ESP-AT User Guide

Read the Docs

2023 年 07 月 21 日

Contents

1 入门指南 **3**

1.1 ESP-AT 是什么 . . . 3
1.2 硬件连接 . . . 4
1.3 下载指导 . . . 13

2 AT 固件 **25**

2.1 发布的固件 . . . 25
2.2 发布的固件 . . . 26
2.3 发布的固件 . . . 26

3 AT 命令集 **29**

3.1 基础 AT 命令 . . . 29
3.2 Wi-Fi AT 命令集 . . . 57
3.3 TCP/IP AT 命令 . . . 91
3.4 [ESP32 Only] Bluetooth® Low Energy AT 命令集 . . . 133
3.5 [ESP32 Only] Classic Bluetooth® AT 命令集 . . . 184
3.6 MQTT AT Commands . . . 207
3.7 HTTP AT 命令集 . . . 220
3.8 [ESP32 Only] 以太网 AT 命令 . . . 225
3.9 [ESP8266 Only] 信令测试 AT 命令 . . . 229
3.10 [ESP32 & ESP32-S2 & ESP32-C3] 驱动 AT 命令 . . . 229
3.11 Web server AT Commands . . . 241
3.12 ESP-AT 不同版本命令集支持对比 . . . 243
3.13 AT 命令分类 . . . 249
3.14 参数信息保存在 flash 中的 AT 命令 . . . 250
3.15 AT 消息 . . . 250

4 AT 命令示例 **253**

4.1 TCP-IP AT Examples . . . 253
4.2 [ESP32 Only] BLE AT Examples . . . 253
4.3 MQTT AT Examples . . . 253
4.4 [ESP32 Only] Ethernet AT 示例 . . . 254
4.5 Web Server AT 示例 . . . 254

5 如何编译和开发自己的 AT 工程 277
5.1 Build Your Own ESP-AT Project . . . 277
5.2 如何修改 `AT port` 管脚 . . . 285
5.3 如何添加自定义 AT 命令 . . . 290
5.4 如何创建默认出厂参数 bin 文件 . . . 290
5.5 如何自定义 Ble services . . . 290
5.6 如何自定义分区 . . . 291
5.7 如何使用 ESP-AT 经典蓝牙 . . . 292
5.8 How to enable ESP-AT Ethernet . . . 292
5.9 如何增加一个新的平台支持 . . . 292
5.10 ESP32 SDIO AT 指南 . . . 294
5.11 SPI AT 指南 . . . 296
5.12 How to implement OTA update . . . 296
5.13 如何更新 esp-idf 版本 . . . 302
5.14 如何了解 ESP-AT 同一平台不同模组差异 . . . 303
5.15 AT API Reference . . . 303

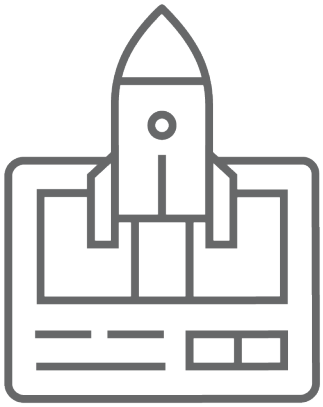
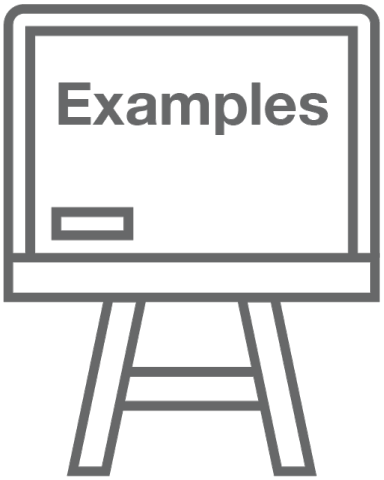
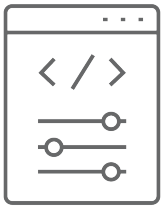
6 第三方定制化 AT 命令和固件 317
6.1 腾讯云 IoT AT 命令和固件 . . . 317

7 Index of Abbreviations 379

索引 385

[English]

本文档为 ESP-AT 官方文档。

		
入门	AT Binary 列表	AT 命令集
		
AT 命令示例	编译和开发	第三方定制化 AT 命令和固件

CHAPTER 1

入门指南

[English]

本指南详细介绍 ESP-AT 是什么、如何连接硬件、以及如何下载和烧录 AT 固件，由以下章节组成：

1.1 ESP-AT 是什么

[English]

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，您可以快速加入无线网络、连接云平台、实现数据通信以及远程控制等功能，真正的通过无线通讯实现万物互联。

ESP-AT 是基于 ESP-IDF 或 ESP8266_RTOS_SDK 实现的软件工程。它使 ESP 模组作为从机，MCU 作为主机。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 提供了大量功能不同的 AT 命令，如 Wi-Fi 命令、TCP/IP 命令、Bluetooth LE 命令、Bluetooth 命令、MQTT 命令、HTTP 命令、Ethernet 命令等。

> **注意：** 当前的 ESP-AT 工程基于 ESP-IDF 或 ESP8266_RTOS_SDK，不基于 ESP8266 NonOS SDK。

AT 命令以 “AT” 开始，代表 Attention，以新的一行 (CR LF) 为结尾。输入的每条命令都会返回 `OK` 或 `ERROR` 的响应，表示当前命令的最终执行结果。注意，所有 AT 命令均为串行执行，每次只能执行一条命令。因此，在使用 AT 命令时，应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...` 提示。更多有关 AT 命令的信息可参见*AT 命令集*。

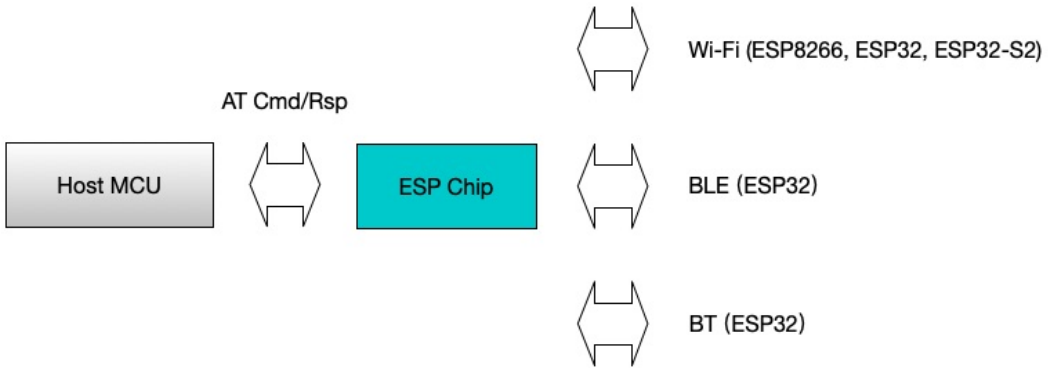


图 1: ESP-AT 概览

默认配置下，MCU 通过 UART 连接至 ESP 模组、发送 AT 命令以及接收 AT 响应。但是，您也可以根据实际使用情况修改程序，使用其他的通信接口，例如 SDIO。

同样，您也可以基于 ESP-AT 工程，自行开发更多的 AT 命令，以实现更多的功能。

1.2 硬件连接

[English]

本文档主要介绍下载和烧录 AT 固件、发送 AT 命令和接收 AT 响应所需要的硬件以及硬件之间该如何连接，主要涉及以下 ESP 系列的模组：

- *ESP32 系列*
- *ESP32-S2 系列*
- *ESP32-C3 系列*
- *ESP8266 系列*

对于不同系列的模组，AT 默认固件所支持的命令会有所差异。具体可参考*如何了解 ESP-AT 同一平台不同模组差异*。

1.2.1 硬件准备

表 1: ESP-AT 测试所需硬件

硬件	功能
ESP 开发板	从机
USB 数据线（连接 ESP 开发板和 PC）	下载固件、输出日志数据连接
PC	主机，将固件下载至从机
USB 数据线（连接 PC 和 USB 转 UART 串口模块）	发送 AT 命令、接收 AT 响应数据连接
USB 转 UART 串口模块	转换 USB 信号和 TTL 信号
杜邦线（连接 USB 转 UART 串口模块和 ESP 开发板）	发送 AT 命令、接收 AT 响应数据连接

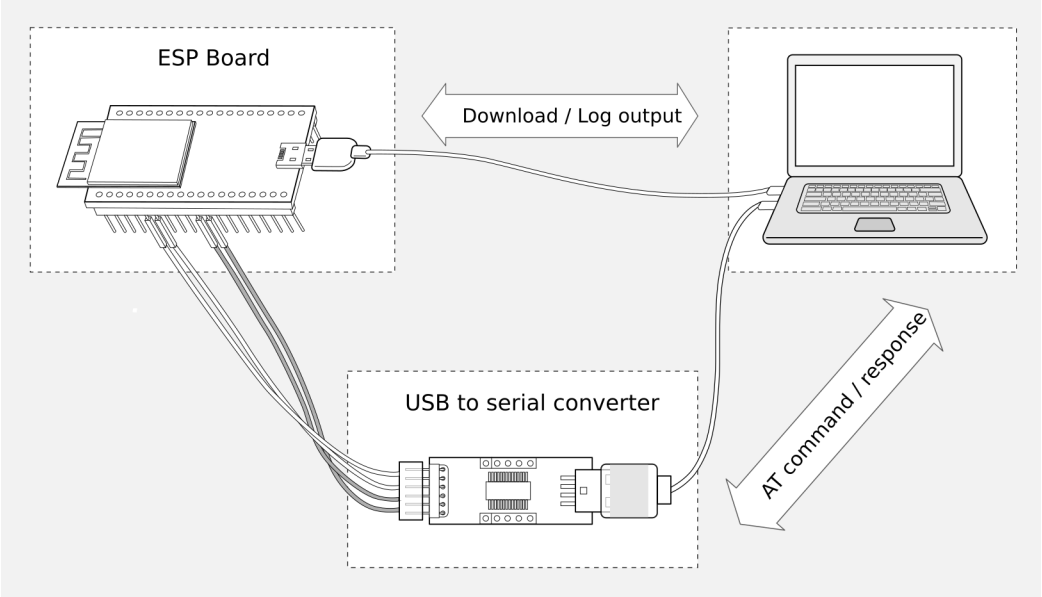


图 2: ESP-AT 测试硬件连接示意图

注意：上图使用 4 根杜邦线连接 ESP 开发板和 USB 转 UART 串口模块，但如果您不使用硬件流控功能，只需 2 根杜邦线连接 TX/RX 即可。

1.2.2 ESP32 系列

ESP32 AT 采用两个 UART 接口：UART0 用于下载固件和输出日志，UART1 用于发送 AT 命令和接收 AT 响应。

所有 ESP32 模组均连接 GPIO1 和 GPIO3 作为 UART0，但连接不同的 GPIO 作为 UART1，下文将详细介绍如何连接 ESP32 系列模组。

更多有关 ESP32 模组和开发板的信息可参考 ESP32 系列模组 和 ESP32 系列开发板。

ESP32-WROOM-32 系列

表 2: ESP32-WROOM-32 系列硬件连接管脚分配

功能	ESP 开发板管脚	其它设备管脚
下载固件/输出日志 [1]	**UART0** • GPIO3 (RX) • GPIO1 (TX)	**PC** • TX • RX
AT 命令/响应 [2]	**UART1** • GPIO16 (RX) • GPIO17 (TX) • GPIO15 (CTS) • GPIO14 (RTS)	**USB 转 UART 串口模块** • TX • RX • RTS • CTS

说明 1：ESP 开发板和 PC 之间的管脚连接已内置在 ESP 开发板上，您只需使用 USB 数据线连接开发板和 PC 即可。

说明 2：CTS/RTS 管脚只有在使用硬件流控功能时才需连接。

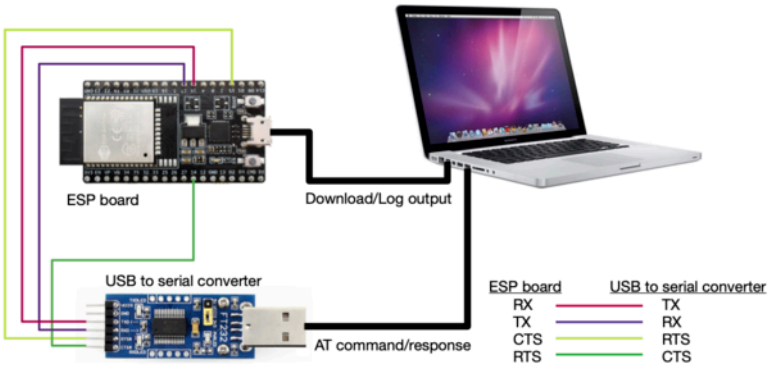


图 3: ESP32-WROOM-32 系列硬件连接示意图

如果需要直接基于 ESP32-WROOM-32 模组进行连接，请参考 《ESP32-WROOM-32 技术规格书》 。

ESP32-WROVER 系列

表 3: ESP32-WROVER 系列硬件连接管脚分配

功能	ESP 开发板管脚	其它设备管脚
下载固件/输出日志 [1]	**UART0** • GPIO3 (RX) • GPIO1 (TX)	**PC** • TX • RX
AT 命令/响应 [2]	**UART1** • GPIO19 (RX) • GPIO22 (TX) • GPIO15 (CTS) • GPIO14 (RTS)	**USB 转 UART 串口模块** • TX • RX • RTS • CTS

说明 1：ESP 开发板和 PC 之间的管脚连接已内置在 ESP 开发板上，您只需使用 USB 数据线连接开发板和 PC 即可。

说明 2：CTS/RTS 管脚只有在使用硬件流控功能时才需连接。

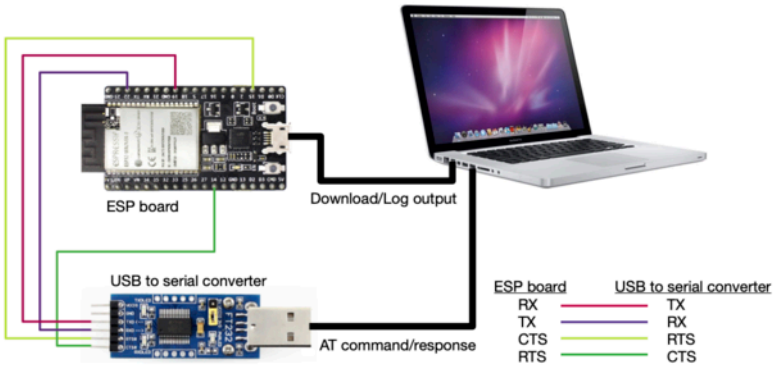


图 4: ESP32-WROVER 系列硬件连接示意图

如果需要直接基于 ESP32-WROVER 模组进行连接，请参考 《ESP32-WROVER 技术规格书》 。

ESP32-PICO 系列

表 4: ESP32-PICO 系列硬件连接管脚分配

功能	ESP 开发板管脚	其它设备管脚
下载固件/输出日志 [1]	**UART0** • GPIO3 (RX) • GPIO1 (TX)	**PC** • TX • RX
AT 命令/响应 [2]	**UART1** • GPIO19 (RX) • GPIO22 (TX) • GPIO15 (CTS) • GPIO14 (RTS)	**USB 转 UART 串口模块** • TX • RX • RTS • CTS

说明 1：ESP 开发板和 PC 之间的管脚连接已内置在 ESP 开发板上，您只需使用 USB 数据线连接开发板和 PC 即可。

说明 2：CTS/RTS 管脚只有在使用硬件流控功能时才需连接。

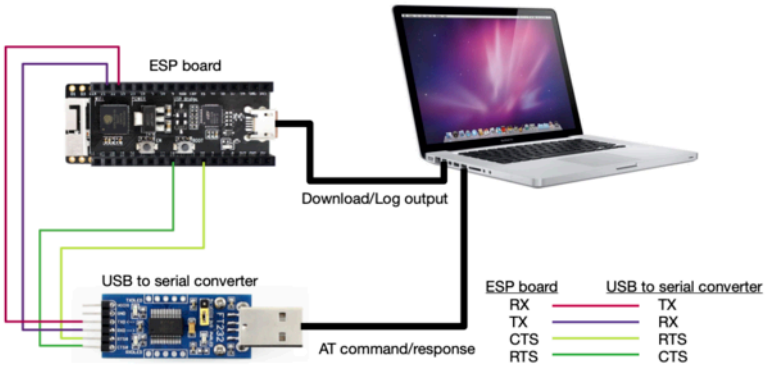


图 5: ESP32-PICO 系列硬件连接示意图

如果需要直接基于 ESP32-PICO-D4 进行连接，请参考 《ESP32-PICO-D4 技术规格书》 。

ESP32-SOLO 系列

表 5: ESP32-SOLO 系列硬件连接管脚分配

功能	ESP 开发板管脚	其它设备管脚
下载固件/输出日志 [1]	**UART0** • GPIO3 (RX) • GPIO1 (TX)	**PC** • TX • RX
AT 命令/响应 [2]	**UART1** • GPIO16 (RX) • GPIO17 (TX) • GPIO15 (CTS) • GPIO14 (RTS)	**USB 转 UART 串口模块** • TX • RX • RTS • CTS

说明 1：ESP 开发板和 PC 之间的管脚连接已内置在 ESP 开发板上，您只需使用 USB 数据线连接开发板和 PC 即可。

说明 2：CTS/RTS 管脚只有在使用硬件流控功能时才需连接。

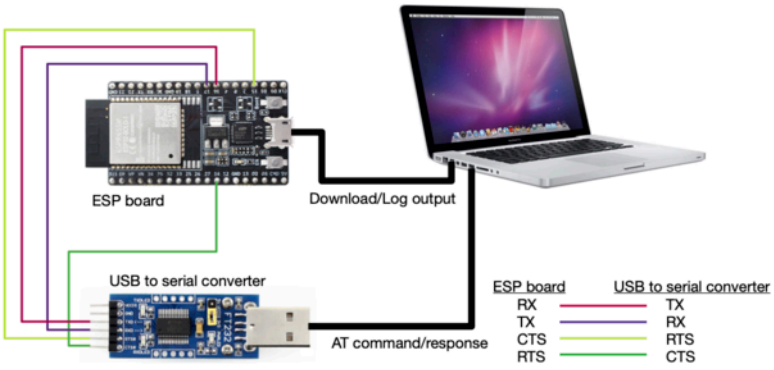


图 6: ESP32-SOLO 系列硬件连接示意图

如果需要直接基于 ESP32-SOLO-1 进行连接，请参考 《ESP32-SOLO-1 技术规格书》 。

1.2.3 ESP32-S2 系列

ESP32-S2 AT 采用两个 UART 接口：UART0 用于下载固件和输出日志，UART1 用于发送 AT 命令和接收 AT 响应。

表 6: ESP32-S2 Series 系列硬件连接管脚分配

功能	ESP 开发板管脚	其它设备管脚
下载固件/输出日志 [1]	**UART0** • GPIO44 (RX) • GPIO43 (TX)	**PC** • TX • RX
AT 命令/响应 [2]	**UART1** • GPIO21 (RX) • GPIO17 (TX) • GPIO20 (CTS) • GPIO19 (RTS)	**USB 转 UART 串口模块** • TX • RX • RTS • CTS

说明 1：ESP 开发板和 PC 之间的管脚连接已内置在 ESP 开发板上，您只需使用 USB 数据线连接开发板和 PC 即可。

说明 2：CTS/RTS 管脚只有在使用硬件流控功能时才需连接。

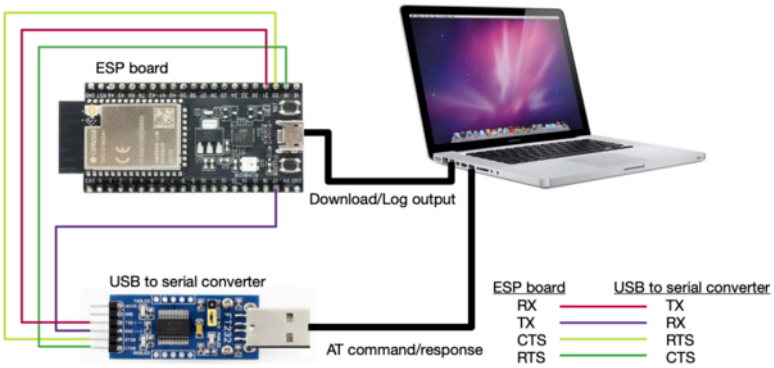


图 7: ESP32-S2 系列硬件连接示意图

如果需要直接基于 ESP32-S2-WROOM 模组进行连接，请参考 《ESP32-S2-WROOM & ESP32-S2-WROOM-I 技术规格书》 。

1.2.4 ESP32-C3 系列

ESP32-C3 AT 采用两个 UART 接口：UART0 用于下载固件和输出日志，UART1 用于发送 AT 命令和接收 AT 响应。

表 7: ESP32-C3 Series 系列硬件连接管脚分配

功能	ESP 开发板管脚	其它设备管脚
下载固件/输出日志 [1]	**UART0** • GPIO20 (RX) • GPIO21 (TX)	**PC** • TX • RX
AT 命令/响应 [2]	**UART1** • GPIO6 (RX) • GPIO7 (TX) • GPIO5 (CTS) • GPIO4 (RTS)	**USB 转 UART 串口模块** • TX • RX • RTS • CTS

说明 1：ESP 开发板和 PC 之间的管脚连接已内置在 ESP 开发板上，您只需使用 USB 数据线连接开发板和 PC 即可。

说明 2：CTS/RTS 管脚只有在使用硬件流控功能时才需连接。

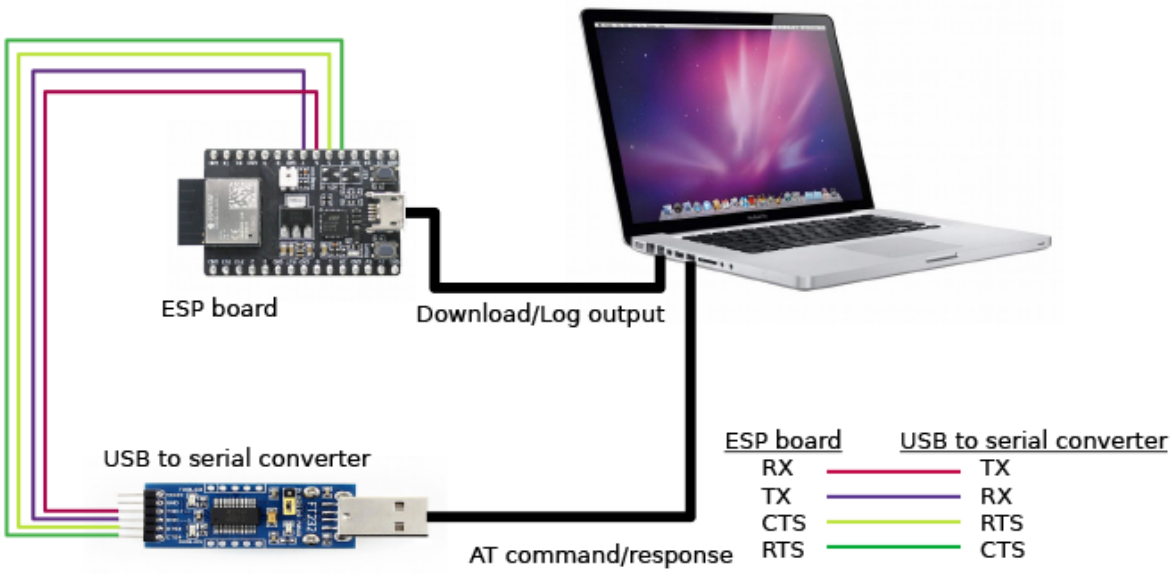


图 8: ESP32-C3 系列硬件连接示意图

如果需要直接基于 ESP32-C3-MINI-1 模组进行连接，请参考 《ESP32-C3-MINI-1 技术规格书》 。

1.2.5 ESP8266 系列

ESP8266 AT 采用两个 UART 接口：UART0 用于下载固件、发送 AT 命令以及接收 AT 响应；UART1 用于输出日志。

表 8: ESP8266 系列硬件连接管脚分配

功能	ESP 开发板管脚	其它设备管脚
下载固件	**UART0** • GPIO3 (RX) • GPIO1 (TX)	**PC** • TX • RX
AT 命令/响应 [2]	**UART0** • GPIO13 (RX) • GPIO15 (TX) • GPIO3 (CTS) • GPIO1 (RTS)	**USB 转 UART 串口模块** • TX • RX • RTS • CTS
输出日志	**UART1** • GPIO2 (TX)	**USB 转 UART 串口模块** • RX

说明 1：ESP 开发板和 PC 之间的管脚连接已内置在 ESP 开发板上，您只需使用 USB 数据线连接开发板和 PC 即可。

说明 2：CTS/RTS 管脚只有在使用硬件流控功能时才需连接。

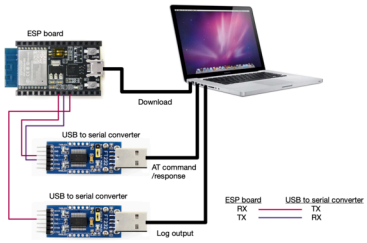


图 9: ESP8266 系列硬件连接示意图

注意： 默认配置下，基于 ESP8266_RTOS_SDK 的 ESP-WROOM-02 AT 固件具有 swap 功能，会将 RX/TX 与 CTS/RTS 进行交换。若想使用硬件流控功能，您需要断开 UART1, 从 ESP 开发板拆焊 CP2102N 芯片，并将开发板与 USB 转 UART 串口模块的 3.3 V 和 GND 相连进行供电。

如果您基于 ESP8266 芯片自行设计模组，并且采用 GPIO15/GPIO13 进行 AT 命令通讯，则您需要关注 GPIO5 管脚的走线，具体请参考 《ESP8266 硬件设计指南》 > 图 ESP8266EX UART SWAP。

ESP8266 AT 发布版固件 中，上电后会将 GPIO5 输出为高电平，以此控制 GPIO15 和 MCU 的导通状态。

如果需要直接基于 ESP-WROOM-02 或 ESP-WROOM-02D/02U 模组进行连接, 请参考 《ESP-WROOM-02 技术规格书》 或 《ESP-WROOM-02D/02U 技术规格书》 。

更多有关 ESP8266 模组的信息可参考 ESP8266 文档。

1.3 下载指导

[English]

本文档以 ESP32-WROOM-32 模组为例，介绍如何下载 ESP 模组对应的 AT 固件，以及如何将固件烧录到模组上，其它 ESP 系列模组也可参考本文档。

下载和烧录 AT 固件之前，请确保您已正确连接所需硬件，具体可参考硬件连接。

对于不同系列的模组，AT 默认固件所支持的命令会有所差异。具体可参考如何了解 *ESP-AT* 同一平台不同模组差异。

1.3.1 下载 AT 固件

请按照以下步骤将 AT 固件下载至 PC：

- 前往*AT 固件*
- 找到您的模组所对应的 AT 固件
- 点击相应链接进行下载

此处，我们下载了 ESP32-WROOM-32 对应的 `ESP32-WROOM-32_AT_Bin_V2.1` 固件，该固件的目录结构及其中各个 bin 文件介绍如下，其它 ESP 系列模组固件的目录结构及 bin 文件也可参考如下介绍：

```
.
├── at_customize.bin                // 二级分区表
├── bootloader                      // bootloader
│   └── bootloader.bin
├── customized_partitions           // AT 自定义 bin 文件
│   ├── ble_data.bin
```

(下页继续)

(续上页)

```
|   ├── 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                  // 烧录固件的参数
├── esp-at.bin                       // AT 应用固件
├── factory                          // 量产所需打包好的 bin 文件
|   ├── factory_WROOM-32.bin
|   └── factory_parameter.log
├── 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                  // 一级分区列表
|   └── partition-table.bin
└── phy_init_data.bin                // phy 初始值信息
```

其中，download.config 文件包含烧录固件的参数：

```
--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
```

- --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 40 MHz；
- `--flash_size 4MB` 代表此固件适用的 flash 最小为 4 MB；
- `0x10000 ota_data_initial.bin` 代表在 `0x10000` 地址烧录 `ota_data_initial.bin` 文件。

1.3.2 烧录 AT 固件至设备

请根据您的操作系统选择对应的烧录方法。

Windows

开始烧录之前，请下载 Flash 下载工具。更多有关 Flash 下载工具的介绍，请参考压缩包中 `readme.pdf` 文件或 `doc` 文件夹。

- 打开 Flash 下载工具；
- 根据您的需求选择一种模式；（此处，我们选择 `Developer Mode`。）

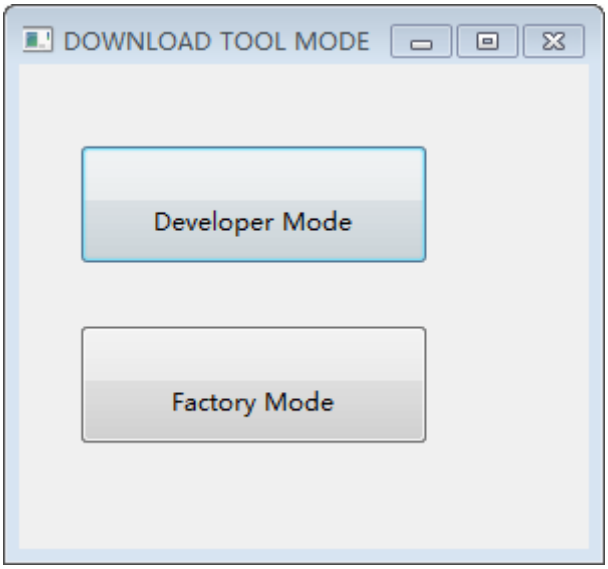


图 10: Flash 下载工具的模式

- 选择对应的 ESP 下载工具，如 ESP8266 设备应选择 “ESP8266 DownloadTool”，ESP32-S2 设备应选择 “ESP32-S2 DownloadTool”；（此处，我们选择 “ESP32 DownloadTool”。）
- 将 AT 固件烧录至设备，以下两种方式任选其一：
  - 直接下载打包好的量产固件至 `0x0` 地址：勾选 “DoNotChgBin”，使用量产固件的默认配置；
  - 分开下载多个 bin 文件至不同的地址：根据 `download.config` 文件进行配置，请勿勾选 “DoNotChgBin”；

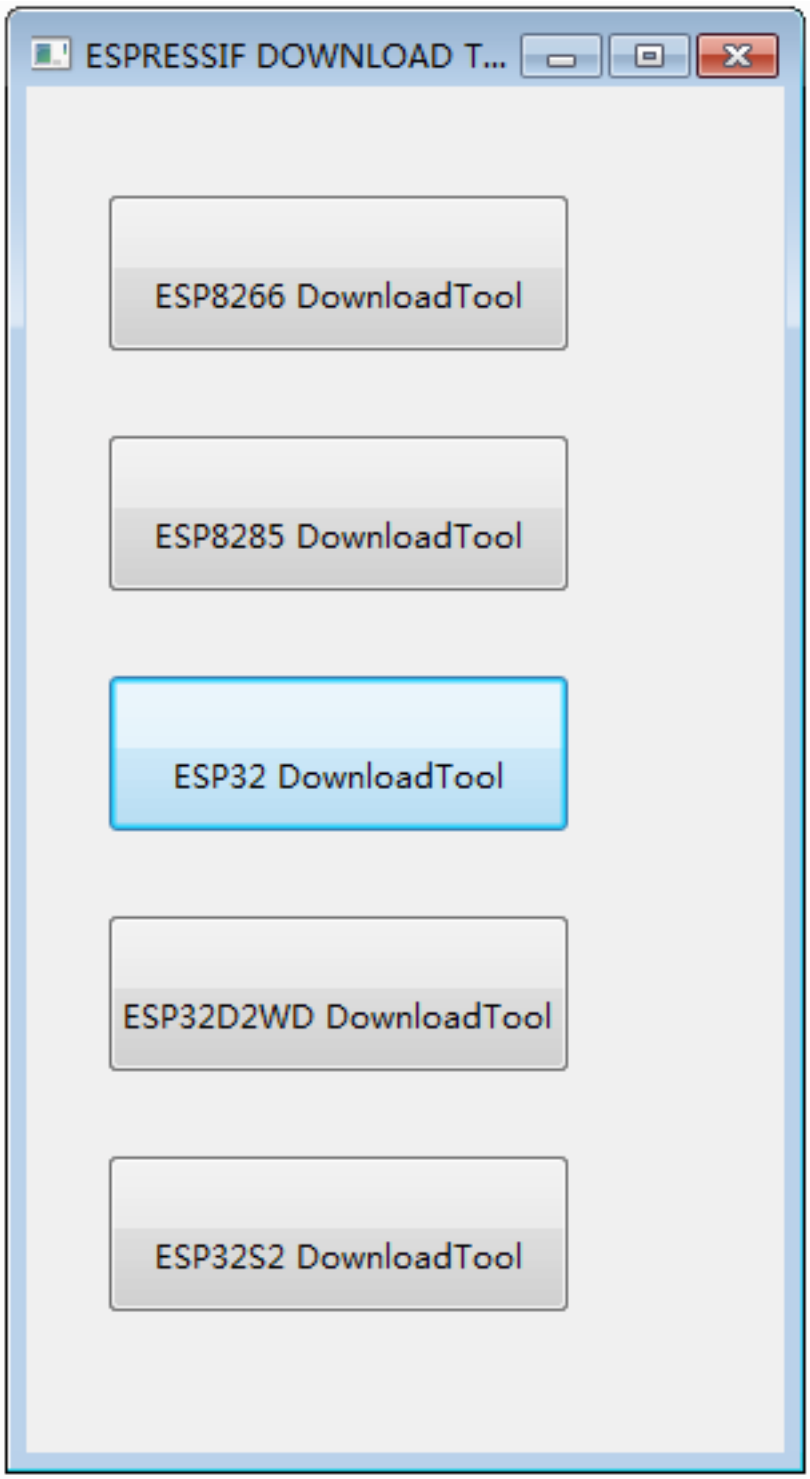


图 11: Flash 下载工具选择

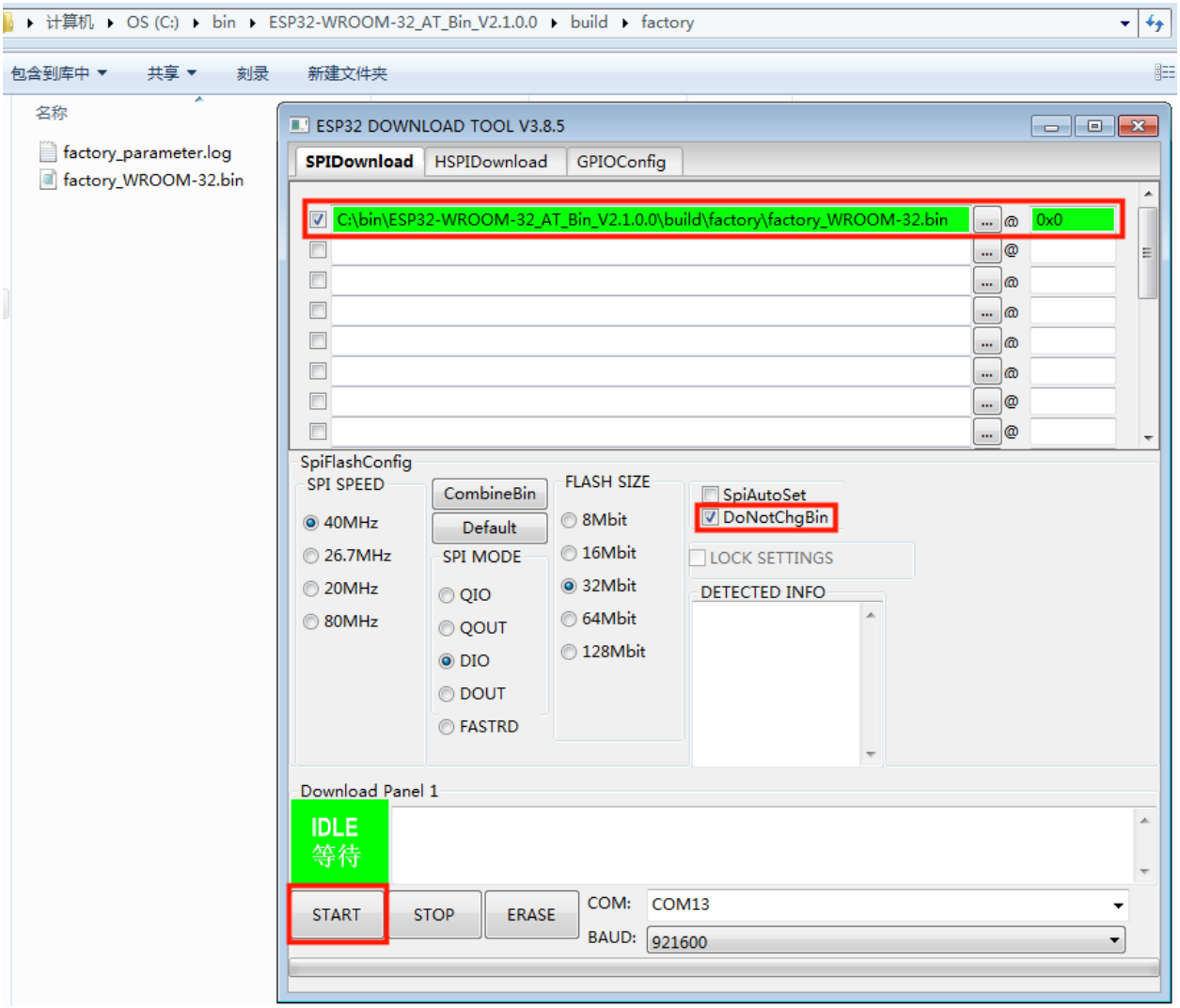


图 12: 下载至单个地址界面图

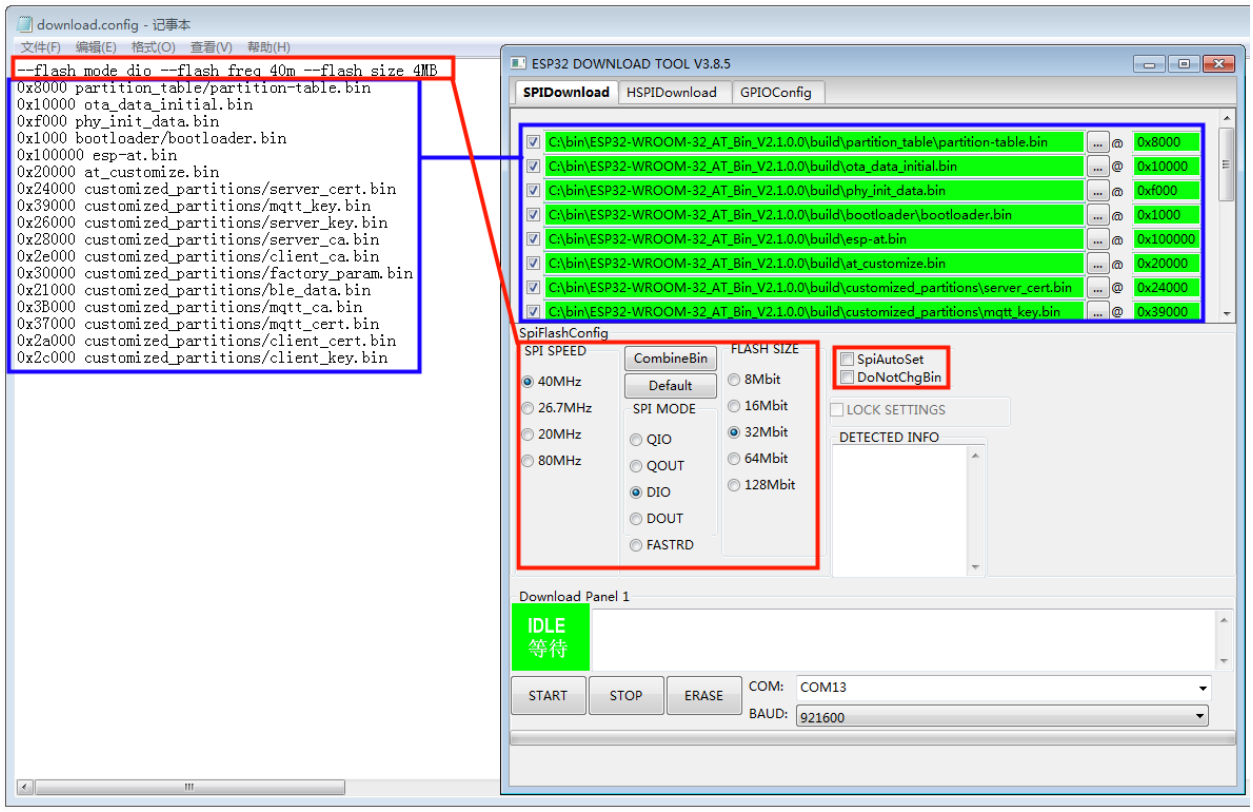


图 13: 下载至多个地址界面图

为了避免烧录出现问题，请查看开发板的下载接口的 COM 端口号，并从 “COM:” 下拉列表中选择该端口号。有关如何查看端口号的详细介绍请参考 在 Windows 上查看端口。

烧录完成后，请检查 *AT* 固件是否烧录成功。

Linux 或 macOS

开始烧录之前，请安装 esptool.py。

以下两种方式任选其一，将 AT 固件烧录至设备：

- 分开下载多个 bin 文件至不同的地址：输入以下命令，替换 `PORTNAME` 和 `download.config` 参数；

```
esptool.py --chip auto --port PORTNAME --baud 115200 --before default_reset --
↪after hard_reset write_flash -z download.config
```

  将 `PORTNAME` 替换成开发板的下载接口名称，若您无法确定该接口名称，请参考 在 Linux 和 macOS 上查看端口。

  将 `download.config` 替换成该文件里的参数列表。

  以下是将固件烧录至 ESP32-WROOM-32 模组输入的命令：

```
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 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
```

- 直接下载打包好的量产固件至 `0x0` 地址：输入以下命令，替换 `PORTNAME` 和 `FILEDIRECTORY` 参数；

```
esptool.py --chip auto --port PORTNAME --baud 115200 --before default_reset --
↪after hard_reset write_flash -z --flash_mode dio --flash_freq 40m --flash_size␣
↪4MB 0x0 FILEDIRECTORY
```

  将 `PORTNAME` 替换成开发板的下载接口名称，若您无法确定该接口名称，请参考 在 Linux 和 macOS 上查看端口。

  将 `FILEDIRECTORY` 替换成打包好的量产固件的文件路径，通常情况下，文件路径是 `factory/XXX.bin`。

  以下是将固件烧录至 ESP32-WROOM-32 模组输入的命令：

```
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
```

烧录完成后, 请检查 *AT* 固件是否烧录成功。

1.3.3 检查 AT 固件是否烧录成功

请按照以下步骤检查 AT 固件是否烧录成功：

- 打开串口工具，如 SecureCRT；
- 串口：选择用于发送或接收“AT 命令/响应”的串口（详情请见硬件连接）；
- 波特率：115200；
- 数据位：8；
- 检验位：None；
- 停止位：1；
- 流控：None；
- 输入“AT+GMR”命令，并且换行 (CR LF)；

若如下图所示，响应是 OK, 则表示 AT 固件烧录成功。

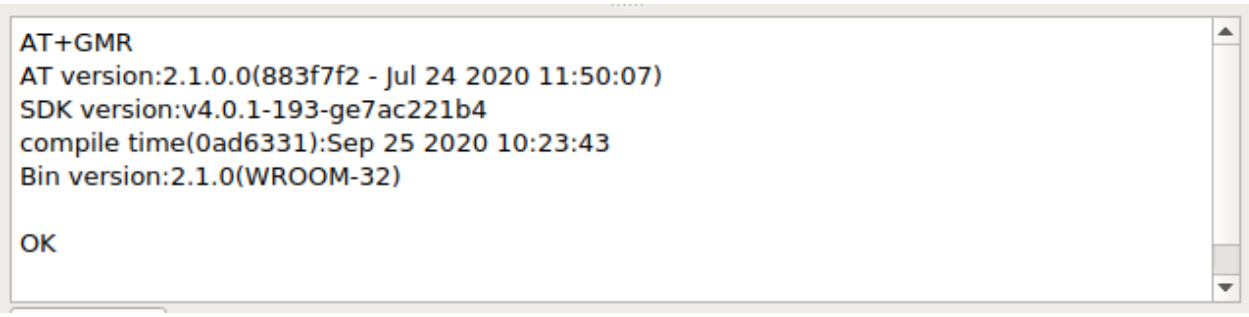


图 14: AT 响应

否则，您需要检查 ESP 设备开机日志，可以通过“下载/输出日志”串口在电脑上查看。若日志和下面的日志相似，则说明 ESP-AT 固件已经正确初始化了。

ESP32 开机日志：

```
ets Jun  8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
```

(下页继续)

(续上页)

```
load:0x3fff0030,len:4
load:0x3fff0034,len:7184
ho 0 tail 12 room 4
load:0x40078000,len:13200
load:0x40080400,len:4564
entry 0x400806f4
I (30) boot: ESP-IDF v4.2 2nd stage bootloader
I (31) boot: compile time 11:23:19
I (31) boot: chip revision: 0
I (33) boot.esp32: SPI Speed      : 40MHz
I (38) boot.esp32: SPI Mode       : DIO
I (42) boot.esp32: SPI Flash Size : 4MB
I (47) boot: Enabling RNG early entropy source...
I (52) boot: Partition Table:
I (56) boot: ## Label            Usage          Type ST Offset   Length
I (63) boot:  0 phy_init         RF data          01 01 0000f000 00001000
I (71) boot:  1 otadata          OTA data         01 00 00010000 00002000
I (78) boot:  2 nvs              WiFi data        01 02 00012000 0000e000
I (86) boot:  3 at_customize     unknown          40 00 00020000 000e0000
I (93) boot:  4 ota_0            OTA app          00 10 00100000 00180000
I (101) boot:  5 ota_1            OTA app          00 11 00280000 00180000
I (108) boot: End of partition table
I (112) esp_image: segment 0: paddr=0x00100020 vaddr=0x3f400020 size=0x2a300 (172800)␣
↪map
I (187) esp_image: segment 1: paddr=0x0012a328 vaddr=0x3ffbdb60 size=0x039e8 ( 14824)␣
↪load
I (194) esp_image: segment 2: paddr=0x0012dd18 vaddr=0x40080000 size=0x00404 (  1028)␣
↪load
I (194) esp_image: segment 3: paddr=0x0012e124 vaddr=0x40080404 size=0x01ef4 (  7924)␣
↪load
I (206) esp_image: segment 4: paddr=0x00130020 vaddr=0x400d0020 size=0x10a470␣
↪(1090672) map
I (627) esp_image: segment 5: paddr=0x0023a498 vaddr=0x400822f8 size=0x1c3a0 (115616)␣
↪load
I (678) esp_image: segment 6: paddr=0x00256840 vaddr=0x400c0000 size=0x00064 (   100)␣
↪load
I (695) boot: Loaded app from partition at offset 0x100000
I (695) boot: Disabling RNG early entropy source...
max tx power=78,ret=0
2.1.0
```

ESP32-S2 开机日志：

```
ESP-ROM:esp32s2-rc4-20191025
Build:Oct 25 2019
rst:0x1 (POWERON),boot:0x8 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3ffe6100,len:0x4
load:0x3ffe6104,len:0x1a24
load:0x4004c000,len:0x1a6c
load:0x40050000,len:0x20fc
entry 0x4004c35c
I (46) boot: ESP-IDF v4.2 2nd stage bootloader
I (46) boot: compile time 11:24:34
I (46) boot: chip revision: 0
I (47) qio_mode: Enabling default flash chip QIO
I (53) boot.esp32s2: SPI Speed      : 80MHz
I (57) boot.esp32s2: SPI Mode       : QIO
I (62) boot.esp32s2: SPI Flash Size : 4MB
I (67) boot: Enabling RNG early entropy source...
I (72) boot: Partition Table:
I (76) boot: ## Label            Usage          Type ST Offset   Length
I (83) boot:  0 phy_init         RF data          01 01 0000f000 00001000
I (91) boot:  1 otadata          OTA data         01 00 00010000 00002000
I (98) boot:  2 nvs              WiFi data        01 02 00012000 0000e000
I (106) boot:  3 at_customize     unknown          40 00 00020000 000e0000
I (113) boot:  4 ota_0            OTA app          00 10 00100000 00180000
I (121) boot:  5 ota_1            OTA app          00 11 00280000 00180000
I (128) boot: End of partition table
I (133) esp_image: segment 0: paddr=0x00100020 vaddr=0x3f000020 size=0x21bec (138220)␣
↪map
I (167) esp_image: segment 1: paddr=0x00121c14 vaddr=0x3ffc9330 size=0x02fe0 ( 12256)␣
↪load
I (169) esp_image: segment 2: paddr=0x00124bfc vaddr=0x40024000 size=0x00404 (  1028)␣
↪load
I (173) esp_image: segment 3: paddr=0x00125008 vaddr=0x40024404 size=0x0b010 ( 45072)␣
↪load
I (193) esp_image: segment 4: paddr=0x00130020 vaddr=0x40080020 size=0xb0784 (722820)␣
↪map
I (324) esp_image: segment 5: paddr=0x001e07ac vaddr=0x4002f414 size=0x09f18 ( 40728)␣
↪load
I (334) esp_image: segment 6: paddr=0x001ea6cc vaddr=0x40070000 size=0x0001c (    28)␣
↪load
I (346) boot: Loaded app from partition at offset 0x100000
I (346) boot: Disabling RNG early entropy source...
max tx power=78,ret=0
```

(下页继续)

(续上页)

```
2.1.0
```

ESP32-C3 开机日志:

```
ESP-ROM:esp32c3-20200918
Build:Sep 18 2020
rst:0x1 (POWERON),boot:0xc (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:2
load:0x3fcd6100,len:0x14
load:0x3fcd6114,len:0x179c
load:0x403ce000,len:0x894
load:0x403d0000,len:0x2bf8
entry 0x403ce000
I (54) boot: ESP-IDF v4.3-beta1 2nd stage bootloader
I (55) boot: compile time 12:09:42
I (55) boot: chip revision: 1
I (57) boot_comm: chip revision: 1, min. bootloader chip revision: 0
I (64) boot.esp32c3: SPI Speed      : 40MHz
I (68) boot.esp32c3: SPI Mode       : DIO
I (73) boot.esp32c3: SPI Flash Size : 4MB
I (78) boot: Enabling RNG early entropy source...
I (83) boot: Partition Table:
I (87) boot: ## Label            Usage          Type ST Offset   Length
I (94) boot:  0 phy_init         RF data          01 01 0000f000 00001000
I (102) boot:  1 otadata          OTA data         01 00 00010000 00002000
I (109) boot:  2 nvs              WiFi data        01 02 00012000 0000e000
I (117) boot:  3 at_customize     unknown          40 00 00020000 000e0000
I (124) boot:  4 ota_0            OTA app          00 10 00100000 00180000
I (132) boot:  5 ota_1            OTA app          00 11 00280000 00180000
I (139) boot: End of partition table
I (144) boot: No factory image, trying OTA 0
I (149) boot_comm: chip revision: 1, min. application chip revision: 0
I (156) esp_image: segment 0: paddr=00100020 vaddr=3c140020 size=29cc8h (171208) map
I (201) esp_image: segment 1: paddr=00129cf0 vaddr=3fc8f000 size=03be8h ( 15336) load
I (205) esp_image: segment 2: paddr=0012d8e0 vaddr=40380000 size=02738h ( 10040) load
I (210) esp_image: segment 3: paddr=00130020 vaddr=42000020 size=135bf0h (1268720) map
I (489) esp_image: segment 4: paddr=00265c18 vaddr=40382738 size=0c778h ( 51064) load
I (502) esp_image: segment 5: paddr=00272398 vaddr=50000000 size=00004h (     4) load
I (508) boot: Loaded app from partition at offset 0x100000
I (544) boot: Set actual ota_seq=1 in otadata[0]
I (544) boot: Disabling RNG early entropy source...
max tx power=78,ret=0
2.1.0
```

ESP8266 开机日志：

```
...
boot: ESP-IDF v3.4-rc 2nd stage bootloader
I (54) boot: compile time 11:18:21
I (54) boot: SPI Speed      : 80MHz
I (57) boot: SPI Mode       : DIO
I (61) boot: SPI Flash Size : 2MB
I (65) boot: Partition Table:
I (68) boot: ## Label            Usage          Type ST Offset   Length
I (75) boot:  0 otadata          OTA data         01 00 00009000 00002000
I (83) boot:  1 phy_init         RF data          01 01 0000f000 00001000
I (90) boot:  2 ota_0            OTA app          00 10 00010000 000e0000
I (98) boot:  3 at_customize     unknown          40 00 000f0000 00020000
I (105) boot:  4 ota_1            OTA app          00 11 00110000 000e0000
I (112) boot:  5 nvs              WiFi data        01 02 001f0000 00010000
I (120) boot: End of partition table
I (124) boot: No factory image, trying OTA 0
I (129) esp_image: segment 0: paddr=0x00010010 vaddr=0x40210010 size=0xac0d0 (704720)␣
↪map
I (138) esp_image: segment 1: paddr=0x000bc0e8 vaddr=0x402bc0e0 size=0x1aba8 (109480)␣
↪map
I (146) esp_image: segment 2: paddr=0x000d6c98 vaddr=0x3ffe8000 size=0x00788 (  1928)␣
↪load
I (155) esp_image: segment 3: paddr=0x000d7428 vaddr=0x40100000 size=0x00080 (   128)␣
↪load
I (164) esp_image: segment 4: paddr=0x000d74b0 vaddr=0x40100080 size=0x059c4 ( 22980)␣
↪load
I (173) boot: Loaded app from partition at offset 0x10000
phy_version: 1163.0, 665d56c, Jun 24 2020, 10:00:08, RTOS new
max tx power=78,ret=0
2.0.0
```

如果您尚不了解 ESP-AT 工程，请阅读*ESP-AT 是什么*。

如果您想学习如何使用 ESP-AT，请首先阅读硬件连接，了解所需的硬件以及硬件之间如何连接，然后再阅读下载指导，了解如何下载和烧录 AT 固件。

CHAPTER 2

AT 固件

Please refer to [English]

2.1 发布的固件

推荐下载最新版本的固件。

2.1.1 ESP32-WROOM-32 Series

- v2.1.0.0 ESP32-WROOM-32_AT_Bin_V2.1.0.0.zip（推荐）
- v2.0.0.0 ESP32-WROOM-32_AT_Bin_V2.0.0.0.zip

2.1.2 ESP32-WROVER-32 Series

- v2.1.0.0 ESP32-WROVER_AT_Bin_V2.1.0.0.zip（推荐）
- v2.0.0.0 ESP32-WROVER_AT_Bin_V2.0.0.0.zip

2.1.3 ESP32-PICO Series

- v2.1.0.0 ESP32-PICO-D4_AT_Bin_V2.1.0.0.zip（推荐）
- v2.0.0.0 ESP32-PICO-D4_AT_Bin_V2.0.0.0.zip

2.1.4 ESP32-SOLO Series

- v2.1.0.0 ESP32-SOLO_AT_Bin_V2.1.0.0.zip（推荐）
- v2.0.0.0 ESP32-SOLO_AT_Bin_V2.0.0.0.zip

2.2 发布的固件

推荐下载最新版本的固件。

2.2.1 ESP32-S2-WROOM Series

- v2.1.0.0 ESP32-S2-WROOM_AT_Bin_V2.1.0.0.zip（推荐）

2.2.2 ESP32-S2-WROVER Series

- v2.1.0.0 ESP32-S2-WROVER_AT_Bin_V2.1.0.0.zip（推荐）

2.2.3 ESP32-S2-SOLO Series

- v2.1.0.0 ESP32-S2-SOLO_AT_Bin_V2.1.0.0.zip（推荐）

2.2.4 ESP32-S2-MINI Series

- v2.1.0.0 ESP32-S2-MINI_AT_Bin_V2.1.0.0.zip（推荐）

2.3 发布的固件

推荐下载最新版本的固件。

说明：乐鑫没有为 **1 MB ESP8285/8266 系列芯片**发布单独的版本，但是您可以参考如何从 GitHub 下载最新临时版本 AT 固件并从 GitHub 上的 CI （持续集成）上下载 1 MB 的固件（**请切换到 release/v2.2.0.0_esp8266 分支后在 Artifacts 页面下载 `esp8285-1MB-at`**）。

2.3.1 ESP-WROOM-02 Series

- v2.2.1.0 ESP8266-IDF-AT_V2.2.1.0.zip（推荐）
- v2.2.0.0 ESP8266-IDF-AT_V2.2.0.0.zip
- v2.1.0.0 ESP8266-IDF-AT_V2.1.0.0.zip
- v2.0.0.0 ESP8266-IDF-AT_V2.0.0.0.zip

CHAPTER 3

AT 命令集

[English]

本章将具体介绍如何使用各类 AT 命令。如果命令前标注，如 [ESP32 Only]，则表示该（类）命令只适用于 ESP32 系列，若命令前无任何标注，则表示支持所有 ESP 系列，包括 ESP32、ESP8266、ESP32-S2、ESP32-C3。

3.1 基础 AT 命令

[English]

- *AT*：测试 AT 启动
- *AT+RST*：重启模块
- *AT+GMR*：查看版本信息
- *AT+CMD*：查询当前固件支持的所有命令及命令类型
- *AT+GSLP*：进入 Deep-sleep 模式
- *ATE*：开启或关闭 AT 回显功能
- *AT+RESTORE*：恢复出厂设置
- *AT+UART_CUR*：设置 UART 当前临时配置，不保存到 flash
- *AT+UART_DEF*：设置 UART 默认配置, 保存到 flash
- *AT+SLEEP*：设置 sleep 模式

- *AT+SYSRAM*：查询当前剩余堆空间和最小堆空间
- *AT+SYSMSG*：查询/设置系统提示信息
- *AT+USERRAM*：操作用户的空闲 RAM
- *AT+SYSFLASH*：查询或读写 flash 用户分区
- [ESP32 Only] *AT+FS*：文件系统操作
- *AT+RFPOWER*：查询/设置 RF TX Power
- *AT+SYSROLLBACK*：回滚到以前的固件
- *AT+SYSTIMESTAMP*：查询/设置本地时间戳
- *AT+SYSLOG*：启用或禁用 AT 错误代码提示
- *AT+SLEEPWKCFG*：设置 Light-sleep 唤醒源和唤醒 GPIO
- *AT+SYSSTORE*：设置参数存储模式
- *AT+SYSREG*：读写寄存器
- [ESP32-S2 Only] *AT+SYSTEMP*：查询 ESP32-S2 内部温度

3.1.1 AT：测试 AT 启动

执行命令

命令：

```
AT
```

响应：

```
OK
```

3.1.2 AT+RST：重启模块

执行命令

命令：

```
AT+RST
```

响应：

```
OK
```

3.1.3 AT+GMR：查看版本信息

执行命令

命令：

```
AT+GMR
```

响应：

```
<AT version info>
<SDK version info>
<compile time>
<Bin version>

OK
```

