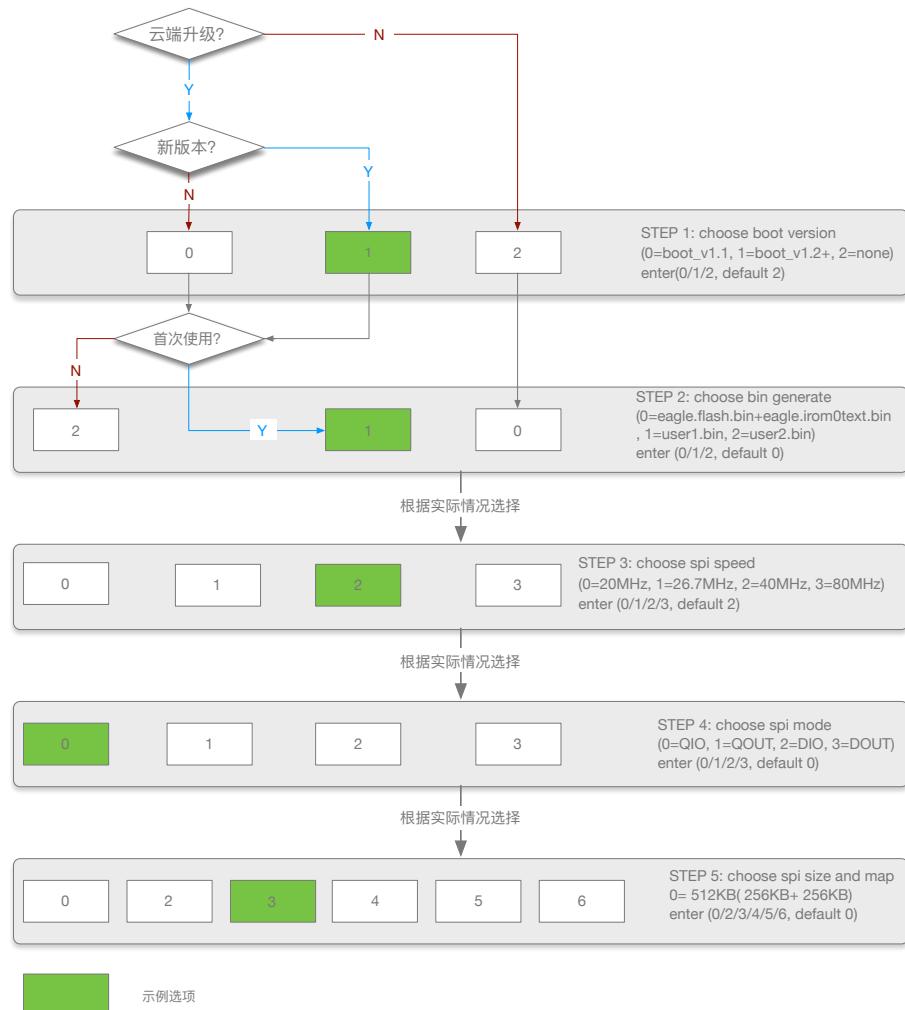


图 5-1. 编译 SDK

**说明:**

- 颜色标示部分为示例选项，您可以按照实际需求选择。
- 关于 FOTA 和 Non-FOTA 的 BIN 文件，请参考“1.4 节 ESP8266 FW”。
- 步骤 5 中的选项 5 和 6 仅 `sdk_v1.1.0 + boot 1.4 + flash download tool_v1.2` 及之后的版本支持。
- 编译生成 `user1.bin` 后，先运行 `make clean` 清除上次编译生成的临时文件后，再编译生成 `user2.bin`。
- 关于步骤 5 中的 Flash 布局，请参考“第 4 章 Flash 布局”。

3. 编译成功后系统显示生成的 BIN 文件及其下载到 Flash 中的地址，如下所示。

```
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
```



!!!

