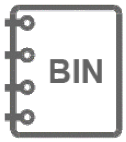
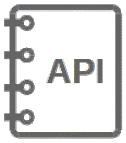
ESP-AT User Guide

Read the Docs

2022 年 04 月 25 日

Contents

This is the documentation for the ESP-AT.

				
入门	AT Binary 列表	AT 命令集	AT 命令示例	编译和开发

CHAPTER 1

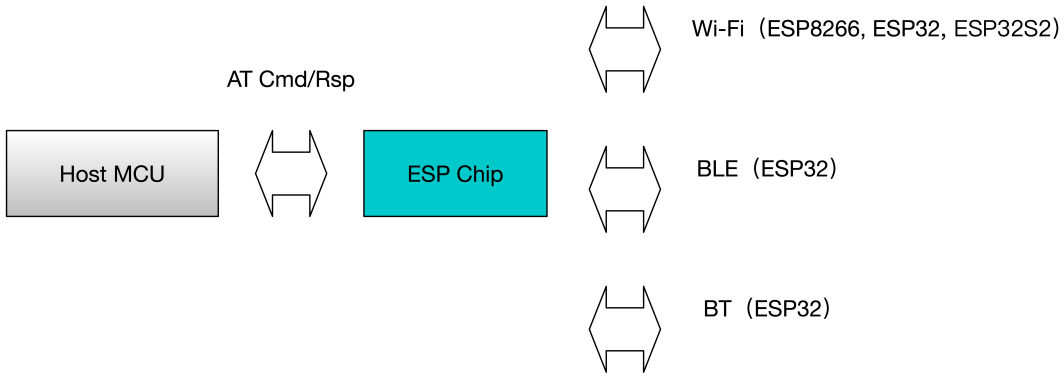
开始

[English]

1.1 ESP-AT 是什么?

ESP-AT 是乐鑫开发的可直接用于量产的物联网应用固件，旨在降低客户开发成本，快速形成产品。通过 ESP-AT 指令，你可以快速的加入无线网络、连接云平台、实现数据通信以及控制等功能，真正的通过无线通讯实现万物互联。

ESP-AT 是基于 ESP-IDF/ESP8266-RTOS-SDK 实现的一个软件工程。它让 ESP 模组作为 slave，连接一个 host MCU 进行工作。host MCU 发送 AT 命令给 ESP 模组，控制 ESP 模组执行不同的操作，并接收 ESP 模组返回的 AT 响应。ESP-AT 提供了大量 AT 命令实现不同的功能，例如，有 Wi-Fi 命令，TCPIP 命令，BLE 命令，BT 命令，MQTT 命令，HTTP 命令，Ethernet 命令等。使得 host MCU 可以直接通过发 AT 命令给 ESP 模组，来实现这些功能。



avatar

“AT” 是 “Attention” 的缩写。AT 命令以 “AT” 为开始，以新的一行 (CR LF) 为结尾。所有命令均为串行执行，每次只能执行一条 AT 命令，每条命令都会返回 `OK` 或者 `ERROR` 来指示当前命令的最终执行状态。在使用 AT 命令时，应该等待上一条命令执行完毕后，再发送下一条命令，如果上一条命令未执行完毕，又发送了新的命令，则会返回 `busy p...` 相关提示，具体可参见命令的详细介绍。默认配置下，host MCU 应通过 UART 连接到 ESP 模组，并通过 UART 发送 AT 命令、接收 AT 响应。但是，开发者也可以自行修改程序，使用其他的通信接口，例如 SDIO。同样，开发者也可以基于 ESP-AT 工程，自行开发更多的 AT 命令，以实现更多的其他功能。

1.2 硬件连接

1.2.1 ESP32 Series

ESP32 AT 需要使用两个串口，UART0 用作下载和打印 log，UART1 发送 AT 命令，由于 ESP32 模组较多，并且每个模组占用的管脚不一样，所以针对不同的模组，UART1 分配的管脚不一样，在采用官方固件时，请一定采用模组对应的 AT 固件。

请参考 https://docs.espressif.com/projects/esp-idf/en/stable/get-started/establish-serial-connection.html，针对不同的模组，下载对应的 USB 驱动。

模组和开发板信息请参考 https://docs.espressif.com/projects/esp-idf/en/stable/hw-reference/modules-and-boards.html#wroom-solo-wrover-and-pico-modules

ESP32-WROOM-32 Series

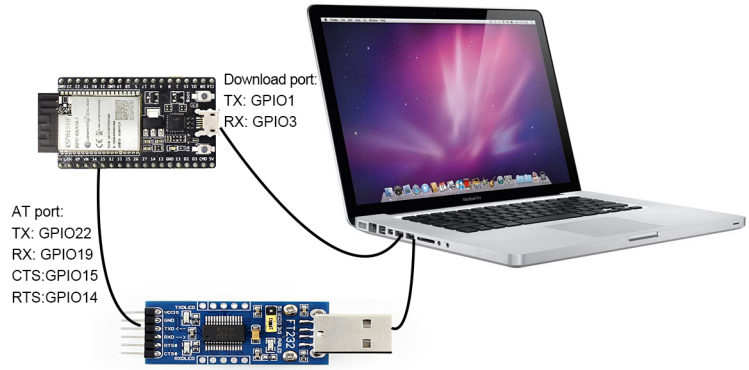
开发板连线示意图

avatar

如果需要基于模组进行连接，请参考官方模组和开发板 datasheet 进行操作。

ESP32-WROVER-32 Series

开发板连线示意图



avatar

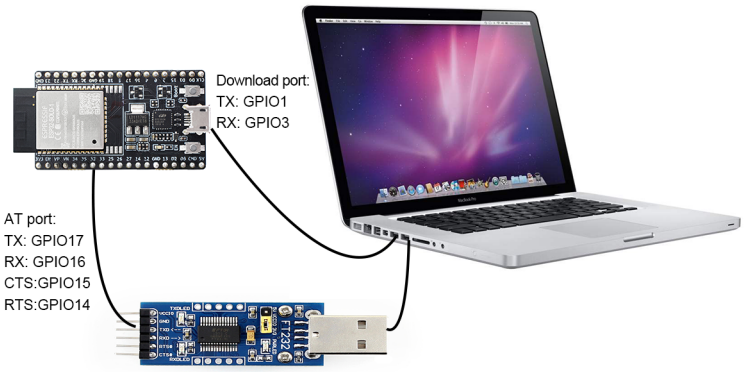
如果需要基于模组进行连接，请参考官方模组和开发板 datasheet 进行操作。

ESP32-PICO Series

开发板连线示意图 avatar

如果需要基于模组进行连接，请参考官方模组和开发板 datasheet 进行操作。

ESP32-SOLO Series



开发板连线示意图 avatar

如果需要基于模组进行连接，请参考官方模组和开发板 datasheet 进行操作。

1.2.2 ESP32S2 Series

ESP32S2 AT 需要使用两个串口，UART0 用作下载和打印 log，UART1 发送 AT 命令。

请参考 https://docs.espressif.com/projects/esp-idf/en/latest/esp32s2/get-started/establish-serial-connection.html，针对不同的模组，下载对应的 USB 驱动。

模组和开发板信息请参考 https://docs.espressif.com/projects/esp-idf/en/latest/esp32s2/hw-reference/index.html

开发板连线示意图 avatar

如果需要基于模组进行连接，请参考官方模组和开发板 datasheet 进行操作。

1.2.3 ESP8266 Series

ESP8266 AT 需要使用两个串口，UART0 用作下载和发送 AT 命令，UART1 用于打印 log，官方默认适配的模组为 ESP-WROOM-02D。

模组和开发板信息请参考 https://www.espressif.com/en/products/socs/esp8266

开发板连线示意图 avatar

如果需要基于模组进行连接，请参考官方模组和开发板 datasheet 进行操作。

1.3 如何烧录 ESP-AT 固件

本文档将针对 windows、linux 和 macos 平台，分别介绍如何使用烧录 AT 固件，其中，AT 固件可从https://docs.espressif.com/projects/esp-at/en/latest/AT_Binary_Lists/index.html 获取，以 ESP32-WROOM-32_AT_Bin_V2.1 为例，介绍各个 bin 文件的用途以及参数含义。目录结构如下：

```
.
  at_customize.bin                  // 二级分区表
  bootloader                        // bootloader
      bootloader.bin
  customized_partitions             // AT 自定义 bin 文件
      ble_data.bin
      client_ca.bin
      client_cert.bin
      client_key.bin
      factory_param.bin
      factory_param_WROOM-32.bin
      mqtt_ca.bin
      mqtt_cert.bin
      mqtt_key.bin
      server_ca.bin
      server_cert.bin
      server_key.bin
  download.config                   // 下载参数信息
  esp-at.bin                        // AT 应用固件
  factory                           // 量产所需打包好的文件
      factory_WROOM-32.bin
      factory_parameter.log
```

(下页继续)

(续上页)

```
  flasher_args.json                 // 下载参数信息新的格式
  ota_data_initial.bin              // ota data 区初始值
  partition_table                   // 一级分区表
      partition-table.bin
  phy_init_data.bin                 // phy 初始值信息
```

下载所需参数保存在 download.config 中，如下：

```
--flash_mode dio --flash_freq 40m --flash_size 4MB
0x8000 partition_table/partition-table.bin
0x10000 ota_data_initial.bin
0xf000 phy_init_data.bin
0x1000 bootloader/bootloader.bin
0x100000 esp-at.bin
0x20000 at_customize.bin
0x24000 customized_partitions/server_cert.bin
0x39000 customized_partitions/mqtt_key.bin
0x26000 customized_partitions/server_key.bin
0x28000 customized_partitions/server_ca.bin
0x2e000 customized_partitions/client_ca.bin
0x30000 customized_partitions/factory_param.bin
0x21000 customized_partitions/ble_data.bin
0x3B000 customized_partitions/mqtt_ca.bin
0x37000 customized_partitions/mqtt_cert.bin
0x2a000 customized_partitions/client_cert.bin
0x2c000 customized_partitions/client_key.bin
```

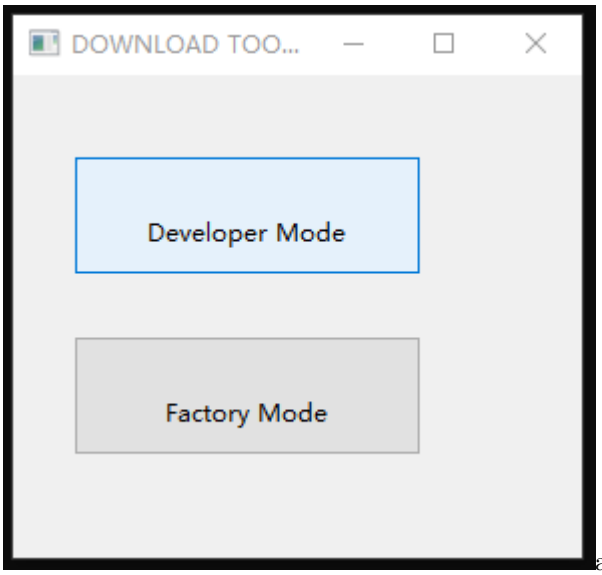
参数含义：

- `--flash_mode dio`: 代表此固件采用的 flash dio 模式进行编译
- `--flash_freq 40m`: 代表此固件采用的 flash 通讯频率为 40MHz
- `--flash_size 2MB`: 代表此固件适用的 flash 最小为 2MB
- `0x10000 ota_data_initial.bin`: 代表在 0x10000 位置烧录 ota_data_initial.bin 文件，后面参数与此类似。

