ESP32-C2

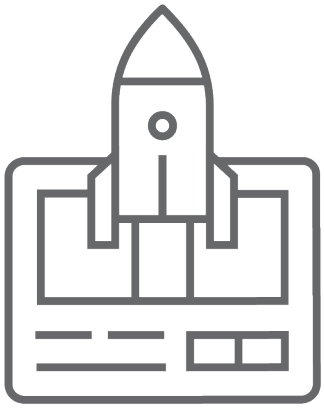
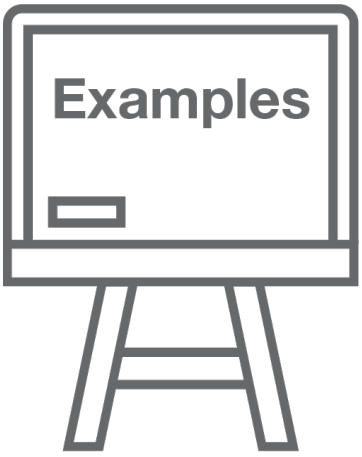
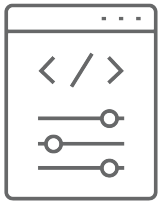
ESP-AT 用户指南

Release v2.3.0.0-esp32c3-361-gb5430077
乐鑫信息科技
2022 年 11 月 25 日

Table of contents

这里是乐鑫 ESP-AT 开发框架的文档中心。ESP-AT 作为由 Espressif Systems 发起和提供技术支持的官方项目，适用于 Windows、Linux、macOS 上的 ESP32、ESP32-C2、ESP32-C3、ESP8266、和 ESP32-S2 系列芯片。

本文档仅包含针对 ESP32-C2 芯片的 ESP-AT 使用。

<table>
<tr><td>  </td><td> 
</td><td> 
</td></tr>
<tr><td>入门</td><td>AT Binary 列表</td><td>AT 命令集</td></tr>
<tr><td> 
</td><td> 
</td><td>  </td></tr>
<tr><td>AT 命令示例</td><td>编译和开发</td><td>第三方定制化 AT 命令和固件</td></tr>
</table>

Chapter 1

入门指南

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

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 实现的软件工程。它使 ESP32-C2 模组作为从机，MCU 作为主机。MCU 发送 AT 命令给 ESP32-C2 模组，控制 ESP32-C2 模组执行不同的操作，并接收 ESP32-C2 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。

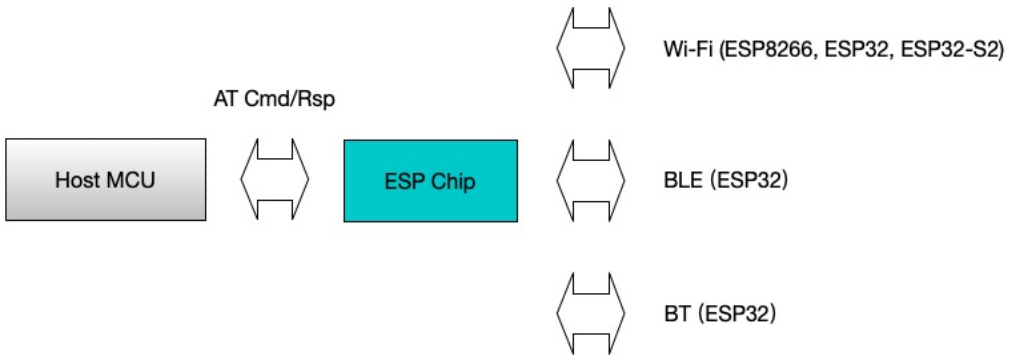


图 1: ESP-AT 概览

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*。

默认配置下，MCU 通过 UART 连接至 ESP32-C2 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

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。

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*ESP-AT 固件差异*。

1.2.1 硬件准备

表 1: ESP-AT 测试所需硬件

硬件	功能
ESP32-C2 开发板	从机
USB 数据线（连接 ESP32-C2 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 ESP32-C2 开发板）	发送 AT 命令、接收 AT 响应数据连接

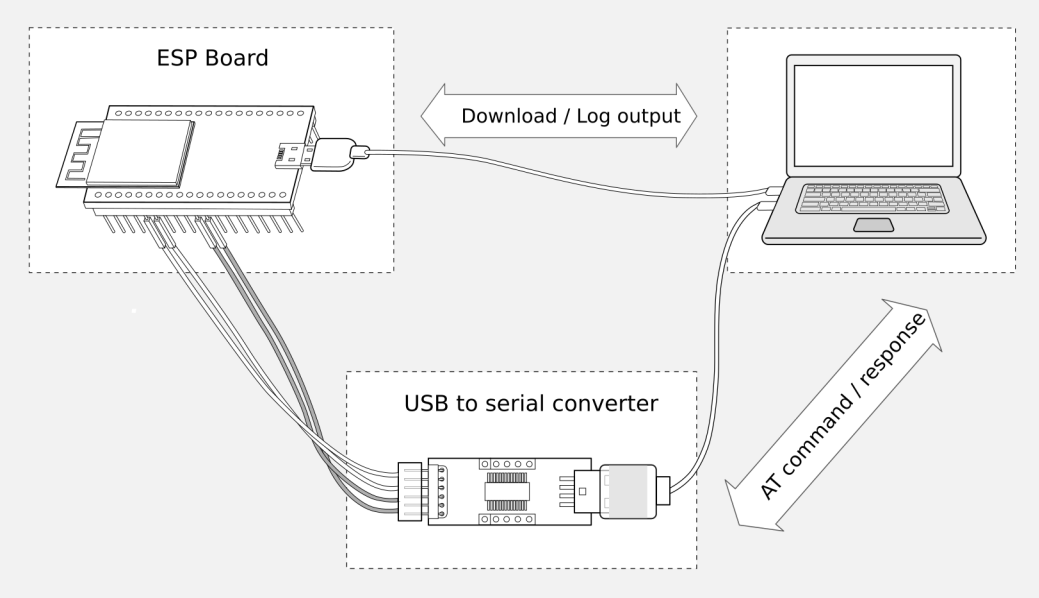


图 2: ESP-AT 测试硬件连接示意图

注意：

- 官方提供的默认*AT 固件* 仅支持 26 MHz 晶振。如果您的 ESP32-C2 晶振为 40 MHz，请参考*编译 ESP-AT 工程* 自行编译 ESP32-C2 AT 固件。在第五步配置：

```
python build.py menuconfig -> Component config -> Hardware Settings -> Main␣
↪XTAL Config -> Main XTAL frequency -> 40 MHz
```

- 上图使用 4 根杜邦线连接 ESP32-C2 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 ESP32C2-4MB 系列

ESP32C2-4MB 系列指的是内置 ESP32-C2/ESP8684 芯片，同时有 4 MB flash 的模组/开发板，例如：ESP32C2 MINI 系列设备、ESP32C2 WROOM 系列设备。

ESP32C2-4MB 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。默认情况下，UART0 和 UART1 均使用 `115200` 波特率进行通信。

表 2: ESP32C2-4MB Series 系列硬件连接管脚分配

功能	ESP32C2-4MB 开发板管脚	其它设备管脚
下载固件/输出日志 [1]	**UART0** • GPIO19 (RX) • GPIO20 (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：ESP32C2-4MB 开发板和 PC 之间的管脚连接已内置在 ESP32C2-4MB 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。

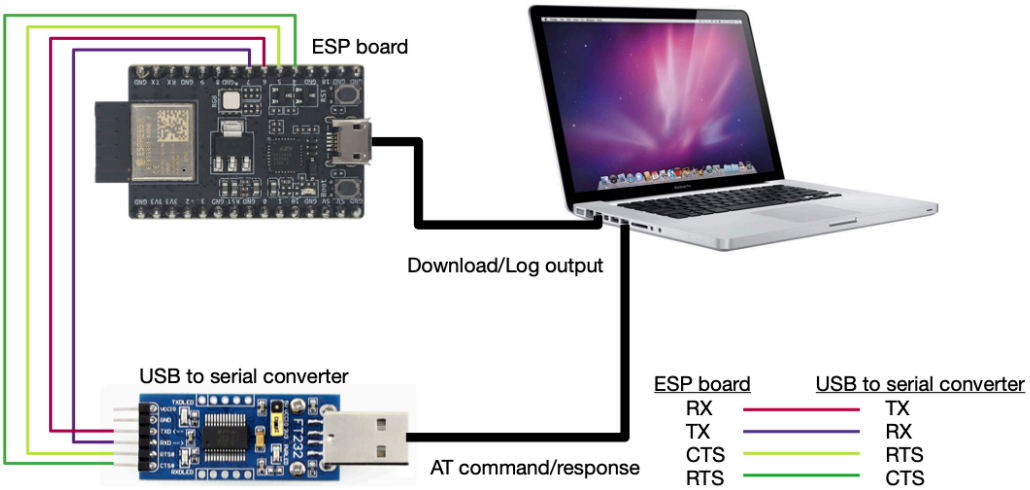


图 3: ESP32C2-4MB 系列硬件连接示意图

如果需要直接基于 ESP32C2-4MB 模组进行连接，请参考对应模组的 技术规格书。

1.2.3 ESP32C2-2MB 系列

ESP32C2-2MB 系列指的是内置 ESP32-C2/ESP8684 芯片，同时有 2 MB flash 的模组/开发板。

ESP32C2-2MB 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。默认情况下，UART0 和 UART1 均使用 `115200` 波特率进行通信。

表 3: ESP32C2-2MB Series 系列硬件连接管脚分配

功能	ESP32C2-2MB 开发板管脚	其它设备管脚
下载固件 [1]	**UART0** • GPIO19 (RX) • GPIO20 (TX)	**PC** • TX • RX
AT 命令/响应 [2]	**UART1** • GPIO6 (RX) • GPIO7 (TX) • GPIO19 (CTS) • GPIO20 (RTS)	**USB 转 UART 串口模块** • TX • RX • RTS • CTS
输出日志	**UART0** • GPIO8 (TX)	**USB 转 UART 串口模块** • RX

说明 1：ESP32C2-2MB 开发板和 PC 之间的管脚连接已内置在 ESP32C2-2MB 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。

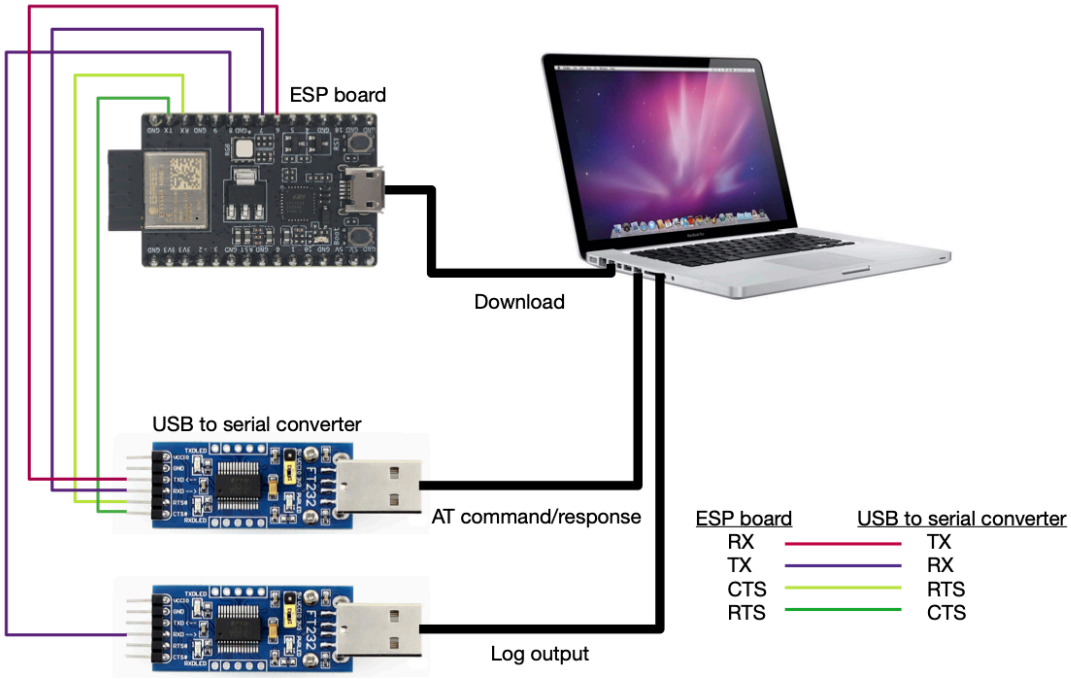


图 4: ESP32C2-2MB 系列硬件连接示意图

如果需要直接基于 ESP32C2-2MB 模组进行连接，请参考对应模组的 技术规格书。

1.3 下载指导

本文档以 ESP8684-MINI-1 模组为例，介绍如何下载 ESP8684-MINI-1 模组对应的 AT 固件，以及如何将固件烧录到模组上，其它 ESP32-C2 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*ESP-AT 固件差异*。

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 ESP8684-MINI-1 对应的 `ESP32C2-4MB_AT_Bin_V2.5.0.0` 固件，该固件的目录结构及其中各个 bin 文件介绍如下，其它 ESP32-C2 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
│   ├── client_ca.bin
│   ├── client_cert.bin
│   ├── client_key.bin
│   ├── factory_param.bin
│   ├── factory_param_MINI-1.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
├── esp-at.elf
├── esp-at.map
├── factory                         // 量产所需打包好的 bin 文件
│   ├── factory_MINI-1.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
└── sdkconfig                       // AT 固件对应的编译配置
```

其中，`download.config` 文件包含烧录固件的参数：

```
--flash_mode dio --flash_freq 60m --flash_size 4MB
0x0 bootloader/bootloader.bin
0x60000 esp-at.bin
0x8000 partition_table/partition-table.bin
0xd000 ota_data_initial.bin
0xf000 phy_init_data.bin
0x1e000 at_customize.bin
```

(下页继续)

(续上页)