参数

- **<AT version info>**：AT 核心库的版本信息，它们在 esp-at/components/at/lib/ 目录下。代码是闭源的，无开放计划。
- **<SDK version info>**：AT 使用的平台 SDK 版本信息，它们定义在 esp-at/module_config/module_{platform}_default/IDF_VERSION 文件中。
- **<compile time>**：固件生成时间。
- **<Bin version>**: AT 固件版本信息。版本信息可以在 menuconfig 中修改。

说明

- 如果您在使用 ESP-AT 固件中有任何问题，请先提供 AT+GMR 版本信息。

示例

```
AT+GMR
AT version:2.2.0.0-dev(ca41ec4 - ESP32 - Sep 16 2020 11:28:17)
SDK version:v4.0.1-193-ge7ac221b4
compile time(98b95fc):Oct 29 2020 11:23:25
Bin version:2.1.0(MINI-1)
```

(下页继续)

(续上页)

```
OK
```

3.1.4 AT+CMD：查询当前固件支持的所有命令及命令类型

查询命令

命令：

```
AT+CMD?
```

响应：

```
+CMD:<index>,<AT command name>,<support test command>,<support query command>,
↪<support set command>,<support execute command>

OK
```

参数

- **<index>**：AT 命令序号
- **<AT command name>**：AT 命令名称
- **<support test command>**：0 表示不支持，1 表示支持
- **<support query command>**：0 表示不支持，1 表示支持
- **<support set command>**：0 表示不支持，1 表示支持
- **<support execute command>**：0 表示不支持，1 表示支持

3.1.5 AT+GSLP：进入 Deep-sleep 模式

设置命令

命令：

```
AT+GSLP=<time>
```

响应：

```
<time>

OK
```

参数

- **<time>**：设备进入 Deep-sleep 的时长，单位：毫秒。设定时间到后，设备自动唤醒，调用深度睡眠唤醒桩，然后加载应用程序。
  - ESP32 和 ESP32-S2 设备：最大 Deep-sleep 时长约为 28.8 天（2^{31}-1 毫秒）。
  - ESP8266 设备：最大 Deep-sleep 时长约为 3 小时，由于硬件条件限制，更长的睡眠时间会造成设置失败或内部时间溢出。

说明

- ESP8266 设备必须将 GPIO16 连接到 RST 管脚才能在睡眠时长结束后自行唤醒。
- 由于外部因素的影响，所有设备进入 Deep-sleep 的实际时长与理论时长之间会存在差异。
- 也可通过直接触发 ESP8266 的 RST 管脚的低电平脉冲将其从 Deep-sleep 中唤醒。

3.1.6 ATE：开启或关闭 AT 回显功能

执行命令

命令：

```
ATE0
```

或

```
ATE1
```

响应：

```
OK
```

参数

- **ATE0**：关闭回显
- **ATE1**：开启回显

3.1.7 AT+RESTORE：恢复出厂设置

执行命令

命令：

```
AT+RESTORE
```

响应：

```
OK
```

说明

- 该命令将擦除所有保存到 flash 的参数，并恢复为默认参数。
- 运行该命令会重启设备。

3.1.8 AT+UART_CUR：设置 UART 当前临时配置，不保存到 flash

查询命令

命令：

```
AT+UART_CUR?
```

响应：

```
+UART_CUR:<baudrate>,<databits>,<stopbits>,<parity>,<flow control>

OK
```

设置命令

命令：

```
AT+UART_CUR=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>
```

响应：

```
OK
```

参数

- **<baudrate>**：UART 波特率
  - ESP32 和 ESP32-S2 设备：支持范围为 80 ~ 5000000
  - ESP8266 设备：支持范围为 80 ~ 4500000
- **<databits>**：数据位
  - 5：5 bit 数据位
  - 6：6 bit 数据位
  - 7：7 bit 数据位
  - 8：8 bit 数据位
- **<stopbits>**：停止位
  - 1：1 bit 停止位
  - 2：1.5 bit 停止位
  - 3：2 bit 停止位
- **<parity>**：校验位
  - 0：None
  - 1：Odd
  - 2：Even
- **<flow control>**：流控
  - 0：不使能流控
  - 1：使能 RTS
  - 2：使能 CTS
  - 3：同时使能 RTS 和 CTS

说明

- 查询命令返回的是 UART 配置参数的实际值，由于时钟分频的原因，可能与设定值有细微的差异。
- 本设置不保存到 flash。
- 使用硬件流控功能需要连接设备的 CTS/RTS 管脚，详情请见硬件连接 和 components/customized_partitions/raw_data/factory_param/factory_param_data.csv。

示例

```
AT+UART_CUR=115200,8,1,0,3
```

3.1.9 AT+UART_DEF：设置 UART 默认配置，保存到 flash

查询命令

命令：

```
AT+UART_DEF?
```

响应：

```
+UART_DEF:<baudrate>,<databits>,<stopbits>,<parity>,<flow control>

OK
```

设置命令

命令：

```
AT+UART_DEF=<baudrate>,<databits>,<stopbits>,<parity>,<flow control>
```

响应：

```
OK
```

参数

- **<baudrate>**：UART 波特率
  - ESP32 和 ESP32-S2 设备：支持范围为 80 ~ 5000000
  - ESP8266 设备：支持范围为 80 ~ 4500000
- **<databits>**：数据位
  - 5：5 bit 数据位
  - 6：6 bit 数据位
  - 7：7 bit 数据位
  - 8：8 bit 数据位
- **<stopbits>**：停止位
  - 1：1 bit 停止位
  - 2：1.5 bit 停止位
  - 3：2 bit 停止位
- **<parity>**：校验位
  - 0：None
  - 1：Odd
  - 2：Even
- **<flow control>**：流控
  - 0：不使能流控
  - 1：使能 RTS
  - 2：使能 CTS
  - 3：同时使能 RTS 和 CTS

说明

- 配置更改将保存在 NVS 分区，当设备再次上电时仍然有效。
- 使用硬件流控功能需要连接设备的 CTS/RTS 管脚，详情请见硬件连接 和 `components/customized_partitions/raw_data/factory_param/factory_param_data.csv`。

示例

```
AT+UART_DEF=115200,8,1,0,3
```

3.1.10 AT+SLEEP：设置睡眠模式

查询命令

命令：

```
AT+SLEEP?
```

响应：

```
+SLEEP:<sleep mode>

OK
```

设置命令

命令：

```
AT+SLEEP=<sleep mode>
```

响应：

```
OK
```

参数

- **<sleep mode>**：
  - 0：禁用睡眠模式
  - 1：Modem-sleep DTIM 模式，射频模块将根据 AP 的 DTIM 定期关闭
  - 2：Light-sleep 模式，CPU 将自动进入睡眠，射频模块也将根据*AT+CWJAP* 命令设置的 `listen interval` 参数定期关闭
  - 3：Modem-sleep listen interval 模式，射频模块将根据*AT+CWJAP* 命令设置的 `listen interval` 参数定期关闭

说明

- 只有在 station 模式，才可以设置 Modem-sleep 和 Light-sleep 睡眠模式
- 设置 Light-sleep 模式前，建议提前通过*AT+SLEEPWKCFG* 命令设置好唤醒源，否则没法唤醒，设备将一直处于睡眠状态
- 设置 Light-sleep 模式后，如果 Light-sleep 唤醒条件不满足时，设备将自动进入睡眠模式，当 Light-sleep 唤醒条件满足时，设备将自动从睡眠模式中唤醒
- 查询命令可以返回 0、1、3，但不能返回 2（由于 ESP-IDF 不支持此查询）。换句话说，如果设置 `AT+SLEEP=2`，那么 `AT+SLEEP?` 会返回 3，而不是 2

示例

```
AT+SLEEP=0
```

3.1.11 AT+SYSRAM：查询当前剩余堆空间和最小堆空间

查询命令

命令：

```
AT+SYSRAM?
```

响应：

```
+SYSRAM:<remaining RAM size>,<minimum heap size>
OK
```

参数

- **<remaining RAM size>**：当前剩余堆空间，单位：byte
- **<minimum heap size>**：最小堆空间，单位：byte

示例

```
AT+SYSRAM?
+SYSRAM:148408,84044
OK
```

3.1.12 AT+SYSMSG：查询/设置系统提示信息

查询命令

功能：

查询当前系统提示信息状态

命令：

```
AT+SYSMSG?
```

响应：

```
+SYSMSG:<state>
OK
```

设置命令

功能：

设置系统提示信息

命令：

```
AT+SYSMSG=<state>
```

响应：

```
OK
```

参数

- **<state>**：
  - Bit0：退出 Wi-Fi 透传模式 时是否打印提示信息
    - 0：不打印
    - 1：打印 +QUITT
  - Bit1：连接时提示信息类型
    - 0：使用简单版提示信息，如 XX,CONNECT
    - 1：使用详细版提示信息，如 +LINK_CONN:status_type,link_id,ip_type,terminal_type,remote_ip,remote_port,local_port
  - Bit2：连接状态提示信息，适用于 Wi-Fi 透传模式、Bluetooth LE SPP 及 Bluetooth SPP
    - 0：不打印提示信息
    - 1：当 Wi-Fi、socket、Bluetooth LE 或 Bluetooth 状态发生改变时，打印提示信息，如：

```
- "CONNECT\r\n" 或以 "+LINK_CONN:" 开头的提示信息
- "CLOSED\r\n"
- "WIFI CONNECTED\r\n"
- "WIFI GOT IP\r\n"
- "WIFI GOT IPv6 LL\r\n"
- "WIFI GOT IPv6 GL\r\n"
- "WIFI DISCONNECT\r\n"
- "+ETH_CONNECTED\r\n"
- "+ETH_DISCONNECTED\r\n"
- 以 "+ETH_GOT_IP:" 开头的提示信息
- 以 "+STA_CONNECTED:" 开头的提示信息
- 以 "+STA_DISCONNECTED:" 开头的提示信息
- 以 "+DIST_STA_IP:" 开头的提示信息
- 以 "+BLECONN:" 开头的提示信息
- 以 "+BLEDISCONN:" 开头的提示信息
```

说明

- 若*AT+SYSSTORE=1*，配置更改将被保存在 NVS 分区。
- 若设 Bit0 为 1，退出 Wi-Fi 透传模式 时会提示 +QUITT。
- 若设 Bit1 为 1，将会影响*AT+CIPSTART* 和*AT+CIPSERVER* 命令，系统将提示“+LINK_CONN:status_type,link_id,ip_type,terminal_type,remote_ip,remote_port,local_port”，而不是“XX,CONNECT”。

示例

```
// 退出 Wi-Fi 透传模式时不打印提示信息
// 连接时打印详细版提示信息
// 连接状态发生改变时不打印信息
AT+SYSMSG=2
```

3.1.13 AT+USERRAM: 操作用户的空闲 RAM

查询命令

功能：

查询用户当前可用的空闲 RAM 大小

命令：

```
AT+USERRAM?
```

响应：

```
+USERRAM:<size>

OK
```

设置命令

功能：

分配、读、写、擦除、释放用户 RAM 空间

命令：

```
AT+USERRAM=<operation>,<size>[,<offset>]
```

响应：

```
+USERRAM:<length>,<data>     // 只有是读操作时，才会有这个回复

OK
```

参数

- **<operation>**：
  - 0：释放用户 RAM 空间
  - 1：分配用户 RAM 空间
  - 2：向用户 RAM 写数据
  - 3：从用户 RAM 读数据
  - 4：清除用户 RAM 上的数据
- **<size>**: 分配/读/写的用户 RAM 大小
- **<offset>**: 读/写 RAM 的偏移量。默认：0

说明

- 请在执行任何其他操作之前分配用户 RAM 空间。
- 当 <operator> 为 write 时，系统收到此命令后先换行返回 >，此时您可以输入要写的数据，数据长度应与 <length> 一致。
- 当 <operator> 为 read 时并且长度大于 1024，ESP-AT 会以同样格式多次回复，每次回复最多携带 1024 字节数据，最终以 \r\nOK\r\n 结束。

示例

```
// 分配 1 KB 用户 RAM 空间
AT+USERRAM=1,1024

// 向 RAM 空间开始位置写入 500 字节数据
AT+USERRAM=2,500

// 从 RAM 空间偏移 100 位置读取 64 字节数据
AT+USERRAM=3,64,100

// 释放用户 RAM 空间
AT+USERRAM=0
```

3.1.14 AT+SYSFLASH：查询或读写 flash 用户分区

查询命令

功能：

查询 flash 用户分区

命令：

```
AT+SYSFLASH?
```

响应：

```
+SYSFLASH:<partition>,<type>,<subtype>,<addr>,<size>
OK
```

设置命令

功能：

读、写、擦除 flash 用户分区

命令：

```
AT+SYSFLASH=<operation>,<partition>,<offset>,<length>
```

响应：

```
+SYSFLASH:<length>,<data>
OK
```

参数

- **<operation>**：
  - 0：擦除分区
  - 1：写分区
  - 2：读分区
- **<partition>**：用户分区名称
- **<offset>**：偏移地址
- **<length>**：数据长度
- **<type>**：用户分区类型

- **<subtype>**：用户分区子类型
- **<addr>**：用户分区地址
- **<size>**：用户分区大小

说明

- 使用本命令需烧录 at_customize.bin，详细信息可参考如何自定义分区。
- 擦除分区时，设置指令可省略 <offset> 和 <length> 参数，用于完整擦除该目标分区。例如，指令 AT+SYSFLASH=0,"ble_data" 可擦除整个“ble_data”区域。如果擦除分区时不省略 <offset> 和 <length> 参数，则这两个参数值要求是 4 KB 的整数倍。
- 关于分区的定义可参考 ESP-IDF 分区表。
- 当 <operator> 为 write 时，系统收到此命令后先换行返回 >，此时您可以输入要写的数据，数据长度应与 <length> 一致。
- 写分区前，请先擦除该分区。
- 写 PKI bin 时，参数 <length> 应为 4 字节的整数倍。

示例

```
// 从 "ble_data" 分区偏移地址 0 处读取 100 字节
AT+SYSFLASH=2,"ble_data",0,100

// 在 "ble_data" 分区偏移地址 100 处写入 10 字节
AT+SYSFLASH=1,"ble_data",100,10

// 从 "ble_data" 分区偏移地址 4096 处擦除 8192 字节
AT+SYSFLASH=0,"ble_data",4096,8192
```

3.1.15 [ESP32 Only] AT+FS：文件系统操作

设置命令

命令：

```
AT+FS=<type>,<operation>,<filename>,<offset>,<length>
```

响应：

```
OK
```

参数

- **<type>**：目前仅支持 FATFS
  - 0：FATFS
- **<operation>**:
  - 0：删除文件
  - 1：写文件
  - 2：读文件
  - 3：查询文件大小
  - 4：查询路径下文件，目前仅支持根目录
- **<offset>**：偏移地址，仅针对读写操作设置
- **<length>**：长度，仅针对读写操作设置

说明

- 使用本命令需烧录 at_customize.bin，详细信息可参考 ESP-IDF 分区表 和如何自定义分区。
- 若读取数据的长度大于实际文件大小，仅返回实际长度的数据。
- 当 <operator> 为 write 时，系统收到此命令后先换行返回 >，此时您可以输入要写的数据，数据长度应与 <length> 一致。

示例

```
// 删除某个文件
AT+FS=0,0,"filename"

// 在某个文件偏移地址 100 处写入 10 字节
AT+FS=0,1,"filename",100,10

// 从某个文件偏移地址 0 处读取 100 字节
AT+FS=0,2,"filename",0,100

// 列出根目录下所有文件
AT+FS=0,4,"."
```

3.1.16 AT+RFPOWER：查询/设置 RF TX Power

查询命令

功能：

查询 RF TX Power

命令：

```
AT+RFPOWER?
```

响应：

```
+RFPOWER:<wifi_power>,<ble_adv_power>,<ble_scan_power>,<ble_conn_power>
OK
```

设置命令

命令：

```
AT+RFPOWER=<wifi_power>[,<ble_adv_power>,<ble_scan_power>,<ble_conn_power>]
```

响应：

```
OK
```

参数

- **<wifi_power>**：单位为 0.25 dBm，比如设定的参数值为 78，则实际的 RF Power 值为 78 * 0.25 dBm = 19.5 dBm。配置后可运行 `AT+RFPOWER?` 命令确认实际的 RF Power 值。
  - ESP32 和 ESP32-S2 设备的取值范围为 [40,78]：

设定值	实际值	实际 dBm
[34,43]	34	8.5
[44,51]	44	11
[52,55]	52	13
[56,59]	56	14
[60,65]	60	15
[66,71]	66	16.5
[72,77]	72	18
78	78	19.5

  - ESP32-C3 设备的取值范围为 [40,84]：

设定值	实际值	实际 dBm
[40,80]	< 设定值 >	< 设定值 > * 0.25
[81,84]	80	20

  - ESP8266 设备的取值范围为 [40,82]：

设定值	实际值	实际 dBm
[40,47]	32	8
[48,55]	48	12
[56,63]	56	14
[64,67]	64	16
[68,73]	68	17
[74,77]	74	18.5
[78,81]	78	19.5
82	82	20.5

- **<ble_adv_power>**：Bluetooth LE 广播的 RF TX Power，取值范围为 [0,7]。
  - 0：7 dBm
  - 1：4 dBm
  - 2：1 dBm
  - 3：-2 dBm
  - 4：-5 dBm
  - 5：-8 dBm
  - 6：-11 dBm
  - 7：-14 dBm
- **<ble_scan_power>**：Bluetooth LE 扫描的 RF TX Power，取值范围及对应值同 `<ble_adv_power>` 参数。
- **<ble_conn_power>**：Bluetooth LE 连接的 RF TX Power，取值范围及对应值同 `<ble_adv_power>` 参数。

3.1.17 说明

- 由于 RF TX Power 分为不同的等级，而每个等级都有与之对应的取值范围，所以通过 `esp_wifi_get_max_tx_power` 查询到的 `wifi_power` 的值可能与 `esp_wifi_set_max_tx_power` 设定的值存在差异，但不会比该值大。

3.1.18 AT+SYSROLLBACK：回滚到以前的固件

执行命令

命令：

```
AT+SYSROLLBACK
```

响应：

```
OK
```

说明

- 本命令不通过 OTA 升级，只会回滚到另一 OTA 分区的固件。

3.1.19 AT+SYSTIMESTAMP：查询/设置本地时间戳

查询命令

功能：

查询本地时间戳

命令：

```
AT+SYSTIMESTAMP?
```

响应：

```
+SYSTIMESTAMP:<Unix_timestamp>
OK
```

设置命令

功能：

设置本地时间戳，当 SNTP 时间更新后，将与之同步更新

命令：

```
AT+SYSTIMESTAMP=<Unix_timestamp>
```

响应：

```
OK
```

参数

- **<Unix-timestamp>**：Unix 时间戳，单位：秒。

示例

```
AT+SYSTIMESTAMP=1565853509     //2019-08-15 15:18:29
```

3.1.20 AT+SYSLOG：启用或禁用 AT 错误代码提示

查询命令

功能：

查询 AT 错误代码提示是否启用

命令：

```
AT+SYSLOG?
```

响应：

```
+SYSLOG:<status>

OK
```

设置命令

功能：

启用或禁用 AT 错误代码提示

命令：

```
AT+SYSLOG=<status>
```

响应：

```
OK
```

参数

- **<status>**：错误代码提示状态
  - 0：禁用
  - 1：启用

示例

```
// 启用 AT 错误代码提示
AT+SYSLOG=1

OK
AT+FAKE
ERR CODE:0x01090000

ERROR
```

```
// 禁用 AT 错误代码提示
AT+SYSLOG=0

OK
AT+FAKE
// 不提示 `ERR CODE:0x01090000`

ERROR
```

AT 错误代码是一个 32 位十六进制数值，定义如下：

类型	子类型	扩展
bit32 ~ bit24	bit23 ~ bit16	bit15 ~ bit0

- **category**：固定值 0x01
- **subcategory**：错误类型

错误类型	错误代码	说明
ESP_AT_SUB_OK	0x00	OK
ESP_AT_SUB_COMMON_ERROR	0x01	保留
ESP_AT_SUB_NO_TERMINATOR	0x02	未找到结束符（应以“rn”结尾）
ESP_AT_SUB_NO_AT	0x03	未找到起始 AT（输入的可能是 at、At 或 aT）
ESP_AT_SUB_PARA_LENGTH_MISMATCH	0x04	参数长度不匹配
ESP_AT_SUB_PARA_TYPE_MISMATCH	0x05	参数类型不匹配
ESP_AT_SUB_PARA_NUM_MISMATCH	0x06	参数数量不匹配
ESP_AT_SUB_PARA_INVALID	0x07	无效参数
ESP_AT_SUB_PARA_PARSE_FAIL	0x08	解析参数失败
ESP_AT_SUB_UNSUPPORT_CMD	0x09	不支持该命令
ESP_AT_SUB_CMD_EXEC_FAIL	0x0A	执行命令失败
ESP_AT_SUB_CMD_PROCESSING	0x0B	仍在执行上一条命令
ESP_AT_SUB_CMD_OP_ERROR	0x0C	命令操作类型错误

- **extension**：错误扩展信息，不同的子类型有不同的扩展信息，详情请见 `components/at/include/esp_at.h`。

例如，错误代码 `ERR CODE:0x01090000` 表示“不支持该命令”。

3.1.21 AT+SLEEPWKCFG：设置 Light-sleep 唤醒源和唤醒 GPIO

设置命令

命令：

```
AT+SLEEPWKCFG=<wakeup source>,<param1>[,<param2>]
```

响应：

```
OK
```

参数

- **<wakeup source>**: 唤醒源
  - 0：保留配置，暂不支持
  - 1：保留配置，暂不支持
  - 2：GPIO 唤醒
- **<param1>**:
  - 当唤醒源为定时器时，该参数表示睡眠时间，单位：毫秒
  - 当唤醒源为 GPIO 时，该参数表示 GPIO 管脚
- **<param2>**:
  - 当唤醒源为 GPIO 时，该参数表示唤醒电平
  - 0：低电平
  - 1：高电平

说明

- 在 ESP8266 平台上，作为 RTC IO 的 GPIO16 不能设为唤醒 ESP8266 设备的 GPIO 管脚。

示例

```
// GPIO12 置为低电平时唤醒
AT+SLEEPWKCFG=2,12,0
```

3.1.22 AT+SYSSTORE：设置参数存储模式

查询命令

功能：

查询 AT 参数存储模式

命令：

```
AT+SYSSTORE?
```

响应：

```
+SYSSTORE:<store_mode>

OK
```

设置命令

命令：

```
AT+SYSSTORE=<store_mode>
```

响应：

```
OK
```

参数

- **<store_mode>**：参数存储模式
  - 0：命令配置不存入 flash
  - 1：命令配置存入 flash（默认）

说明

- 该命令只影响设置命令，不影响查询命令，因为查询命令总是从 RAM 中调用。
- 本命令会影响以下命令：
  - *AT+SYSMSG*
  - *AT+CWMODE*
  - *AT+CIPV6*
  - *AT+CWJAP*
  - *AT+CWSAP*
  - *AT+CWRECONNCFG*
  - *AT+CIPAP*
  - *AT+CIPSTA*
  - *AT+CIPAPMAC*
  - *AT+CIPSTAMAC*
  - *AT+CIPDNS*

- *AT+CIPSSLCCONF*
- *AT+CIPRECONNINTV*
- *AT+CIPTCPOPT*
- *AT+CWDHCPS*
- *AT+CWDHCP*
- *AT+CWSTAPROTO*
- *AT+CWAPPROTO*
- *AT+CWJEAP*
- *AT+CIPETH*
- *AT+CIPETHMAC*
- *AT+BLENAME*
- *AT+BTNAME*
- *AT+BLEADVPARAM*
- *AT+BLEADVDATA*
- *AT+BLEADVDATAEX*
- *AT+BLESCANRSPDATA*
- *AT+BLESCANPARAM*
- *AT+BTSCANMODE*
- *AT+BLECONNPARAM*

示例

```
AT+SYSSTORE=0
AT+CWMODE=1  // 不存入 flash
AT+CWJAP="test","1234567890" // 不存入 flash

AT+SYSSTORE=1
AT+CWMODE=3  // 存入 flash
AT+CWJAP="test","1234567890" // 存入 flash
```

3.1.23 AT+SYSREG：读写寄存器

设置命令

命令：

```
AT+SYSREG=<direct>,<address>[,<write value>]
```

响应：

```
+SYSREG:<read value>  // 仅适用于读寄存器时
OK
```

参数

- **<direct>**：读或写寄存器
  - 0：读寄存器
  - 1：写寄存器
- **<address>**：(uint32) 寄存器地址，详情请参考相关的《技术参考手册》
- **<write value>**：(uint32) 写入值，仅适用于写寄存器时

说明

- AT 不检查寄存器地址，因此请确保操作的寄存器地址有效

示例

```
// ESP32-S2 IO33 输出，0x3F40402C 由基础地址 0x3F404000 与相对地址 0x2C (GPIO_ENABLE1_REG)␣
↪相加而来
AT+SYSREG=1,0x3F40402C,0x2

// ESP32-S2 IO33 输出高电平
AT+SYSREG=1,0x3F404010,0x2

// ESP32-S2 IO33 输出低电平
AT+SYSREG=1,0x3F404010,0x0
```

3.1.24 [ESP32-S2 Only] AT+SYSTEMP：查询 ESP32-S2 内部温度

查询命令

命令：

```
AT+SYSTEMP?
```

响应：

```
+SYSTEMP:<temperature>
OK
```

参数

- **<temperature>**：测量输出值，单位：摄氏度

说明

- 测量范围：-10℃ ~ 80℃，误差小于 1℃

示例

```
AT+SYSTEMP?
+SYSTEMP:21.59
OK
```

3.2 Wi-Fi AT 命令集

[English]

- *AT+CWMODE*：查询/设置 Wi-Fi 模式 (Station/SoftAP/Station+SoftAP)
- *AT+CWJAP*：连接 AP
- *AT+CWRECONNCFG*：查询/设置 Wi-Fi 重连配置
- *AT+CWLAPOPT*：设置*AT+CWLAP* 命令扫描结果的属性
- *AT+CWLAP*：扫描当前可用的 AP
- *AT+CWQAP*：断开与 AP 的连接
- *AT+CWSAP*：配置 ESP32 SoftAP 参数

- *AT+CWLIF*：查询连接到 ESP SoftAP 的 station 信息
- *AT+CWQIF*：断开 station 与 ESP SoftAP 的连接
- *AT+CWDHCP*：启用/禁用 DHCP
- *AT+CWDHCPS*：查询/设置 ESP SoftAP DHCP 分配的 IP 地址范围
- *AT+CWAUTOCONN*：上电是否自动连接 AP
- *AT+CWAPPROTO*：查询/设置 SoftAP 模式下 802.11 b/g/n 协议标准
- *AT+CWSTAPROTO*：设置 Station 模式下 802.11 b/g/n 协议标准
- *AT+CIPSTAMAC*：查询/设置 ESP Station 的 MAC 地址
- *AT+CIPAPMAC*：查询/设置 ESP SoftAP 的 MAC 地址
- *AT+CIPSTA*：查询/设置 ESP Station 的 IP 地址
- *AT+CIPAP*：查询/设置 ESP SoftAP 的 IP 地址
- *AT+CWSTARTSMART*：开启 SmartConfig
- *AT+CWSTOPSMART*：停止 SmartConfig
- *AT+WPS*：设置 WPS 功能
- *AT+MDNS*：设置 mDNS 功能
- [ESP32 Only] *AT+CWJEAP*：连接 WPA2 企业版 AP
- *AT+CWHOSTNAME*：查询/设置 ESP Station 的主机名称
- *AT+CWCOUNTRY*：查询/设置 Wi-Fi 国家代码

3.2.1 AT+CWMODE：查询/设置 Wi-Fi 模式 (Station/SoftAP/Station+SoftAP)

查询命令

功能：

查询 ESP 设备的 Wi-Fi 模式

命令：

```
AT+CWMODE?
```

响应：

```
+CWMODE:<mode>
OK
```

设置命令

功能：

设置 ESP 设备的 Wi-Fi 模式

命令：

```
AT+CWMODE=<mode>[,<auto_connect>]
```

响应：

```
OK
```

参数

- **<mode>**：模式
  - 0: 无 Wi-Fi 模式，并且关闭 Wi-Fi RF
  - 1: Station 模式
  - 2: SoftAP 模式
  - 3: SoftAP+Station 模式
- **<auto_connect>**：切换 ESP 设备的 Wi-Fi 模式时（例如，从 SoftAP 或无 Wi-Fi 模式切换为 Station 模式或 SoftAP+Station 模式），是否启用自动连接 AP 的功能，默认值：1。参数缺省时，使用默认值，也就是能自动连接。
  - 0: 禁用自动连接 AP 的功能
  - 1: 启用自动连接 AP 的功能，若之前已经将自动连接 AP 的配置保存到 flash 中，则 ESP 设备将自动连接 AP

说明

- 若*AT+SYSSTORE=1*，本设置将保存在 NVS 分区

示例

```
AT+CWMODE=3
```

3.2.2 AT+CWJAP：连接 AP

查询命令

功能：

查询与 ESP Station 连接的 AP 信息

命令：

```
AT+CWJAP?
```

响应：

```
+CWJAP:<ssid>,<bssid>,<channel>,<rssi>,<pci_en>,<reconn_interval>,<listen_interval>,
↪<scan_mode>,<pmf>
OK
```

设置命令

功能：

设置 ESP Station 需连接的 AP

命令：

```
AT+CWJAP=[<ssid>],[<pwd>][,<bssid>][,<pci_en>][,<reconn_interval>][,<listen_interval>
↪][,<scan_mode>][,<jap_timeout>][,<pmf>]
```

响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
[WIFI GOT IPv6 LL]
[WIFI GOT IPv6 GL]
```

或

```
+CWJAP:<error code>
ERROR
```

执行命令

功能：

将 ESP station 连接至上次 Wi-Fi 配置中的 AP

命令：

```
AT+CWJAP
```

响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
[WIFI GOT IPv6 LL]
[WIFI GOT IPv6 GL]
```

或

```
+CWJAP:<error code>
ERROR
```

参数

- **<ssid>**：目标 AP 的 SSID
  - 如果 SSID 和密码中有 `,`、`"`、`\\` 等特殊字符，需转义
- **<pwd>**：密码最长 64 字节 ASCII
- **<bssid>**：目标 AP 的 MAC 地址，当多个 AP 有相同的 SSID 时，该参数不可省略
- **<channel>**：信道号
- **<rssi>**：信号强度
- **<pci_en>**：PCI 认证
  - 0: ESP station 可与任何一种加密方式的 AP 连接，包括 OPEN 和 WEP
  - 1: ESP station 可与除 OPEN 和 WEP 之外的任何一种加密方式的 AP 连接
- **<reconn_interval>**：Wi-Fi 重连间隔，单位：秒，默认值：1，最大值：7200
  - 0: 断开连接后，ESP station 不重连 AP
  - [1,7200]: 断开连接后，ESP station 每隔指定的时间与 AP 重连
- **<listen_interval>**：监听 AP beacon 的间隔，单位为 AP beacon 间隔，默认值：3，范围：[1,100]

- **<scan_mode>**：扫描模式
  - 0: 快速扫描，找到目标 AP 后终止扫描，ESP station 与第一个扫描到的 AP 连接
  - 1: 全信道扫描，所有信道都扫描后才终止扫描，ESP station 与扫描到的信号最强的 AP 连接
- **<jap_timeout>**：*AT+CWJAP* 命令超时的最大值，单位：秒，默认值：15，范围：[3,600]
- **<pmf>**：PMF（Protected Management Frames，受保护的管理帧），默认值 0
  - 0 表示禁用 PMF
  - bit 0: 具有 PMF 功能，提示支持 PMF，如果其他设备具有 PMF 功能，则 ESP 设备将优先选择以 PMF 模式连接
  - bit 1: 需要 PMF，提示需要 PMF，设备将不会关联不支持 PMF 功能的设备
- **<error code>**：错误码，仅供参考
  - 1: 连接超时
  - 2: 密码错误
  - 3: 无法找到目标 AP
  - 4: 连接失败
  - 其它值: 发生未知错误

说明

- 如果*AT+SYSSTORE=1*，配置更改将保存到 NVS 分区
- 使用本命令需要开启 station 模式
- 本命令中的 `<reconn_interval>` 参数与*AT+CWRECONNCFG* 命令中的 `<interval_second>` 参数相同。如果运行本命令时不设置 `<reconn_interval>` 参数，Wi-Fi 重连间隔时间将采用默认值 1
- 如果同时省略 `<ssid>` 和 `<password>` 参数，将使用上一次设置的值
- 执行命令与设置命令的超时时间相同，默认为 15 秒，可通过参数 `<jap_timeout>` 设置
- 想要获取 IPv6 地址，需要先设置*AT+CIPV6=1*
- 回复 `OK` 代表 IPv4 网络已经准备就绪，而不代表 IPv6 网络准备就绪。当前 ESP-AT 以 IPv4 网络为主，IPv6 网络为辅。
- `WIFI GOT IPv6 LL` 代表已经获取到本地链路 IPv6 地址，这个地址是通过 EUI-64 本地计算出来的，不需要路由器参与。由于并行时序，这个打印可能在 `OK` 之前，也可能在 `OK` 之后。
- `WIFI GOT IPv6 GL` 代表已经获取到全局 IPv6 地址，该地址是由 AP 下发的前缀加上内部计算出来的后缀进行组合而来的，需要路由器参与。由于并行时序，这个打印可能在 `OK` 之前，也可能在 `OK` 之后；也可能由于 AP 不支持 IPv6 而不打印。

示例

```
// 如果目标 AP 的 SSID 是 "abc", 密码是 "0123456789", 则命令是:
AT+CWJAP="abc","0123456789"

// 如果目标 AP 的 SSID 是 "ab\,c", 密码是 "0123456789"\", 则命令是:
AT+CWJAP="ab\\\,c","0123456789\"\\"

// 如果多个 AP 有相同的 SSID "abc", 可通过 BSSID 找到目标 AP:
AT+CWJAP="abc","0123456789","ca:d7:19:d8:a6:44"

// 如果 ESP-AT 要求通过 PMF 连接 AP, 则命令是:
AT+CWJAP="abc","0123456789",,,,,,,3
```

3.2.3 AT+CWRECONNCFG：查询/设置 Wi-Fi 重连配置

查询命令

功能：

查询 Wi-Fi 重连配置

命令：

```
AT+CWRECONNCFG?
```

响应：

```
+CWRECONNCFG:<interval_second>,<repeat_count>
OK
```

设置命令

功能：

设置 Wi-Fi 重连配置

命令：

```
AT+CWRECONNCFG=<interval_second>,<repeat_count>
```

响应：

```
OK
```

参数

- **<interval_second>**：Wi-Fi 重连间隔，单位：秒，默认值：0，最大值 7200
  - 0: 断开连接后，ESP station 不重连 AP
  - [1,7200]: 断开连接后，ESP station 每隔指定的时间与 AP 重连
- **<repeat_count>**：ESP 设备尝试重连 AP 的次数，本参数在 <interval_second> 不为 0 时有效，默认值：0，最大值：1000
  - 0: ESP station 始终尝试连接 AP
  - [1,1000]: ESP station 按照本参数指定的次数重连 AP

示例

```
// ESP station 每隔 1 秒尝试重连 AP，共尝试 100 次
AT+CWRECONNCFG=1,100

// ESP station 在断开连接后不重连 AP
AT+CWRECONNCFG=0,0
```

说明

- 本命令中的 <interval_second> 参数与*AT+CWJAP* 中的 [<reconn_interval>] 参数相同
- 该命令适用于被动断开 AP、Wi-Fi 模式切换和开机后 Wi-Fi 自动连接

3.2.4 AT+CWLAPOPT：设置 AT+CWLAP 命令扫描结果的属性

设置命令

命令：

```
AT+CWLAPOPT=<reserved>,<print mask>[,<rssi filter>][,<authmode mask>]
```

响应：

```
OK
```

或者

```
ERROR
```

参数

- **<reserved>**：保留项
- **<print mask>**：*AT+CWLAP* 的扫描结果是否显示以下参数，默认值：0x7FF，若 bit 设为 1，则显示对应参数，若设为 0，则不显示对应参数
  - bit 0: 是否显示 <ecn>
  - bit 1: 是否显示 <ssid>
  - bit 2: 是否显示 <rssi>
  - bit 3: 是否显示 <mac>
  - bit 4: 是否显示 <channel>
  - bit 5: 是否显示 <freq_offset>
  - bit 6: 是否显示 <freqcal_val>
  - bit 7: 是否显示 <pairwise_cipher>
  - bit 8: 是否显示 <group_cipher>
  - bit 9: 是否显示 <bgn>
  - bit 10: 是否显示 <wps>
- **[<rssi filter>]**：*AT+CWLAP* 的扫描结果是否按照本参数过滤，也即，是否过滤掉信号强度低于 `rssi filter` 参数值的 AP，单位：dBm，默认值：–100，范围：[–100,40]
- **[<authmode mask>]**：*AT+CWLAP* 的扫描结果是否显示以下认证方式的 AP，默认值：0xFFFF，如果 `bit x` 设为 1，则显示对应认证方式的 AP，若设为 0，则不显示
  - bit 0: 是否显示 `OPEN` 认证方式的 AP
  - bit 1: 是否显示 `WEP` 认证方式的 AP
  - bit 2: 是否显示 `WPA_PSK` 认证方式的 AP
  - bit 3: 是否显示 `WPA2_PSK` 认证方式的 AP
  - bit 4: 是否显示 `WPA_WPA2_PSK` 认证方式的 AP
  - bit 5: 是否显示 `WPA2_ENTERPRISE` 认证方式的 AP
  - bit 6: 是否显示 `WPA3_PSK` 认证方式的 AP
  - bit 7: 是否显示 `WPA2_WPA3_PSK` 认证方式的 AP
  - [ESP32-C3 Only] bit 8: 是否显示 `WAPI_PSK` 认证方式的 AP

示例

```
// 第一个参数为 1，表示 AT+CWLAP 命令扫描结果按照信号强度 RSSI 值排序
// 第二个参数为 31，即 0x1F，表示所有值为 1 的 bit 对应的参数都会显示出来
AT+CWLAPOPT=1,31
AT+CWLAP

// 只显示认证方式为 OPEN 的 AP
AT+CWLAPOPT=1,31,-100,1
AT+CWLAP
```

3.2.5 AT+CWLAP：扫描当前可用的 AP

设置命令

功能：

列出符合特定条件的 AP，如指定 SSID、MAC 地址或信道号

命令：

```
AT+CWLAP=[<ssid>,<mac>,<channel>,<scan_type>,<scan_time_min>,<scan_time_max>]
```

执行命令

功能：

列出当前可用的 AP

命令：

```
AT+CWLAP
```

响应：

```
+CWLAP:<ecn>,<ssid>,<rssi>,<mac>,<channel>,<freq_offset>,<freqcal_val>,<pairwise_
↪cipher>,<group_cipher>,<bgn>,<wps>
OK
```

参数

- **<ecn>**：加密方式
  - 0: OPEN
  - 1: WEP
  - 2: WPA_PSK
  - 3: WPA2_PSK
  - 4: WPA_WPA2_PSK
  - 5: WPA2_ENTERPRISE
  - 6: WPA3_PSK
  - 7: WPA2_WPA3_PSK
  - [ESP32-C3 Only] 8: WAPI_PSK
- **<ssid>**：字符串参数，AP 的 SSID
- **<rssi>**：信号强度
- **<mac>**：字符串参数，AP 的 MAC 地址
- **<channel>**：信道号
- **<scan_type>**：Wi-Fi 扫描类型
  - 0: 主动扫描
  - 1: 被动扫描
- **<scan_time_min>**：每个信道最短扫描时间，单位：毫秒，范围：[0,1500]，如果扫描类型为被动扫描，本参数无效
- **<scan_time_max>**：每个信道最长扫描时间，单位：毫秒，范围：[0,1500]，如果设为 0，固件采用参数默认值，主动扫描为 120 ms，被动扫描为 360 ms
- **<freq_offset>**：频偏（保留项目）
- **<freqcal_val>**：频率校准值（保留项目）
- **<pairwise_cipher>**：成对加密类型
  - 0: None
  - 1: WEP40
  - 2: WEP104
  - 3: TKIP
  - 4: CCMP

  - 5: TKIP and CCMP
  - 6: AES-CMAC-128
  - 7: 未知
- **<group_cipher>**：组加密类型，与 <pairwise_cipher> 参数的枚举值相同
- **<bgn>**：802.11 b/g/n，若 bit 设为 1，则表示使能对应模式，若设为 0，则表示禁用对应模式
  - bit 0: 是否使能 802.11b 模式
  - bit 1: 是否使能 802.11g 模式
  - bit 2: 是否使能 802.11n 模式
- **<wps>**：wps flag
  - 0: 不支持 WPS
  - 1: 支持 WPS

说明

- 对于 ESP8266 设备，**<scan_time_min>** 和 **<scan_time_max>** 必须设置为相同的值。如果这两个参数被省略或者同时设置为 0 或者设置的值不相等，则固件采用参数默认值，主动扫描为 120 ms，被动扫描为 360 ms。

示例

```
AT+CWLAP="Wi-Fi","ca:d7:19:d8:a6:44",6,0,400,1000

// 寻找指定 SSID 的 AP
AT+CWLAP="Wi-Fi"
```

3.2.6 AT+CWQAP：断开与 AP 的连接

执行命令

命令：

```
AT+CWQAP
```

响应：

```
OK
```

3.2.7 AT+CWSAP：配置 ESP SoftAP 参数

查询命令

功能：

查询 ESP SoftAP 的配置参数

命令：

```
AT+CWSAP?
```

响应：

```
+CWSAP:<ssid>,<pwd>,<channel>,<ecn>,<max conn>,<ssid hidden>
OK
```

设置命令

功能：

设置 ESP SoftAP 的配置参数

命令：

```
AT+CWSAP=<ssid>,<pwd>,<chl>,<ecn>[,<max conn>][,<ssid hidden>]
```

响应：

```
OK
```

参数

- **<ssid>**：字符串参数，接入点名称
- **<pwd>**：字符串参数，密码，范围：8 ~ 64 字节 ASCII
- **<channel>**：信道号
- **<ecn>**：加密方式，不支持 WEP
  - 0: OPEN
  - 2: WPA_PSK
  - 3: WPA2_PSK
  - 4: WPA_WPA2_PSK
- **[<max conn>]**：允许连入 ESP SoftAP 的最多 station 数目，取值范围：[1,10]

- **[<ssid hidden>]**：
  - 0: 广播 SSID（默认）
  - 1: 不广播 SSID

说明

- 本指令只有当*AT+CWMODE=2* 或者*AT+CWMODE=3* 时才有效
- 若*AT+SYSSTORE=1*，配置更改将保存在 NVS 分区

示例

```
AT+CWSAP="ESP","1234567890",5,3
```

3.2.8 AT+CWLIF：查询连接到 ESP SoftAP 的 station 信息

执行命令

命令：

```
AT+CWLIF
```

响应：

```
+CWLIF:<ip addr>,<mac>

OK
```

参数

- **<ip addr>**：连接到 ESP SoftAP 的 station 的 IP 地址
- **<mac>**：连接到 ESP SoftAP 的 station 的 MAC 地址

说明

- 本指令无法查询静态 IP，仅支持在 ESP SoftAP 和连入的 station DHCP 均使能的情况下有效

3.2.9 AT+CWQIF：断开 station 与 ESP SoftAP 的连接

执行命令

功能：

断开所有连入 ESP SoftAP 的 station

命令：

```
AT+CWQIF
```

响应：

```
OK
```

设置命令

功能：

断开某个连入 ESP SoftAP 的 station

命令：

```
AT+CWQIF=<mac>
```

响应：

```
OK
```

参数

- **<mac>**：需断开连接的 station 的 MAC 地址

3.2.10 AT+CWDHCP：启用/禁用 DHCP

查询命令

命令：

```
AT+CWDHCP?
```

响应：

```
<state>
```

设置命令

功能：

启用/禁用 DHCP

命令：

```
AT+CWDHCP=<operate>,<mode>
```

响应：

```
OK
```

参数

- **<operate>**：
  - 0: 禁用
  - 1: 启用
- **<mode>**：
  - Bit0: Station 的 DHCP
  - Bit1: SoftAP 的 DHCP
- **<state>**：DHCP 的状态
  - Bit0:
    - 0: 禁用 Station 的 DHCP
    - 1: 启用 Station 的 DHCP
  - Bit1:

        * 0: 禁用 SoftAP 的 DHCP
        * 1: 启用 SoftAP 的 DHCP
    - Bit2 (ESP32 only):
        * 0: 禁用 Ethernet 的 DHCP
        * 1: 启用 Ethernet 的 DHCP

说明

- 若*AT+SYSSTORE=1*，配置更改将保存到 NVS 分区
- 本设置命令与设置静态 IP 地址的命令会相互影响，如*AT+CIPSTA* 和*AT+CIPAP*
    - 若启用 DHCP，则静态 IP 地址会被禁用
    - 若启用静态 IP，则 DHCP 会被禁用
    - 最后一次配置会覆盖上一次配置

示例

```
// 启用 Station DHCP，如果原 DHCP mode 为 2，则现 DHCP mode 为 3
AT+CWDHCP=1,1

// 禁用 SoftAP DHCP，如果原 DHCP mode 为 3，则现 DHCP mode 为 1
AT+CWDHCP=0,2
```

3.2.11 AT+CWDHCPS：查询/设置 ESP SoftAP DHCP 分配的 IP 地址范围

查询命令

命令：

```
AT+CWDHCPS?
```

响应：

```
+CWDHCPS=<lease time>,<start IP>,<end IP>
OK
```

设置命令

功能：

设置 ESP SoftAP DHCP 服务器分配的 IP 地址范围

命令：

```
AT+CWDHCPS=<enable>,<lease time>,<start IP>,<end IP>
```

响应：

```
OK
```

参数

- **<enable>**：
  - 1: 设置 DHCP server 信息，后续参数必须填写
  - 0: 清除 DHCP server 信息，恢复默认值，后续参数无需填写
- **<lease time>**：租约时间，单位：分钟，取值范围：[1,2880]
- **<start IP>**：ESP SoftAP DHCP 服务器 IP 地址池的起始 IP
- **<end IP>**：ESP SoftAP DHCP 服务器 IP 地址池的结束 IP

说明

- 若*AT+SYSSTORE=1*，配置更改将保存到 NVS 分区
- 本命令必须在 ESP SoftAP 模式使能，且开启 DHCP server 的情况下使用
- 设置的 IP 地址范围必须与 ESP SoftAP 在同一网段

示例

```
AT+CWDHCPS=1,3,"192.168.4.10","192.168.4.15"

AT+CWDHCPS=0 // 清除设置，恢复默认值
```

3.2.12 AT+CWAUTOCONN：上电是否自动连接 AP

设置命令

命令：

```
AT+CWAUTOCONN=<enable>
```

响应：

```
OK
```

参数

- **<enable>**：
  - 1: 上电自动连接 AP（默认）
  - 0: 上电不自动连接 AP

说明

- 本设置保存到 NVS 区域

示例

```
AT+CWAUTOCONN=1
```

3.2.13 AT+CWAPPROTO：查询/设置 SoftAP 模式下 802.11 b/g/n 协议标准

查询命令

命令：

```
AT+CWAPPROTO?
```

响应：

```
+CWAPPROTO=<protocol>
OK
```

设置命令

命令：

```
AT+CWAPPROTO=<protocol>
```

响应：

```
OK
```

参数

- **<protocol>**：
  - bit0: 802.11b 协议标准
  - bit1: 802.11g 协议标准
  - bit2: 802.11n 协议标准

说明

- 当前，ESP 设备只支持 802.11b、802.11bg 或 802.11bgn 协议标准
- 默认情况下，ESP8266 设备的 PHY mode 是 802.11bg 模式，非 ESP8266 设备的 PHY mode 是 802.11bgn 模式

3.2.14 AT+CWSTAPROTO：设置 Station 模式下 802.11 b/g/n 协议标准

查询命令

命令：

```
AT+CWSTAPROTO?
```

响应：

```
+CWSTAPROTO=<protocol>
OK
```

设置命令

命令：

```
AT+CWSTAPROTO=<protocol>
```

响应：

```
OK
```

参数

- **<protocol>**：
  - bit0: 802.11b 协议标准
  - bit1: 802.11g 协议标准
  - bit2: 802.11n 协议标准

说明

- 当前，ESP 设备只支持 802.11b、802.11bg 或 802.11bgn 协议标准
- 默认情况下，ESP8266 设备的 PHY mode 是 802.11bg 模式，非 ESP8266 设备的 PHY mode 是 802.11bgn 模式

3.2.15 AT+CIPSTAMAC：查询/设置 ESP Station 的 MAC 地址

查询命令

功能：

查询 ESP Station 的 MAC 地址

命令：

```
AT+CIPSTAMAC?
```

响应：

```
+CIPSTAMAC:<mac>
OK
```

设置命令

功能：

设置 ESP Station 的 MAC 地址

命令：

```
AT+CIPSTAMAC=<mac>
```

响应：

```
OK
```

参数

- **<mac>**：字符串参数，表示 ESP Station 的 MAC 地址

说明

- 若*AT+SYSSTORE=1*，配置更改将保存到 NVS 分区
- ESP SoftAP 的 MAC 地址与 ESP Station 不同，不要为二者设置同样的 MAC 地址
- MAC 地址的 Bit 0 不能为 1，例如，MAC 地址可以是“1a:…”，但不可以是“15:…”
- FF:FF:FF:FF:FF:FF 和 00:00:00:00:00:00 是无效地址，不能设置

示例

```
AT+CIPSTAMAC="1a:fe:35:98:d3:7b"
```

3.2.16 AT+CIPAPMAC：查询/设置 ESP SoftAP 的 MAC 地址

查询命令

功能：

查询 ESP SoftAP 的 MAC 地址

命令：

```
AT+CIPAPMAC?
```

响应：

```
+CIPAPMAC:<mac>
OK
```

设置命令

功能：

设置 ESP SoftAP 的 MAC 地址

命令：

```
AT+CIPAPMAC=<mac>
```

响应：

```
OK
```

参数

- **<mac>**：字符串参数，表示 ESP SoftAP 的 MAC 地址

说明

- 若*AT+SYSSTORE=1*，配置更改将保存到 NVS 分区
- ESP SoftAP 的 MAC 地址与 ESP Station 不同，不要为二者设置同样的 MAC 地址
- MAC 地址的 Bit 0 不能为 1，例如，MAC 地址可以是“18:…”，但不可以是“15:…”
- FF:FF:FF:FF:FF:FF 和 00:00:00:00:00:00 是无效地址，不能设置

示例

```
AT+CIPAPMAC="18:fe:35:98:d3:7b"
```

3.2.17 AT+CIPSTA：查询/设置 ESP Station 的 IP 地址

查询命令

功能：

查询 ESP Station 的 IP 地址

命令：

```
AT+CIPSTA?
```

响应：

```
+CIPSTA:ip:<"ip">
+CIPSTA:gateway:<"gateway">
+CIPSTA:netmask:<"netmask">
+CIPSTA:ip6ll:<"ipv6 addr">
+CIPSTA:ip6gl:<"ipv6 addr">

OK
```

设置命令

功能：

设置 ESP Station 的 IPv4 地址

命令：

```
AT+CIPSTA=<"ip">[,<"gateway">,<"netmask">]
```

响应：

```
OK
```

参数

- **<" ip" >**：字符串参数，表示 ESP station 的 IPv4 地址
- **<" gateway" >**：网关
- **<" netmask" >**：子网掩码
- **<" ipv6 addr" >**：ESP station 的 IPv6 地址

说明

- 使用查询命令时，只有当 ESP station 连入 AP 或者配置过静态 IP 地址后，才能查询到它的 IP 地址
- 若*AT+SYSSTORE=1*，配置更改将保存到 NVS 分区
- 本设置命令与设置 DHCP 的命令相互影响，如*AT+CWDHCP*
  - 若启用静态 IP 地址，则禁用 DHCP
  - 若启用 DHCP，则禁用静态 IP 地址

- 最后一次配置会覆盖上一次配置

示例

```
AT+CIPSTA="192.168.6.100","192.168.6.1","255.255.255.0"
```

3.2.18 AT+CIPAP：查询/设置 ESP SoftAP 的 IP 地址

查询命令

功能：

查询 ESP SoftAP 的 IP 地址

命令：

```
AT+CIPAP?
```

响应：

```
+CIPAP:ip:<"ip">
+CIPAP:gateway:<"gateway">
+CIPAP:netmask:<"netmask">
+CIPAP:ip6ll:<"ipv6 addr">
+CIPAP:ip6gl:<"ipv6 addr">

OK
```

设置命令

功能：

设置 ESP SoftAP 的 IPv4 地址

命令：

```
AT+CIPAP=<"ip">[,<"gateway">,<"netmask">]
```

响应：

```
OK
```

参数

- **<" ip" >**：字符串参数，表示 ESP SoftAP 的 IPv4 地址
- **<" gateway" >**：网关
- **<" netmask" >**：子网掩码
- **<" ipv6 addr" >**：ESP SoftAP 的 IPv6 地址

说明

- 若*AT+SYSSTORE=1*，配置更改将保存到 NVS 分区
- 本设置命令与设置 DHCP 的命令相互影响，如*AT+CWDHCP*
  - 若启用静态 IP 地址，则禁用 DHCP
  - 若启用 DHCP，则禁用静态 IP 地址
  - 最后一次配置会覆盖上一次配置

示例

```
AT+CIPAP="192.168.5.1","192.168.5.1","255.255.255.0"
```

3.2.19 AT+CWSTARTSMART：开启 SmartConfig

执行命令

功能：

开启 ESP-TOUCH+AirKiss 兼容模式

命令：

```
AT+CWSTARTSMART
```

设置命令

功能：

开启某指定类型的 SmartConfig

命令：

```
AT+CWSTARTSMART=<type>[,<auth floor>]
```

响应：

```
OK
```

参数

- **<type>**：类型
  - 1: ESP-TOUCH
  - 2: AirKiss
  - 3: ESP-TOUCH+AirKiss
- **<auth floor>**: Wi-Fi 认证模式阈值，ESP-AT 不会连接到 authmode 低于此阈值的 AP
  - 0: OPEN（默认）
  - 1: WEP
  - 2: WPA_PSK
  - 3: WPA2_PSK
  - 4: WPA_WPA2_PSK
  - 5: WPA2_ENTERPRISE
  - 6: WPA3_PSK
  - 7: WPA2_WPA3_PSK

说明

- 更多有关 SmartConfig 的信息，请参考 ESP-TOUCH 使用指南；
- SmartConfig 仅支持在 ESP Station 模式下调用；
- 消息 `Smart get Wi-Fi info` 表示 SmartConfig 成功获取到 AP 信息，之后 ESP 尝试连接 AP；
- 消息 `Smartconfig connected Wi-Fi` 表示成功连接到 AP；
- 因为 ESP 设备需要将 SmartConfig 配网结果同步给手机端，所以建议在消息 `Smartconfig connected Wi-Fi` 输出后延迟超过 6 秒再调用*AT+CWSTOPSMART*；
- 可调用*AT+CWSTOPSMART* 停止 SmartConfig，然后再执行其他命令。注意，在 SmartConfig 过程中请勿执行其他命令。

示例

```
AT+CWMODE=1
AT+CWSTARTSMART
```

3.2.20 AT+CWSTOPSMART：停止 SmartConfig

执行命令

命令：

```
AT+CWSTOPSMART
```

响应：

```
OK
```

说明

- 无论 SmartConfig 成功与否，都请在执行其他命令之前调用*AT+CWSTOPSMART* 释放 SmartConfig 占用的内存

示例

```
AT+CWMODE=1
AT+CWSTARTSMART
AT+CWSTOPSMART
```

3.2.21 AT+WPS：设置 WPS 功能

设置命令

命令：

```
AT+WPS=<enable>[,<auth floor>]
```

响应：

```
OK
```

参数

- **<enable>**：
  - 1: 开启 PBC 类型的 WPS
  - 0: 关闭 PBC 类型的 WPS
- **<auth floor>**: Wi-Fi 认证模式阈值，ESP-AT 不会连接到 authmode 低于此阈值的 AP
  - 0: OPEN（默认）
  - 1: WEP
  - 2: WPA_PSK
  - 3: WPA2_PSK
  - 4: WPA_WPA2_PSK
  - 5: WPA2_ENTERPRISE
  - 6: WPA3_PSK
  - 7: WPA2_WPA3_PSK

说明

- WPS 功能必须在 ESP Station 使能的情况下调用
- WPS 不支持 WEP 加密方式

示例

```
AT+CWMODE=1
AT+WPS=1
```

3.2.22 AT+MDNS：设置 mDNS 功能

设置命令

命令：

```
AT+MDNS=<enable>[,<hostname>,<service_name>,<port>]
```

响应：

```
OK
```

参数

- **<enable>**：
  - 1: 开启 mDNS 功能，后续参数需要填写
  - 0: 关闭 mDNS 功能，后续参数无需填写
- **<hostname>**：mDNS 主机名称
- **<service_name>**：mDNS 服务名称
- **<port>**：mDNS 端口

示例

```
AT+MDNS=1,"espressif","_iot",8080
AT+MDNS=0
```

3.2.23 [ESP32 Only] AT+CWJEAP：连接 WPA2 企业版 AP

查询命令

功能：

查询 ESP station 连入的企业版 AP 的配置信息

命令：

```
AT+CWJEAP?
```

响应：

```
+CWJEAP:<ssid>,<method>,<identity>,<username>,<password>,<security>
OK
```

设置命令

功能：

连接到目标企业版 AP

命令：

```
AT+CWJEAP=<ssid>,<method>,<identity>,<username>,<password>,<security>[,<jeap_timeout>]
```

响应：

```
OK
```

或

```
+CWJEAP:Timeout
ERROR
```

参数

- **<ssid>**：企业版 AP 的 SSID
  - 如果 SSID 或密码中包含 ,、"、\\ 等特殊字符，需转义
- **<method>**：WPA2 企业版认证方式
  - 0: EAP-TLS
  - 1: EAP-PEAP
  - 2: EAP-TTLS
- **<identity>**：阶段 1 的身份，字符串限制为 1 ~ 32
- **<username>**：阶段 2 的用户名，范围：1 ~ 32 字节，EAP-PEAP、EAP-TTLS 两种认证方式需设置本参数，EAP-TLS 方式无需设置本参数
- **<password>**：阶段 2 的密码，范围：1 ~ 32 字节，EAP-PEAP、EAP-TTLS 两种认证方式需设置本参数，EAP-TLS 方式无需设置本参数
- **<security>**：
  - Bit0: 客户端证书
  - Bit1: 服务器证书
- **[<jeap_timeout>]**：*AT+CWJEAP* 命令的最大超时时间，单位：秒，默认值：15，范围：[3,600]