1.3.1 Windows 平台

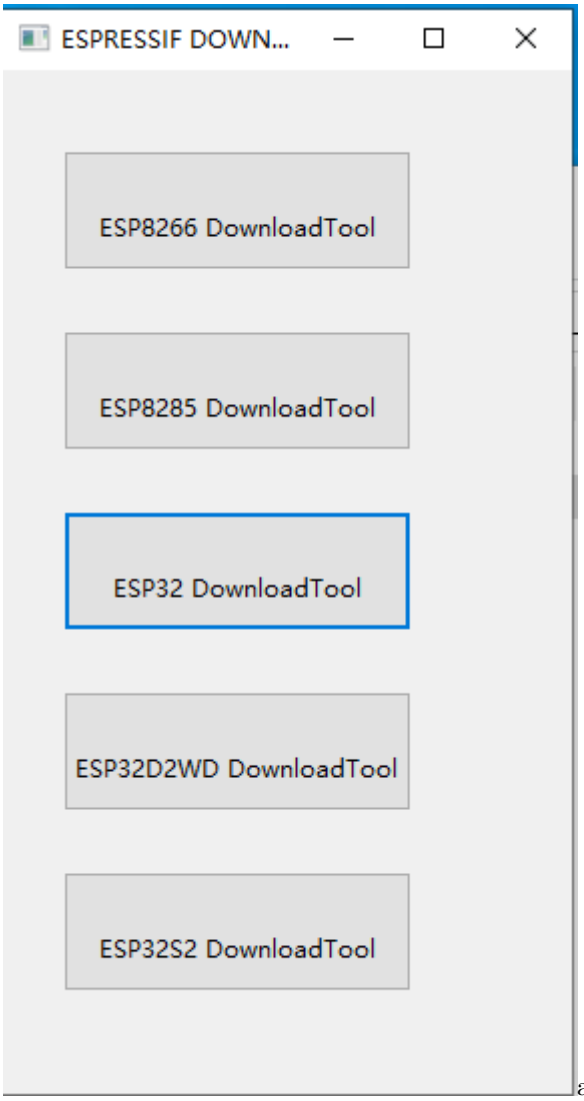
windows 平台下载工具下载地址：https://www.espressif.com/zh-hans/support/download/other-tools，打开工具，如下界面，在此以 ESP-WROOM-32 模组 “Developer Mode” 下载方式为例，如果采用采用量产方式，可以选择 “Factory Mode”，更详细的用法请参见下载工具中的 readme.pdf。

- 打开下载工具



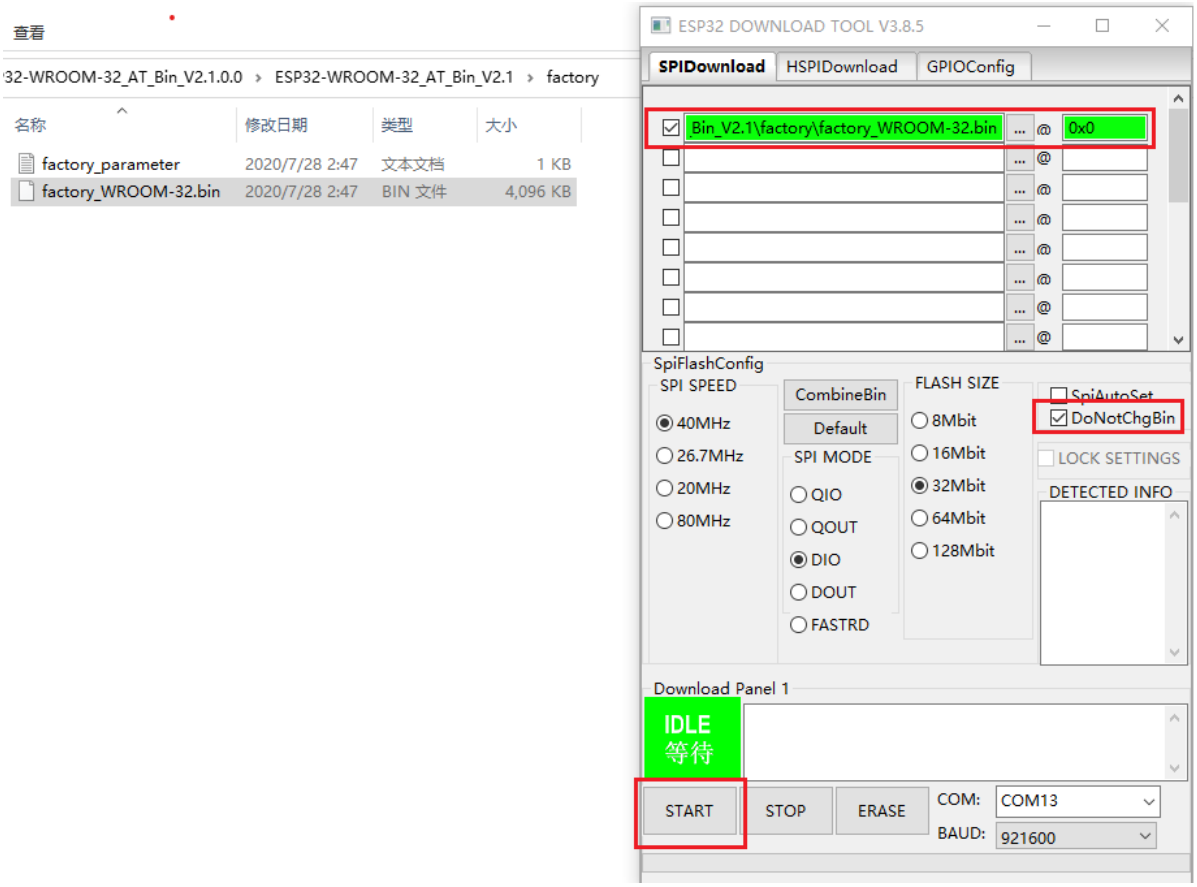
avatar

- 选择对应的芯片类型，如果是 ESP8266 芯片，请选择 “ESP8266 DownloadTool”；如果是 ESP32S2 芯片，请选择 “ESP32S2 DownloadTool”



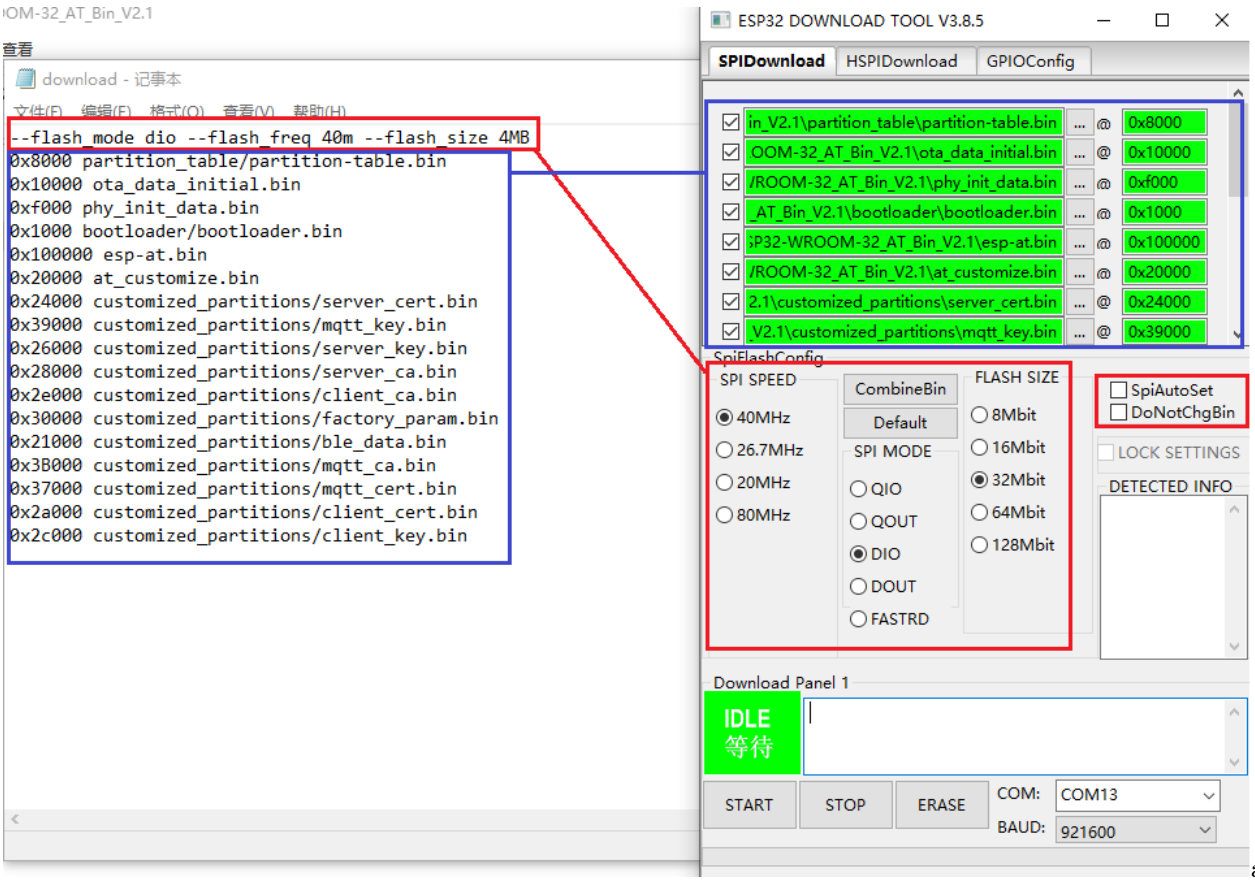
avatar

- 可采用采用量产固件下载或者多个 bin 下载，此处选择其中一个方式即可
  - 打包好的量产固件直接下载



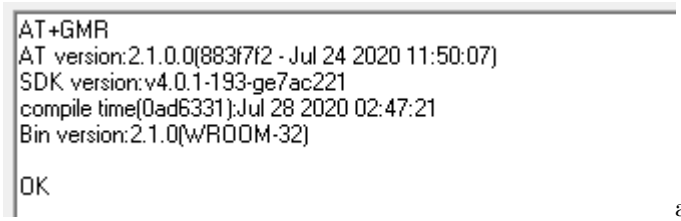
avatar

- 多个 bin 文件分开下载



avatar

- 通过 AT UART 发送“AT+GMR\r\n”命令进行验证，切记：“\r\n”为工具上的回车换行字符。



avatar

1.3.2 Linux 或 macos 平台

Linux 平台下载工具安装方法请参考 https://github.com/espressif/esptool

在此以 ESP-WROOM-32 模组为例，打开 shell 界面，根据自己的 UART port 端口，输入

```
esptool.py --chip auto --port /dev/tty.usbserial-0001 --baud 115200 --before default_
↪reset --after hard_reset write_flash -z
```

和 download.config 文件中的内容

```
--flash_mode dio --flash_freq 40m --flash_size 4MB 0x8000 partition_table/partition-
↪table.bin 0x10000 ota_data_initial.bin 0xf000 phy_init_data.bin 0x1000 bootloader/
↪bootloader.bin 0x100000 esp-at.bin 0x20000 at_customize.bin 0x24000 customized_
↪partitions/server_cert.bin 0x39000 customized_partitions/mqtt_key.bin 0x26000␣
↪customized_partitions/server_key.bin 0x28000 customized_partitions/server_ca.bin␣
↪0x2e000 customized_partitions/client_ca.bin 0x30000 customized_partitions/factory_
↪param.bin 0x21000 customized_partitions/ble_data.bin 0x3B000 customized_partitions/
```

(下页继续)

(续上页)