```
0x1F000 customized_partitions/server_cert.bin
0x21000 customized_partitions/server_key.bin
0x23000 customized_partitions/server_ca.bin
0x25000 customized_partitions/client_cert.bin
0x27000 customized_partitions/client_key.bin
0x29000 customized_partitions/client_ca.bin
0x32000 customized_partitions/mqtt_cert.bin
0x34000 customized_partitions/mqtt_key.bin
0x36000 customized_partitions/mqtt_ca.bin
0x2B000 customized_partitions/factory_param.bin
```

- `--flash_mode dio` 代表此固件采用的 flash dio 模式进行编译；
- `--flash_freq 60m` 代表此固件采用的 flash 通讯频率为 60 MHz；
- `--flash_size 4MB` 代表此固件适用的 flash 最小为 4 MB；
- `0xd000 ota_data_initial.bin` 代表在 `0xd000` 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 `doc` 文件夹。

- 打开 Flash 下载工具；
- 选择芯片类型；（此处，我们选择 `ESP32C2`。）
- 根据您的需求选择一种工作模式；（此处，我们选择 `develop`。）
- 根据您的需求选择一种下载接口；（此处，我们选择 `uart`。）

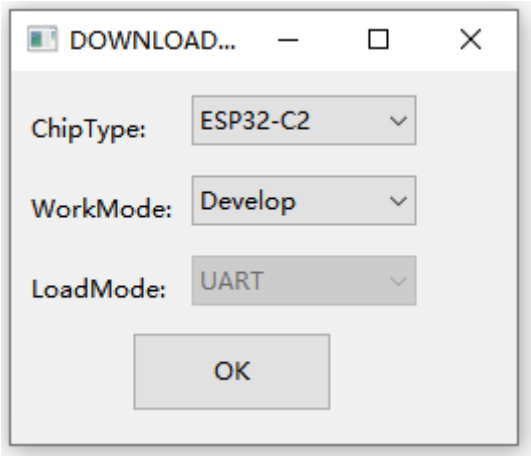


图 5: 固件下载配置选择

- 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”；

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*。

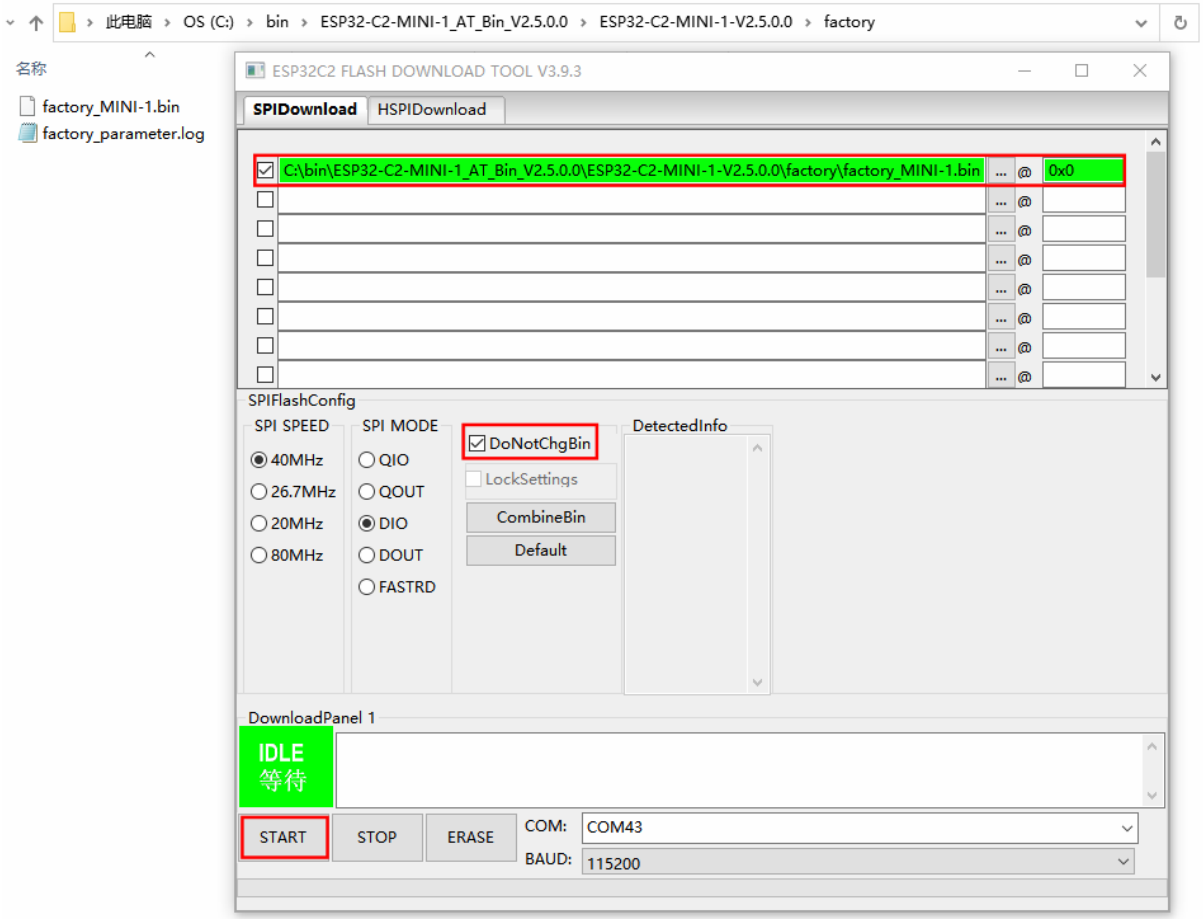


图 6: 下载至单个地址界面图（点击放大）

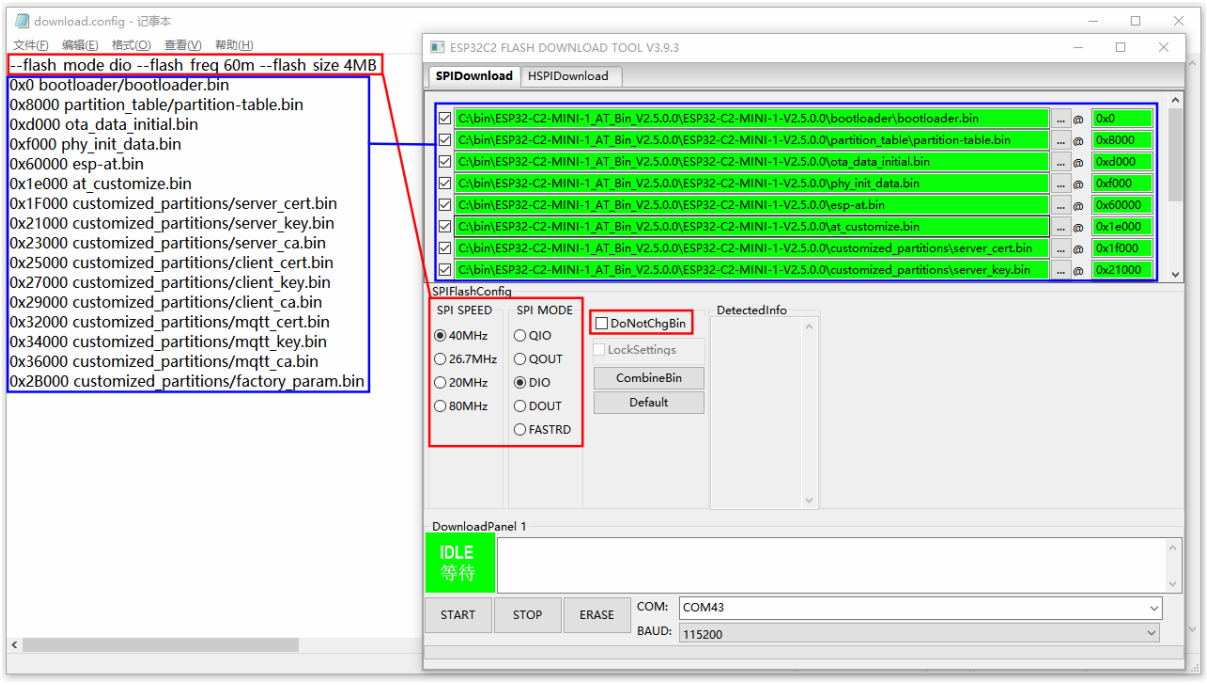


图 7: 下载至多个地址界面图（点击放大）

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 ESP8684-MINI-1 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 60m --flash_size 4MB 0x0 bootloader/bootloader.bin␣
↪0x60000 esp-at.bin 0x8000 partition_table/partition-table.bin 0xd000␣
↪ota_data_initial.bin 0xf000 phy_init_data.bin 0x1e000 at_customize.
↪bin 0x1F000 customized_partitions/server_cert.bin 0x21000 customized_
↪partitions/server_key.bin 0x23000 customized_partitions/server_ca.
↪bin 0x25000 customized_partitions/client_cert.bin 0x27000 customized_
↪partitions/client_key.bin 0x29000 customized_partitions/client_ca.
↪bin 0x32000 customized_partitions/mqtt_cert.bin 0x34000 customized_
↪partitions/mqtt_key.bin 0x36000 customized_partitions/mqtt_ca.bin␣
↪0x2B000 customized_partitions/factory_param.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 ESP8684-MINI-1 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 60m --flash_size 4MB 0x0 factory/factory_MINI-1.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。

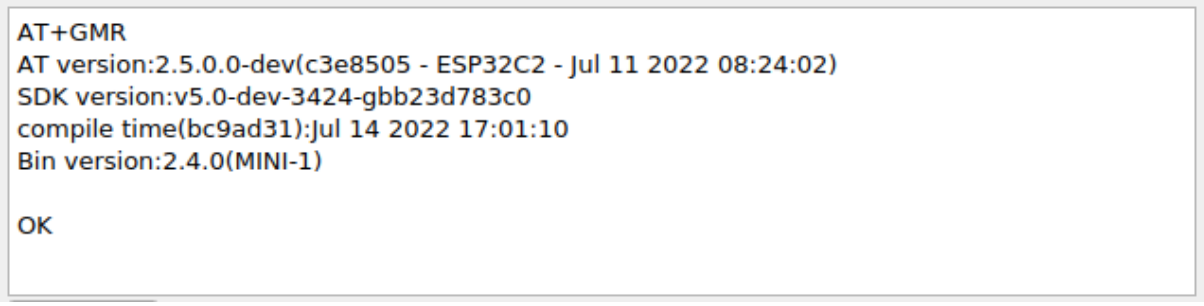


图 8: AT 响应

否则，您需要通过以下方式之一检查 ESP32-C2 设备开机日志：

方法 1：

- 打开串口工具，如 SecureCRT；
- 串口：选择用于“下载固件/输出日志”的串口，串口详情请参阅硬件连接。
- 波特率：115200；
- 数据位：8；
- 检验位：None；
- 停止位：1；
- 流控：None；
- 直接按开发板的 RST 键，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。

方法 2：

- 打开两个串口工具，如 SecureCRT；
- 串口：分别选择用于发送或接收“AT 命令/响应”的串口以及用于“下载固件/输出日志”的串口，串口详情请参阅硬件连接。
- 波特率：115200；
- 数据位：8；
- 检验位：None；
- 停止位：1；
- 流控：None；
- 在发送或接收“AT 命令/响应”的串口输入*AT+RST* 命令，并且换行 (CR LF)，若“下载固件/输出日志”的串口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-C2 开机日志:

```
ESP-ROM:esp8684-api2-20220127
Build:Jan 27 2022
rst:0x1 (POWERON),boot:0xc (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fcd6108,len:0x18b0
load:0x403ae000,len:0x854
load:0x403b0000,len:0x2724
entry 0x403ae000
I (21) boot: ESP-IDF v5.0-dev-3424-gbb23d783c0 2nd stage bootloader
I (21) boot: compile time 19:44:11
I (21) boot: chip revision: 0
I (24) boot.esp32c2: MMU Page Size  : 64K
I (29) boot.esp32c2: SPI Speed      : 60MHz
I (34) boot.esp32c2: SPI Mode       : DIO
I (38) boot.esp32c2: SPI Flash Size : 4MB
I (43) boot: Enabling RNG early entropy source...
I (49) boot: Partition Table:
I (52) boot: ## Label            Usage          Type ST Offset   Length
I (59) boot:  0 otadata          OTA data         01 00 0000d000 00002000
I (67) boot:  1 phy_init         RF data          01 01 0000f000 00001000
```

(下页继续)

(续上页)

```
I (74) boot:  2 nvs              WiFi data        01 02 00010000 0000e000
I (82) boot:  3 at_customize     unknown          40 00 0001e000 00042000
I (89) boot:  4 ota_0            OTA app          00 10 00060000 001d0000
I (97) boot:  5 ota_1            OTA app          00 11 00230000 001d0000
I (104) boot: End of partition table
I (108) esp_image: segment 0: paddr=00060020 vaddr=3c0d0020 size=279d8h (162264)␣
↪map
I (153) esp_image: segment 1: paddr=00087a00 vaddr=3fcab2b0 size=018a0h (  6304)␣
↪load
I (155) esp_image: segment 2: paddr=000892a8 vaddr=40380000 size=06d70h ( 28016)␣
↪load
I (166) esp_image: segment 3: paddr=00090020 vaddr=42000020 size=c10f8h (790776)␣
↪map
I (344) esp_image: segment 4: paddr=00151120 vaddr=40386d70 size=04534h ( 17716)␣
↪load
I (353) boot: Loaded app from partition at offset 0x60000
I (353) boot: Disabling RNG early entropy source...
module_name:MINI-1
max tx power=78,ret=0
2.5.0
```

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

2.1 发布的固件

推荐下载最新版本的固件。目前，乐鑫只发布了以下 ESP32-C2 系列模组的 AT 固件。

备注： 如果您的模组没有发布的固件，可以使用相同硬件配置的模组的固件（点击*ESP-AT 固件差异* 查看与您的模组硬件配置相同的固件），或者为您的模组创建出厂参数二进制文件（*如何生成出厂参数二进制文件*）。

2.1.1 ESP32-C2 2MB 系列

- v3.0.0.0 ESP32-C2-2MB_AT_Bin_V3.0.0.0.zip （推荐）

2.1.2 ESP32-C2 4MB 系列

- v3.0.0.0 ESP32-C2-4MB_AT_Bin_V3.0.0.0.zip （推荐）

本文档包含以下小节：

- *下载 ESP32-C2 AT 发布版固件*
- *AT 固件简介*：AT 固件包含哪些二进制文件及其作用
- *我该选哪种类型的固件？*：不同类型的 AT 固件及其获取方式、适用情况、优缺点等
- *获取固件后，接来下做什么？*

备注： 若需下载其他芯片系列的发布版固件，请在页面左上方的下拉菜单栏选择相应的芯片，即可跳转至该芯片的文档进行下载。

2.2 AT 固件简介

ESP-AT 固件包含了若干个特定功能的二进制文件：

- factory/factory_xxx.bin 是这些特定功能的二进制文件的合集。您可以仅烧录 factory/factory_xxx.bin 到起始地址为 0 的 flash 空间中，或者根据 download.config 文件中的信息将若干个二进制文件烧录到 flash 中对应起始地址的空间中。
- at_customize.bin 提供了用户分区表，该表列出了 ble_data.bin 分区、SSL 证书分区、MQTT 证书分区以及 factory_param_XXX.bin 分区和其它一些分区的的起始地址和分区大小。您可以通过 AT 命令*AT+FS* 和*AT+SYSFLASH* 来读和写该文件中罗列的分区里的内容。
- factory_param_XXX.bin 指明了不同 ESP32-C2 模组之间的硬件配置（见下表）。请确保您的模组使用了正确的固件。更多有关该参数文件的信息请参考修改参数的固件

表 1: ESP32-C2 模组的硬件配置

模组	UART 管脚 (TX、RX、CTS、RTS)	Factory Parameter Bin
ESP32C2-4MB 系列	• GPIO7 • GPIO6 • GPIO5 • GPIO4	factory_param_ESP32C2-4MB.bin

- ble_data.bin 在 ESP32-C2 工作于 Bluetooth LE 服务端的时候提供蓝牙服务；
- server_cert.bin、server_key.bin 和 server_ca.bin 是 SSL 服务端示例证书；
- client_cert.bin、client_key.bin 和 client_ca.bin 是 SSL 客户端示例证书；
- mqtt_cert.bin、mqtt_key.bin 和 mqtt_ca.bin 是 MQTT SSL 客户端示例证书；

如果某些功能没有使用到，则不需要将相应的二进制文件下载到 flash 中。

2.3 我该选哪种类型的固件？

ESP-AT 固件有以下几种类型，其中下载或准备固件的工作量自上而下依次递增，支持的模组类型也自上而下依次递增。

2.3.1 官方发布版固件（推荐）

官方发布版固件又称“发布版固件”、“官方固件”、“默认固件”，为乐鑫官方团队测试并发布的固件，固件会根据内部开发计划周期性发布，此种固件可直接基于乐鑫 OTA 服务器升级固件。如果 **官方发布版固件**完全满足您的项目需求，建议您优先选择 **官方发布版固件**。如果官方固件不支持您的模组，您可以根据硬件差异，选择和您的模组硬件配置相近的固件进行测试验证。

- 获取途径：*ESP32-C2 AT 固件*
- 优点：
  - 稳定
  - 可靠
  - 获取固件工作量小
- 缺点：
  - 更新周期长
  - 覆盖的模组有限
- 参考文档：
  - 硬件连接
  - 固件下载及烧录指南

  - 有关 ESP-AT 固件支持/不支持哪些芯片系列，请参考 ESP-AT GitHub 首页 readme.md

2.3.2 GitHub 临时固件

GitHub 临时固件为每次将代码推送到 GiHub 时都会生成但并未达到固件发布周期条件的固件，或者说是开发中的固件，包括 **官方发布版固件**的临时版本和适配过但是不计划正式发布的固件，其中前者可直接基于乐鑫 OTA 服务器升级固件。

- 获取途径：请参考*如何从 GitHub 下载最新临时版本 AT 固件*。
- 优点：
  - 实时性强，新的特性和漏洞修补都会实时同步出来。
  - 包含一些非正式发布的固件，如基于 SDIO 通讯的固件、基于 SPI 通讯的固件。
  - 获取固件工作量小。
- 缺点：基于非正式发布的 commit 生成的固件未经过完整的测试，可能会存在一些风险，需要您自己做完整的测试。

2.3.3 修改参数的固件

修改参数的固件指的是只修改参数区域而并不需要重新编译的固件，适用于固件功能满足项目要求、但只有某些参数不满足的情况下，如出厂波特率、UART IO 管脚的等参数的变更，此种固件可直接基于乐鑫 OTA 服务器升级固件。

- 关于如何修改参数文件，请参考*如何生成出厂参数二进制文件*。
- 优点：
  - 不需要重新编译固件。
  - 固件稳定、可靠。
- 缺点：需要基于发布版的固件修改，更新周期长，覆盖的模组有限。

2.3.4 自行编译的固件

当您需要进行二次开发时可采用此种方式。需要自己部署 OTA 服务器以支持 OTA 功能。

- 关于如何自行编译固件，请参考*编译 ESP-AT 工程*。
- 优点：功能、周期自己可控。
- 缺点：需要自己搭建环境编译。

2.4 获取固件后，接来下做什么？

当您获取到固件后，请参考*硬件连接* 与*固件下载及烧录指南* 连接 PC 和 ESP 设备、并将固件烧录至设备。

Chapter 3

AT 命令集

本章将具体介绍如何使用各类 AT 命令。

3.1 基础 AT 命令

- *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+SYSFLASH*：查询或读写 flash 用户分区
- *AT+FS*：文件系统操作
- *AT+FSMOUNT*：挂载/卸载文件系统
- *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

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-C2 - 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。
  - 0 表示立即重启
  - 最大 Deep-sleep 时长约为 28.8 天（2^{31}-1 毫秒）。

说明

- 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。

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-C2 设备：支持范围为 80 ~ 5000000
- **<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-C2 设备：支持范围为 80 ~ 5000000
- **<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 模式
    - 单 Wi-Fi 模式
      - 射频模块将根据 AP 的 DTIM 定期关闭
    - 单 BLE 模式
      - 在 BLE 广播态下，射频模块将根据广播间隔定期关闭
      - 在 BLE 连接态下，射频模块将根据连接间隔定期关闭
  - 2：Light-sleep 模式
    - 单 Wi-Fi 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
    - 单 BLE 模式
      - 在 BLE 广播态下，CPU 将自动进入睡眠，射频模块也将根据广播间隔定期关闭
      - 在 BLE 连接态下，CPU 将自动进入睡眠，射频模块也将根据连接间隔定期关闭
  - 3：Modem-sleep listen interval 模式
    - 单 Wi-Fi 模式
      - 射频模块将根据*AT+CWJAP* 命令设置的 listen interval 参数定期关闭
    - 单 BLE 模式
      - 在 BLE 广播态下，射频模块将根据广播间隔定期关闭
      - 在 BLE 连接态下，射频模块将根据连接间隔定期关闭

说明

- 当禁用睡眠模式后，Bluetooth LE 不可以被初始化。当 Bluetooth LE 初始化后，不可以禁用睡眠模式。
- Modem-sleep 模式和 Light-sleep 模式均可以在 Wi-Fi 模式或者 BLE 模式下设置，但在 Wi-Fi 模式下，这两种模式只能在 station 模式下设置
- 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
- 对于 BLE 模式下的 Light-sleep 模式，用户必须确保外接 32KHz 晶振，否则，Light-sleep 模式会以 Modem-sleep 模式工作。
- AT+SLEEP 更多示例请参考文档*Sleep AT 示例*。

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>**：运行时的最小堆空间，单位：byte。当 `<minimum heap size>` 小于或接近于 10 KB 时，ESP32-C2 的 Wi-Fi 和低功耗蓝牙的功能可能会受影响。

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, Bluetooth LE SPP 及 Bluetooth SPP 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
```