**说明:**

您可以进入 `/home/esp8266/Share/ESP8266_NONOS_SDK/bin` 目录检查生成的 `BIN` 文件。

--<  
END

### 5.2.2. ESP8266\_NONOS\_SDK\_v0.9.4 及之前版本

对于 ESP8266\_NONOS\_SDK\_v0.9.4 及之前版本软件，FOTA 的编译步骤如下。

1. 执行 `./gen_misc_plus.sh 1`, 在 `/ESP8266_NONOS_SDK/bin/upgrade` 路径下生成 `user1.bin`。
2. 执行 `make clean` 清除之前的编译信息。
3. 执行 `./gen_misc_plus.sh 2`, 在 `/ESP8266_NONOS_SDK/bin/upgrade` 路径下生成 `user2.bin`。

**说明:**

- 详细的 FOTA 功能说明, 请参考文档《*ESP8266 Non-OS SDK 升级指南*》。
- *ESP8266\_NONOS\_SDK\_v0.7* 及以前的版本为 *Non-FOTA*。

--<  
END



# 6.

# 下载固件

## 6.1. 下载步骤

1. 进入 Windows 系统。
2. 双击 **ESP\_DOWNLOAD\_TOOL.exe** 打开 Flash 工具。

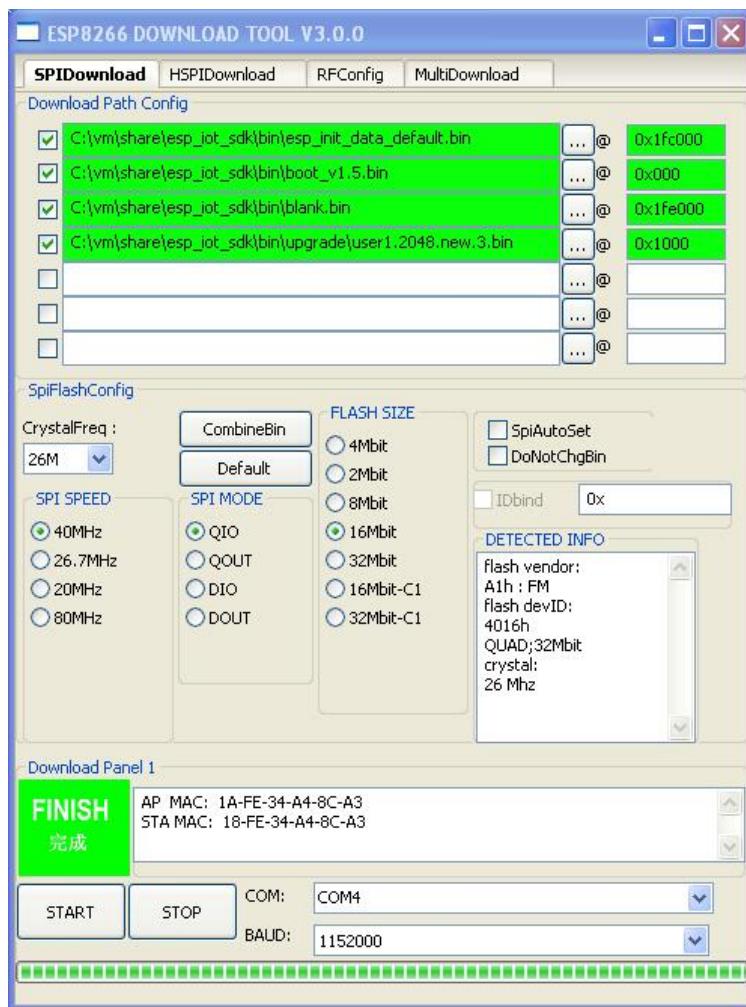


图 6-1. ESP8266 DOWNLOAD TOOL - SPIDownload

SPIDownload 适用于 SPI Flash 的下载。

HSPIDownload 适用于 HSPI Flash 的下载。

RFConfig 射频初始化设置。

MutiDownload 适用于多个母板的下载。



3. 在 **Download Path Config** 区域内双击 选择需要下载的 BIN 文件，在 **ADDR** 内设置相应的下载地址。
4. 配置 SPIDownload 相关选项。