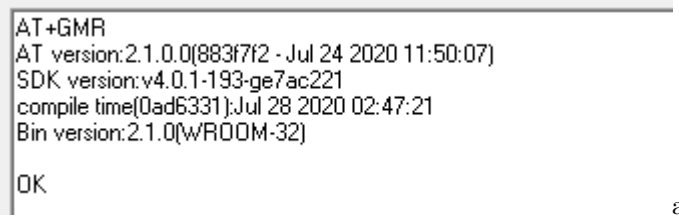
总的命令如下：

```
esptool.py --chip auto --port /dev/tty.usbserial-0001 --baud 921600 --before default_
↪reset --after hard_reset write_flash -z --flash_mode dio --flash_freq 40m --flash_size␣
↪4MB 0x8000 partition_table/partition-table.bin 0x10000 ota_data_initial.bin 0xf000 phy_
↪init_data.bin 0x1000 bootloader/bootloader.bin 0x100000 esp-at.bin 0x20000 at_
↪customize.bin 0x24000 customized_partitions/server_cert.bin 0x39000 customized_
↪partitions/mqtt_key.bin 0x26000 customized_partitions/server_key.bin 0x28000␣
↪customized_partitions/server_ca.bin 0x2e000 customized_partitions/client_ca.bin␣
↪0x30000 customized_partitions/factory_param.bin 0x21000 customized_partitions/ble_data.
↪bin 0x3B000 customized_partitions/mqtt_ca.bin 0x37000 customized_partitions/mqtt_cert.
↪bin 0x2a000 customized_partitions/client_cert.bin 0x2c000 customized_partitions/client_
↪key.bin
```

或者输入

```
esptool.py --chip auto --port /dev/tty.usbserial-0001 --baud 115200 --before default_
↪reset --after hard_reset write_flash -z --flash_mode dio --flash_freq 40m --flash_size␣
↪4MB 0x0 factory/factory_WROOM-32.bin
```

通过 AT UART 发送 “AT+GMR\r\n” 命令进行验证，切记：“\r\n” 为工具上的回车换行字符。



avatar

CHAPTER 2

AT 固件

Please refer to [English]

2.1 发布的固件

2.1.1 ESP32-WROOM-32 Series

- v2.1.0.0 ESP32-WROOM-32_AT_Bin_V2.1.0.0.zip
- v2.0.0.0 ESP32-WROOM-32_AT_Bin_V2.0.zip

2.1.2 ESP32-WROVER-32 Series

- v2.1.0.0 ESP32-WROVER_AT_Bin_V2.1.0.0.zip
- v2.0.0.0 ESP32-WROVER_AT_Bin_V2.0.zip

2.1.3 ESP32-PICO Series

- v2.1.0.0 ESP32-PICO-D4_AT_Bin_V2.1.0.0.zip
- v2.0.0.0 ESP32-PICO-D4_AT_Bin_V2.0.zip

2.1.4 ESP32-SOLO Series

- v2.1.0.0 ESP32-SOLO_AT_Bin_V2.1.0.0.zip
- v2.0.0.0 ESP32-SOLO_AT_Bin_V2.0.zip

2.2 发布的固件

2.2.1 ESP32-S2-WROOM Series

- v2.1.0.0 ESP32-S2-WROOM_AT_Bin_V2.1.0.0.zip

2.2.2 ESP32-S2-WROVER Series

- v2.1.0.0 ESP32-S2-WROVER_AT_Bin_V2.1.0.0.zip

2.2.3 ESP32-S2-SOLO Series

- v2.1.0.0 ESP32-S2-SOLO_AT_Bin_V2.1.0.0.zip

2.2.4 ESP32-S2-MINI Series

- v2.1.0.0 ESP32-S2-MINI_AT_Bin_V2.1.0.0.zip

2.3 发布的固件

2.3.1 ESP-WROOM-02 Series

- v2.1.0.0 ESP8266-IDF-AT_V2.1.0.0.zip
- v2.0.0.0 ESP8266-IDF-AT_V2.0_0.zip

CHAPTER 3

AT Command Set

Here is a list of AT commands. Some of the AT commands can only work on the ESP32, which is marked as [ESP32 Only]; others can work on both the ESP8266 and ESP32.

3.1 Basic AT 命令集

See: `/docs/en/AT_Command_Set/Basic_AT_Commands.md`

3.2 Wi-Fi AT 命令集

See: `/docs/en/AT_Command_Set/Wi-Fi_AT_Commands.md`

3.3 TCP-IP AT 命令集

See: `/docs/en/AT_Command_Set/TCP-IP_AT_Commands.md`

3.4 [ESP32_Only] BLE AT 命令集

See: `/docs/en/AT_Command_Set/BLE_AT_Commands.md`

3.5 [ESP32_Only] BT AT 命令集

See: /docs/en/AT_Command_Set/BT_AT_Commands.md

3.6 MQTT AT 命令集

See: /docs/en/AT_Command_Set/MQTT_AT_Commands.md

3.7 HTTP AT 命令集

See: /docs/en/AT_Command_Set/HTTP_AT_Commands.md

3.8 [ESP32_Only] Ethernet AT 命令集

See: /docs/en/AT_Command_Set/Ethernet_AT_Commands.md

3.9 信令测试 AT 命令集

See: /docs/en/AT_Command_Set/Signaling_Test_AT_Commands.md

Before checking the command set details, please review some common information on command types, configurations that can be saved in the flash, as well as messages returned after entering commands.

- *AT Command Types*
- *AT Commands with Configuration Saved in the Flash*
- *AT Messages*

3.10 AT Command Types

Generic AT command has four types:

Type	Command Format	Description
Test Command	AT+=?	Queries the Set Commands' internal parameters and their range of values.
Query Command	AT+?	Returns the current value of parameters.
Set Command	AT+=<···>	Sets the value of user-defined parameters in commands, and runs these commands.
Execute Command	AT+	Runs commands with no user-defined parameters.

- Not all AT commands support all four types mentioned above.
- Square brackets [ ] designate parameters that may be omitted; default value of the parameter will be used instead.

  Below are examples of entering command *AT+CWJAP* with some parameters omitted:

```
AT+CWJAP="ssid","password"
AT+CWJAP="ssid","password","11:22:33:44:55:66"
```

- If the parameter which is not the last one is omitted, you can give a , to indicate it.

  Example:

```
AT+CWJAP="ssid","password",,1
```

- String values need to be included in double quotation marks, for example: `AT+CWSAP="ESP756290","21030826",1,4`.
- Escape character syntax is needed if a string contains any special characters, such as `,`, `"` or `\`:
  - `\\`: escape backslash itself
  - `\,`: escape comma which is used to separate each parameter
  - `\"`: escape double quotation marks which used to mark string input
  - `\<any>`: escape `<any>` character means that drop backslash symbol and only use `<any>` character

    Example:

```
AT+CWJAP="comma\,backslash\\ssid","1234567890"
AT+MQTTPUB=0,"topic","\"{\"sensor\":012}\"",1,0
```

- The default baud rate of AT command is 115200.
- AT commands are ended with a new-line (CR-LF), so the serial tool should be set into "New Line Mode" .
- Definitions of AT command error codes are provided in *AT API Reference*:
  - *esp_at_error_code*

  - *esp_at_para_parse_result_type*
  - *esp_at_result_code_string_index*

3.11 AT Commands with Configuration Saved in the Flash

Configuration settings entered by the following AT Commands will always be saved in the flash NVS Area, so they can be automatically restored on reset:

- *AT+UART_DEF*: for example, `AT+UART_DEF=115200,8,1,0,3`
- *AT+SAVETRANSLINK* : for example, `AT+SAVETRANSLINK=1,"192.168.6.10",1001`
- *AT+CWAUTOCONN*: for example, `AT+CWAUTOCONN=1`

Saving of configuration settings by several other commands can be switched on or off with *AT+SYSSTORE* command. Please see description of *AT+SYSSTORE* for details.

3.12 AT Messages

Messages	Description
ready	The AT firmware is ready.
ERROR	AT command error, or error occurred during execution.
WIFI CONNECTED	ESP station connected to an AP.
WIFI GOT IP	ESP station got IP address.
WIFI DISCONNECT	ESP station disconnected from an AP.
busy p…	Busy processing. The system is in process of handling the previous command, cannot accept the newly input.
<conn_id>,CONNECT	A network connection of which ID is <conn_id> is established.
<conn_id>,CLOSED	A network connection of which ID is <conn_id> ends.
+IPD	Network data received.
+STA_CONNECTED: <sta_mac>	A station connects to the ESP softAP.
+DIST_STA_IP: <sta_mac>,<sta_ip>	ESP softAP distributes an IP address to the station connected.
+STA_DISCONNECTED: <sta_mac>	A station disconnects from the ESP softAP.
+BLECONN	A BLE connection established.
+BLEDISCONN	A BLE connection ends.
+READ	A read operation from BLE connection.
+WRITE	A write operation from BLE connection.
+NOTIFY	A notification from BLE connection.
+INDICATE	An indication from BLE connection.
+BLESECNTFYKEY	BLE SMP key
+BLEAUTHCMPL	BLE SMP pairing completed.

CHAPTER 4

AT 命令示例

Please refer to [English]

4.1 TCP-IP AT Examples

See: /docs/en/AT_Command_Examples/TCP-IP_AT_Examples.md

4.2 [ESP32_Only] BLE AT Examples

See: /docs/en/AT_Command_Examples/BLE_AT_Examples.md

4.3 MQTT AT Examples

See: /docs/en/AT_Command_Examples/MQTT_AT_Examples.md

CHAPTER 5

如何编译和开发自己的 AT 工程

[English]

5.1 How to clone project and compile it

- *ESP32 platform*
- *ESP32S2 platform*
- *ESP8266 platform*

For specific supported modules, please refer to factory_param_data.csv

5.1.1 ESP32 platform

Hardware Introduction

The WROOM32 Board sends AT commands through UART1 by default.

- GPIO16 is RXD
- GPIO17 is TXD
- GPIO14 is RTS
- GPIO15 is CTS

The debug log will be output through UART0 by default, where TXD0 is GPIO1 and RXD0 is GPIO3, but user can change it in menuconfig if needed:

`make menuconfig` –> `Component config` –> `Common ESP-related` –> `UART for console output`

注解: Please pay attention to possible conflicts of the pins

- If choosing `AT through HSPI`, you can get the information of the HSPI pin by `make menuconfig` –> `Component config` –> `AT` –> `AT hspi settings`
- If enabling `AT ethernet support`, you can get the information of the ethernet pin from `ESP32_AT_Ethernet.md`.

Compiling and flashing the project

Suppose you have completed the installation of the compiler environment for ESP-IDF, if not, you should complete it referring to ESP-IDF Getting Started Guide. Then, to compile ESP-AT project properly, please do the following additional steps:

```
step 1: install python>=3.8
step 2: [install pip](https://pip.pypa.io/en/latest/installing/)
step 3: install the following python packages with pip: pip install pyyaml xlrd
```

Compiling the ESP-AT is the same as compiling any other project based on the ESP-IDF:

注解: Please do not set `IDF_PATH` unless you know ESP-AT project in particular. ESP-IDF will automatically be cloned.

---

1. You can clone the project into an empty directory using command:

   ```
   git clone --recursive https://github.com/espressif/esp-at.git
   ```

2. `rm sdkconfig` to remove the old configuration and `rm -rf esp-idf` to remove the old ESP-IDF if you want to compile other esp platform AT.
3. Set the latest default configuration by `make defconfig`.
4. `make menuconfig` -> `Serial flasher config` to configure the serial port for downloading.
5. `make flash` or `make flash SILENCE=1` to compile the project and download it into the flash, and `make flash SILENCE=1` will remove some logs to reduce firmware size.
   - Or you can call `make` to compile it, and follow the printed instructions to download the bin files into flash by yourself.

- `make print_flash_cmd` can be used to print the addresses of downloading.
- More details are in the esp-idf README.
- If enable BT feature, the firmware size will be much larger, please make sure it does not exceed the ota partition size.

6. `make factory_bin` to combine factory bin, by default, the factory bin is 4MB flash size, DIO flash mode and 40MHz flash speed. If you want use this command, you must first run:: - `make print_flash_cmd | tail -n 1 > build/download.config` to generate `build/download.config`.
7. If the ESP-AT bin fails to boot, and prints "ota data partition invalid" , you should run `make erase_flash` to erase the entire flash.

5.1.2 ESP32S2 platform

Hardware Introduction

The WROOM32S2 Board sends AT commands through UART1 by default.