或

```
// 透传模式下，Wi-Fi、socket、Bluetooth LE 或 Bluetooth 状态改变时会打印提示信息
AT+SYSMSG=4
```

3.1.13 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*。
- 烧录二级用户分区前，请参考*如何生成 PKI 文件* 生成二进制用户分区文件。
- 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
```

3.1.14 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

- 本命令会自动挂载文件系统。*AT+FS* 文件系统操作完成后，强烈建议使用*AT+FSMOUNT=0* 命令卸载文件系统，来释放大量 RAM 空间。
- 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
```

(下页继续)

(续上页)

```
AT+FS=0,2,"filename",0,100

// 列出根目录下所有文件
AT+FS=0,4,"."
```

3.1.15 AT+FSMOUNT：挂载/卸载 FS 文件系统

设置命令

命令：

```
AT+FSMOUNT=<mount>
```

响应：

```
OK
```

参数

- **<mount>**：
  - 0：卸载 FS 文件系统
  - 1：挂载 FS 文件系统

说明

- *AT+FS* 文件系统操作完成后，强烈建议使用本命令*AT+FSMOUNT=0* 命令卸载文件系统，来释放大量 RAM 空间。

示例

```
// 手动卸载文件系统
AT+FSMOUNT=0

// 手动挂载文件系统
AT+FSMOUNT=1
```

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>
OK
```

设置命令

命令：

```
AT+RFPOWER=<wifi_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-C2 设备的取值范围为 [40,84]：

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

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

- **ESP32C2-4MB AT 固件支持此命令，而 ESP32C2-2MB AT 固件由于采用了压缩 OTA 分区，因此不支持此命令。**
- 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：定时器唤醒
  - 1：保留配置
  - 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

示例

```
// 定时器唤醒
AT+SLEEPWKCFG=0,1000

// 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*

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

3.2 Wi-Fi AT 命令集

- *AT+CWMODE*：查询/设置 Wi-Fi 模式 (Station/SoftAP/Station+SoftAP)
- *AT+CWSTATE*：查询 Wi-Fi 状态和 Wi-Fi 信息
- *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-C2 SoftAP 参数
- *AT+CWLIF*：查询连接到 ESP32-C2 SoftAP 的 station 信息
- *AT+CWQIF*：断开 station 与 ESP32-C2 SoftAP 的连接
- *AT+CWDHCP*：启用/禁用 DHCP
- *AT+CWDHCPS*：查询/设置 ESP32-C2 SoftAP DHCP 分配的 IPv4 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 ESP32-C2 Station 的 MAC 地址
- *AT+CIPAPMAC*：查询/设置 ESP32-C2 SoftAP 的 MAC 地址
- *AT+CIPSTA*：查询/设置 ESP32-C2 Station 的 IP 地址
- *AT+CIPAP*：查询/设置 ESP32-C2 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
- *AT+CWJEAP*：连接 WPA2 企业版 AP
- *AT+CWHOSTNAME*：查询/设置 ESP32-C2 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 ESP32-C2 设备的 Wi-Fi 模式

命令：

```
AT+CWMODE?
```

响应：

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

设置命令

功能：

设置 ESP32-C2 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 ESP32-C2 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 ESP32-C2 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+CWSTATE：查询 Wi-Fi 状态和 Wi-Fi 信息

查询命令

功能：

查询 ESP32-C2 设备的 Wi-Fi 状态和 Wi-Fi 信息

命令：

```
AT+CWSTATE?
```

响应：

```
+CWSTATE:<state>,<"ssid">

OK
```

参数

- **<state>**：当前 Wi-Fi 状态
  - 0: ESP32-C2 station 尚未进行任何 Wi-Fi 连接
  - 1: ESP32-C2 station 已经连接上 AP，但尚未获取到 IPv4 地址
  - 2: ESP32-C2 station 已经连接上 AP，并已经获取到 IPv4 地址
  - 3: ESP32-C2 station 正在进行 Wi-Fi 连接或 Wi-Fi 重连
  - 4: ESP32-C2 station 处于 Wi-Fi 断开状态
- **<"ssid">**：目标 AP 的 SSID

说明

- 当 ESP32-C2 station 没有连接上 AP 时，推荐使用此命令查询 Wi-Fi 信息；当 ESP32-C2 station 已连接上 AP 后，推荐使用*AT+CWJAP* 命令查询 Wi-Fi 信息

3.2.3 AT+CWJAP：连接 AP

查询命令

功能：

查询与 ESP32-C2 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 ESP32-C2 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 ESP32-C2 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 63 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: ESP32-C2 station 可与任何一种加密方式的 AP 连接，包括 OPEN 和 WEP
  - 1: ESP32-C2 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，ESP32-C2 station 不重连 AP
  - [1,7200]: 断开连接后，ESP32-C2 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，ESP32-C2 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，ESP32-C2 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 1
  - 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 ESP32-C2 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
- 当 ESP32-C2 station 已连接上 AP 后，推荐使用此命令查询 Wi-Fi 信息；当 ESP32-C2 station 没有连接上 AP 时，推荐使用*AT+CWSTATE* 命令查询 Wi-Fi 信息
- 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
```

3.2.4 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，ESP32-C2 station 不重连 AP
  - [1,7200]: 断开连接后，ESP32-C2 station 每隔指定的时间与 AP 重连
- **<repeat_count>**：ESP32-C2 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: ESP32-C2 station 始终尝试连接 AP
  - [1,1000]: ESP32-C2 station 按照本参数指定的次数重连 AP

示例

```
// ESP32-C2 station 每隔 1 秒尝试重连 AP，共尝试 100 次
AT+CWRECONNCFG=1,100

// ESP32-C2 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

3.2.5 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
  - bit 8: 是否显示 `WAPI_PSK` 认证方式的 AP
  - bit 9: 是否显示 `OWE` 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
```

3.2.6 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
  - 8: WAPI_PSK
  - 9: OWE
- **<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，默认值为：0
  - 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

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"
```

3.2.7 AT+CWQAP：断开与 AP 的连接

执行命令

命令：

```
AT+CWQAP
```

响应：

```
OK
```

3.2.8 AT+CWSAP：配置 ESP32-C2 SoftAP 参数

查询命令

功能：

查询 ESP32-C2 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 ESP32-C2 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 ~ 63 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 ESP32-C2 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
- 默认 SSID 因设备而异，因为它由设备的 MAC 地址组成。您可以使用*AT+CWSAP?* 查询默认的 SSID。

示例

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

3.2.9 AT+CWLIF：查询连接到 ESP32-C2 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 ESP32-C2 SoftAP 的 station 的 IP 地址
- **<mac>**：连接到 ESP32-C2 SoftAP 的 station 的 MAC 地址

说明

- 本指令无法查询静态 IP，仅支持在 ESP32-C2 SoftAP 和连入的 station DHCP 均使能的情况下有效

3.2.10 AT+CWQIF：断开 station 与 ESP32-C2 SoftAP 的连接

执行命令

功能：

断开所有连入 ESP32-C2 SoftAP 的 station

命令：

```
AT+CWQIF
```

响应：

```
OK
```

设置命令

功能：

断开某个连入 ESP32-C2 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

3.2.11 AT+CWDHCP：启用/禁用 DHCP

查询命令

命令：

```
AT+CWDHCP?
```

响应：

```
+CWDHCP:<state>
OK
```

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:
    - 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 IPv4 地址的命令会相互影响，如*AT+CIPSTA* 和*AT+CIPAP*
  - 若启用 DHCP，则静态 IPv4 地址会被禁用
  - 若启用静态 IPv4，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
```

3.2.12 AT+CWDHCPS：查询/设置 ESP32-C2 SoftAP DHCP 分配的 IPv4 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 ESP32-C2 SoftAP DHCP 服务器分配的 IPv4 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>**：ESP32-C2 SoftAP DHCP 服务器 IPv4 地址池的起始 IP
- **<end IP>**：ESP32-C2 SoftAP DHCP 服务器 IPv4 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 ESP32-C2 SoftAP 模式使能，且开启 DHCP server 的情况下使用
- 设置的 IPv4 地址范围必须与 ESP32-C2 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
```

3.2.13 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
```

3.2.14 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，ESP32-C2 设备支持的 PHY mode 见：Wi-Fi 协议模式
- 默认情况下，ESP32-C2 设备的 PHY mode 是 802.11bgn 模式

3.2.15 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，ESP32-C2 设备支持的 PHY mode 见：Wi-Fi 协议模式
- 默认情况下，ESP32-C2 设备的 PHY mode 是 802.11bgn 模式

3.2.16 AT+CIPSTAMAC：查询/设置 ESP32-C2 Station 的 MAC 地址

查询命令

功能：

查询 ESP32-C2 Station 的 MAC 地址

命令：

```
AT+CIPSTAMAC?
```

响应：

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

设置命令

功能：

设置 ESP32-C2 Station 的 MAC 地址

命令：

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

响应：

```
OK
```

参数

- **<mac>**：字符串参数，表示 ESP32-C2 Station 的 MAC 地址

说明

- 若*AT+SYSSTORE=1*，配置更改将保存到 NVS 分区
- ESP32-C2 Station 的 MAC 地址与 ESP32-C2 SoftAP 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"
```

3.2.17 AT+CIPAPMAC：查询/设置 ESP32-C2 SoftAP 的 MAC 地址

查询命令

功能：

查询 ESP32-C2 SoftAP 的 MAC 地址

命令：

```
AT+CIPAPMAC?
```

响应：

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

设置命令

功能：

设置 ESP32-C2 SoftAP 的 MAC 地址

命令：

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

响应：

```
OK
```

参数

- **<mac>**：字符串参数，表示 ESP32-C2 SoftAP 的 MAC 地址

说明

- 若*AT+SYSSTORE=1*，配置更改将保存到 NVS 分区
- 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"
```

3.2.18 AT+CIPSTA：查询/设置 ESP32-C2 Station 的 IP 地址

查询命令

功能：

查询 ESP32-C2 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 ESP32-C2 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 ESP32-C2 station 的 IPv4 地址
- **<”gateway”>**：网关
- **<”netmask”>**：子网掩码
- **<”ipv6 addr”>**：ESP32-C2 station 的 IPv6 地址

说明

- 使用查询命令时，只有当 ESP32-C2 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 IPv4 地址，则禁用 DHCP
  - 若启用 DHCP，则禁用静态 IPv4 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"
```

3.2.19 AT+CIPAP：查询/设置 ESP32-C2 SoftAP 的 IP 地址

查询命令

功能：

查询 ESP32-C2 SoftAP 的 IP 地址

命令：

```
AT+CIPAP?
```

响应：

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

OK
```

设置命令

功能：

设置 ESP32-C2 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 ESP32-C2 SoftAP 的 IPv4 地址
- **<" gateway" >**：网关
- **<" netmask" >**：子网掩码
- **<" ipv6 addr" >**：ESP32-C2 SoftAP 的 IPv6 地址

说明

- 本设置命令仅适用于 IPv4 网络，不适用于 IPv6 网络
- 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 IPv4 地址，则禁用 DHCP
  - 若启用 DHCP，则禁用静态 IPv4 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"
```

3.2.20 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>][,<"esptouch v2 key">]
```

响应：

```
OK
```

参数

- **<type>**：类型
  - 1: ESP-TOUCH
  - 2: AirKiss
  - 3: ESP-TOUCH+AirKiss
  - 4: ESP-TOUCH v2
- **<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

- **<" esptouch v2 key" >**：ESP-TOUCH v2 的解密秘钥，用于解密 Wi-Fi 密码和自定义数据。长度应为 16 字节。

说明

- 更多有关 SmartConfig 的信息，请参考 ESP-TOUCH 使用指南；
- SmartConfig 仅支持在 ESP32-C2 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 ESP32-C2 尝试连接 AP；
- 消息 `+SCRD:<length>,<rvd data>` 表示 ESP-Touch v2 成功获取到自定义数据；
- 消息 `Smartconfig connected Wi-Fi` 表示成功连接到 AP；
- 因为 ESP32-C2 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
```

3.2.21 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
```

3.2.22 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 ESP32-C2 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
```

3.2.23 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+CWMODE=1
AT+CWJAP="1234567890","1234567890"
AT+MDNS=1,"espressif","_iot",8080
AT+MDNS=0
```