示例

```
// 连接至 EAP-TLS 认证方式的企业版 AP，设置身份，验证服务器证书，加载客户端证书
AT+CWJEAP="dlink11111",0,"example@espressif.com",,,3

// 连接至 EAP-PEAP 认证方式的企业版 AP，设置身份、用户名、密码，不验证服务器证书，不加载客户端证书
AT+CWJEAP="dlink11111",1,"example@espressif.com","espressif","test11",0
```

错误代码:

WPA2 企业版错误码以 `ERR CODE:0x<%08x>` 格式打印：

AT_EAP_MALLOC_FAILED	0x8001
AT_EAP_GET_NVS_CONFIG_FAILED	0x8002
AT_EAP_CONN_FAILED	0x8003
AT_EAP_SET_WIFI_CONFIG_FAILED	0x8004
AT_EAP_SET_IDENTITY_FAILED	0x8005
AT_EAP_SET_USERNAME_FAILED	0x8006
AT_EAP_SET_PASSWORD_FAILED	0x8007
AT_EAP_GET_CA_LEN_FAILED	0x8008
AT_EAP_READ_CA_FAILED	0x8009
AT_EAP_SET_CA_FAILED	0x800A
AT_EAP_GET_CERT_LEN_FAILED	0x800B
AT_EAP_READ_CERT_FAILED	0x800C
AT_EAP_GET_KEY_LEN_FAILED	0x800D
AT_EAP_READ_KEY_FAILED	0x800E
AT_EAP_SET_CERT_KEY_FAILED	0x800F
AT_EAP_ENABLE_FAILED	0x8010
AT_EAP_ALREADY_CONNECTED	0x8011
AT_EAP_GET_SSID_FAILED	0x8012
AT_EAP_SSID_NULL	0x8013
AT_EAP_SSID_LEN_ERROR	0x8014
AT_EAP_GET_METHOD_FAILED	0x8015
AT_EAP_CONN_TIMEOUT	0x8016
AT_EAP_GET_IDENTITY_FAILED	0x8017
AT_EAP_IDENTITY_LEN_ERROR	0x8018
AT_EAP_GET_USERNAME_FAILED	0x8019
AT_EAP_USERNAME_LEN_ERROR	0x801A
AT_EAP_GET_PASSWORD_FAILED	0x801B
AT_EAP_PASSWORD_LEN_ERROR	0x801C
AT_EAP_GET_SECURITY_FAILED	0x801D
AT_EAP_SECURITY_ERROR	0x801E
AT_EAP_METHOD_SECURITY_UNMATCHED	0x801F
AT_EAP_PARAMETER_COUNTS_ERROR	0x8020
AT_EAP_GET_WIFI_MODE_ERROR	0x8021
AT_EAP_WIFI_MODE_NOT_STA	0x8022
AT_EAP_SET_CONFIG_FAILED	0x8023
AT_EAP_METHOD_ERROR	0x8024

说明

- 若*AT+SYSSTORE=1*，配置更改将保存到 NVS 分区
- 使用本命令需开启 Station 模式
- 使用 TLS 认证方式需使能客户端证书

3.2.24 AT+CWHOSTNAME：查询/设置 ESP Station 的主机名称

查询命令

功能：

查询 ESP Station 的主机名称

命令：

```
AT+CWHOSTNAME?
```

响应：

```
+CWHOSTNAME:<hostname>

OK
```

设置命令

功能：

设置 ESP Station 的主机名称

命令：

```
AT+CWHOSTNAME=<hostname>
```

响应：

```
OK
```

若没开启 Station 模式，则返回：

```
ERROR
```

参数

- **<hostname>**：ESP Station 的主机名称，最大长度：32 字节

说明

- 配置更改不保存到 flash

示例

```
AT+CWMODE=3
AT+CWHOSTNAME="my_test"
```

3.2.25 AT+CWCOUNTRY：查询/设置 Wi-Fi 国家代码

查询命令

功能：

查询 Wi-Fi 国家代码

命令：

```
AT+CWCOUNTRY?
```

响应：

```
+CWCOUNTRY:<country_policy>,<country_code>,<start_channel>,<total_channel_count>

OK
```

设置命令

功能：

设置 Wi-Fi 国家代码

命令：

```
AT+CWCOUNTRY=<country_policy>,<country_code>,<start_channel>,<total_channel_count>
```

响应：

```
OK
```

参数

- **<country_policy>**：
  - 0: 将国家代码改为 ESP 设备连入的 AP 的国家代码
  - 1: 不改变国家代码，始终保持本命令设置的国家代码
- **<country_code>**：国家代码，最大长度：3 个字符
- **<start_channel>**：起始信号道，范围：[1,14]
- **<total_channel_count>**：信道总个数

说明

- 配置更改不保存到 flash

示例

```
AT+CWMODE=3
AT+CWCOUNTRY=1,"CN",1,13
```

3.3 TCP/IP AT 命令

[English]

- *AT+CIPV6*: 启用/禁用 IPv6 网络 (IPv6)
- *AT+CIPSTATUS*：查询 TCP/UDP/SSL 连接状态和信息
- *AT+CIPDOMAIN*：域名解析
- *AT+CIPSTART*：建立 TCP 连接、UDP 传输或 SSL 连接
- *AT+CIPSTARTEX*：建立自动分配 ID 的 TCP 连接、UDP 传输或 SSL 连接
- *[仅适用透传模式]* +++: 退出透传模式
- *AT+CIPSEND*：在普通传输模式 或 Wi-Fi 透传模式 下发送数据
- *AT+CIPSENDEX*：在普通传输模式 下采用扩展的方式发送数据
- *AT+CIPCLOSE*：关闭 TCP/UDP/SSL 连接
- *AT+CIFSR*：查询本地 IP 地址和 MAC 地址

- *AT+CIPMUX*：启用/禁用多连接模式
- *AT+CIPSERVER*：建立/关闭 TCP 或 SSL 服务器
- *AT+CIPSERVERMAXCONN*：查询/设置服务器允许建立的最大连接数
- *AT+CIPMODE*：查询/设置传输模式
- *AT+SAVETRANSLINK*：设置开机透传模式 信息
- *AT+CIPSTO*：查询/设置本地 TCP 服务器超时时间
- *AT+CIPSNTPCFG*：查询/设置时区和 SNTP 服务器
- *AT+CIPSNTPTIME*：查询 SNTP 时间
- *AT+CIUPDATE*：通过 Wi-Fi 升级固件
- *AT+CIPDINFO*：设置 +IPD 消息详情
- *AT+CIPSSLCCONF*：查询/设置 SSL 客户端配置
- *AT+CIPSSLCCN*：查询/设置 SSL 客户端的公用名 (common name)
- *AT+CIPSSLCSNI*：查询/设置 SSL 客户端的 SNI
- *AT+CIPSSLCALPN*：查询/设置 SSL 客户端 ALPN
- *AT+CIPSSLCPSK*：查询/设置 SSL 客户端的 PSK
- *AT+CIPRECONNINTV*：查询/设置 Wi-Fi 透传模式 下的 TCP/UDP/SSL 重连间隔
- *AT+CIPRECVMODE*：查询/设置 socket 接收模式
- *AT+CIPRECVDATA*：获取被动接收模式下的 socket 数据
- *AT+CIPRECVLEN*：查询被动接收模式下 socket 数据的长度
- *AT+PING*：ping 对端主机
- *AT+CIPDNS*：查询/设置 DNS 服务器信息
- *AT+CIPTCPOPT*：查询/设置 socket 选项

3.3.1 AT+CIPV6: 启用/禁用 IPv6 网络 (IPv6)

查询命令

功能：

查询 IPv6 网络是否使能

命令：

```
AT+CIPV6?
```

响应：

```
+CIPV6:<enable>

OK
```

设置命令

功能：

启用/禁用 IPv6 网络

命令：

```
AT+CIPV6=<enable>
```

响应：

```
OK
```

参数

- **<enable>**: IPv6 网络使能状态。默认值：0
  - 0: 禁用 IPv6 网络
  - 1: 启用 IPv6 网络

说明

- 在使用基于 IPv6 网络的上层应用前，需要先启用 IPv6 网络。（例如：基于 IPv6 网络使用 TCP/UDP/SSL/PING/DNS，也称为 TCP6/UDP6/SSL6/PING6/DNS6 或 TCPv6/UDPv6/SSLv6/PINGv6/DNSv6）

3.3.2 AT+CIPSTATUS：查询 TCP/UDP/SSL 连接状态和信息

执行命令

命令：

```
AT+CIPSTATUS
```

响应：

```
STATUS:<stat>
+CIPSTATUS:<link ID>,<"type">,<"remote IP">,<remote port>,<local port>,<tetype>
OK
```

参数

- **<stat>**：ESP station 接口的状态
  - 0: ESP station 为未初始化状态
  - 1: ESP station 为已初始化状态，但还未开始 Wi-Fi 连接
  - 2: ESP station 已连接 AP，获得 IP 地址
  - 3: ESP station 已建立 TCP、UDP 或 SSL 传输
  - 4: ESP 设备所有的 TCP、UDP 和 SSL 均断开
  - 5: ESP station 开始过 Wi-Fi 连接，但尚未连接上 AP 或从 AP 断开
- **<link ID>**：网络连接 ID (0 ~ 4)，用于多连接的情况
- **<" type" >**：字符串参数，表示传输类型：" TCP"、" UDP"、" SSL"、" TCPv6"、" UDPv6" 或 "SSLv6"
- **<" remote IP" >**：字符串参数，表示远端 IPv4 地址或 IPv6 地址
- **<remote port>**：远端端口值
- **<local port>**：ESP 本地端口值
- **<tetype>**:
  - 0: ESP 设备作为客户端
  - 1: ESP 设备作为服务器

3.3.3 AT+CIPDOMAIN：域名解析

设置命令

命令：

```
AT+CIPDOMAIN=<"domain name">[,<ip network>]
```

响应：

```
+CIPDOMAIN:<"IP address">
OK
```

参数

- **<" domain name" >**：待解析的域名
- **<ip network>**：首选 IP 网络。默认值：1
  - 1：首选解析为 IPv4 地址
  - 2：只解析为 IPv4 地址
  - 3：只解析为 IPv6 地址
- **<" IP address" >**：解析出的 IP 地址

示例

```
AT+CWMODE=1                          // 设置 station 模式
AT+CWJAP="SSID","password"           // 连接网络
AT+CIPDOMAIN="iot.espressif.cn"      // 域名解析

// 域名解析，只解析为 IPv4 地址
AT+CIPDOMAIN="iot.espressif.cn",2

// 域名解析，只解析为 IPv6 地址
AT+CIPDOMAIN="ipv6.test-ipv6.com",3

// 域名解析，首选解析为 IPv4 地址
AT+CIPDOMAIN="ds.test-ipv6.com",1
```

3.3.4 AT+CIPSTART：建立 TCP 连接、UDP 传输或 SSL 连接

建立 TCP 连接

设置命令

命令：

```
// 单连接 (AT+CIPMUX=0)：
AT+CIPSTART=<"type">,<"remote host">,<remote port>[,<keep alive>][,<"local IP">]

// 多连接 (AT+CIPMUX=1)：
AT+CIPSTART=<link ID>,<"type">,<"remote host">,<remote port>[,<keep alive>][,<"local
↪IP">]
```

响应：

```
CONNECT

OK
```

参数

- **<link ID>**：网络连接 ID (0 ~ 4)，用于多连接的情况
- **<" type" >**：字符串参数，表示网络连接类型，" TCP" 或 "TCPv6"。默认值：" TCP"
- **<" remote host" >**：字符串参数，表示远端 IPv4 地址、IPv6 地址，或域名
- **<remote port>**：远端端口值
- **<keep alive>**：TCP keep-alive 间隔，默认值：0
  - 0: 禁用 TCP keep-alive 功能
  - 1 ~ 7200: 检测间隔，单位：秒
- **<" local IP" >**：连接绑定的本机 IPv4 地址或 IPv6 地址，该参数在本地多网络接口时和本地多 IP 地址时非常有用。默认为禁用，如果您想使用，需自行设置，空值也为有效值

说明

- 如果想基于 IPv6 网络建立 TCP 连接，需要先设置*AT+CIPV6=1*，再通过*AT+CWJAP* 获取到一个 IPv6 地址
- `<keep alive>` 参数最终会配置到 socket 选项 `TCP_KEEPIDLE`，keepalive 另外的 socket 选项 `TCP_KEEPINTVL` 默认会使用 `1`，`TCP_KEEPCNT` 默认会使用 `3`

示例

```
AT+CIPSTART="TCP","iot.espressif.cn",8000
AT+CIPSTART="TCP","192.168.101.110",1000
AT+CIPSTART="TCP","192.168.101.110",1000,,"192.168.101.100"
AT+CIPSTART="TCPv6","test-ipv6.com",80
AT+CIPSTART="TCPv6","fe80::860d:8eff:fe9d:cd90",1000,,"fe80::411c:1fdb:22a6:4d24"

// esp-at 已通过 AT+CWJAP 获取到 IPv6 全局地址
AT+CIPSTART="TCPv6","2404:6800:4005:80b::2004",80,,
↪"240e:3a1:2070:11c0:32ae:a4ff:fe80:65ac"
```

建立 UDP 传输

设置命令

命令：

```
// 单连接：(AT+CIPMUX=0)
AT+CIPSTART=<"type">,<"remote host">,<remote port>[,<local port>,<mode>,<"local IP">]

// 多连接：(AT+CIPMUX=1)
AT+CIPSTART=<link ID>,<"type">,<"remote host">,<remote port>[,<local port>,<mode>,<
↪"local IP">]
```

响应：

```
CONNECT

OK
```

参数

- **<link ID>**：网络连接 ID (0 ~ 4)，用于多连接的情况
- **<" type" >**：字符串参数，表示网络连接类型，" UDP" 或 "UDPv6"。默认值：" TCP"
- **<" remote host" >**：字符串参数，表示远端 IPv4 地址、IPv6 地址，或域名
- **<remote port>**：远端端口值
- **<local port>**：ESP 设备的 UDP 端口值
- **<mode>**：在 UDP Wi-Fi 透传下，本参数的值必须设为 0
  - 0: 接收到 UDP 数据后，不改变对端 UDP 地址信息（默认）
  - 1: 仅第一次接收到与初始设置不同的对端 UDP 数据时，改变对端 UDP 地址信息为发送数据设备的 IP 地址和端口
  - 2: 每次接收到 UDP 数据时，都改变对端 UDP 地址信息为发送数据的设备的 IP 地址和端口
- **<" local IP" >**：连接绑定的本机 IPv4 地址或 IPv6 地址，该参数在本地多网络接口时和本地多 IP 地址时非常有用。默认为禁用，如果您想使用，需自行设置，空值也为有效值

说明

- 如果 UDP 连接中的远端 IP 地址是 IPv4 组播地址 (224.0.0.0 ~ 239.255.255.255)，ESP 设备将发送和接收 UDPv4 组播
- 如果 UDP 连接中的远端 IP 地址是 IPv4 广播地址 (255.255.255.255)，ESP 设备将发送和接收 UDPv4 广播
- 如果 UDP 连接中的远端 IP 地址是 IPv6 组播地址 (FF00:0:0:0:0:0:0:0 ~ FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)，ESP 设备将基于 IPv6 网络，发送和接收 UDP 组播
- 使用参数 <mode> 前，需先设置参数 <local port>
- 如果想基于 IPv6 网络建立 UDP 连接，需要先设置*AT+CIPV6=1*，再通过*AT+CWJAP* 获取到一个 IPv6 地址

示例

```
// UDPv4 单播
AT+CIPSTART="UDP","192.168.101.110",1000,1002,2
AT+CIPSTART="UDP","192.168.101.110",1000,,,"192.168.101.100"

// 基于 IPv6 网络的 UDP 单播
AT+CIPSTART="UDPv6","fe80::32ae:a4ff:fe80:65ac",1000,,,"fe80::5512:f37f:bb03:5d9b"

// 基于 IPv6 网络的 UDP 多播
AT+CIPSTART="UDPv6","FF02::FC",1000,1002,0
```

建立 SSL 连接

设置命令

命令：

```
AT+CIPSTART=[<link ID>,]<"type">,<"remote host">,<remote port>[,<keep alive>,<"local␣
↪IP">]
```

响应：

```
OK
```

参数

- **<link ID>**：网络连接 ID (0 ~ 4)，用于多连接的情况
- **<" type" >**：字符串参数，表示网络连接类型，" SSL" 或 "SSLv6"。默认值：" TCP"
- **<" remote host" >**：字符串参数，表示远端 IPv4 地址、IPv6 地址，或域名
- **<remote port>**：远端端口值
- **<keep alive>**：SSL 保留配置，默认值：0
- **<" local IP" >**：连接绑定的本机 IPv4 地址或 IPv6 地址，该参数在本地多网络接口时和本地多 IP 地址时非常有用。默认为禁用，如果您想使用，需自行设置，空值也为有效值

说明

- SSL 连接数量取决于可用内存和最大连接数量；由于内存的限制，ESP8266 只能够创建一条 SSL 连接
- SSL 连接需占用大量内存，内存不足会导致系统重启
- 如果 AT+CIPSTART 命令是基于 SSL 连接，且每个数据包的超时时间为 10 秒，则总超时时间会变得更长，具体取决于握手数据包的个数
- 如果想基于 IPv6 网络建立 SSL 连接，需要先设置*AT+CIPV6=1*, 再通过*AT+CWJAP* 获取到一个 IPv6 地址
- <keep alive> 参数最终会配置到 socket 选项 TCP_KEEPIDLE，keepalive 另外的 socket 选项 TCP_KEEPINTVL 默认会使用 1，TCP_KEEPCNT 默认会使用 3

示例

```
AT+CIPSTART="SSL","iot.espressif.cn",8443
AT+CIPSTART="SSL","192.168.101.110",1000,,"192.168.101.100"

// esp-at 已通过 AT+CWJAP 获取到 IPv6 全局地址
AT+CIPSTART="SSLv6","240e:3a1:2070:11c0:6972:6f96:9147:d66d",1000,,
↪"240e:3a1:2070:11c0:55ce:4e19:9649:b75"
```

3.3.5 AT+CIPSTARTEX：建立自动分配 ID 的 TCP 连接、UDP 传输或 SSL 连接

本命令与*AT+CIPSTART* 相似，不同点在于：在多连接的情况下 (*AT+CIPMUX=1*) 无需手动分配 ID，系统会自动为新建的连接分配 ID。

3.3.6 [仅适用透传模式] +++：退出透传模式

特殊执行命令

功能：

退出透传模式，进入透传接收模式

Command:

```
// 仅适用透传模式
+++
```

说明

- 此特殊执行命令包含有三个相同的 + 字符（即 ASCII 码：0x2b），同时命令结尾没有 CR-LF 字符
- 确保第一个 + 字符前至少有 20 ms 时间间隔内没有其他输入，第三个 + 字符后至少有 20 ms 时间间隔内没有其他输入，三个 + 字符之间至多有 20 ms 时间间隔内没有其他输入。否则，+ 字符会被当做普通透传数据发送出去
- 本条特殊执行命令没有命令回复

3.3.7 AT+CIPSEND：在普通传输模式或 Wi-Fi 透传模式下发送数据

设置命令

功能：

普通传输模式 下，指定长度发送数据

命令：

```
// 单连接：(AT+CIPMUX=0)
AT+CIPSEND=<length>

// 多连接：(AT+CIPMUX=1)
AT+CIPSEND=<link ID>,<length>
```

(下页继续)

(续上页)

```
// UDP 传输可指定对端主机和端口
AT+CIPSEND=[<link ID>,]<length>[,<"remote host">,<remote port>]
```

响应：

```
OK


>
```

上述响应表示 AT 已准备好接收串行数据，此时您可以输入数据，当 AT 接收到的数据长度达到 <length> 后，数据传输开始。

如果未建立连接或数据传输时连接被断开，返回：

```
ERROR
```

如果数据传输成功，返回：

```
SEND OK
```

执行命令

功能：

进入 Wi-Fi 透传模式

命令：

```
AT+CIPSEND
```

响应：

```
OK
>
```

或

```
ERROR
```

进入 Wi-Fi 透传模式，ESP8266 设备每次最大接收 2048 字节，最大发送 1460 字节；其他 ESP 设备每次最大接收 8192 字节，最大发送 2920 字节。如果当前接收的数据长度大于最大发送字节数，AT 将立即发送；否则，接收的数据将在 20 ms 内发送。当输入单独一包+++ 时，退出透传模式 下的数据发送模式，请至少间隔 1 秒再发下一条 AT 命令。

本命令必须在开启透传模式 以及单连接下使用。若为 Wi-Fi UDP 透传，*AT+CIPSTART* 命令的参数 <mode> 必须设置为 0。

参数

- **<link ID>**：网络连接 ID (0 ~ 4)，用于多连接的情况
- **<length>**：数据长度，最大值：2048 字节
- **<" remote host" >**：UDP 传输可以指定对端主机：IPv4 地址、IPv6 地址，或域名
- **<remote port>**：UDP 传输可以指定对端端口

3.3.8 AT+CIPSENDEX：在普通传输模式下采用扩展的方式发送数据

设置命令

功能：

普通传输模式 下，指定长度发送数据，或者使用字符串 \0 (0x5c, 0x30 ASCII) 触发数据发送

命令：

```
// 单连接：(AT+CIPMUX=0)
AT+CIPSENDEX=<length>

// 多连接：(AT+CIPMUX=1)
AT+CIPSENDEX=<link ID>,<length>

// UDP 传输可指定对端 IP 地址和端口：
AT+CIPSENDEX=[<link ID>,]<length>[,<"remote host">,<remote port>]
```

响应：

```
OK
>
```

上述响应表示 AT 已准备好接收串行数据，此时您可以输入指定长度的数据，当 AT 接收到的数据长度达到 <length> 后或数据中出现 \0 字符时，数据传输开始。

如果未建立连接或数据传输时连接被断开，返回：

```
ERROR
```

如果数据传输成功，返回：

```
SEND OK
```

参数

- **<link ID>**：网络连接 ID (0 ~ 4)，用于多连接的情况
- **<length>**：数据长度，最大值：2048 字节
- **<" remote host" >**：UDP 传输可以指定对端主机：IPv4 地址、IPv6 地址，或域名
- **<remote port>**：UDP 传输可以指定对端端口

说明

- 当数据长度满足要求时，或数据中出现 \0 字符时 (0x5c，0x30 ASCII)，数据传输开始，系统返回普通命令模式，等待下一条 AT 命令
- 如果数据中包含 \<any>，则会去掉反斜杠，只使用 <any> 符号
- 如果需要发送 \0，请转义为 \\0

3.3.9 AT+CIPCLOSE：关闭 TCP/UDP/SSL 连接

设置命令

功能：

关闭多连接模式下的 TCP/UDP/SSL 连接

命令：

```
AT+CIPCLOSE=<link ID>
```

执行命令

功能：

关闭单连接模式下的 TCP/UDP/SSL 连接

```
AT+CIPCLOSE
```

响应：

```
OK
```

参数

- **<link ID>**：需关闭的网络连接 ID，如果设为 5，则表示关闭所有连接

3.3.10 AT+CIFSR：查询本地 IP 地址和 MAC 地址

执行命令

命令：

```
AT+CIFSR
```

响应：

```
+CIFSR:APIP,<"APIP">
+CIFSR:APIP6LL,<"APIP6LL">
+CIFSR:APIP6GL,<"APIP6GL">
+CIFSR:APMAC,<"APMAC">
+CIFSR:STAIP,<"STAIP">
+CIFSR:STAIP6LL,<"STAIP6LL">
+CIFSR:STAIP6GL,<"STAIP6GL">
+CIFSR:STAMAC,<"STAMAC">
+CIFSR:ETHIP,<"ETHIP">
+CIFSR:ETHIP6LL,<"ETHIP6LL">
+CIFSR:ETHIP6GL,<"ETHIP6GL">
+CIFSR:ETHMAC,<"ETHMAC">

OK
```

参数

- **<" APIP" >**: ESP SoftAP 的 IPv4 地址
- **<" APIP6LL" >**: ESP SoftAP 的 IPv6 本地链路地址
- **<" APIP6GL" >**: ESP SoftAP 的 IPv6 全局地址
- **<" APMAC" >**: ESP SoftAP 的 MAC 地址
- **<" STAIP" >**: ESP station 的 IPv4 地址
- **<" STAIP6LL" >**: ESP station 的 IPv6 本地链路地址
- **<" STAIP6GL" >**: ESP station 的 IPv6 全局地址
- **<" STAMAC" >**: ESP station 的 MAC 地址
- **<" ETHIP" >**: ESP ethernet 的 IPv4 地址

- **<"ETHIP6LL">**: ESP ethernet 的 IPv6 本地链路地址
- **<"ETHIP6GL">**: ESP ethernet 的 IPv6 全局地址
- **<"ETHMAC">**: ESP ethernet 的 MAC 地址

说明

- 只有当 ESP 设备获取到有效接口信息后，才能查询到它的 IP 地址和 MAC 地址

3.3.11 AT+CIPMUX：启用/禁用多连接模式

查询命令

功能：

查询连接模式

命令：

```
AT+CIPMUX?
```

响应：

```
+CIPMUX:<mode>
OK
```

设置命令

功能：

设置连接模式

命令：

```
AT+CIPMUX=<mode>
```

响应：

```
OK
```

参数

- **<mode>**：连接模式，默认值：0
  - 0: 单连接
  - 1: 多连接

说明

- 只有当所有连接都断开时才可更改连接模式
- 只有普通传输模式 (*AT+CIPMODE=0*)，才能设置为多连接
- 如果建立了 TCP/SSL 服务器，想切换为单连接，必须关闭服务器 (*AT+CIPSERVER=0*)

示例

```
AT+CIPMUX=1
```

3.3.12 AT+CIPSERVER：建立/关闭 TCP 或 SSL 服务器

查询命令

功能：

查询 TCP/SSL 服务器状态

命令：

```
AT+CIPSERVER?
```

响应：

```
+CIPSERVER:<mode>[,<port>,<"type">][,<CA enable>]

OK
```

设置命令

命令：

```
AT+CIPSERVER=<mode>[,<param2>][,<"type">][,<CA enable>]
```

响应：

```
OK
```

参数

- **<mode>**：
  - 0: 关闭服务器
  - 1: 建立服务器
- **<param2>**：参数 <mode> 不同，则此参数意义不同：
- 如果 <mode> 是 1，<param2> 代表端口号。默认值：333
- 如果 <mode> 是 0，<param2> 代表服务器是否关闭所有客户端。默认值：0
  - 0：关闭服务器并保留现有客户端连接
  - 1：关闭服务器并关闭所有连接
- **<" type" >**：服务器类型：" TCP"，" TCPv6"，" SSL"，或 "SSLv6" . 默认值：" TCP"。由于内存限制，此参数不适用于 ESP8266 设备
- **<CA enable>**：由于内存限制，此参数不适用于 ESP8266 设备
  - 0：不使用 CA 认证
  - 1：使用 CA 认证

说明

- 多连接情况下 (*AT+CIPMUX=1*)，才能开启服务器
- 创建服务器后，自动建立服务器监听，最多只允许创建一个服务器
- 当有客户端接入，会自动占用一个连接 ID
- 如果想基于 IPv6 网络建立服务器，需要先设置*AT+CIPV6=1*，再通过*AT+CWJAP* 获取到一个 IPv6 地址

示例

```
// 建立 TCP 服务器
AT+CIPMUX=1
AT+CIPSERVER=1,80

// 建立 SSL 服务器
AT+CIPMUX=1
AT+CIPSERVER=1,443,"SSL",1

// 基于 IPv6 网络，创建 SSL 服务器
AT+CIPMUX=1
AT+CIPSERVER=1,443,"SSLv6",0

// 关闭服务器并且关闭所有连接
AT+CIPSERVER=0,1
```

3.3.13 AT+CIPSERVERMAXCONN：查询/设置服务器允许建立的最大连接数

查询命令

功能：

查询 TCP 或 SSL 服务器允许建立的最大连接数

命令：

```
AT+CIPSERVERMAXCONN?
```

响应：

```
+CIPSERVERMAXCONN:<num>
OK
```

设置命令

功能：

设置 TCP 或 SSL 服务器允许建立的最大连接数

命令：

```
AT+CIPSERVERMAXCONN=<num>
```

响应：

```
OK
```

参数

- **<num>**：TCP 或 SSL 服务器允许建立的最大连接数

说明

- 如需设置最大连接数 (AT+CIPSERVERMAXCONN=<num>)，请在创建服务器之前设置。

示例

```
AT+CIPMUX=1
AT+CIPSERVERMAXCONN=2
AT+CIPSERVER=1,80
```

3.3.14 AT+CIPMODE：查询/设置传输模式

查询命令

功能：

查询传输模式

命令：

```
AT+CIPMODE?
```

响应：

```
+CIPMODE:<mode>
OK
```

设置命令

功能：

设置传输模式

命令：

```
AT+CIPMODE=<mode>
```

响应：

```
OK
```

参数

- **<mode>**:
  - 0: 普通传输模式
  - 1: Wi-Fi 透传接收模式，仅支持 TCP 单连接、UDP 固定通信对端、SSL 单连接的情况

说明

- 配置更改不保存到 flash。

示例

```
AT+CIPMODE=1
```

3.3.15 AT+SAVETRANSLINK：设置开机透传模式信息

设置开机进入 TCP/SSL 透传模式信息

设置命令

命令：

```
AT+SAVETRANSLINK=<mode>,<"remote host">,<remote port>[,<"type">,<keep alive>]
```

响应：

```
OK
```

参数

- **<mode>**:
  - 0: 关闭 ESP 上电进入 Wi-Fi 透传模式
  - 1: 开启 ESP 上电进入 Wi-Fi 透传模式
- **<" remote host" >**：字符串参数，表示远端 IPv4 地址、IPv6 地址，或域名
- **<remote port>**：远端端口值
- **<" type" >**：字符串参数，表示传输类型：" TCP"，" TCPv6"，" SSL"，或 "SSLv6"。默认值：" TCP"
- **<keep alive>**：TCP keep-alive 间隔，默认值：0
  - 0: 禁用 keep-alive 功能
  - 1 ~ 7200: 检测间隔，单位：秒

说明

- 本设置将 Wi-Fi 开机透传模式 信息保存在 NVS 区，若参数 <mode> 为 1 ，下次上电自动进入透传模式。需重启生效。
- 只要远端 IP 地址（域名）、端口的值符合规范，本设置就会被保存到 flash。
- 如果想基于 IPv6 网络建立透传连接，需要先设置*AT+CIPV6=1*，再通过*AT+CWJAP* 获取到一个 IPv6 地址

示例

```
AT+SAVETRANSLINK=1,"192.168.6.110",1002,"TCP"
AT+SAVETRANSLINK=1,"www.baidu.com",443,"SSL"
AT+SAVETRANSLINK=1,"240e:3a1:2070:11c0:55ce:4e19:9649:b75",8080,"TCPv6"
AT+SAVETRANSLINK=1,"240e:3a1:2070:11c0:55ce:4e19:9649:b75",8080,"SSLv6
```

设置开机进入 UDP 透传模式信息

设置

命令：

```
AT+SAVETRANSLINK=<mode>,<"remote host">,<remote port>,[<"type">,<local port>]
```

响应：

```
OK
```

参数

- **<mode>**:
  - 0: 关闭 ESP 上电进入 Wi-Fi 透传模式
  - 1: 开启 ESP 上电进入 Wi-Fi 透传模式
- **<" remote host" >**：字符串参数，表示远端 IPv4 地址、IPv6 地址，或域名
- **<remote port>**：远端端口值
- **<" type" >**：字符串参数，表示传输类型：" UDP" 或 "UDPv6"。默认值：" TCP"
- **[<local port>]**：开机进入 UDP 传输时，使用的本地端口

说明

- 本设置将 Wi-Fi 开机透传模式 信息保存在 NVS 区，若参数 <mode> 为 1 ，下次上电自动进入透传模式。需重启生效
- 只要远端 IP 地址（域名）、端口的值符合规范，本设置就会被保存到 flash
- 如果想基于 IPv6 网络建立透传连接，需要先设置*AT+CIPV6=1*，再通过*AT+CWJAP* 获取到一个 IPv6 地址

示例

```
AT+SAVETRANSLINK=1,"192.168.6.110",1002,"UDP",1005
AT+SAVETRANSLINK=1,"240e:3a1:2070:11c0:55ce:4e19:9649:b75",8081,"UDPv6",1005
```

3.3.16 AT+CIPSTO：查询/设置本地 TCP/SSL 服务器超时时间

查询命令

功能：

查询本地 TCP/SSL 服务器超时时间

命令：

```
AT+CIPSTO?
```

响应：

```
+CIPSTO:<time>
OK
```

设置命令

功能：

设置本地 TCP/SSL 服务器超时时间

命令：

```
AT+CIPSTO=<time>
```

响应：

```
OK
```

参数

- **<time>**：本地 TCP/SSL 服务器超时时间，单位：秒，取值范围：[0,7200]

说明

- 当 TCP/SSL 客户端在 <time> 时间内未发生数据通讯时，ESP 服务器会断开此连接。
- 如果设置参数 <time> 为 0，则连接永远不会超时，不建议这样设置。
- 在设定的时间内，当客户端发起与服务器的通信时，计时器将重新计时。超时后，客户端被关闭。在设定的时间内，如果服务器发起与客户端的通信，计时器将不会重新计时。超时后，客户端被关闭。

示例

```
AT+CIPMUX=1
AT+CIPSERVER=1,1001
AT+CIPSTO=10
```

3.3.17 AT+CIPSNTPCFG：查询/设置时区和 SNTP 服务器

查询命令

命令：

```
AT+CIPSNTPCFG?
```

响应：

```
+CIPSNTPCFG:<enable>,<timezone>,<SNTP server1>[,<SNTP server2>,<SNTP server3>]
OK
```

设置命令

命令：

```
AT+CIPSNTPCFG=<enable>,<timezone>[,<SNTP server1>,<SNTP server2>,<SNTP server3>]
```

响应：

```
OK
```

参数

- **<enable>**：设置 SNTP 服务器：
  - 1: 设置 SNTP 服务器；
  - 0: 不设置 SNTP 服务器。
- **<timezone>**：支持以下两种格式：
  - 第一种格式的范围：[-12,14]，它以小时为单位，通过与协调世界时 (UTC) 的偏移来标记大多数时区（UTC–12:00 至 UTC+14:00）；
  - 第二种格式为 UTC 偏移量，UTC 偏移量指定了你需要加多少时间到 UTC 时间上才能得到本地时间，通常显示为 [+|-][hh]mm。如果当地时区在本初子午线以西，则为负数，如果在东边，则为正数。小时 (hh) 必须在 -12 到 14 之间，分钟 (mm) 必须在 0 到 59 之间。例如，如果您想把时区设置为新西兰查塔姆群岛，即 UTC+12:45，您应该把 <timezone> 参数设置为 1245，更多信息请参考 UTC 偏移量。
- **[<SNTP server1>]**：第一个 SNTP 服务器。
- **[<SNTP server2>]**：第二个 SNTP 服务器。
- **[<SNTP server3>]**：第三个 SNTP 服务器。

说明

- 设置命令若未填写以上三个 SNTP 服务器参数，则默认使用“cn.ntp.org.cn” 、” ntp.sjtu.edu.cn” 和“us.pool.ntp.org” 其中之一。
- 对于查询命令，查询的 <timezone> 参数可能会和设置的 <timezone> 参数不一样。因为 <timezone> 参数支持第二种 UTC 偏移量格式，例如：设置 AT+CIPSNTPCFG=1,015，那么查询时，ESP-AT 会忽略时区参数的前导 0，即设置值是 15。不属于第一种格式，所以按照第二种 UTC 偏移量格式解析，也就是 UTC+00:15，也就是查询出来的是 0 时区。

示例

```
// 使能 SNTP 服务器，设置中国时区 (UTC+08:00)
AT+CIPSNTPCFG=1,8,"cn.ntp.org.cn","ntp.sjtu.edu.cn"
或
AT+CIPSNTPCFG=1,800,"cn.ntp.org.cn","ntp.sjtu.edu.cn"

// 使能 SNTP 服务器，设置美国纽约的时区 (UTC-05:00)
AT+CIPSNTPCFG=1,-5,"0.pool.ntp.org","time.google.com"
或
AT+CIPSNTPCFG=1,-500,"0.pool.ntp.org","time.google.com"

// 使能 SNTP 服务器，设置新西兰时区查塔姆群岛的时区 (Chatham Islands, UTC+12:45)
AT+CIPSNTPCFG=1,1245,"0.pool.ntp.org","time.google.com"
```

3.3.18 AT+CIPSNTPTIME：查询 SNTP 时间

查询命令

命令：

```
AT+CIPSNTPTIME?
```

响应：

```
+CIPSNTPTIME:<asctime style time>
OK
```

说明

- 有关 asctime 时间的定义请见 asctime man page。

示例

```
AT+CIPSNTPCFG=1,8,"cn.ntp.org.cn","ntp.sjtu.edu.cn"

OK

AT+CIPSNTPTIME?
+CIPSNTPTIME:Mon Dec 12 02:33:32 2016
OK
```

3.3.19 AT+CIUPDATE：通过 Wi-Fi 升级固件

ESP-AT 在运行时，通过 Wi-Fi 从指定的服务器上下载新固件到某些分区，从而升级固件。

查询命令

功能：

查询 ESP 设备的升级状态

命令：

```
AT+CIUPDATE?
```

响应：

```
+CIPUPDATE:<state>
OK
```

执行命令

功能：

通过 OTA 升级到 TCP/SSL 服务器上最新版本的固件

命令：

```
AT+CIUPDATE
```

响应：

```
+CIPUPDATE:<state>
OK
```

或

```
ERROR
```

设置命令

功能：

升级到服务器上指定版本的固件

命令：

```
AT+CIUPDATE=<ota mode>[,<version>][,<firmware name>][,<nonblocking>]
```

响应：

```
+CIPUPDATE:<state>
OK
```

或

```
ERROR
```

参数

- **<ota mode>**:
  - 0: 通过 HTTP OTA；
  - 1: 通过 HTTPS OTA，如果无效，请检查 `./build.py menuconfig > Component config > AT > OTA based upon ssl` 是否使能，更多信息请见*Build Your Own ESP-AT Project*。
- **<version>**：AT 版本，如 `v1.2.0.0`、`v1.1.3.0` 或 `v1.1.2.0`。
- **<firmware name>**：升级的固件，如 `ota`、`mqtt_ca`、`client_ca` 或其它 `at_customize.csv` 中自定义的分区。
- **<nonblocking>**:
  - 0: 阻塞模式的 OTA（此模式下，直到 OTA 升级成功或失败后才可以发送 AT 命令）；
  - 1: 非阻塞模式的 OTA（此模式下，升级完成后 (+CIPUPDATE:4) 需手动重启)。
- **<state>**:
  - 0: 空闲；

  - 1: 找到服务器；
  - 2: 连接至服务器；
  - 3: 获得升级版本；
  - 4: 完成升级；
  - -1: 升级失败。

说明

- 升级速度取决于网络状况。
- 如果网络条件不佳导致升级失败，AT 将返回 ERROR，请等待一段时间再试。
- 如果您直接使用乐鑫提供的 AT BIN, 本命令将从 Espressif Cloud 下载 AT 固件升级。
- 如果您使用的是自行编译的 AT BIN，请自行实现 AT+CIUPDATE FOTA 功能，可参考 ESP-AT 工程提供的示例 FOTA。
- 建议升级 AT 固件后，调用*AT+RESTORE* 恢复出厂设置。
- 不建议升级到旧版本。

示例

```
AT+CIUPDATE
AT+CIUPDATE=1
AT+CIUPDATE=1,"v1.2.0.0"
AT+CIUPDATE=1,"v2.2.0.0","mqtt_ca"
AT+CIUPDATE=1,"V2.2.0.0","ota",1
AT+CIUPDATE=1,,,1
AT+CIUPDATE=1,,"ota",1
AT+CIUPDATE=1,"V2.2.0.0",,1
```

3.3.20 AT+CIPDINFO：设置 +IPD 消息详情

设置命令

命令：

```
AT+CIPDINFO=<mode>
```

响应：

```
OK
```

参数

- **<mode>**:
  - 0: 在“+IPD”和“+CIPRECVDATA”消息中，不提示对端 IP 地址和端口信息
  - 1: 在“+IPD”和“+CIPRECVDATA”消息中，提示对端 IP 地址和端口信息

示例

```
AT+CIPDINFO=1
```

3.3.21 AT+CIPSSLCCONF：查询/设置 SSL 客户端配置

查询命令

功能：

查询 ESP 作为 SSL 客户端时每个连接的配置信息

命令：

```
AT+CIPSSLCCONF?
```

响应：

```
+CIPSSLCCONF:<link ID>,<auth_mode>,<pki_number>,<ca_number>
OK
```

设置命令

命令：

```
// 单连接: (AT+CIPMUX=0)
AT+CIPSSLCCONF=<auth_mode>[,<pki_number>][,<ca_number>]

// 多连接: (AT+CIPMUX=1)
AT+CIPSSLCCONF=<link ID>,<auth_mode>[,<pki_number>][,<ca_number>]
```

响应：

```
OK
```

参数

- **<link ID>**：网络连接 ID (0 ~ max)，在多连接的情况下，若参数值设为 max，则表示所有连接，本参数默认值为 5。
- **<auth_mode>**:
  - 0: 不认证，此时无需填写 <pki_number> 和 <ca_number> 参数；
  - 1: ESP-AT 提供客户端证书供服务器端 CA 证书校验；
  - 2: ESP-AT 客户端载入 CA 证书来校验服务器端的证书；
  - 3: 相互认证。
- **<pki_number>**：证书和私钥的索引，如果只有一个证书和私钥，其值应为 0。
- **<ca_number>**：CA 的索引，如果只有一个 CA，其值应为 0。

说明

- 如果想要本配置立即生效，请在建立 SSL 连接前运行本命令。
- 配置更改将保存在 NVS 区，如果您使用*AT+SAVETRANSLINK* 命令设置开机进入 Wi-Fi SSL 透传模式，ESP 将在下次上电时基于本配置建立 SSL 连接。

3.3.22 AT+CIPSSLCCN：查询/设置 SSL 客户端的公用名 (common name)

查询命令

功能：

查询每个 SSL 连接中客户端的通用名称

命令：

```
AT+CIPSSLCCN?
```

响应：

```
+CIPSSLCCN:<link ID>,<"common name">
OK
```

设置命令

命令：

```
// 单连接：(AT+CIPMUX=0)
AT+CIPSSLCCN=<"common name">

// 多连接：(AT+CIPMUX=1)
AT+CIPSSLCCN=<link ID>,<"common name">
```

响应：

```
OK
```

参数

- **<link ID>**：网络连接 ID (0 ~ max)，在单连接的情况下，本参数值为 0；在多连接的情况下，若参数值设为 max，则表示所有连接；本参数默认值为 5。
- **<" common name" >**：本参数用来认证服务器发送的证书中的公用名。公用名最大长度为 64 字节。

说明

- 如果想要本配置立即生效，请在建立 SSL 连接前运行本命令。

3.3.23 AT+CIPSSLCSNI：查询/设置 SSL 客户端的 SNI

查询命令

功能：

查询每个连接的 SNI 配置

命令：

```
AT+CIPSSLCSNI?
```

响应：

```
+CIPSSLCSNI:<link ID>,<"sni">
OK
```

设置命令

命令：

```
单连接：(AT+CIPMUX=0)
AT+CIPSSLCSNI=<"sni">
多连接：(AT+CIPMUX=1)
AT+CIPSSLCSNI=<link ID>,<"sni">
```

响应：

```
OK
```

参数

- **<link ID>**：网络连接 ID (0 ~ max)，在单连接的情况下，本参数值为 0；在多连接的情况下，若参数值设为 max，则表示所有连接；本参数默认值为 5。
- **<" sni" >**：ClientHello 里的 SNI。SNI 最大长度为 64 字节。

说明

- 如果想要本配置立即生效，请在建立 SSL 连接前运行本命令。

3.3.24 AT+CIPSSLCALPN：查询/设置 SSL 客户端 ALPN

查询命令

功能：

查询 ESP 作为 SSL 客户端时每个连接的 ALPN 配置

命令：

```
AT+CIPSSLCALPN?
```

响应：

```
+CIPSSLCALPN:<link ID>,<"alpn">[,<"alpn">][,<"alpn">]

OK
```

设置命令

命令：

```
// 单连接：(AT+CIPMUX=0)
AT+CIPSSLCALPN=<counts>[,<"alpn">][,<"alpn">][,<"alpn">]

// 多连接：(AT+CIPMUX=1)
AT+CIPSSLCALPN=<link ID>,<counts>[,<"alpn">][,<"alpn">[,<"alpn">]
```

响应：

```
OK
```

参数

- **<link ID>**：网络连接 ID (0 ~ max)，在单连接的情况下，本参数值为 0；在多连接的情况下，若参数值设为 max，则表示所有连接；本参数默认值为 5。
- **<counts>**：ALPN 的数量。范围：[0,5]。
- 0: 清除 ALPN 配置。
- [1,5]: 设置 ALPN 配置。
- **<" alpn" >**：字符串参数，表示 ClientHello 中的 ALPN。ALPN 最大长度受限于命令的最大长度。

说明

- 如果想要本配置立即生效，请在建立 SSL 连接前运行本命令。

3.3.25 AT+CIPSSLCPSK：查询/设置 SSL 客户端的 PSK

查询命令

功能：

查询 ESP 作为 SSL 客户端时每个连接的 PSK 配置

命令：

```
AT+CIPSSLCPSK?
```

响应：

```
+CIPSSLCPSK:<link ID>,<"psk">,<"hint">
OK
```

设置命令

命令：

```
// 单连接：(AT+CIPMUX=0)
AT+CIPSSLCPSK=<"psk">,<"hint">

// 多连接：(AT+CIPMUX=1)
AT+CIPSSLCPSK=<link ID>,<"psk">,<"hint">
```

响应：

```
OK
```

参数

- **<link ID>**：网络连接 ID (0 ~ max)，在单连接的情况下，本参数值为 0；在多连接的情况下，若参数值设为 max，则表示所有连接；本参数默认值为 5。
- **<" psk" >**：PSK identity，最大长度：32。
- **<" hint" >**：PSK hint，最大长度：32。

说明

- 如果想要本配置立即生效，请在建立 SSL 连接前运行本命令。

3.3.26 AT+CIPRECONNINTV：查询/设置 Wi-Fi 透传模式下的 TCP/UDP/SSL 重连间隔

查询命令

功能：

查询 Wi-Fi 透传模式 下的自动重连间隔

命令：

```
AT+CIPRECONNINTV?
```

响应：

```
+CIPRECONNINTV:<interval>
OK
```

设置命令

功能：

设置 Wi-Fi 透传模式 下 TCP/UDP/SSL 传输断开后自动重连的间隔

命令：

```
AT+CIPRECONNINTV=<interval>
```

响应：

```
OK
```

参数

- **<interval>**：自动重连间隔时间，单位：100 毫秒，默认值：1，范围：[1,36000]。

说明

- 若*AT+SYSSTORE=1* 时，配置更改将保存在 NVS 区。

示例

```
AT+CIPRECONNINTV=10
```

3.3.27 AT+CIPRECVMODE：查询/设置 socket 接收模式

查询命令

功能：

查询 socket 接收模式

命令：

```
AT+CIPRECVMODE?
```

响应：

```
+CIPRECVMODE:<mode>
OK
```

设置命令

命令：

```
AT+CIPRECVMODE=<mode>
```

响应：

```
OK
```

参数

- **<mode>**：socket 数据接收模式，默认值：0。
  - 0: 主动模式，ESP-AT 将所有接收到的 socket 数据立即发送给主机 MCU，头为“+IPD”。
  - 1: 被动模式，ESP-AT 将所有接收到的 socket 数据保存到内部缓存区 (socket 接收窗口，ESP8266 设备默认为 5760 字节，非 ESP8266 设备默认值为 5744 字节)，等待 MCU 读取。对于 TCP 和 SSL 连接，如果缓存区满了，将阻止 socket 传输；对于 UDP 传输，如果缓存区满了，则会发生数据丢失。

说明

- 该配置不能用于 Wi-Fi 透传模式。
- 当 ESP-AT 在被动模式下收到 socket 数据时，会根据情况的不同提示不同的信息：
  - 多连接时 (AT+CIPMUX=1)，提示 `+IPD,<link ID>,<len>`；
  - 单连接时 (AT+CIPMUX=0)，提示 `+IPD,<len>`。
- `<len>` 表示缓存区中 socket 数据的总长度。
- 一旦有 `+IPD` 报出，应该运行*AT+CIPRECVDATA* 来读取数据。否则，在前一个 `+IPD` 被读取之前，下一个 `+IPD` 将不会被报告给主机 MCU。
- 在断开连接的情况下，缓冲的 socket 数据仍然存在，MCU 仍然可以读取，直到发送*AT+CIPCLOSE*。换句话说，如果 `+IPD` 已经被报告，那么在你发送*AT+CIPCLOSE* 或通过*AT+CIPRECVDATA* 命令读取所有数据之前，这个连接的 `CLOSED` 信息永远不会出现。

示例

```
AT+CIPRECVMODE=1
```

3.3.28 AT+CIPRECVDATA：获取被动接收模式下的 socket 数据

设置命令

命令：

```
// 单连接：(AT+CIPMUX=0)
AT+CIPRECVDATA=<len>

// 多连接：(AT+CIPMUX=1)
AT+CIPRECVDATA=<link_id>,<len>
```

响应：

```
+CIPRECVDATA:<actual_len>,<data>
OK
```

或

```
+CIPRECVDATA:<actual_len>,<remote IP>,<remote port>,<data>
OK
```

参数

- **<link_id>**：多连接模式下的连接 ID。
- **<len>**：最大值为：0x7fffffff，如果实际收到的数据长度比本参数值小，则返回实际长度的数据。
- **<actual_len>**：实际获取的数据长度。
- **<data>**：获取的数据。
- **[<remote IP>]**：字符串参数，表示对端 IP 地址，通过*AT+CIPDINFO=1* 命令使能。
- **[<remote port>]**：对端端口，通过*AT+CIPDINFO=1* 命令使能。

示例

```
AT+CIPRECVMODE=1

// 例如，如果主机 MCU 从 0 号连接中收到 100 字节的数据，
// 则会提示消息 "+IPD,0,100"，
// 然后，您可以通过运行以下命令读取这 100 字节的数据：
AT+CIPRECVDATA=0,100
```

3.3.29 AT+CIPRECVLEN：查询被动接收模式下 socket 数据的长度

查询命令

功能：

查询某一连接中缓存的所有的数据长度

命令：

```
AT+CIPRECVLEN?
```

响应：

```
+CIPRECVLEN:<data length of link0>,<data length of link1>,<data length of link2>,
↪<data length of link3>,<data length of link4>
OK
```

参数

- **<data length of link>**：某一连接中缓冲的所有的数据长度。

说明

- SSL 连接中，ESP-AT 将返回加密数据的长度，所以返回的长度会大于真实数据的长度。

示例

```
AT+CIPRECVLEN?
+CIPRECVLEN:100,,,,,
OK
```

3.3.30 AT+PING：ping 对端主机

设置命令

功能：

ping 对端主机

命令：

```
AT+PING=<"host">
```

响应：

```
+PING:<time>

OK
```

或

```
+PING:TIMEOUT   // 只有在域名解析失败或 PING 超时情况下，才会有这个回复

ERROR
```

参数

- **<"host">**：字符串参数，表示对端主机的 IPv4 地址，IPv6 地址，或域名。
- **<time>**：ping 的响应时间，单位：毫秒。

说明

- 如果想基于 IPv6 网络 ping 对端主机，需要先设置*AT+CIPV6=1*，再通过*AT+CWJAP* 获取到一个 IPv6 地址
- 如果远端主机是域名字符串，则 ping 将先通过 DNS 进行域名解析（优先解析 IPv4 地址），再 ping 对端主机 IP 地址

示例

```
AT+PING="192.168.1.1"
AT+PING="www.baidu.com"

// 下一代互联网国家工程中心
AT+PING="240c::6666"
```

3.3.31 AT+CIPDNS：查询/设置 DNS 服务器信息

查询命令

功能：

查询当前 DNS 服务器信息

命令：

```
AT+CIPDNS?
```

响应：

```
+CIPDNS:<enable>[,<"DNS IP1">,<"DNS IP2">,<"DNS IP3">]
OK
```

设置命令

功能：

设置 DNS 服务器信息

命令：

```
AT+CIPDNS=<enable>[,<"DNS IP1">,<"DNS IP2">,<"DNS IP3">]
```

响应：

```
OK
```

或

```
ERROR
```

参数

- **<enable>**：设置 DNS
  - 0: 启用自动获取 DNS 设置，DNS 将会恢复为 `208.67.222.222`，只有当 DHCP 更新时才会生效；
  - 1: 启用手动设置 DNS 信息，如果不设置参数 `<DNS IPx>` 的值，则使用默认值 `208.67.222.222`。
- **<DNS IP1>**：第一个 DNS IP 地址，对于设置命令，只有当 <enable> 参数为 1 时，也就是启用手动 DNS 设置，本参数才有效；如果设置 <enable> 为 1，并为本参数设置一个值，当您运行查询命令时，ESP-AT 将把该参数作为当前的 DNS 设置返回。
- **<DNS IP2>**：第二个 DNS IP 地址，对于设置命令，只有当 <enable> 参数为 1 时，也就是启用手动 DNS 设置，本参数才有效；如果设置 <enable> 为 1，并为本参数设置一个值，当您运行查询命令时，ESP-AT 将把该参数作为当前的 DNS 设置返回。
- **<DNS IP3>**：第三个 DNS IP 地址，对于设置命令，只有当 <enable> 参数为 1 时，也就是启用手动 DNS 设置，本参数才有效；如果设置 <enable> 为 1，并为本参数设置一个值，当您运行查询命令时，ESP-AT 将把该参数作为当前的 DNS 设置返回。

说明

- 若*AT+SYSSTORE=1*，配置更改将保存在 NVS 区。
- 这三个参数不能设置在同一个服务器上。
- 当 `<enable>` 为 0 时，DNS 服务器可能会根据 ESP 设备所连接的路由器的配置而改变。

示例

```
AT+CIPDNS=0
AT+CIPDNS=1,"208.67.222.222","114.114.114.114","8.8.8.8"

// 第一个基于 IPv6 的 DNS 服务器：下一代互联网国家工程中心
// 第二个基于 IPv6 的 DNS 服务器：google-public-dns-a.google.com
// 第三个基于 IPv6 的 DNS 服务器：江苏省主 DNS 服务器
AT+CIPDNS=1,"240c::6666","2001:4860:4860::8888","240e:5a::6666"
```

3.3.32 AT+CIPTCPOPT：查询/设置 socket 选项

查询命令

功能：

查询当前 socket 选项

命令：

```
AT+CIPTCPOPT?
```

响应：

```
+CIPTCPOPT:<link_id>,<so_linger>,<tcp_nodelay>,<so_sndtimeo>
OK
```

设置命令

命令：

```
// 单连接：(AT+CIPMUX=0)：
AT+CIPTCPOPT=[<so_linger>],[<tcp_nodelay>],[<so_sndtimeo>]

// 多连接：(AT+CIPMUX=1)：
AT+CIPTCPOPT=<link ID>,[<so_linger>],[<tcp_nodelay>],[<so_sndtimeo>]
```

响应：

```
OK
```

或

```
ERROR
```

参数

- **<link_id>**：网络连接 ID (0 ~ max)，在多连接的情况下，若参数值设为 max，则表示所有连接；本参数默认值为 5。
- **<so_linger>**：配置 socket 的 SO_LINGER 选项，单位：秒，默认值：-1。
  - = -1: 关闭；
  - = 0: 开启，linger time = 0；
  - > 0: 开启，linger time = <so_linger>；

- **<tcp_nodelay>**：配置 socket 的 TCP_NODELAY 选项，默认值：0。
  - 0: 禁用 TCP_NODELAY
  - 1: 启用 TCP_NODELAY
- **<so_sndtimeo>**：配置 socket 的 SO_SNDTIMEO 选项，单位：毫秒，默认值：0。

3.4 [ESP32 Only] Bluetooth® Low Energy AT 命令集

[English]

ESP32 AT 固件支持 蓝牙核心规范 5.0 版本。

- [ESP32 Only] *AT+BLEINIT*：Bluetooth LE 初始化
- [ESP32 Only] *AT+BLEADDR*：设置 Bluetooth LE 设备地址
- [ESP32 Only] *AT+BLENAME*：查询/设置 Bluetooth LE 设备名称
- [ESP32 Only] *AT+BLESCANPARAM*：查询/设置 Bluetooth LE 扫描参数
- [ESP32 Only] *AT+BLESCAN*：使能 Bluetooth LE 扫描
- [ESP32 Only] *AT+BLESCANRSPDATA*：设置 Bluetooth LE 扫描响应
- [ESP32 Only] *AT+BLEADVPARAM*：查询/设置 Bluetooth LE 广播参数
- [ESP32 Only] *AT+BLEADVDATA*：设置 Bluetooth LE 广播数据
- [ESP32 Only] *AT+BLEADVDATAEX*：自动设置 Bluetooth LE 广播数据
- [ESP32 Only] *AT+BLEADVSTART*：开始 Bluetooth LE 广播
- [ESP32 Only] *AT+BLEADVSTOP*：停止 Bluetooth LE 广播
- [ESP32 Only] *AT+BLECONN*：建立 Bluetooth LE 连接
- [ESP32 Only] *AT+BLECONNPARAM*：查询/更新 Bluetooth LE 连接参数
- [ESP32 Only] *AT+BLEDISCONN*：断开 Bluetooth LE 连接
- [ESP32 Only] *AT+BLEDATALEN*：设置 Bluetooth LE 数据包长度
- [ESP32 Only] *AT+BLECFGMTU*：设置 Bluetooth LE MTU 长度
- [ESP32 Only] *AT+BLEGATTSSRVCRE*：GATTS 创建服务
- [ESP32 Only] *AT+BLEGATTSSRVSTART*：GATTS 开启服务
- [ESP32 Only] *AT+BLEGATTSSRVSTOP*：GATTS 停止服务
- [ESP32 Only] *AT+BLEGATTSSRV*：GATTS 发现服务
- [ESP32 Only] *AT+BLEGATTSCHAR*：GATTS 发现服务特征
- [ESP32 Only] *AT+BLEGATTSNTFY*：服务器 notify 服务特征值给客户端