- GPIO18 is RXD
- GPIO17 is TXD
- GPIO19 is RTS
- GPIO20 is CTS

The debug log will output through UART0 by default, which TXD0 is GPIO1 and RXD0 is GPIO3, but user can change it in menuconfig if needed:

`make menuconfig –> Component config –> Common ESP-related –> UART for console output`

Compiling and flashing the project

Suppose you have completed the installation of the compiler environment for ESP-IDF, if not, you should complete it referring to ESP-IDF Getting Started Guide. If required download the compiler toolchain. Then, to compile ESP-AT project properly, please do the following additional steps:

```
step1:python > 3.8.0
step2:[install pip](https://pip.pypa.io/en/latest/installing/)
step3:install the following python packages with pip3: pip3 install pyyaml xlrd
```

Compiling the ESP-AT is the same as compiling any other project based on the ESP-IDF:

注解: Please do not set `IDF_PATH` unless you know ESP-AT project in particular. ESP-IDF will automatically be cloned.

---

1. You can clone the project into an empty directory using command:

```
git clone --recursive https://github.com/espressif/esp-at.git
```

2. `rm sdkconfig` to remove the old configuration and `rm -rf esp-idf` to remove the old ESP-IDF if you want to compile other esp platform AT.

3. Set esp module information:

```
export ESP_AT_PROJECT_PLATFORM=PLATFORM_ESP32S2
export ESP_AT_MODULE_NAME=WROOM
export ESP_AT_PROJECT_PATH=$(pwd)
```

4. `./esp-idf/tools/idf.py -DIDF_TARGET=esp32s2 build` to compile the project and download it into the flash, and `./esp-idf/tools/idf.py -DIDF_TARGET=esp32s2 -DSILENCE=1 build` will remove some logs to reduce firmware size.

Follow the printed instructions to download the bin files into flash by yourself.

5.1.3 ESP8266 platform

Hardware Introduction

The ESP8266 WROOM 02 Board sends AT commands through UART0 by default.

- GPIO13 is RXD
- GPIO15 is TXD
- GPIO1 is RTS
- GPIO3 is CTS

The debug log will output through UART1 by default, which TXD0 is GPIO2, but user can change it in menuconfig if needed:

`make menuconfig` –> `Component config` –> `ESP8266-specific` –> `UART for console output`

Compiling and flashing the project

Suppose you have completed the installation of the compiler environment for ESP-IDF, if not, you should complete it referring to ESP8266 RTOS SDK Getting Started Guide. Then, to compile ESP-AT project properly, please do the following additional steps:

```
step1:install python 2.7 or python 3.x
step2:[install pip](https://pip.pypa.io/en/latest/installing/)
step3:install the following python packages with pip: pip install pyyaml xlrd
```

Compiling the ESP-AT is the same as compiling any other project based on the ESP-IDF:

注解: Please do not set `IDF_PATH` unless you know ESP-AT project in particular. ESP-IDF will automatically be cloned.**

---

1. You can clone the project into an empty directory using command:

```
git clone --recursive https://github.com/espressif/esp-at.git
```

2. Change the Makefile from:

```
export ESP_AT_PROJECT_PLATFORM ?= PLATFORM_ESP32
export ESP_AT_MODULE_NAME ?= WROOM-32
```

   to be:

```
export ESP_AT_PROJECT_PLATFORM ?= PLATFORM_ESP8266
export ESP_AT_MODULE_NAME ?= WROOM-02
```

3. `rm sdkconfig` to remove the old configuration and `rm -rf esp-idf` to remove the old ESP-IDF if you want to compile other esp platform AT.
4. Set the latest default configuration by `make defconfig`.
5. `make menuconfig` -> `Serial flasher config` to configure the serial port for downloading.
6. `make flash` or `make flash SILENCE=1` to compile the project and download it into the flash, and `make flash SILENCE=1` will remove some logs to reduce firmware size.
   - Or you can call `make` to compile it, and follow the printed instructions to download the bin files into flash by yourself.
   - `make print_flash_cmd` can be used to print the addresses of downloading.
   - More details are in the ESP-IDF README.
7. `make factory_bin` to combine factory bin, by default, the factory bin is 4MB flash size, DIO flash mode and 40MHz flash speed. If you want use this command, you must fisrt run `make print_flash_cmd | tail -n 1 > build/download.config` to generate `build/download.config`.
8. If the ESP-AT bin fails to boot, and prints "ota data partition invalid" , you should run `make erase_flash` to erase the entire flash.

5.2 如何修改 AT port 管脚

在 esp-at 工程中，默认使用了两个 UART: UART0 和 UART1. 在有些情况下，用户可能想要修改管脚配置已满足自己的产品需求. 由于 esp-at 当前可支持 ESP8266 和 ESP32 两个平台，另个平台硬件有些差异，所以 UART 的配置方式也有少许差异.

5.2.1 ESP32 平台

ESP32 的 UART 管脚可以通过管脚映射的方式进行修改, 具体请参见 ESP32 技术参考手册, 在官方 release 固件中，UART0 作为 Log 的打印，默认管脚为

```
TX ---> GPIO1
RX ---> GPIO3
```

可以通过 make menuconfig > Component config > Common ESP-related > UART for console output 进行修改. UART1 作为 AT 命令通讯使用 (只能为 UART1, 但管脚可修改), 默认管脚配置在 factory_param.bin 中，可以在 customized_partitions/raw_data/factory_param/factory_param_data.csv 文件中修改，不同的模组固件可能管脚不同，关于 factory_param_data.csv 的含义描述，可参阅 ESP_AT_Factory_Parameter_Bin.md. 比如 WROOM-32 模组

Parameter	Value
platform	PLATFORM_ESP32
module_name	WROOM-32
magic_flag	0xfcfc
version	1
module_id	1
tx_max_power	78
uart_port	1
start_channel	1
channel_num	13
country_code	CN
uart_baudrate	115200
uart_tx_pin	17
uart_rx_pin	16
uart_ctx_pin	15
uart_rts_pin	14
tx_control_pin	-1
rx_control_pin	-1

发送命令的 AT port 管脚分别为

```
TX ---> GPIO17
RX ---> GPIO16
CTS ---> GPIO15
RTX ---> GPIO14
```

如果想要使用 GPIO1 (TX)、GPIO3 (RX) 同时作为 Log 打印和 AT 命令输入，可以采用如下操作:

1. 打开 customized_partitions/raw_data/factory_param/factory_param_data.csv 文件
2. 修改 WROOM-32 模组的 uart_port 为 0，uart_tx_pin 为 1 以及 uart_rx_pin 为 3，如下

Parameter	Value
platform	PLATFORM_ESP32
module_name	WROOM-32
magic_flag	0xfcfc
version	1
module_id	1
tx_max_power	78
uart_port	0
start_channel	1
channel_num	13
country_code	CN
uart_baudrate	115200
uart_tx_pin	1
uart_rx_pin	3
uart_ctx_pin	-1
uart_rts_pin	-1
tx_control_pin	-1
rx_control_pin	-1

3. 然后保存，重新编译固件，并完全烧录固件即可. 注意：一定要同时烧录对应的 factory_param.bin.

5.2.2 ESP8266 平台

ESP8266 共有两组 UART 口，分别为：UART0 和 UART1，其中，UART1 只有 TX 功能（GPIO2)。所以只能使用 UART0 作为命令输入口. 由于 ESP8266 UART pin 并不能像 ESP32 那样任意映射，只能使用 GPIO15 作为 TX、GPIO13 作为 RX, 或者使用 GPIO1 作为 TX、GPIO3 作为 RX. 默认 LOG UART 为 UART1, TX 为 GPIO2; AT port UART 为 UART0, TX 为 GPIO15, RX 为 GPIO13.

如果想要使用 GPIO1 (TX)、GPIO3 (RX) 同时作为 Log 打印和 AT 命令输入，可以采用如下操作 (WROOM-02 为例):

1. `make menuconfig` > `Component config` > `ESP8266-specific` > `UART for console output` > `Default: UART0`
2. 修改 customized_partitions/raw_data/factory_param/factory_param_data.csv 文件中 `WROOM-02` 模组的 `uart_tx_pin` 和 `uart_rx_pin` 分别为 1 和 3，如下

Parameter	Value
platform	PLATFORM_ESP8266
module_name	WROOM-02
magic_flag	0xfcfc
...	...
uart_baudrate	115200
uart_tx_pin	1
uart_rx_pin	3
uart_ctx_pin	-1
uart_rts_pin	-1
...	...

3. 然后保存，重新编译固件，并完全烧录固件即可. 注意：一定要同时烧录对应的 `factory_param.bin`.

5.3 How to add user-defined AT commands

AT firmware is based on the Espressif IoT Development Framework (ESP-IDF). Espressif Systems' AT commands are provided in `libat_core.a`, which is included in the *AT BIN firmware*.

Examples of implementing user-defined AT commands are provided in main/interface/uart/at_uart_task.c.

- The total length of an AT command cannot be longer than 256 bytes.
- Only alphabetic characters (A~Z, a~z), numeric characters (0~9), and some other characters (`!`, `%`, `-`, `.`, `/`, `:`, `_`) are supported when naming user-defined AT commands.
- The structure, *`esp_at_cmd_struct`*, is used to define *four types* of a command.

5.4 如何创建默认出厂参数 bin 文件

See: `/docs/en/Compile_and_Develop/How_to_create_factory_parameter_bin.md`

5.5 如何自定义 Ble services

5.5.1 BLE services 的文件位置

BLE Service 的源文件路径是: `esp-at/components/customized_partitions/raw_data/ble_data/example.csv`. 如果需要自定义服务，那么需要这么做:

- 修改 `BLE Service` 文件
- 使用 esp-at/tools/BLEService.py 重新生成 ble_data.bin
- 将 ble_data.bin 下载到 ESP32，具体下载的地址在 `module_config/module_esp32_default/partitions_at.csv` 分区表里面有定义。

5.5.2 如何修改 BLE services 文件

BLE services 的定义类似于一个多元数组，每一行是一个 GATT 结构体，每个服务都是由 service 定义、characteristic 定义以及一些可选的 description 组成。

用户可以定义多个服务，比如 Service A、Service B 和 Service C，这三个服务就需要依次排序，因为每个服务的定义都是类似的，我们只拿其中一个举例说明。

首先，需要说明的是，所有服务定义的第一行都是固定的，第一行用来定义了服务的 UUID，标志着一个服务定义的开始。例如示例中，0x2800 表示这一行定义了一个 Primary Service, 具体的这个服务的 UUID 在 value 字段 (用户也可定义为 SIG 颁布的 UUID，例如 0x180A, 也可以自定义，例如示例中是一个自定义的服务，UUID 为 0xA002)。

- 例如:

从第二行开始就是这个服务所包含的 characteristics 的定义, 每个 characteristic 至少有两行组成，第一行是 characteristic 的申明，它的 UUID 是固定的 0x2803，value 字段表示的是这一行可读可写的属性，这里直接参照示例，不要修改，全部是可读可写。第二行是 characteristic 的本身，UUID 可以用户自己定义，这里 value 字段就是 characteristic 本身的数值。

- 例如:

某些 characteristic 后面还会跟着 description。比如，如果这个 characteristic 是可 notify 的，后面就必须跟着 UUID 为 0x2902 的 description。

- 例如:

定义完所有的 characteristics，就是一个服务的结束，如果还想定义其他的 services ，同样的方法依次排列即可

5.6 如何自定义分区

See: `/docs/en/Compile_and_Develop/How_to_customize_partitions.md`

5.7 如何使用 ESP-AT 经典蓝牙

See: /docs/en/Compile_and_Develop/How_to_use_ESP_AT_Classic_Bluetooth.md

5.8 如何使用 ESP-AT ethernet 接口

See: /docs/en/Compile_and_Develop/How_to_enable_ESP_AT_Ethernet.md

5.9 如何增加一个新的平台支持

当前工程根据不同的模组采用了不同的配置方式，具体配置信息在 module_config 目录下，如果未指定对应的模组配置信息，将采用平台默认的配置信息，现在已支持 esp32 和 esp8266 平台。工程默认为 PLATFORM_ESP32 平台的 WROOM-32 模组。对于同款芯片不同通讯接口的模组，由于在 esp-at 中编译的代码不同，我们不能采用默认的 module_espxxxx_default 配置。