**说明：**

下载文件与地址根据 *SPI Flash* 的容量及实际的使用需求而不同，具体信息可参考“第 4 章 Flash 布局”。

表 6-1. SPIDownload 配置说明

配置项	配置说明
<b>SPI FLASH CONFIG</b>	
<b>CrystalFreq</b>	根据实际选用的晶振型号选择晶振频率。
<b>CombineBin</b>	将勾选的 BIN 文件合成一个 <b>target.bin</b> ，下载地址为 0x0000。
<b>Default</b>	将 SPI Flash 的配置恢复到默认值。
<b>SPI SPEED</b>	选择 SPI Flash 的读写速度，最大值为 80 MHz。
<b>SPI MODE</b>	根据实际使用的 Flash 选择对应的模式。如果 Flash 采用 Dual SPI，选择 <b>DIO</b> 或 <b>DOUT</b> ；如果 Flash 采用 Quad SPI，选择 <b>QIO</b> 或 <b>QOUT</b> 。 <b>注意：</b> 若用户使用 <i>ISSI Flash</i> ，请参考“附录-配置 <i>ISSI Flash QIO</i> 模式”。
<b>FLASH SIZE</b>	根据实际编译的配置对应选择的 Flash 大小。
<b>SpiAutoSet</b>	不建议勾选 <b>SpiAutoSet</b> ，推荐用户根据实际情况对 Flash 进行手动配置。 用户如果了勾选 <b>SpiAutoSet</b> ，下载工具将会按照默认的 Flash map 下载，16 Mbit 和 32 Mbit 的 Flash map 会被设置为 512 Kbyte +512 Kbyte。
<b>DoNotChgBin</b>	<ul style="list-style-type: none"><li>• 用户可勾选 <b>DoNotChgBin</b>，Flash 的运行频率，方式和布局会以用户编译时的配置选项为准。</li><li>• 如果不勾选该选项，Flash 的运行频率，方式和布局会以下载工具最终的配置为准。</li></ul>
<b>Download Panel</b>	
<b>START</b>	点击 <b>START</b> 开始下载。当下载结束后，左边绿色状态显示 <b>完成</b> 。
<b>STOP</b>	点击 <b>STOP</b> 停止下载。
<b>MAC Address</b>	下载成功后，系统会显示 ESP8266 STA 和ESP8266 AP 的 MAC 地址。
<b>COM PORT</b>	选择 ESP8266 的 COM 端口。
<b>BAUDRATE</b>	选择下载的波特率，默認為 115200。

5. 下载完成后，在 ESP-LAUNCHER 开发板上将 **GPI00 Control** 拨到外侧，并重新上电，可进入运行模式。

— — ←  
END

## 6.2. 查看打印信息

下载固件后，您可以使用串口调试工具查看终端打印信息。

您需要配置串口调试工具的以下选项。

表 6-2. 串口调试配置

配置项	配置说明
协议类型	串口
端口号	根据实际连入的设备所在的端口号设置。
波特率	<p>设备运行时的波特率，与设备晶振有关。</p> <ul style="list-style-type: none"><li>• 69120 (晶振 24 M)</li><li>• 74880 (晶振 26 M)</li><li>• 115200 (晶振 40 M)</li></ul> <p>ESP8266 AT 示例默认支持 115200 波特率，用户不可修改。 ESP8266 IOT Demo 及其他示例默认为 74880 波特率，用户可以修改。</p>
数据位	8
校验	无
流控	无