- [ESP32 Only] *AT+BLEGATTSIND*：服务器 indicate 服务特征值给客户端
- [ESP32 Only] *AT+BLEGATTSSETATTR*：GATTS 设置服务特征值
- [ESP32 Only] *AT+BLEGATTCPRIMSRV*：GATTC 发现基本服务
- [ESP32 Only] *AT+BLEGATTCINCLSRV*：GATTC 发现包含的服务
- [ESP32 Only] *AT+BLEGATTCCHAR*：GATTC 发现服务特征
- [ESP32 Only] *AT+BLEGATTCRD*：GATTC 读取服务特征值
- [ESP32 Only] *AT+BLEGATTCWR*：GATTC 写服务特征值
- [ESP32 Only] *AT+BLESPPCFG*：查询/设置 Bluetooth LE SPP 参数
- [ESP32 Only] *AT+BLESPP*：进入 Bluetooth LE SPP 模式
- [ESP32 Only] *AT+BLESECPARAM*：查询/设置 Bluetooth LE 加密参数
- [ESP32 Only] *AT+BLEENC*：发起 Bluetooth LE 加密请求
- [ESP32 Only] *AT+BLEENCRSP*：回复对端设备发起的配对请求
- [ESP32 Only] *AT+BLEKEYREPLY*：给对方设备回复密钥（传统连接阶段）
- [ESP32 Only] *AT+BLECONFREPLY*：给对方设备回复确认结果（传统连接阶段）
- [ESP32 Only] *AT+BLEENCDEV*：查询绑定的 Bluetooth LE 加密设备列表
- [ESP32 Only] *AT+BLEENCCLEAR*：清除 Bluetooth LE 加密设备列表
- [ESP32 Only] *AT+BLESETKEY*：设置 Bluetooth LE 静态配对密钥
- [ESP32 Only] *AT+BLEHIDINIT*：Bluetooth LE HID 协议初始化
- [ESP32 Only] *AT+BLEHIDKB*：发送 Bluetooth LE HID 键盘信息
- [ESP32 Only] *AT+BLEHIDMUS*：发送 Bluetooth LE HID 鼠标信息
- [ESP32 Only] *AT+BLEHIDCONSUMER*：发送 Bluetooth LE HID consumer 信息
- [ESP32 Only] *AT+BLUFI*：开启或关闭 BluFi
- [ESP32 Only] *AT+BLUFINAME*：查询/设置 BluFi 设备名称

3.4.1 [ESP32 Only] AT+BLEINIT：Bluetooth LE 初始化

查询命令

功能：

查询 Bluetooth LE 是否初始化

命令：

```
AT+BLEINIT?
```

响应：

若已初始化，AT 返回：

```
+BLEINIT:<role>
OK
```

若未初始化，AT 返回：

```
+BLEINIT:0
OK
```

设置命令

功能：

设置 Bluetooth LE 初始化角色

命令：

```
AT+BLEINIT=<init>
```

响应：

```
OK
```

参数

- **<init>**:
  - 0: 注销 Bluetooth LE
  - 1: client 角色
  - 2: server 角色

说明

- 使用相关命令之前，请先下载“at_customize.bin”文件，详情请见*如何自定义 Ble services*。
- 使用其它 Bluetooth LE 命令之前，请先调用本命令，初始化 Bluetooth LE 角色。
- Bluetooth LE 角色初始化后，不能直接切换。如需切换角色，需要先调用*AT+RST* 命令重启系统，再重新初始化 Bluetooth LE 角色。
- 若使用 ESP 作为 Bluetooth LE server，需烧录 service bin 到 flash：
  - 对于如何生成 service bin 文件，请参考 esp-at/tools/readme.md；
  - service bin 文件的烧录地址，请见 esp-at/module_config/module_${platform}_default/at_customize.csv 文件中“ble_data”对应的地址。

示例

```
AT+BLEINIT=1
```

3.4.2 [ESP32 Only] AT+BLEADDR：设置 Bluetooth LE 设备地址

查询命令

功能：

查询 Bluetooth LE 设备的公共地址

命令：

```
AT+BLEADDR?
```

响应：

```
+BLEADDR:<BLE_public_addr>
OK
```

设置命令

功能：

设置 Bluetooth LE 设备的地址类型

命令：

```
AT+BLEADDR=<addr_type>[,<random_addr>]
```

响应：

```
OK
```

参数

- **<addr_type>**:
  - 0: 公共地址 (Public Address)
  - 1: 随机地址 (Random Address)

说明

- 静态地址 (Static Address) 应满足以下条件：
  - 地址最高两位应为 1；
  - 随机地址部分至少有 1 位为 0；
  - 随机地址部分至少有 1 位为 1。

示例

```
AT+BLEADDR=1,"f8:7f:24:87:1c:7b"     // 设置随机设备地址的静态地址
AT+BLEADDR=1                         // 设置随机设备地址的私有地址
AT+BLEADDR=0                         // 设置公共设备地址
```

3.4.3 [ESP32 Only] AT+BLENAME：查询/设置 Bluetooth LE 设备名称

查询命令

功能：

查询 Bluetooth LE 设备名称

命令：

```
AT+BLENAME?
```

响应：

```
+BLENAME:<device_name>
OK
```

设置命令

功能：

设置 Bluetooth LE 设备名称

命令：

```
AT+BLENAME=<device_name>
```

响应：

```
OK
```

参数

- **<device_name>**：Bluetooth LE 设备名称，最大长度：32，默认名称为 “BLE_AT”。

说明

- 若*AT+SYSSTORE=1*，配置更改将保存在 NVS 区。

示例

```
AT+BLENAME="esp_demo"
```

3.4.4 [ESP32 Only] AT+BLESCANPARAM：查询/设置 Bluetooth LE 扫描参数

查询命令

功能：

查询 Bluetooth LE 扫描参数

命令：

```
AT+BLESCANPARAM?
```

响应：

```
+BLESCANPARAM:<scan_type>,<own_addr_type>,<filter_policy>,<scan_interval>,<scan_
↪window>
OK
```

设置命令

功能：

设置 Bluetooth LE 扫描参数

命令：

```
AT+BLESCANPARAM=<scan_type>,<own_addr_type>,<filter_policy>,<scan_interval>,<scan_
↪window>
```

响应：

```
OK
```

参数

- **<scan_type>**：扫描类型
  - 0: 被动扫描
  - 1: 主动扫描
- **<own_addr_type>**：地址类型
  - 0: 公共地址
  - 1: 随机地址
  - 2: RPA 公共地址
  - 3: RPA 随机地址
- **<filter_policy>**：扫描过滤方式
  - 0: BLE_SCAN_FILTER_ALLOW_ALL
  - 1: BLE_SCAN_FILTER_ALLOW_ONLY_WLST
  - 2: BLE_SCAN_FILTER_ALLOW_UND_RPA_DIR
  - 3: BLE_SCAN_FILTER_ALLOW_WLIST_PRA_DIR
- **<scan_interval>**：扫描间隔
- **<scan_window>**：扫描窗口

说明

- <scan_window> 参数的值不能大于 <scan_interval> 的值。

示例

```
AT+BLEINIT=1    // 角色：客户端
AT+BLESCANPARAM=0,0,0,100,50
```

3.4.5 [ESP32 Only] AT+BLESCAN：使能 Bluetooth LE 扫描

设置命令

功能：

开始/停止 Bluetooth LE 扫描

命令：

```
AT+BLESCAN=<enable>[,<interval>][,<filter_type>,<filter_param>]
```

响应：

```
+BLESCAN:<addr>,<rssi>,<adv_data>,<scan_rsp_data>,<addr_type>
OK
```

参数

- **<enable>**：
  - 1: 开始持续扫描
  - 0: 停止持续扫描
- **[<interval>]**：扫描持续时间，单位：秒。
  - 若设置停止扫描，无需设置本参数；
  - 若设置开始扫描，需设置本参数：
  - 本参数设为 0 时，则表示开始持续扫描；
  - 本参数设为非 0 值时，例如 AT+BLESCAN=1,3，则表示扫描 3 秒后自动结束扫描，然后返回扫描结果。
- **[<filter_type>]**：过滤选项
  - 1: “MAC”

  - 2: “NAME”

- **<filter_param>**：过滤参数，表示对方设备 MAC 地址或名称
- **<addr>**：Bluetooth LE 地址
- **<rssi>**：信号强度
- **<adv_data>**：广播数据
- **<scan_rsp_data>**：扫描响应数据
- **<addr_type>**：广播设备地址类型

说明

- 响应中的 OK 和 +BLESCAN:<addr>,<rssi>,<adv_data>,<scan_rsp_data>,<addr_type> 在输出顺序上没有严格意义上的先后顺序。OK 可能在 +BLESCAN:<addr>,<rssi>,<adv_data>,<scan_rsp_data>,<addr_type> 之前输出，也有可能在 +BLESCAN:<addr>,<rssi>,<adv_data>,<scan_rsp_data>,<addr_type> 之后输出。

示例

```
AT+BLEINIT=1     // 角色：客户端
AT+BLESCAN=1     // 开始扫描
AT+BLESCAN=0     // 停止扫描
AT+BLESCAN=1,3,1,"24:0A:C4:96:E6:88"  // 开始扫描，过滤类型为 MAC 地址
AT+BLESCAN=1,3,2,"ESP-AT"  // 开始扫描，过滤类型为设备名称
```

3.4.6 [ESP32 Only] AT+BLESCANRSPDATA：设置 Bluetooth LE 扫描响应

设置命令

功能：

设置 Bluetooth LE 扫描响应

命令：

```
AT+BLESCANRSPDATA=<scan_rsp_data>
```

响应：

```
OK
```

参数

- **<scan_rsp_data>**：扫描响应数据，为 HEX 字符串。例如，若想设置扫描响应数据为“0x11 0x22 0x33 0x44 0x55”，则命令为 `AT+BLESCANRSPDATA="1122334455"`。

示例

```
AT+BLEINIT=2    // 角色：服务器
AT+BLESCANRSPDATA="1122334455"
```

3.4.7 [ESP32 Only] AT+BLEADVPARAM：查询/设置 Bluetooth LE 广播参数

查询命令

功能：

查询广播参数

命令：

```
AT+BLEADVPARAM?
```

响应：

```
+BLEADVPARAM:<adv_int_min>,<adv_int_max>,<adv_type>,<own_addr_type>,<channel_map>,
↪<adv_filter_policy>,<peer_addr_type>,<peer_addr>
OK
```

设置命令

功能：

设置广播参数

命令：

```
AT+BLEADVPARAM=<adv_int_min>,<adv_int_max>,<adv_type>,<own_addr_type>,<channel_map>[,
↪<adv_filter_policy>][,<peer_addr_type>][,<peer_addr>]
```

响应：

```
OK
```

参数

- **<adv_int_min>**：最小广播间隔，本参数值应小于 <adv_int_max> 参数值。参数范围：0x0020 ~ 0x4000。
- **<adv_int_max>**：最大广播间隔，本参数值应大于 <adv_int_min> 参数值。参数范围：0x0020 ~ 0x4000。
- **<adv_type>**：
  - 0: ADV_TYPE_IND
  - 2: ADV_TYPE_SCAN_IND
  - 3: ADV_TYPE_NONCONN_IND
- **<own_addr_type>**：Bluetooth LE 地址类型
  - 0: BLE_ADDR_TYPE_PUBLIC
  - 1: BLE_ADDR_TYPE_RANDOM
- **<channel_map>**：广播信道
  - 1: ADV_CHNL_37
  - 2: ADV_CHNL_38
  - 4: ADV_CHNL_39
  - 7: ADV_CHNL_ALL
- **[<adv_filter_policy>]**：广播过滤器规则
  - 0: ADV_FILTER_ALLOW_SCAN_ANY_CON_ANY
  - 1: ADV_FILTER_ALLOW_SCAN_WLST_CON_ANY
  - 2: ADV_FILTER_ALLOW_SCAN_ANY_CON_WLST
  - 3: ADV_FILTER_ALLOW_SCAN_WLST_CON_WLST
- **[<peer_addr_type>]**：对方 Bluetooth LE 地址类型
  - 0: PUBLIC
  - 1: RANDOM
- **[<peer_addr>]**：对方 Bluetooth LE 地址

示例

```
AT+BLEINIT=2    // 角色: 服务器
AT+BLEADVPARAM=50,50,0,0,4,0,0,"12:34:45:78:66:88"
```

3.4.8 [ESP32 Only] AT+BLEADVDATA：设置 Bluetooth LE 广播数据

设置命令

功能：

设置广播数据

命令：

```
AT+BLEADVDATA=<adv_data>
```

响应：

```
OK
```

参数

- **<adv_data>**：广播数据，为 HEX 字符串。例如，若想设置广播数据为 “0x11 0x22 0x33 0x44 0x55”，则命令为 `AT+BLEADVDATA="1122334455"`。

说明

- 如果之前已经使用命令*AT+BLEADVDATAEX*=<dev_name>,<uuid>,<manufacturer_data>,<include_power>设置了广播数据，则会被本命令设置的广播数据覆盖。

示例

```
AT+BLEINIT=2    // 角色: 服务器
AT+BLEADVDATA="1122334455"
```

3.4.9 [ESP32 Only] AT+BLEADVDATAEX：自动设置 Bluetooth LE 广播数据

查询命令

功能：

查询广播数据的参数

命令：

```
AT+BLEADVDATAEX?
```

响应：

```
+BLEADVDATAEX:<dev_name>,<uuid>,<manufacturer_data>,<include_power>

OK
```

设置命令

功能：

设置广播数据并开始广播

命令：

```
AT+BLEADVDATAEX=<dev_name>,<uuid>,<manufacturer_data>,<include_power>
```

响应：

```
OK
```

参数

- **<dev_name>**：字符串参数，表示设备名称。例如，若想设置设备名称为“just-test”，则命令为 `AT+BLEADVSTARTEX="just-test",<uuid>,<manufacturer_data>,<include_power>`。
- **<uuid>**：字符串参数。例如，若想设置 UUID 为“0xA002”，则命令为 `AT+BLEADVSTARTEX=<dev_name>,"A002",<manufacturer_data>,<include_power>`。
- **<manufacturer_data>**：制造商数据，为 HEX 字符串。例如，若想设置制造商数据为“0x11 0x22 0x33 0x44 0x55”，则命令为 `AT+BLEADVSTARTEX=<dev_name>,<uuid>,"1122334455",<include_power>`。
- **<include_power>**：若广播数据需包含 TX 功率，本参数应该设为 `1`；否则，为 `0`。

说明

- 如果之前已经使用命令*AT+BLEADVDATA*=<adv_data> 设置了广播数据，则会被本命令设置的广播数据覆盖。

示例

```
AT+BLEINIT=2    // 角色：服务器
AT+BLEADVDATAEX="ESP-AT","A002","0102030405",1
```

3.4.10 [ESP32 Only] AT+BLEADVSTART：开始 Bluetooth LE 广播

执行命令

功能：

开始广播

命令：

```
AT+BLEADVSTART
```

响应：

```
OK
```

说明

- 若未使用命令*AT+BLEADVPARAM*=<adv_parameter> 设置广播参数，则使用默认广播参数。
- 若未使用命令*AT+BLEADVDATA*=<adv_data> 设置广播数据，则发送全 0 数据包。
- 若之前已经使用命令*AT+BLEADVDATA*=<adv_data> 设置过广播数据，则会被*AT+BLEADVDATAEX*=<dev_name>,<uuid>,<manufacturer_data>,<include_power> 设置的广播数据覆盖，相反，如果先使用 AT+BLEADVDATAEX，则会被 AT+BLEADVDATA 设置的广播数据覆盖。

示例

```
AT+BLEINIT=2    // 角色: 服务器
AT+BLEADVSTART
```

3.4.11 [ESP32 Only] AT+BLEADVSTOP：停止 Bluetooth LE 广播

执行命令

功能：

停止广播

命令：

```
AT+BLEADVSTOP
```

响应：

```
OK
```

说明

- 若开始广播后，成功建立 Bluetooth LE 连接，则会自动结束 Bluetooth LE 广播，无需调用本命令。

示例

```
AT+BLEINIT=2    // 角色: 服务器
AT+BLEADVSTART
AT+BLEADVSTOP
```

3.4.12 [ESP32 Only] AT+BLECONN：建立 Bluetooth LE 连接

查询命令

功能：

查询 Bluetooth LE 连接

命令：

```
AT+BLECONN?
```

响应：

```
+BLECONN:<conn_index>,<remote_address>
OK
```

若未建立连接，则响应不显示 <conn_index> 和 <remote_address> 参数。

设置命令

功能：

建立 Bluetooth LE 连接

命令：

```
AT+BLECONN=<conn_index>,<remote_address>[,<addr_type>,<timeout>]
```

响应：

```
OK
```

若建立连接成功，则提示：

```
+BLECONN:<conn_index>,<remote_address>
```

若失败，则提示：

```
+BLECONN:<conn_index>,-1
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<remote_address>**：对方 Bluetooth LE 设备地址。
- **[<addr_type>]**：广播设备地址类型。
- **[<timeout>]**：连接超时时间，单位：秒。范围：[3,30]。

说明

- 建议在建立新连接之前，先运行*AT+BLESCAN* 命令扫描设备，确保目标设备处于广播状态。
- 最大连接超时为 30 秒。
- 如果 Bluetooth LE server 已初始化且连接已成功建立，则可以使用此命令在对等设备 (GATTC) 中发现服务。还可以使用以下 GATTC 命令：
  - *AT+BLEGATTCPRIMSRV*
  - *AT+BLEGATTCINCLSRV*
  - *AT+BLEGATTCCHAR*
  - *AT+BLEGATTCRD*
  - *AT+BLEGATTCWR*
  - *AT+BLEGATTSIND*

示例

```
AT+BLEINIT=1    // 角色：客户端
AT+BLECONN=0,"24:0a:c4:09:34:23",0,10
```

3.4.13 [ESP32 Only] AT+BLECONNPARAM：查询/更新 Bluetooth LE 连接参数

查询命令

功能：

查询 Bluetooth LE 连接参数

命令：

```
AT+BLECONNPARAM?
```

响应：

```
+BLECONNPARAM:<conn_index>,<min_interval>,<max_interval>,<cur_interval>,<latency>,
↪<timeout>
OK
```

设置命令

功能：

更新 Bluetooth LE 连接参数

命令：

```
AT+BLECONNPARAM=<conn_index>,<min_interval>,<max_interval>,<latency>,<timeout>
```

响应：

```
OK
```

若设置失败，则提示以下信息：

```
+BLECONNPARAM: <conn_index>,-1
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<min_interval>**：最小连接间隔，范围：0x0006 ~ 0x0C80。
- **<max_interval>**：最大连接间隔，范围：0x0006 ~ 0x0C80。
- **<cur_interval>**：当前连接间隔。
- **<latency>**：延迟，范围：0x0000 ~ 0x01F3。
- **<timeout>**：超时，范围：0x000A ~ 0x0C80。

说明

- 本命令要求先建立连接，并且仅支持 client 角色更新连接参数。

示例

```
AT+BLEINIT=1    // 角色：客户端
AT+BLECONN=0,"24:0a:c4:09:34:23"
AT+BLECONNPARAM=0,12,14,1,500
```

3.4.14 [ESP32 Only] AT+BLEDISCONN：断开 Bluetooth LE 连接

执行命令

功能：

断开 Bluetooth LE 连接

命令：

```
AT+BLEDISCONN=<conn_index>
```

响应：

```
OK  // 收到 AT+BLEDISCONN 命令
+BLEDISCONN:<conn_index>,<remote_address>  // 运行命令成功
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<remote_address>**：对方 Bluetooth LE 设备地址。

说明

- 仅支持客户端运行本命令断开连接。

示例

```
AT+BLEINIT=1   // 角色：客户端
AT+BLECONN=0,"24:0a:c4:09:34:23"
AT+BLEDISCONN=0
```

3.4.15 [ESP32 Only] AT+BLEDATALEN：设置 Bluetooth LE 数据包长度

设置命令

功能：

设置 Bluetooth LE 数据包长度

命令：

```
AT+BLEDATALEN=<conn_index>,<pkt_data_len>
```

响应：

```
OK
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<pkt_data_len>**：数据包长度，范围：0x001b ~ 0x00fb。

说明

- 需要先建立 Bluetooth LE 连接，才能设置数据包长度。

示例

```
AT+BLEINIT=1    // 角色：客户端
AT+BLECONN=0,"24:0a:c4:09:34:23"
AT+BLEDATALEN=0,30
```

3.4.16 [ESP32 Only] AT+BLECFGMTU：设置 Bluetooth LE MTU 长度

查询命令

功能：

查询 MTU（maximum transmission unit，最大传输单元）长度

命令：

```
AT+BLECFGMTU?
```

响应：

```
+BLECFGMTU:<conn_index>,<mtu_size>
OK
```

设置命令

功能：

设置 MTU 的长度

命令：

```
AT+BLECFGMTU=<conn_index>,<mtu_size>
```

响应：

```
OK  // 收到本命令
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<mtu_size>**：MTU 长度。

说明

- 本命令要求先建立 Bluetooth LE 连接。
- 仅支持客户端运行本命令设置 MTU 的长度。
- MTU 的实际长度需要协商，响应 OK 只表示尝试协商 MTU 长度，因此设置长度不一定生效，建议调用*AT+BLECFGMTU?* 查询实际 MTU 长度。

示例

```
AT+BLEINIT=1   // 角色：客户端
AT+BLECONN=0,"24:0a:c4:09:34:23"
AT+BLECFGMTU=0,300
```

3.4.17 [ESP32 Only] AT+BLEGATTSSRVCRE：GATTS 创建服务

执行命令

功能：

GATTS (Generic Attributes Server) 创建 Bluetooth LE 服务

命令：

```
AT+BLEGATTSSRVCRE
```

响应：

```
OK
```

说明

- 使用 ESP 作为 Bluetooth LE server 创建服务，需烧录 service bin 文件到 flash 中。
  - 如何生成 service bin 文件，请参考 esp-at/tools/readme.md。
  - service bin 文件的烧录地址为 esp-at/module_config/module_${platform}_default/at_customize.csv 文件中的“ble_data”地址。
- Bluetooth LE server 初始化后，请及时调用本命令创建服务；如果先建立 Bluetooth LE 连接，则无法创建服务。
- 如果 Bluetooth LE client 已初始化成功，可以使用此命令创建服务；也可以使用其他一些相应的 GATTS 命令，例如启动和停止服务、设置服务特征值和 notification/indication，具体命令如下：
  - *AT+BLEGATTSSRVCRE* (建议在 Bluetooth LE 连接建立之前使用)
  - *AT+BLEGATTSSRVSTART* (建议在 Bluetooth LE 连接建立之前使用)
  - *AT+BLEGATTSSRV*
  - *AT+BLEGATTSCHAR*
  - *AT+BLEGATTSNTFY*
  - *AT+BLEGATTSIND*
  - *AT+BLEGATTSSETATTR*

示例

```
AT+BLEINIT=2    // 角色：服务器
AT+BLEGATTSSRVCRE
```

3.4.18 [ESP32 Only] AT+BLEGATTSSRVSTART：GATTS 开启服务

执行命令

功能：

GATTS 开启全部服务

命令：

```
AT+BLEGATTSSRVSTART
```

设置命令

功能：

GATTS 开启某指定服务

命令：

```
AT+BLEGATTSSRVSTART=<srv_index>
```

响应：

```
OK
```

参数

- **<srv_index>**：服务序号，从 1 开始递增。

示例

```
AT+BLEINIT=2    // 角色：服务器
AT+BLEGATTSSRVCRE
AT+BLEGATTSSRVSTART
```

3.4.19 [ESP32 Only] AT+BLEGATTSSRVSTOP：GATTS 停止服务

执行命令

功能：

GATTS 停止全部服务

命令：

```
AT+BLEGATTSSRVSTOP
```

设置命令

功能：

GATTS 停止某指定服务

命令：

```
AT+BLEGATTSSRVSTOP=<srv_index>
```

响应：

```
OK
```

参数

- **<srv_index>**：服务序号，从 1 开始递增。

示例

```
AT+BLEINIT=2    // 角色：服务器
AT+BLEGATTSSRVCRE
AT+BLEGATTSSRVSTART
AT+BLEGATTSSRVSTOP
```

3.4.20 [ESP32 Only] AT+BLEGATTSSRV：GATTS 发现服务

查询命令

功能：

GATTS 发现服务

命令：

```
AT+BLEGATTSSRV?
```

响应：

```
+BLEGATTSSRV:<srv_index>,<start>,<srv_uuid>,<srv_type>
OK
```

参数

- **<srv_index>**：服务序号，从 1 开始递增。
- **<start>**：
  - 0: 服务未开始；
  - 1: 服务已开始。
- **<srv_uuid>**：服务的 UUID。
- **<srv_type>**：服务的类型：
  - 0: 次要服务；
  - 1: 首要服务。

示例

```
AT+BLEINIT=2    // 角色：服务器
AT+BLEGATTSSRVCRE
AT+BLEGATTSSRV?
```

3.4.21 [ESP32 Only] AT+BLEGATTSCHAR：GATTS 发现服务特征

查询命令

功能：

GATTS 发现服务特征

命令：

```
AT+BLEGATTSCHAR?
```

响应：

对于服务特征信息，响应如下：

```
+BLEGATTSCHAR:"char",<srv_index>,<char_index>,<char_uuid>,<char_prop>
```

对于描述符信息，响应如下：

```
+BLEGATTSCHAR:"desc",<srv_index>,<char_index>,<desc_index>
OK
```

参数

- **<srv_index>**：服务序号，从 1 开始递增。
- **<char_index>**：服务特征的序号，从 1 起始递增。
- **<char_uuid>**：服务特征的 UUID。
- **<char_prop>**：服务特征的属性。
- **<desc_index>**：特征描述符序号。
- **<desc_uuid>**：特征描述符的 UUID。

示例

```
AT+BLEINIT=2    // 角色：服务器
AT+BLEGATTSSRVCRE
AT+BLEGATTSSRVSTART
AT+BLEGATTSCHAR?
```

3.4.22 [ESP32 Only] AT+BLEGATTSNTFY：服务器 notify 服务特征值给客户端

设置命令

功能：

服务器 notify 服务特征值给客户端

命令：

```
AT+BLEGATTSNTFY=<conn_index>,<srv_index>,<char_index>,<length>
```

响应：

```
>
```

符号 > 表示 AT 准备好接收串口数据，此时您可以输入数据，当数据长度达到参数 <length> 的值时，执行 notify 操作。

若数据传输成功，则提示：

```
OK
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<srv_index>**：服务序号，可运行*AT+BLEGATTSCHAR?* 查询。
- **<char_index>**：服务特征的序号，可运行*AT+BLEGATTSCHAR?* 查询。
- **<length>**：数据长度。

示例

```
AT+BLEINIT=2       // 角色：服务器
AT+BLEGATTSSRVCRE
AT+BLEGATTSSRVSTART
AT+BLEADVSTART     // 开始广播，当 client 连接后，必须配置接收 notify
AT+BLEGATTSCHAR?  // 查询允许 notify 客户端的特征
// 例如，使用 3 号服务的 6 号特征 notify 长度为 4 字节的数据，使用如下命令：
AT+BLEGATTSNTFY=0,3,6,4
// 提示 ">" 符号后，输入 4 字节的数据，如 "1234"，然后数据自动传输
```

3.4.23 [ESP32 Only] AT+BLEGATTSIND：服务器 indicate 服务特征值给客户端

设置命令

功能：

服务器 indicate 服务特征值给客户端

命令：

```
AT+BLEGATTSIND=<conn_index>,<srv_index>,<char_index>,<length>
```

响应：

```
>
```

符号 > 表示 AT 准备好接收串口数据，此时您可以输入数据，当数据长度达到参数 <length> 的值时，执行 indicate 操作。

若数据传输成功，则提示：

```
OK
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<srv_index>**：服务序号，可运行*AT+BLEGATTSCHAR?* 查询。
- **<char_index>**：服务特征的序号，可运行*AT+BLEGATTSCHAR?* 查询。
- **<length>**：数据长度。

示例

```
AT+BLEINIT=2       // 角色：服务器
AT+BLEGATTSSRVCRE
AT+BLEGATTSSRVSTART
AT+BLEADVSTART     // 开始广播，当 client 连接后，必须配置接收 indication
AT+BLEGATTSCHAR?   // 查询客户端可以接收 indication 的特征
// 例如，使用 3 号服务的 7 号特征 indicate 长度为 4 字节的数据，命令如下：
AT+BLEGATTSIND=0,3,7,4
// 提示 ">" 符号后，输入 4 字节的数据，如 "1234"，然后数据自动传输
```

3.4.24 [ESP32 Only] AT+BLEGATTSSETATTR：GATTS 设置服务特征值

设置命令

功能：

GATTS 设置服务特征值或描述符值

命令：

```
AT+BLEGATTSSETATTR=<srv_index>,<char_index>,[<desc_index>],<length>
```

响应：

```
>
```

符号 > 表示 AT 准备好接收串口数据，此时您可以输入数据，当数据长度达到参数 <length> 的值时，执行设置操作。

若数据传输成功，则提示：

```
OK
```

参数

- **<srv_index>**：服务序号，可运行*AT+BLEGATTSCHAR?* 查询。
- **<char_index>**：服务特征的序号，可运行*AT+BLEGATTSCHAR?* 查询。
- **[<desc_index>]**：特征描述符序号：
  - 若填写，则设置描述符的值；
  - 若未填写，则设置特征值。
- **<length>**：数据长度。

说明

- 如果 <length> 参数值大于支持的最大长度，则设置会失败。关于 service table，请见 *components/customized_partitions/raw_data/ble_data*。

示例

```
AT+BLEINIT=2    // 角色：服务器
AT+BLEGATTSSRVCRE
AT+BLEGATTSSRVSTART
AT+BLEGATTSCHAR?
// 例如，向 1 号服务的 1 号特征写入长度为 1 字节的数据，命令如下：
AT+BLEGATTSSETATTR=1,1,,1
// 提示 ">" 符号后，输入 1 字节的数据即可，例如 "8"，然后设置开始
```

3.4.25 [ESP32 Only] AT+BLEGATTCPRIMSRV：GATTC 发现基本服务

查询命令

功能：

GATTC (Generic Attributes Client) 发现基本服务

命令：

```
AT+BLEGATTCPRIMSRV=<conn_index>
```

响应：

```
+BLEGATTCPRIMSRV:<conn_index>,<srv_index>,<srv_uuid>,<srv_type>
OK
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<srv_index>**：服务序号，从 1 开始递增。
- **<srv_uuid>**：服务的 UUID。
- **<srv_type>**：服务的类型：
  - 0: 次要服务；
  - 1: 首要服务。

说明

- 使用本命令，需要先建立 Bluetooth LE 连接。

示例

```
AT+BLEINIT=1    // 角色：客户端
AT+BLECONN=0,"24:12:5f:9d:91:98"
AT+BLEGATTCPRIMSRV=0
```

3.4.26 [ESP32 Only] AT+BLEGATTCINCLSRV：GATTC 发现包含的服务

设置命令

功能：

GATTC 发现包含服务

命令：

```
AT+BLEGATTCINCLSRV=<conn_index>,<srv_index>
```

响应：

```
+BLEGATTCINCLSRV:<conn_index>,<srv_index>,<srv_uuid>,<srv_type>,<included_srv_uuid>,
↪<included_srv_type>
OK
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<srv_index>**：服务序号，可运行*AT+BLEGATTCPRIMSRV*=<conn_index> 查询。
- **<srv_uuid>**：服务的 UUID。
- **<srv_type>**：服务的类型：
  - 0: 次要服务；
  - 1: 首要服务。
- **<included_srv_uuid>**：包含服务的 UUID。
- **<included_srv_type>**：包含服务的类型：
  - 0: 次要服务；

  - 1: 首要服务。

说明

- 使用本命令，需要先建立 Bluetooth LE 连接。

示例

```
AT+BLEINIT=1    // 角色：客户端
AT+BLECONN=0,"24:12:5f:9d:91:98"
AT+BLEGATTCPRIMSRV=0
AT+BLEGATTCINCLSRV=0,1   // 根据前一条命令的查询结果，指定 index 查询
```

3.4.27 [ESP32 Only] AT+BLEGATTCCHAR：GATTC 发现服务特征

设置命令

功能：

GATTC 发现服务特征

命令：

```
AT+BLEGATTCCHAR=<conn_index>,<srv_index>
```

响应：

对于服务特征信息，响应如下：

```
+BLEGATTCCHAR:"char",<conn_index>,<srv_index>,<char_index>,<char_uuid>,<char_prop>
```

对于描述符信息，响应如下：

```
+BLEGATTCCHAR:"desc",<conn_index>,<srv_index>,<char_index>,<desc_index>,<desc_uuid>
OK
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<srv_index>**：服务序号，可运行*AT+BLEGATTCPRIMSRV*=<conn_index> 查询。
- **<char_index>**：服务特征的序号，从 1 开始递增。
- **<char_uuid>**：服务特征的 UUID。
- **<char_prop>**：服务特征的属性。
- **<desc_index>**：特征描述符序号。
- **<desc_uuid>**：特征描述符的 UUID。

说明

- 使用本命令，需要先建立 Bluetooth LE 连接。

示例

```
AT+BLEINIT=1    // 角色：客户端
AT+BLECONN=0,"24:12:5f:9d:91:98"
AT+BLEGATTCPRIMSRV=0
AT+BLEGATTCCHAR=0,1 // 根据前一条命令的查询结果，指定 index 查询
```

3.4.28 [ESP32 Only] AT+BLEGATTCRD：GATTC 读取服务特征值

设置命令

功能：

GATTC 读取服务特征值或描述符值

命令：

```
AT+BLEGATTCRD=<conn_index>,<srv_index>,<char_index>[,<desc_index>]
```

响应：

```
+BLEGATTCRD:<conn_index>,<len>,<value>
OK
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<srv_index>**：服务序号，可运行*AT+BLEGATTCPRIMSRV*=<conn_index> 查询。
- **<char_index>**：服务特征序号，可运行*AT+BLEGATTCCHAR*=<conn_index>,<srv_index> 查询。
- **[<desc_index>]**：特征描述符序号：
  - 若设置，读取目标描述符的值；
  - 若未设置，读取目标特征的值。
- **<len>**：数据长度。
- **<char_value>**：服务特征值，HEX 字符串，运行*AT+BLEGATTCRD*=<conn_index>,<srv_index>,<char_index> 读取。例如，若响应为 `+BLEGATTCRD:1,30`，则表示数据长度为 1，内容为 “0x30”。
- **[<desc_value>]**：服务特征描述符的值，HEX 字符串，运行*AT+BLEGATTCRD*=<conn_index>,<srv_index>,<char_index>,<desc 读取。例如，若响应为 `+BLEGATTCRD:4,30313233`，则表示数据长度为 4，内容为 “0x30 0x31 0x32 0x33”。

说明

- 使用本命令，需要先建立 Bluetooth LE 连接。
- 若目标服务特征不支持读操作，则返回 “ERROR”。

示例

```
AT+BLEINIT=1    // 角色：客户端
AT+BLECONN=0,"24:12:5f:9d:91:98"
AT+BLEGATTCPRIMSRV=0
AT+BLEGATTCCHAR=0,3 // 根据前一条命令的查询结果，指定 index 查询
// 例如，读取第 3 号服务的第 2 号特征的第 1 号描述符信息，命令如下：
AT+BLEGATTCRD=0,3,2,1
```

3.4.29 [ESP32 Only] AT+BLEGATTCWR：GATTC 写服务特征值

设置命令

功能：

GATTC 写服务特征值或描述符值

命令：

```
AT+BLEGATTCWR=<conn_index>,<srv_index>,<char_index>[,<desc_index>],<length>
```

Response:

```
>
```

符号 > 表示 AT 准备好接收串口数据，此时您可以输入数据，当数据长度达到参数 <length> 的值时，执行写入操作。

若数据传输成功，则提示：

```
OK
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<srv_index>**：服务序号，可运行*AT+BLEGATTCPRIMSRV*=<conn_index> 查询。
- **<char_index>**：服务特征序号，可运行*AT+BLEGATTCCHAR*=<conn_index>,<srv_index> 查询。
- **[<desc_index>]**：特征描述符序号：
  - 若设置，则写目标描述符的值；
  - 若未设置，则写目标特征的值。
- **<length>**：数据长度。

说明

- 使用本命令，需要先建立 Bluetooth LE 连接。
- 若目标服务特征不支持写操作，则返回 “ERROR”。

示例

```
AT+BLEINIT=1    // 角色：客户端
AT+BLECONN=0,"24:12:5f:9d:91:98"
AT+BLEGATTCPRIMSRV=0
AT+BLEGATTCCHAR=0,3 // 根据前一条命令的查询结果，指定 index 查询
// 例如，向第 3 号服务的第 4 号特征，写入长度为 6 字节的数据，命令如下：
AT+BLEGATTCWR=0,3,4,,6
// 提示 ">" 符号后，输入 6 字节的数据即可，如 "123456"，然后开始写入
```

3.4.30 [ESP32 Only] AT+BLESPPCFG：查询/设置 Bluetooth LE SPP 参数

查询命令

功能：

查询 Bluetooth LE SPP (Serial Port Profile) 参数

命令：

```
AT+BLESPPCFG?
```

响应：

```
+BLESPPCFG:<tx_service_index>,<tx_char_index>,<rx_service_index>,<rx_char_index>
OK
```

设置命令

功能：

设置或重置 Bluetooth LE SPP 参数

命令：

```
AT+BLESPPCFG=<cfg_enable>[,<tx_service_index>,<tx_char_index>,<rx_service_index>,<rx_
↪char_index>]
```

响应：

```
OK
```

参数

- **<cfg_enable>**：
  - 0: 重置所有 SPP 参数，后面四个参数无需填写；
  - 1: 后面四个参数需要填写。
- **<tx_service_index>**：tx 服务序号，可运行*AT+BLEGATTCPRIMSRV*=<conn_index> 和*AT+BLEGATTSSRV?* 查询。
- **<tx_char_index>**：tx 服务特征序号，可运行*AT+BLEGATTCCHAR*=<conn_index>,<srv_index> 和*AT+BLEGATTSCHAR?* 查询。
- **<rx_service_index>**：rx 服务序号，可运行*AT+BLEGATTCPRIMSRV*=<conn_index> 和*AT+BLEGATTSSRV?* 查询。

- **<rx_char_index>**：rx 服务特征序号，可运行*AT+BLEGATTCCHAR*=<conn_index>,<srv_index>和*AT+BLEGATTSCHAR?* 查询。

说明

- 对于 Bluetooth LE 客户端，tx 服务特征属性必须是 `write with response` 或 `write without response`，rx 服务特征属性必须是 `indicate` 或 `notify`。
- 对于 Bluetooth LE 服务器，tx 服务特征属性必须是 `indicate` 或 `notify`，rx 服务特征属性必须是 `write with response` 或 `write without response`。

示例

```
AT+BLESPPCFG=0           // 重置 Bluetooth LE SPP 参数
AT+BLESPPCFG=1,3,5,3,7   // 设置 Bluetooth LE SPP 参数
AT+BLESPPCFG?            // 查询 Bluetooth LE SPP 参数
```

3.4.31 [ESP32 Only] AT+BLESPP：进入 Bluetooth LE SPP 模式

执行命令

功能：

进入 Bluetooth LE SPP 模式

命令：

```
AT+BLESPP
```

响应：

```
>
```

说明

- 若 Bluetooth LE SPP 参数非法，则命令返回 `ERROR`。
- 在 SPP 传输中，若未设置*AT+SYSMSG* 为 1，则 AT 不会提示任何连接状态变更信息。

示例

```
AT+BLESPP   // 进入 Bluetooth LE SPP 模式
```

3.4.32 [ESP32 Only] AT+BLESECPARAM：查询/设置 Bluetooth LE 加密参数

查询命令

功能：

查询 Bluetooth LE SMP 加密参数

命令：

```
AT+BLESECPARAM?
```

响应：

```
+BLESECPARAM:<auth_req>,<iocap>,<key_size>,<init_key>,<rsp_key>,<auth_option>
OK
```

设置命令

功能：

设置 Bluetooth LE SMP 加密参数

命令：

```
AT+BLESECPARAM=<auth_req>,<iocap>,<key_size>,<init_key>,<rsp_key>[,<auth_option>]
```

响应：

```
OK
```

参数

- **<auth_req>**：认证请求。
  - 0: NO_BOND
  - 1: BOND
  - 4: MITM
  - 8: SC_ONLY

  - 9: SC_BOND
  - 12: SC_MITM
  - 13: SC_MITM_BOND
- **<iocap>**：输入输出能力。
  - 0: DisplayOnly
  - 1: DisplayYesNo
  - 2: KeyboardOnly
  - 3: NoInputNoOutput
  - 4: Keyboard display
- **<key_size>**：密钥长度，取值范围 7 ~ 16 字节。
- **<init_key>**：多个比特位组成的初始密钥。
- **<rsp_key>**：多个比特位组成的响应密钥。
- **<auth_option>**：安全认证选项：
  - 0: 自动选择安全等级；
  - 1: 如果无法满足之前设定的安全等级，则会断开连接。

说明

- `<init_key>` 和 `<rsp_key>` 参数的比特位组合模式如下：
  - Bit0: 用于交换初始密钥和响应密钥的加密密钥；
  - Bit1: 用于交换初始密钥和响应密钥的 IRK 密钥；
  - Bit2: 用于交换初始密钥和响应密钥的 CSRK 密钥；
  - Bit3: 用于交换初始密钥和响应密钥的 link 密钥（仅用于 Bluetooth LE 和 BR/EDR 共存模式）。

示例

```
AT+BLESECPARAM=1,4,16,3,3,0
```

3.4.33 [ESP32 Only] AT+BLEENC：发起 Bluetooth LE 加密请求

设置命令

功能：

发起配对请求

命令：

```
AT+BLEENC=<conn_index>,<sec_act>
```

响应：

```
OK
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<sec_act>**：
  - 0: SEC_NONE；
  - 1: SEC_ENCRYPT；
  - 2: SEC_ENCRYPT_NO_MITM；
  - 3: SEC_ENCRYPT_MITM。

说明

- 使用本命令前，请先设置安全参数、建立与对方设备的连接。

示例

```
AT+BLESECPARAM=1,4,16,3,3
AT+BLEENC=0,3
```

3.4.34 [ESP32 Only] AT+BLEENCRSP：回复对端设备发起的配对请求

设置命令

功能：

回复对端设备发起的配对请求

命令：

```
AT+BLEENCRSP=<conn_index>,<accept>
```

响应：

```
OK
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<accept>**：
  - 0: 拒绝；
  - 1: 接受。

说明

- 使用本命令后，AT 会在配对请求流程结束后输出配对结果。

```
+BLEAUTHCMPL:<conn_index>,<enc_result>
```

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<enc_result>**：
  - 0: 加密配对成功；
  - 1: 加密配对失败。

示例

```
AT+BLEENCRSP=0,1
```

3.4.35 [ESP32 Only] AT+BLEKEYREPLY：给对方设备回复密钥（传统连接阶段）

设置命令

功能：

回复配对密钥

命令：

```
AT+BLEKEYREPLY=<conn_index>,<key>
```

响应：

```
OK
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<key>**：配对密钥。

示例

```
AT+BLEKEYREPLY=0,649784
```

3.4.36 [ESP32 Only] AT+BLECONFREPLY：给对方设备回复确认结果（传统连接阶段）

设置命令

功能：

回复配对结果

命令：

```
AT+BLECONFREPLY=<conn_index>,<confirm>
```

响应：

```
OK
```

参数

- **<conn_index>**：Bluetooth LE 连接号，范围：[0,2]。
- **<confirm>**：
  - 0: 否
  - 1: 是

示例

```
AT+BLECONFREPLY=0,1
```

3.4.37 [ESP32 Only] AT+BLEENCDEV：查询绑定的 Bluetooth LE 加密设备列表

查询命令

功能：

查询绑定的 Bluetooth LE 加密设备列表

命令：

```
AT+BLEENCDEV?
```

响应：

```
+BLEENCDEV:<enc_dev_index>,<mac_address>
OK
```

参数

- **<enc_dev_index>**：已绑定设备的连接号。
- **<mac_address>**：MAC 地址。

示例

```
AT+BLEENCDEV?
```

3.4.38 [ESP32 Only] AT+BLEENCCLEAR：清除 Bluetooth LE 加密设备列表

设置命令

功能：

从安全数据库列表中删除某一连接号的设备

命令：

```
AT+BLEENCCLEAR=<enc_dev_index>
```

响应：

```
OK
```

执行命令

功能：

删除安全数据库所有设备

命令：

```
AT+BLEENCCLEAR
```

响应：

```
OK
```

参数

- **<enc_dev_index>**：已绑定设备的连接号。

示例

```
AT+BLEENCCLEAR
```

3.4.39 [ESP32 Only] AT+BLESETKEY：设置 Bluetooth LE 静态配对密钥

查询命令

功能：

查询 Bluetooth LE 静态配对密钥，若未设置，则 AT 返回 -1

命令：

```
AT+BLESETKEY?
```

响应：

```
+BLESETKEY:<static_key>
OK
```

设置命令

功能：

为所有 Bluetooth LE 连接设置一个 Bluetooth LE 静态配对密钥

命令：

```
AT+BLESETKEY=<static_key>
```

响应：

```
OK
```

参数

- **<static_key>**：Bluetooth LE 静态配对密钥。

示例

```
AT+BLESETKEY=123456
```

3.4.40 [ESP32 Only] AT+BLEHIDINIT：Bluetooth LE HID 协议初始化

查询命令

功能：

查询 Bluetooth LE HID 协议初始化情况

命令：

```
AT+BLEHIDINIT?
```

响应：

若未初始化，则 AT 返回：

```
+BLEHIDINIT:0
OK
```

若已初始化，则 AT 返回：

```
+BLEHIDINIT:1
OK
```

设置命令

功能：

初始化 Bluetooth LE HID 协议

命令：

```
AT+BLEHIDINIT=<init>
```

响应：

```
OK
```

参数

- **<init>**：
  - 0: 取消 Bluetooth LE HID 协议的初始化；
  - 1: 初始化 Bluetooth LE HID 协议。

说明

- Bluetooth LE HID 无法与通用 GATT/GAP 命令同时使用。

示例

```
AT+BLEHIDINIT=1
```

3.4.41 [ESP32 Only] AT+BLEHIDKB：发送 Bluetooth LE HID 键盘信息

设置命令

功能：

发送键盘信息

命令：

```
AT+BLEHIDKB=<Modifier_keys>,<key_1>,<key_2>,<key_3>,<key_4>,<key_5>,<key_6>
```

响应：

```
OK
```

参数

- **<Modifier_keys>**：组合键。
- **<key_1>**：键代码 1。
- **<key_2>**：键代码 2。
- **<key_3>**：键代码 3。
- **<key_4>**：键代码 4。
- **<key_5>**：键代码 5。
- **<key_6>**：键代码 6。

说明

- 更多键代码的信息，请参考 Universal Serial Bus HID Usage Tables 的 Keyboard/Keypad Page 章节。

示例

```
AT+BLEHIDKB=0,4,0,0,0,0,0    // 输入字符串 "a"
```

3.4.42 [ESP32 Only] AT+BLEHIDMUS：发送 Bluetooth LE HID 鼠标信息

设置命令

功能：

发送鼠标信息

命令：

```
AT+BLEHIDMUS=<buttons>,<X_displacement>,<Y_displacement>,<wheel>
```

响应：

```
OK
```

参数

- **<buttons>**：鼠标按键。
- **<X_displacement>**：X 位移。
- **<Y_displacement>**：Y 位移。
- **<wheel>**：滚轮。

示例

```
AT+BLEHIDMUS=0,10,10,0
```

3.4.43 [ESP32 Only] AT+BLEHIDCONSUMER：发送 Bluetooth LE HID consumer 信息

设置命令

功能：

发送 consumer 信息

命令：

```
AT+BLEHIDCONSUMER=<consumer_usage_id>
```

响应：

```
OK
```

参数

- **<consumer_usage_id>**：consumer ID，如 power、reset、help、volume 等。详情请参考 HID Usage Tables for Universal Serial Bus (USB) 中的 Consumer Page (0x0C) 章节。

示例

```
AT+BLEHIDCONSUMER=233    // 调高音量
```

3.4.44 [ESP32 Only] AT+BLUFI：开启或关闭 BluFi

查询命令

功能：

查询 BluFi 状态

命令：

```
AT+BLUFI?
```

响应：

若 BluFi 未开启，则返回：

```
+BLUFI:0
OK
```

若 BluFi 已开启，则返回：

```
+BLUFI:1
OK
```

设置命令

功能：

开启或关闭 BluFi

命令：

```
AT+BLUFI=<option>[,<auth floor>]
```

响应：

```
OK
```

参数

- **<option>**：
  - 0: 关闭 BluFi；
  - 1: 开启 BluFi。
- **<auth floor>**：Wi-Fi 认证模式阈值，ESP-AT 不会连接到认证模式低于此阈值的 AP：
  - 0: OPEN（默认）；
  - 1: WEP；
  - 2: WPA_PSK；
  - 3: WPA2_PSK；
  - 4: WPA_WPA2_PSK；
  - 5: WPA2_ENTERPRISE；
  - 6: WPA3_PSK；
  - 7: WPA2_WPA3_PSK。

示例

```
AT+BLUFI=1
```

3.4.45 [ESP32 Only] AT+BLUFINAME：查询/设置 BluFi 设备名称

查询命令

功能：

查询 BluFi 名称

命令：

```
AT+BLUFINAME?
```

响应：

```
+BLUFINAME:<device_name>
OK
```

设置命令

功能：

设置 BluFi 设备名称

命令：

```
AT+BLUFINAME=<device_name>
```

响应：

```
OK
```

参数

- **<device_name>**：BluFi 设备名称。

说明

- 如需设置 BluFi 设备名称，请在运行*AT+BLUFI=1* 命令前设置，否则将使用默认名称 `BLUFI_DEVICE`。
- BluFi 设备名称最大长度为 29 字节。

示例

```
AT+BLUFINAME="BLUFI_DEV"
AT+BLUFINAME?
```

3.5 [ESP32 Only] Classic Bluetooth® AT 命令集

[English]

ESP32 AT 固件支持 蓝牙核心规范 5.0 版本。

- [ESP32 Only] *AT+BTINIT*：Classic Bluetooth 初始化
- [ESP32 Only] *AT+BTNAME*：查询/设置 Classic Bluetooth 设备名称
- [ESP32 Only] *AT+BTSCANMODE*：设置 Classic Bluetooth 扫描模式
- [ESP32 Only] *AT+BTSTARTDISC*：开始发现周边 Classic Bluetooth 设备
- [ESP32 Only] *AT+BTSPPINIT*：Classic Bluetooth SPP 协议初始化
- [ESP32 Only] *AT+BTSPPCONN*：查询/建立 SPP 连接
- [ESP32 Only] *AT+BTSPPDISCONN*：断开 SPP 连接
- [ESP32 Only] *AT+BTSPPSTART*：开启 Classic Bluetooth SPP 协议
- [ESP32 Only] *AT+BTSPPSEND*：发送数据到对方 Classic Bluetooth SPP 设备
- [ESP32 Only] *AT+BTA2DPINIT*：Classic Bluetooth A2DP 协议初始化
- [ESP32 Only] *AT+BTA2DPCONN*：查询/建立 A2DP 连接
- [ESP32 Only] *AT+BTA2DPDISCONN*：断开 A2DP 连接
- [ESP32 Only] *AT+BTA2DPSRC*：查询/设置音频文件 URL
- [ESP32 Only] *AT+BTA2DPCTRL*：控制音频播放
- [ESP32 Only] *AT+BTSECPARAM*：查询/设置 Classic Bluetooth 安全参数
- [ESP32 Only] *AT+BTKEYREPLY*：输入简单配对密钥 (Simple Pair Key)
- [ESP32 Only] *AT+BTPINREPLY*：输入传统配对密码 (Legacy Pair PIN Code)
- [ESP32 Only] *AT+BTSECCFM*：给对方设备回复确认结果（传统连接阶段）

- [ESP32 Only] *AT+BTENCDEV*：查询 Classic Bluetooth 加密设备列表
- [ESP32 Only] *AT+BTENCCLEAR*：清除 Classic Bluetooth 加密设备列表
- [ESP32 Only] *AT+BTCOD*：设置设备类型
- [ESP32 Only] *AT+BTPOWER*：查询/设置 Classic Bluetooth 的 TX 功率

3.5.1 [ESP32 Only] AT+BTINIT：Classic Bluetooth 初始化

查询命令

功能：

查询 Classic Bluetooth 初始化状态

命令：

```
AT+BTINIT?
```

响应：

若已初始化，则返回：

```
+BTINIT:1
OK
```

若未初始化，则返回：

```
+BTINIT:0
OK
```

设置命令

功能：

初始化或注销 Classic Bluetooth

命令：

```
AT+BTINIT=<init>
```

响应：

```
OK
```

参数

- **<init>**：
  - 0: 注销 Classic Bluetooth；
  - 1: 初始化 Classic Bluetooth。

示例

```
AT+BTINIT=1
```

3.5.2 [ESP32 Only] AT+BTNAME：查询/设置 Classic Bluetooth 设备名称

查询命令

功能：

查询 Classic Bluetooth 设备名称

命令：

```
AT+BTNAME?
```

响应：

```
+BTNAME:<device_name>
OK
```

设置命令

功能：

设置 Classic Bluetooth 设备名称

命令：

```
AT+BTNAME=<device_name>
```

响应：

```
OK
```

参数

- **<device_name>**：Classic Bluetooth 设备名称，最大长度为：32。

说明

- 若*AT+SYSSTORE=1*，配置更改将保存在 NVS 区。
- 默认 Classic Bluetooth 设备名称为“ESP32_AT”。

示例

```
AT+BTNAME="esp_demo"
```

3.5.3 [ESP32 Only] AT+BTSCANMODE：设置 Classic Bluetooth 扫描模式

设置命令

功能：

设置 Classic Bluetooth 扫描模式

命令：

```
AT+BTSCANMODE=<scan_mode>
```

响应：

```
OK
```

参数

- **<scan_mode>**：
  - 0: 不可发现且不可连接；
  - 1: 可连接但不可发现；
  - 2: 既可发现也可连接；
  - 3: 可发现但不可连接。

示例

```
AT+BTSCANMODE=2    // 既可发现也可连接
```

3.5.4 [ESP32 Only] AT+BTSTARTDISC：开始发现周边 Classic Bluetooth 设备

设置命令

功能：

开始发现 Classic Bluetooth 设备

命令：

```
AT+BTSTARTDISC=<inq_mode>,<inq_len>,<inq_num_rsps>
```

响应：

```
+BTSTARTDISC:<bt_addr>,<dev_name>,<major_dev_class>,<minor_dev_class>,<major_srv_
↪class>,<rssi>

OK
```

参数

- **<inq_mode>**：
  - 0: general inquiry mode；
  - 1: limited inquiry mode。
- **<inq_len>**：inquiry 时长，范围：0x01 ~ 0x30。
- **<inq_num_rsps>**：可以收到的 inquiry responses 的数量，若设为 0，AT 将收到无限个 response。
- **<bt_addr>**：Classic Bluetooth 地址。
- **<dev_name>**：设备名称。
- **<major_dev_class>**：主要设备类型：
  - 0x0: 其他；
  - 0x1: 计算机；
  - 0x2: 电话（手机、无绳、支付电话、调制解调器）；
  - 0x3: LAN、网络接入点；
  - 0x4: 音频/视频（耳机、扬声器、立体声、视频显示、VCR）；

  - 0x5: 配件（鼠标、游戏杆、键盘）；
  - 0x6: 成像（打印、扫描仪、相机、显示）；
  - 0x7: 可穿戴；
  - 0x8: 玩具；
  - 0x9: 健康；
  - 0x1F: 未分类。
- **<minor_dev_class>**：请参考 次要设备类型 (Minor Device Class field)。
- **<major_srv_class>**：主要服务类型：
  - 0x0: 无效值；
  - 0x1: 有限可发现模式 (Limited Discoverable Mode)；
  - 0x8: 定位（位置标志）；
  - 0x10: 网络，如 LAN、点对点；
  - 0x20: 渲染，如打印、扬声器；
  - 0x40: 捕捉，如扫描仪、麦克风；
  - 0x80: 对象传输，如 v-Inbox、v-Folder；
  - 0x100: 音频，如扬声器、麦克风、耳机服务；
  - 0x200: 电话，如无绳电话、调制解调器、耳机服务；
  - 0x400: 信息，如 WEB 服务器、WAP 服务器。
- **<rssi>**：信号强度。

示例

```
AT+BTINIT=1
AT+BTSCANMODE=2
AT+BTSTARTDISC=0,10,10
```

3.5.5 [ESP32 Only] AT+BTSPPINIT：Classic Bluetooth SPP 协议初始化

查询命令

功能：

查询 Classic Bluetooth SPP 协议初始化状态

命令：

```
AT+BTSPPINIT?
```

响应：

若已初始化，则返回：

```
+BTSPPINIT:1
OK
```

若未初始化，则返回：

```
+BTSPPINIT:0
OK
```

设置命令

功能：

初始化或注销 Classic Bluetooth SPP 协议

命令：

```
AT+BTSPPINIT=<init>
```

响应：

```
OK
```

参数

- **<init>**：
  - 0: 注销 Classic Bluetooth SPP 协议；
  - 1: 初始化 Classic Bluetooth SPP 协议，角色为 master；
  - 2: 初始化 Classic Bluetooth SPP 协议，角色为 slave。

示例

```
AT+BTSPPINIT=1     // master
AT+BTSPPINIT=2     // slave
```

3.5.6 [ESP32 Only] AT+BTSPPCONN：查询/建立 SPP 连接

查询命令

功能：

查询 Classic Bluetooth SPP 连接

命令：

```
AT+BTSPPCONN?
```

响应：

```
+BTSPPCONN:<conn_index>,<remote_address>
OK
```

如果未建立连接，则返回：

```
+BTSPPCONN:-1
```

设置命令

功能：

建立 Classic Bluetooth SPP 连接

命令：

```
AT+BTSPPCONN=<conn_index>,<sec_mode>,<remote_address>
```

响应：

```
OK
```

若建立连接成功，则 AT 返回：

```
+BTSPPCONN:<conn_index>,<remote_address>
```

若建立连接失败，则 AT 返回：

```
+BTSPPCONN:<conn_index>,-1
```

参数

- **<conn_index>**：Classic Bluetooth SPP 连接号，当前只支持单连接，连接号为 0。
- **<sec_mode>**：
  - 0x0000: 无安全保障；
  - 0x0001: 需要授权（仅对外连接需要）；
  - 0x0036: 需要加密；
  - 0x3000: 中间人保护；
  - 0x4000: 最少 16 位密码。
- **<remote_address>**：对方 Classic Bluetooth SPP 设备地址。

示例

```
AT+BTSPPCONN=0,0,"24:0a:c4:09:34:23"
```

3.5.7 [ESP32 Only] AT+BTSPPDISCONN：断开 SPP 连接

执行命令

功能：

断开 Classic Bluetooth SPP 连接

命令：

```
AT+BTSPPDISCONN=<conn_index>
```

响应：

```
OK
```

若命令运行成功，则返回：

```
+BTSPPDISCONN:<conn_index>,<remote_address>
```

若命令运行失败，则返回：

```
+BTSPPDISCONN:-1
```

参数

- **<conn_index>**：Classic Bluetooth SPP 连接号，当前只支持单连接，连接号为 0。
- **<remote_address>**：对方 Classic Bluetooth A2DP 设备地址。

示例