假设新的平台为 ESP32 SDIO AT，我们需要使用 SDIO 作为通讯介质，我们以此为例，进行阐述如何添加新的平台设备。

5.9.1 1. 创建模块信息

假设平台名称为 PLATFORM_ESP32, 模块名称为 WROOM32-SDIO, 打开 components/customized_partitions/raw_data/factory_param/factory_param_data.csv, 按照标题

在最后添加

5.9.2 2. 修改工程模块信息

打开 Makefile，修改平台名称和模块名称，对于英文字母，请使用大写格式

```
export ESP_AT_PROJECT_PLATFORM ?= PLATFORM_ESP32
export ESP_AT_MODULE_NAME ?= WROOM32-SDIO
```

**

5.9.3 3. 创建平台相关的配置

- 3.1 进入 module_config, 将同款芯片的默认配置 module_esp32_default 拷贝一份为 module_esp32-sdio
- 3.2 此处我们不需要修改 partition 分区表和 IDF 版本，所以 at_customize.csv、IDF_VERSION 和 partitions_at.csv 都不做修改

- 3.3 修改 sdkconfig.defaults 文件
  - 配置使用 module_esp32-sdio 目录下的分区表文件，需要修改如下配置：

```
CONFIG_PARTITION_TABLE_CUSTOM_FILENAME="module_config/module_esp32-sdio/partitions_
↪at.csv"

CONFIG_PARTITION_TABLE_FILENAME="module_config/module_esp32-sdio/partitions_at.csv"

CONFIG_AT_CUSTOMIZED_PARTITION_TABLE_FILE="module_config/module_esp32-sdio/at_
↪customize.csv"
```

  - 使用 sdio 配置，由于工程中已经包含选择 sdio 的配置，所以我们只需要将选择 sdio 的配置加入到 sdkconfig.defaults 文件即可
    * 移除 UART AT 相关配置

```
CONFIG_AT_BASE_ON_UART=y
```

    并新增

```
CONFIG_AT_BASE_ON_SDIO=y
```

5.9.4 4. 修改链接的库文件

由于 ESP32 SDIO AT 和 ESP32 UART AT 是同一个平台，使用的是相同的 at core 库，所以我们不需要再新增库文件。若需要使用新的 lib，则将 lib 复制到 components/at/lib 目录下，并将 lib 命名为 libxxxx_at_core.a，其中 xxxx 为平台名称。假如根目录下的 Makefile 设置的 ESP_AT_PROJECT_PLATFORM ?= PLATFORM_ESP8848，那么库的名称就要命名为 libesp8848_at_core.a。

5.10 ESP32 SDIO AT 指南

5.10.1 简介

SDIO AT 基于 ESP32 AT，使用 SDIO 协议进行通讯，其中 ESP32 作为 SDIO slave 与 MCU 进行通信。SDIO 协议需要至少 4 根线：CMD，CLK，DAT0 和 DAT1；

- 对于一线模式，DAT1 作为中断线；
- 对于四线模式，需要增加 DAT2 和 DAT3。

SDIO slave 管脚如下所示：

- CLK GPIO14

- CMD GPIO15
- DAT0 GPIO2
- DAT1 GPIO4
- DAT2 GPIO12 （四线）
- DAT3 GPIO13 （四线）

5.10.2 SDIO 下载

ESP-SDIO-TESTBOARD-V1 流程

1. 开关 1、2、3、4，5 拨至 ON，其他均为 OFF。
2. PC 为 master 烧录固件。烧录完成后，slave 侧 ESP32 模组的灯自动亮起，表示 master 成功运行，为 slave 供电。
3. PC 烧写 SDIO AT 程序到 slave。

注意：如果你使用 ESP32-DevKitC 或者 ESP-WROVER-KIT V2（或更早之前的板子）来验证 SDIO AT，首先请参照 SDIO demo 中的 board-compatibility 对 strapping 管脚进行处理，在此之后，强烈建议先运行 SDIO demo 保证 SDIO 数据传输正常，再测试 SDIO AT。

5.10.3 SDIO 交互流程

Host 侧

1. SDIO slave 模组上电（此步骤仅针对 ESP-SDIO-TESTBOARD-V1 开发板）
   - ESP-SDIO-TESTBOARD-V1 包含了一个 master 和 3 个 slave（ESP32,ESP8266 以及 ESP8089）
   - 如果使用 ESP32 作为 SDIO slave，需要将 GPIO5 拉低，参见 `slave_power_on`。
2. 初始化 SDIO host
   - SDIO host 初始化主要是 SDIO 协议的初始化，包括设置 1 线或者 4 线，SDIO 频率，初始化 SD mode。
3. 协商 SDIO 通讯
   - 这部分主要按照 SDIO spec 的要求，跟 SDIO slave 协商参数。
   - 特别需要注意的是，如果 SDIO host 和 slave 同时重启，那么，协商需要等待 slave 初始化完成后再开始。一般 host 会在启动时添加延时，等待 slave 启动完成，再开始协商 SDIO 通信。
4. 接收数据
   - 接收数据主要依靠监测 DAT1 的中断信号。当接收到中断信号后，host 读取中断源并判断中断信号，如果中断是 slave 有数据要发送，host 会调用 CMD53 读取 slave 的数据。

5. 发送数据

   - SDIO AT DEMO 中，发送数据通过串口输入，然后 host 调用 CMD53 将数据发送过去。
   - 需要注意的是，如果发送超时，那么有可能 slave 侧出现异常，此时 host 和 slave 需要重新初始化 SDIO。
   - 在调用 AT+RST 或者 AT+RESTORE 命令后，host 和 slave 也同样需要重新初始化 SDIO。

Slave 侧

SDIO slave 的处理与 SDIO host 类似，slave 在接收到 SDIO host 发送的数据后，通知 AT core 并将数据发送给 AT core 进行处理，在 AT core 处理完成后，再发送数据给 SDIO host。

1. 初始化 SDIO slave

   - 调用 `sdio_slave_initialize` 初始化 SDIO slave driver
   - 调用 `sdio_slave_recv_register_buf` 注册接收用的 buffer，为了加快接收速度，此处注册了多个接收 buffer。
   - 调用 `sdio_slave_recv_load_buf` 加载刚刚注册的 buffer，准备接收数据
   - `sdio_slave_set_host_intena` 用于设置 host 可用中断，主要用到的是新数据包发送中断 `SDIO_SLAVE_HOSTINT_SEND_NEW_PACKET`
   - 调用 `sdio_slave_start` 在硬件上开始接收和发送

2. 发送数据

   - 因为 SDIO slave 发送的数据需要保证能被 DMA 访问，所以需要先把 AT 中的数据拷贝到可被 DMA 访问的内存中，然后调用 sdio_slave_transmit 进行发送。

3. 接收数据

   - 为了优化接收 SDIO 数据传输给 AT core 的速率，在调用 `sdio_slave_recv` 接收 SDIO 数据后，使用了循环链表将接收到的数据传输到 AT core。

5.11 How to implement OTA update

The following steps guide the users in creating a device on iot.espressif.cn and updating the OTA BIN on it.

1. Open the website http://iot.espressif.cn. If using SSL OTA, it should be https://iot.espressif.cn.
2. Click "Join" in the upper right corner of the webpage, and enter your name, email address, and password.
3. Click on "Device" in the upper right corner of the webpage, and click on "Create" to create a device.
4. A key is generated when the device is successfully created, as the figure below shows.

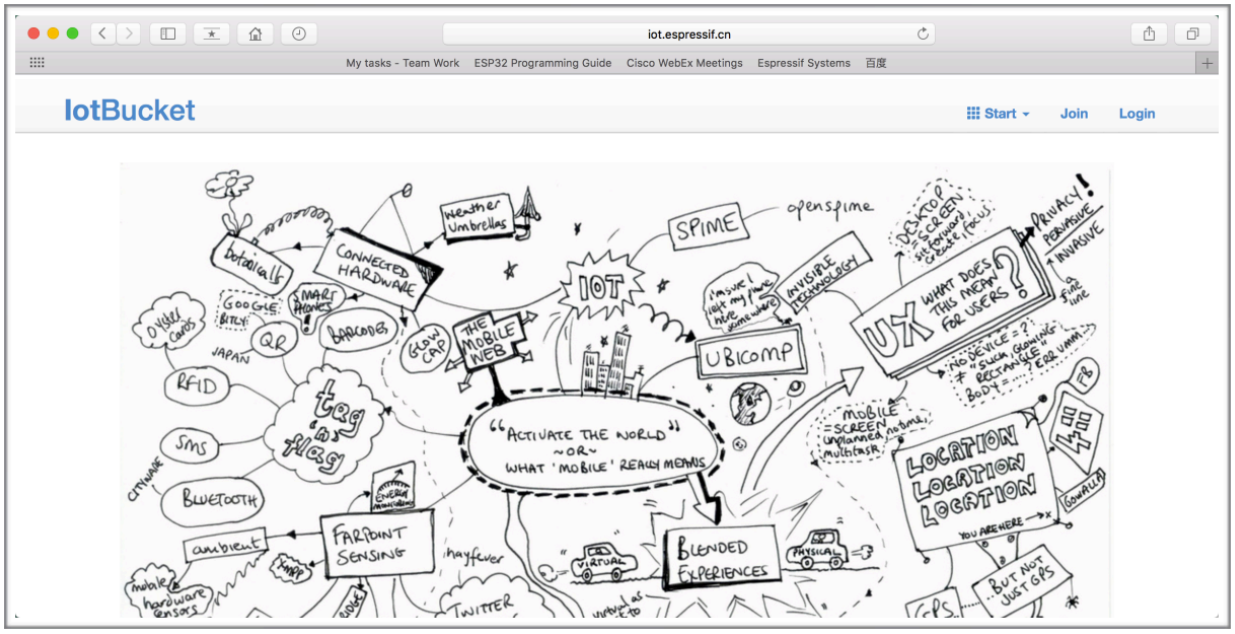


图 1: Open iot.espressif.cn website

图 2: Join iot.espressif.cn website

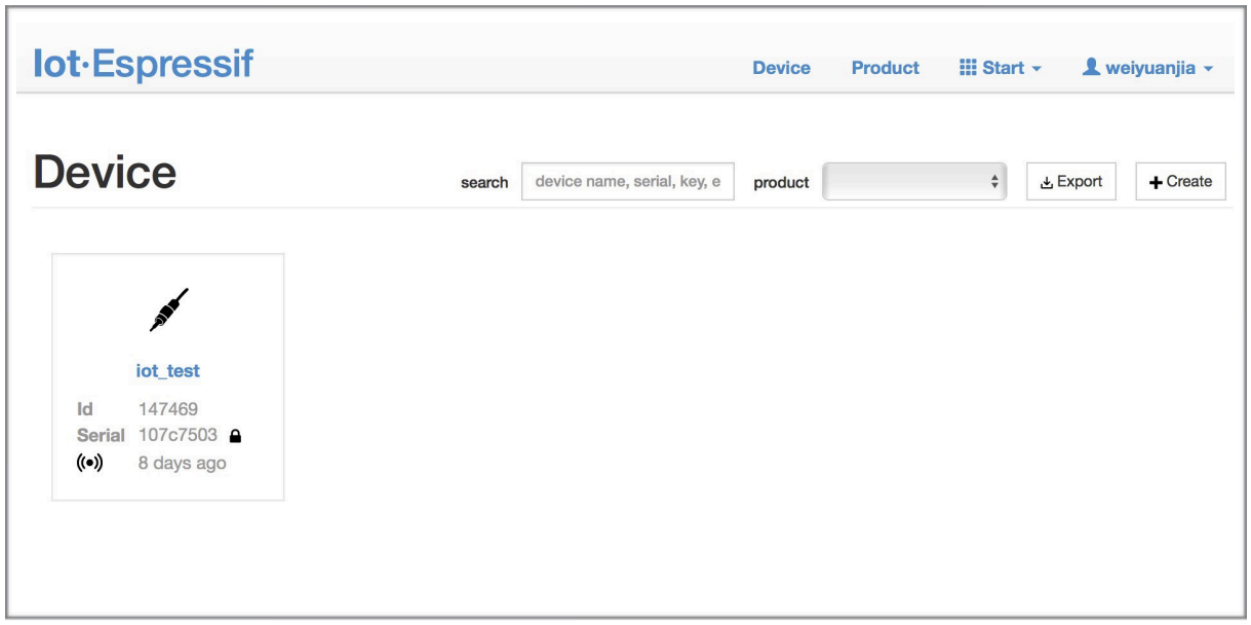


图 3: Click on "Device"