3.2.24 AT+CWJEAP：连接 WPA2 企业版 AP

查询命令

功能：

查询 ESP32-C2 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

3.2.25 AT+CWHOSTNAME：查询/设置 ESP32-C2 Station 的主机名称

查询命令

功能：

查询 ESP32-C2 Station 的主机名称

命令：

```
AT+CWHOSTNAME?
```

响应：

```
+CWHOSTNAME:<hostname>

OK
```

设置命令

功能：

设置 ESP32-C2 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>**：ESP32-C2 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"
```

3.2.26 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 ESP32-C2 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，各国国家代码请参考 ISO 3166-1 alpha-2 标准。
- **<start_channel>**：起始信号道，范围：[1,14]
- **<total_channel_count>**：信道总个数

说明

- 详细说明请参考：Wi-Fi 国家/地区代码。
- 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

- *AT+CIPV6*: 启用/禁用 IPv6 网络 (IPv6)
- *AT+CIPSTATE*：查询 TCP/UDP/SSL 连接信息
- *AT+CIPSTATUS (弃用)*：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
- *[仅适用数据模式] +++*: 退出数据模式
- *AT+CIPSEND*：在普通传输模式 或 Wi-Fi 透传模式 下发送数据
- *AT+CIPSENDL*：在普通传输模式 下并行发送长数据
- *AT+CIPSENDLCFG*：设置*AT+CIPSENDL* 命令的属性
- *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+CIPSNTPINTV*：查询/设置 SNTP 时间同步的间隔
- *AT+CIPFWVER*：查询服务器已有的 AT 固件版本
- *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/设置套接字接收模式
- *AT+CIPRECVDATA*：获取被动接收模式下的套接字数据
- *AT+CIPRECVLEN*：查询被动接收模式下套接字数据的长度
- *AT+PING*：ping 对端主机
- *AT+CIPDNS*：查询/设置 DNS 服务器信息
- *AT+CIPTCPOPT*：查询/设置套接字选项

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+CIPSTATE：查询 TCP/UDP/SSL 连接信息

查询命令

命令：

```
AT+CIPSTATE?
```

响应：

当有连接时，AT 返回：

```
+CIPSTATE:<link ID>,<"type">,<"remote IP">,<remote port>,<local port>,<tetype>

OK
```

当没有连接时，AT 返回：

```
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>**：ESP32-C2 本地端口值
- **<tetype>**:
  - 0: ESP32-C2 设备作为客户端
  - 1: ESP32-C2 设备作为服务器

3.3.3 AT+CIPSTATUS (弃用)：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>**：ESP32-C2 station 接口的状态
  - 0: ESP32-C2 station 为未初始化状态
  - 1: ESP32-C2 station 为已初始化状态，但还未开始 Wi-Fi 连接
  - 2: ESP32-C2 station 已连接 AP，获得 IP 地址
  - 3: ESP32-C2 station 已建立 TCP、UDP 或 SSL 传输
  - 4: ESP32-C2 设备所有的 TCP、UDP 和 SSL 均断开
  - 5: ESP32-C2 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>**：ESP32-C2 本地端口值
- **<tetype>**:
  - 0: ESP32-C2 设备作为客户端
  - 1: ESP32-C2 设备作为服务器

说明

- 建议您使用*AT+CWSTATE* 命令查询 Wi-Fi 状态，使用*AT+CIPSTATE* 命令查询 TCP/UDP/SSL 状态。

3.3.4 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
```

3.3.5 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：

单连接模式下，返回：

```
CONNECT

OK
```

多连接模式下，返回：

```
<link ID>,CONNECT

OK
```

参数

- **<link ID>**：网络连接 ID (0 ~ 4)，用于多连接的情况。该参数范围取决于 menuconfig 中的两个配置项。一个是 AT 组件中的配置项 AT_SOCKET_MAX_CONN_NUM，默认值为 5。另一个是 LWIP 组件中的配置项 LWIP_MAX_SOCKETS，默认值为 10。要修改该参数的范围，您需要修改配置项 AT_SOCKET_MAX_CONN_NUM 的值并确保该值不大于 LWIP_MAX_SOCKETS 的值。（请参考*编译 ESP-AT 工程* 获取更多信息。）
- **<" 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>**：配置套接字的 SO_KEEPALIVE 选项（参考：SO_KEEPALIVE 介绍），单位：秒。

- 范围：[0,7200]。
  - 0：禁用 keep-alive 功能；（默认）
  - 1 ~ 7200：开启 keep-alive 功能。TCP_KEEPIDLE 值为 **<keep_alive>**，TCP_KEEPINTVL 值为 1，TCP_KEEPCNT 值为 3。
- 本命令中的 <keep_alive> 参数与*AT+CIPTCPOPT* 命令中的 <keep_alive> 参数相同，最终值由后设置的命令决定。如果运行本命令时不设置 <keep_alive> 参数，则默认使用上次配置的值。
- **<" 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

示例

```
AT+CIPSTART="TCP","iot.espressif.cn",8000
AT+CIPSTART="TCP","192.168.101.110",1000
AT+CIPSTART="TCP","192.168.101.110",2500,60
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：

单连接模式下，返回：

```
CONNECT

OK
```

多连接模式下，返回：

```
<link ID>,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>**：ESP32-C2 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)，ESP32-C2 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)，ESP32-C2 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)，ESP32-C2 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
- 如果想接收长度大于 1460 字节的 UDP 包，请自行编译 *ESP-AT* 工程，在第五步配置工程里选择：Component config -> LWIP -> Enable reassembly incoming fragmented IP4 packets

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：

```
// 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：

单连接模式下，返回：

```
CONNECT

OK
```

多连接模式下，返回：

```
<link ID>,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，" 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>**：配置套接字的 SO_KEEPALIVE 选项（参考：SO_KEEPALIVE 介绍），单位：秒。
- 范围：[0,7200]。
  - 0：禁用 keep-alive 功能；（默认）
  - 1 ~ 7200：开启 keep-alive 功能。TCP_KEEPIDLE 值为 **<keep_alive>**，TCP_KEEPINTVL 值为 1，TCP_KEEPCNT 值为 3。
- 本命令中的 <keep_alive> 参数与*AT+CIPTCPOPT* 命令中的 <keep_alive> 参数相同，最终值由后设置的命令决定。如果运行本命令时不设置 <keep_alive> 参数，则默认使用上次配置的值。
- **<" 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
- 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

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"
```

3.3.6 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。

3.3.7 [仅适用数据模式] +++：退出数据模式

特殊执行命令

功能：

退出数据模式，进入命令模式

Command:

```
// 仅适用数据模式
+++
```

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，+ 字符会被当做普通数据发送出去
- 本条特殊执行命令没有命令回复
- 请至少间隔 1 秒再发下一条 AT 命令

3.3.8 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。如果您要发送的数据长度大于 8192 字节，请使用*AT+CIPSENDL* 命令发送。

命令：

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

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

// 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，ESP32-C2 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：8192 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

- 您可以使用*AT+CIPTCPOPT* 命令来为每个 TCP 连接配置套接字选项。例如：设置 <so_sndtimeo> 为 5000，则 TCP 发送会在 5 秒内返回，无论成功还是失败。这可以节省 MCU 等待 AT 命令回复的时间。

3.3.9 AT+CIPSENDL：在普通传输模式下并行发送长数据

设置命令

功能：

普通传输模式 下，指定长度，并行发送数据（AT 命令端口接收数据和 AT 往对端发送数据是并行的）。您可以使用*AT+CIPSENDLCFG* 命令配置本条命令。如果您要发送的数据长度小于 8192 字节，您也可以使用*AT+CIPSEND* 命令发送。

命令：

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

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

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

响应：

```
OK

>
```

上述响应表示 AT 进入数据模式 并且已准备好接收 AT 命令端口的数据，此时您可以输入数据，一旦 AT 命令端口接收到数据，数据就会被发往底层协议，数据传输开始。

如果传输已开始，系统会根据*AT+CIPSENDLCFG* 配置上报消息：

```
+CIPSENDL:<had sent len>,<port recv len>
```

如果传输被+++ 命令取消，系统返回：

```
SEND CANCELLED
```

如果所有数据没有被完全发出去，系统最终返回：

```
SEND FAIL
```

如果所有数据被成功发往协议栈，系统最终返回：

```
SEND OK
```

当连接断开时，您可以发送+++ 命令取消传输，同时 ESP32-C2 设备会从数据模式 退出。否则，AT 命令端口会一直接收数据，直到收到指定的 <length> 长度数据后，才会退出数据模式。

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：2^{31} - 1 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
- **<had sent len>**：成功发到底层协议栈的数据长度
- **<port recv len>**：AT 命令端口收到的数据总长度

说明

- 您可以使用*AT+CIPTCPOPT* 命令来为每个 TCP 连接配置套接字选项。例如：设置 <so_sndtimeo> 为 5000，则 TCP 发送会在 5 秒内返回，无论成功还是失败。这可以节省 MCU 等待 AT 命令回复的时间。

3.3.10 AT+CIPSENDLCFG: 设置 AT+CIPSENDL 命令的属性

查询命令

功能：

查询*AT+CIPSENDL* 命令的配置

命令：

```
AT+CIPSENDLCFG?
```

响应：

```
+CIPSENDLCFG:<report size>,<transmit size>

OK
```

设置命令

功能：

设置*AT+CIPSENDL* 命令的配置

命令：

```
AT+CIPSENDLCFG:<report size>[,<transmit size>]
```

响应：

```
OK
```

参数

- **<report size>**: *AT+CIPSENDL* 命令中的上报块大小。默认值：1024。范围：[100,2 [20]]。例如：设置 <report size> 值为 100，则*AT+CIPSENDL* 命令回复里的 <had sent len> 上报序列为（100，200，300，400，……）。最后的 <had sent len> 上报值总是等于实际传输的数据长度。
- **<transmit size>**: *AT+CIPSENDL* 命令中的传输块大小，它指定了数据发往协议栈的数据块大小。默认值：2920。范围：[100,2920]。如果收到的数据长度大于等于 <transmit size>，则数据会被立即发往底层协议栈；否则，数据会等待 20 毫秒后再发往底层协议栈。

说明

- 对于吞吐量小但对实时性要求高的设备，推荐您设置较小的 <transmit size>。也推荐您通过*AT+CIPTCPOPT* 命令设置 TCP_NODELAY 属性。
- 对于吞吐量大的设备，推荐您设置较大的 <transmit size>。也推荐您阅读如何提高 *ESP-AT* 吞吐性能。

3.3.11 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：8192 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
- 您可以使用*AT+CIPTCPOPT* 命令来为每个 TCP 连接配置套接字选项。例如：设置 <so_sndtimeo> 为 5000，则 TCP 发送会在 5 秒内返回，无论成功还是失败。这可以节省 MCU 等待 AT 命令回复的时间。

3.3.12 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>
```

响应：

```
<link ID>,CLOSED

OK
```

执行命令

功能：

关闭单连接模式下的 TCP/UDP/SSL 连接

```
AT+CIPCLOSE
```

响应：

```
CLOSED

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

3.3.13 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" >**: ESP32-C2 SoftAP 的 IPv4 地址
- **<" APIP6LL" >**: ESP32-C2 SoftAP 的 IPv6 本地链路地址
- **<" APIP6GL" >**: ESP32-C2 SoftAP 的 IPv6 全局地址
- **<" APMAC" >**: ESP32-C2 SoftAP 的 MAC 地址
- **<" STAIP" >**: ESP32-C2 station 的 IPv4 地址
- **<" STAIP6LL" >**: ESP32-C2 station 的 IPv6 本地链路地址
- **<" STAIP6GL" >**: ESP32-C2 station 的 IPv6 全局地址
- **<" STAMAC" >**: ESP32-C2 station 的 MAC 地址
- **<" ETHIP" >**: ESP32-C2 ethernet 的 IPv4 地址
- **<" ETHIP6LL" >**: ESP32-C2 ethernet 的 IPv6 本地链路地址
- **<" ETHIP6GL" >**: ESP32-C2 ethernet 的 IPv6 全局地址
- **<" ETHMAC" >**: ESP32-C2 ethernet 的 MAC 地址

说明

- 只有当 ESP32-C2 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

3.3.14 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
```

3.3.15 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"
- **<CA enable>**：
  - 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
- 关闭服务器时参数 <"type"> 和 <CA enable> 必须省略

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
```

3.3.16 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，范围：[1,5]。如果您想修改该参数的上限阈值，请参考*AT+CIPSTART* 命令中参数 <link ID> 的描述。

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
```

3.3.17 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
```

3.3.18 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 ESP32-C2 上电进入 Wi-Fi 透传模式
  - 1: 开启 ESP32-C2 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>**：配置套接字的 SO_KEEPALIVE 选项（参考：SO_KEEPALIVE 介绍），单位：秒。
- 范围：[0,7200]。
  - 0：禁用 keep-alive 功能；（默认）
  - 1 ~ 7200：开启 keep-alive 功能。TCP_KEEPIDLE 值为 **<keep_alive>**，TCP_KEEPINTVL 值为 1，TCP_KEEPCNT 值为 3。
- 本命令中的 <keep_alive> 参数与*AT+CIPTCPOPT* 命令中的 <keep_alive> 参数相同，最终值由后设置的命令决定。如果运行本命令时不设置 <keep_alive> 参数，则默认使用上次配置的值。

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 ESP32-C2 上电进入 Wi-Fi 透传模式
  - 1: 开启 ESP32-C2 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
```

3.3.19 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，ESP32-C2 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
```

3.3.20 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"
```

3.3.21 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+CWMODE=1
AT+CWJAP="1234567890","1234567890"
AT+CIPSNTPCFG=1,8,"cn.ntp.org.cn","ntp.sjtu.edu.cn"
AT+CIPSNTPTIME?
+CIPSNTPTIME:Tue Oct 19 17:47:56 2021
OK

或

AT+CWMODE=1
AT+CWJAP="1234567890","1234567890"
AT+CIPSNTPCFG=1,530
AT+CIPSNTPTIME?
+CIPSNTPTIME:Tue Oct 19 15:17:56 2021
OK
```

3.3.22 AT+CIPSNTPINTV：查询/设置 SNTP 时间同步的间隔

查询命令

命令：

```
AT+CIPSNTPINTV?
```

响应：

```
+CIPSNTPINTV:<interval second>

OK
```

设置命令

命令：

```
AT+CIPSNTPINTV=<interval second>
```

响应：

```
OK
```

参数

- **<interval second>**：SNTP 时间同步间隔。单位：秒。范围：[15,4294967]。

说明

- 配置了时间同步间隔，意味着 ESP32-C2 多久一次向 NTP 服务器获取新的时间。

示例

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

OK

// 每小时同步一次时间
AT+CIPSNTPINTV=3600

OK
```

3.3.23 AT+CIPFWVER：查询服务器已有的 AT 固件版本

查询命令

功能：

查询服务器已有的 ESP32-C2 AT 固件版本

命令：

```
AT+CIPFWVER?
```

响应：

```
+CIPFWVER:<"version">

OK
```

参数

- **<" version" >**：ESP32-C2 AT 固件版本

说明

- 在选择要升级的 OTA 版本时，强烈不建议从高版本向低版本升级。

3.3.24 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 ESP32-C2 设备的升级状态

命令：

```
AT+CIUPDATE?
```

响应：

```
+CIPUPDATE:<state>

OK
```

执行命令

功能：

在阻塞模式下通过 OTA 升级到 TCP 服务器上最新版本的固件

命令：

```
AT+CIUPDATE
```

响应：

请参考设置命令中的响应

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：

如果 OTA 在阻塞模式下成功，返回：

```
+CIPUPDATE:1
+CIPUPDATE:2
+CIPUPDATE:3
+CIPUPDATE:4

OK
```

如果 OTA 在非阻塞模式下成功，返回：

```
OK
+CIPUPDATE:1
+CIPUPDATE:2
+CIPUPDATE:3
+CIPUPDATE:4
```

如果在阻塞模式下 OTA 失败，返回：

```
+CIPUPDATE:<state>

ERROR
```

如果在非阻塞模式下 OTA 失败，返回：

```
OK
+CIPUPDATE:<state>
+CIPUPDATE:-1
```

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*编译 ESP-AT 工程*。
- **<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>**:
  - 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: 非阻塞模式下 OTA 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 功能或者使用*AT+USEROTA* 或者*AT+WEBSERVER* 命令，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。
- OTA 过程的超时时间为 3 分钟。
- 非阻塞模式响应中的 `OK` 和 `+CIPUPDATE:<state>` 在输出顺序上没有严格意义上的先后顺序。OK 可能在 `+CIPUPDATE:<state>` 之前输出，也有可能在 `+CIPUPDATE:<state>` 之后输出。
- 不建议升级到旧版本。
- 请参考*如何实现 OTA 升级* 获取更多 OTA 命令。

示例