```
AT+BTSPPDISCONN=0
```

3.5.8 [ESP32 Only] AT+BTSPPSEND：发送数据到对方 Classic Bluetooth SPP 设备

执行命令

功能：

进入 Classic Bluetooth SPP 模式

命令：

```
AT+BTSPPSEND
```

响应：

```
>
```

设置命令

功能：

发送数据到对方 Classic Bluetooth SPP 设备

命令：

```
AT+BTSPPSEND=<conn_index>,<data_len>
```

响应：

```
OK
```

参数

- **<conn_index>**：Classic Bluetooth SPP 连接号，当前只支持单连接，连接号为 0。
- **<data_len>**：发送数据的长度。

说明

- 系统收到此命令后先换行返回 >，然后 ESP 设备进入 UART Bluetooth 透传模式，当系统收到只含有+++的包时，设备返回到普通命令模式，请等待一秒再发送下一个 AT 命令。

示例

```
AT+BTSPPSEND=0,100
AT+BTSPPSEND
```

3.5.9 [ESP32 Only] AT+BTSPPSTART：开启 Classic Bluetooth SPP 协议

执行命令

功能：

开启 Classic Bluetooth SPP 协议

命令：

```
AT+BTSPPSTART
```

响应：

```
OK
```

说明

- 在 SPP 传输中，如果未设置*AT+SYSMSG* 命令的 bit2 为 1，则系统不会提示任何连接状态改变的信息。

示例

```
AT+BTSPPSTART
```

3.5.10 [ESP32 Only] AT+BTA2DPINIT：Classic Bluetooth A2DP 协议初始化

查询命令

功能：

查询 Classic Bluetooth A2DP 协议的初始化状态

命令：

```
AT+BTA2DPINIT?
```

响应：

若已初始化，则返回：

```
+BTA2DPINIT:1
OK
```

若未初始化，则返回：

```
+BTA2DPINIT:0
OK
```

设置命令

功能：

初始化或注销 Classic Bluetooth A2DP 协议

命令：

```
AT+BTA2DPINIT=<role>,<init_val>
```

响应：

```
OK
```

参数

- **<role>**：角色
  - 0: source；
  - 1: sink。
- **<init_val>**：
  - 0: 注销 Classic Bluetooth A2DP 协议；
  - 1: 初始化 Classic Bluetooth A2DP 协议。

示例

```
AT+BTA2DPINIT=0,1
```

3.5.11 [ESP32 Only] AT+BTA2DPCONN：查询/建立 A2DP 连接

查询命令

功能：

查询 Classic Bluetooth A2DP 连接

命令：

```
AT+BTA2DPCONN?
```

响应：

```
+BTA2DPCONN:<conn_index>,<remote_address>
OK
```

若未建立连接，则 AT 不会返回 <conn_index> 和 <remote_address> 参数。

设置命令

功能：

建立 Classic Bluetooth A2DP 连接

命令：

```
AT+BTA2DPCONN=<conn_index>,<remote_address>
```

响应：

```
OK
```

若建立连接成功，则返回：

```
+BTA2DPCONN:<conn_index>,<remote_address>
```

若建立连接失败，则返回：

```
+BTA2DPCONN:<conn_index>,-1
```

参数

- **<conn_index>**：Classic Bluetooth A2DP 连接号，当前只支持单连接，连接号为 0。
- **<remote_address>**：对方 Classic Bluetooth A2DP 设备地址。

示例

```
AT+BTA2DPCONN=0,0,0,"24:0a:c4:09:34:23"
```

3.5.12 [ESP32 Only] AT+BTA2DPDISCONN：断开 A2DP 连接

执行命令

功能：

断开 Classic Bluetooth A2DP 连接

命令：

```
AT+BTA2DPDISCONN=<conn_index>
```

响应：

```
+BTA2DPDISCONN:<conn_index>,<remote_address>
OK
```

参数

- **<conn_index>**：Classic Bluetooth A2DP 连接号，当前只支持单连接，连接号为 0。
- **<remote_address>**：对方 Classic Bluetooth A2DP 设备地址。

示例

```
AT+BTA2DPDISCONN=0
```

3.5.13 [ESP32 Only] AT+BTA2DPSRC：查询/设置音频文件 URL

查询命令

功能：

查询音频文件 URL

命令：

```
AT+BTA2DPSRC?
```

响应：

```
+BTA2DPSRC:<url>,<type>
OK
```

执行命令

功能：

设置音频文件 URL

命令：

```
AT+BTA2DPSRC=<conn_index>,<url>
```

响应：

```
OK
```

参数

- **<conn_index>**：Classic Bluetooth A2DP 连接号，当前只支持单连接，连接号为 0。
- **<url>**：源文件路径，当前只支持 HTTP、HTTPS 和 FLASH。
- **<type>**：音频文件类型，如“mp3”。

说明

- 当前只支持 mp3 格式文件。

示例

```
AT+BTA2DPSRC=0,"https://dl.espressif.com/dl/audio/ff-16b-2c-44100hz.mp3"
AT+BTA2DPSRC=0,"flash://spiffs/zhifubao.mp3"
```

3.5.14 [ESP32 Only] AT+BTA2DPCTRL：控制音频播放

执行命令

功能：

控制音频播放

命令：

```
AT+BTA2DPCTRL=<conn_index>,<ctrl>
```

响应：

```
OK
```

参数

- **<conn_index>**：Classic Bluetooth A2DP 连接号，当前只支持单连接，连接号为 0。
- **<ctrl>**：控制类型：
  - 0: A2DP Sink，停止播放；
  - 1: A2DP Sink，开始播放；
  - 2: A2DP Sink，快进；
  - 3: A2DP Sink，后退；

- 4: A2DP Sink，快进启动；
- 5: A2DP Sink，快进停止；
- 0: A2DP Source，停止播放；
- 1: A2DP Source，开始播放；
- 2: A2DP Source，暂停播放。

示例

```
AT+BTA2DPCTRL=0,1  // 开始播放音频
```

3.5.15 [ESP32 Only] AT+BTSECPARAM：查询/设置 Classic Bluetooth 安全参数

查询命令

功能：

查询 Classic Bluetooth 安全参数

命令：

```
AT+BTSECPARAM?
```

响应：

```
+BTSECPARAM:<io_cap>,<pin_type>,<pin_code>
OK
```

设置命令

功能：

设置 Classic Bluetooth 安全参数

命令：

```
AT+BTSECPARAM=<io_cap>,<pin_type>,<pin_code>
```

响应：

```
OK
```

参数

- **<io_cap>**：输入输出能力：
  - 0: DisplayOnly；
  - 1: DisplayYesNo；
  - 2: KeyboardOnly；
  - 3: NoInputNoOutput。
- **<pin_type>**：使用可变或固定密码：
  - 0: 可变密码；
  - 1: 固定密码。
- **<pin_code>**：传统配对密码，最大长度：16 字节。

说明

- 若设置 <pin_type> 为 0，则会自动忽略 <pin_code> 参数。

示例

```
AT+BTSECPARAM=3,1,"9527"
```

3.5.16 [ESP32 Only] AT+BTKEYREPLY：输入简单配对密钥 (Simple Pair Key)

执行命令

功能：

输入简单配对密钥 (Simple Pair Key)

命令：

```
AT+BTKEYREPLY=<conn_index>,<Key>
```

响应：

```
OK
```

参数

- **<conn_index>**：Classic Bluetooth 连接号，当前只支持单连接，连接号为 0。
- **<Key>**：简单配对密钥 (Simple Pair Key)。

示例

```
AT+BTKEYREPLY=0,123456
```

3.5.17 [ESP32 Only] AT+BTPINREPLY：输入传统配对密码 (Legacy Pair PIN Code)

执行命令

功能：

输入传统配对密码 (Legacy Pair PIN Code)

命令：

```
AT+BTPINREPLY=<conn_index>,<Pin>
```

响应：

```
OK
```

参数

- **<conn_index>**：Classic Bluetooth 连接号，当前只支持单连接，连接号为 0。
- **<Pin>**：传统配对密码 (Legacy Pair PIN Code)。

示例

```
AT+BTPINREPLY=0,"6688"
```

3.5.18 [ESP32 Only] AT+BTSECCFM：给对方设备回复确认结果（传统连接阶段）

执行命令

功能：

给对方设备回复确认结果（传统连接阶段）

命令：

```
AT+BTSECCFM=<conn_index>,<accept>
```

响应：

```
OK
```

参数

- **<conn_index>**：Classic Bluetooth 连接，当前只支持单连接，连接号为 0。
- **<accept>**：拒绝或接受：
  - 0: 拒绝；
  - 1: 接受。

示例

```
AT+BTSECCFM=0,1
```

3.5.19 [ESP32 Only] AT+BTENCDEV：查询 Classic Bluetooth 加密设备列表

查询命令

功能：

查询绑定设备

命令：

```
AT+BTENCDEV?
```

响应：

```
+BTENCDEV:<enc_dev_index>,<mac_address>
OK
```

参数

- **<enc_dev_index>**：绑定设备序号。
- **<mac_address>**：MAC 地址。

示例

```
AT+BTENCDEV?
```

3.5.20 [ESP32 Only] AT+BTENCCLEAR：清除 Classic Bluetooth 加密设备列表

设置命令

功能：

从安全数据库列表中删除某一序号的设备

命令：

```
AT+BTENCCLEAR=<enc_dev_index>
```

响应：

```
OK
```

执行命令

功能：

删除安全数据库所有设备

命令：

```
AT+BLEENCCLEAR
```

响应：

```
OK
```

参数

- **<enc_dev_index>**：绑定设备序号。

示例

```
AT+BTENCCLEAR
```

3.5.21 [ESP32 Only] AT+BTCOD：设置设备类型

设置命令

功能：

设置 Classic Bluetooth 设备类型

命令：

```
AT+BTCOD=<major>,<minor>,<service>
```

响应：

```
OK
```

参数

- **<major>**：主要设备类型 (major class)；
- **<minor>**：次要设备类型 (minor class)；
- **<service>**：服务类型 (service class)。

示例

```
AT+BTCOD=6,32,32    // 打印机
```

3.5.22 [ESP32 Only] AT+BTPOWER：查询/设置 Classic Bluetooth 的 TX 功率

查询命令

功能：

查询 Classic Bluetooth 的 TX 功率

命令：

```
AT+BTPOWER?
```

响应：

```
+BTPOWER:<min_tx_power>,<max_tx_power>
OK
```

设置命令

功能：

设置 Classic Bluetooth 的 TX 功率

命令：

```
AT+BTPOWER=<min_tx_power>,<max_tx_power>
```

响应：

```
OK
```

参数

- **<min_tx_power>**：最小功率水平，范围：[0,7]。
- **<max_tx_power>**：最大功率水平，范围：[0,7]。

示例

```
AT+BTPOWER=5,6   // 设置 Classic Bluetooth tx 功率
```

3.6 MQTT AT Commands

[English]

- *AT+MQTTUSERCFG*：设置 MQTT 用户属性
- *AT+MQTTLONGCLIENTID*：设置 MQTT 客户端 ID
- *AT+MQTTLONGUSERNAME*：设置 MQTT 登陆用户名
- *AT+MQTTLONGPASSWORD*：设置 MQTT 登陆密码
- *AT+MQTTCONNCFG*：设置 MQTT 连接属性
- *AT+MQTTCONN*：连接 MQTT Broker
- *AT+MQTTPUB*：发布 MQTT 消息（字符串）
- *AT+MQTTPUBRAW*：发布 MQTT 消息（二进制）
- *AT+MQTTSUB*：订阅 MQTT Topic
- *AT+MQTTUNSUB*：取消订阅 MQTT Topic
- *AT+MQTTCLEAN*：断开 MQTT 连接
- *MQTT AT 错误码*
- *MQTT AT 说明*

3.6.1 AT+MQTTUSERCFG：设置 MQTT 用户属性

设置命令

功能：

配置 MQTT 用户属性

命令：

```
AT+MQTTUSERCFG=<LinkID>,<scheme>,<"client_id">,<"username">,<"password">,<cert_key_ID>
↪,<CA_ID>,<"path">
```

响应：

```
OK
```

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<scheme>**：由于 ESP8266 内存限制，不支持 MQTT over TLS，即 <scheme> 只能取 1 或 6。
  - 1: MQTT over TCP；
  - 2: MQTT over TLS（不校验证书）；
  - 3: MQTT over TLS（校验 server 证书）；
  - 4: MQTT over TLS（提供 client 证书）；
  - 5: MQTT over TLS（校验 server 证书并且提供 client 证书）；
  - 6: MQTT over WebSocket（基于 TCP）；
  - 7: MQTT over WebSocket Secure（基于 TLS，不校验证书）；
  - 8: MQTT over WebSocket Secure（基于 TLS，校验 server 证书）；
  - 9: MQTT over WebSocket Secure（基于 TLS，提供 client 证书）；
  - 10: MQTT over WebSocket Secure（基于 TLS，校验 server 证书并且提供 client 证书）。
- **<client_id>**：MQTT 客户端 ID，最大长度：256 字节。
- **<username>**：用户名，用于登陆 MQTT broker，最大长度：64 字节。
- **<password>**：密码，用于登陆 MQTT broker，最大长度：64 字节。
- **<cert_key_ID>**：证书 ID，目前 ESP-AT 仅支持一套 cert 证书，参数为 0。
- **<CA_ID>**：CA ID，目前 ESP-AT 仅支持一套 CA 证书，参数为 0。
- **<path>**：资源路径，最大长度：32 字节。

说明

- 每条 AT 命令的总长度不能超过 256 字节。

3.6.2 AT+MQTTLONGCLIENTID：设置 MQTT 客户端 ID

设置命令

功能：

设置 MQTT 客户端 ID

命令：

```
AT+MQTTLONGCLIENTID=<LinkID>,<length>
```

响应：

```
OK


>
```

上述响应表示 AT 已准备好接收 MQTT 客户端 ID，此时您可以输入客户端 ID，当 AT 接收到的客户端 ID 长度达到 <length> 后，返回：

```
OK
```

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<length>**：MQTT 客户端 ID 长度。范围：[1,1024]。

说明

- *AT+MQTTUSERCFG* 命令也可以设置 MQTT 客户端 ID，二者之间的差别包括：
  - AT+MQTTLONGCLIENTID 命令可以用来设置相对较长的客户端 ID，因为 AT+MQTTUSERCFG 命令的长度受限；
  - 应在设置 AT+MQTTUSERCFG 后再使用 AT+MQTTLONGCLIENTID。

3.6.3 AT+MQTTLONGUSERNAME：设置 MQTT 登陆用户名

设置命令

功能：

设置 MQTT 用户名

命令：

```
AT+MQTTLONGUSERNAME=<LinkID>,<length>
```

响应：

```
OK


>
```

上述响应表示 AT 已准备好接收 MQTT 用户名，此时您可以输入 MQTT 用户名，当 AT 接收到的 MQTT 用户名长度达到 <length> 后，返回：

```
OK
```

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<length>**：MQTT 用户名长度。范围：[1,1024]。

说明

- *AT+MQTTUSERCFG* 命令也可以设置 MQTT 用户名，二者之间的差别包括：
  - AT+MQTTLONGUSERNAME 命令可以用来设置相对较长的用户名，因为 AT+MQTTUSERCFG 命令的长度受限。
  - 应在设置 AT+MQTTUSERCFG 后再使用 AT+MQTTLONGUSERNAME。

3.6.4 AT+MQTTLONGPASSWORD：设置 MQTT 登陆密码

设置命令

功能：

设置 MQTT 密码

命令：

```
AT+MQTTLONGPASSWORD=<LinkID>,<length>
```

响应：

```
OK

>
```

上述响应表示 AT 已准备好接收 MQTT 密码，此时您可以输入 MQTT 密码，当 AT 接收到的 MQTT 密码长度达到 <length> 后，返回：

```
OK
```

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<length>**：MQTT 密码长度。范围：[1,1024]。

说明

- *AT+MQTTUSERCFG* 命令也可以设置 MQTT 密码，二者之间的差别包括：
  - AT+MQTTLONGPASSWORD 可以用来设置相对较长的密码，因为 AT+MQTTUSERCFG 命令的长度受限；
  - 应在设置 AT+MQTTUSERCFG 后再使用 AT+MQTTLONGPASSWORD。

3.6.5 AT+MQTTCONNCFG：设置 MQTT 连接属性

设置命令

功能：

设置 MQTT 连接属性

命令：

```
AT+MQTTCONNCFG=<LinkID>,<keepalive>,<disable_clean_session>,<"lwt_topic">,<"lwt_msg">,
↪<lwt_qos>,<lwt_retain>
```

响应：

```
OK
```

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<keepalive>**：MQTT ping 超时时间，单位：秒。范围：[0,7200]。默认值：0，会被强制改为 120 秒。
- **<disable_clean_session>**：设置 MQTT 清理会话标志，有关该参数的更多信息请参考 MQTT 3.1.1 协议中的 Clean Session 章节。
  - 0: 使能清理会话
  - 1: 禁用清理会话
- **<lwt_topic>**：遗嘱 topic，最大长度：128 字节。
- **<lwt_msg>**：遗嘱 message，最大长度：64 字节。

- **<lwt_qos>**：遗嘱 QoS，参数可选 0、1、2，默认值：0。
- **<lwt_retain>**：遗嘱 retain，参数可选 0 或 1，默认值：0。

3.6.6 AT+MQTTCONN：连接 MQTT Broker

查询命令

功能：

查询 ESP 设备已连接的 MQTT broker

命令：

```
AT+MQTTCONN?
```

响应：

```
+MQTTCONN:<LinkID>,<state>,<scheme><"host">,<port>,<"path">,<reconnect>
OK
```

设置命令

功能：

连接 MQTT Broker

命令：

```
AT+MQTTCONN=<LinkID>,<"host">,<port>,<reconnect>
```

响应：

```
OK
```

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<host>**：MQTT broker 域名，最大长度：128 字节。
- **<port>**：MQTT broker 端口，最大端口：65535。
- **<path>**：资源路径，最大长度：32 字节。
- **<reconnect>**：
  - 0: MQTT 不自动重连；

  - 1: MQTT 自动重连，会消耗较多的内存资源。

- **<state>**：MQTT 状态：
  - 0: MQTT 未初始化；
  - 1: 已设置 AT+MQTTUSERCFG；
  - 2: 已设置 AT+MQTTCONNCFG；
  - 3: 连接已断开；
  - 4: 已建立连接；
  - 5: 已连接，但未订阅 topic；
  - 6: 已连接，已订阅过 topic。
- **<scheme>**：
  - 1: MQTT over TCP；
  - 2: MQTT over TLS（不校验证书）；
  - 3: MQTT over TLS（校验 server 证书）；
  - 4: MQTT over TLS（提供 client 证书）；
  - 5: MQTT over TLS（校验 server 证书并且提供 client 证书）；
  - 6: MQTT over WebSocket（基于 TCP）；
  - 7: MQTT over WebSocket Secure（基于 TLS，不校验证书）；
  - 8: MQTT over WebSocket Secure（基于 TLS，校验 server 证书）；
  - 9: MQTT over WebSocket Secure（基于 TLS，提供 client 证书）；
  - 10: MQTT over WebSocket Secure（基于 TLS，校验 server 证书并且提供 client 证书）。

3.6.7 AT+MQTTPUB：发布 MQTT 消息（字符串）

设置命令

功能：

通过 topic 发布 MQTT **字符串**消息，若要发布 二**进制**消息，请使用*AT+MQTTPUBRAW* 命令。

命令：

```
AT+MQTTPUB=<LinkID>,<"topic">,<"data">,<qos>,<retain>
```

响应：

```
OK
```

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<topic>**：MQTT topic，最大长度：128 字节。
- **<data>**：MQTT 字符串消息。
- **<qos>**：发布消息的 QoS，参数可选 0、1、或 2，默认值：0。
- **<retain>**：发布 retain。

说明

- 每条 AT 命令的总长度不能超过 256 字节。
- 本命令不能发送数据 \0，若需要发送该数据，请使用*AT+MQTTPUBRAW* 命令。

3.6.8 AT+MQTTPUBRAW：发布 MQTT 消息（二进制）

设置命令

功能：

通过 topic 发布 MQTT 二**进制**消息

命令：

```
AT+MQTTPUBRAW=<LinkID>,<"topic">,<length>,<qos>,<retain>
```

响应：

```
OK
>
```

符号 > 表示 AT 准备好接收串口数据，此时您可以输入数据，当数据长度达到参数 <length> 的值时，数据传输开始。

若传输成功，则 AT 返回：

```
+MQTTPUB:OK
```

若传输失败，则 AT 返回：

```
+MQTTPUB:FAIL
```

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<topic>**：MQTT topic，最大长度：128 字节。
- **<length>**：MQTT 消息长度，不同 ESP 设备的最大长度不同：
  - 对于 ESP32 设备：最大长度受到可利用内存的限制；
  - 对于 ESP8266 设备，最大长度受到可利用内存和 `MQTT_BUFFER_SIZE_BYTE` 宏的限制。该宏的默认值为 `512`，可在 `build.py menuconfig` 中设置它的值，以此更改对最大长度的限制。该宏的值 = 消息的最大长度 + MQTT 报头长度（取决于 topic 名称的长度）。
- **<qos>**：发布消息的 QoS，参数可选 0、1、或 2，默认值：0。
- **<retain>**：发布 retain。

3.6.9 AT+MQTTSUB：订阅 MQTT Topic

查询命令

功能：

查询已订阅的 topic

命令：

```
AT+MQTTSUB?
```

响应：

```
+MQTTSUB:<LinkID>,<state>,<"topic1">,<qos>
+MQTTSUB:<LinkID>,<state>,<"topic2">,<qos>
+MQTTSUB:<LinkID>,<state>,<"topic3">,<qos>
...
OK
```

设置命令

功能：

订阅指定 MQTT topic 的指定 QoS，支持订阅多个 topic

命令：

```
AT+MQTTSUB=<LinkID>,<"topic">,<qos>
```

响应：

```
OK
```

当 AT 接收到已订阅的 topic 的 MQTT 消息时，返回：

```
+MQTTSUBRECV:<LinkID>,<"topic">,<data_length>,data
```

若已订阅过该 topic，则返回：

```
ALREADY SUBSCRIBE
```

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<state>**：MQTT 状态：
  - 0: MQTT 未初始化；
  - 1: 已设置 AT+MQTTUSERCFG；
  - 2: 已设置 AT+MQTTCONNCFG；
  - 3: 连接已断开；
  - 4: 已建立连接；
  - 5: 已连接，但未订阅 topic；
  - 6: 已连接，已订阅过 MQTT topic。
- **<topic>**：订阅的 topic。
- **<qos>**：订阅的 QoS。

3.6.10 AT+MQTTUNSUB：取消订阅 MQTT Topic

设置命令

功能：

客户端取消订阅指定 topic，可多次调用本命令，以取消订阅不同的 topic。

命令：

```
AT+MQTTUNSUB=<LinkID>,<"topic">
```

响应：

```
OK
```

若未订阅过该 topic，则返回：

```
NO UNSUBSCRIBE

OK
```

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<topic>**：MQTT topic，最大长度：128 字节。

3.6.11 AT+MQTTCLEAN：断开 MQTT 连接

设置命令

功能：

断开 MQTT 连接，释放资源。

命令：

```
AT+MQTTCLEAN=<LinkID>
```

响应：

```
OK
```

参数

- **<LinkID>**：当前仅支持 link ID 0。

3.6.12 MQTT AT 错误码

MQTT 错误码以 `ERR CODE:0x<%08x>` 形式打印。

错误类型	错误码
AT_MQTT_NO_CONFIGURED	0x6001
AT_MQTT_NOT_IN_CONFIGURED_STATE	0x6002
AT_MQTT_UNINITIATED_OR_ALREADY_CLEAN	0x6003
AT_MQTT_ALREADY_CONNECTED	0x6004
AT_MQTT_MALLOC_FAILED	0x6005
AT_MQTT_NULL_LINK	0x6006
AT_MQTT_NULL_PARAMTER	0x6007
AT_MQTT_PARAMETER_COUNTS_IS_WRONG	0x6008
AT_MQTT_TLS_CONFIG_ERROR	0x6009
AT_MQTT_PARAM_PREPARE_ERROR	0x600A
AT_MQTT_CLIENT_START_FAILED	0x600B
AT_MQTT_CLIENT_PUBLISH_FAILED	0x600C
AT_MQTT_CLIENT_SUBSCRIBE_FAILED	0x600D
AT_MQTT_CLIENT_UNSUBSCRIBE_FAILED	0x600E
AT_MQTT_CLIENT_DISCONNECT_FAILED	0x600F
AT_MQTT_LINK_ID_READ_FAILED	0x6010
AT_MQTT_LINK_ID_VALUE_IS_WRONG	0x6011
AT_MQTT_SCHEME_READ_FAILED	0x6012
AT_MQTT_SCHEME_VALUE_IS_WRONG	0x6013
AT_MQTT_CLIENT_ID_READ_FAILED	0x6014
AT_MQTT_CLIENT_ID_IS_NULL	0x6015
AT_MQTT_CLIENT_ID_IS_OVERLENGTH	0x6016
AT_MQTT_USERNAME_READ_FAILED	0x6017
AT_MQTT_USERNAME_IS_NULL	0x6018
AT_MQTT_USERNAME_IS_OVERLENGTH	0x6019
AT_MQTT_PASSWORD_READ_FAILED	0x601A
AT_MQTT_PASSWORD_IS_NULL	0x601B
AT_MQTT_PASSWORD_IS_OVERLENGTH	0x601C
AT_MQTT_CERT_KEY_ID_READ_FAILED	0x601D
AT_MQTT_CERT_KEY_ID_VALUE_IS_WRONG	0x601E

下页继续

表 2 – 续上页

错误类型	错误码
AT_MQTT_CA_ID_READ_FAILED	0x601F
AT_MQTT_CA_ID_VALUE_IS_WRONG	0x6020
AT_MQTT_CA_LENGTH_ERROR	0x6021
AT_MQTT_CA_READ_FAILED	0x6022
AT_MQTT_CERT_LENGTH_ERROR	0x6023
AT_MQTT_CERT_READ_FAILED	0x6024
AT_MQTT_KEY_LENGTH_ERROR	0x6025
AT_MQTT_KEY_READ_FAILED	0x6026
AT_MQTT_PATH_READ_FAILED	0x6027
AT_MQTT_PATH_IS_NULL	0x6028
AT_MQTT_PATH_IS_OVERLENGTH	0x6029
AT_MQTT_VERSION_READ_FAILED	0x602A
AT_MQTT_KEEPALIVE_READ_FAILED	0x602B
AT_MQTT_KEEPALIVE_IS_NULL	0x602C
AT_MQTT_KEEPALIVE_VALUE_IS_WRONG	0x602D
AT_MQTT_DISABLE_CLEAN_SESSION_READ_FAILED	0x602E
AT_MQTT_DISABLE_CLEAN_SESSION_VALUE_IS_WRONG	0x602F
AT_MQTT_LWT_TOPIC_READ_FAILED	0x6030
AT_MQTT_LWT_TOPIC_IS_NULL	0x6031
AT_MQTT_LWT_TOPIC_IS_OVERLENGTH	0x6032
AT_MQTT_LWT_MSG_READ_FAILED	0x6033
AT_MQTT_LWT_MSG_IS_NULL	0x6034
AT_MQTT_LWT_MSG_IS_OVERLENGTH	0x6035
AT_MQTT_LWT_QOS_READ_FAILED	0x6036
AT_MQTT_LWT_QOS_VALUE_IS_WRONG	0x6037
AT_MQTT_LWT_RETAIN_READ_FAILED	0x6038
AT_MQTT_LWT_RETAIN_VALUE_IS_WRONG	0x6039
AT_MQTT_HOST_READ_FAILED	0x603A
AT_MQTT_HOST_IS_NULL	0x603B
AT_MQTT_HOST_IS_OVERLENGTH	0x603C
AT_MQTT_PORT_READ_FAILED	0x603D
AT_MQTT_PORT_VALUE_IS_WRONG	0x603E
AT_MQTT_RECONNECT_READ_FAILED	0x603F
AT_MQTT_RECONNECT_VALUE_IS_WRONG	0x6040
AT_MQTT_TOPIC_READ_FAILED	0x6041
AT_MQTT_TOPIC_IS_NULL	0x6042
AT_MQTT_TOPIC_IS_OVERLENGTH	0x6043

下页继续

表 2 - 续上页

错误类型	错误码
AT_MQTT_DATA_READ_FAILED	0x6044
AT_MQTT_DATA_IS_NULL	0x6045
AT_MQTT_DATA_IS_OVERLENGTH	0x6046
AT_MQTT_QOS_READ_FAILED	0x6047
AT_MQTT_QOS_VALUE_IS_WRONG	0x6048
AT_MQTT_RETAIN_READ_FAILED	0x6049
AT_MQTT_RETAIN_VALUE_IS_WRONG	0x604A
AT_MQTT_PUBLISH_LENGTH_READ_FAILED	0x604B
AT_MQTT_PUBLISH_LENGTH_VALUE_IS_WRONG	0x604C
AT_MQTT_RECV_LENGTH_IS_WRONG	0x604D
AT_MQTT_CREATE_SEMA_FAILED	0x604E
AT_MQTT_CREATE_EVENT_GROUP_FAILED	0x604F
AT_MQTT_URI_PARSE_FAILED	0x6050
AT_MQTT_IN_DISCONNECTED_STATE	0x6051
AT_MQTT_HOSTNAME_VERIFY_FAILED	0x6052

3.6.13 MQTT AT 说明

- 一般来说，AT MQTT 命令都会在 10 秒内响应，但 `AT+MQTTCONN` 命令除外。例如，如果路由器不能上网，命令 `AT+MQTTPUB` 会在 10 秒内响应，但 `AT+MQTTCONN` 命令在网络环境不好的情况下，可能需要更多的时间用来重传数据包。
- 如果 `AT+MQTTCONN` 是基于 TLS 连接，每个数据包的超时时间为 10 秒，则总超时时间会根据握手数据包的数量而变得更长。
- 当 MQTT 连接断开时，会提示 `+MQTTDISCONNECTED:<LinkID>` 消息。
- 当 MQTT 连接建立时，会提示 `+MQTTCONNECTED:<LinkID>,<scheme>,<"host">,port,<"path">,<reconnect>` 消息。

3.7 HTTP AT 命令集

[English]

- *AT+HTTPCLIENT*：发送 HTTP 客户端请求
- *AT+HTTPGETSIZE*：获取 HTTP 资源大小
- *AT+HTTPCPOST*：Post 指定长度的 HTTP 数据
- *HTTP AT 错误码*

3.7.1 AT+HTTPCLIENT：发送 HTTP 客户端请求

设置命令

命令：

```
AT+HTTPCLIENT=<opt>,<content-type>,<"url">,[<"host">],[<"path">],<transport_type>[,<
↪"data">][,<"http_req_header">][,<"http_req_header">][...]
```

响应：

```
+HTTPCLIENT:<size>,<data>

OK
```

参数

- **<opt>**：HTTP 客户端请求方法：
  - 1: HEAD
  - 2: GET
  - 3: POST
  - 4: PUT
  - 5: DELETE
- **<content-type>**：客户端请求数据类型：
  - 0: `application/x-www-form-urlencoded`
  - 1: `application/json`
  - 2: `multipart/form-data`
  - 3: `text/xml`
- **<" url" >**：HTTP URL，当后面的 `<host>` 和 `<path>` 参数为空时，本参数会自动覆盖这两个参数。
- **<" host" >**：域名或 IP 地址。
- **<" path" >**：HTTP 路径。
- **<transport_type>**：HTTP 客户端传输类型，默认值为 1：
  - 1: `HTTP_TRANSPORT_OVER_TCP`
  - 2: `HTTP_TRANSPORT_OVER_SSL`
- **<" data" >**: 当 `<opt>` 是 POST 请求时，本参数为发送给 HTTP 服务器的数据。当 `<opt>` 不是 POST 请求时，这个参数不存在（也就是，不需要输入逗号来表示有这个参数）。

- **<" http_req_header" >**: 可发送多个请求头给服务器。

说明

- 如果 `url` 参数不为空，HTTP 客户端将使用它并忽略 `host` 参数和 `path` 参数；如果 `url` 参数被省略或字符串为空，HTTP 客户端将使用 `host` 参数和 `path` 参数。
- 某些已发布的固件默认不支持 HTTP 客户端命令（详情请见*如何了解 ESP-AT 同一平台不同模组差异*），但是可通过以下方式使其支持该命令：`./build.py menuconfig > Component config > AT > AT http command support`，然后编译项目（详情请见*Build Your Own ESP-AT Project*）。
- 在 ESP8266 平台上，如果 URL 是 HTTPS 类型或 `<transport_type>` 参数是 2，需要将 at_process_task 任务堆栈增加到 4096 以上。即配置：*./build.py menuconfig -> Component config -> AT -> The stack size of the AT process task*，否则会由于堆栈不足而导致重启。

示例

```
// HEAD 请求
AT+HTTPCLIENT=1,0,"http://httpbin.org/get","httpbin.org","/get",1

// GET 请求
AT+HTTPCLIENT=2,0,"http://httpbin.org/get","httpbin.org","/get",1

// POST 请求
AT+HTTPCLIENT=3,0,"http://httpbin.org/post","httpbin.org","/post",1,"field1=value1&
↪field2=value2"
```

3.7.2 AT+HTTPGETSIZE：获取 HTTP 资源大小

设置命令

命令：

```
AT+HTTPGETSIZE=<url>
```

响应：

```
+HTTPGETSIZE:<size>

OK
```

参数

- **<url>**：HTTP URL。
- **<size>**：HTTP 资源大小。

说明

- 某些已发布的固件默认不支持 HTTP 客户端命令（详情请见*如何了解 ESP-AT 同一平台不同模组差异*），但是可通过以下方式使其支持该命令：`./build.py menuconfig > Component config > AT > AT http command support`，然后编译项目（详情请见*Build Your Own ESP-AT Project*）。
- 在 ESP8266 平台上，如果 URL 是 HTTPS 类型或 `<transport_type>` 参数是 2，需要将 at_process_task 任务堆栈增加到 4096 以上。即配置：*./build.py menuconfig -> Component config -> AT -> The stack size of the AT process task*，否则会由于堆栈不足而导致重启。

示例

```
AT+HTTPGETSIZE="http://www.baidu.com/img/bdlogo.gif"
```

3.7.3 AT+HTTPCPOST：Post 指定长度的 HTTP 数据

设置命令

命令：

```
AT+HTTPCPOST=<url>,<length>[,<http_req_header_cnt>][,<http_req_header>..<http_req_
↪header>]
```

响应：

```
OK
>
```

符号 > 表示 AT 准备好接收串口数据，此时您可以输入数据，当数据长度达到参数 `<length>` 的值时，传输开始。

若传输成功，则返回：

```
SEND OK
```

若传输失败，则返回：

```
SEND FAIL
```

参数

- **<url>**：HTTP URL。
- **<length>**：需 POST 的 HTTP 数据长度。最大长度等于系统可分配的堆空间大小。
- **<http_req_header_cnt>**：<http_req_header> 参数的数量。
- **[<http_req_header>]**：可发送多个请求头给服务器。

说明

- 在 ESP8266 平台上，如果 URL 是 HTTPS 类型或 <transport_type> 参数是 2，需要将 at_process_task 任务堆栈增加到 4096 以上。即配置：*./build.py menuconfig -> Component config -> AT -> The stack size of the AT process task*，否则会由于堆栈不足而导致重启。

3.7.4 HTTP AT 错误码

- HTTP 客户端：

HTTP 客户端错误码	说明
0x7190	Bad Request
0x7191	Unauthorized
0x7192	Payment Required
0x7193	Forbidden
0x7194	Not Found
0x7195	Method Not Allowed
0x7196	Not Acceptable
0x7197	Proxy Authentication Required
0x7198	Request Timeout
0x7199	Conflict
0x719a	Gone
0x719b	Length Required
0x719c	Precondition Failed
0x719d	Request Entity Too Large
0x719e	Request-URI Too Long
0x719f	Unsupported Media Type
0x71a0	Requested Range Not Satisfiable
0x71a1	Expectation Failed

- HTTP 服务器：

HTTP 服务器错误码	说明
0x71f4	Internal Server Error
0x71f5	Not Implemented
0x71f6	Bad Gateway
0x71f7	Service Unavailable
0x71f8	Gateway Timeout
0x71f9	HTTP Version Not Supported

- HTTP AT：
  - `AT+HTTPCLIENT` 命令的错误码为 `0x7000+Standard HTTP Error Code`（更多有关 Standard HTTP/1.1 Error Code 的信息，请参考 RFC 2616）。
  - 例如，若 AT 在调用 `AT+HTTPCLIENT` 命令时收到 HTTP error 404，则会返回 `0x7194` 错误码 (`hex(0x7000+404)=0x7194`)。

3.8 [ESP32 Only] 以太网 AT 命令

[English]

- 准备工作
- [ESP32 Only] *AT+CIPETHMAC*：查询/设置 ESP 以太网的 MAC 地址
- [ESP32 Only] *AT+CIPETH*：查询/设置 ESP 以太网的 IP 地址

3.8.1 准备工作

运行以太网 AT 命令之前，请做好以下准备工作：

> **注意：** 本节内容以 ESP32-Ethernet-Kit 开发板为例介绍运行以太网 AT 命令前的准备工作。如果您使用的是其它模组或开发板，请查阅对应的技术规格书获取 RX/TX 管脚号。

- 修改 AT UART 管脚（因为默认的 AT UART 管脚和以太网功能管脚冲突）：
  - 打开 factory_param_data.csv 表格文件；
  - 将 `WROVER-32` 的 `uart_tx_pin` 从 GPIO22 改为 GPIO2，`uart_rx_pin` 从 GPIO19 改为 GPIO4，`uart_cts_pin` 从 GPIO15 改为 GPIO1，`uart_rts_pin` 从 GPIO14 改为 GPIO1 （硬件流控功能可选，这里未使用该功能），更多信息请见*如何修改 AT port 管脚*。
- 使能 `AT ethernet support`，更多信息请见*How to enable ESP-AT Ethernet*。

- 编译后将该工程烧录至 ESP32-Ethernet-Kit。
- 连接硬件：
  - 连接主机 MCU（如 PC，可使用 USB 转串口模块）至 ESP32-Ethernet-Kit 的 GPIO2 (TX) 和 GPIO4 (RX)，不使用流控功能则无需连接 CTS/RTS；
  - ESP32-Ethernet-Kit 连接以太网网络。

3.8.2 [ESP32 Only] AT+CIPETHMAC：查询/设置 ESP 以太网的 MAC 地址

查询命令

功能：

查询 ESP 以太网的 MAC 地址

命令：

```
AT+CIPETHMAC?
```

响应：

```
+CIPETHMAC:<"mac">
OK
```

设置命令

功能：

设置 ESP 以太网的 MAC 地址

命令：

```
AT+CIPETHMAC=<"mac">
```

响应：

```
OK
```

参数

- **<" mac" >**：字符串参数，表示以太网接口的 MAC 地址。

说明

- 固件默认不支持以太网 AT 命令 (详情请见*如何了解 ESP-AT 同一平台不同模组差异*)，但是可通过以下方式使其支持该命令：`./build.py menuconfig > Component config > AT > AT ethernet support`，然后编译工程（详情请见*Build Your Own ESP-AT Project*）。
- 若*AT+SYSSTORE=1*，配置更改将保存在 NVS 区。
- 以太网接口的 MAC 地址不能与其他接口的相同。
- ESP MAC 地址的 bit0 不能设为 1。例如，可设为“1a:…”，但不可设为“15:…”。
- `FF:FF:FF:FF:FF:FF` 和 `00:00:00:00:00:00` 为无效 MAC 地址，不能设置。

示例

```
AT+CIPETHMAC="1a:fe:35:98:d4:7b"
```

3.8.3 [ESP32 Only] AT+CIPETH：查询/设置 ESP 以太网的 IP 地址

查询命令

功能：

查询 ESP 以太网的 IP 地址

命令：

```
AT+CIPETH?
```

响应：

```
+CIPETH:ip:<ip>
+CIPETH:gateway:<gateway>
+CIPETH:netmask:<netmask>
OK
```

设置命令

功能：

设置 ESP 以太网的 IP 地址

命令：

```
AT+CIPETH=<ip>[,<gateway>,<netmask>]
```

响应：

```
OK
```

参数

- **<ip>**：字符串参数，表示 ESP 以太网的 IP 地址。
- **[<gateway>]**：网关。
- **[<netmask>]**：网络掩码。

说明

- 固件默认不支持以太网 AT 命令 (详情请见*如何了解 ESP-AT 同一平台不同模组差异*)，但是可通过以下方式使其支持该命令：`./build.py menuconfig > Component config > AT > AT ethernet support`，然后编译工程（详情请见*Build Your Own ESP-AT Project*）。
- 若*AT+SYSSTORE=1*，配置更改将保存在 NVS 区。
- 本命令的设置命令与 DHCP 相互影响，如*AT+CWDHCP*：
  - 若启用静态 IP，则 DHCP 会被禁用；
  - 若启用 DHCP，则静态 IP 会被禁用；
  - 最后一次配置会覆盖上一次配置。

示例

```
AT+CIPETH="192.168.6.100","192.168.6.1","255.255.255.0"
```

3.9 [ESP8266 Only] 信令测试 AT 命令

[English]

- *AT+FACTPLCP*：发送长 PLCP 或短 PLCP

3.9.1 [ESP8266 Only] AT+FACTPLCP：发送长 PLCP 或短 PLCP

设置命令

命令：

```
AT+FACTPLCP=<enable>,<tx_with_long>
```

响应：

```
OK
```

参数

- **<enable>**：启用/禁用手动配置：
  - 0: 禁用手动配置，将使用 <tx_with_long> 参数的默认值；
  - 1: 启用手动配置，AT 发送的 PLCP 类型取决于 <tx_with_long> 参数。
- **<tx_with_long>**：发送长 PLCP 或短 PLCP：
  - 0: 发送短 PLCP（默认）；
  - 1: 发送长 PLCP。

3.10 [ESP32 & ESP32-S2 & ESP32-C3] 驱动 AT 命令

[English]

- *AT+DRVADC*：读取 ADC 通道值
- *AT+DRVPWMINIT*：初始化 PWM 驱动器
- *AT+DRVPWMDUTY*：设置 PWM 占空比
- *AT+DRVPWMFADE*：设置 PWM 渐变
- *AT+DRVI2CINIT*：初始化 I2C 主机驱动
- *AT+DRVI2CRD*：读取 I2C 数据

- *AT+DRVI2CWRDATA*：写入 I2C 数据
- *AT+DRVI2CWRBYTES*：写入不超过 4 字节的 I2C 数据
- *AT+DRVSPICONFGPIO*：配置 SPI GPIO
- *AT+DRVSPIINIT*：初始化 SPI 主机驱动
- *AT+DRVSPIRD*：读取 SPI 数据
- *AT+DRVSPIWR*：写入 SPI 数据

3.10.1 AT+DRVADC：读取 ADC 通道值

设置命令

命令：

```
AT+DRVADC=<channel>,<atten>
```

响应：

```
+DRVADC:<raw data>
OK
```

参数

- **<channel>**：ADC1 通道，范围：0 ~ 7，相应管脚请查看技术规格书。
- **<atten>**：衰减值：
  - 0: 0 dB 衰减，满量程电压为 1.1 V；
  - 1: 2.5 dB 衰减，满量程电压 1.5 V；
  - 2: 6 dB 衰减，满量程电压为 2.2 V；
  - 3: 11 dB 衰减，满量程电压为 3.9 V。
- **<raw data>**：ADC 通道值，电压值为 raw_data/2 ^{width} * atten。

说明

- ESP-AT 只支持 ADC1。
- ESP32 和 ESP32-C3 支持 12 位宽度，ESP32-S2 只支持 13 位宽度。

示例

```
AT+DRVADC=0,0    // ADC1 0 通道，电压：0 ~ 1.1 V
+DRVADC:2048     // ESP32 和 ESP32-C3 的电压为 2048 / 4096 * 1.1 = 0.55
                 // ESP32-S2 的电压为 2048 / 8192 * 1.1 = 0.264
OK
```

3.10.2 AT+DRVPWMINIT：初始化 PWM 驱动器

设置命令

命令：

```
AT+DRVPWMINIT=<freq>,<duty_res>,<ch0_gpio>[,...,<ch3_gpio>]
```

响应：

```
OK
```

参数

- **<freq>**：LEDC 定时器频率，单位：Hz，范围：1 Hz ~ 8 MHz。
- **<duty_res>**：LEDC 通道占空比分辨率，范围：0 ~ 20 位。
- **<chx_gpio>**：LEDC 通道 x 的输出 GPIO。例如，如果您想将 GPIO16 作为通道 0，需设置 <ch0_gpio> 为 16。

说明

- ESP-AT 最多能支持 4 个通道。
- 使用本命令初始化的通道数量直接决定了其它 PWM 命令（如*AT+DRVPWMDUTY* 和*AT+DRVPWMFADE*）能够设置的通道。例如，如果您只初始化了两个通道，那么 AT+DRVPWMDUTY 命令只能用来更改这两个通道的 PWM 占空比。
- 频率和占空比分辨率相互影响。更多信息请见 频率和占空比分辨率支持范围。

示例

```
AT+DRVPWMINIT=5000,13,17,16,18,19  // 设置 4 个通道，频率为 5 kHz，占空比分辨率为 13 位
AT+DRVPWMINIT=10000,10,17          // 只初始化通道 0，频率为 10 kHz，占空比分辨率为 10 位，其
它 PWM 相关命令只能设置一个通道
```

3.10.3 AT+DRVPWMDUTY：设置 PWM 占空比

设置命令

命令：

```
AT+DRVPWMDUTY=<ch0_duty>[,...,<ch3_duty>]
```

响应：

```
OK
```

参数

- **<duty>**：LEDC 通道占空比，范围：[0,2 占空比分辨率]。

说明

- ESP-AT 最多能支持 4 个通道。
- 若某个通道无需设置占空比，直接省略该参数。

示例

```
AT+DRVPWMDUTY=255,512    // 设置通道 0 的占空比为 255，设置通道 1 的占空比为 512
AT+DRVPWMDUTY=,,0        // 只设置通道 2 的占空比为 0
```

3.10.4 AT+DRVPWMFADE：设置 PWM 渐变

设置命令

命令：

```
AT+DRVPWMFADE=<ch0_target_duty>,<ch0_fade_time>[,...,<ch3_target_duty>,<ch3_fade_time>
↪]
```

响应：

```
OK
```

参数

- **<target_duty>**：目标渐变占空比，范围：[0,2 $^{duty\_resolution}$−1]。
- **<fade_time>**：渐变的最长时间，单位：毫秒。

说明

- ESP-AT 最多能支持 4 个通道。
- 若某个通道无需设置 <target_duty> 和 <fade_time>，直接省略即可。

示例

```
AT+DRVPWMFADE=,,0,1000              // 使用一秒的时间将通道 1 的占空比设置为 0
AT+DRVPWMFADE=1024,1000,0,2000,  // 使用一秒的时间将通道 0 的占空比设置为 1024、两秒的时间将通道 1 的占空比设为 0
```

3.10.5 AT+DRVI2CINIT：初始化 I2C 主机驱动

设置命令

命令：

```
AT+DRVI2CINIT=<num>,<scl_io>,<sda_io>,<clock>
```

响应：

```
OK
```

参数

- **<num>**：I2C 端口号，范围：0 ~ 1。如果未设置后面的参数，AT 将不初始化该 I2C 端口。
- **<scl_io>**：I2C SCL 信号的 GPIO 号。
- **<sda_io>**：I2C SDA 信号的 GPIO 号。
- **<clock>**：主机模式下的 I2C 时钟频率，单位：Hz，最大值：1 MHz。

说明

- 本指令只支持 I2C 主机。

示例

```
AT+DRVI2CINIT=0,25,26,1000  // 初始化 I2C0，SCL：GPIO25，SDA：GPIO26，I2C 时钟频率：1 kHz
AT+DRVI2CINIT=0             // 取消 I2C0 初始化
```

3.10.6 AT+DRVI2CRD：读取 I2C 数据

设置命令

命令：

```
AT+DRVI2CRD=<num>,<address>,<length>
```

响应：

```
+DRVI2CRD:<read data>
OK
```

参数

- **<num>**：I2C 端口号，范围：0 ~ 1。
- **<address>**：I2C 从机设备地址：
  - 7 位地址：0 ~ 0x7F；
  - 10 位地址：第一个字节的前七个位是 1111 0XX，其中最后两位 XX 是 10 位地址的最高两位。例如，如果 10 位地址为 0x2FF (b’1011111111)，那么输入的地址为 0x7AFF (b’111101011111111)。
- **<length>**：I2C 数据长度，范围：1 ~ 2048。
- **<read data>**：I2C 数据。

说明

- I2C 传输超时时间为一秒。

示例

```
AT+DRVI2CRD=0,0x34,1      // I2C0 从地址 0x34 处读取 1 字节的数据
AT+DRVI2CRD=0,0x7AFF,1    // I2C0 从 10 位地址 0x2FF 处读取 1 字节的数据

// I2C0 读地址 0x34，寄存器地址 0x27，读 2 字节
AT+DRVI2CWRBYTES=0,0x34,1,0x27      // I2C0 先写设备地址 0x34、寄存器地址 0x27
AT+DRVI2CRD=0,0x34,2                // I2C0 读地址 2 字节
```

3.10.7 AT+DRVI2CWRDATA：写入 I2C 数据

设置命令

命令：

```
AT+DRVI2CWRDATA=<num>,<address>,<length>
```

响应：

```
OK
>
```

收到上述响应后，请输入您想写入的数据，当数据达到参数指定长度后，数据传输开始。

若数据传输成功，则返回：

```
OK
```

若数据传输失败，则返回：

```
ERROR
```

参数

- **<num>**：I2C 端口号，范围：0 ~ 1。
- **<address>**：I2C 从机设备地址：
  - 7 位地址：0 ~ 0x7F；
  - 10 位地址：第一个字节的前七个位是 1111 0XX，其中最后两位 XX 是 10 位地址的最高两位。例如，如果 10 位地址为 0x2FF (b’1011111111)，那么输入的地址为 0x7AFF (b’111101011111111)。
- **<length>**：I2C 数据长度，范围：1 ~ 2048。

说明

- I2C 传输超时时间为一秒。

示例

```
AT+DRVI2CWRDATA=0,0x34,10    // I2C0 写入 10 字节数据至地址 0x34
```

3.10.8 AT+DRVI2CWRBYTES：写入不超过 4 字节的 I2C 数据

设置命令

命令：

```
AT+DRVI2CWRBYTES=<num>,<address>,<length>,<data>
```

响应：

```
OK
```

参数

- **<num>**：I2C 端口号，范围：0 ~ 1。
- **<address>**：I2C 从机设备地址。
  - 7 位地址：0 ~ 0x7F。
  - 10 位地址：第一个字节的前七个位是 1111 0XX，其中最后两位 XX 是 10 位地址的最高两位。例如，如果 10 位地址为 0x2FF (b’1011111111)，那么输入的地址为 0x7AFF (b’111101011111111)。
- **<length>**：待写入的 I2C 数据长度，范围：1 ~ 4 字节。

- **<data>**：参数 <length> 指定长度的数据，范围：0 ~ 0xFFFFFFFF。

说明

- I2C 传输超时时间为一秒。

示例

```
AT+DRVI2CWRBYTES=0,0x34,2,0x1234      // I2C0 写入 2 字节数据 0x1234 至地址 0x34
AT+DRVI2CWRBYTES=0,0x7AFF,2,0x1234    // I2C0 写入 2 字节数据 0x1234 至 10 位地址 0x2FF

// I2C0 写地址 0x34、寄存器地址 0x27，数据为 c0xFF
AT+DRVI2CWRBYTES=0,0x34,2,0x27FF
```

3.10.9 AT+DRVSPICONFGPIO：配置 SPI GPIO

设置命令

命令：

```
AT+DRVSPICONFGPIO=<mosi>,<miso>,<sclk>,<cs>
```

响应：

```
OK
```

参数

- **<mosi>**：主出从入信号对应的 GPIO 管脚。
- **<miso>**：主入从出信号对应 GPIO 管脚，若不使用，置位 -1。
- **<sclk>**：SPI 时钟信号对应的 GPIO 管脚。
- **<cs>**：选择从机的信号对应 GPIO 管脚，若不使用，置位 -1。

3.10.10 AT+DRVSPIINIT：初始化 SPI 主机驱动

设置命令

命令：

```
AT+DRVSPIINIT=<clock>,<mode>,<cmd_bit>,<addr_bit>,<dma_chan>[,bits_msb]
```

响应：

```
OK
```

参数

- **<clock>**：时钟速度，分频数为 80 MHz，单位：Hz，最大值：40 MHz。
- **<mode>**：SPI 模式，范围：0 ~ 3。
- **<cmd_bit>**：命令阶段的默认位数，范围：0 ~ 16。
- **<addr_bit>**：地址阶段的默认位数，范围：0 ~ 64。
- **<dma_chan>**：通道 1 或 2，不需要 DMA 时也可为 0。
- **<bits_msb>**：SPI 数据格式：
  - bit0:
    - 0: 先传输 MSB（默认）；
    - 1: 先传输 LSB。
  - bit1:
    - 0: 先接收 MSB（默认）；
    - 1: 先接收 LSB。

说明

- 请在 SPI 初始化前配置 SPI GPIO。

示例

```
AT+DRVSPIINIT=102400,0,0,0,0,3 // SPI 时钟：100 kHz；模式：0；命令阶段和地址阶段默认位数均为 0；
不使用 DMA；先传输和接收 LSB
OK
AT+DRVSPIINIT=0   // 删除 SPI 驱动
OK
```

3.10.11 AT+DRVSPIRD：读取 SPI 数据

设置命令

命令：

```
AT+DRVSPIRD=<data_len>[,<cmd>,<cmd_len>][,<addr>,<addr_len>]
```

响应：

```
+DRVSPIRD:<read data>
OK
```

参数

- **<data_len>**：待读取的 SPI 数据长度，范围：1 ~ 4092 字节。
- **<cmd>**：命令数据，数据长度由 <cmd_len> 参数设定。
- **<cmd_len>**：本次传输中的命令长度，范围：0 ~ 2 字节。
- **<addr>**：命令地址，地址长度由 <addr_len> 参数设定。
- **<addr_len>**：本次传输中地址长度，范围：0 ~ 4 字节。

说明

- 若不使用 DMA，<data_len> 参数每次能够设定的最大值为 64 字节。

示例

```
AT+DRVSPIRD=2  // 读取 2 字节数据
+DRVI2CREAD:ffff
OK

AT+DRVSPIRD=2,0x03,1,0x001000,3  // 读取 2 字节数据，<cmd> 为 0x03，<cmd_len> 为 1 字节，
↪<addr> 为 0x1000，<addr_len> 为 3 字节
+DRVI2CREAD:ffff
OK
```

3.10.12 AT+DRVSPIWR：写入 SPI 数据

设置命令

命令：

```
AT+DRVSPIWR=<data_len>[,<cmd>,<cmd_len>][,<addr>,<addr_len>]
```

响应：

当 <data_len> 参数值大于 0，AT 返回：

```
OK
>
```

收到上述响应后，请输入您想写入的数据，当数据达到参数指定长度后，数据传输开始。

若数据传输成功，AT 返回：

```
OK
```

当 <data_len> 参数值为 0 时，也即 AT 只传输命令和地址，不传输 SPI 数据，此时 AT 返回：

```
OK
```

参数

- **<data_len>**：SPI 数据长度，范围：0 ~ 4092。
- **<cmd>**：命令数据，数据长度由 <cmd_len> 参数设定。
- **<cmd_len>**：本次传输中的命令长度，范围：0 ~ 2 字节。
- **<addr>**：命令地址，地址长度由 <addr_len> 参数设定。

- **<addr_len>**：本次传输中地址长度，范围：0 ~ 4 字节。

说明

- 若不使用 DMA，<data_len> 参数每次能够设定的最大值为 64 字节。

示例

```
AT+DRVSPIWR=2  // 写入 2 字节数据
OK
>              // 开始接收串行数据
OK

AT+DRVSPIWR=0,0x03,1,0x001000,3  // 写入 0 字节数据，<cmd> 为 0x03，<cmd_len> 为 1 字节，
↪<addr> 为 0x1000，<addr_len> 为 3 字节
OK
```

3.11 Web server AT Commands

- *AT+WEBSERVER*: Enable/disable Wi-Fi connection configuration via web server.

3.11.1 AT+WEBSERVER: Enable/disable Wi-Fi connection configuration via web server

Set Command

Command:

```
AT+WEBSERVER=<enable>,<server_port>,<connection_timeout>
```

Response:

```
OK
```

Parameters

- **<enable>**: Enable or disable web server.
  - 0: Disable the web server and release related resources.
  - 1: Enable web server, which means that you can use WeChat or a browser to configure Wi-Fi connection information.
- **<server_port>**: The web server port number.
- **<connection_timeout>**: The timeout for the every connection. Unit: second. Range:[21,60].

Notes

- There are two ways to provide the HTML files needed by the web server. One is to use fatfs file system (non ESP8266 chips), and you need to enable AT FS command at this time. The other one is to use embedded files to storge HTML files (default setting).
- Please make sure that the max number of open sockets is not less than 12, you may change the number by `./build.py menuconfig` > `Component config` > `LWIP` > `Max bumber of open sockets` and compile the project (see *Build Your Own ESP-AT Project*).
- The default firmware does not support web server AT commands (see *如何了解 ESP-AT 同一平台不同模组差异*), but you can enable it by `./build.py menuconfig` > `Component config` > `AT` > `AT WEB Server command support` and compile the project (see *Build Your Own ESP-AT Project*).
- For ESP8266 devices, you may need to turn off some unnecessary options to store the necessary html files.
- For more examples, please refer to *Web Server AT 示例*.

Example

```
// Enable the web server with port 80, and the timeout for the every connection is 50 seconds:
AT+WEBSERVER=1,80,50