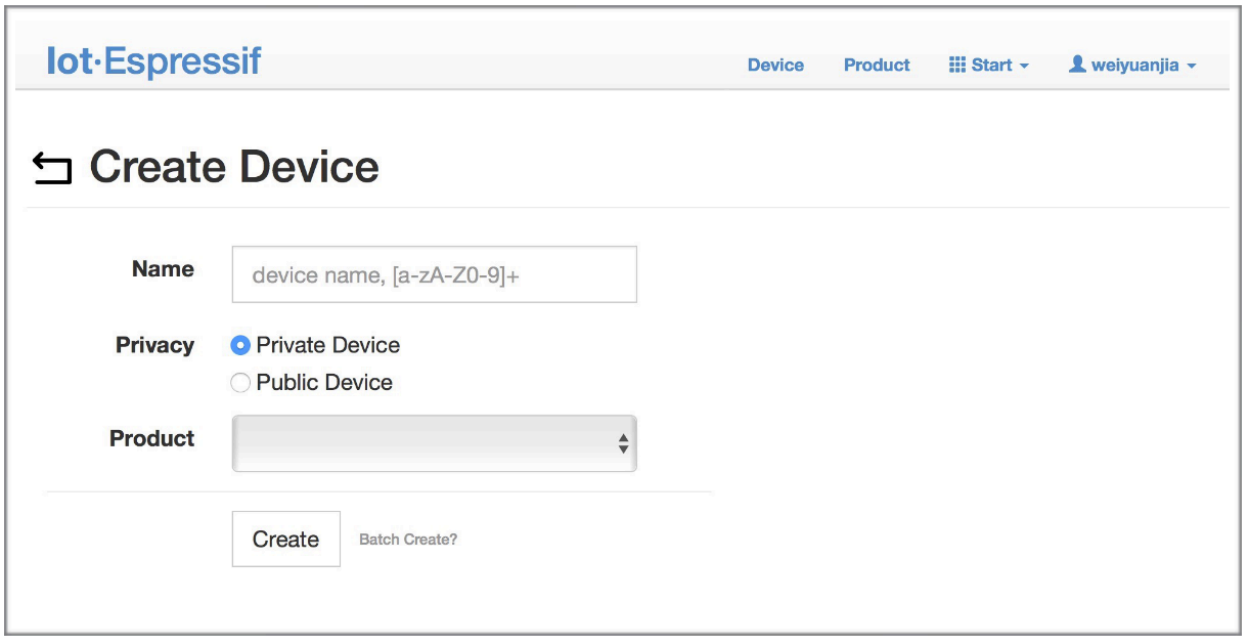


图 4: Click on "Create"

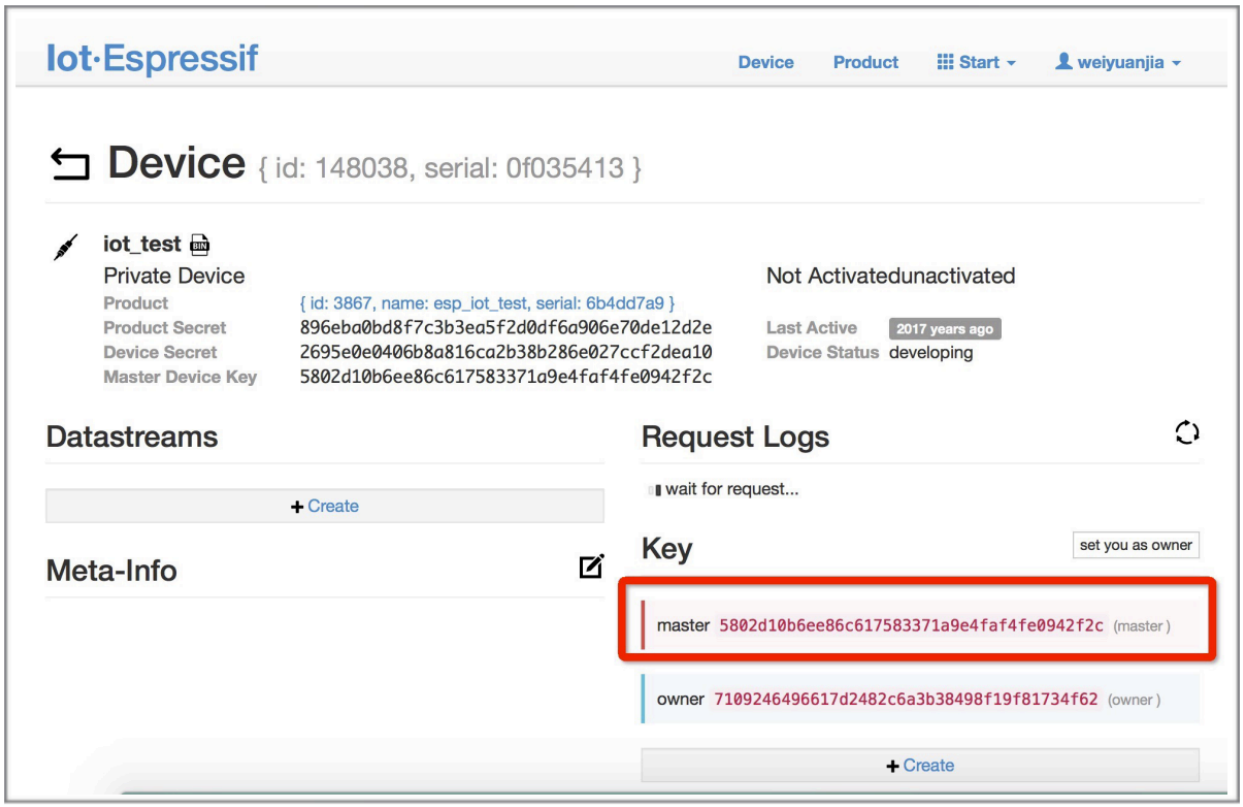


图 5: A key has been generated

5. Use the key to compile your own OTA BIN. The process of configuring the AT OTA token key is as follows:

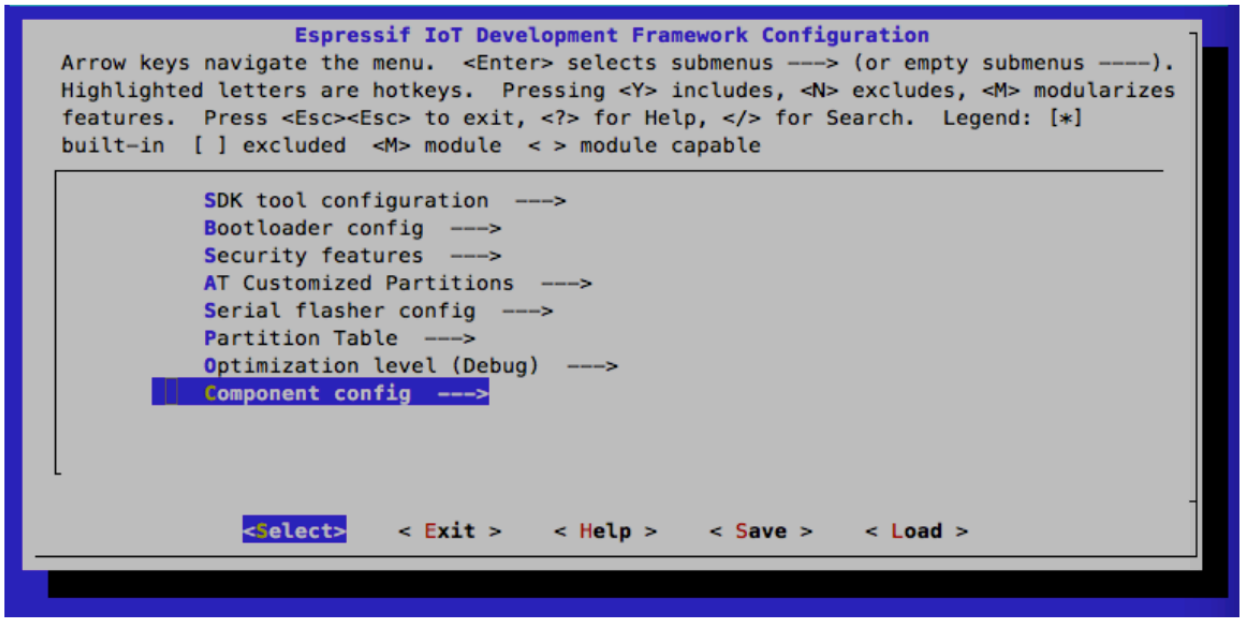


图 6: Configuring the AT OTA token key - Step 1

注解: If using SSL OTA, the option "OTA based upon ssl" should be selected.

---

6. Click on "Product" to enter the webpage, as shown below. Click on the device created. Enter version and corename under "ROM Deploy". Rename the BIN compiled in Step 5 as "ota.bin" and save the configuration.
7. Click on the ota.bin to save it as the current version.
8. Run the command AT+CIUPDATE on the ESP device. If the network is connected, OTA update will be done.

5.12 如何更新 esp-idf 版本

当前工程默认支持 ESP32 UART AT 和 ESP8266 UART AT 平台，每个平台对应一套配置文件，配置文件目录可以通过 Makefile 文件中的 `ESP_AT_MODULE_CONFIG_DIR` 变量指定，默认 ESP32 UART AT 配置目录为 `module_config/module_esp32_default`，ESP8266 UART AT 配置文件目录为 `module_config/module_esp8266_default`，具体的版本信息在配置目录下的 IDF_VERSION 文件中，如 ESP32 平台的配置信息为：