```
AT+CWMODE=1
AT+CWJAP="1234567890","1234567890"
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
```

3.3.25 AT+CIPDINFO：设置 +IPD 消息详情

查询命令

命令：

```
AT+CIPDINFO?
```

响应：

```
+CIPDINFO:true
OK
```

或

```
+CIPDINFO:false
OK
```

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
```

3.3.26 AT+CIPSSLCCONF：查询/设置 SSL 客户端配置

查询命令

功能：

查询 ESP32-C2 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，ESP32-C2 将在下次上电时基于本配置建立 SSL 连接。
- 如果您想使用自己的证书或者使用多套证书，请参考文档：*如何生成 PKI 文件*。
- 如果 <auth_mode> 配置为 2 或者 3，为了校验服务器的证书有效期，请在发送*AT+CIPSTART* 命令前确保 ESP32-C2 已获取到当前时间。（您可以发送*AT+CIPSNTPCFG* 命令来配置 SNTP，获取当前时间，发送*AT+CIPSNTPTIME?* 命令查询当前时间。）

3.3.27 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。

3.3.28 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。

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

查询命令

功能：

查询 ESP32-C2 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。

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

查询命令

功能：

查询 ESP32-C2 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。

3.3.31 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
```

3.3.32 AT+CIPRECVMODE：查询/设置套接字接收模式

查询命令

功能：

查询套接字接收模式

命令：

```
AT+CIPRECVMODE?
```

响应：

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

设置命令

命令：

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

响应：

```
OK
```

参数

- **<mode>**：套接字数据接收模式，默认值：0。
  - 0: 主动模式，ESP-AT 将所有接收到的套接字数据立即发送给主机 MCU，头为 “+IPD”（套接字接收窗口为 5760 字节，每次向 MCU 最大发送 2920 字节有效数据）。
  - 1: 被动模式，ESP-AT 将所有接收到的套接字数据保存到内部缓存区（套接字接收窗口，默认值为 5760 字节），等待 MCU 读取。对于 TCP 和 SSL 连接，如果缓存区满了，将阻止套接字传输；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 在被动模式下收到套接字数据时，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>` 表示缓存区中套接字数据的总长度。
- 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，缓冲的套接字数据仍然存在，MCU 仍然可以读取，直到发送*AT+CIPCLOSE*（AT 作为客户端）或*AT+CIPSERVER=0,1*（AT 作为服务器）。换句话说，如果 `+IPD` 已经被报告，那么在你发送*AT+CIPCLOSE* 或发送*AT+CIPSERVER=0,1* 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
```

3.3.33 AT+CIPRECVDATA：获取被动接收模式下的套接字数据

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
```

3.3.34 AT+CIPRECVLEN：查询被动接收模式下套接字数据的长度

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
```

3.3.35 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"
```

3.3.36 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 服务器
  - 0: 启用自动获取 DNS 服务器设置，DNS 服务器将会恢复为 208.67.222.222 和 8.8.8.8，只有当 ESP32-C2 station 完成了 DHCP 过程，DNS 服务器才有可能会更新。
  - 1: 启用手动设置 DNS 服务器信息，如果不设置参数 <DNS IPx> 的值，则使用默认值 208.67.222.222 和 8.8.8.8。
- **<DNS IP1>**：第一个 DNS 服务器 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 服务器 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 服务器 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 ESP32-C2 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"
```

3.3.37 AT+CIPTCPOPT：查询/设置套接字选项

查询命令

功能：

查询当前套接字选项

命令：

```
AT+CIPTCPOPT?
```

响应：

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

设置命令

命令：

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

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

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>**：配置套接字的 SO_LINGER 选项（参考：SO_LINGER 介绍），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>**：配置套接字的 TCP_NODELAY 选项（参考：TCP_NODELAY 介绍），默认值：0。
  - 0: 禁用 TCP_NODELAY
  - 1: 启用 TCP_NODELAY
- **<so_sndtimeo>**：配置套接字的 SO_SNDTIMEO 选项（参考：SO_SNDTIMEO 介绍），单位：毫秒，默认值：0。
- **<keep_alive>**：配置套接字的 SO_KEEPALIVE 选项（参考：SO_KEEPALIVE 介绍），单位：秒。
- 范围：[0,7200]。
  - 0：禁用 keep-alive 功能；（默认）
  - 1 ~ 7200：开启 keep-alive 功能。TCP_KEEPIDLE 值为 **<keep_alive>**，TCP_KEEPINTVL 值为 1，TCP_KEEPCNT 值为 3。
- 本命令中的 <keep_alive> 参数与*AT+CIPSTART* 命令中的 <keep_alive> 参数相同，最终值由后设置的命令决定。如果运行本命令时不设置 <keep_alive> 参数，则默认使用上次配置的值。

说明

- 在配置套接字选项前，**请充分了解该选项功能，以及配置后可能的影响**。
- SO_LINGER 选项不建议配置较大的值。例如配置 SO_LINGER 值为 60，则*AT+CIPCLOSE* 命令在收不到对端 TCP FIN 包情况下，会导致 AT 阻塞 60 秒，从而无法响应其它命令。因此，SO_LINGER 建议保持默认值。

- TCP_NODELAY 选项适用于吞吐量小但对实时性要求高的场景。开启后，*LwIP* 会加快 TCP 的发送，但如果网络环境较差，会由于重传而导致吞吐降低。因此，TCP_NODELAY 建议保持默认值。
- SO_SNDTIMEO 选项适用于*AT+CIPSTART* 命令未配置 keepalive 参数的应用场景。配置本选项后，*AT+CIPSEND*、*AT+CIPSENDL*、*AT+CIPSENDEX* 命令将会在该超时内退出，无论是否发送成功。这里，SO_SNDTIMEO 建议配置为 5 ~ 10 秒。
- SO_KEEPALIVE 选项适用于主动定时检测连接是否断开的应用场景，通常 AT 作为 TCP 服务器时建议配置该选项。配置本选项后，会增加额外的网络带宽。SO_KEEPALIVE 建议配置值不小于 60 秒。

3.4 Bluetooth® Low Energy AT 命令集

注意：ESP32C2-4MB AT 固件支持 BluFi，而 ESP32C2-2MB AT 固件不支持 BluFi。如果您需要 ESP32C2-2MB 支持 BluFi 功能，请自行编译 *ESP-AT 工程*，在第五步配置工程里选择：

- `Component config -> Bluetooth -> Bluetooth -> Host -> NimBLE - BLE only`
- `Component config -> Bluetooth -> NimBLE Options -> Enable blufi functionality`

- *AT+BLUFI*：开启或关闭 BluFi
- *AT+BLUFINAME*：查询/设置 BluFi 设备名称
- *AT+BLUFISEND*: 发送 BluFi 用户自定义数据

3.4.1 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
```

3.4.2 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?
```

3.4.3 AT+BLUFISEND: 发送 BluFi 用户自定义数据

设置命令

功能：

发送 BluFi 用户自定义数据给手机端

命令：

```
AT+BLUFISEND=<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，开始传输数据。

若数据传输成功，则提示：

```
OK
```

参数

- **<length>**：数据长度，单位：字节。

说明

- 自定义数据的长度不能超过 600 字节。
- 如果 ESP 收到手机发来的用户自定义数据，那么会以 +BLUFIDATA:<len>,<data> 格式打印。

示例

```
AT+BLUFISEND=4
// 提示 ">" 符号后，输入 4 字节的数据即可，如 "1234"，然后数据会被自动发送给手机
```

3.5 MQTT AT 命令集

- *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+MQTTALPN*：设置 MQTT 应用层协议协商（ALPN）
- *AT+MQTTCONN*：连接 MQTT Broker
- *AT+MQTTPUB*：发布 MQTT 消息（字符串）
- *AT+MQTTPUBRAW*：发布长 MQTT 消息
- *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*

3.5.1 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>**：
  - 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。
- 如果 <scheme> 配置为 3、5、8、10，为了校验服务器的证书有效期，请在发送*AT+MQTTCONN* 命令前确保 ESP32-C2 已获取到当前时间。（您可以发送*AT+CIPSNTPCFG* 命令来配置 SNTP，获取当前时间，发送*AT+CIPSNTPTIME?* 命令查询当前时间。）

3.5.2 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。

3.5.3 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。

3.5.4 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。

3.5.5 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。

3.5.6 AT+MQTTALPN：设置 MQTT 应用层协议协商（ALPN）

设置命令

功能：

设置 MQTT 应用层协议协商（ALPN）

命令：

```
AT+MQTTALPN=<LinkID>,<alpn_counts>[,<"alpn">][,<"alpn">][,<"alpn">]
```

响应：

```
OK
```

参数

- **<LinkID>**：当前仅支持 link ID 0。
- **<alpn_counts>**：<”alpn”> 参数个数。范围：[0,5]。
  - 0：清除 MQTT ALPN 配置
  - [1,5]：设置 MQTT ALPN 配置
- **<”alpn”>**：字符串参数，表示 ClientHello 中的 ALPN，用户可以发送多个 ALPN 字段到服务器。

说明

- 整条 AT 命令长度应小于 256 字节。
- 只有在 MQTT 基于 TLS 或 WSS 时，MQTT ALPN 字段才会生效。
- 应在设置 `AT+MQTTUSERCFG` 后再使用 `AT+MQTTALPN`。

示例

```
AT+CWMODE=1
AT+CWJAP="ssid","password"
AT+CIPSNTPCFG=1,8,"ntp1.aliyun.com","ntp2.aliyun.com"
AT+MQTTUSERCFG=0,5,"ESP32-C2","espressif","1234567890",0,0,""
AT+MQTTALPN=0,2,"mqtt-ca.cn","mqtt-ca.us"
AT+MQTTCONN=0,"192.168.200.2",8883,1
```

3.5.7 AT+MQTTCONN：连接 MQTT Broker

查询命令

功能：

查询 ESP32-C2 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。如果 MQTT 建立连接后又断开，则无法再次使用本命令重新建立连接，您需要先发送*AT+MQTTCLEAN=0* 命令清理信息，重新配置参数，再建立新的连接。
  - 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）。

3.5.8 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。如果您发布消息的数据量相对较多，已经超过了单条 AT 指令的长度阈值 256 字节，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。

示例

```
AT+CWMODE=1
AT+CWJAP="ssid","password"
AT+MQTTUSERCFG=0,1,"ESP32-C2","espressif","1234567890",0,0,""
AT+MQTTCONN=0,"192.168.10.234",1883,0
AT+MQTTPUB=0,"topic","\"{\"timestamp\":\"20201121085253\"}\"",0,0
```

3.5.9 AT+MQTTPUBRAW：发布长 MQTT 消息

设置命令

功能：

通过 topic 发布长 MQTT 消息。如果您发布消息的数据量相对较少，不大于单条 AT 指令的长度阈值 256 字节，也可以使用*AT+MQTTPUB* 命令。

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 ESP32-C2 设备的最大长度受到可利用内存的限制。
- **<qos>**：发布消息的 QoS，参数可选 0、1、或 2，默认值：0。
- **<retain>**：发布 retain。

3.5.10 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。

3.5.11 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。

3.5.12 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。

3.5.13 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

下页继续

表 2 – 续上页

错误类型	错误码
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
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

下页继续

表 2 - 续上页

错误类型	错误码
AT_MQTT_TOPIC_IS_NULL	0x6042
AT_MQTT_TOPIC_IS_OVERLENGTH	0x6043
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

3.5.14 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。

3.6 HTTP AT 命令集

- *AT+HTTPCLIENT*：发送 HTTP 客户端请求
- *AT+HTTPGETSIZE*：获取 HTTP 资源大小
- *AT+HTTPCGET*：获取 HTTP 资源
- *AT+HTTPCPOST*：Post 指定长度的 HTTP 数据
- *AT+HTTPCPUT*：Put 指定长度的 HTTP 数据
- *AT+HTTPURLCFG*：设置/获取长的 HTTP URL
- *HTTP AT 错误码*

3.6.1 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

- 如果包含 URL 的整条命令的长度超过了 256 字节，请先使用*AT+HTTPURLCFG* 命令预配置 URL，然后本命令里的 `<"url">` 参数需要设置为 `""`。
- 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*ESP-AT 固件差异*），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*编译 ESP-AT 工程*）。

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"
```

3.6.2 AT+HTTPGETSIZE：获取 HTTP 资源大小

设置命令

命令：

```
AT+HTTPGETSIZE=<"url">[,<tx size>][,<rx size>][,<timeout>]
```

响应：

```
+HTTPGETSIZE:<size>

OK
```

参数

- **<" url" >**：HTTP URL。
- **<tx size>**：HTTP 发送缓存大小。单位：字节。默认值：2048。范围：[0,10240]。
- **<rx size>**：HTTP 接收缓存大小。单位：字节。默认值：2048。范围：[0,10240]。
- **<timeout>**：网络超时。单位：毫秒。默认值：5000。范围：[0,180000]。
- **<size>**：HTTP 资源大小。

说明

- 如果包含 URL 的整条命令的长度超过了 256 字节，请先使用*AT+HTTPURLCFG* 命令预配置 URL，然后本命令里的 <"url"> 参数需要设置为 ""。
- 某些已发布的固件默认不支持 HTTP 客户端命令（详情请见*ESP-AT 固件差异*），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*编译 ESP-AT 工程*）。

示例

```
AT+HTTPGETSIZE="http://www.baidu.com/img/bdlogo.gif"
```

3.6.3 AT+HTTPCGET：获取 HTTP 资源

设置命令

命令：

```
AT+HTTPCGET=<"url">[,<tx size>][,<rx size>][,<timeout>]
```

响应：

```
+HTTPCGET:<size>,<data>
OK
```

参数

- **<" url" >**：HTTP URL。
- **<tx size>**：HTTP 发送缓存大小。单位：字节。默认值：2048。范围：[0,10240]。
- **<rx size>**：HTTP 接收缓存大小。单位：字节。默认值：2048。范围：[0,10240]。
- **<timeout>**：网络超时。单位：毫秒。默认值：5000。范围：[0,180000]。

说明

- 如果包含 URL 的整条命令的长度超过了 256 字节，请先使用*AT+HTTPURLCFG* 命令预配置 URL，然后本命令里的 <"url"> 参数需要设置为 ""。

3.6.4 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>]**：HTTP 请求头。可发送多个请求头给服务器。

说明

- 如果包含 URL 的整条命令的长度超过了 256 字节，请先使用*AT+HTTPURLCFG* 命令预配置 URL，然后本命令里的 <"url"> 参数需要设置为 ""。

3.6.5 AT+HTTPCPUT：Put 指定长度的 HTTP 数据

设置命令

命令：

```
AT+HTTPCPUT=<"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 Put 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>]**：HTTP 请求头。可发送多个请求头给服务器。

说明

- 如果包含 URL 的整条命令的长度超过了 256 字节，请先使用*AT+HTTPURLCFG* 命令预配置 URL，然后本命令里的 <"url"> 参数需要设置为 ""。

3.6.6 AT+HTTPURLCFG：设置/获取长的 HTTP URL

查询命令

命令：

```
AT+HTTPURLCFG?
```

响应：

```
[+HTTPURLCFG:<url length>,<data>]
OK
```

设置命令

命令：

```
AT+HTTPURLCFG=<url length>
```

响应：

```
OK

>
```

符号 > 表示 AT 准备好接收串口数据，此时您可以输入 URL，当数据长度达到参数 <url length> 的值时，系统返回：

```
SET OK
```

参数

- **<url length>**：HTTP URL 长度。单位：字节。
  - 0：清除 HTTP URL 配置。
  - [8,8192]：设置 HTTP URL 配置。
- **<data>**：HTTP URL 数据。

3.6.7 HTTP AT 错误码

- HTTP 客户端：

HTTP 客户端错误码	说明
0x7000	建立连接失败
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`)。

3.7 信令测试 AT 命令

- *AT+FACTPLCP*：发送长 PLCP 或短 PLCP

3.7.1 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。

3.8 驱动 AT 命令

- *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

3.8.1 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。
- ESP32-C2 设备的取值范围为 [0,4]。

通道	管脚
0	GPIO0
1	GPIO1
2	GPIO2
3	GPIO3
4	GPIO4

- **<atten>**：衰减值。
- 0: 0 dB 衰减，有效测量范围为 [0, 750] mV。
- 1: 2.5 dB 衰减，有效测量范围为 [0, 1050] mV。
- 2: 6 dB 衰减，有效测量范围为 [0, 1300] mV。
- 3: 11 dB 衰减，有效测量范围为 [0, 2500] mV。
- **<raw data>**：ADC 通道值。

说明

- ESP-AT 只支持 ADC1。
- ESP32-C2 支持 12 位宽度。
- 对于如何将通道值转换为电压，可以参考 ADC 转换。

示例