// Disable the web server, the command should be:
AT+WEBSERVER=0
```

3.12 ESP-AT 不同版本命令集支持对比

[English]

本文档主要列举了旧版 NONOS-AT 向新版 ESP-AT 迁移时，需要注意的命令上的差异。

- NONOS-AT: **不推荐**使用的 ESP8266 AT 版本，早期用于 ESP8266 系列芯片，基于 ESP8266_NONOS_SDK 开发，已停止更新。
- ESP-AT: **推荐**使用的版本，支持多个芯片平台，与 NONOS-AT 相比，支持更多种类的命令，如 Bluetooth 命令、Bluetooth LE 命令、以太网命令、驱动命令等。更多有关 ESP-AT 的信息请参阅*ESP-AT 是什么*和*AT 固件*。

注意： 下表列出了所有的 NONOS-AT 命令，以及它们在 ESP-AT 中的支持情况。下表并不包含所有的 ESP-AT 命令，若想了解所有 ESP-AT 命令，请参考*AT 命令集*。

下表中第一列的命令链接均指向 ESP-AT 命令。

表 3: ESP-AT 不同版本命令集支持对比表

命令	简介	NONOS-AT	ESP-AT
AT	测试 AT 启动	✓	✓
AT+RST	重启模块	✓	✓
AT+GMR	查询版本信息	✓	✓
AT+GSLP	进入 deep-sleep 模式	✓	✓
ATE	开关回显功能	✓	✓
AT+RESTORE	恢复出厂设置	✓	✓
AT+UART_CUR	配置 UART，不保存到 flash	✓	✓
AT+UART_DEF	配置 UART，保存到 flash	✓	✓
AT+SLEEP	设置 sleep 模式	✓ 1: light sleep 2: modem sleep	ESP8266 ✓ ESP32 ✓ ESP32-S2 ✗ [1]
AT+WAKEUPGPIO	配置 GPIO 唤醒 light sleep	✓	✗ [3] 相似命令参考*AT+SLEEPWKCFG*

下页继续

表 3 – 续上页

命令	简介	NONOS-AT	ESP-AT
AT+RFPOWER	设置 RF TX Power	✓ 范围：[0,82] 单位：0.25 dBm	✓ [2]
AT+RFVDD	根据 VDD33 设置 RF TX Power	✓	✗
AT+SYSRAM	查询系统当前剩余内存	✓	✓ 新增最小峰值内存参数
AT+SYSADC	查询 ADC 值	✓	✗
AT+SYSIOSETCFG	设置 IO 工作模式	✓	✗
AT+SYSIOGETCFG	查询 IO 工作模式	✓	✗
AT+SYSGPIODIR	设置 GPIO 工作为输入或输出	✓	✗
AT+SYSGPIOWRITE	设置 GPIO 的输出电平	✓	✗
AT+SYSGPIOREAD	读取 GPIO 的电平状态	✓	✗
AT+SYSMSG	设置系统消息	✗	✓
AT+SYSMSG_CUR	设置当前系统消息，不保存到 flash	✓	✗ [3] 相似命令参考*AT+SYSMSG*
AT+SYSMSG_DEF	设置默认系统消息，保存到 flash	✓	✗ [3] 相似命令参考*AT+SYSMSG*
AT+CWMODE	设置 Wi-Fi 模式	✗	✓ 新增切换模式自动连接
AT+CWMODE_CUR	设置当前 Wi-Fi 模式，不保存到 flash	✓	✗ [3] 相似命令参考*AT+CWMODE*
AT+CWMODE_DEF	设置默认 Wi-Fi 模式，保存到 flash	✓	✗ [3] 相似命令参考*AT+CWMODE*
AT+CWJAP	连接 AP	✓	✓ 新增更多功能
AT+CWJAP_CUR	连接 AP，参数不保存到 flash	✓	✗ [3] 相似命令参考*AT+CWJAP*
AT+CWJAP_DEF	连接 AP，参数保存到 flash	✓	✗ [3] 相似命令参考*AT+CWJAP*

下页继续

表 3 – 续上页

命令	简介	NONOS-AT	ESP-AT
AT+CWLAPOPT	设置 CWLAP 命令的属性	✓	✓ 新增更多功能
AT+CWLAP	扫描当前可用 AP	✓	✓ 回复不同
AT+CWQAP	断开与 AP 连接	✓	✓
AT+CWSAP	配置 softAP 参数	✗	✓
AT+CWSAP_CUR	配置 softAP 参数，不保存到 flash	✓	✗ [3] 相似命令参考*AT+CWSAP*
AT+CWSAP_DEF	配置 softAP 参数，保存到 flash	✓	✗ [3] 相似命令参考*AT+CWSAP*
AT+CWLIF	查询连接到 softAP 的 Station 信息	✓	✓
AT+CWDHCP	设置 DHCP	✗	✓
AT+CWDHCP_CUR	设置 DHCP，不保存到 flash	✓	✗ [3] 相似命令参考*AT+CWDHCP*
AT+CWDHCP_DEF	设置 DHCP，保存到 flash	✓	✗ [3] 相似命令参考*AT+CWDHCP*
AT+CWDHCPS	设置 softAP DHCP 分配的 IP 范围	✗	✓
AT+CWDHCPS_CUR	设置 softAP DHCP 分配的 IP 范围，不保存到 flash	✓	✗ [3] 相似命令参考*AT+CWDHCPS*
AT+CWDHCPS_DEF	设置 softAP DHCP 分配的 IP 范围，保存到 flash	✓	✗ [3] 相似命令参考*AT+CWDHCPS*
AT+CWAUTOCONN	上电是否自动连接 AP	✓	✓
AT+CIPSTAMAC	设置 Station 接口的 MAC 地址	✗	✓
AT+CIPSTAMAC_CUR	设置 Station 接口 MAC 地址，不保存到 flash	✓	✗ [3] 相似命令参考*AT+CIPSTAMAC*

下页继续

表 3 – 续上页

命令	简介	NONOS-AT	ESP-AT
AT+CIPSTAMAC_DEF	设置 Station 接口 MAC 地址，保存到 flash	✓	✗[3] 相似命令参考*AT+CIPSTAMAC*
AT+CIPAPMAC	设置 softAP 的 MAC 地址	✗	✓
AT+CIPAPMAC_CUR	设置 softAP 的 MAC 地址，不保存到 flash	✓	✗[3] 相似命令参考*AT+CIPAPMAC*
AT+CIPAPMAC_DEF	设置 softAP 的 MAC 地址，保存到 flash	✓	✗[3] 相似命令参考*AT+CIPAPMAC*
AT+CIPSTA	设置 Station 的 IP 地址	✗	✓
AT+CIPSTA_CUR	设置 Station 的 IP 地址，不保存到 flash	✓	✗[3] 相似命令参考*AT+CIPSTA*
AT+CIPSTA_DEF	设置 Station 的 IP 地址，保存到 flash	✓	✗[3] 相似命令参考*AT+CIPSTA*
AT+CIPAP	设置 softAP 的 IP 地址	✗	✓
AT+CIPAP_CUR	设置 softAP 的 IP 地址，不保存到 flash	✓	✗[3] 相似命令参考*AT+CIPAP*
AT+CIPAP_DEF	设置 softAP 的 IP 地址，保存到 flash	✓	✗[3] 相似命令参考*AT+CIPAP*
AT+CWSTARTSMART	开启 SmartConfig	✓	✓
AT+CWSTOPSMART	停止 SmartConfig	✓	✓
AT+CWSTARTDISCOVER	开启可被局域网网内微信探测模式	✓	✗
AT+CWSTOPDISCOVER	关闭可被局域网网内微信探测模式	✓	✗
AT+WPS	设置 WPS 功能	✓	✓
AT+MDNS	设置 mDNS 功能	✓	✓

下页继续

表 3 – 续上页

命令	简介	NONOS-AT	ESP-AT
AT+CWJEAP	连接企业级加密路由器	✗	ESP8266 ✗ ESP32 ✓ ESP32-S2 ✗
AT+CWJEAP_CUR	连接企业级加密路由器，不保存到 flash	✓	✗
AT+CWJEAP_DEF	连接企业级加密路由器，保存到 flash	✓	✗
AT+CWHOSTNAME	设置 Station 的主机名称	✓	✓
AT+CWCOUNTRY	设置 Wi-Fi 国家码	✗	✓
AT+CWCOUNTRY_CUR	设置 Wi-Fi 国家码，不保存到 flash	✓	✗ [3] 相似命令参考*AT+CWCOUNTRY*
AT+CWCOUNTRY_DEF	设置 Wi-Fi 国家码，保存到 flash	✓	✗ [3] 相似命令参考*AT+CWCOUNTRY*
AT+CIPSTATUS	查询网网络连接信息	✓	✓
AT+CIPDOMAIN	域名解析功能	✓	✓
AT+CIPSTART	建立 TCP/UDP/SSL 连接	✓	✓
AT+CIPSSLSIZE	设置 SSL Buffer size	✓	✗
AT+CIPSSLCCONF	配置 SSL 连接属性	✓	✓ 参数不同
AT+CIPSEND	发送数据	✓	✓
AT+CIPSENDEX	发送长度为 length 数据，或在长度内 0 结尾的数据	✓	✓
AT+CIPSENDBUF	数据写入 TCP 发包缓存	✓	✗ [3]
AT+CIPBUFRESET	重新计数	✓	✗ [3]
AT+CIPBUFSTATUS	查询 TCP 发包缓存的状态	✓	✗ [3]
AT+CIPCHECKSEQ	查询写入 TCP 发包缓存的某包是否发送成功	✓	✗ [3]
AT+CIPCLOSEMODE	设置 TCP 连接的断开方式	✓	✗ [3]
AT+CIPCLOSE	关闭 TCP/UDP/SSL 传输	✓	✓
AT+CIFSR	查询本地 IP 地址	✓	✓
AT+CIPMUX	设置多连接	✓	✓

下页继续

表 3 – 续上页

命令	简介	NONOS-AT	ESP-AT
AT+CIPSERVER	建立 TCP/SSL 服务器	✓ 不支持 SSL server	✓ ESP32 和 ESP32-S2 支持 SSL server，ESP8266 不支持 SSL server
AT+CIPSERVERMAXCONN	设置服务器允许建立的最大连接数	✓	✓
AT+CIPMODE	设置传输模式	✓	✓
AT+SAVETRANSLINK	设置开机透传模式信息	✓	✓
AT+CIPSTO	设置 TCP 服务器器超时时间	✓	✓
AT+PING	Ping 功能	✓	✓
AT+CIUPDATE	通过 Wi-Fi 升级固件	✓	✓ 支持更多参数
AT+CIPDINFO	接收网络数据时是否提示对端 IP 和端口	✓	✓
AT+CIPRECVMODE	设置 TCP 连接的数据接收方式	✓	✓
AT+CIPRECVDATA	被动接收模式时，读取缓存的 TCP 数据	✓	✓ 回复有差异
AT+CIPRECVLEN	被动接收模式时，查询缓存 TCP 数据的长度	✓	✓
AT+CIPSNTPCFG	设置时域和 SNTP 服务器	✓	✓ 支持更多功能
AT+CIPSNTPTIME	查询 SNTP 时间	✓	✓
AT+CIPDNS	自定义 DNS 服务器	✗	✓
AT+CIPDNS_CUR	自定义 DNS 服务器，不保存到 flash	✓	✗ [3] 相似命令参考*AT+CIPDNS*
AT+CIPDNS_DEF	自定义 DNS 服务器，保存到 flash	✓	✗ [3] 相似命令参考*AT+CIPDNS*
AT+SYSFLASH	读写 flash 用户分区	✗	✓

说明 1：新版 ESP-AT 中的 AT+SLEEP：

- ESP8266 and ESP32 ✓
  - 1：modem sleep by DTIM
  - 2：light sleep
  - 3：modem sleep by listen interval
- ESP32-S2 ✗

说明 2：新版 ESP-AT 中的 AT+RFPOWER：

- ESP8266 ✓，范围：[40,82]，单位：0.25 dBm
- ESP32 ✓，范围：[40,78]，单位：0.25 dBm，且支持 Bluetooth LE
- ESP32-S2 ✓，范围：[40,78]，单位：0.25 dBm

说明 3：新版 ESP-AT 不添加此命令。

强烈建议在使用命令之前先阅读以下内容，了解 AT 命令的一些基本信息。

- *AT 命令分类*
- *参数信息保存在 flash 中的 AT 命令*
- *AT 消息*

3.13 AT 命令分类

通用 AT 命令有四种类型：

类型	命令格式	说明
测试命令	AT+< 命令名称 >=?	查询设置命令的内部参数及其取值范围
查询命令	AT+< 命令名称 >?	返回当前参数值
设置命令	AT+< 命令名称 >=<…>	设置用户自定义的参数值，并运行命令
执行命令	AT+< 命令名称 >	运行无用户自定义参数的命令

- 不是每条 AT 命令都具备上述四种类型的命令。
- 命令里输入参数，当前只支持字符串参数和整形数字参数。
- 尖括号 < > 内的参数不可以省略。
- 方括号 [ ] 内的参数可以省略，省略时使用默认值。例如，运行*AT+CWJAP* 命令时省略某些参数：

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

- 当省略的参数后仍有参数要填写时，必须使用 ,，以示分隔，如：

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

- 使用双引号表示字符串参数，如：

```
AT+CWSAP="ESP756290","21030826",1,4
```

- 特殊字符需作转义处理，如 ,、"、\ 等。

  - \\：转义反斜杠。
  - \,：转义逗号，分隔参数的逗号无需转义。
  - \"：转义双引号，表示字符串参数的双引号无需转义。
  - \<any>：转义 <any> 字符，即只使用 <any> 字符，不使用反斜杠。

- 只有 AT **命令**中的特殊字符需要转义，其它地方无需转义。例如，AT 命令口打印 > 等待输入数据时，该数据不需要转义。

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

- AT 命令的默认波特率为 115200。
- 每条 AT 命令的长度不应超过 256 字节。
- AT 命令以新行 (CR-LF) 结束，所以串口工具应设置为“新行模式”。
- AT 命令错误代码的定义请见*AT API Reference*：
  - *esp_at_error_code*
  - *esp_at_para_parse_result_type*
  - *esp_at_result_code_string_index*

3.14 参数信息保存在 flash 中的 AT 命令

以下 AT 命令的参数更改将始终保存在 flash 的 NVS 区域中，因此重启后，会直接使用。

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

其它一些命令的参数更改是否保存到 flash 可以通过*AT+SYSSTORE* 命令来配置，具体请参见命令的详细说明。

3.15 AT 消息

从 ESP-AT 命令端口返回的 ESP-AT 消息有两种类型：ESP-AT 响应（被动）和 ESP-AT 消息报告（主动）。

- ESP-AT 响应（被动）

  每个输入的 ESP-AT 命令都会返回响应，告诉发送者 ESP-AT 命令的执行结果。

表 4: ESP-AT 响应

AT 响应	说明
OK	AT 命令处理完毕，返回 OK
ERROR	AT 命令错误或执行过程中发生错误
SEND OK	数据已发送到协议栈（针对于*AT+CIPSEND* 和*AT+CIPSENDEX* 命令），但不代表数据已经发到对端
SEND FAIL	向协议栈发送数据时发生错误（针对于*AT+CIPSEND* 和*AT+CIPSENDEX* 命令）
+<Command Name>:...	详细描述 AT 命令处理结果

- ESP-AT 消息报告（主动）

  ESP-AT 会报告系统中重要的状态变化或消息。

表 5: ESP-AT 消息报告

ESP-AT 消息报告	说明
ready	ESP-AT 固件已经准备就绪
busy p…	系统繁忙，正在处理上一条命令，无法处理新的命令
ERR CODE:<0x%08x>	不同命令的错误代码
Will force to restart!!!	立即重启模块
smartconfig type:<xxx>	Smartconfig 类型
Smart get wifi info	Smartconfig 已获取 SSID 和 PASSWORD
smartconfig connected wifi	Smartconfig 完成，ESP-AT 已连接到 Wi-Fi
WIFI CONNECTED	Wi-Fi station 接口已连接到 AP
WIFI GOT IP	Wi-Fi station 接口已获取 IPv4 地址
WIFI GOT IPv6 LL	Wi-Fi station 接口已获取 IPv6 链路本地地址
WIFI GOT IPv6 GL	Wi-Fi station 接口已获取 IPv6 全局地址
WIFI DISCONNECT	Wi-Fi station 接口已与 AP 断开连接
+ETH_CONNECTED	以太网接口已连接
+ETH_GOT_IP	以太网接口已获取 IPv4 地址
+ETH_DISCONNECTED	以太网接口已断开
[<conn_id>,]CONNECT	ID 为 <conn_id> 的网络连接已建立（默认情况下，ID 为 0）
[<conn_id>,]CLOSED	ID 为 <conn_id> 的网络连接已断开（默认情况下，ID 为 0）
+LINK_CONN	TCP/UDP/SSL 连接的详细信息
+STA_CONNECTED: <sta_mac>	station 已连接到 ESP-AT 的 Wi-Fi softAP 接口

下页继续

表 5 – 续上页

ESP-AT 消息报告	说明
+DIST_STA_IP: <sta_mac>,<sta_ip>	ESP-AT 的 Wi-Fi softAP 接口给 station 分配 IP 地址
+STA_DISCONNECTED: <sta_mac>	station 与 ESP-AT 的 Wi-Fi softAP 接口的连接断开
>	ESP-AT 正在等待用户输入数据
Recv <xxx> bytes	ESP-AT 从命令端口已接收到 <xxx> 字节
+IPD	ESP-AT 已收到来自网络的数据
SEND Canceled	取消在 Wi-Fi 普通传输模式 下发送数据
Have <xxx> Connections	已达到服务器的最大连接数
+QUITT	ESP-AT 退出 Wi-Fi 透传模式
NO CERT FOUND	在自定义分区中没有找到有效的设备证书
NO PRVT_KEY FOUND	在自定义分区中没有找到有效的私钥
NO CA FOUND	在自定义分区中没有找到有效的 CA 证书
+MQTTCONNECTED	MQTT 已连接到 broker
+MQTTDISCONNECTED	MQTT 与 broker 已断开连接
+MQTTSUBRECV	MQTT 已从 broker 收到数据
+MQTTPUB:FAIL	MQTT 发布数据失败
+MQTTPUB:OK	MQTT 发布数据完成
+BLECONN	Bluetooth LE 连接已建立
+BLEDISCONN	Bluetooth LE 连接已断开
+READ	通过 Bluetooth LE 连接进行读取操作
+WRITE	通过 Bluetooth LE 进行写入操作
+NOTIFY	来自 Bluetooth LE 连接的 notification
+INDICATE	来自 Bluetooth LE 连接的 indication
+BLESECNTFYKEY	Bluetooth LE SMP 密钥
+BLESECREQ:<conn_index>	收到来自 Bluetooth LE 连接的加密配对请求
+BLEAUTHCMPL:<conn_index>,<enc_result>	Bluetooth LE SMP 配对完成

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`

4.4 [ESP32 Only] Ethernet AT 示例

See `/docs/en/AT_Command_Examples/Ethernet_AT_Examples`.

4.5 Web Server AT 示例

[English]

本文档主要介绍 AT web server 的使用，主要涉及以下几个方面的应用：

- 使用浏览器进行 *Wi-Fi* 配网
- 使用浏览器进行 *OTA* 固件升级
- 使用微信小程序进行 *Wi-Fi* 配网
- 使用微信小程序进行 *OTA* 固件升级
- *[ESP32][ESP32-S Series][ESP32-C Series]* 使用 *Captive Portal* 功能

注解： 默认的 AT 固件并不支持 AT web server 的功能，请参考 *Web server AT Commands* 启用该功能。

4.5.1 使用浏览器进行 Wi-Fi 配网

简介

通过 web server，手机或 PC 可以设置 ESP 设备的 Wi-Fi 连接信息。您可以使用手机或电脑连接到 ESP 设备的 AP，通过浏览器打开配网网页，并将 Wi-Fi 配置信息发送给 ESP 设备，然后 ESP 设备将根据该配置信息连接到指定的路由器。

流程

整个配网流程可以分为以下三步：

- 配网设备连接 *ESP* 设备
- 使用浏览器发送配网信息
- 通知配网结果

配网设备连接 ESP 设备

首先，为了让配网设备连接 ESP 设备，ESP 设备需要配置成 AP + STA 模式，并创建一个 WEB 服务器等待配网信息，对应的 AT 命令如下：

1. 清除之前的配网信息，如果不清除配网信息，可能因为依然保留之前的配置信息从而导致 WEB 服务器无法创建。

   - Command

   ```
   AT+RESTORE
   ```

2. 配置 ESP 设备为 Station + SoftAP 模式。

   - Command

   ```
   AT+CWMODE=3
   ```

3. 设置 SoftAP 的 ssid 和 password（如设置默认连接 ssid 为 *pos_softap*，无密码的 Wi-Fi）。

   - Command

   ```
   AT+CWSAP="pos_softap","",11,0,3
   ```

4. 使能多连接。

   - Command

   ```
   AT+CIPMUX=1
   ```

5. 创建 web server，端口：80，最大连接时间：25 s（默认最大为 60 s）。

   - Command

   ```
   AT+WEBSERVER=1,80,25
   ```

然后，使用上述命令启动 web server 后，打开配网设备的 Wi-Fi 连接功能，连接 ESP 设备的 AP：

图 1: 浏览器连接 ESP AP

使用浏览器发送配网信息

在配网设备连接到 ESP 设备后，即可发送 HTTP 请求，配置待接入的路由器的信息：（注意，浏览器配网不支持配网设备作为待接入 AP，例如，如果使用手机连接到 ESP 的 AP，则该手机不建议作为 ESP 设备待接入的热点。）在浏览器中输入 web server 默认的 IP 地址（如果未设置 ESP 设备的 SoftAP IP 地址，默认为 192.168.4.1，您可以通过 AT+CIPAP? 命令查询当前的 SoftAP IP 地址），打开配网界面，输入拟连接的路由器的 ssid、password，点击“开始配网”即可开始配网：

图 2: 浏览器打开配网界面

用户也可以点击配网页面中 SSID 输入框右方的下拉框，查看 ESP 模块附近的 AP 列表，选择目标 AP 并输入 password 后，点击“开始配网”即可启动配网：

图 3: 浏览器获取 Wi-Fi AP 列表示意图

通知配网结果

配网成功后网页显示如下：

图 4: 浏览器配网成功

说明 1：配网成功后，网页将自动关闭，若想继续访问网页，请重新输入 ESP 设备的 IP 地址，重新打开网页。

同时，在串口端将收到如下信息：

```
+WEBSERVERRSP:1       // 代表启动配网
WIFI CONNECTED        // 代表正在连接
WIFI GOT IP           // 连接成功并获取到 IP
+WEBSERVERRSP:2       // 代表网页端收到配网成功结果，此时可以释放 WEB 资源
```

如果配网失败，网页将显示：

同时，在串口端将收到如下信息：

图 5: 浏览器配网失败

```
+WEBSERVERRSP:1       // 代表启动配网，没有后续发起连接以及获取 IP 的信息，MCU 可以在收到该条消息后建立计时，若计时超时，则配网失败。
```

常见故障排除

说明 1：配网页面收到提示“数据发送失败”。请检查 ESP 模块的 Wi-Fi AP 是否正确开启，以及 AP 的相关配置，并确认已经输入正确的 AT 命令成功启用 web server。

4.5.2 使用浏览器进行 OTA 固件升级

简介

浏览器打开 web server 的网页后，可以选择进入 OTA 升级页面，通过网页对 ESP 模块进行固件升级。

流程

- 打开 *OTA* 配置页面
- 选择并发送新版固件
- 通知固件发送结果

打开 OTA 配置页面

如图，点击网页右下角“OTA 升级”选项，打开 OTA 配置页面后，可以查看当前固件版本、AT Core 版本：

说明 1：仅当浏览器连接 ESP 模块的 AP，或者访问 OTA 配置页面的设备与 ESP 模块连接在同一个子网中时，才可以打开该配置界面。

说明 2：网页上显示的“当前固件版本”为当前用户编译的应用程序版本号，用户可通过 `./build.py menuconfig -> Component config -> AT -> AT firmware version` (参考*Build Your Own ESP-AT Project*) 更改该版本号，建立固件版本与应用程序的同步关系，以便于管理应用程序固件版本。


ESP-AT
请选择新版本固件，然后点击升级按钮进行升级：
浏览
固件升级
当前 app 固件版本： 2.1.0
AT Core 版本： 2.2.0.0

图 6: OTA 配置页面

选择并发送新版固件

如图，点击页面中的“浏览”按钮，选择待发送的新版固件：

图 7: 选择待发送的新版固件

说明 1：在发送新版固件之前，系统会对选择的固件进行检查。固件命名的后缀必须为.bin，且其大小不超过 2M。

通知固件发送结果

如图，固件发送成功，将提示“升级成功”：

同时，在串口端将收到如下信息：

```
+WEBSERVERRSP:3       // 代表开始接收 OTA 固件数据
+WEBSERVERRSP:4       // 代表成功接收 OTA 固件数据并且对数据的校验正确，此时 MCU 可以选择重启␣
↪ESP 设备，以应用新版本的固件
```

若接收的 OTA 固件数据校验失败，在串口端将收到如下信息：


ESP-AT
请选择新版本固件，然后点击升级按钮进行升
提示
升级成功
确定
固件升级
当前 app 固件版本： 2.1.0
AT Core 版本： 2.2.0.0

图 8: 新版固件发送成功

```
+WEBSERVERRSP:3       // 代表开始接收 OTA 固件数据
+WEBSERVERRSP:5       // 代表接收的 OTA 固件数据校验失败，用户可以选择重新打开 OTA 配置界面，按照上述步骤进行 OTA 固件升级
```

4.5.3 使用微信小程序进行 Wi-Fi 配网

简介

微信小程序配网是通过微信小程序连接 ESP 设备创建的 AP，并通过微信小程序将需要连接的 AP 信息传输给 ESP 设备，ESP 设备通过这些信息连接到对应的 AP，并通知微信小程序配网结果的解决方案。

流程

整个配网流程可以分为以下四步：

- 配置 *ESP* 设备参数
- 加载微信小程序
- 目标 *AP* 选择
- 执行配网

配置 ESP 设备参数

为了让小程序连接 ESP 设备，ESP 设备需要配置成 AP + STA 模式，并创建一个 WEB 服务器等待小程序连接，对应的 AT 命令如下：

1. 清除之前的配网信息，如果不清除配网信息，可能因为依然保留之前的配置信息从而导致 WEB 服务器无法创建。
   - Command

   ```
   AT+RESTORE
   ```

2. 配置 ESP 设备为 Station + SoftAP 模式。
   - Command

   ```
   AT+CWMODE=3
   ```

3. 设置 SoftAP 的 ssid 和 password（如设置默认连接 ssid 为 *pos_softap*，password 为 *espressif*）。
   - Command

```
AT+CWSAP="pos_softap","espressif",11,3,3
```

注解： 微信小程序默认向 ssid 为 *pos_softap*，password 为 *espressif* 的 SoftAP 发起连接，请确保将 ESP 设备的参数按照上述配置进行设置。

1. 使能多连接。

   - Command

   ```
   AT+CIPMUX=1
   ```

2. 创建 web server，端口：80，最大连接时间：40 s（默认最大为 60 s）。

   - Command

   ```
   AT+WEBSERVER=1,80,40
   ```

加载微信小程序

打开手机微信，扫描下面的二维码：

图 9: 获取小程序的二维码

打开微信小程序，进入配网界面：

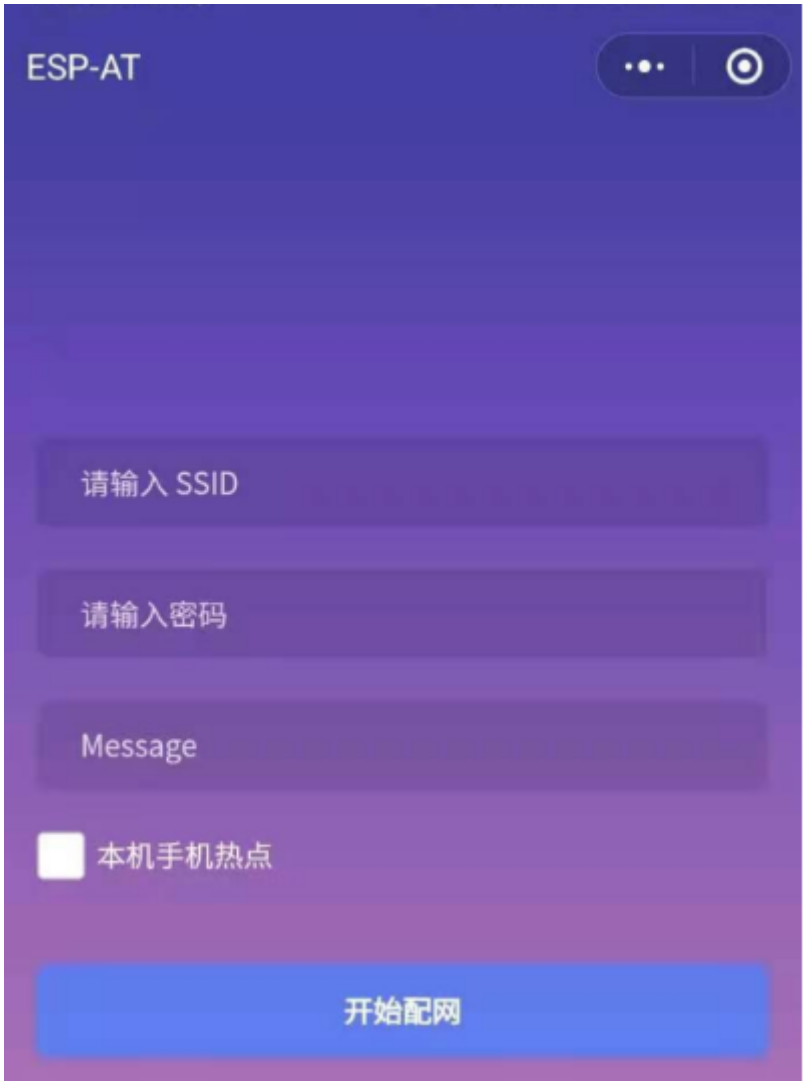


图 10: 小程序配网界面

目标 AP 选择

加载微信小程序后，根据待连接的目标 AP，可将配网情况分为两种情况：

1. 待接入的目标 AP 为本机配网手机提供的热点。此时请选中微信小程序页面的“本机手机热点”选项框。

2. 待接入的目标 AP 不是本机配网手机提供的热点，如路由器等 AP。此时请确保“本机手机热点”选项框未被选中。

执行配网

待接入的目标 AP 不是本机配网手机

这里以待接入的热点为路由器为例，介绍配网的过程：

1. 打开手机 Wi-Fi，连接路由器：

图 11: 配网设备连接拟接入的路由器

2. 打开微信小程序，可以看到小程序页面已经自动显示当前路由器的 ssid 为”FAST_FWR310_02”。

注意：如果当前页面未显示已经连接的路由器的 ssid，请点击下图中的“重新进入小程序”，刷新当前页面：

3. 输入路由器的 password 后，点击“开始配网”。

4. 配网成功，小程序页面显示：

同时，在串口端将收到如下信息：

```
+WEBSERVERRSP:1       // 代表启动配网
WIFI CONNECTED        // 代表正在连接
WIFI GOT IP           // 连接成功并获取到 IP
+WEBSERVERRSP:2       // 代表小程序收到配网成功结果，此时可以释放 WEB 资源
```

5. 若配网失败，则小程序页面显示：

同时，在串口端将收到如下信息：

图 12: 小程序获取拟接入的路由器信息

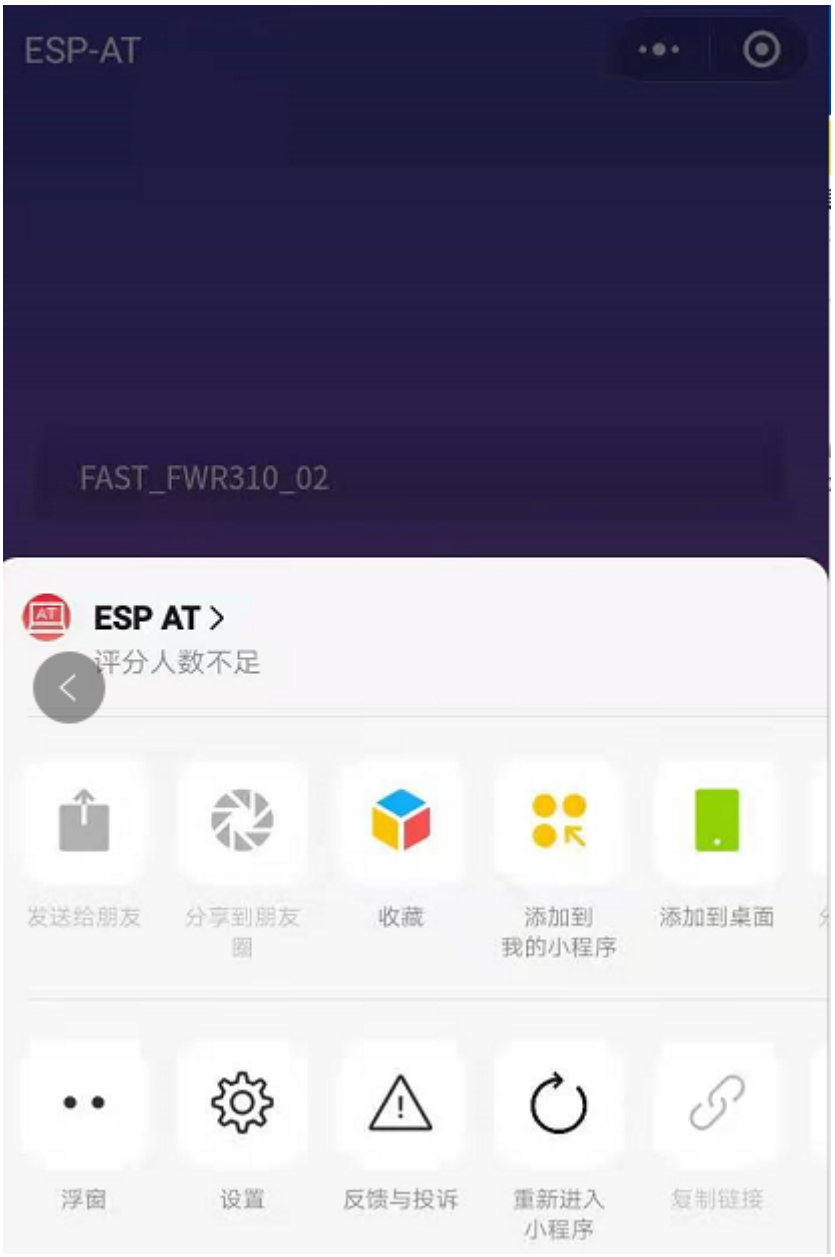

ESP-AT
FAST_FWR310_02
ESP AT >
评分人数不足
发送给朋友
分享到朋友圈
收藏
添加到我的小程序
添加到桌面
浮窗
设置
反馈与投诉
重新进入小程序
复制链接

图 13: 重新进入小程序

图 14: 小程序启动 ESP 模块连接路由器

图 15: 小程序配网成功界面

图 16: 小程序配网失败界面

```
+WEBSERVERRSP:1       // 代表启动配网，没有后续发起连接以及获取 IP 的信息，MCU 可以在收到该条消息后建立计时，若计时超时，则配网失败。
```

待接入的目标 AP 为本机配网手机

如果正在配网的手机作为待接入 AP，则用户不需要输入 ssid，只需要输入本机的 AP 的 password，并根据提示及时打开手机 AP 即可（如果手机支持同时打开 Wi-Fi 和分享热点，也可提前打开手机 AP）。

1. 选中微信小程序页面的“本机手机热点”选项框，输入本机热点的 password 后，点击“开始配网”。

图 17: 输入本机 AP 的 password

2. 启动配网后，在收到提示“连接手机热点中”的提示后，请检查本机手机热点已经开启，此时 ESP 设备将自动扫描周围热点并发起连接。

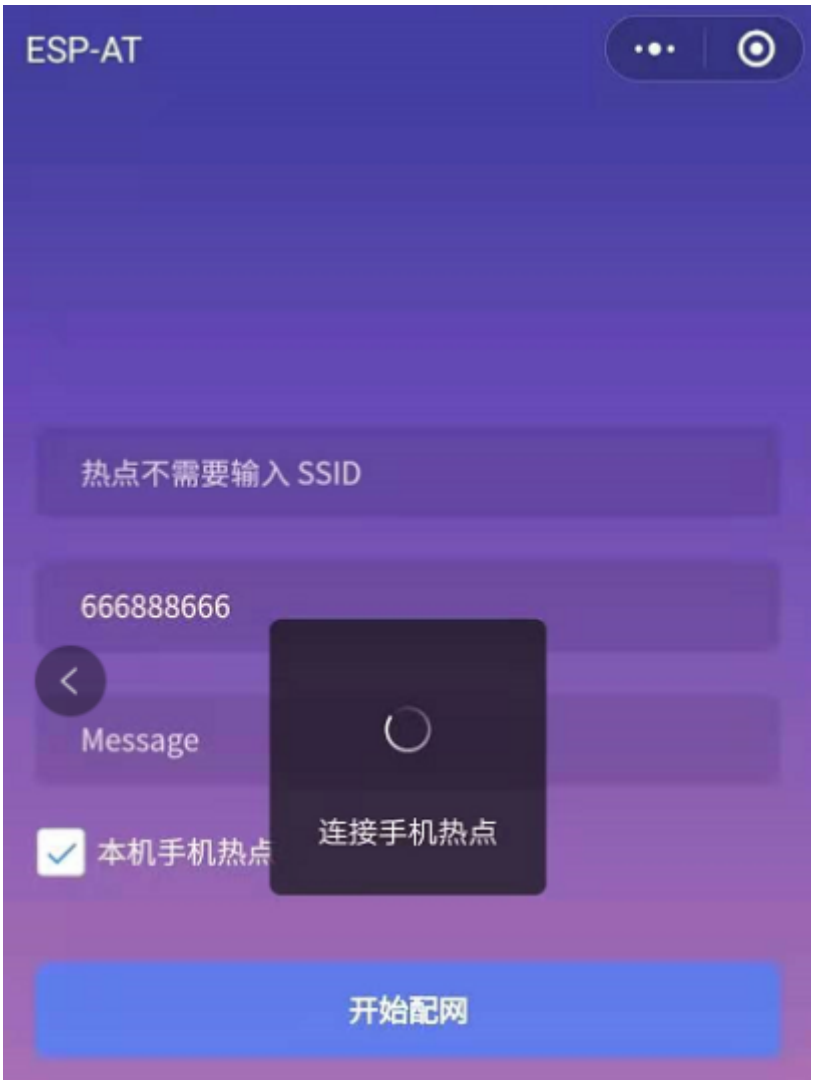


图 18: 开始连接本机 AP

3. 配网结果在小程序页面的显示以及串口端输出的数据与上述“待接入的目标 AP 不是本机配网手机”时的情况一样，请参考上文。

常见故障排除

说明 1：配网页面收到提示“数据发送失败”。请检查 ESP 模块的 Wi-Fi AP 是否正确开启，以及 AP 的相关配置，并确认已经输入正确的 AT 命令成功启用 web server。

说明 2：配网页面收到提示“连接 AP 失败”。请检查配网设备的 Wi-Fi 连接功能是否打开，检查 ESP 模块的 Wi-Fi AP 是否正确开启，以及 AP 的 ssid、password 是否按上述步骤进行配置。

说明 3：配网页面收到提示“系统保存的 Wi-Fi 配置过期”。请手动使用手机连接 ESP 模块 AP，确认 ESP 模块的 ssid、password 已经按照上述步骤进行配置。

4.5.4 使用微信小程序进行 OTA 固件升级

微信小程序支持在线完成 ESP 设备的固件升级，请参考上述*配置 ESP 设备参数* 的具体步骤完成 ESP 模块的配置（如果已经在配网时完成配置，不用重复配置）。完成配置后，设备执行 OTA 固件升级的流程与使用浏览器进行 OTA 固件升级类似，请参考*使用浏览器进行 OTA 固件升级*。

4.5.5 [ESP32][ESP32-S Series][ESP32-C Series] 使用 Captive Portal 功能

简介

Captive Portal，是一种“强制认证主页”技术，当使用支持 Captive Portal 的 station 设备连接到提供 Captive Portal 服务的 AP 设备时，将触发 station 设备的浏览器跳转到指定的网页。更多关于 Captive Portal 的介绍，请参考 Captive Portal Wiki。

注解： 默认情况下 AT web 并未启用该功能，可以通过 `./build.py menuconfig > Component config > AT > AT WEB Server command support > AT WEB captive portal support` 启用该功能，然后编译工程（请参考*Build Your Own ESP-AT Project*）。此外，启用该功能，可能导致使用微信小程序进行配网或 OTA 固件升级时发生页面跳转，建议仅在使用浏览器访问 AT web 时启用该功能。

流程

启用 Captive Portal 功能后，请参考上述*配网设备连接 ESP 设备* 的具体步骤完成 ESP 模块的配置，然后连接 ESP 设备的 AP：

如上图，station 设备连接打开 Captive Portal 功能的 ESP 设备的 AP 后，提示“需登录/认证”，然后将自动打开浏览器，并跳转到 AT web 的主界面。若不能自动跳转，请根据 station 设备的提示，点击“认证”或点击上图中的“pos_softap”热点的名称，手动触发 Captive Portal 自动打开浏览器，进入到 AT web 的主界面。

图 19: 连接打开 Captive Portal 功能的 AP

常见故障排除

说明 1：通信双方（station 设备、AP 设备）都支持 Captive Portal 功能才能保证该功能正常使用，因此，若设备连接 ESP 设备的 AP 后未提示“需登录/认证”，并且没有自动进入到 AT web 的主界面，可能是 station 设备不支持该功能，此时，请参考上述使用浏览器发送配网信息 的具体步骤手动打开 AT web 的主界面。

CHAPTER 5

如何编译和开发自己的 AT 工程

[English]

5.1 Build Your Own ESP-AT Project

This document details how to build your own ESP-AT project and flash the generated binary files into your ESP devices, including ESP32, ESP32-S2, ESP32-C3, and ESP8266. It comes in handy when the default *AT 固件* cannot meet your needs, for example, to customize the *AT port* pins, Bluetooth service, partitions, and so on.

The structure of this document is as follows:

- *Overview*: Overview of the steps to build an ESP-AT project.
- *ESP32, ESP32-S2 and ESP32-C3 Series*: Details steps to build a project for ESP32, ESP32-S2, and ESP32-C3 series.
- *ESP8266 Series*: Details steps to build a project for ESP8266.

5.1.1 Overview

Before compiling an ESP-AT project, you need first get started with ESP-IDF and set up the environment for ESP-IDF, because ESP-AT is based on ESP-IDF.

After the environment is ready, install the tools and ESP-AT SDK. Then, connect your ESP device to PC. Use `./build.py menuconfig` to set up some configuration for the project. Build the project and flash the generated bin files onto your ESP device.

注解： **Please pay attention to possible conflicts of pins**. If choosing `AT through HSPI`, you can get the information of the HSPI pin by `./build.py menuconfig` -> `Component config` -> `AT` -> `AT hspi settings`.

5.1.2 ESP32, ESP32-S2 and ESP32-C3 Series

This section describes how to compile an ESP-AT project for ESP32, ESP32-S2 and ESP32-C3 series.

Get Started with ESP-IDF

Get started with ESP-IDF before compiling an ESP-AT project, because ESP-AT is developed based on ESP-IDF, and the supported version varies from series to series:

表 1: ESP-IDF Versions for Different Series

Project	IDF Version	IDF Documentation Version
ESP32 ESP-AT	release/v4.2	ESP-IDF Get Started Guide v4.2
ESP32-S2 ESP-AT	release/v4.2	ESP-IDF Get Started Guide v4.2
ESP32-C3 ESP-AT	release/v4.3	ESP-IDF Get Started Guide v4.3

First, set up the development environment for ESP-IDF according to Step 1 to 4 of *ESP-IDF Get Started Guide* (click the corresponding link in the table above to navigate to the documentation).

Then, start a simple project of `hello_world` according to Step 5 to 10 of *ESP-IDF Get Started Guide* to make sure your environment works and familiarize yourself with the process of starting a new project based on ESP-IDF. It is not a must, but you are strongly recommended to do so.

After finishing all the ten steps (if not, at least the first four steps), you can move onto the following steps that are ESP-AT specific.

注解： Please do not set `IDF_PATH` during the process, otherwise, you would encounter some unexpected issues when compiling ESP-AT projects later.

Get ESP-AT

To compile an ESP-AT project, you need the software libraries provided by Espressif in the ESP-AT repository.

To get ESP-AT, navigate to your installation directory and clone the repository with git clone, following instructions below specific to your operating system.

- Linux or macOS

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

- Windows

```
cd %userprofile%\esp
git clone --recursive https://github.com/espressif/esp-at.git
```

If you are located in China or have difficulties to access GitHub, you can also use `git clone https://gitee.com/EspressifSystems/esp-at.git` to get ESP-AT, which may be faster.

ESP-AT will be downloaded into `~/esp/esp-at` on Linux or macOS, or `%userprofile%\esp\esp-at` on Windows.

注解: This guide uses the directory `~/esp` on Linux or macOS, or `%userprofile%\esp` on Windows as an installation folder for ESP-AT. You can use any directory, but you will need to adjust paths for the commands respectively. Keep in mind that ESP-AT does not support spaces in paths.

Connect Your Device

Connect your device to the PC with a USB cable to download firmware and log output. See 硬件连接 for more information. Note that you do not need to set up the "AT command/response" connection if you do not send AT commands and receive AT responses during the compiling process. You can change default port pins referring to *如何修改 AT port 管脚*.

Configure

In this step, you will clone the `esp-idf` folder into the `esp-at` folder, set up the development environment in the newly cloned folder, and configure your project.

1. Navigate to `~/esp/esp-at` directory.
2. Run the project configuration utility `menuconfig` to configure.

```
./build.py menuconfig
```

3. Select the following configuration options for your ESP device if it is your first time.

- Select the `Platform name` for your ESP device. For example, select `PLATFORM_ESP32` for ESP32 series of products, `PLATFORM_ESP32S2` for ESP32-S2 series of products. `Platform name` is defined in factory_param_data.csv .
- Select the `Module name` for your ESP device. For example, select `WROOM-32` for the ESP32-WROOM-32D module. `Module name` is defined in factory_param_data.csv .
- Enable or disable `silence mode`. If enabled, it will remove some logs and reduce the firmware size. Generally, it should be disabled.
- The above three option items will not appear if the file `build/module_info.json` exists. So please delete it if you want to reconfigure the module information.

4. Now, the `esp-idf` folder is created in `esp-at` folder. This `esp-idf` is different from that in Step **Get Started with ESP-IDF**.

- If the terminal prompt an error message like the following, please proceed with the next step to set up the develop environment in the `esp-at/esp-idf`.

```
The following Python requirements are not satisfied:
...
Please follow the instructions found in the "Set up the tools" section of ESP-IDF␣
↪Get Started Guide.
```

- If you have compiled an ESP-AT project for an ESP series before and want to switch to another series, you must run `rm -rf esp-idf` to remove the old esp/idf and then proceed with the next step.
- If your esp-idf is upgraded, you are recommended to proceed with the next step.

5. Set up the development environment in the `esp-at/esp-idf`.

- Set up the tools in the folder if it is the first time you compile the ESP-AT project. See Step 3 of *ESP-IDF Get Started Guide*.
- Set up environment variables in the folder **every time** you compile an ESP-AT project. See Step 4 of *ESP-IDF Get Started Guide*.
- **Install `pyyaml` and `xlrd` packages with pip in the folder if you have not done it**.

```
python -m pip install pyyaml xlrd
```

If the previous steps have been done correctly, the following menu appears after you run `./build.py menuconfig`:

You are using this menu to set up project-specific configuration, e.g. changing *AT port* pins, enabling Classic Bluetooth function, etc. If you made no changes, it will run with the default configuration.

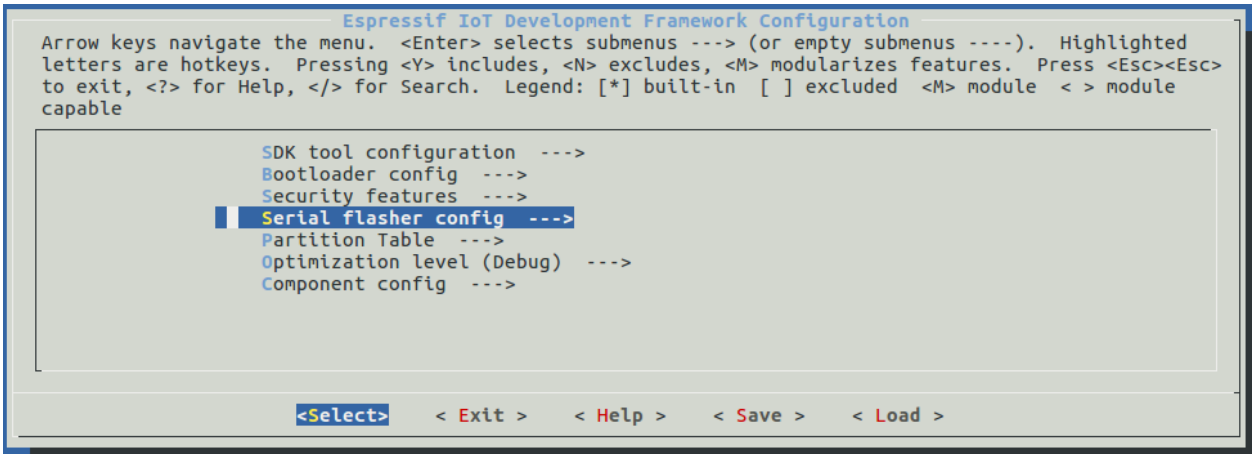


图 1: Project configuration - Home window

Build the Project

Build the project by running:

```
./build.py build
```

- If Bluetooth feature is enabled, the firmware size will be much larger. Please make sure it does not exceed the ota partition size.
- After compiled, the combined factory bin will be created in `build/factory`. See *如何了解 ESP-AT 同一平台不同模组差异* for more information.

Flash onto the Device

Flash the binaries that you just built onto your ESP32 board by running:

```
./build.py -p (PORT) flash
```

- Note that you may need to replace `PORT` with your ESP device's serial port name.
- Or you can follow the printed instructions to flash the bin files into flash. Note that you may also need to replace the `PORT`.
- If the ESP-AT bin fails to boot and prints "ota data partition invalid", you should run `./build.py erase_flash` to erase the entire flash, and then re-flash the AT firmware.

5.1.3 ESP8266 Series

This section describes how to compile an ESP-AT project for ESP8266 series.

Get Started with ESP-IDF

Get started with ESP-IDF before compiling an ESP-AT project, because ESP-AT is developed based on ESP-IDF:

First, set up the development environment according to ESP8266 RTOS SDK Get Started Guide v3.4.

Then, start a simple project of `hello_world` according to the Guide to make sure your environment works and familiarize yourself with the process of starting a new project based on ESP-IDF. It is not a must, but you are strongly recommended to do so.

After setting up the development environment and starting a simple project, you can move onto the following steps that are ESP-AT specific.

注解： Please do not set `IDF_PATH` during the process, otherwise, you would encounter some unexpected issues when compiling ESP-AT projects later.

Get ESP-AT

To compile an ESP-AT project, you need the software libraries provided by Espressif in the ESP-AT repository.

To get ESP-AT, navigate to your installation directory and clone the repository with git clone, following instructions below specific to your operating system.

- Linux or macOS

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

- Windows

```
cd %userprofile%\esp
git clone --recursive https://github.com/espressif/esp-at.git
```

If you are located in China or have difficulties to access GitHub, you can also use `git clone https://gitee.com/EspressifSystems/esp-at.git` to get ESP-AT, which may be faster.

ESP-AT will be downloaded into `~/esp/esp-at` on Linux or macOS, or `%userprofile%\esp\esp-at` on Windows.

注解： This guide uses the directory `~/esp` on Linux and macOS or `%userprofile%\esp` on Windows as an

installation folder for ESP-AT. You can use any directory, but you will need to adjust paths for the commands respectively. Keep in mind that ESP-AT does not support spaces in paths.

Checkout to ESP8266 Branch

```
git checkout release/v2.2.0.0_esp8266
```

Connect Your Device

Connect your device to the PC with a USB cable to download firmware and log output. See 硬件连接 for more information. Note that you do not need to set up the "AT command/response" connection if you do not send AT commands and receive AT responses during the compiling process. You can change default port pins referring to *如何修改 AT port 管脚*.

Configure

In this step, you will clone the `esp-idf` folder into the `esp-at` folder, set up the development environment in the newly cloned folder, and configure your project.

1. Navigate back to your `~/esp/esp-at` directory.
2. Run the project configuration utility `menuconfig` to configure.

```
./build.py menuconfig
```

3. Select the following configuration options for your ESP device if it is the first time you build the ESP-AT project.

- Select the `Platform name` for your ESP device, i.e., select `PLATFORM_ESP8266` for your ESP8266 device. `Platform name` is defined in factory_param_data.csv .
- Select the `Module name` for your ESP device. For example, select `WROOM-02` for the ESP-WROOM-02D module. `Module name` is defined in factory_param_data.csv .
- Enable or disable `silence mode`. If enabled, it will remove some logs and reduce the firmware size. Generally, it should be disabled.
- The above three option items will not appear if the file `build/module_info.json` exists. So please delete it if you want to reconfigure the module information.

4. Now, the `esp-idf` folder is created in `esp-at` folder. This `esp-idf` is different from that in Step **Get Started with ESP-IDF**.

- If the terminal prompt an error message like the following, please proceed with the next step to set up the develop environment in the `esp-at/esp-idf`.

```
The following Python requirements are not satisfied:
...
Please follow the instructions found in the "Set up the tools" section of ESP-IDF
↪Get Started Guide.
```

- If you have compiled an ESP-AT project for an ESP series before and want to switch to another series, you must run `rm -rf esp-idf` to remove the old esp/idf and then proceed with the next step.
- If your esp-idf is upgraded, you are recommended to proceed with the next step.

5. Set up the development environment in the `esp-at/esp-idf`.

- Set up the tools in the folder if it is the first time you compile an ESP-AT project. See *ESP8266 RTOS SDK Get Started Guide* v3.4 for more information.
- Set up environment variables in the folder **every time** you compile an ESP-AT project. See *ESP8266 RTOS SDK Get Started Guide* v3.4 for more information.
- **Install `pyyaml` and `xlrd` packages with pip in the folder if you have not done it**.