### 6.2.1. ESP8266 IOT Demo

若下载 ESP8266 IOT Demo 固件，在运行模式下，系统显示如下初始化信息，如 SDK 的版本信息等，并在最后显示“finish”字样，代表固件正常运行。

```
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

若下载 ESP8266 AT 固件或者使用开发板或者模组中默认的固件，在运行模式下，系统的打印信息末尾显示“Ready”字样。在终端输入指令“AT”，系统显示“OK”，代表固件正常运行。

#### 说明：

因 AT 固件强制设置波特率为 115200，与 ESP8266 默认的波特率 74880 不符，因此系统初始化的信息会显示为乱码，只要最后显示“Ready”字样，均为正常情况。

关于 AT 指令，请参考《ESP8266 AT 指令集》。

## 6.3. 射频初始化设置（可选）

在下载之前，用户可以进入 **RF InitConfig** 页签修改射频初始化设置，生成的 **esp\_init\_data\_setting.bin** 可代替 **esp\_init\_data\_default.bin** 下载到 Flash 中。射频初始化设置包括**RF InitConfig** 选项设置和 **RF InitConfig** 数据设置两部分。

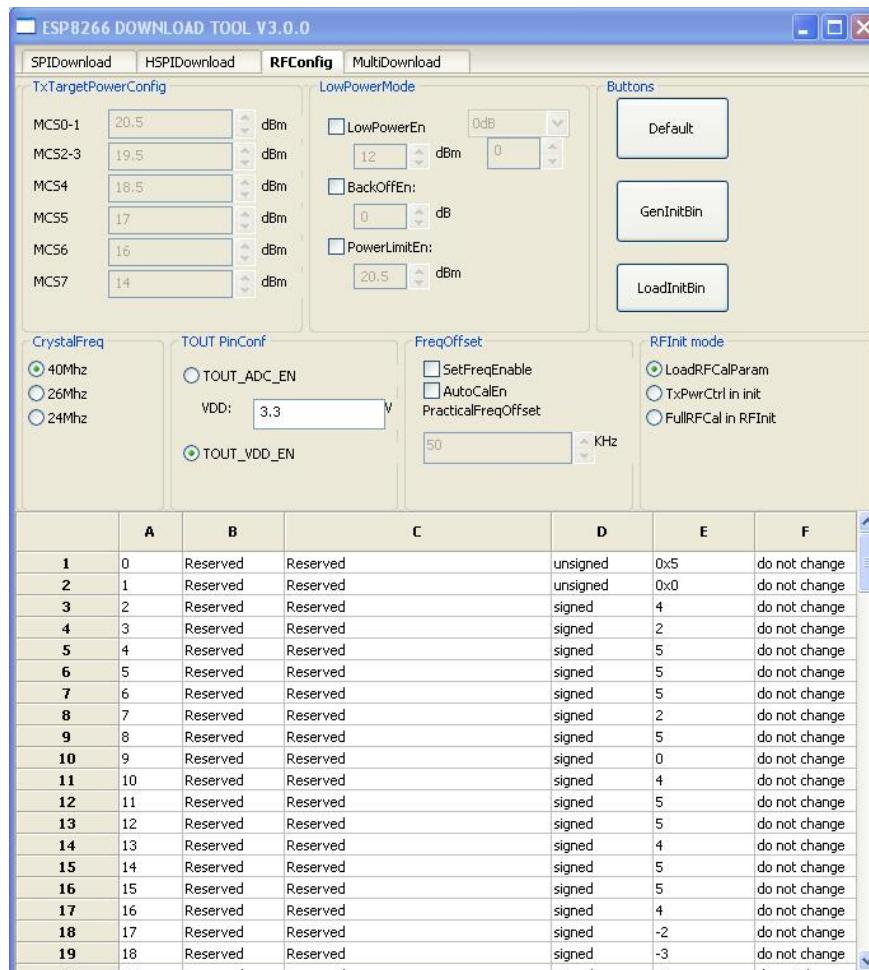


图 6-2. ESP8266 DOWNLOAD TOOL - RF InitConfig



### 6.3.1. RF InitConfig 选项

*RF InitConfig* 选项如图 6-2 的上半部分，配置说明如表 6-3 所示。

表 6-3. RF InitConfig 选项设置

配置项	配置说明
TxTargetPowerConfig	用户无需配置，会根据 LowPowerMode 中的配置项而变化。
LowPowerMode	<p>根据实际情况进行低功耗模式的配置。</p> <ul style="list-style-type: none"><li>• <i>LowPowerEn</i>: 开启低功耗模式，为每个速率设定统一的功耗值。</li><li>• <i>PowerLimitEn</i>: 限定输出功率的最大值。</li><li>• <i>BackOffEn</i>: 为每个速率的功率设置统一的避退阶数。</li></ul> <p> <b>说明:</b></p> <p><i>LowPowerEn</i> 和 <i>PowerLimitEn</i> 不能同时配置。</p>
CrystalFreq	<p>根据实际选用的晶振型号选择晶振频率。</p> <p> <b>说明:</b></p> <p>若在下载时选择了不同的选项，会覆盖该设置。</p>
TOUT PinConf	<p>根据实际情况进行 TOUT 管脚的配置，建议保持默认设置。</p> <ul style="list-style-type: none"><li>• <i>TOUT_ADC_EN</i>: 在 TOUT 管脚接外部电路的情况下，使用芯片内部的 ADC 端口测量外部电压值 (0 V ~ 1 V)。</li><li>• <i>TOUT_VDD_EN</i>: 在 TOUT 管脚悬空的情况下，通过 uint16 system_get_vdd33(void) 测量 VDD33 的电压值。</li></ul> <p> <b>注意:</b></p> <ul style="list-style-type: none"><li>• <i>TOUT_ADC_EN</i> 和 <i>TOUT_VDD_EN</i> 不能同时配置。</li><li>• 选择 <i>TOUT_ADC_EN</i> 时，需要填写真实的 VDD3P3 管脚 3 和 4 上的电源电压。</li></ul>
FreqOffset	<ul style="list-style-type: none"><li>• <i>SetFreqEnable</i>: 手动设置频偏值。</li><li>‣ <i>PracticalFreqOffset</i>: 选择 <i>SetFreqEnable</i> 时，该选项有效。</li><li>• <i>AutoCalEn</i>: 自动调整频偏值。</li></ul>