```
// ESP32-C2 设备设置为 0 dB 衰减，有效测量范围为 [0, 750] mV
// 电压为 2048 / 4095 * 750 = 375.09 mV
AT+DRVADC=0,0
+DRVADC:2048

OK
```

3.8.2 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
```

3.8.3 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
```

3.8.4 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
```

3.8.5 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
```

3.8.6 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
```

3.8.7 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
```

3.8.8 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
```

3.8.9 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。

3.8.10 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
```

3.8.11 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
```

3.8.12 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
```

3.9 Web 服务器 AT 命令

- *AT+WEBSERVER*: 启用/禁用通过 Web 服务器配置 Wi-Fi 连接

3.9.1 AT+WEBSERVER: 启用/禁用通过 Web 服务器配置 Wi-Fi 连接

设置命令

命令：

```
AT+WEBSERVER=<enable>,<server_port>,<connection_timeout>
```

响应：

```
OK
```

参数

- **<enable>**: 启用/禁用 Web 服务器。
  - 0: 禁用 Web 服务器并释放相关资源。
  - 1: 启用 Web 服务器，您可以通过微信或者浏览器配置 Wi-Fi 连接信息。
- **<server_port>**: Web 服务器端口号。
- **<connection_timeout>**: 每个连接的超时时间。单位：秒。范围：[21,60]。

说明

- 有两种方法可以提供 Web 服务器所需的 HTML 文件。一种是使用 FAT 文件系统，此时需要启用 AT FS 命令。另一种是使用嵌入文件来存储 HTML 文件（默认设置）。
- 请确保开放的 socket 的最大数目不能小于 12，您可以在 menuconfig 中设置此项 `./build.py menuconfig > Component config > LWIP > Max number of open sockets`，然后重新编译工程（参考文档编译 *ESP-AT* 工程）。
- AT 固件默认不支持 Web 服务器 AT 命令（参考文档 see *ESP-AT* 固件差异），但您可以在 menuconfig 中设置支持 Web 服务器 AT 命令 `./build.py menuconfig > Component config > AT > AT WEB Server command support`，然后重新编译工程（参考文档编译 *ESP-AT* 工程）。
- ESP-AT 在 ESP32-C2 系列设备中支持强制门户 (captive portal)，可参考示例。
- 更多示例可参考文档*Web Server AT* 示例。
- 该命令的实现开源，源码请参考 at/src/at_web_server_cmd.c。
- 请参考如何实现 *OTA* 升级 获取更多 OTA 命令。

示例

```
// 启用 Web 服务器，端口 80，每个连接的超时时间 50 秒
AT+WEBSERVER=1,80,50

// 禁用 Web 服务器
AT+WEBSERVER=0
```

3.10 用户 AT 命令

- *AT+USERRAM*：操作用户的空闲 RAM
- *AT+USEROTA*：根据指定 URL 升级固件
- *AT+USERWKMCUCFG*：设置 AT 唤醒 MCU 的配置
- *AT+USERMCUSLEEP*：MCU 指示自己睡眠状态
- *AT+USERDOCS*：查询固件对应的用户文档链接

3.10.1 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
```

(下页继续)

(续上页)

```
AT+USERRAM=3,64,100

// 释放用户 RAM 空间
AT+USERRAM=0
```

3.10.2 AT+USEROTA：根据指定 URL 升级固件

ESP-AT 在运行时，升级到指定 URL 上的新固件。

设置命令

功能：

升级到 URL 指定版本的固件

命令：

```
AT+USEROTA=<url len>
```

响应：

```
OK

>
```

上述响应表示 AT 已准备好接收 URL，此时您可以输入 URL，当 AT 接收到的 URL 长度达到 <url len> 后，返回：

```
Recv <url len> bytes
```

AT 输出上述信息之后，升级过程开始。如果升级完成，返回：

```
OK
```

如果参数错误或者固件升级失败，返回：

```
ERROR
```

参数

- **<url len>**：URL 长度。最大值：8192 字节

说明

- 您可以从 *GitHub Actions* 里下载 所需要的 OTA 固件，也可以自行编译 *ESP-AT* 工程 生成所需要的 OTA 固件。
- 如果您使用的是 ESP32C2-2MB 模组配置，OTA 固件为 build/custom_ota_binaries/esp-at.bin.xz.packed；如果您使用的是 ESP32C2-4MB 模组配置，OTA 固件为 build/esp-at.bin。
- 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。
- 建议升级 AT 固件后，调用 *AT+RESTORE* 恢复出厂设置。
- AT+USEROTA 支持 HTTP 和 HTTPS。
- AT 输出 > 字符后，数据中的特殊字符不需要转义字符进行转义，也不需要以新行结尾（CR-LF）。

- 当 URL 为 HTTPS 时，不建议 SSL 认证。如果要求 SSL 认证，您必须自行生成 PKI 文件然后将它们下载到对应的分区中，之后在 AT+USEROTA 命令的实现代码中加载证书。对于 PKI 文件请参考*如何生成 PKI 文件*。对于 AT+USEROTA 命令，可参考 ESP-AT 工程提供的示例 USEROTA。
- 请参考*如何实现 OTA 升级* 获取更多 OTA 命令。

示例

```
AT+USEROTA=36

OK

>
Recv 36 bytes

OK
```

3.10.3 AT+USERWKMCUCFG：设置 AT 唤醒 MCU 的配置

设置命令

功能：

此命令配置 AT 如何检查 MCU 的唤醒状态，以及 AT 如何唤醒 MCU。

- 当 MCU 是醒来的状态，AT 将直接向 MCU 发送数据，不会发送唤醒信号。
- 当 MCU 是睡眠的状态，AT 准备向 MCU 主动发送数据时（主动发送的数据和*ESP-AT 消息报告* 中定义的相同），AT 会先发送唤醒信号再发送数据。MCU 被唤醒或者超时后会清除唤醒信号。

命令：

```
AT+USERWKMCUCFG=<enable>,<wake mode>,<wake number>,<wake signal>,<delay time>[,
↪<check mcu awake method>]
```

响应：

```
OK
```

参数

- **<enable>**：启用或禁用唤醒配置。
  - 0：禁用唤醒 MCU 配置
  - 1：使能唤醒 MCU 配置
- **<wake mode>**：唤醒模式。
  - 1：GPIO 唤醒
  - 2：UART 唤醒
- **<wake number>**：该参数的意义取决于 <wake mode> 的值。
  - 如果 <wake mode> 是 1，<wake number> 代表唤醒管脚 GPIO 编号。用户需要保证配置的唤醒管脚没有用作其它用途，否则需要用户做兼容性处理。
  - 如果 <wake mode> 是 2，<wake number> 代表唤醒 UART 编号。当前只支持 1，即支持 UART1 唤醒 MCU。
- **<wake signal>**：该参数的意义取决于 <wake mode> 的值。
  - 如果 <wake mode> 是 1，<wake signal> 代表唤醒电平。
    - 0：低电平
    - 1：高电平
  - 如果 <wake mode> 是 2，<wake signal> 代表唤醒字节。范围：[0,255]。
- **<delay time>**：最大等待时间。单位：毫秒。范围：[0,60000]。该参数的意义取决于 <wake mode> 的值。

  - 如果 <wake mode> 是 1，则在 <delay time> 期间内，将一直保持 <wake signal> 电平。<delay time> 到后，则反转 <wake signal> 电平。
  - 如果 <wake mode> 是 2，则立即发送 <wake signal> 字节，进入等待直到超时。
- **<check mcu awake method>**：AT 检查 MCU 是否处于醒来的状态。
  - Bit 0：是否开启与*AT+USERMCUSLEEP* 命令的关联。默认开启。即：收到 AT+USERMCUSLEEP=0 命令，指示 MCU 醒来；收到 AT+USERMCUSLEEP=1 命令，指示 MCU 睡眠。
  - Bit 1：是否开启与*AT+SLEEP=1/2/3* 命令的关联。默认禁用。即：收到 AT+SLEEP=0 命令，指示 MCU 醒来；收到 AT+USERMCUSLEEP=1/2/3 命令，指示 MCU 睡眠。
  - Bit 2：是否开启 <delay time> 超时后指示 MCU 醒来功能。默认禁用。即：禁用时，delay time 后，指示 MCU 睡眠；使能时，delay time 后，指示 MCU 醒来。
  - Bit 3（暂未实现）：是否开启 GPIO 指示 MCU 醒来功能。默认不支持。

说明

- 此命令只需要配置一次。
- 每次 AT 向 MCU 主动发送数据前，会先发送唤醒信号再进入等待，<delay time> 时间到了之后直接发送数据。此超时会降低与 MCU 间的传输效率。
- 如果在 <delay time> 毫秒之前，AT 收到 <check mcu awake method> 里的任意唤醒事件，则立即清除唤醒状态；否则会等待 <delay time> 超时后，会自动清除唤醒状态。

示例

```
// 使能唤醒 MCU 配置。每次 AT 向 MCU 发送数据前，会先使用 Wi-Fi 模块的 GPIO18␣
↪管脚，高电平唤醒 MCU，同时保持高电平 10 秒。
AT+USERWKMCUCFG=1,1,18,1,10000,3

// 禁用唤醒 MCU 配置
AT+USERWKMCUCFG=0
```

3.10.4 AT+USERMCUSLEEP：MCU 指示自己睡眠状态

设置命令

功能：

在*AT+USERWKMCUCFG* 命令的 <check mcu awake method> Bit 0 配置情况下，此命令才会生效。用于告知 AT 当前 MCU 的睡眠状态。

命令：

```
AT+USERMCUSLEEP=<state>
```

响应：

```
OK
```

参数

- **<state>**：
  - 0：指示 MCU 醒来。
  - 1：指示 MCU 睡眠。

示例

```
// MCU 告知 AT 当前 MCU 醒来
AT+USERMCUSLEEP=0
```

3.10.5 AT+USERDOCS：查询固件对应的用户文档链接

查询命令

功能：

查询当前运行固件对应的中英文用户文档链接。

命令：

```
AT+USERDOCS?
```

响应：

```
+USERDOCS:<"en url">
+USERDOCS:<"cn url">

OK
```

参数

- **<" en url" >**：英文文档链接
- **<" cn url" >**：中文文档链接

示例

```
AT+USERDOCS?
+USERDOCS:"https://docs.espressif.com/projects/esp-at/en/latest/esp32c2/index.html"
+USERDOCS:"https://docs.espressif.com/projects/esp-at/zh_CN/latest/esp32c2/index.
↪html"

OK
```

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*

3.11 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*

3.12 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。

备注： AT 命令里的参数保存，是通过 NVS 库实现的。因此，如果命令配置相同的参数值，则不会写入 flash；如果命令配置不同的参数值，flash 也不会被频繁擦除。

3.13 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。响应的最后一条消息必然是 OK 或者 ERROR。

表 3: 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）
SET OK	URL 已经成功设置（针对于*AT+HTTPURLCFG* 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 4: 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
+SCRD:<length>,"<reserved data>"	ESP-Touch v2 已获取自定义数据
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
+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 在非透传模式下，已收到来自网络的数据
透传模式 下的数据	ESP-AT 在透传模式下，已收到来自网络或蓝牙的数据
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

下页继续

表 4 – 续上页

ESP-AT 消息报告	说明
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
+BLUFIDATA:<len>,<data>	ESP 设备收到从手机端发送的 BluFi 用户自定义数据

Chapter 4

AT 命令示例

4.1 TCP-IP AT 示例

本文档主要介绍在 ESP32-C2 设备上运行*TCP/IP AT 命令* 命令的详细示例。

4.1.1 ESP32-C2 设备作为 TCP 客户端建立单连接

1. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

   响应：

```
OK
```

2. 连接到路由器。
   命令：

```
AT+CWJAP="espressif","1234567890"
```

   响应：

```
WIFI CONNECTED
WIFI GOT IP
```

(下页继续)

(续上页)

```
OK
```

说明：

- 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。

3. 查询 ESP32-C2 设备 IP 地址。
   命令：

```
AT+CIPSTA?
```

   响应：

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"

OK
```

   说明：

   - 您的查询结果可能与上述响应中的不同。

4. PC 与 ESP32-C2 设备连接同一个路由。
   在 PC 上使用网络调试工具，创建一个 TCP 服务器。例如 TCP 服务器的 IP 地址为 `192.168.3.102`，端口为 `8080`。
5. ESP32-C2 设备作为客户端通过 TCP 连接到 TCP 服务器，服务器 IP 地址为 `192.168.3.102`，端口为 `8080`。
   命令：

```
AT+CIPSTART="TCP","192.168.3.102",8080
```

   响应：

```
CONNECT

OK
```

6. 发送 4 字节数据。
   命令：

```
AT+CIPSEND=4
```

   响应：

```
OK

>
```

   输入 4 字节数据，例如输入数据是 `test`，之后 AT 将会输出以下信息。

```
Recv 4 bytes

SEND OK
```

   说明：

   - 若输入的字节数目超过 `AT+CIPSEND` 命令设定的长度 (n)，则系统会响应 `busy p...`，并发送数据的前 n 个字节，发送完成后响应 `SEND OK`。

7. 接收 4 字节数据。
   假设 TCP 服务器发送 4 字节的数据（数据为 `test`），则系统会提示：

```
+IPD,4:test
```

4.1.2 ESP32-C2 设备作为 TCP 服务器建立多连接

当 ESP32-C2 设备作为 TCP 服务器时，必须通过*AT+CIPMUX=1* 命令使能多连接，因为可能有多个 TCP 客户端连接到 ESP32-C2 设备。

以下是 ESP32-C2 设备作为 softAP 建立 TCP 服务器的示例；如果是 ESP32-C2 设备作为 station，可在连接路由器后按照同样方法建立服务器。

1. 设置 Wi-Fi 模式为 softAP。
   命令：

   ```
   AT+CWMODE=2
   ```

   响应：

   ```
   OK
   ```

2. 使能多连接。
   命令：

   ```
   AT+CIPMUX=1
   ```

   响应：

   ```
   OK
   ```

3. 设置 softAP。
   命令：

   ```
   AT+CWSAP="ESP32_softAP","1234567890",5,3
   ```

   响应：

   ```
   OK
   ```

4. 查询 softAP 信息。
   命令：

   ```
   AT+CIPAP?
   ```

   响应：

   ```
   AT+CIPAP?
   +CIPAP:ip:"192.168.4.1"
   +CIPAP:gateway:"192.168.4.1"
   +CIPAP:netmask:"255.255.255.0"