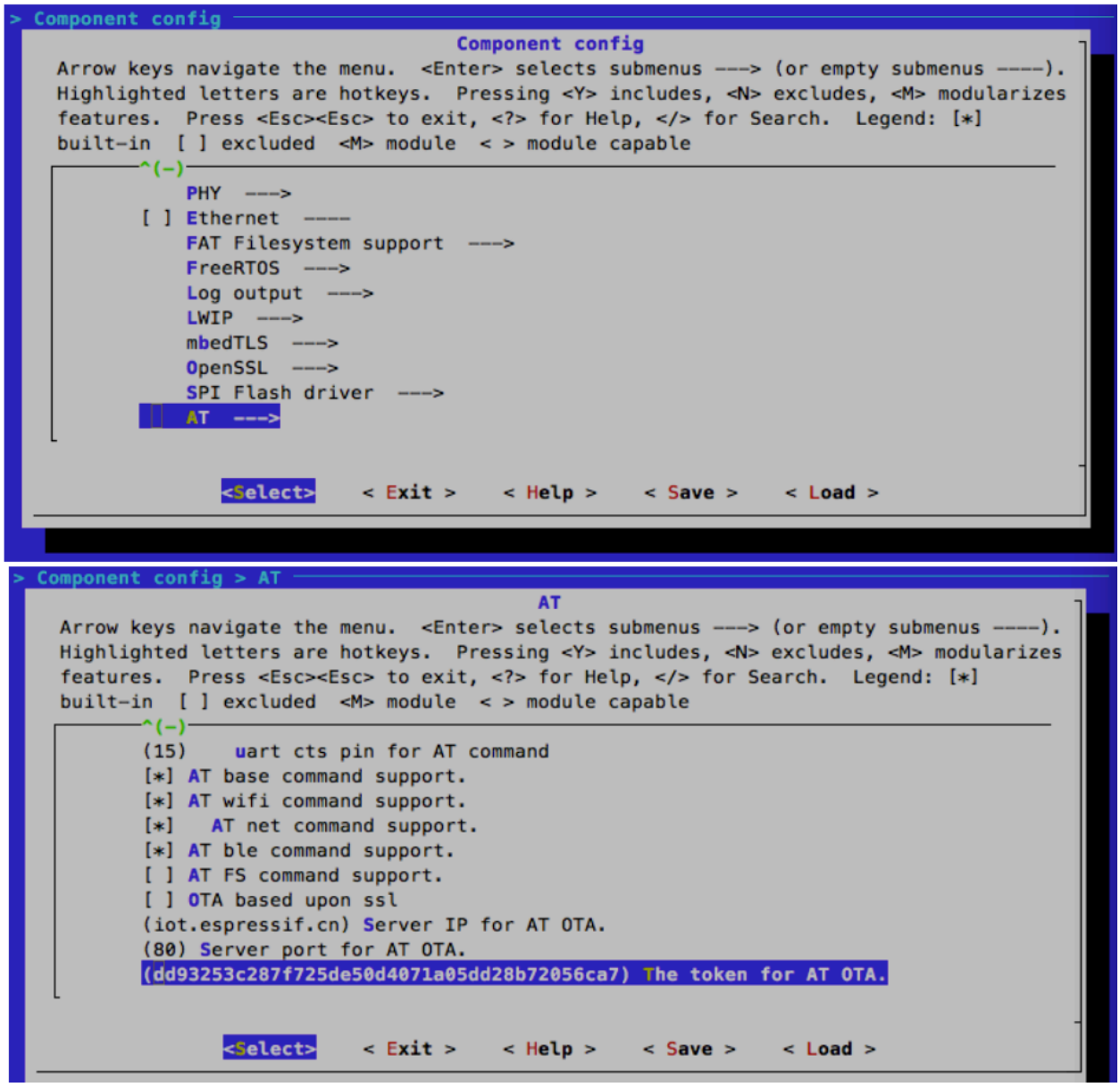


图 7: Configuring the AT OTA token key - Step 2 and 3

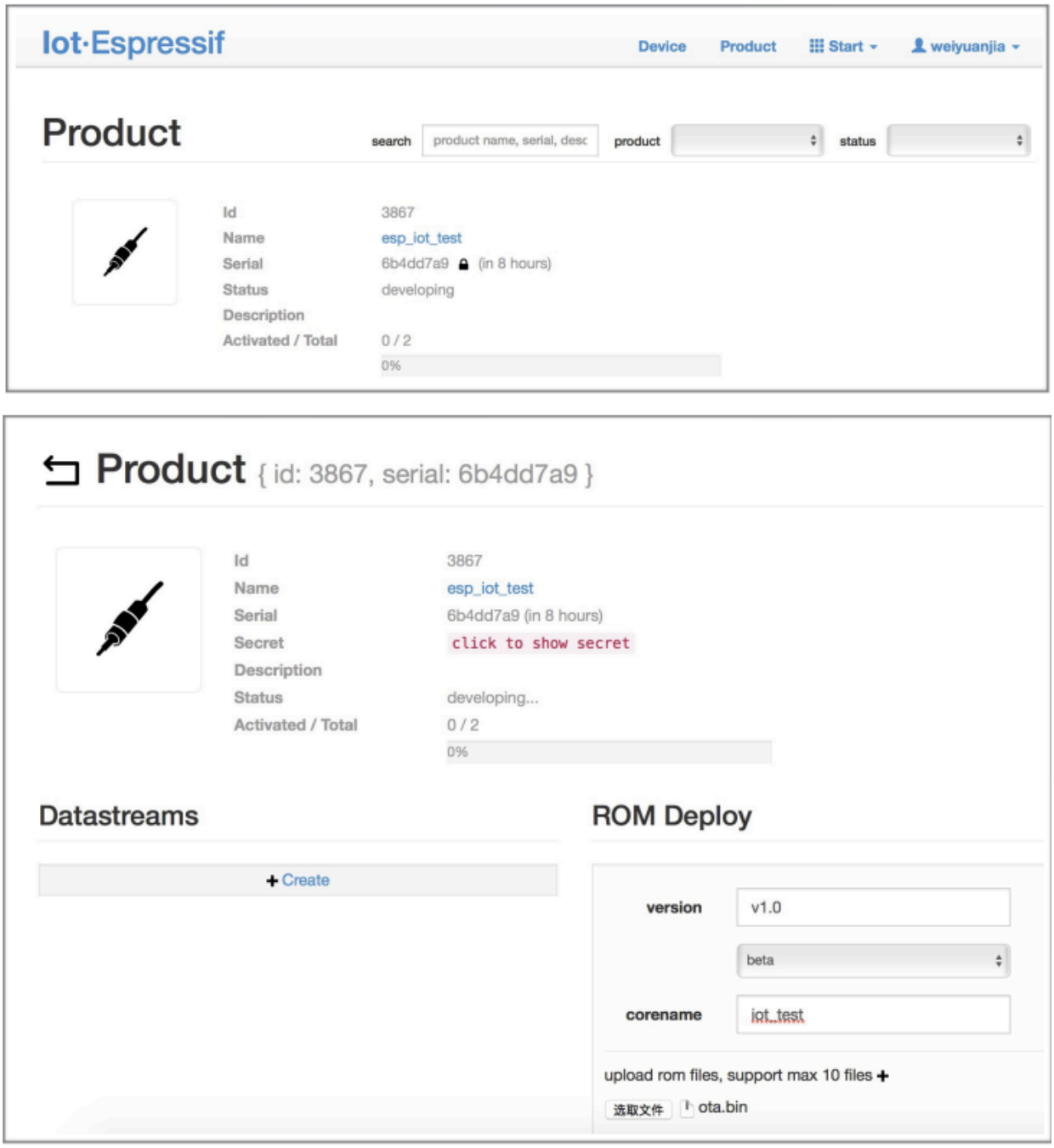


图 8: Enter version and corename

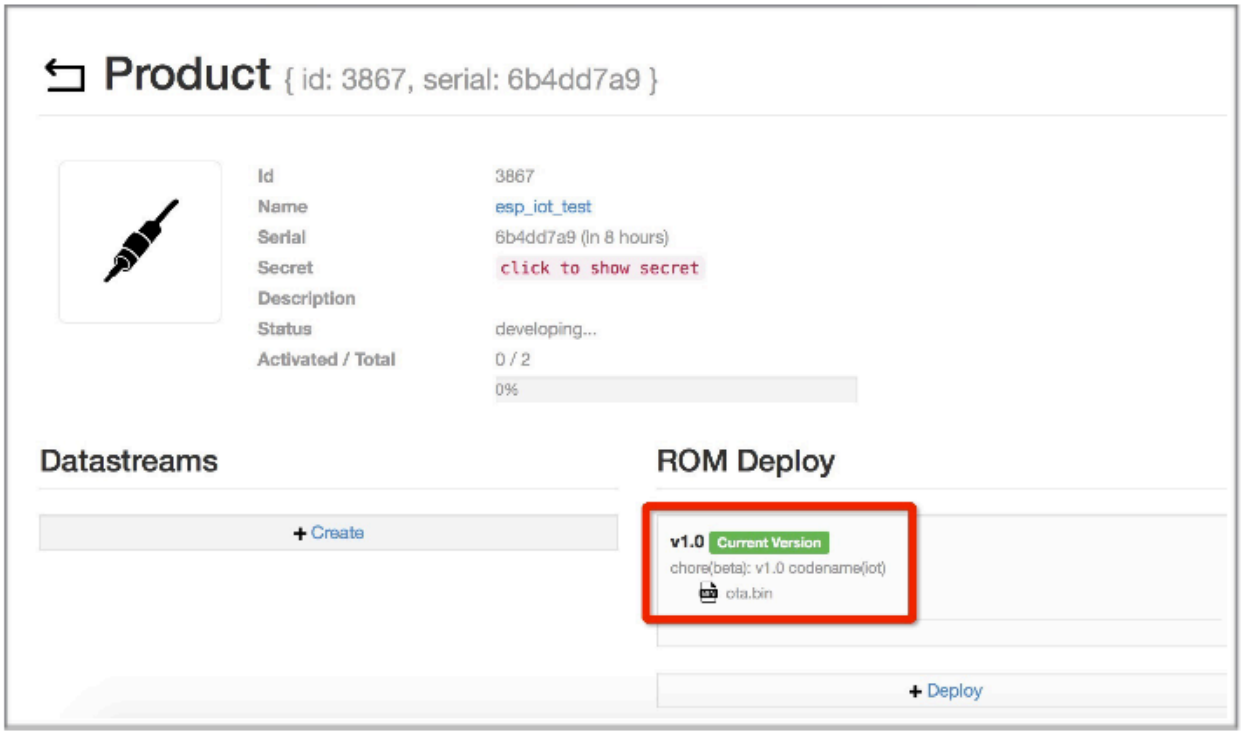


图 9: Save the current version of ota.bin

```
branch:master
commit:7fa98593bc179ea50a1bc8244d5b94bac59c9a10
repository:https://github.com/espressif/esp-idf.git
```

第一行的 branch 代表当前所使用 idf 的分支名称第二行的 commit 代表当前所使用 idf 的 commit id 第三行的 repository 代表当前所使用 idf 的仓库 url

如果想要更新时，只需要将上面的 branch、commit 和 repository 修改为自己想要的分支即可。

当更新 *esp-idf* 版本信息后，建议将当前工程下的 *esp-idf* 目录删除，下一次编译时，会重新 *clone* 新的 *esp-idf*

注意：如果 repository url 发生了变化，必须将当前工程下的 esp-idf 目录删除，否则会更新失败。

5.13 AT API Reference

5.13.1 Header File

- at/include/esp_at.h

5.13.2 Functions

void **esp_at_module_init**(uint32_t *netconn_max*, **const** uint8_t **custom_version*)

This function should be called only once, before any other AT functions are called.

Parameters

- `netconn_max`: the maximum number of the link in the at module
- `custom_version`: version information by custom

esp_at_para_parse_result_type **esp_at_get_para_as_digit**(int32_t *para_index*, int32_t **value*)

Parse digit parameter from command string.

Return

- ESP_AT_PARA_PARSE_RESULT_OK : succeed
- ESP_AT_PARA_PARSE_RESULT_FAIL : fail
- ESP_AT_PARA_PARSE_RESULT_OMITTED : this parameter is OMITTED

Parameters

- `para_index`: the index of parameter
- `value`: the value parsed

esp_at_para_parse_result_type **esp_at_get_para_as_str**(int32_t *para_index*, uint8_t ***result*)

Parse string parameter from command string.

Return

- ESP_AT_PARA_PARSE_RESULT_OK : succeed
- ESP_AT_PARA_PARSE_RESULT_FAIL : fail
- ESP_AT_PARA_PARSE_RESULT_OMITTED : this parameter is OMITTED

Parameters

- `para_index`: the index of parameter
- `result`: the pointer that point to the result.

void **esp_at_port_recv_data_notify_from_isr**(int32_t *len*)

Calling the esp_at_port_recv_data_notify_from_isr to notify at module that at port received data. When received this notice,at task will get data by calling get_data_length and read_data in esp_at_device_ops. This function MUST be used in isr.

Parameters

- `len`: data length

bool **esp_at_port_recv_data_notify**(int32_t *len*, uint32_t *msec*)

Calling the esp_at_port_recv_data_notify to notify at module that at port received data. When received this notice,at task will get data by calling get_data_length and read_data in esp_at_device_ops. This function MUST NOT be used in isr.