```
python -m pip install pyyaml xlrd
```

If the previous steps have been done correctly, the following menu appears after you run `./build.py menuconfig`:

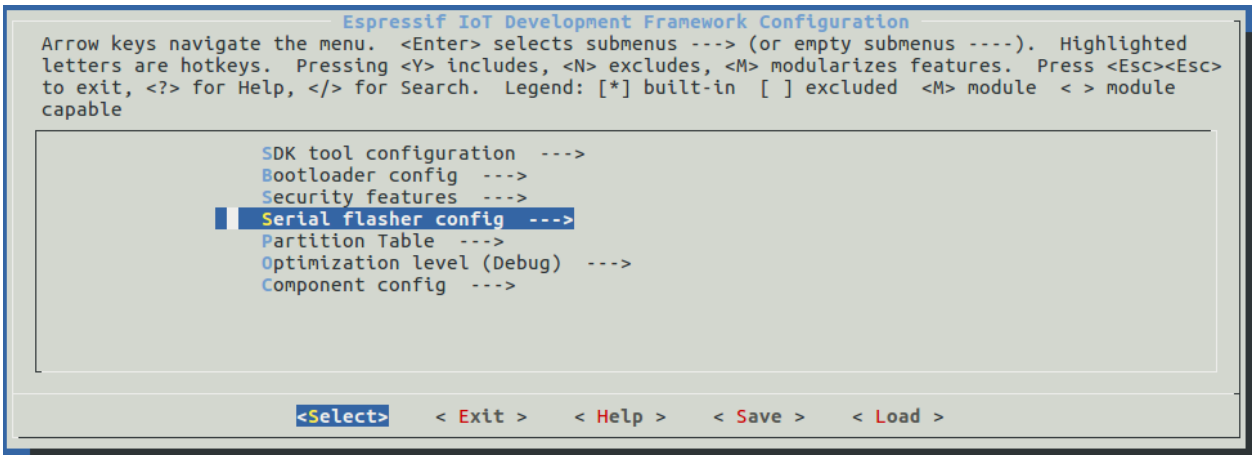


图 2: Project configuration - Home window

You are using this menu to set up project-specific configuration, e.g. changing *AT port* pins, enabling Classic Bluetooth function, etc. If you made no changes, it will run with the default configuration.

Build the Project

Build the project by running:

```
./build.py build
```

- After compiled, the combined factory bin will be created in `build/factory`. By default, the factory bin is 2 MB flash size, DIO flash mode and 80 MHz flash speed. See 如何了解 *ESP-AT* 同一平台不同模组差异 for more information.

Flash onto the Device

Flash the binaries that you just built onto your ESP8266 board by running:

```
./build.py -p (PORT) flash
```

- Note that you may need to replace `PORT` with your ESP device's serial port name.
- Or you can follow the printed instructions to flash the bin files into flash. Note that you may also need to replace the `PORT`.
- If the ESP-AT bin fails to boot and prints "ota data partition invalid", you should run `./build.py erase_flash` to erase the entire flash, and then re-flash the AT firmware.

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 `./build.py 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
reserved1	0
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
reserved1	0
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. `./build.py 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`.

5.2.3 ESP32-C3 平台

ESP32-C3 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-C3 系列芯片技术规格书, 在官方 release 固件中，UART0 作为 Log 的打印，默认管脚为

```
TX ---> GPIO21
RX ---> GPIO20
```

可以通过 `./build.py menuconfig > Component config > Common ESP-related > Channel 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 `MINI-1` 模组

Parameter	Value
platform	PLATFORM_ESP32C3
module_name	MINI-1
magic_flag	0xfcfc
version	2
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	7
uart_rx_pin	6
uart_cts_pin	5
uart_rts_pin	4
tx_control_pin	-1
rx_control_pin	-1

发送命令的 AT port 管脚分别为

```
TX ---> GPIO7
RX ---> GPIO6
CTS ---> GPIO5
RTX ---> GPIO4
```

如果想要使用 GPIO21 (TX)、GPIO20 (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 `MINI-1` 模组的 `uart_port` 为 0，`uart_tx_pin` 为 21 以及 `uart_rx_pin` 为 20，如下

Parameter	Value
platform	PLATFORM_ESP32C3
module_name	MINI-1
magic_flag	0xfcfc
version	2
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	21
uart_rx_pin	20
uart_cts_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`.

5.3 如何添加自定义 AT 命令

[English]

See `../../en/Compile_and_Develop/How_to_add_user-defined_AT_commands.rst`

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`（ESP32 模块）或者 `module_config/module_esp32c3_default/partitions_at.csv`（ESP32-C3 模块）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 How to enable ESP-AT Ethernet

5.8.1 Overview

Initialises the Ethernet interface and enables it, then sends DHCP requests and tries to obtain a DHCP lease. If successful then you will be able to ping the device.

5.8.2 PHY Configuration

Use `./build.py menuconfig` to set the PHY model. These configuration items will vary depending on the hardware configuration you are using.

The default configuration is correct for Espressif's Ethernet board with TP101 PHY. ESP32 AT supports up to four Ethernet PHY: `LAN8720`, `IP101`, `DP83848` and `RTL8201`. `TLK110` PHY is no longer supported because TI stoped production. If you want to use other phy, follow the document to design.

Compiling

1. `./build.py menuconfig` -> `Component config` -> `AT` -> `AT ethernet support` to enable ethernet.
2. `./build.py menuconfig` -> `Component config` -> `AT` -> `Ethernet PHY` to choose ethernet.
3. Recompile the `esp-at` project, download AT bin into flash.

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 SPI AT 指南

[English]

ESP8266 AT 不支持 SPI 传输，推荐使用 ESP32-C3 AT 进行 SPI 传输，详情请见 ESP32-C3 SPI AT 指南。

5.12 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.
5. Use the key to compile your own OTA BIN. The process of configuring the AT OTA token key is as follows:

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.

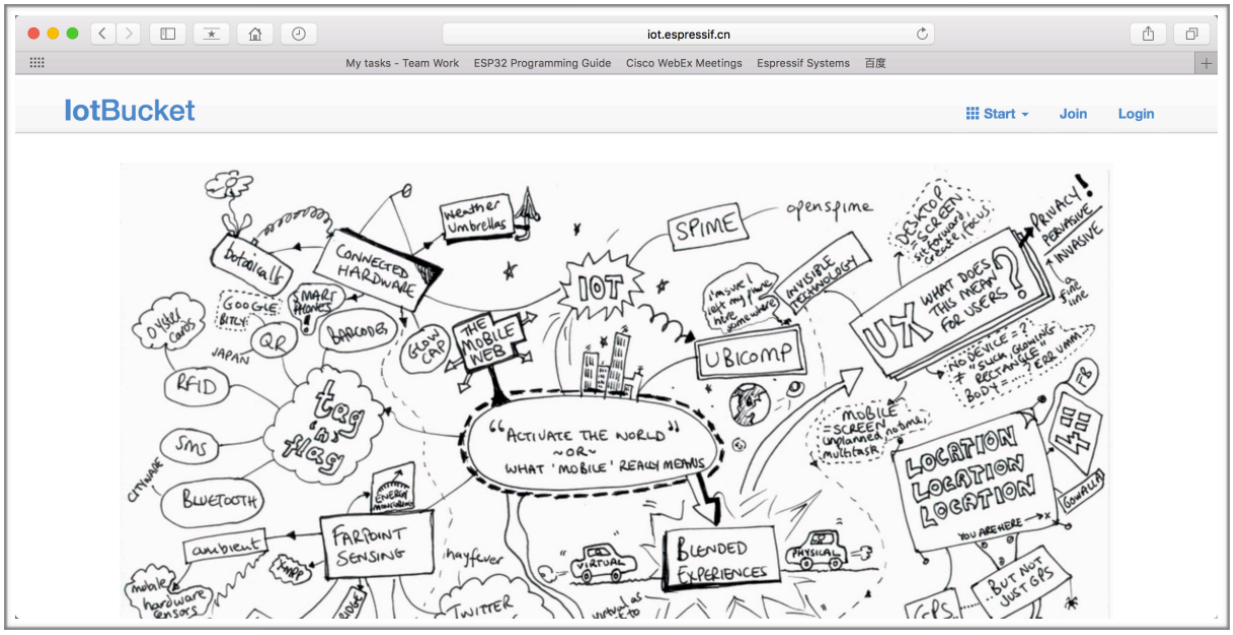


图 3: Open iot.espressif.cn website

图 4: Join iot.espressif.cn website

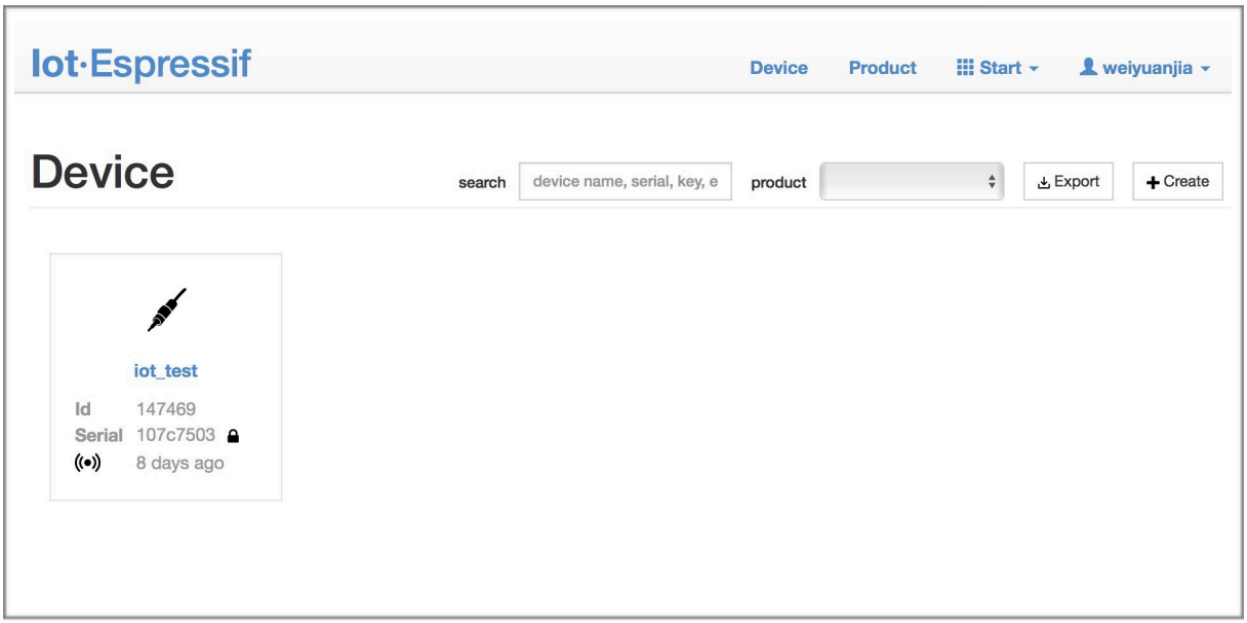


图 5: Click on “Device”

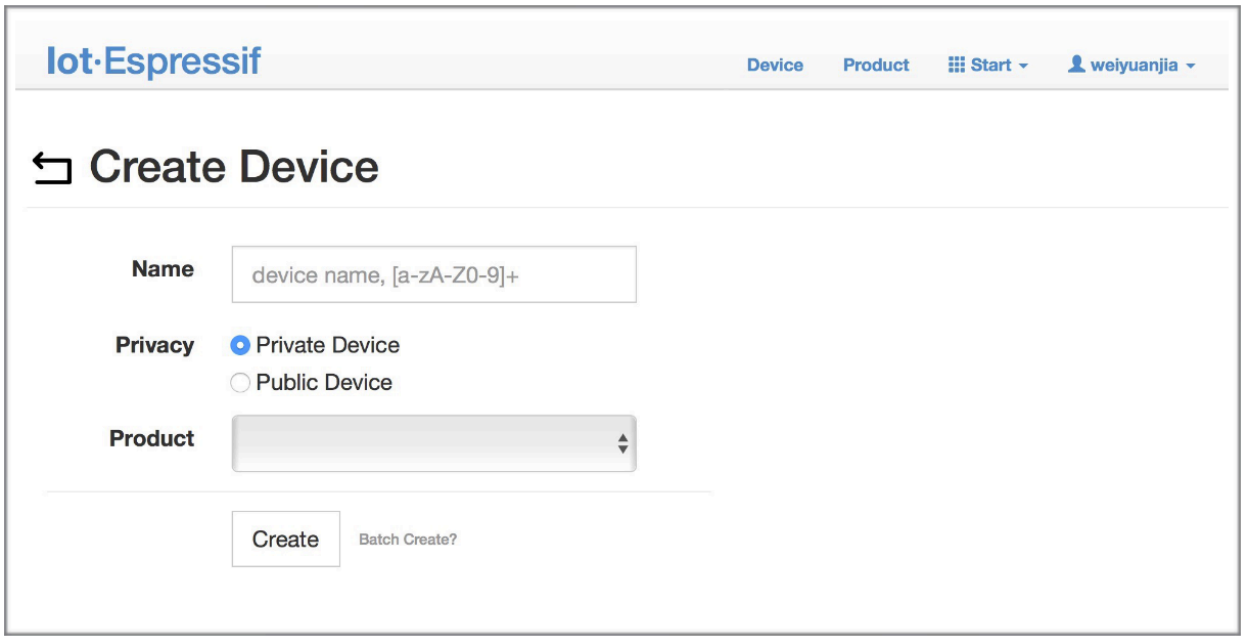


图 6: Click on “Create”

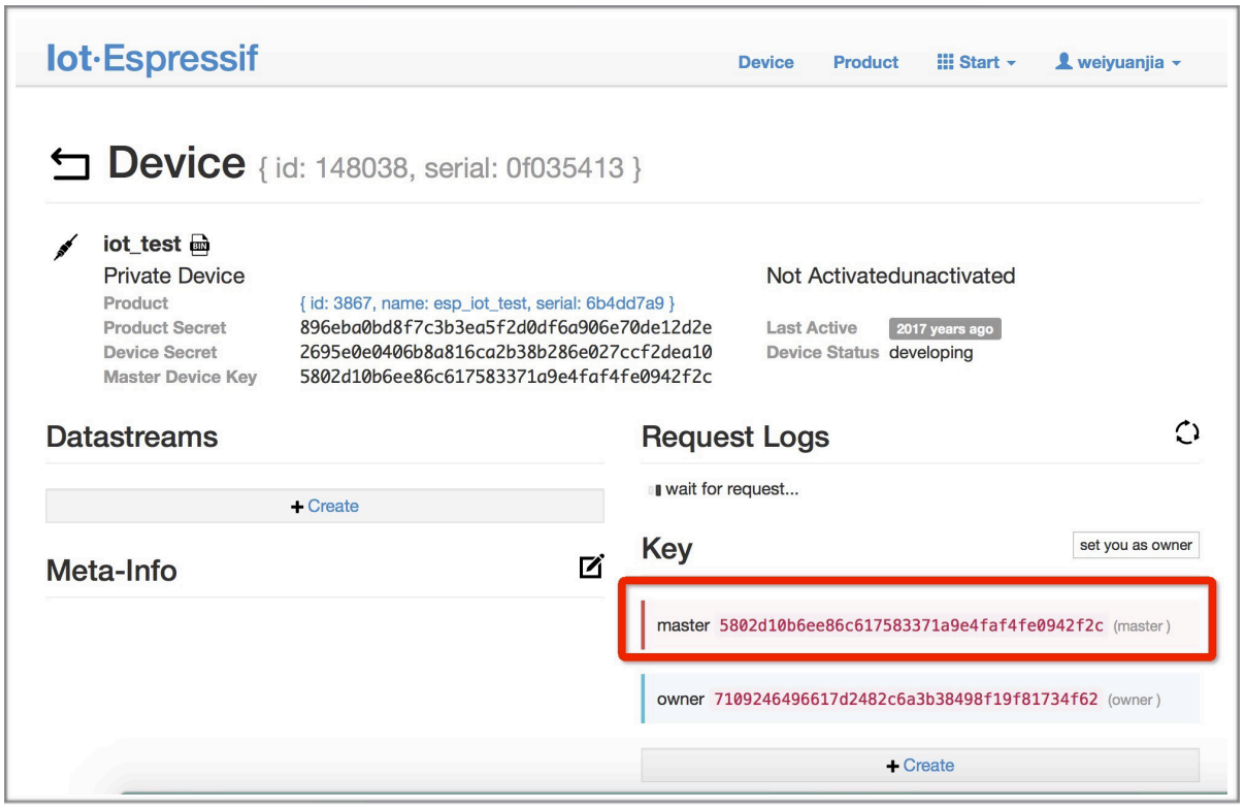


图 7: A key has been generated

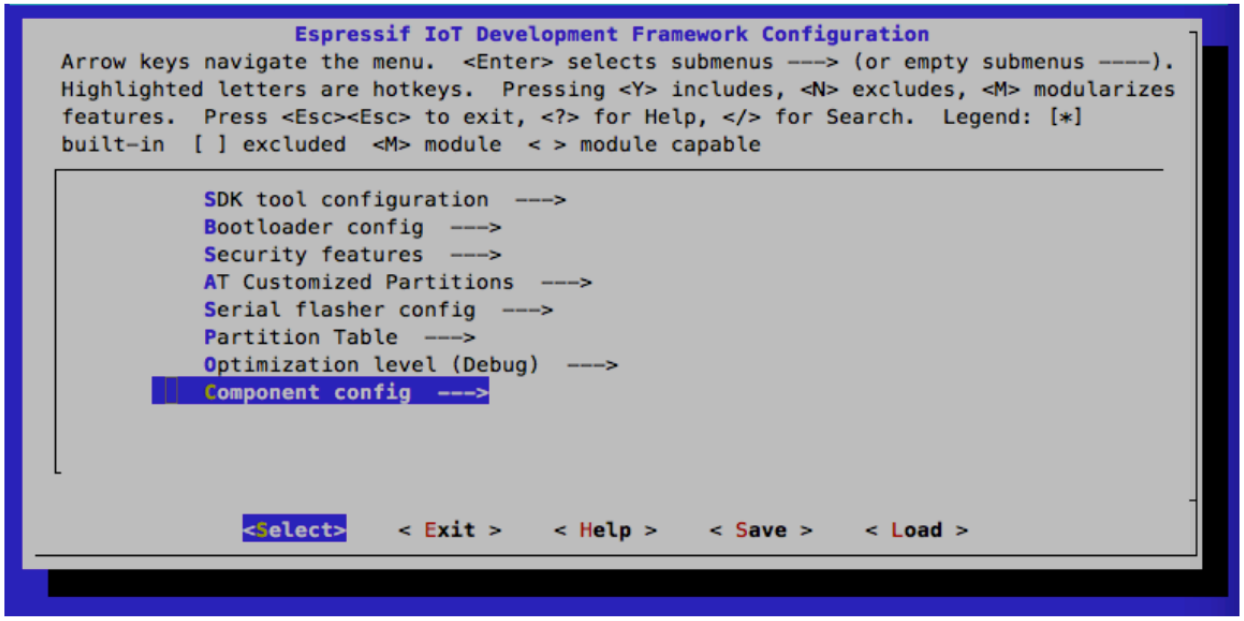


图 8: Configuring the AT OTA token key - Step 1

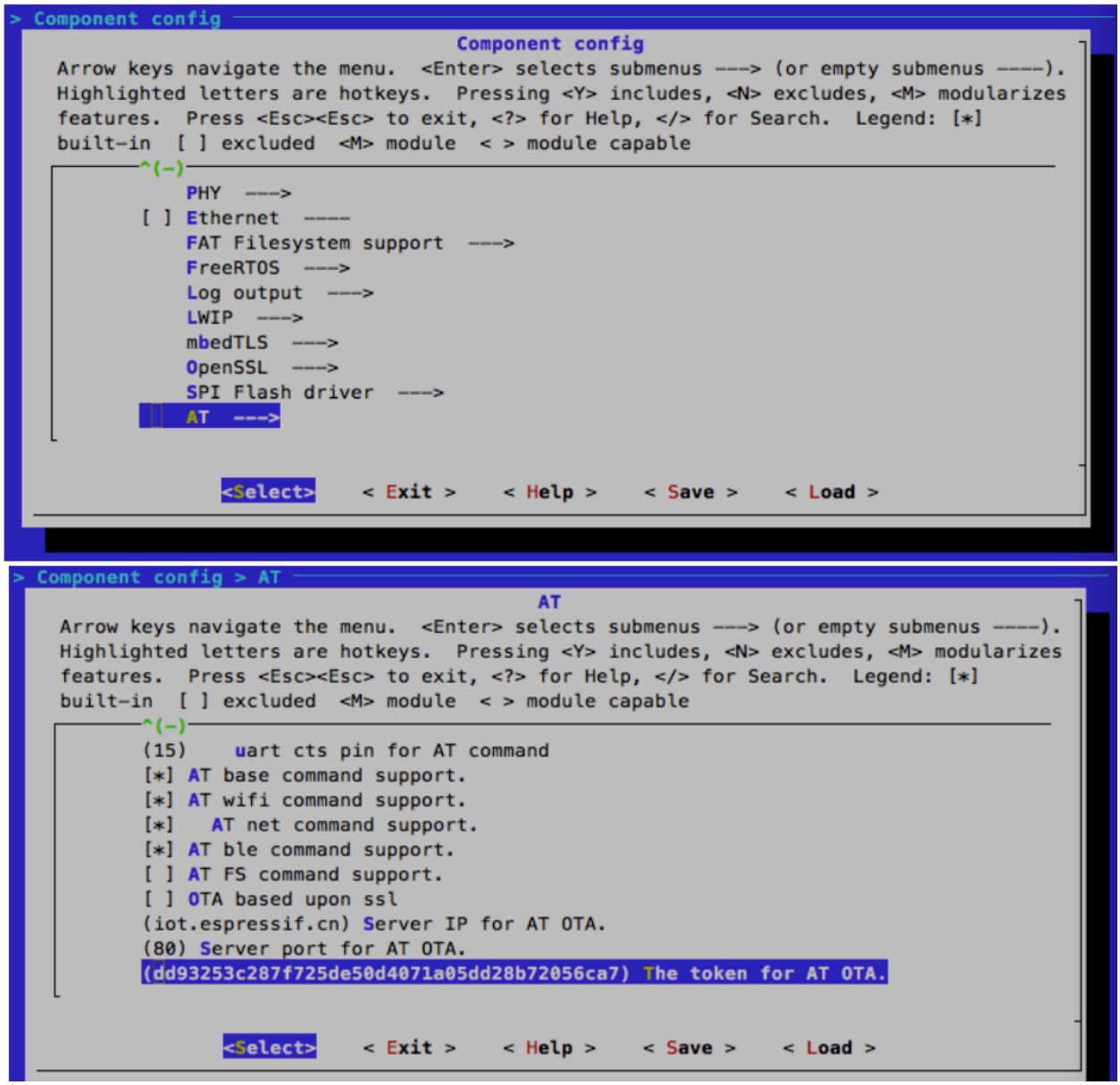


图 9: Configuring the AT OTA token key - Step 2 and 3

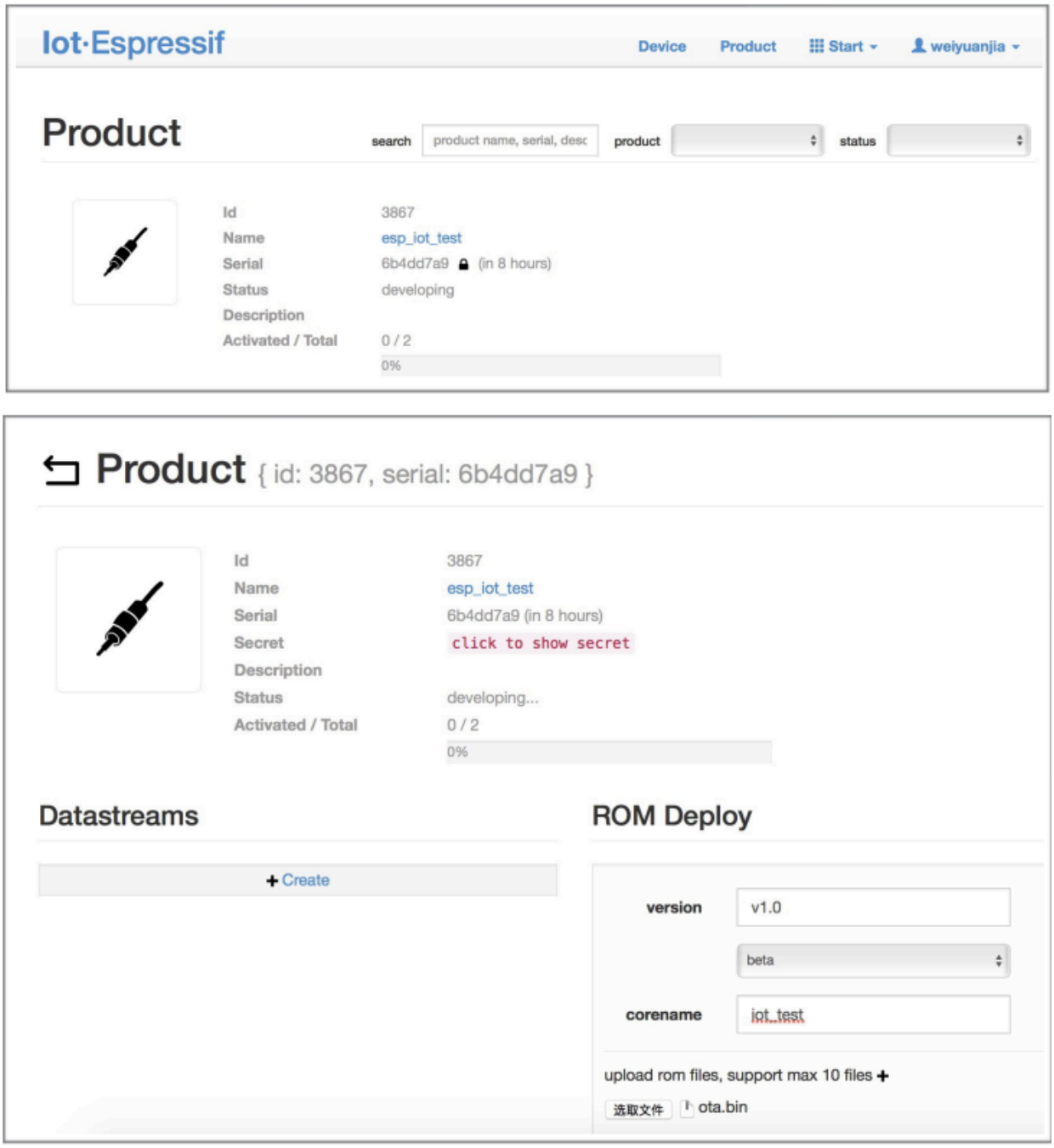


图 10: Enter version and corename

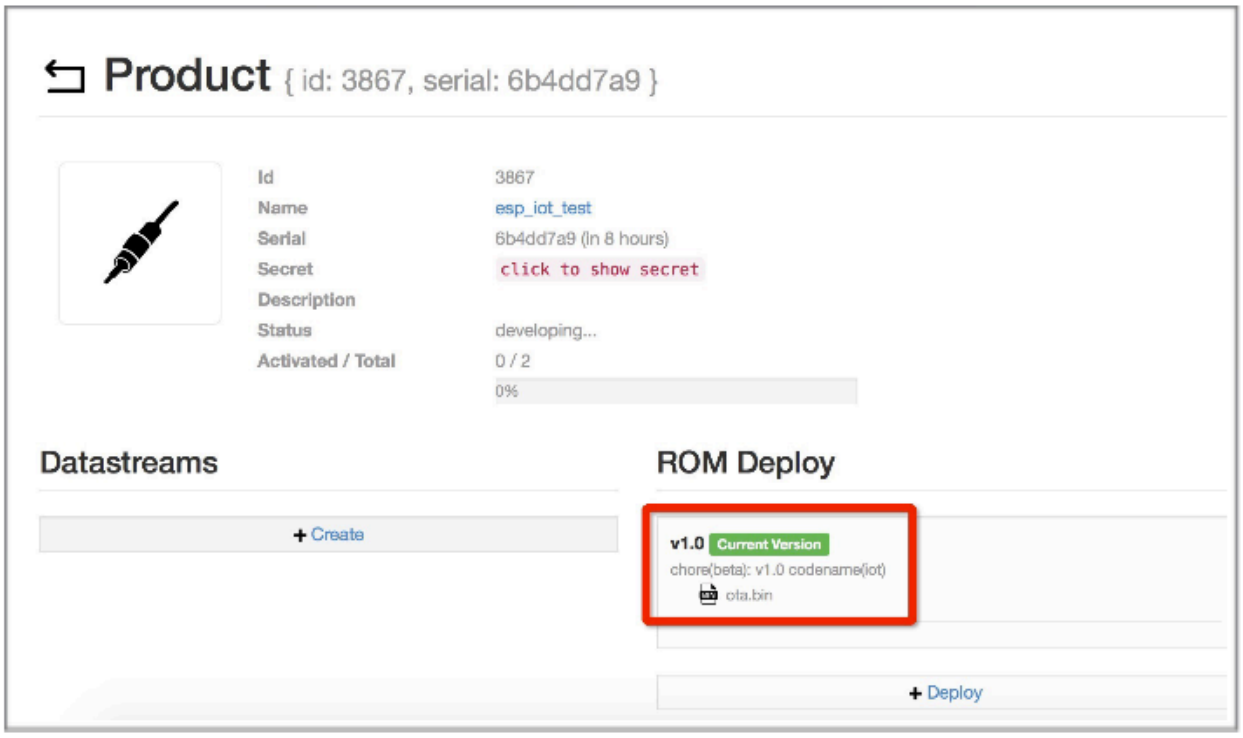


图 11: Save the current version of ota.bin

5.13 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：

```
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。

5.14 如何了解 ESP-AT 同一平台不同模组差异

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

5.15 AT API Reference

5.15.1 Header File

- at/include/esp_at_core.h

5.15.2 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_active_write_data**(uint8_t **data*, int32_t *len*)

call pre_active_write_data_callback() first and then 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_user_cmd_regist**(void)

regist at user command set. If not,you can not use AT user 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_driver_cmd_regist**(void)

regist at driver command set. If not,you can not use AT driver 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);
```

(下页继续)

(续上页)

```
    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

void **at_handle_result_code**(*esp_at_result_code_string_index* _code_, void *_pbuf_)

5.15.3 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_sleep_callback__) (*at_sleep_mode_t* mode)
callback before enter modem sleep and light sleep

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

void (*__pre_restart_callback__) (void)
callback before restart

void (*__pre_active_write_data_callback__) (*at_write_data_fn_t*)
callback before write data

5.15.4 Macros

__at_min__ (x, y)

__at_max__ (x, y)

__ESP_AT_ERROR_NO__ (subcategory, extension)

__ESP_AT_CMD_ERROR_OK__
No Error

__ESP_AT_CMD_ERROR_NON_FINISH__
terminator character not found (“\r\n” expected)

__ESP_AT_CMD_ERROR_NOT_FOUND_AT__
Starting “AT” not found (or at, At or aT entered)

ESP_AT_CMD_ERROR_PARA_LENGTH(which_para)
parameter length mismatch

ESP_AT_CMD_ERROR_PARA_TYPE(which_para)
parameter type mismatch

ESP_AT_CMD_ERROR_PARA_NUM(need, given)
parameter number mismatch

ESP_AT_CMD_ERROR_PARA_INVALID(which_para)
the parameter is invalid

ESP_AT_CMD_ERROR_PARA_PARSE_FAIL(which_para)
parse parameter fail

ESP_AT_CMD_ERROR_CMD_UNSUPPORT
the command is not supported

ESP_AT_CMD_ERROR_CMD_EXEC_FAIL(result)
the command execution failed

ESP_AT_CMD_ERROR_CMD_PROCESSING
processing of previous command is in progress

ESP_AT_CMD_ERROR_CMD_OP_ERROR
the command operation type is error

5.15.5 Type Definitions

typedef int32_t (***at_write_data_fn_t**)(uint8_t *data, int32_t len)

typedef void (***esp_at_port_specific_callback_t**)(void)
AT specific callback type.

5.15.6 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 at_sleep_mode_t
Values:

AT_DISABLE_SLEEP = 0

AT_MIN_MODEM_SLEEP

AT_LIGHT_SLEEP

AT_MAX_MODEM_SLEEP

AT_SLEEP_MAX

enum esp_at_module

module number,Now just AT module

Values:

ESP_AT_MODULE_NUM = 0x01

AT module

enum esp_at_error_code

subcategory number

Values:

ESP_AT_SUB_OK = 0x00

OK

ESP_AT_SUB_COMMON_ERROR = 0x01

reserved

ESP_AT_SUB_NO_TERMINATOR = 0x02

terminator character not found ("\r\n" expected)

ESP_AT_SUB_NO_AT = 0x03

Starting "AT" not found (or at, At or aT entered)

ESP_AT_SUB_PARA_LENGTH_MISMATCH = 0x04

parameter length mismatch

ESP_AT_SUB_PARA_TYPE_MISMATCH = 0x05

parameter type mismatch

ESP_AT_SUB_PARA_NUM_MISMATCH = 0x06

parameter number mismatch

ESP_AT_SUB_PARA_INVALID = 0x07

the parameter is invalid

ESP_AT_SUB_PARA_PARSE_FAIL = 0x08

parse parameter fail

ESP_AT_SUB_UNSUPPORT_CMD = 0x09

the command is not supported

ESP_AT_SUB_CMD_EXEC_FAIL = 0x0A
the command execution failed

ESP_AT_SUB_CMD_PROCESSING = 0x0B
processing of previous command is in progress

ESP_AT_SUB_CMD_OP_ERROR = 0x0C
the command operation type is error

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, just change internal status

ESP_AT_RESULT_CODE_PROCESS_DONE = 0x06
response nothing, just change internal status

ESP_AT_RESULT_CODE_OK_AND_INPUT_PROMPT = 0x07
response nothing, just change internal status

ESP_AT_RESULT_CODE_MAX

5.15.7 Header File

- at/include/esp_at.h

5.15.8 Functions

const char ***esp_at_get_current_module_name**(void)

get current module name

const char ***esp_at_get_module_name_by_id**(uint32_t *id*)

get module name by index

uint32_t **esp_at_get_module_id**(void)

get current module id

void **esp_at_board_init**(void)

init peripheral and default parameters in factory_param.bin

bool **esp_at_web_server_cmd_regist**(void)

regist WiFi config via web command. If not,you can not use web server to config wifi connect

5.15.9 Macros

ESP_AT_PORT_TX_WAIT_MS_MAX

ESP_AT_FACTORY_PARAMETER_SIZE

CHAPTER 6

第三方定制化 AT 命令和固件

6.1 腾讯云 IoT AT 命令和固件

6.1.1 腾讯云 IoT AT 命令集

本文档主要介绍腾讯云 IoT AT 命令、错误码及应用说明，仅针对 ESP8266 设备，下表为本文档的目录。

- 说明
  - 术语解释
  - 符号说明
  - *ESP-AT* 命令说明
- *TC* 设备信息设置命令
  - *AT+TCDEVINFOSET*：平台设备信息设置
  - *AT+TCPRDINFOSET*：平台产品信息设置
  - *AT+TCDEVREG*：执行设备动态注册
  - *AT+TCMODULE*：模组信息读取
  - *AT+TCRESTORE*：清除模组设备信息
- *TC MQTT* 命令

  - *AT+TCMQTTCONN*：配置 *MQTT* 连接参数
  - *AT+TCMQTTDISCONN*：断开 *MQTT* 连接
  - *AT+TCMQTTPUB*：向某个 *Topic* 发布消息
  - *AT+TCMQTTPUBL*：向某个 *Topic* 发布长消息
  - *AT+TCMQTTPUBRAW*：向某个 *Topic* 发布二进制数据消息
  - *AT+TCMQTTSUB*：订阅 *MQTT* 某个 *Topic*
  - *AT+TCMQTTUNSUB*：取消已经订阅的 *Topic*
  - *AT+TCMQTTSTATE*：查询 *MQTT* 连接状态
- *TC OTA* 命令
  - *AT+TCOTASET*：*OTA* 功能使能控制及版本设置
  - *AT+TCFWINFO*：读取模组缓存的固件信息
  - *AT+TCREADFWDATA*：读取模组缓存的固件数据
  - 模组配合腾讯云 *IoT* 平台进行 *OTA* 功能流程框图
- *URC* 模组主动上报 *MCU* 消息
  - *+TCMQTTRCVPUB*（收到订阅的 *Topic* 时上报的消息）
  - *+TCMQTTDISCON*（*MQTT* 断开时上报的信息）
  - *+TCMQTTRECONNECTING*（*MQTT* 正在重连时上报的信息）
  - *+TCMQTTRECONNECTED*（*MQTT* 重连成功时上报的信息）
  - *+TCOTASTATUS*（上报 *OTA* 状态）
- *Wi-Fi* 配网及 *AT* 辅助命令
  - *AT+TCSTARTSMART*：以 *SmartConfig* 方式进行 *Wi-Fi* 配网及设备绑定
  - *AT+TCSTOPSMART*：退出 *SmartConfig* 方式 *Wi-Fi* 配网状态
  - *AT+TCSAP*：以 *softAP* 方式进行 *Wi-Fi* 配网及设备绑定
  - *AT+TCSTOPSAP*：退出 *softAP* 方式 *Wi-Fi* 配网状态
  - *AT+TCMODINFOSET*：*ESP* 模组信息设置
  - *AT+TCMQTTSRV*：设置腾讯云 *MQTT* 服务器地址
  - *AT+TCVER*：读取模组固件 *IoT SDK* 版本信息
- *TC* 网关子设备命令
  - *AT+TCGWBIND*：网关绑定子设备命令
  - *AT+TCGWONLINE*：网关代理子设备上下线命令

  - *AT+TCSUBDEVINFOSET*：子设备信息设置
  - *AT+TCSUBDEVPRDSET*：子设备产品信息设置
  - *AT+TCSUBDEVREG*：执行子设备动态注册
- 错误码
  - 服务端相关 *err code*
  - *CME ERROR* 列表扩展
  - 设备动态注册错误码
  - 模组配网及设备绑定错误类型
  - 网关子设备命令相关错误类型
- 应用说明
  - 密钥认证方式连接腾讯云 *MQTT* 服务器
  - 订阅消息
  - 发布消息
  - 数据通讯应用协议
  - 使用建议

说明

术语解释

术语	含义
MQTT	一种基于轻量级代理的 Pub/Sub 模型的消息传输协议
MCU	微控制单元，一般为通讯模组的上位机
Topic	主题，Pub/Sub 模型中消息的通信媒介，Pub/Sub 必须要有主题，只有当订阅了某个主题后，才能收到相应主题数据信息，才能进行通信
Pub	设备端的发布协议，意思是往 Topic 中发布消息
Sub	设备端的订阅协议，意思是从 Topic 中订阅消息
URC	全称 Unsolicited Result Code，非请求结果码，一般为模组给 MCU 的串口返回

更多信息请参考 腾讯物联网通信词汇表 及其它相关文档。

符号说明

1. 本文档所有语法声明中（包括测试命令、读取命令、设置命令），所有形如 "xxx" 的双引号引注信息，都是确定内容的信息，例：

- 命令

```
AT+TCDEVINFOSET=?
```

- 响应

```
+TCDEVINFOSET: "TLSMODE (0/1/2)","PRODUCTID","DEVICENAME"[,"DEVICESECRET"]

OK
```

"ProductId"、"DeviceName" 等指确定的字符串 “ProductId”、” DeviceName”

2. 本文档所有语法声明中（包括测试命令、读取命令、设置命令），所有形如 <xxx> 的尖角括号引注信息，都是指变量信息，例：

- 命令

```
AT+TCDEVINFOSET?
```

- 响应

```
+TCDEVINFOSET: <tlsmode>,<productId>,<devicename>,[,<devicesecret>]

OK
```

<productId>、<devicename> 等参数指实际的产品 ID 和设备名称，如 CTQS08Y5LG、"Dev01"

3. 在表示具体的数据时，字符串类型和枚举类型的数据需要由双引号 "xx" 引注，数值型数据直接以数据表示。例：

- 命令

```
AT+TCCERTADD="cdev_cert.crt",1428
```

- 响应

```
OK
>
+TCCERTADD: OK
```

1428 表示数值型数据，"cdev_cert.crt" 表示字符串型，建议用户参照示例编写程序。

4. 关于空格，只有回码的冒号和信息之间有一个空格，其他都没有空格。

5. 校验和 (BCC) 生成方法，返回十进制校验和：

```
int CalcCheck(BYTE* Bytes, int len){
      int i, result;
      for (result = Bytes[0], i = 1; i < len ; i++){
              result ^= Bytes[i];
      }
      return result;
}
```

ESP-AT 命令说明

ESP8266 的 AT 命令集及使用说明请参考乐鑫官方 ESP-AT 用户指南 及 GitHub ESP-AT 项目。

对于 ESP-AT 机制，有如下注意事项：

1. 每条 AT 命令总字符长度不可超过 256 字节，否则会报错。
2. 每条 AT 命令都应以 /r/n 为结束符。
3. 如果 AT 命令的参数内容包含了特殊字符如双引号 " 、逗号 , 等，需要加 \ 进行转义，比如 PUB 消息的 payload 采用的 JSON 数据格式为 {"action": "publish_test", "count":"0"}，则应该转义为 {\"action\":\"publish_test\"\,\"count\":\"0\"} 再传入，否则会报错。
4. 如果上一个 AT 命令还没处理完成，再发送新的命令会返回如下错误：

```
ERR CODE:0x010b0000

busy p...
```

TC 设备信息设置命令

AT+TCDEVINFOSET：平台设备信息设置

功能

设置腾讯云物联网平台创建的产品及设备信息

测试命令

命令：

```
AT+TCDEVINFOSET=?
```

响应：

```
+TCDEVINFO:"TLS_MODE (1)","PRODUCT_ID","DEVICE_NAME","DEVICE_SECRET_BCC","PRODUCT_
↪REGION"

OK
```

读取命令

命令：

```
AT+TCDEVINFOSET?
```

响应：

```
+TCDEVINFOSET:<tls_mode>,<product_id>,<device_name>,<devicesecret_checksum>,<product_
↪region>

OK
```

或

```
+CME ERROR: <err>
```

说明：

- ESP8266 模组仅返回 <tls_mode> 为 1，且不返回 devicesecret 的字符串内容，只返回 devicesecret 字符串的校验和 (BCC)

设置命令

命令：

```
AT+TCDEVINFOSET=<tls_mode>,<product_id>,<device_name>,<device_secret>[,<product_
↪region>]
```

响应：

```
OK
```

或

```
+CME ERROR: <err>
```

说明：

- 如果模组已经连接腾讯云 MQTT 服务器，则返回错误，用户需要先发送断开连接命令 (*AT+TCMQTTDISCONN*) 才能执行该命令
- 如果输入合法，首先返回 OK，接下来返回设备信息设置成功与否:
  - `+TCDEVINFOSET:OK`：设置成功
  - `TCDEVINFOSET:FAIL<err_code>`：设置失败

参数

- **<tls_mode>**：接入方式，必填项，ESP8266 模组仅支持模式 1
  - 0：直连模式
  - 1：TLS 密钥方式
  - 2：TLS 证书方式，数值类型
- **<product_id>**：产品 id，必填项，字符串类型，最大长度 10 字节
- **<device_name>**：设备名称，必填项，字符串类型，最大长度 48 字节
- **<device_secret>**：设备密钥，必填项，字符串类型，最大长度 44 字节
- **<product_region>**：产品区域，选填项，字符串类型，最大长度 24 字节，如果不提供，默认为中国大陆公有云“ap-guangzhou”

示例

```
// 设置成功
AT+TCDEVINFOSET=1,"CTQS08Y5LG","Dev01","ZHNkIGRzZCA="
OK
+TCDEVINFOSET:OK
```

AT+TCPRDINFOSET：平台产品信息设置

功能

设置腾讯云物联网平台创建的产品信息，适用于产品级密钥场景

测试命令

命令：

```
AT+TCPRDINFOSET=?
```

响应：

```
+TCPRDINFOSET:"TLS_MODE(1)","PRODUCT_ID","PRODUCT_SECRET_BCC","DEVICE_NAME","PRODUCT_
↪REGION"

OK
```

读取命令

命令：

```
AT+TCPRDINFOSET?
```

响应：

```
+TCPRDINFOSET:<tls_mode>,<product_ID>,<product_secret_checksum>,<device_name>,
↪<product_region>

OK
```

设置命令

命令：

```
AT+TCPRDINFOSET=<tls_mode>,<product_ID>,<product_secret>,<device_name>,<product_
↪region>
```

响应：

```
OK
```

或

```
+CME ERROR: <err>
```

说明：

- 如果模组已经连接腾讯云 MQTT 服务器，则返回错误，用户需要先发送断开连接命令 (*AT+TCMQTTDISCONN*) 才能执行该命令
- 如果输入合法，首先返回 OK，接下来返回设备信息设置成功与否
  - +TCPRDINFOSET:OK：设置成功，产品数据会保存到 flash，掉电不丢失
  - +TCPRDINFOSET:FAIL,<err_code>：设置失败

参数

- **<tls_mode>**：接入方式，必填项
  - 0：直连模式，
  - 1：TLS 密钥方式
  - 2：TLS 证书方式，数值类型
- **<product_ID>**：产品 ID，必填项，字符串类型，最大长度 10
- **<product_secret>**：产品密钥，必填项，字符串类型，最大长度 32
- **<device_name>**：设备名称，必填项，字符串类型，最大长度 48
- **<product_region>**：产品区域，选填项，字符串类型，最大长度 24 字节，如果不提供，默认为中国大陆公有云“ap-guangzhou”

示例

```
AT+TCPRDINFOSET=1,"CTQS08Y5LG","ZHNkIGRzZCA=","Dev01"

OK
+TCPRDINFOSET:OK
```

AT+TCDEVREG：执行设备动态注册

功能

采用产品级密钥场景下，执行设备动态注册并获取设备信息

测试命令

命令：

```
AT+TCDEVREG=?
```

响应：

```
OK
```

执行命令

命令：

```
AT+TCDEVREG
```

响应：

```
OK
```

或

```
+CME ERROR: <err>
```

说明

使用产品级密钥场景下执行动态注册的逻辑说明：

1. 如果模组上面没有完整的设备信息，即设备未注册未激活，则正常注册，返回成功/失败。
2. 模组上已存在一个设备 A，且是已注册未激活状态，如果用户使用*AT+TCPRDINFOSET* 提供的设备信息也是 A，则正常注册，云端会重新分配 PSK 或证书，返回成功/失败。
3. 模组上已存在一个设备 A，且是已注册已激活状态，如果用户使用*AT+TCPRDINFOSET* 提供的设备信息也是 A，则会注册失败，AT 命令返回错误，用户需要更换设备信息或在云端将设备重置。

4. 模组已存在一个设备 A 的信息，如果用户使用*AT+TCPRDINFOSET* 提供了一个新的设备 B 的信息，则会使用新的设备 B 的信息去注册，注册成功则覆盖原来设备 A 的信息，注册失败则原有的设备 A 信息不变。
5. 正常情况下，设备动态注册仅需执行一次，执行成功后，设备密钥信息已经保存在模组 flash 中，后续上电初始化时可通过命令*AT+TCDEVINFOSET?*？查询是否存在正确的设备信息并正常连接腾讯云 MQTT 服务。

示例

```
AT+TCDEVREG

OK
+TCDEVREG:OK
```

AT+TCMODULE：模组信息读取

功能

获取模组相关的硬件及软件信息

执行命令

命令：

```
AT+TCMODULE
```

响应：

```
Module HW name: 模组硬件信息
Module FW version: 模组固件信息
Module Mac addr: ESP8266 Wi-Fi 模组 mac 地址
Module FW compiled time: 模组固件编译生成时间
Module Flash size: 模组 flash 大小
OK
```

示例

```
AT+TCMODULE
Module HW name: ESP-WROOM-02D
Module FW version: QCloud_AT_ESP8266_v2.0.0
Module Mac addr: 3c:71:bf:33:b0:2e
Module FW compiled time: Jun 17 2020 16:25:27
Module Flash size: 2MB
OK
```

AT+TCRESTORE：清除模组设备信息

功能

清除模组 flash 上保存的腾讯云设备信息

测试命令

命令：

```
AT+TCRESTORE=?
```

响应：

```
OK
```

执行命令

命令：

```
AT+TCRESTORE
```

响应：

```
OK
```

或

```
+CME ERROR: <err>
```

说明：

- 如果模组已经连接腾讯云 MQTT 服务器，则返回错误，用户需要先发送断开连接命令 (*AT+TCMQTTDISCONN*) 才能执行该命令。
- 如果状态允许，则返回 OK，然后清除模组上面存储的腾讯云相关设备及产品信息，以及缓存的 OTA 固件信息，并重启模组。
- 该命令不会清除模组信息（即通过*AT+TCMODULE* 可以读取的信息）以及 ESP8266 的 NVS 数据包括 Wi-Fi 配置，如果需要清除 Wi-Fi 配置信息需要执行 AT+RESTORE。

示例

```
AT+TCRESTORE

OK
```

TC MQTT 命令

AT+TCMQTTCONN：配置 MQTT 连接参数

功能

配置 MQTT 连接参数，包括客户端和服务器的心跳间隔、会话控制、并连接腾讯云端服务器

测试命令

命令：

```
AT+TCMQTTCONN=?
```

响应：

```
+TCMQTTCONN:<TLSMODE_SELECTED>,<CMDTIMEOUT_VALUE>,<KEEPALIVE>(max 690s),<CLEAN_
↪SESSION>(0/1),<RECONNECT>(0/1)

OK
```

读取命令

命令：

```
AT+TCMQTTCONN?
```

响应：

```
+TCMQTTCONN:<tlsmode>,<cmdtimeout>,<keepalive>,<clean_session>,<reconnect>

OK
```

说明：

- KEEPALIVE 的默认值为 240，CLEAN_SESSION 的默认值为 1

设置命令

命令：

```
AT+TCMQTTCONN=<tlsmode>,<cmdtimeout>,<keepalive>,<clean_session>,<reconnect>
```

响应：

```
OK
```

或

```
+CME ERR: <err>
```

参数

- **<tlsmode>**：接入方式，必填项，ESP8266 模组仅支持 <tlsmode> 为 1 的模式
  - 0：直连模式
  - 1：TLS 密钥方式
  - 2：TLS 证书方式，整型
- **<cmdtimeout>**：命令超时时间，必填项，整型，MQTT 连接、发布、订阅的超时时间，单位毫秒，建议设置为 5000，可以根据网络环境调整该值。范围为 1000 ~ 10000 毫秒
- **<keepalive>**：心跳间隔，必填项，整型，范围 60 ~ 690 秒，默认值为 240
- **<clean_session>**：是否清除会话，必填项，整型
  - 0：不清除

  - 1：清除（默认）

- **<reconnect>**：MQTT 断连后是否重连，必填项，整型
  - 0：不自动重连
  - 1：自动重连
- 该命令前置依赖*AT+TCDEVINFOSET* 命令

示例

```
AT+TCMQTTCONN=1,5000,240,1,1

OK
+TCMQTTCONN:OK
```

AT+TCMQTTDISCONN：断开 MQTT 连接

功能

断开与腾讯云的 MQTT 连接

测试命令

命令：

```
AT+TCMQTTDISCONN=?
```

响应：

```
OK
```

执行命令

命令：

```
AT+TCMQTTDISCONN
```

响应：

```
OK
```

或

```
+CME ERROR: <err>
```

说明：

- 如果模组处于 OTA 状态中，执行该命令会先取消 OTA 后台任务再断开 MQTT 连接
- 未连接状态下返回 `+CME ERROR: <err>`

示例

```
AT+TCMQTTDISCONN

OK
```

AT+TCMQTTPUB：向某个 Topic 发布消息

功能

向某个 Topic 发布消息

测试命令

命令：

```
AT+TCMQTTPUB=?
```

响应：

```
+TCMQTTPUB: "TOPIC_NAME(maxlen 128)", "QOS(0/1)","PAYLOAD"

OK
```

设置命令

命令：

```
AT+TCMQTTPUB=<topic>,<qos>,<message>
```

响应：

```
OK
```

或

```
+CME ERR: <err>
```

说明：

- 如果模组尚未连接腾讯云 MQTT 服务器，则返回错误，用户需要先发送连接命令 (*AT+TCMQTTCONN*) 才能发布消息。
- 如果模组处于 OTA 下载状态中，由于 ESP8266 平台资源限制，执行该命令可能会出现超时错误。如非必要，不建议在 OTA 下载过程中执行该命令。
- 如果输入合法，首先返回 OK，接下来返回消息发布成功与否。如果是 QoS1 消息，会等到收到 PUBACK 或超时失败再返回。
  - `+TCMQTTPUB: OK`：发布成功
  - `+TCMQTTPUB: FAIL,<err_code>`：发布失败

参数

- **<topic>**：发布消息的 Topic name，字符串最大长度 128
- **<qos>**：QoS 值，暂只支持 0 和 1
- **<message>**：发布的消息体的内容

说明

- 注意每条 AT 命令总字符长度不可超过 256 字节，否则会报错，关于消息体内容格式及长度请参考*ESP-AT 命令说明* 章节。

示例

```
// 消息发布成功
AT+TCMQTTPUB="iot-ee54phlu/device1/get",1,"hello world"

OK
+TCMQTTPUB: OK
```

AT+TCMQTTPUBL：向某个 Topic 发布长消息

功能

向某 Topic 发布长消息，用于*AT+TCMQTTPUB* 消息体长度较大场景

测试命令

命令：

```
AT+TCMQTTPUBL=?
```

响应：

```
+TCMQTTPUBL: "TOPIC_NAME(maxlen 128)", "QOS(0/1)","LEN(1-2048)"

OK
```

设置命令

命令：

```
AT+TCMQTTPUBL=<topic>,<qos>,<msg_length>
```

响应：

```
OK
>
```

或

```
+CME ERR:<err>
```

说明：

- 如果模组尚未连接腾讯云 MQTT 服务器，则返回错误，用户需要先发送连接命令 (*AT+TCMQTTCONN*) 才能发布消息。
- 如果模组处于 OTA 下载状态中，由于内存资源限制，不支持该发布消息命令，会返回错误。
- 如果输入合法，首先返回 OK，接下来返回 >，进入接收消息 payload 状态，读到 <msg_length> 长度的数据后，结束接收并返回发送 MQTT 消息结果：
  - +TCMQTTPUBL:OK：发布成功
  - +TCMQTTPUBL:FAIL,<err_code>：发布失败

- 进入接收消息 payload 状态后，有 20 秒钟左右的超时时间，如果超时后收到的数据长度小于 <msg_length>，或者收到 +++\r\n，则退出接收消息 payload 状态，返回错误 +CME ERR:<err>，并且不会发送该 MQTT 消息。
- 消息 payload 不会回显。

参数

- **<topic>**：发布消息的 Topic name，最大字符串长度 128
- **<qos>**：QoS 值，暂只支持 0 和 1
- **<msg_length>**：发布的消息体的长度，最大长度 2048。该长度不包括结尾的 /r/n，关于消息体内容格式请参考本文档*ESP-AT 命令说明* 章节

示例

```
// 消息发布成功
AT+TCMQTTPUBL="iot-ee54phlu/device1/get",1,11
>

Hello,world
OK

+TCMQTTPUBL: OK
```

AT+TCMQTTPUBRAW：向某个 Topic 发布二进制数据消息

功能

向某 Topic 发布二进制数据消息，可以发布自定义的任意数据而非文本或者 JSON 数据，模组透传不做任何转义处理。

测试命令

命令：

```
AT+TCMQTTPUBRAW=?
```

响应：

```
+TCMQTTPUBRAW: "TOPIC_NAME(maxlen128)", "QOS(0/1)","LEN(1-2048)"

OK
```

设置命令

命令：

```
AT+TCMQTTPUBRAW=<topic>,<qos>,<msg_length>
```

响应：

```
OK
>
```

或

```
+CME ERR:<err>
```

说明：

- 如果模组尚未连接腾讯云 MQTT 服务器，则返回错误，用户需要先发送连接命令 (*AT+TCMQTTCONN*) 才能发布消息。
- 如果模组处于 OTA 下载状态中，由于内存资源限制，不支持该发布消息命令，会返回错误。
- 如果输入合法，首先返回 `OK`，接下来返回 `>`，进入接收消息 `payload` 状态，读到 `<msg_length>` 长度的数据后，结束接收并返回发送 MQTT 消息结果：
  - `+TCMQTTPUBRAW:OK`：发布成功
  - `+TCMQTTPUBRAW:FAIL,<err_code>`：发布失败
- 进入接收消息 payload 状态后，有 20 秒钟左右的超时时间，如果超时后收到的数据长度小于 `<msg_length>`，或者收到 `+++\r\n`，则退出接收消息 payload 状态，返回错误 `+CME ERR:<err>`，并且不会发送该 MQTT 消息。
- 消息 payload 不会回显。

参数

- **<topic>**：发布消息的 Topic name，最大字符串长度 128
- **<qos>**：QoS 值，暂只支持 0 和 1
- **<msg_length>**：发布的消息体的长度，最大长度 2048，该长度不包括结尾的 /r/n

示例

```
// 消息发布成功
AT+TCMQTTPUBRAW="$thing/up/raw/iot-ee54phlu/device1",1,10
>

 0x0102030405060708090A
 OK

 +TCMQTTPUBRAW: OK
```

AT+TCMQTTSUB：订阅 MQTT 某个 Topic

功能

订阅 MQTT 某个 Topic，Wi-Fi 模组最多支持订阅 10 个 Topic

测试命令

命令：

```
AT+TCMQTTSUB=?
```

响应：

```
+TCMQTTSUB:"TOPIC_NAME(maxlen 128)","QOS(0/1)"

OK
```

读取命令

命令：

```
AT+TCMQTTSUB?
```

响应：

```
OK
```

或

```
+TCMQTTSUB: <topic>,<qos>
:
:list of sub topic
+TCMQTTSUB: <topic_n>,<qos>

OK
```

说明：

- 如果有已经订阅的消息，返回已订阅的 Topic 列表

设置命令

命令：

```
AT+TCMQTTSUB=<topic>,<qos>
```

响应：

```
OK
```

或

```
+CME ERROR:<err>
```

说明：

- 如果模组尚未连接腾讯云 MQTT 服务器，则返回错误，用户需要先发送连接命令 (*AT+TCMQTTCONN*) 才能订阅消息。
- 如果模组处于 OTA 下载状态中，不支持该命令，会返回错误。

- 如果模组尚未连接腾讯云 MQTT 服务器，则返回错误，用户需要先发送连接命令 (*AT+TCMQTTCONN*) 才能订阅消息。
- 如果模组处于 OTA 下载状态中，不支持该命令，会返回错误。

- 如果输入合法，首先返回 OK，然后返回订阅成功与否，该命令会等到收到 SUBACK 或超时失败再返回。
  - +TCMQTTSUB:OK：订阅成功
  - +TCMQTTSUB:FAIL,<err_code>：订阅失败

参数

- **<topic>**：订阅的 Topic name，最大长度 128
- **<qos>**：QoS 值，暂只支持 0 和 1

示例

```
AT+TCMQTTSUB="iot-ee54phlu/device1/control",0

OK
+TCMQTTSUB: OK
```

AT+TCMQTTUNSUB：取消已经订阅的 Topic

功能

取消已订阅的 Topic

测试命令

命令：

```
AT+TCMQTTUNSUB=?
```

响应：

```
+TCMQTTUNSUB: "TOPIC_NAME"

OK
```

读取命令

命令：

```
AT+TCMQTTUNSUB?
```

响应：

```
OK
```

设置命令

命令：

```
AT+TCMQTTUNSUB=<topic>
```

响应：

```
OK
```

或

```
+CME ERROR:<err>
```

说明：

- 如果模组尚未连接腾讯云 MQTT 服务器，则返回错误，用户需要先发送连接命令 (*AT+TCMQTTCONN*) 才能订阅消息。
- 如果模组处于 OTA 下载状态中，不支持该命令，会返回错误。
- 如果输入合法，首先返回 `OK`，然后返回取消订阅成功与否：
  - `+TCMQTTUNSUB:OK`：取消订阅成功；
  - `+TCMQTTUNSUB:FAIL,<err_code>`：取消订阅失败。

参数

- **<topic>**：取消订阅的 Topic

AT+TCMQTTSTATE：查询 MQTT 连接状态

功能

查询 MQTT 连接状态

测试命令

命令：

```
AT+TCMQTTSTATE=?
```

响应：

```
OK
```

读取命令

命令：

```
AT+TCMQTTSTATE?
```

响应：

```
+TCMQTTSTATE: <state>

OK
```

参数

- **<state>**：MQTT 连接状态
  - 0：MQTT 已断开
  - 1：MQTT 已连接

示例

```
AT+TCMQTTSTATE?

+TCMQTTSTATE: 1
OK
```

TC OTA 命令

AT+TCOTASET：OTA 功能使能控制及版本设置

功能

OTA 功能使能控制及版本设置

测试命令

命令：

```
AT+TCOTASET=?
```

响应：

```
+TCOTASET: 1(ENABLE)/0(DISABLE),"FW_version"

OK
```

读取命令

命令：

```
AT+TCOTASET?
```

响应：

```
OK
+TCOTASET: <ctlstate>,<fw_ver>
```

或

```
+CME ERROR:<err>
```

设置命令

命令：

```
AT+TCOTASET=<ctlstate>,<fw_ver>
```

响应：

```
OK
```

或

```
+CME ERROR:<err>
```

说明：

- 如果输入合法，模组会先返回 OK，然后订阅 OTA 的 Topic（用户无须手动订阅 Topic），启动 OTA 后台任务，并上报本地版本，返回执行结果。如果后台任务已经启动并且不处于下载状态，则执行该命令会再次上报本地固件版本。如果已经在 OTA 下载状态中，执行该命令则会返回错误。
- 该命令执行成功之后，模组会处于监听升级命令状态，这个时候如果用户通过控制台下发升级固件的命令，模组解析命令成功之后会进入 OTA 下载状态并上报 +TCOTASTATUS:ENTERUPDATE 的 URC 给 MCU。进入 OTA 下载状态之后，会禁用部分 AT 命令，直到固件下载结束。
- 当固件下载结束，成功会上报 +TCOTASTATUS:UPDATESUCCESS，失败会上报 +TCOTASTATUS:UPDATEFAIL，并退出后台任务。这个时候需要再次执行该命令，才会重新启动后台下载任务。
- 固件下载支持断点续传，异常失败重新启动后，已下载部分无需重新下载。
- 通过该命令启动固件升级任务，会支持 MCU 测固件下载以及模组自身的固件升级。对于模组自身的固件升级，在固件下载成功之后会上报 +TCOTASTATUS:UPDATERESET，并在 2 秒后自动重启进入新固件。
  - +TCOTASET:OK: OTA 功能设置 OK
  - +TCOTASET:FAIL,<err_code>：OTA 功能设置失败

参数

- **<ctlstate>**：OTA 使能控制，布尔型，0 关闭，1 使能。enable 上报本地版本并启动后台下载任务；disable 则取消后台下载任务
- **<fw_ver>**：系统当前固件版本信息，字符型，版本格式：V.R.C，譬如 1.0.0. 长度 1 ~ 32 字节

示例

```
AT+TCOTASET=1,"1.0.1"
OK
+TCOTASET:OK
```

AT+TCFWINFO：读取模组缓存的固件信息

功能

读取模组缓存的固件信息

测试命令

命令：

```
AT+TCFWINFO=?
```

响应：

```
+TCFWINFO: "FW_VERSION","FW_SIZE","FW_MD5","FW_MAX_SIZE_OF_MODULE"

OK
```

说明：

- FW_MAX_SIZE_OF_MODULE 是用户待升级的 OTA 固件的最大字节数，模组根据自身资源情况返回，最小必须是 128 KB

读取命令

命令：

```
AT+TCFWINFO?
```

响应：

```
OK
+TCFWINFO:<fw_verion>,<fw_size>,<fw_md5>,<module_buffer_size>
```

或

```
+CME ERROR:<err>
```

说明：

- 每执行一次固件信息读取，已读取的固件数据偏移位置初始化为 0
- 如果已经在 OTA 下载状态中，则返回错误

示例

```
AT+TCFWINFO？

OK
+TCFWINFO:"2.0.0",516360,"93412d9ab8f3039caed9667a1d151e86"
```