   OK
   ```

   说明：
   - 您查询到的地址可能与上述响应中的不同。

5. 建立 TCP 服务器，默认端口为 333。
   命令：

   ```
   AT+CIPSERVER=1
   ```

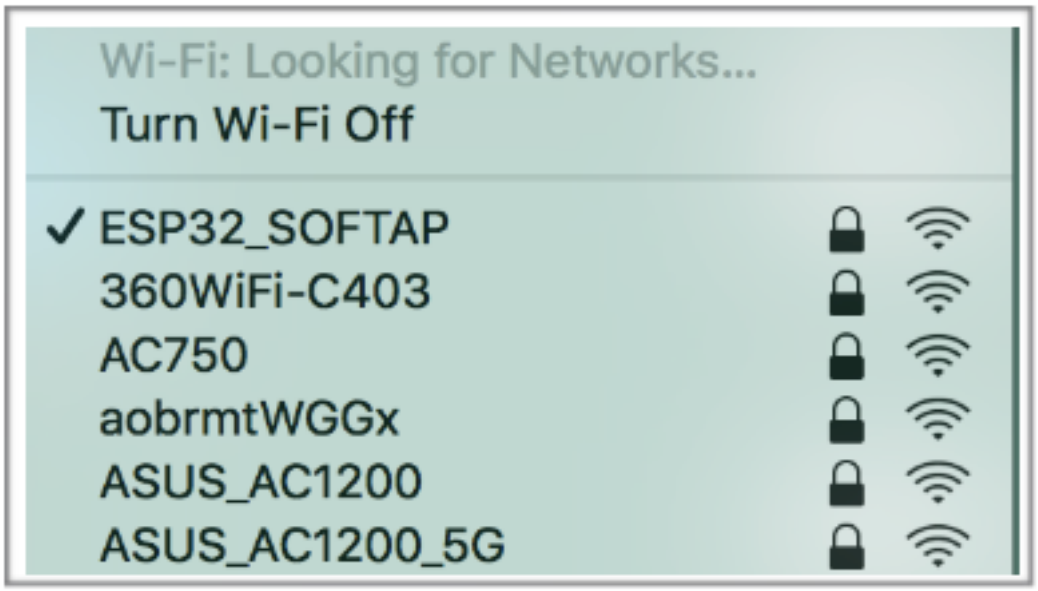
   响应：

   ```
   OK
   ```

6. PC 连接到 ESP32-C2 设备的 softAP。
7. 在 PC 上使用网络调试工具创建一个 TCP 客户端，连接到 ESP32-C2 设备创建的 TCP 服务器。
8. 发送 4 字节数据到网络连接 ID 为 0 的链路上。
   命令：

   ```
   AT+CIPSEND=0,4
   ```

   响应：




```
OK

>
```

输入 4 字节数据，例如输入数据是 `test`，之后 AT 将会输出以下信息。

```
Recv 4 bytes

SEND OK
```

说明：

- 若输入的字节数目超过 `AT+CIPSEND` 命令设定的长度 (n)，则系统会响应 `busy p...`，并发送数据的前 n 个字节，发送完成后响应 `SEND OK`。

9. 从网络连接 ID 为 0 的链路上接收 4 字节数据。
   假设 TCP 服务器发送 4 字节的数据（数据为 `test`），则系统会提示：

```
+IPD,0,4:test
```

10. 关闭 TCP 连接。
    命令：

```
AT+CIPCLOSE=0
```

响应：

```
0,CLOSED

OK
```

4.1.3 远端 IP 地址和端口固定的 UDP 通信

1. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。

命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

说明：

- 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。

3. 查询 ESP32-C2 设备 IP 地址。
   命令：

```
AT+CIPSTA?
```

响应：

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"

OK
```

说明：

- 您的查询结果可能与上述响应中的不同。

4. PC 与 ESP32-C2 设备连接到同一个路由。
   在 PC 上使用网络调试工具，创建一个 UDP 传输。例如 PC 的 IP 地址为 `192.168.3.102`，端口为 `8080`。
5. 使能多连接。
   命令：

```
AT+CIPMUX=1
```

响应：

```
OK
```

6. 创建 UDP 传输。分配网络连接 ID 为 4，远程 IP 地址为 `192.168.3.102`，远端端口为 `8080`，本地端口为 `1112`，模式为 `0`。

重要： `AT+CIPSTART` 命令的参数 `mode` 决定了 UDP 通信的远端 IP 地址和端口是否固定。若参数为 0，则代表系统会分配一个特定网络连接 ID，以确保通信过程中远端的 IP 地址和端口不会被改变，且数据发送端和数据接收端不会被其它设备代替。

命令：

```
AT+CIPSTART=4,"UDP","192.168.3.102",8080,1112,0
```

响应：

```
4,CONNECT

OK
```

说明：

- `"192.168.3.102"` 和 `8080` 为 UDP 传输的远端 IP 地址和远端端口，也就是 PC 建立的 UDP 配置。
- `1112` 为 ESP32-C2 设备的 UDP 本地端口，您可自行设置，如不设置则为随机值。
- `0` 表示 UDP 远端 IP 地址和端口是固定的，不能更改。比如有另外一台 PC 创建了 UDP 端并且向 ESP32-C2 设备端口 1112 发送数据，ESP32-C2 设备仍然会接收来自 UDP 端口 1112 的数据，如果使用 AT 命令 `AT+CIPSEND=4,X`，那么数据仍然只会发送到第一台 PC 端。但是如果这个参数未设置为 `0`，那么数据将会被发送到新的 PC 端。

7. 发送 7 字节数据到网络连接 ID 为 4 的链路上。
   命令：

```
AT+CIPSEND=4,7
```

   响应：

```
OK

>
```

   输入 7 字节数据，例如输入数据是 abcdefg，之后 AT 将会输出以下信息。

```
Recv 7 bytes

SEND OK
```

   说明：
   - 若输入的字节数目超过 AT+CIPSEND 命令设定的长度 (n)，则系统会响应 busy p...，并发送数据的前 n 个字节，发送完成后响应 SEND OK。
8. 从网络连接 ID 为 4 的链路上接收 4 字节数据。
   假设 PC 发送 4 字节的数据（数据为 test），则系统会提示：

```
+IPD,4,4:test
```

9. 关闭网络连接 ID 为 4 的 UDP 连接。
   命令：

```
AT+CIPCLOSE=4
```

   响应：

```
4,CLOSED

OK
```

4.1.4 远端 IP 地址和端口可变的 UDP 通信

1. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

   响应：

```
OK
```

2. 连接到路由器。
   命令：

```
AT+CWJAP="espressif","1234567890"
```

   响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

   说明：
   - 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。
3. 查询 ESP32-C2 设备 IP 地址。
   命令：

```
AT+CIPSTA?
```

响应：

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"

OK
```

说明：

- 您的查询结果可能与上述响应中的不同。

4. PC 与 ESP32-C2 设备连接到同一个路由。
   在 PC 上使用网络调试工具，创建一个 UDP 传输。例如 IP 地址为 192.168.3.102，端口为 8080。
5. 使能单连接。
   命令：

```
AT+CIPMUX=0
```

响应：

```
OK
```

6. 创建 UDP 传输。远程 IP 地址为 192.168.3.102，远端端口为 8080，本地端口为 1112，模式为 2。
   命令：

```
AT+CIPSTART="UDP","192.168.3.102",8080,1112,2
```

响应：

```
CONNECT

OK
```

说明：

- "192.168.3.102" 和 *8080* 为 UDP 传输的远端 IP 地址和远端端口，也就是 PC 建立的 UDP 配置。
- 1112 为 ESP32-C2 设备的 UDP 本地端口，您可自行设置，如不设置则为随机值。
- 2 表示当前 UDP 传输建立后，UDP 传输远端信息仍然会更改；UDP 传输的远端信息会自动更改为最近一次与 ESP32-C2 设备 UDP 通信的远端 IP 地址和端口。

7. 发送 4 字节数据。
   命令：

```
AT+CIPSEND=4
```

响应：

```
OK

>
```

输入 4 字节数据，例如输入数据是 test，之后 AT 将会输出以下信息。

```
Recv 4 bytes

SEND OK
```

说明：

- 若输入的字节数目超过 AT+CIPSEND 命令设定的长度 (n)，则系统会响应 busy p...，并发送数据的前 n 个字节，发送完成后响应 SEND OK。

8. 发送 UDP 包给其它 UDP 远端。例如发送 4 字节数据，远端主机的 IP 地址为 192.168.3.103，远端端口为 1000。
   若需要发 UDP 包给其它 UDP 远端，只需指定对方 IP 地址和端口即可。
   命令：

```
AT+CIPSEND=4,"192.168.3.103",1000
```

响应：

```
OK

>
```

输入 4 字节数据，例如输入数据是 test，之后 AT 将会输出以下信息。

```
Recv 4 bytes

SEND OK
```

9. 接收 4 字节数据。
   假设 PC 发送 4 字节的数据（数据为 test），则系统会提示：

```
+IPD,4:test
```

10. 关闭 UDP 连接。
    命令：

```
AT+CIPCLOSE
```

响应：

```
CLOSED

OK
```

4.1.5 ESP32-C2 设备作为 SSL 客户端建立单连接

1. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。
   命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

说明：

- 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。

3. 查询 ESP32-C2 设备 IP 地址。
   命令：

```
AT+CIPSTA?
```

响应：

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
```

（下页继续）

(续上页)

```
+CIPSTA:netmask:"255.255.255.0"

OK
```

说明：

- 您的查询结果可能与上述响应中的不同。

4. PC 与 ESP32-C2 设备连接同一个路由。
5. 在 PC 上使用 OpenSSL 命令，创建一个 SSL 服务器。例如 SSL 服务器的 IP 地址为 `192.168.3.102`，端口为 `8070`。
   命令：

```
openssl s_server -cert /home/esp-at/components/customized_partitions/raw_data/
↪server_cert/server_cert.crt -key /home/esp-at/components/customized_
↪partitions/raw_data/server_key/server.key -port 8070
```

   响应：

```
ACCEPT
```

6. ESP32-C2 设备作为客户端通过 SSL 连接到 SSL 服务器，服务器 IP 地址为 `192.168.3.102`，端口为 `8070`。
   命令：

```
AT+CIPSTART="SSL","192.168.3.102",8070
```

   响应：

```
CONNECT

OK
```

7. 发送 4 字节数据。
   命令：

```
AT+CIPSEND=4
```

   响应：

```
OK

>
```

   输入 4 字节数据，例如输入数据是 `test`，之后 AT 将会输出以下信息。

```
Recv 4 bytes

SEND OK
```

   说明：

   - 若输入的字节数目超过 `AT+CIPSEND` 命令设定的长度 (n)，则系统会响应 `busy p...`，并发送数据的前 n 个字节，发送完成后响应 `SEND OK`。

8. 接收 4 字节数据。
   假设 TCP 服务器发送 4 字节的数据（数据为 `test`），则系统会提示：

```
+IPD,4:test
```

4.1.6 ESP32-C2 设备作为 SSL 服务器建立多连接

当 ESP32-C2 设备作为 SSL 服务器时，必须通过*AT+CIPMUX=1* 命令使能多连接，因为可能有多个客户端连接到 ESP32-C2 设备。

以下是 ESP32-C2 设备作为 softAP 建立 SSL 服务器的示例；如果是 ESP32-C2 设备作为 station，可在连接路由器后，参照本示例中的建立连接 SSL 服务器的相关步骤。

1. 设置 Wi-Fi 模式为 softAP。
   命令：

```
AT+CWMODE=2
```

   响应：

```
OK
```

2. 使能多连接。
   命令：

```
AT+CIPMUX=1
```

   响应：

```
OK
```

3. 配置 ESP32-C2 softAP。
   命令：

```
AT+CWSAP="ESP32_softAP","1234567890",5,3
```

   响应：

```
OK
```

4. 查询 softAP 信息。
   命令：

```
AT+CIPAP?
```

   响应：

```
AT+CIPAP?
+CIPAP:ip:"192.168.4.1"
+CIPAP:gateway:"192.168.4.1"
+CIPAP:netmask:"255.255.255.0"

OK
```

   说明：
   - 您查询到的地址可能与上述响应中的不同。

5. 建立 SSL 服务器，端口为 8070。
   命令：

```
AT+CIPSERVER=1,8070,"SSL"
```

   响应：

```
OK
```

6. PC 连接到 ESP32-C2 设备的 softAP。
7. 在 PC 上使用 OpenSSL 命令，创建一个 SSL 客户端，连接到 ESP32-C2 设备创建的 SSL 服务器。
   命令：

```
openssl s_client -host 192.168.4.1 -port 8070
```

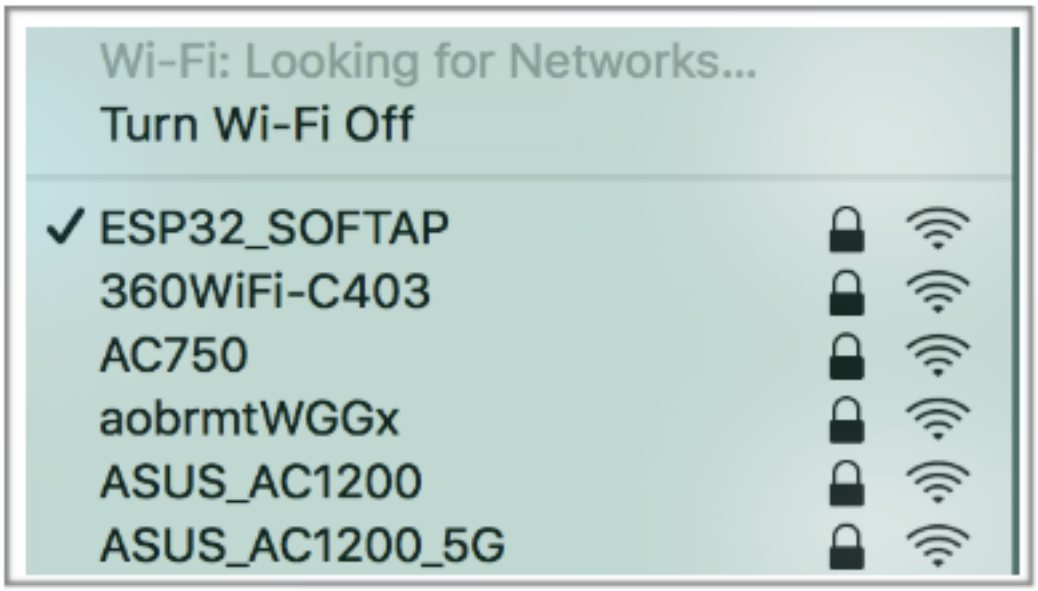
   ESP32-C2 设备上的响应：

```
CONNECT
```

8. 发送 4 字节数据到网络连接 ID 为 0 的链路上。
   命令：

```
AT+CIPSEND=0,4
```

   响应：




```
OK

>
```

输入 4 字节数据，例如输入数据是 `test`，之后 AT 将会输出以下信息。

```
Recv 4 bytes

SEND OK
```

说明：

- 若输入的字节数目超过 `AT+CIPSEND` 命令设定的长度 (n)，则系统会响应 `busy p...`，并发送数据的前 n 个字节，发送完成后响应 `SEND OK`。

9. 从网络连接 ID 为 0 的链路上接收 4 字节数据。
   假设 SSL 服务器发送 4 字节的数据（数据为 `test`），则系统会提示：

```
+IPD,0,4:test
```

10. 关闭 SSL 连接。
    命令：

```
AT+CIPCLOSE=0
```

响应：

```
0,CLOSED

OK
```

4.1.7 ESP32-C2 设备作为 SSL 客户端建立双向认证单连接

本示例中使用的证书是 esp-at 中默认的证书，您也可以自己生成证书，并烧录，然后您需要将下面的 SSL 服务器证书路径替换为您的证书路径。获取 SSL 证书，请参考 esp-at/tools/README.md 了解如何生成证书 bin 和烧录地址请参考 esp-at/module_config/module_name/at_customize.csv。

1. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

2. 连接到路由器。
   命令：

```
AT+CWJAP="espressif","1234567890"
```

   响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

   说明：
   - 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。
3. 设置 SNTP 服务器。
   命令：

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

   响应：

```
OK
```

   说明：
   - 您可以根据自己国家的时区设置 SNTP 服务器。
4. 查询 SNTP 时间。
   命令：

```
AT+CIPSNTPTIME?
```

   响应：

```
+CIPSNTPTIME:Mon Oct 18 20:12:27 2021
OK
```

   说明：
   - 您可以查询 SNTP 时间与实时时间是否相符来判断您设置的 SNTP 服务器是否生效。
5. 查询 ESP32-C2 设备 IP 地址。
   命令：

```
AT+CIPSTA?
```

   响应：

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"

OK
```

   说明：
   - 您的查询结果可能与上述响应中的不同。
6. PC 与 ESP32-C2 设备连接同一个路由。
7. 在 PC 上使用 OpenSSL 命令，创建一个 SSL 服务器。例如 SSL 服务器的 IP 地址为 `192.168.3.102`，端口为 `8070`。
   命令：

```
openssl s_server -CAfile /home/esp-at/components/customized_partitions/raw_
↪data/server_ca/server_ca.crt -cert /home/esp-at/components/customized_
↪partitions/raw_data/server_cert/server_cert.crt -key /home/esp-at/components/
↪customized_partitions/raw_data/server_key/server.key -port 8070 -verify_
↪return_error -verify_depth 1 -Verify 1
```

   ESP32-C2 设备上的响应：