Return

- true : succeed
- false : fail

Parameters

- `len`: data length
- `msec`: timeout time,The unit is millisecond. It waits forever,if msec is portMAX_DELAY.

void **esp_at_transmit_terminal_from_isr**(void)

terminal transparent transmit mode,This function MUST be used in isr.

void **esp_at_transmit_terminal**(void)

terminal transparent transmit mode,This function MUST NOT be used in isr.

bool **esp_at_custom_cmd_array_regist**(const *esp_at_cmd_struct* *custom_at_cmd_array, uint32_t *cmd_num*)

regist at command set, which defined by custom,

Parameters

- `custom_at_cmd_array`: at command set
- `cmd_num`: command number

void **esp_at_device_ops_regist**(*esp_at_device_ops_struct* *ops)

regist device operate functions set,

Parameters

- `ops`: device operate functions set

bool **esp_at_custom_net_ops_regist**(int32_t *link_id*, *esp_at_custom_net_ops_struct* *ops)

bool **esp_at_custom_ble_ops_regist**(int32_t *conn_index*, *esp_at_custom_ble_ops_struct* *ops)

void **esp_at_custom_ops_regist**(*esp_at_custom_ops_struct* *ops)

regist custom operate functions set interacting with AT,

Parameters

- `ops`: custom operate functions set

uint32_t **esp_at_get_version**(void)

get at module version number,

Return at version bit31~bit24: at main version bit23~bit16: at sub version bit15~bit8 : at test version bit7~bit0 : at custom version

void **esp_at_response_result**(uint8_t *result_code*)

response AT process result,

Parameters

- `result_code`: see esp_at_result_code_string_index

int32_t **esp_at_port_write_data**(uint8_t **data*, int32_t *len*)

write data into device,

Return

- >= 0 : the real length of the data written
- others : fail.

Parameters

- `data`: data buffer to be written
- `len`: data length

int32_t **esp_at_port_read_data**(uint8_t **data*, int32_t *len*)

read data from device,

Return

- >= 0 : the real length of the data read from device
- others : fail

Parameters

- `data`: data buffer
- `len`: data length

bool **esp_at_port_wait_write_complete**(int32_t *timeout_msec*)

wait for transmitting data completely to peer device,

Return

- true : succeed,transmit data completely
- false : fail

Parameters

- `timeout_msec`: timeout time,The unit is millisecond.

int32_t **esp_at_port_get_data_length**(void)

get the length of the data received,

Return

- >= 0 : the length of the data received
- others : fail

bool **esp_at_base_cmd_regist**(void)

regist at base command set. If not,you can not use AT base command

bool **esp_at_wifi_cmd_regist**(void)

regist at wifi command set. If not,you can not use AT wifi command

bool **esp_at_net_cmd_regist**(void)

regist at net command set. If not,you can not use AT net command

bool **esp_at_mdns_cmd_regist**(void)

regist at mdns command set. If not,you can not use AT mdns command

bool **esp_at_wps_cmd_regist**(void)

regist at wps command set. If not,you can not use AT wps command

bool **esp_at_smartconfig_cmd_regist**(void)

regist at smartconfig command set. If not,you can not use AT smartconfig command

bool **esp_at_ping_cmd_regist**(void)

regist at ping command set. If not,you can not use AT ping command

bool **esp_at_http_cmd_regist**(void)

regist at http command set. If not,you can not use AT http command

bool **esp_at_mqtt_cmd_regist**(void)

regist at mqtt command set. If not,you can not use AT mqtt command

bool **esp_at_ble_cmd_regist**(void)

regist at ble command set. If not,you can not use AT ble command

bool **esp_at_ble_hid_cmd_regist**(void)

regist at ble hid command set. If not,you can not use AT ble hid command

bool **esp_at_blufi_cmd_regist**(void)

regist at blufi command set. If not,you can not use AT blufi command

bool **esp_at_bt_cmd_regist**(void)

regist at bt command set. If not,you can not use AT bt command

bool **esp_at_bt_spp_cmd_regist**(void)
regist at bt spp command set. If not,you can not use AT bt spp command

bool **esp_at_bt_a2dp_cmd_regist**(void)
regist at bt a2dp command set. If not,you can not use AT bt a2dp command

bool **esp_at_fs_cmd_regist**(void)
regist at fs command set. If not,you can not use AT fs command

bool **esp_at_eap_cmd_regist**(void)
regist at WPA2 Enterprise AP command set. If not,you can not use AT EAP command

bool **esp_at_eth_cmd_regist**(void)
regist at ethernet command set. If not,you can not use AT ethernet command

bool **esp_at_custom_cmd_line_terminator_set**(uint8_t **terminator*)
Set AT command terminator, by default, the terminator is "\r\n" You can change it by calling this function, but it just supports one character now.

Return

- true : succeed,transmit data completely
- false : fail

Parameters

- `terminator`: the line terminator

uint8_t ***esp_at_custom_cmd_line_terminator_get**(void)
Get AT command line terminator,by default, the return string is "\r\n" .

Return the command line terminator

const esp_partition_t ***esp_at_custom_partition_find**(esp_partition_type_t *type*, esp_partition_subtype_t *subtype*, **const** char **label*)
Find the partition which is defined in at_customize.csv.

Return pointer to esp_partition_t structure, or NULL if no partition is found. This pointer is valid for the lifetime of the application

Parameters

- `type`: the type of the partition
- `subtype`: the subtype of the partition
- `label`: Partition label

void **esp_at_port_enter_specific**(*esp_at_port_specific_callback_t callback*)

Set AT core as specific status, it will call callback if receiving data. for example:

static void wait_data_callback (void) { xSemaphoreGive(sync_sema); }

void process_task(void* para) { vSemaphoreCreateBinary(sync_sema); xSemaphoreTake(sync_sema,portMAX_DELAY); esp_at_port_write_data((uint8_t *)" >" ,strlen(">")); esp_at_port_enter_specific(wait_data_callback); while(xSemaphoreTake(sync_sema,portMAX_DELAY)) { len = esp_at_port_read_data(data, data_len); TODO: } }

Parameters

- `callback`: which will be called when received data from AT port

void **esp_at_port_exit_specific**(void)

Exit AT core as specific status.

const uint8_t ***esp_at_get_current_cmd_name**(void)

Get current AT command name.

esp_err_t **esp_at_wifi_event_handler**(void **ctx*, system_event_t **event*)

Wi-Fi event handler callback, which used in AT core.

Return

- ESP_OK: succeed
- others: fail

Parameters

- `ctx`: reserved for user
- `event`: event type defined in this file

5.13.3 Structures

struct esp_at_cmd_struct

esp_at_cmd_struct used for define at command

Public Members

char ***at_cmdName**

at command name

uint8_t (***at_testCmd**)(uint8_t *cmd_name)

Test Command function pointer

uint8_t (*`at_queryCmd`)(uint8_t *cmd_name)
Query Command function pointer

uint8_t (*`at_setupCmd`)(uint8_t para_num)
Setup Command function pointer

uint8_t (*`at_exeCmd`)(uint8_t *cmd_name)
Execute Command function pointer

`struct esp_at_device_ops_struct`
esp_at_device_ops_struct device operate functions struct for AT

Public Members

int32_t (*`read_data`)(uint8_t *data, int32_t len)
read data from device

int32_t (*`write_data`)(uint8_t *data, int32_t len)
write data into device

int32_t (*`get_data_length`)(void)
get the length of data received

bool (*`wait_write_complete`)(int32_t timeout_msec)
wait write finish

`struct esp_at_custom_net_ops_struct`
esp_at_custom_net_ops_struct custom socket callback for AT

Public Members

int32_t (*`recv_data`)(uint8_t *data, int32_t len)
callback when socket received data

void (*`connect_cb`)(void)
callback when socket connection is built

void (*`disconnect_cb`)(void)
callback when socket connection is disconnected

`struct esp_at_custom_ble_ops_struct`
esp_at_custom_ble_ops_struct custom ble callback for AT

Public Members

int32_t (*`recv_data`)(uint8_t *data, int32_t len)
callback when ble received data

void (*connect_cb)(void)
callback when ble connection is built

void (*disconnect_cb)(void)
callback when ble connection is disconnected

struct esp_at_custom_ops_struct
esp_at_ops_struct some custom function interacting with AT

Public Members

void (*status_callback)(*esp_at_status_type* status)
callback when AT status changes

void (*pre_deepsleep_callback)(void)
callback before enter deep sleep

void (*pre_restart_callback)(void)
callback before restart

5.13.4 Macros

ESP_AT_ERROR_NO(subcategory, extension)

ESP_AT_CMD_ERROR_OK

ESP_AT_CMD_ERROR_NON_FINISH

ESP_AT_CMD_ERROR_NOT_FOUND_AT

ESP_AT_CMD_ERROR_PARA_LENGTH(which_para)

ESP_AT_CMD_ERROR_PARA_TYPE(which_para)

ESP_AT_CMD_ERROR_PARA_NUM(need, given)

ESP_AT_CMD_ERROR_PARA_INVALID(which_para)

ESP_AT_CMD_ERROR_PARA_PARSE_FAIL(which_para)

ESP_AT_CMD_ERROR_CMD_UNSUPPORT

ESP_AT_CMD_ERROR_CMD_EXEC_FAIL(result)

ESP_AT_CMD_ERROR_CMD_PROCESSING

ESP_AT_CMD_ERROR_CMD_OP_ERROR

5.13.5 Type Definitions

typedef void (***esp_at_port_specific_callback_t**)(void)

AT specific callback type.

5.13.6 Enumerations

enum esp_at_status_type

esp_at_status some custom function interacting with AT

Values:

ESP_AT_STATUS_NORMAL = 0x0

Normal mode.Now mcu can send AT command

ESP_AT_STATUS_TRANSMIT

Transparent Transmition mode

enum esp_at_module

module number,Now just AT module

Values:

ESP_AT_MODULE_NUM = 0x01

enum esp_at_error_code

subcategory number

Values:

ESP_AT_SUB_OK = 0x00

OK

ESP_AT_SUB_COMMON_ERROR = 0x01

ESP_AT_SUB_NO_TERMINATOR = 0x02

not end with "\r\n"

ESP_AT_SUB_NO_AT = 0x03

not found AT or at or At or aT

ESP_AT_SUB_PARA_LENGTH_MISMATCH = 0x04

parameter length not match

ESP_AT_SUB_PARA_TYPE_MISMATCH = 0x05

parameter length not match

ESP_AT_SUB_PARA_NUM_MISMATCH = 0x06

parameter number not match

ESP_AT_SUB_PARA_INVALID = 0x07

ESP_AT_SUB_PARA_PARSE_FAIL = 0x08
parse parameter fail

ESP_AT_SUB_UNSUPPORT_CMD = 0x09

ESP_AT_SUB_CMD_EXEC_FAIL = 0x0A

ESP_AT_SUB_CMD_PROCESSING = 0x0B
previous command is processing

ESP_AT_SUB_CMD_OP_ERROR = 0x0C

enum esp_at_para_parse_result_type
the result of AT parse

Values:

ESP_AT_PARA_PARSE_RESULT_FAIL = -1
parse fail,Maybe the type of parameter is mismatched,or out of range

ESP_AT_PARA_PARSE_RESULT_OK = 0
Successful

ESP_AT_PARA_PARSE_RESULT_OMITTED
the parameter is OMITTED.

enum esp_at_result_code_string_index
the result code of AT command processing

Values:

ESP_AT_RESULT_CODE_OK = 0x00
"OK"

ESP_AT_RESULT_CODE_ERROR = 0x01
"ERROR"

ESP_AT_RESULT_CODE_FAIL = 0x02
"ERROR"

ESP_AT_RESULT_CODE_SEND_OK = 0x03
"SEND OK"

ESP_AT_RESULT_CODE_SEND_FAIL = 0x04
"SEND FAIL"

ESP_AT_RESULT_CODE_IGNORE = 0x05
response nothing

ESP_AT_RESULT_CODE_PROCESS_DONE = 0x06
response nothing

ESP_AT_RESULT_CODE_MAX

- genindex

索引

E