RFInt mode	<p>用户可选择射频的初始化行为：</p> <ul style="list-style-type: none"><li>• <i>LoadRFCalParam</i>: 初始化时射频数据直接从 Flash 读取，不作任何校准，耗时约 2 ms，初始电流最小。</li><li>• <i>TxPwrCtrl in init</i>: 初始化时仅做 Tx Power 校准，其他从 Flash 读取，耗时约 20 ms，初始电流较小。</li><li>• <i>FullRFCal in RFInit</i>: 初始化时进行全部的校准，耗时约 200 ms，初始电流较大。</li></ul>

### 6.3.2. RF InitConfig 参数

*RF InitConfig* 参数如图 6-2 的下半部分，参数说明如表 6-4 所示。



表 6-4. RF InitConfig 参数设置

配置项	配置说明
A	代表 <code>esp_init_data_setting.bin</code> (0~127 字节) 文件中的字节, 如 A = 0 代表 <code>esp_init_data_setting.bin</code> 文件中的 0 字节。
B	配置项的名称, 标示为 Reserved 不能修改。
C	配置项的名称, 标示为 Reserved 不能修改。
D	配置项的数据类型。包括无符号数 (unsigned) 和有符号数 (signed)。
E	配置项的十六进制值。

**⚠ 注意:**

请勿修改标识为 Reserved 的参数。

下面介绍如何修改 112 ~ 114 字节的参数, 初始设置如图 6-3 所示。

A	B	C	D	E	F
112	tx_param42	freq_correct_en	unsigned	0	bit[0]:0->do not correct freq
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

图 6-3. 112 ~ 114 字节参数

### 修改 RF 初始化参数

114 字节用于控制 ESP8266 上电时 RF 的初始化, 参数设置如表 6-5 所示。

**📖 说明:**

`ESP8266_NONOS_SDK_V1.5.3` 和 `ESP8266_RTOS_SDK_V1.3.0` 及之后版本支持。

表 6-5. 修改 RF 初始化参数

配置项	配置说明
114 字节 = 0	RF 初始化仅做 VDD33 校准, 耗时约 2 ms, 初始电流最小。
114 字节 = 1	114 字节默认值为 1。 RF 初始化做 VDD33 和 TX power CAL, 耗时约 18 ms, 初始电流较小。
114 字节 = 2	同“114 字节 = 0”。
114 字节 = 3	RF 初始化进行全部 RF CAL, 耗时约 200 ms, 初始电流较大。

### 修改频偏

112 字节和 113 字节是与频偏相关的参数, 参数设置如表 6-6 所示。

**说明:**

*ESP8266\_NONOS\_SDK\_V1.4.0* 和 *ESP8266\_RTOS\_SDK\_V1.3.0* 及之后版本支持。

表 6-6. 修改频偏参数

配置项	配置说明
112 字节, 默认值为 3。	
bit 0	拥有 112 字节的最高优先级。 <ul style="list-style-type: none"><li>bit 0 = 0 : 不修改频偏。</li><li>bit 0 = 1 : 修改频偏。</li></ul>
bit 1	值为 0 表示 bbpll 为 168 M, 可修改正或负频偏。 此功能可能影响数字外设的正常工作, 不建议使用。 值为 1 表示 bbpll 为 160 M, 可修改正频偏。
{bit 3, bit 2}	值为 0 表示内部自测且跟踪修改频偏, 初始修改频偏为 0; 值为 1 表示强制修改频偏为 113 字节的值, 不再跟踪修改频偏; 值为 2 表示内部自测且跟踪修改频偏, 初始修改频偏为 113 字节的值。
113 字节, 默认值为 0。	
113 字节	是强制修改频偏时的频偏数据, 或者跟踪修改频偏时的初始频偏数据, 数据类型为 sign int8, 单位为 8 kHz。