AT+TCREADFWDATA：读取模组缓存的固件数据

功能

读取模组缓存的固件数据

测试命令

命令：

```
AT+TCREADFWDATA=?
```

响应：

```
+TCREADFWDATA: "LEN_FOR_READ"
OK
```

设置命令

命令：

```
AT+TCREADFWDATA=<len>
```

响应：

```
+CME ERROR:<err>
```

或

```
+TCREADFWDATA:len,hexdata…
```

说明：

- 每读一次，模组实现偏移累加，用户需要根据固件大小判断是否读取完毕。如果 AT 返回成功，但返回的长度小于要读取的长度，则表示固件已经读取到尽头。用户再次读取会返回错误，需要发起*AT+TCFWINFO* 命令将偏移量清零，才可以重新开始读取固件。
- 如果正在 OTA 下载状态中，则返回错误。

参数

- **<len>**：读取的固件长度，整型

示例

```
AT+TCREADFWDATA=512
OK
+TCREADFWDATA:512,01020AF5…..
```

模组配合腾讯云 IoT 平台进行 OTA 功能流程框图

URC 模组主动上报 MCU 消息

+TCMQTTRCVPUB（收到订阅的 Topic 时上报的消息）

功能

收到订阅的 Topic 的消息时上报给 MCU 的信息

消息格式

```
+TCMQTTRCVPUB: <topic>,<message_len>,<message>
```

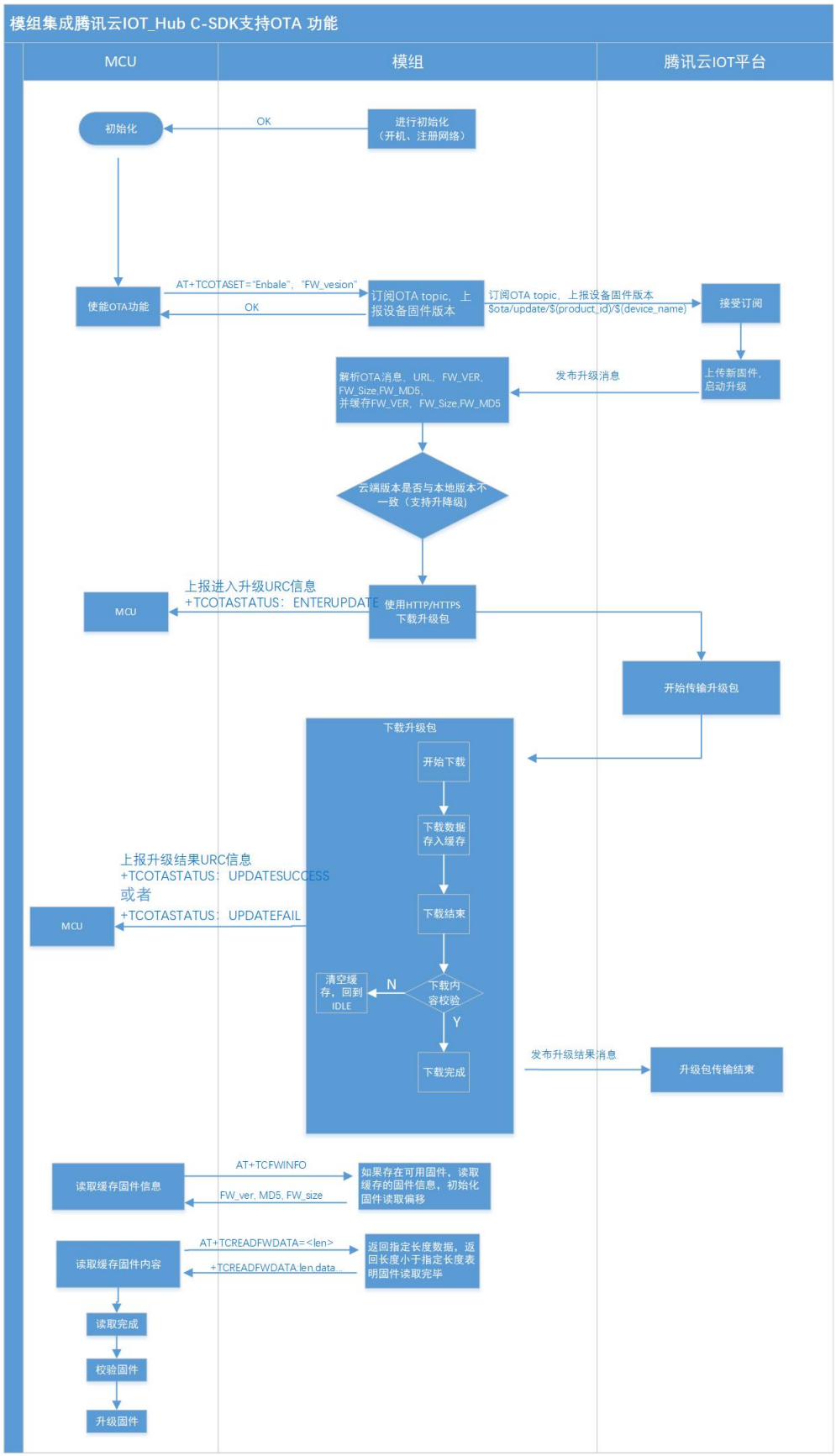


图 1: 模组配合腾讯云 IoT 平台进行 OTA 功能流程框图

参数

- **<topic>**：收到消息的 Topic
- **<message_len>**：数值型，收到消息体的长度（不含 ""）
- **<message>**：收到消息体的内容

说明

- 模组不区分下行数据是二进制数据还是字符串数据，所以 message 内容统一加 ""，如果订阅的 Topic 下行的数据是二进制（Topic 下行的数据是字符串还是二进制，开发者自己需要清楚），需要注意去掉首部和尾部的 "。譬如下示例，Topic $thing/down/raw/CTQS08Y5LG/Dev01 下行二进制数据 1234 为 0x1234，两个字节，是非可见字符，串口工具看到是乱码。

示例

```
+TCMQTTRCVPUB:"CTQS08Y5LG/Dev01/get",11,"hello world"
```

+TCMQTTDISCON（MQTT 断开时上报的信息）

功能

MQTT 连接与服务器断开时上报的 URC 及断开的错误码

说明

- Code 错误码详情可以查询服务端相关 *err code*

示例

```
+TCMQTTDISCON,<err_code>
```

+TCMQTTRECONNECTING（MQTT 正在重连时上报的信息）

功能

MQTT 连接与服务器断开并正在进行自动重连时候上报的 URC

示例

```
+TCMQTTRECONNECTING
```

+TCMQTTRECONNECTED（MQTT 重连成功时上报的信息）

功能

MQTT 连接与服务器断开后自动重连成功时上报的 URC

示例

```
+TCMQTTRECONNECTED
```

+TCOTASTATUS（上报 OTA 状态）

功能

OTA 状态发生变化时上报的 URC

消息格式

```
+TCOTASTATUS: <state>
```

参数

- **<state>**：OTA 状态
  - ENTERUPDATE：模组进入 OTA 固件下载状态
  - UPDATESUCCESS：固件下载成功（包括固件校验和缓存成功）
  - UPDATEFAIL,<err_code>：固件下载失败

- UPDATERESET: 模组自身固件升级成功，在 2 秒后会自动重启

示例

```
+TCOTASTATUS: UPDATESUCCESS
```

Wi-Fi 配网及 AT 辅助命令

AT+TCSTARTSMART：以 SmartConfig 方式进行 Wi-Fi 配网及设备绑定

功能

以 SmartConfig 方式进行 Wi-Fi 配网及腾讯云设备绑定，需要与腾讯连连小程序配合完成。目前仅支持乐鑫 ESP-TOUCH 方式。具体配网协议请参考腾讯云物联网开发平台官网文档。

测试命令

命令：

```
AT+TCSTARTSMART=?
```

响应：

```
AT+TCSTARTSMART: CMD FOR START SMARTCONFIG
OK
```

执行命令

命令：

```
AT+TCSTARTSMART
```

响应：

首先返回

```
OK
```

或

```
+CME ERROR: <err>
```

然后启动配网及绑定后台任务，并返回

```
+TCSTARTSMART:OK      // 进入配网状态成功
```

或

```
+TCSTARTSMART:FAIL,<err_code>     // 进入配网状态失败
```

在配网及绑定操作成功之后返回

```
+TCSTARTSMART:WIFI_CONNECT_SUCCESS
```

或

```
+TCSTARTSMART:WIFI_CONNECT_FAILED, <err_code,sub_code>
```

说明：

- 如果模组处于 MQTT 已连接状态中，则不支持该设置命令，会返回错误。需要先断开 MQTT 连接。
- 该命令执行成功后，蓝色 Wi-Fi 指示灯会进入 500 ms 为周期的闪烁状态，这个时候执行腾讯连连小程序上面的添加设备操作并按照指示进行。
- 如果在 5 分钟内没有执行操作，模组自动退出配网状态，并返回超时错误：`+TCSTARTSMART:FAIL,202`。

示例

```
AT+TCSTARTSMART

OK
+TCSTARTSMART:WIFI_CONNECT_SUCCESS
```

AT+TCSTOPSMART：退出 SmartConfig 方式 Wi-Fi 配网状态

功能

退出 SmartConfig 方式配网状态

测试命令

命令：

```
AT+TCSTOPSMART=?
```

响应：

```
AT+TCSTOPSMART: CMD TO STOP SMARTCONFIG
OK
```

执行命令

命令：

```
AT+TCSTOPSMART
```

响应：

```
OK
```

或

```
+CME ERROR: <err>
```

说明：

- 如果模组处于 MQTT 已连接状态中，则不支持该设置命令，会返回错误。需要先断开 MQTT 连接。

示例

```
AT+TCSTOPSMART

OK
```

AT+TCSAP：以 softAP 方式进行 Wi-Fi 配网及设备绑定

功能

以 softAP 方式进行 Wi-Fi 配网及腾讯云设备绑定，需要与腾讯连连小程序配合完成。具体配网协议请参考腾讯云物联网开发平台官网文档。

测试命令

命令：

```
AT+TCSAP=?
```

响应：

```
+TCSAP=<ssid>[,<pwd>,<ch>]

OK
```

读取命令

命令：

```
AT+TCSAP?
```

响应：

```
OK
```

设置命令

命令：

```
AT+TCSAP=<ssid>[,<pwd>,<ch>]
```

响应：

首先返回

```
OK
```

或

```
+CME ERROR: <err>
```

然后启动配网及绑定后台任务，并返回

```
+TCSAP:OK    // 进入配网状态成功
```

或

```
+TCSAP:FAIL<err_code>     // 进入配网状态失败
```

在配网及绑定操作成功之后返回

```
+TCSAP:WIFI_CONNECT_SUCCESS
```

否则返回

```
+TCSAP:WIFI_CONNECT_FAILED,<err_code,sub_code>
```

说明：

- 如果模组处于 MQTT 已连接状态中，则不支持该设置命令，会返回错误。需要先断开 MQTT 连接。
- 该命令执行成功后，蓝色 Wi-Fi 指示灯会进入 200 ms 为周期的闪烁状态，这个时候执行腾讯连连小程序上面的添加设备操作并按照指示进行。
- 如果在 5 分钟内没有执行操作，模组自动退出配网状态，并返回超时错误：`+TCSAP:FAIL,202`。

参数

- **<ssid>**：热点 ssid，设备作为 softAP 时 ssid，最大长度 32 字节
- **<pwd>**：热点密码，设备作为 softAP 时 psw，最大长度 32 字节，可选参数
- **<ch>**：热点信道，设备作为 softAP 时的信道，可选参数

说明

- 下发此命令后，可以搜索到所配置的 ssid 的热点，手机可以按配置的密码选择连接此热点，模组同时会起一个 UDP server，serverip:192.168.4.1。
- APP 和模组的配网可进行交互数据流。
- 如果只提供 ssid，则会启动无加密的 Wi-Fi 热点。

示例

```
AT+TCSAP="Test-SoftAP","12345678"

OK
+TCSAP:WIFI_CONNECT_SUCCESS
```

AT+TCSTOPSAP：退出 softAP 方式 Wi-Fi 配网状态

功能

退出 softAP 方式配网状态

测试命令

命令：

```
AT+TCSTOPSAP=?
```

响应：

```
AT+TCSTOPSAP: CMD TO STOP SOFTAP
OK
```

执行命令

命令：

```
AT+TCSTOPSAP
```

响应：

```
OK
```

或

```
+CME ERROR: <err>
```

说明：

- 如果模组处于 MQTT 已连接状态中，则不支持该设置命令，会返回错误。需要先断开 MQTT 连接。

示例

```
AT+TCSTOPSAP

OK
```

AT+TCMODINFOSET：ESP 模组信息设置

功能

设置 ESP8266 模组相关的信息，如模组名称，flash 大小等

测试命令

命令：

```
AT+TCMODINFOSET?
```

响应：

```
+TCMODINFOSET:"MODULE NAME","FLASH_SIZE (2/4)","WIFI LED GPIO","FW BASE ADDR","FW MAX␣
↪SIZE","FIXED CONNID"

OK
```

读取命令

命令：

```
AT+TCMODINFOSET?
```

响应：

```
+TCMODINFOSET:<module_name>,<flash_size>,<WiFi_LED_GPIO>,<fw_base_addr>,<fw_max_size>,
↪<fixed_conn_id>

OK
```

设置命令

命令：

```
AT+TCMODINFOSET=<module_name>,<flash_size>,<WiFi_LED_GPIO>,<fw_base_addr>,<fw_max_
↪size>,<fixed_conn_id>
```

响应：

```
OK
```

或

```
+CME ERROR: <err>
```

说明：

- 如果模组已经连接腾讯云 MQTT 服务器，则返回错误，用户需要先发送断开连接命令 (*AT+TCMQTTDISCONN*) 才能执行该命令。
- 如果输入合法，首先返回 `OK`，接下来返回设备信息设置成功与否
  - `+TCMODINFOSET:OK`：设置成功，模组数据会保存到 flash，掉电不丢失
  - `+TCMODINFOSET:FAIL,<err_code>`：设置失败

参数

- **<module_name>**：模组名称，字符串类型，最大长度 30
- **<flash_size>**：模组 flash 大小（单位 MB），2 或者 4，数值类型
- **<WiFi_LED_GPIO>**：模组使用哪个 GPIO 口来控制 Wi-Fi 状态灯，数值类型，有效范围为 ESP8266 GPIO (0-16)
- **<fw_base_addr>**：模组提供给上位机 OTA 升级的固件数据保存地址，数值类型，该值需为 0x1000 的整数倍并且不小于 0x111000
- **<fw_max_size>**：模组提供给上位机 OTA 升级的固件最大空间，数值类型，该值不大于 716800 (700 KB)
- **<fixed_conn_id>**：保留选项，默认为 1

说明

- ESP Wi-Fi 模组固件和模组信息存储于不同 flash 分区，模组固件在启动时候会读取模组信息并做相应配置，这样可以使得同一版本模组固件可以适配不同的模组硬件

示例

```
// 设置成功
AT+TCMODINFOSET="ESP-WROOM-02D",2,0,1118208,716800,1

OK
+TCMODINFOSET:OK
```

AT+TCMQTTSRV：设置腾讯云 MQTT 服务器地址

功能

设置腾讯云 MQTT 服务器 host 地址，适用于私有化部署或者边缘计算场景

测试命令

命令：

```
AT+TCMQTTSRV=?
```

响应：

```
+TCMQTTSRV: "MQTT SERVER IP"

OK
```

读取命令

命令：

```
AT+TCMQTTSRV?
```

响应：

```
+TCMQTTSRV:192.168.10.118
OK
```

设置命令

命令：

```
AT+TCMQTTSRV=<Host addr>
```

响应：

```
OK
```

或

```
+CME ERROR:<err>
```

说明：

- 如果输入合法，首先返回 OK，然后返回设置成功与否
  - +TCMQTTSRV:OK：设置 IP 成功
  - +TCMQTTSRV:FAIL：设置 IP 失败
- 如果模组处于 MQTT 已连接状态中，则不支持该设置命令，会返回错误。需要先断开 MQTT 连接。

参数

- **<Host addr>**：腾讯云 MQTT 服务器 IP 或域名地址

AT+TCVER：读取模组固件 IoT SDK 版本信息

功能

读取模组固件 IoT SDK 版本信息

执行命令

命令：

```
AT+TCVER
```

示例

```
AT+TCVER
Tencent Cloud IoT AT  version: QCloud_AT_ESP8266_v2.0.0
Tencent Cloud IoT SDK version: 3.2.0
Firmware compile time: Jun 17 2020 16:25:27
Tencent Technology Co. Ltd.

OK
```

TC 网关子设备命令

AT+TCGWBIND：网关绑定子设备命令

功能

当 AT 模组用于网关设备上时，可以通过该命令对其下的子设备进行绑定与解绑操作。仅支持密钥方式的子设备。

测试命令

命令：

```
AT+TCGWBIND=?
```

响应：

```
+TCGWBIND:"MODE","PRODUCT_ID","DEVICE_NAME","DEVICE_SECRET"

OK
```

读取命令

命令：

```
AT+TCGWBIND?
```

响应：

```
OK
```

或

```
+TCGWBIND: <product_id>,<device_name>
:
:list of all bind sub-devices
+TCGWBIND: <product_id>,<device_name>

OK
```

说明：

- 读取命令会通过 MQTT 消息去物联网后台查询已经绑定在当前网关的所有子设备信息，并返回子设备列表。

设置命令

命令：

```
AT+TCGWBIND=<mode>,<productId>,<deviceName>[,<deviceSecret>]
```

响应：

```
OK
```

或

```
+CME ERROR: <err>
```

说明：

- 如果模组尚未连接腾讯云 MQTT 服务器，则返回错误，用户需要先发送连接命令 (*AT+TCMQTTCONN*) 才能发布消息。
- 该命令为基于 MQTT 消息的同步操作，会阻塞直至绑定或解绑操作完成或超时退出。如果模组处于 OTA 下载状态中，由于 ESP8266 平台资源限制，执行该命令可能会出现超时错误。如非必要，不建议在 OTA 下载过程中执行该命令。
- 如果输入合法，首先返回 `OK`，接下来返回绑定或解绑子设备操作成功与否
  - `+TCGWBIND:OK`：操作成功。对于绑定操作，重复绑定也返回成功。对于解绑操作，解绑未绑定的设备也返回成功。
  - `+ TCGWBIND:FAIL,<err_code>`：操作失败

参数

- **<mode>**：模式参数，必填项
  - 0：绑定操作
  - 1：解绑操作
- **<productId>**：子设备产品 id，必填项，字符串类型，最大长度 10 字节。
- **<deviceName>**：子设备名称，必填项，字符串类型，最大长度 48 字节。
- **<deviceSecret>**：子设备密钥，可选项，字符串类型，最大长度 44 字节。在解绑操作时候，不需要提供子设备密钥。在绑定操作时候，如果不提供子设备密钥，则网关模组从已经存储的子设备三元组中读取密钥信息（该信息由子设备信息设置命令提供或者子设备动态注册命令获取）。

示例

```
// 绑定子设备成功
AT+TCGWBIND=0,"CTQS08Y5LG","Dev01","ZHNkIGRzZCA="

OK
+TCGWBIND:OK
```

AT+TCGWONLINE：网关代理子设备上下线命令

功能

当 AT 模组用于网关设备上时，可以通过该命令代理其下的子设备上线和下线操作，仅支持密钥方式的子设备

测试命令

命令：

```
AT+TCGWONLINE=?
```

响应：

```
+TCGWONLINE:"MODE","PRODUCT_ID","DEVICE_NAME"

OK
```

读取命令

命令：

```
AT+TCGWONLINE?
```

响应：

```
OK
```

或

```
+TCGWONLINE: <product_id>,<device_name>
:
:list of online sub-device
+TCGWONLINE: <product_id>,<device_name>

OK
```

说明：

- 如果有已经在线的子设备，返回已在线的子设备信息列表

设置命令

命令：

```
AT+TCGWONLINE=<mode>,<productId>,<deviceName>
```

响应：

```
OK
```

或

```
+CME ERROR: <err>
```

说明：

- 如果模组尚未连接腾讯云 MQTT 服务器，则返回错误，用户需要先发送连接命令 (*AT+TCMQTTCONN*) 才能发布消息。
- 该命令为基于 MQTT 消息的同步操作，会阻塞直至上下线操作完成或超时退出。如果模组处于 OTA 下载状态中，由于 ESP8266 平台资源限制，执行该命令可能会出现超时错误。如非必要，不建议在 OTA 下载过程中执行该命令。
- 如果输入合法，首先返回 OK，接下来返回绑定或解绑子设备操作成功与否

  - +TCGWONLINE:OK：操作成功
  - + TCGWONLINE:FAIL,<err_code>：操作失败

参数

- **<mode>**：模式参数，必填项
  - 0：上线操作
  - 1：下线操作
- **<productId>**：子设备产品 id，必填项，字符串类型，最大长度 10 字节
- **<deviceName>**：子设备名称，必填项，字符串类型，最大长度 48 字节

示例

```
// 子设备上线成功
AT+TCGWONLINE=0,"CTQS08Y5LG","Dev01"

OK
+TCGWONLINE:OK

// 子设备上线成功后，网关可以代理子设备上线
AT+TCMQTTPUB="CTQS08Y5LG/Dev01/data",0,"hello world"

OK
+TCMQTTPUB: OK
```

AT+TCSUBDEVINFOSET：子设备信息设置

功能

设置腾讯云物联网平台创建的子设备信息，用于网关代理子设备通讯场景

测试命令

命令：

```
AT+TCSUBDEVINFOSET=?
```

响应：

```
+TCSUBDEVINFOSET:"MODE","PRODUCT_ID","DEVICE_NAME","DEVICE_SECRET_BCC","PRODUCT_REGION
↪"

OK
```

读取命令

命令：

```
AT+TCSUBDEVINFOSET?
```

响应：

```
+TCSUBDEVINFOSET: <product_id>,<device_name>,<devicesecret_checksum>,<product_region>

OK
```

或

```
+CME ERROR: <err>
```

说明：

- 不返回 devicesecret 的字符串内容，只返回 devicesecret 字符串的校验和 (BCC)

设置命令

命令：

```
AT+TCSUBDEVINFOSET=<mode>,<product_id>,<device_name>,<device_secret>[,<product_region>
↪]
```

响应：

```
OK
```

或

```
+CME ERROR: <err>
```

说明：

- 该命令不会影响当前网关的 MQTT 连接
- 如果输入合法，首先返回 OK，接下来返回设备信息设置成功与否

  - +TCSUBDEVINFOSET:OK：设置成功
  - + TCSUBDEVINFOSET:FAIL<err_code>：设置失败

参数

- **<mode>**：模式参数，必填项
  - 0：设置操作
  - 1：删除操作
- **<product_id>**：产品 id，必填项，字符串类型，最大长度 10 字节
- **<device_name>**：设备名称，必填项，字符串类型，最大长度 48 字节
- **<device_secret>**：设备密钥，必填项，字符串类型，最大长度 44 字节
- **<product_region>**：产品区域，选填项，字符串类型，最大长度 24 字节，如果不提供，默认为中国大陆公有云“ap-guangzhou”

示例

```
// 设置成功
AT+TCSUBDEVINFOSET=0,"CTQS08Y5LG","Dev01","ZHNkIGRzZCA="

OK
+TCSUBDEVINFOSET:OK
```

AT+TCSUBDEVPRDSET：子设备产品信息设置

功能

设置腾讯云物联网平台创建的子设备产品信息，适用于网关代理子设备进行动态注册场景

测试命令

命令：

```
AT+TCSUBDEVPRDSET=?
```

响应：

```
+TCSUBDEVPRDSET:"MODE","PRODUCT_ID","PRODUCT_SECRET_BCC","DEVICE_NAME","PRODUCT_REGION
↪"

OK
```

读取命令

命令：

```
AT+TCSUBDEVPRDSET?
```

响应：

```
OK
```

或

```
+TCSUBDEVPRDSET:<product_ID>,<product_secret_checksum>,<device_name>,<product_region>
:
:list of all sub-device
+TCSUBDEVPRDSET:<product_ID>,<product_secret_checksum>,<device_name>,<product_region>

OK
```

说明：

- 如果有已经设置的子设备，返回已设置的子设备信息列表

设置命令

命令：

```
AT+TCSUBDEVPRDSET=<mode>,<product_ID>,<product_secret>,<device_name>[,<product_region>
↪]
```

响应：

```
OK
```

或

```
+CME ERROR: <err>
```

说明：

- 该命令不会影响当前网关的 MQTT 连接
- 如果输入合法，首先返回 OK，接下来返回子设备信息设置成功与否
  - +TCSUBDEVPRDSET:OK：设置成功，产品数据会保存到 flash，掉电不丢失
  - +TCSUBDEVPRDSET:FAIL,<err_code>：设置失败

参数

- **<mode>**：模式参数，必填项
  - 0：设置操作
  - 1：删除操作
- **<product_ID>**：产品 ID，必填项，字符串类型，最大长度 10。
- **<product_secret>**：产品密钥，必填项，字符串类型，最大长度 32。
- **<device_name>**：设备名称，必填项，字符串类型，最大长度 48。
- **<product_region>**：产品区域，选填项，字符串类型，最大长度 24 字节，如果不提供，默认为中国大陆公有云“ap-guangzhou”。

示例

```
// 设置成功
AT+TCSUBDEVPRDSET=0,"CTQS08Y5LG","ZHNkIGRzZCA=","Dev01"

OK
+TCSUBDEVPRDSET:OK
```

AT+TCSUBDEVREG：执行子设备动态注册

功能

设置了子设备产品级密钥场景下，网关代理子设备进行动态注册并存储设备信息

测试命令

命令：

```
AT+TCSUBDEVREG=?
```

响应：

```
+TCSUBDEVREG:"PRODUCT_ID","DEVICE_NAME"

OK
```

读取命令

命令：

```
AT+TCSUBDEVREG?
```

响应：

```
OK
```

或

```
+TCSUBDEVREG: <product_id>,<device_name>
:
:list of registered sub-device
+TCSUBDEVREG: <product_id>,<device_name>

OK
```

说明：

- 如果有已经注册成功的子设备，返回已注册的子设备信息列表

执行命令

命令：

```
AT+TCSUBDEVREG=<productId>,<deviceName>
```

响应：

```
OK
```

或

```
+CME ERROR: <err>
```

说明：

- 由于 ESP8266 平台资源限制，执行该命令时需先断开网关的 MQTT 连接，否则可能会出现 `+CME ERROR:208` 错误。
- 如果执行状态合法，首先返回 `OK`，接下来返回子设备注册成功与否
- `+TCSUBDEVREG:OK`：动态注册成功，子设备密钥信息会保存到 flash
- `+TCSUBDEVREG:FAIL,<err_code>`：动态注册失败，返回错误码，具体参见本文档错误码章节

参数

- **<productId>**：子设备产品 id，必填项，字符串类型，最大长度 10 字节
- **<deviceName>**：子设备名称，必填项，字符串类型，最大长度 48 字节

说明

使用子设备动态注册的逻辑说明：

1. 如果模组上面没有完整的子设备信息，即子设备未注册未激活，则正常注册，返回成功/失败。
2. 模组上已存在一个子设备 A，且是已注册未激活状态，如果用户使用*AT+TCSUBDEVPRDSET* 提供的子设备信息也是 A，则正常注册，云端会重新分配 PSK，返回成功/失败。
3. 模组上已存在一个子设备 A，且是已注册已激活状态，如果用户使用*AT+TCSUBDEVPRDSET* 提供的子设备信息也是 A，则会注册失败，AT 命令返回错误，用户需要更换子设备信息或在云端将子设备重置。
4. 模组已存在子设备 A 的信息，如果用户使用*AT+TCSUBDEVPRDSET* 提供了一个新的设备 B 的信息，则会使用新的设备 B 的信息去注册，注册成功则会增加设备 B 的信息，即模组存在 A 和 B 的设备信息。
5. 正常情况下，设备动态注册仅需执行一次，执行成功后，设备密钥信息已经保存在模组 flash 中，后续上电初始化时可通过命令 AT+TCSUBDEVINFOSET? 查询是否存在正确的子设备信息。
6. 子设备动态注册成功后必须先通过网关绑定命令*AT+TCGWBIND* 进行绑定，再通过*AT+TCGWONLINE* 上线后，才能进行 MQTT 通讯。

示例

```
AT+TCSUBDEVREG="CTQS08Y5LG","Dev01"

OK
+TCSUBDEVREG:OK
```

错误码

服务端相关 err code

表 1: <err> 代码

<err> 代码	中文含义	内部字段
101	设备连接失败	device connect fail
110	设备订阅失败：无 Topic 权限	device subscribe fail: unauthorized operation
111	设备订阅失败：系统错误	device subscribe fail: system error
120	设备退订失败：系统错误	device unsubscribe fail: system error
130	设备发布消息失败：无 Topic 发布权限	device publish message to topic fail：unauthorized operation
131	设备发布消息失败：publish 超过频率限制	device publish message to topic fail：reach max limit
132	设备发布消息失败：payload 超过长度限制	device publish message to topic fail：payload too long

表 2: 执行错误码

执行错误码	中文含义	内部字段
-1001	表示失败返回	QCLOUD_ERR_FAILURE
-1002	表示参数无效错误，比如空指针	QCLOUD_ERR_INVAL
-3,	远程主机关闭连接	QCLOUD_ERR_HTTP_CLOSED
-4,	HTTP 未知错误	QCLOUD_ERR_HTTP
-5,	协议错误	QCLOUD_ERR_HTTP_PRTCL
-6,	域名解析失败	QCLOUD_ERR_HTTP_UNRESOLVED_DNS
-7,	URL 解析失败	QCLOUD_ERR_HTTP_PARSE
-8,	HTTP 连接失败	QCLOUD_ERR_HTTP_CONN
-9,	HTTP 鉴权问题	QCLOUD_ERR_HTTP_AUTH
-10,	HTTP 404	QCLOUD_ERR_HTTP_NOT_FOUND

下页继续

表 2 – 续上页

执行错误码	中文含义	内部字段
-11,	HTTP 超时	QCLOUD_ERR_HTTP_TIMEOUT
-102	表示往等待 ACK 列表中添加元素失败	QCLOUD_ERR_MQTT_PUSH_TO_LIST_FAILED
-103	表示未与 MQTT 服务器建立连接或已经断开连接	QCLOUD_ERR_MQTT_NO_CONN
-104	表示 MQTT 相关的未知错误	QCLOUD_ERR_MQTT_UNKNOWN
-105	表示正在与 MQTT 服务重新建立连接	QCLOUD_ERR_MQTT_ATTEMPTING_RECONNECT
-106	表示重连已经超时	QCLOUD_ERR_MQTT_RECONNECT_TIMEOUT
-107	表示超过可订阅的主题数	QCLOUD_ERR_MQTT_MAX_SUBSCRIPTIONS
-108	表示订阅主题失败, 即服务器拒绝	QCLOUD_ERR_MQTT_SUB
-109	表示无 MQTT 相关报文可以读取	QCLOUD_ERR_MQTT_NOTHING_TO_READ
-110	表示读取的 MQTT 报文有问题	QCLOUD_ERR_MQTT_PACKET_READ
-111	表示 MQTT 相关操作请求超时	QCLOUD_ERR_MQTT_REQUEST_TIMEOUT
-112	表示客户端 MQTT 连接未知错误	QCLOUD_ERR_MQTT_CONNACK_UNKNOWN
-113	表示客户端 MQTT 版本错误	QCLOUD_ERR_MQTT_CONANCK_UNACCEPTABLE_PROTOC
-114	表示客户端标识符错误	QCLOUD_ERR_MQTT_CONNACK_IDENTIFIER_REJECTED
-115	表示服务器不可用	QCLOUD_ERR_MQTT_CONNACK_SERVER_UNAVAILABLE
-116	表示客户端连接参数中的 username 或 password 错误	QCLOUD_ERR_MQTT_CONNACK_BAD_USERDATA
-117	表示客户端连接认证失败	QCLOUD_ERR_MQTT_CONNACK_NOT_AUTHORIZED
-118	表示收到的消息无效	QCLOUD_ERR_RX_MESSAGE_INVAL
-119	表示消息接收缓冲区的长度小于消息的长度	QCLOUD_ERR_BUF_TOO_SHORT
-120	表示该 QoS 级别不支持	QCLOUD_ERR_MQTT_QOS_NOT_SUPPORT
-121	表示取消订阅主题失败, 比如该主题不存在	QCLOUD_ERR_MQTT_UNSUB_FAIL
-132	表示 JSON 解析错误	QCLOUD_ERR_JSON_PARSE
-133	表示 JSON 文档会被截断	QCLOUD_ERR_JSON_BUFFER_TRUNCATED
-134	表示存储 JSON 文档的缓冲区太小	QCLOUD_ERR_JSON_BUFFER_TOO_SMALL
-135	表示 JSON 文档生成错误	QCLOUD_ERR_JSON
-136	表示超过 JSON 文档中的最大 Token 数	QCLOUD_ERR_MAX_JSON_TOKEN
-137	表示超过同时最大的文档请求	QCLOUD_ERR_MAX_APPENDING_REQUEST
-138	表示超过规定最大的 Topic 长度	QCLOUD_ERR_MAX_TOPIC_LENGTH
-601	表示 TCP 连接建立套接字失败	QCLOUD_ERR_TCP_SOCKET_FAILED
-602	表示无法通过主机名获取 IP 地址	QCLOUD_ERR_TCP_UNKNOWN_HOST
-603	表示建立 TCP 连接失败	QCLOUD_ERR_TCP_CONNECT
-604	表示 TCP 读超时	QCLOUD_ERR_TCP_READ_TIMEOUT

下页继续

表 2 – 续上页

执行错误码	中文含义	内部字段
-605	表示 TCP 写超时	QCLOUD_ERR_TCP_WRITE_TIMEOUT
-606	表示 TCP 读错误	QCLOUD_ERR_TCP_READ_FAIL
-607	表示 TCP 写错误	QCLOUD_ERR_TCP_WRITE_FAIL
-608	表示 TCP 对端关闭了连接	QCLOUD_ERR_TCP_PEER_SHUTDOWN
-609	表示底层没有数据可以读取	QCLOUD_ERR_TCP_NOTHING_TO_READ
-701	表示 SSL 初始化失败	QCLOUD_ERR_SSL_INIT
-702	表示 SSL 证书相关问题	QCLOUD_ERR_SSL_CERT
-703	表示 SSL 连接失败	QCLOUD_ERR_SSL_CONNECT
-704	表示 SSL 连接超时	QCLOUD_ERR_SSL_CONNECT_TIMEOUT
-705	表示 SSL 写超时	QCLOUD_ERR_SSL_WRITE_TIMEOUT
-706	表示 SSL 写错误	QCLOUD_ERR_SSL_WRITE
-707	表示 SSL 读超时	QCLOUD_ERR_SSL_READ_TIMEOUT
-708	表示 SSL 读错误	QCLOUD_ERR_SSL_READ
-709	表示底层没有数据可以读取	QCLOUD_ERR_SSL_NOTHING_TO_READ

CME ERROR 列表扩展

<err> 代码	含义
200	Previous command is not complete
201	msg packet over size
202	command timeout
203	check failed
204	Parameter invalid
205	No valid firmware
206	Memory allocation error
207	Flash access error
208	State error or not ready. eg: pub msg when MQTT not connected
209	Command execution error
210	Unknown error
211	Module self-OTA error
212	FLASH ERASE is going on
213	HTTP error

设备动态注册错误码

错误码	内部字段	说明
1000	ErrorCode_SDK_InternalError	内部错误
1004	ErrorCode_SDK_ProductNotExists	产品不存在
1006	ErrorCode_SDK_InvalidParam	参数错误
1010	ErrorCode_SDK_CheckSecretError	验签失败
1011	ErrorCode_SDK_NotSupportRegister	产品不支持动态注册
1012	ErrorCode_SDK_ExceedRegisterTimes	超过设备最大注册次数
1020	ErrorCode_SDK_NoSuchDevice	预创建注册模式未定义设备
1021	ErrorCode_SDK_DeviceHasRegisted	设备已注册
1031	ErrorCode_SDK_ExceedRegisterLimits	设备超过设定最大自动创建注册数量

模组配网及设备绑定错误类型

<err> 代码	含义
1	MQTT connect error
2	APP command error
3	WIFI boarding stop
4	RTOS task error
5	RTOS queue error
6	WIFI STA init error
7	WIFI AP init error
8	WIFI start error
9	WIFI config error
10	WIFI connect error
11	WIFI disconnect error
12	WIFI AP STA error
13	Smartconfig start error
14	Smartconfig data error
15-22	TCP/UDP socket error

网关子设备命令相关错误类型

错误码	描述
0	成功
-1	网关设备未绑定该子设备
-2	系统错误，子设备上线或者下线失败
801	请求参数错误
802	设备名非法，或者设备不存在
803	签名校验失败
804	签名方法不支持
805	签名请求已过期
806	该设备已被绑定
807	非普通设备不能被绑定
808	不允许的操作
809	重复绑定
810	不支持的子设备

应用说明

密钥认证方式连接腾讯云 MQTT 服务器

1. 设置设备信息

   - 命令

   ```
   AT+TCDEVINFOSET="1","CTQS08Y5LG","device1","ZHNkIGRzZCA="
   ```

   - 响应

   ```
   OK
   +TCDEVINFOSET: OK
   ```

2. TLS 密钥方式，超时时间设置为 5000 ms，心跳间隔为 240 s，clean session 为 1，使能自动重连，并连接 MQTT 服务器

   - 命令

   ```
   AT+TCMQTTCONN=1,5000,240,1,1
   ```

   - 响应

   ```
   OK
   +TCMQTTCONN:OK
   ```

订阅消息

- 命令

```
AT+TCMQTTSUB="CTQS08Y5LG/device1/control"
```

- 响应

```
OK
+TCMQTTSUB: OK
```

发布消息

发布消息，如果已经成功订阅过该主题并在云端配置了消息转发引擎，则设备会收到自己发布的消息，并通过 URC 自动上报

- 命令

```
AT+TCMQTTPUB="CTQS08Y5LG/device1/data",0," {\"action\":\"test\"\,\"time\":1565075992}"
```

- 响应

```
OK
+TCMQTTPUB: OK

+TCMQTTRCVPUB:"CTQS08Y5LG/device1/data",35,"{"action":"test","time":1565075992}"
```

数据通讯应用协议

设备通过 MQTT 协议与腾讯云物联网进行数据交互时，可使用下面几种应用协议：

1. 物联网开发平台–数据模板协议

   平台基于物模型和数据模板协议，可实现高效的物联网应用开发，并可让设备与腾讯连连小程序交互，具体请参考文档 数据模板协议。

2. 物联网通信–设备影子协议

   设备影子文档是服务器端为设备缓存的一份状态和配置数据，设备可通过影子数据流进行状态同步，具体请参考文档 设备影子详情。

3. 自定义

   用户可使用自定义的 MQTT 主题和应用协议。

使用建议

上位机或 MCU 使用 ESP8266 定制 AT 固件与腾讯云交互，可按下面不同阶段的使用建议进行相关命令的操作。

1. 检查及配置腾讯云物联网设备信息

   上电之后，MCU 应先检查模组是否配置了物联网设备信息，如果不存在或者设备信息有误，应通过命令配置设备三元组信息。如果使用动态注册，则应查询并设置产品级信息。

   相关命令

   - *AT+TCDEVINFOSET*
   - *AT+TCPRDINFOSET*

2. 查询 Wi-Fi 连接状态及配网操作

   在配置设备信息之后，MCU 可先查询 Wi-Fi 模组是否已经成功连接 Wi-Fi，如果没有联网，则可以通过配网命令使模组进入配网状态并可通过腾讯连连小程序进行配网及设备绑定操作。

   注意如果模组没有设备密钥，并已经配置好产品级密钥及设备名，则在配网成功之后会自动进行动态注册。

   相关命令

   - *AT+CWJAP*
   - *AT+CIPSTA*
   - *AT+TCSTARTSMART*
   - *AT+TCSAP*

3. MQTT 连接及订阅

   在设备信息正确配置及 Wi-Fi 连接成功之后，MCU 可通过 MQTT 连接物联网服务，根据自身应用情况配置连接参数（超时时间/心跳间隔等）以及订阅相应的消息 Topic，并在 MCU 配置相关 MQTT 消息上报及连接状态 URC 的回调处理机制。

   相关命令

   - *AT+TCMQTTCONN*
   - *AT+TCMQTTSUB*
   - *AT+TCMQTTSTATE*

4. MQTT 收发消息

   MCU 在发送消息时，根据消息长度选择使用 PUB 或者 PUBL 命令。注意如果是 JSON 数据需要进行转义处理再发送给模组。

   相关命令

   - *AT+TCMQTTPUB*

- *AT+TCMQTTPUBL*

5. OTA 使能及监听

   建议在 MQTT 连接成功之后，使能 OTA 功能，模组会启动后台 OTA 任务监听云端的升级命令，接收到升级命令后会自动下载固件到模组 flash，并通过 URC 通知 MCU，MCU 需要处理 OTA 相关 URC 消息，在下载成功之后可以通过相关命令读取 MCU 的新版本固件。

   相关命令

   - *AT+TCOTASET*
   - *AT+TCFWINFO*
   - *AT+TCREADFWDATA*

6. 断开 MQTT

   设备主动断开 MQTT 需要执行断开命令，否则云端不会马上感知到设备离线，需要等待心跳超时。

   执行断开命令会取消所有订阅的 Topic，如重新上线需要再次订阅

   相关命令

   - *AT+TCMQTTDISCONN*

6.1.2 腾讯云 IoT AT 固件

ESP8266

- ESP8266-WROOM-02 系列
  - QCloud_AT_ESP8266_v2.2.0
  - QCloud_AT_ESP8266_v2.1.1

CHAPTER 7

Index of Abbreviations

A2DP Advanced Audio Distribution Profile

高级音频分发框架

ADC Analog-to-Digital Converter

模拟数字转换器

ALPN Application Layer Protocol Negotiation

应用层协议协商

AT port AT port is the general name of AT log port (that is used to output log) and AT command port (that is used to send AT commands and receive responses). Please refer to 硬件连接 for default AT port pins and 如何修改 *AT port* 管脚 for how to customize them.

AT 端口是 AT 日志端口（用于输出日志）和 AT 命令端口（用于发送 AT 命令和接收响应）的总称。请参考硬件连接 了解默认的 AT 端口管脚，参考如何修改 *AT port* 管脚 了解如何自定义 AT 端口管脚。

Bluetooth LE Bluetooth Low Energy

低功耗蓝牙

BluFi Wi-Fi network configuration function via Bluetooth channel

BluFi 是一款基于蓝牙通道的 Wi-Fi 网络配置功能

DHCP Dynamic Host Configuration Protocol

动态主机配置协议

DNS Domain Name System

域名系统

DTIM Delivery Traffic Indication Map

延迟传输指示映射

GATTC Generic Attributes client

GATT 客户端

GATTS Generic Attributes server

GATT 服务器

HID Human Interface Device

人机接口设备

I2C Inter-Integrated Circuit

集成电路总线

ICMP Intemet Control Message Protocol

因特网控制报文协议

LWT Last Will and Testament

遗嘱

MAC Media Access Control

MAC 地址

mDNS Multicast Domain Name System

多播 DNS

MSB Most Significant Bit

最高有效位

MTU maximum transmission unit

最大传输单元

NVS Non-Volatile Storage

非易失性存储器

Normal Transmission Mode Default Transmission Mode

In normal transmission mode, users can send AT commands. For examples, users can send MCU data received by AT command port to the opposite end of transmission by *AT+CIPSEND*; and the data received from the opposite end of transmission will also be returned to MCU through AT command port with additional prompt: *+IPD*.

During a normal transmission, if the connection breaks, ESP devices will give a prompt and will not attempt to reconnect.

More details are in *Transmission Mode Shift Diagram*.

普通传输模式 默认传输模式

在普通传输模式下，用户可以发送 AT 命令。例如，用户可以通过*AT+CIPSEND* 命令，发送 AT 命令口收到的 MCU 数据到传输对端。从传输对端收到的数据，会通过 AT 命令口返回给 MCU，同时会附带 *+IPD* 信息。

普通传输模式时，如果连接断开，ESP 不会重连，并提示连接断开。

更多介绍请参考*Transmission Mode Shift Diagram*。

Passthrough Mode Also called as "Passthrough Sending & Receiving Mode".

In passthrough mode, users cannot send AT commands except special +++ command. All MCU data received by AT command port will be sent to the opposite end of transmission without any modification; and the data received from the opposite end of transmission will also be returned to MCU through AT command port without any modification.

During the Wi-Fi passthrough transmission, if the connection breaks, ESP devices will keep trying to reconnect until +++ is input to exit the passthrough transmission.

More details are in *Transmission Mode Shift Diagram*.

透传模式 也称为"透传发送接收模式"。

在透传模式下，用户不能发送其它 AT 命令，除了特别的+++ 命令。AT 命令口收到的所有的 MCU 数据都将无修改地，发送到传输对端。从传输对端收到的数据也会通过 AT 命令口无修改地，返回给 MCU。

Wi-Fi 透传模式传输时，如果连接断开，ESP 会不停地尝试重连，此时单独输入+++ 退出透传，则停止重连。

更多介绍请参考*Transmission Mode Shift Diagram*。

Transmission Mode Shift Diagram

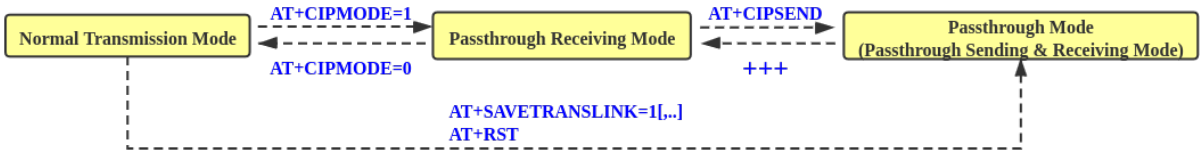


图 1: Transmission Mode Shift Diagram

More details are in the following introduction.

- *Normal Transmission Mode* (普通传输模式)
- *Passthrough Receiving Mode* (透传接收模式)

- *Passthrough Mode* (透传模式)
- *AT+CIPMODE*
- *AT+CIPSEND*
- +++
- *AT+SAVETRANSLINK*

Passthrough Receiving Mode The temporary mode between *Normal Transmission Mode* and *Passthrough Mode*.

In passthrough receiving mode, AT cannot send any data to the opposite end of transmission; but the data received from the opposite end of transmission can be returned to MCU through AT command port without any modification. More details are in *Transmission Mode Shift Diagram*.

透传接收模式 在普通传输模式 和透传模式 之间的一个临时模式。

在透传接收模式，AT 不能发送数据到传输对端；但 AT 可以收到来自传输对端的数据，通过 AT 命令口无修改地返回给 MCU。更多介绍请参考*Transmission Mode Shift Diagram*。

PBC Push Button Configuration

按钮配置

PCI Authentication Payment Card Industry Authentication. In ESP-AT project, it refers to all Wi-Fi authentication modes except OPEN and WEP.

PCI 认证，在 ESP-AT 工程中指的是除 OPEN 和 WEP 以外的 Wi-Fi 认证模式。

PLCP Physical Layer Convergence Procedure

PLCP 协议，即物理层会聚协议

PMF protected management frame

受保护的管理帧

PSK Pre-shared Key

预共享密钥

PWM Pulse-Width Modulation

脉冲宽度调制

QoS Quality of Service

服务质量

RTC Real Time Controller. A group of circuits in SoC that keeps working in any chip mode and at any time.

实时控制器，为 SoC 中的一组电路，在任何芯片模式下都能随时保持工作。

SMP Security Manager Protocol

安全管理协议

SNI Server Name Indication

服务器名称指示

SNTP Simple Network Time Protocol

简单网络时间协议

SPI Serial Peripheral Interface

串行外设接口

SPP Serial Port Profile

SPP 协议，即串口协议

SSL Secure Sockets Layer

SSL 协议，即安全套接字协议

TLS Transport Layer Security

TLS 协议，即传输层安全性协议

URC Unsolicited Result Code

非请求结果码，一般为模组给 MCU 的串口返回

UTC Coordinated Universal Time

协调世界时

UUID universally unique identifier

通用唯一识别码

WEP Wired-Equivalent Privacy

WEP 加密方式，即有线等效加密

WPA Wi-Fi Protected Access

Wi-Fi 保护访问

WPA2 Wi-Fi Protected Access II

Wi-Fi 保护访问 II

WPS Wi-Fi Protected Setup

Wi-Fi 保护设置

- genindex

索引

A

A2DP, **379**
ADC, **379**
ALPN, **379**
AT port, **379**
AT_DISABLE_SLEEP (*C++ enumerator*), 312
at_handle_result_code (*C++ function*), 309
AT_LIGHT_SLEEP (*C++ enumerator*), 313
at_max (*C 宏*), 311
AT_MAX_MODEM_SLEEP (*C++ enumerator*), 313
at_min (*C 宏*), 311
AT_MIN_MODEM_SLEEP (*C++ enumerator*), 313
AT_SLEEP_MAX (*C++ enumerator*), 313
at_sleep_mode_t (*C++ enum*), 312
at_write_data_fn_t (*C++ type*), 312

B

Bluetooth LE, **379**
BluFi, **379**

D

DHCP, **379**
DNS, **380**
DTIM, **380**

E

esp_at_base_cmd_regist (*C++ function*), 306
esp_at_ble_cmd_regist (*C++ function*), 307
esp_at_ble_hid_cmd_regist (*C++ function*), 307
esp_at_blufi_cmd_regist (*C++ function*), 307
esp_at_board_init (*C++ function*), 315
esp_at_bt_a2dp_cmd_regist (*C++ function*), 307
esp_at_bt_cmd_regist (*C++ function*), 307
esp_at_bt_spp_cmd_regist (*C++ function*), 307
ESP_AT_CMD_ERROR_CMD_EXEC_FAIL (*C 宏*), 312
ESP_AT_CMD_ERROR_CMD_OP_ERROR (*C 宏*), 312
ESP_AT_CMD_ERROR_CMD_PROCESSING (*C 宏*), 312
ESP_AT_CMD_ERROR_CMD_UNSUPPORT (*C 宏*), 312
ESP_AT_CMD_ERROR_NON_FINISH (*C 宏*), 311
ESP_AT_CMD_ERROR_NOT_FOUND_AT (*C 宏*), 311
ESP_AT_CMD_ERROR_OK (*C 宏*), 311
ESP_AT_CMD_ERROR_PARA_INVALID (*C 宏*), 312
ESP_AT_CMD_ERROR_PARA_LENGTH (*C 宏*), 311
ESP_AT_CMD_ERROR_PARA_NUM (*C 宏*), 312
ESP_AT_CMD_ERROR_PARA_PARSE_FAIL (*C 宏*), 312
ESP_AT_CMD_ERROR_PARA_TYPE (*C 宏*), 312
esp_at_cmd_struct (*C++ class*), 309
esp_at_cmd_struct::at_cmdName (*C++ member*), 309
esp_at_cmd_struct::at_exeCmd (*C++ member*), 310
esp_at_cmd_struct::at_queryCmd (*C++ member*), 309
esp_at_cmd_struct::at_setupCmd (*C++ member*), 310
esp_at_cmd_struct::at_testCmd (*C++ mem-*

ber), 309
esp_at_custom_ble_ops_regist (*C++ function*), 305
esp_at_custom_ble_ops_struct (*C++ class*), 310
esp_at_custom_ble_ops_struct::connect_cb (*C++ member*), 311
esp_at_custom_ble_ops_struct::disconnect_cb (*C++ member*), 311
esp_at_custom_ble_ops_struct::recv_data (*C++ member*), 311
esp_at_custom_cmd_array_regist (*C++ function*), 304
esp_at_custom_cmd_line_terminator_get (*C++ function*), 308
esp_at_custom_cmd_line_terminator_set (*C++ function*), 307
esp_at_custom_net_ops_regist (*C++ function*), 305
esp_at_custom_net_ops_struct (*C++ class*), 310
esp_at_custom_net_ops_struct::connect_cb (*C++ member*), 310
esp_at_custom_net_ops_struct::disconnect_cb (*C++ member*), 310
esp_at_custom_net_ops_struct::recv_data (*C++ member*), 310
esp_at_custom_ops_regist (*C++ function*), 305
esp_at_custom_ops_struct (*C++ class*), 311
esp_at_custom_ops_struct::pre_active_write_cb (*C++ member*), 311
esp_at_custom_ops_struct::pre_deepsleep_callback (*C++ member*), 311
esp_at_custom_ops_struct::pre_restart_callback (*C++ member*), 311
esp_at_custom_ops_struct::pre_sleep_callback (*C++ member*), 311
esp_at_custom_ops_struct::status_callback (*C++ member*), 311
esp_at_custom_partition_find (*C++ function*), 308
esp_at_device_ops_regist (*C++ function*), 304
esp_at_device_ops_struct (*C++ class*), 310
esp_at_device_ops_struct::get_data_length (*C++ member*), 310
esp_at_device_ops_struct::read_data (*C++ member*), 310
esp_at_device_ops_struct::wait_write_complete (*C++ member*), 310
esp_at_device_ops_struct::write_data (*C++ member*), 310
esp_at_driver_cmd_regist (*C++ function*), 307
esp_at_eap_cmd_regist (*C++ function*), 307
esp_at_error_code (*C++ enum*), 313
ESP_AT_ERROR_NO (*C 宏*), 311
esp_at_eth_cmd_regist (*C++ function*), 307
ESP_AT_FACTORY_PARAMETER_SIZE (*C 宏*), 315
esp_at_fs_cmd_regist (*C++ function*), 307
esp_at_get_current_cmd_name (*C++ function*), 309
esp_at_get_current_module_name (*C++ function*), 315
esp_at_get_module_id (*C++ function*), 315
esp_at_get_module_name_by_id (*C++ function*), 315
esp_at_get_para_as_digit (*C++ function*), 303
esp_at_get_para_as_str (*C++ function*), 303
esp_at_get_version (*C++ function*), 305
esp_at_http_cmd_regist (*C++ function*), 307
esp_at_mdns_cmd_regist (*C++ function*), 307
esp_at_module (*C++ enum*), 313
esp_at_module_init (*C++ function*), 303
ESP_AT_MODULE_NUM (*C++ enumerator*), 313
esp_at_mqtt_cmd_regist (*C++ function*), 307
esp_at_net_cmd_regist (*C++ function*), 306
ESP_AT_PARA_PARSE_RESULT_FAIL (*C++ enumerator*), 314
ESP_AT_PARA_PARSE_RESULT_OK (*C++ enumerator*), 314
ESP_AT_PARA_PARSE_RESULT_OMITTED (*C++ enumerator*), 314
esp_at_para_parse_result_type (*C++ enum*), 314
esp_at_ping_cmd_regist (*C++ function*), 307

esp_at_port_active_write_data (C++ function), 305
esp_at_port_enter_specific (C++ function), 308
esp_at_port_exit_specific (C++ function), 309
esp_at_port_get_data_length (C++ function), 306
esp_at_port_read_data (C++ function), 306
esp_at_port_recv_data_notify (C++ function), 304
esp_at_port_recv_data_notify_from_isr (C++ function), 304
esp_at_port_specific_callback_t (C++ type), 312
ESP_AT_PORT_TX_WAIT_MS_MAX (C 宏), 315
esp_at_port_wait_write_complete (C++ function), 306
esp_at_port_write_data (C++ function), 305
esp_at_response_result (C++ function), 305
ESP_AT_RESULT_CODE_ERROR (C++ enumerator), 314
ESP_AT_RESULT_CODE_FAIL (C++ enumerator), 314
ESP_AT_RESULT_CODE_IGNORE (C++ enumerator), 314
ESP_AT_RESULT_CODE_MAX (C++ enumerator), 314
ESP_AT_RESULT_CODE_OK (C++ enumerator), 314
ESP_AT_RESULT_CODE_OK_AND_INPUT_PROMPT (C++ enumerator), 314
ESP_AT_RESULT_CODE_PROCESS_DONE (C++ enumerator), 314
ESP_AT_RESULT_CODE_SEND_FAIL (C++ enumerator), 314
ESP_AT_RESULT_CODE_SEND_OK (C++ enumerator), 314
esp_at_result_code_string_index (C++ enum), 314
esp_at_smartconfig_cmd_regist (C++ function), 307
ESP_AT_STATUS_NORMAL (C++ enumerator), 312
ESP_AT_STATUS_TRANSMIT (C++ enumerator), 312
esp_at_status_type (C++ enum), 312
ESP_AT_SUB_CMD_EXEC_FAIL (C++ enumerator), 313
ESP_AT_SUB_CMD_OP_ERROR (C++ enumerator), 314
ESP_AT_SUB_CMD_PROCESSING (C++ enumerator), 314
ESP_AT_SUB_COMMON_ERROR (C++ enumerator), 313
ESP_AT_SUB_NO_AT (C++ enumerator), 313
ESP_AT_SUB_NO_TERMINATOR (C++ enumerator), 313
ESP_AT_SUB_OK (C++ enumerator), 313
ESP_AT_SUB_PARA_INVALID (C++ enumerator), 313
ESP_AT_SUB_PARA_LENGTH_MISMATCH (C++ enumerator), 313
ESP_AT_SUB_PARA_NUM_MISMATCH (C++ enumerator), 313
ESP_AT_SUB_PARA_PARSE_FAIL (C++ enumerator), 313
ESP_AT_SUB_PARA_TYPE_MISMATCH (C++ enumerator), 313
ESP_AT_SUB_UNSUPPORT_CMD (C++ enumerator), 313
esp_at_transmit_terminal (C++ function), 304
esp_at_transmit_terminal_from_isr (C++ function), 304
esp_at_user_cmd_regist (C++ function), 306
esp_at_web_server_cmd_regist (C++ function), 315
esp_at_wifi_cmd_regist (C++ function), 306
esp_at_wifi_event_handler (C++ function), 309
esp_at_wps_cmd_regist (C++ function), 307

G

GATTC, **380**
GATTS, **380**

H

HID, **380**

I

I2C, 380
ICMP, 380

L

LWT, 380

M

MAC, 380
mDNS, 380
MSB, 380
MTU, 380

N

Normal Transmission Mode, 380
NVS, 380

P

Passthrough Mode, 381
Passthrough Receiving Mode, 382
PBC, 382
PCI Authentication, 382
PLCP, 382
PMF, 382
PSK, 382
PWM, 382

Q

QoS, 382

R

RTC, 382

S

SMP, 382
SNI, 383
SNTP, 383
SPI, 383
SPP, 383
SSL, 383

T

TLS, 383
Transmission Mode Shift Diagram, 381

U

URC, 383
UTC, 383
UUID, 383

W

WEP, 383
WPA, 383
WPA2, 383
WPS, 383

普通传输模式, 381

透传接收模式, 382
透传模式, 381