### 6.3.3. 设置举例

112 字节和 113 字节需要根据客户的具体需求设置, 举例如下。

#### 1. 只在常温下工作的模组, 不需要修改频偏。

- 可设置 112 字节 = 0, 113 字节 = 0。

#### 2. 只在常温下工作的模组, 不需要内部自测且跟踪修改频偏, 但是本身频偏较大, 可使用强制修改频偏。

- 假设常温时本身频偏为 +160 kHz, 可设置 112 字节 = 0x07, 113 字节 =  $(256 - 160/8) = 236 = 0xEC$ 。
- 假设常温时本身频偏为 -160 kHz, 可设置 112 字节 = 0x05, 113 字节 =  $160/8 = 20 = 0x14$ 。但此功能可能影响数字外设的正常工作, 必须经过大量的应用测试, 不建议使用。

#### 3. 工作温度在 -40 度到 125 度之间变化, 类似彩灯的应用, 需要内部自测且跟踪修改频偏, 但是常温时本身频偏较小, 不需要设置初始频偏。

- 可设置 112 字节 = 0x03, 113 字节 = 0。



4. 工作温度在 -40 度到 125 度之间变化，类似彩灯的应用，需要内部自测且跟踪修改频偏，但是常温时本身频偏较大，需要设置初始频偏。

- 假设常温时本身频偏为 +160 kHz，可设置 112 字节 = 0x0B，113 字节 =  $(256 - 160/8) = 236 = 0xEC$ 。
- 假设常温时本身频偏为 -160 kHz，可设置 112 字节 = 0x09，113 字节 =  $160/8 = 20 = 0x14$ 。但此功能可能影响数字外设的正常工作，必须经过大量的应用测试，不建议使用。

一般建议客户参考上述例三的设置。

设置完成后，点击 **GenInitBin** 按键，生成 **esp\_init\_data\_setting.bin**。

用户也可以点击 **Default** 按键恢复默认值，或点击 **LoadInitBin** 按键，选择导入一个 BIN 文件进行参数配置。



# I. 附录-配置 ISSI Flash QIO 模式

## ⚠ 注意：

下载时，务必选择 *DIO* 或 *DOUT* 模式，否则会报错。

因 ISSI Flash 状态寄存器的特殊性，如需使用 ISSI Flash 的 QIO 模式，您需要修改 *blank.bin* 的前 2 个字节，如表 I-I 所示。当 ESP8266 启动时会检查 *blank.bin* 的前 2 个字节，自动切换为 QIO 模式读 ISSI\_FLASH。*blank.bin* 文件的结构如下。

```
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;  
}
```

表 I-I. *blank.bin* 配置项

配置项	配置说明
不使用二级 boot	修改 <i>to_qio</i> 为 0。
使用二级 boot	修改 <i>use_bin</i> 为 0, <i>to_qio</i> 为 0。修改 <i>version</i> 为当前 boot 版本。 <b>举例：</b> 如使用二级 <i>boot_v1.5.bin</i> , 修改 <i>blank.bin</i> 的前两个字节 FF FF 为 F4 E5。

## 📘 说明：

如要使用 ISSI Flash 的 *DIO* 或 *DOUT* 模式，无需修改 *BIN* 文件。





### 免责申明和版权公告

本文中的信息，包括供参考的 URL 地址，如有变更，恕不另行通知。

文档“按现状”提供，不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档不负任何责任，包括使用本文档内信息产生的侵犯任何专利权行为的责任。本文档在此未以禁止反言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。

Wi-Fi 联盟成员标志归 Wi-Fi 联盟所有。蓝牙标志是 Bluetooth SIG 的注册商标。

文中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。

版权归© 2016 乐鑫所有。保留所有权利。