```
ACCEPT
```

说明：

- 命令中的证书路径可以根据你的证书位置进行调整。

8. ESP32-C2 设备设置 SSL 客户端双向认证配置。
   命令：

```
AT+CIPSSLCCONF=3,0,0
```

   响应：

```
OK
```

9. ESP32-C2 设备作为客户端通过 SSL 连接到 SSL 服务器，服务器 IP 地址为 192.168.3.102，端口为 8070。
   命令：

```
AT+CIPSTART="SSL","192.168.3.102",8070
```

   响应：

```
CONNECT

OK
```

10. 发送 4 字节数据。
    命令：

```
AT+CIPSEND=4
```

    响应：

```
OK

>
```

    输入 4 字节数据，例如输入数据是 test，之后 AT 将会输出以下信息。

```
Recv 4 bytes

SEND OK
```

    说明：

    - 若输入的字节数目超过 AT+CIPSEND 命令设定的长度 (n)，则系统会响应 busy p...，并发送数据的前 n 个字节，发送完成后响应 SEND OK。

11. 接收 4 字节数据。
    假设 TCP 服务器发送 4 字节的数据（数据为 test），则系统会提示：

```
+IPD,4:test
```

4.1.8 ESP32-C2 设备作为 SSL 服务器建立双向认证多连接

当 ESP32-C2 设备作为 SSL 服务器时，必须通过*AT+CIPMUX=1* 命令使能多连接，因为可能有多个客户端连接到 ESP32-C2 设备。

以下是 ESP32-C2 设备作为 station 建立 SSL 服务器的示例；如果是 ESP32-C2 设备作为 softAP，可参考 ESP32-C2 设备作为 SSL 服务器建立多连接示例。

1. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

   响应：

```
OK
```

2. 连接到路由器。
   命令：

```
AT+CWJAP="espressif","1234567890"
```

   响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

   说明：
   - 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。

3. 查询 ESP32-C2 设备 IP 地址。
   命令：

```
AT+CIPSTA?
```

   响应：

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"

OK
```

   说明：
   - 您的查询结果可能与上述响应中的不同。

4. 使能多连接。
   命令：

```
AT+CIPMUX=1
```

   响应：

```
OK
```

5. 建立 SSL 服务器，端口为 8070。
   命令：

```
AT+CIPSERVER=1,8070,"SSL",1
```

   响应：

```
OK
```

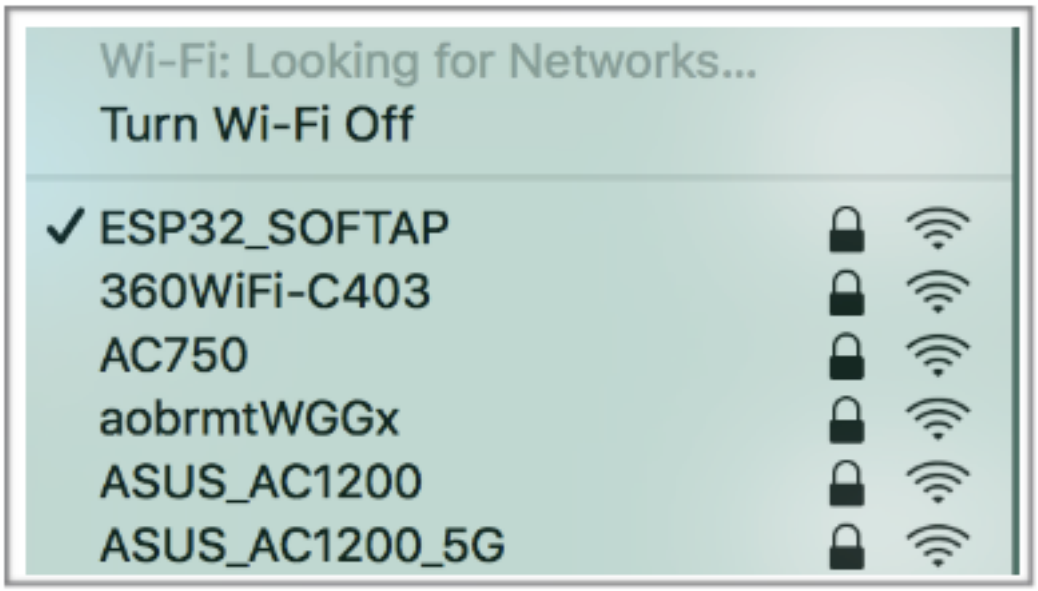
6. PC 与 ESP32-C2 设备连接同一个路由。
7. 在 PC 上使用 OpenSSL 命令，创建一个 SSL 客户端，连接到 ESP32-C2 设备创建的 SSL 服务器。
   命令：

```
openssl s_client -CAfile /home/esp-at/components/customized_partitions/raw_
↪data/client_ca/client_ca_00.crt -cert /home/esp-at/components/customized_
↪partitions/raw_data/client_cert/client_cert_00.crt -key /home/esp-at/
↪components/customized_partitions/raw_data/client_key/client_key_00.key -host␣
↪192.168.3.112 -port 8070
```

   ESP32-C2 设备上的响应：

```
0,CONNECT
```

8. 发送 4 字节数据到网络连接 ID 为 0 的链路上。
   命令：




```
AT+CIPSEND=0,4
```

响应：

```
OK

>
```

输入 4 字节数据，例如输入数据是 `test`，之后 AT 将会输出以下信息。

```
Recv 4 bytes

SEND OK
```

说明：

- 若输入的字节数目超过 `AT+CIPSEND` 命令设定的长度 (n)，则系统会响应 `busy p...`，并发送数据的前 n 个字节，发送完成后响应 `SEND OK`。

9. 从网络连接 ID 为 0 的链路上接收 4 字节数据。
假设 SSL 服务器发送 4 字节的数据（数据为 `test`），则系统会提示：

```
+IPD,0,4:test
```

10. 关闭 SSL 连接。
命令：

```
AT+CIPCLOSE=0
```

响应：

```
0,CLOSED

OK
```

11. 关闭 SSL 服务端。
命令：

```
AT+CIPSERVER=0
```

响应：

```
OK
```

4.1.9 ESP32-C2 设备作为 TCP 客户端，建立单连接，实现 UART Wi-Fi 透传

1. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

   响应：

```
OK
```

2. 连接到路由器。
   命令：

```
AT+CWJAP="espressif","1234567890"
```

   响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

   说明：
   - 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。

3. 查询 ESP32-C2 设备 IP 地址。
   命令：

```
AT+CIPSTA?
```

   响应：

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"

OK
```

   说明：
   - 您的查询结果可能与上述响应中的不同。

4. PC 与 ESP32-C2 设备连接到同一个路由。
   在 PC 上使用网络调试工具，创建一个 TCP 服务器。例如 IP 地址为 `192.168.3.102`，端口为 `8080`。
5. ESP32-C2 设备作为客户端通过 TCP 连接到 TCP 服务器，服务器 IP 地址为 `192.168.3.102`，端口为 `8080`。
   命令：

```
AT+CIPSTART="TCP","192.168.3.102",8080
```

   响应：

```
CONNECT

OK
```

6. 进入 UART Wi-Fi 透传接收模式。
   命令：

```
AT+CIPMODE=1
```

   响应：

```
OK
```

7. 进入 UART Wi-Fi 透传模式 并发送数据。
   命令：

```
AT+CIPSEND
```

响应：

```
OK

>
```

8. 停止发送数据
   在透传发送数据过程中，若识别到单独的一包数据 +++，则系统会退出透传发送。此时请至少等待 1 秒，再发下一条 AT 命令。请注意，如果直接用键盘打字输入 +++，有可能因时间太慢而不能被识别为连续的三个 +。更多介绍请参考*[仅适用透传模式] +++*。

   重要： 使用 +++ 可退出*透传模式*，回到*透传接收模式*，此时 TCP 连接仍然有效。您也可以使用 AT+CIPSEND 命令恢复透传。

9. 退出 UART Wi-Fi *透传接收模式*。
   命令：

```
AT+CIPMODE=0
```

   响应：

```
OK
```

10. 关闭 TCP 连接。
    命令：

```
AT+CIPCLOSE
```

    响应：

```
CLOSED

OK
```

4.1.10 ESP32-C2 设备作为 TCP 服务器，实现 UART Wi-Fi 透传

1. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

   响应：

```
OK
```

2. 连接到路由器。
   命令：

```
AT+CWJAP="espressif","1234567890"
```

   响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

   说明：
   - 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。

3. 设置多连接模式。
   命令：

```
AT+CIPMUX=1
```

响应：

```
OK
```

说明：

- TCP 服务器必须在多连接模式下才能开启。

4. 设置 TCP 服务器最大连接数为 1。
   命令：

```
AT+CIPSERVERMAXCONN=1
```

响应：

```
OK
```

说明：

- 透传模式是点对点的，因此 TCP 服务器的最大连接数只能是 1。

5. 开启 TCP 服务器。
   命令：

```
AT+CIPSERVER=1,8080
```

响应：

```
OK
```

说明：

- 设置 TCP 服务器端口为 8080，您也可以设置为其它端口。

6. 查询 ESP32-C2 设备 IP 地址。
   命令：

```
AT+CIPSTA?
```

响应：

```
+CIPSTA:ip:"192.168.3.112"
+CIPSTA:gateway:"192.168.3.1"
+CIPSTA:netmask:"255.255.255.0"

OK
```

说明：

- 您的查询结果可能与上述响应中的不同。

7. PC 连接到 ESP32-C2 TCP 服务器。
   PC 与 ESP32-C2 设备连接到同一个路由。
   在 PC 上使用网络调试工具，创建一个 TCP 客户端。连接到 ESP32-C2 的 TCP 服务器。地址为 `192.168.3.112`，端口为 `8080`。
   AT 响应：

```
0,CONNECT
```

8. 进入 UART Wi-Fi 透传接收模式。
   命令：

```
AT+CIPMODE=1
```

响应：

```
OK
```

9. 进入 UART Wi-Fi 透传模式 并发送数据。
   命令：

```
AT+CIPSEND
```

响应：

```
OK

>
```

10. 停止发送数据
在透传发送数据过程中，若识别到单独的一包数据 +++，则系统会退出透传发送。此时请至少等待 1 秒，再发下一条 AT 命令。请注意，如果直接用键盘打字输入 +++，有可能因时间太慢而不能被识别为连续的三个 +。更多介绍请参考*[仅适用透传模式] +++*。

重要： 使用 +++ 可退出*透传模式*，回到*透传接收模式*，此时 TCP 连接仍然有效。您也可以使用 AT+CIPSEND 命令恢复透传。

11. 退出 UART Wi-Fi *透传接收模式*。
命令：

```
AT+CIPMODE=0
```

响应：

```
OK
```

12. 关闭 TCP 连接。
命令：

```
AT+CIPCLOSE
```

响应：

```
CLOSED

OK
```

4.1.11 ESP32-C2 设备作为 softAP 在 UDP 传输中实现 UART Wi-Fi 透传

1. 设置 Wi-Fi 模式为 softAP。
命令：

```
AT+CWMODE=2
```

响应：

```
OK
```

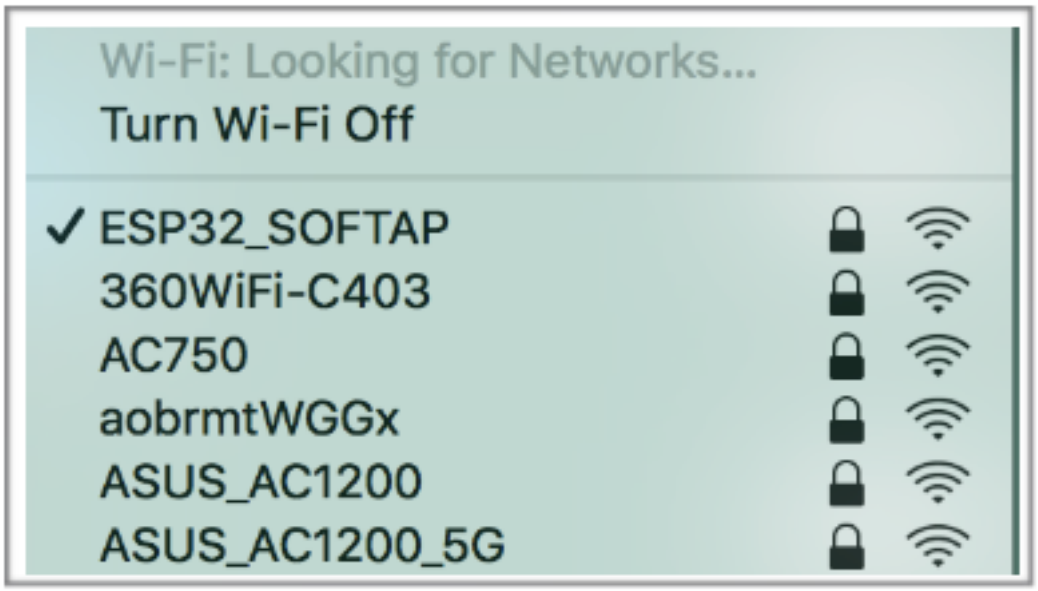
2. 设置 softAP。
命令：

```
AT+CWSAP="ESP32_softAP","1234567890",5,3
```

响应：

```
OK
```

3. PC 连接到 ESP32-C2 设备的 softAP。
4. 创建一个 UDP 端点。
在 PC 上使用网络调试助手，创建一个 UDP 传输。例如 PC 端 IP 地址为 192.168.4.2，端口为 8080。
5. ESP32-C2 与 PC 对应端口建立固定对端 IP 地址和端口的 UDP 传输。远程 IP 地址为 192.168.4.2，远端端口为 8080，本地端口为 2233，模式为 0。
命令：




```
AT+CIPSTART="UDP","192.168.4.2",8080,2233,0
```

响应：

```
CONNECT

OK
```

6. 进入 UART Wi-Fi 透传接收模式。
   命令：

```
AT+CIPMODE=1
```

   响应：

```
OK
```

7. 进入 UART Wi-Fi 透传模式 并发送数据。
   命令：

```
AT+CIPSEND
```

   响应：

```
OK

>
```

8. 停止发送数据
   在透传发送数据过程中，若识别到单独的一包数据 +++，则系统会退出透传发送。此时请至少等待 1 秒，再发下一条 AT 命令。请注意，如果直接用键盘打字输入 +++，有可能因时间太慢而不能被识别为连续的三个 +。更多介绍请参考*[仅适用透传模式] +++*。

   重要： 使用 +++ 可退出透传模式，回到透传接收模式，此时 TCP 连接仍然有效。您也可以使用 AT+CIPSEND 命令恢复透传。

9. 退出 UART Wi-Fi 透传接收模式。
   命令：

```
AT+CIPMODE=0
```

   响应：

```
OK
```

10. 关闭 TCP 连接。
    命令：

```
AT+CIPCLOSE
```

    响应：

```
CLOSED

OK
```

4.2 MQTT AT 示例

本文档主要提供在 ESP32-C2 设备上运行*MQTT AT 命令集* 命令的详细示例。

重要： 文档中所描述的例子均基于设备已连接 Wi-Fi 的前提。

4.2.1 基于 TCP 的 MQTT 连接（需要本地创建 MQTT 代理）（适用于数据量少）

以下示例同时使用两块 ESP32-C2 开发板，其中一块作为 MQTT 发布者（只作为 MQTT 发布者角色），另一块作为 MQTT 订阅者（只作为 MQTT 订阅者角色）。

示例介绍了如何基于 TCP 创建 MQTT 连接。首先您需要创建一个本地 MQTT 代理，假设 MQTT 代理的 IP 地址为 `192.168.3.102`，端口为 `8883`。

重要： 步骤中以 `ESP32-C2 MQTT 发布者`开头的操作只需要在 ESP32-C2 MQTT 发布者端执行即可，以 `ESP32-C2 MQTT 订阅者`开头的操作只需要在 ESP32-C2 MQTT 订阅者端执行即可。如果操作没有特别指明在哪端操作，则需要在发布者端和订阅者端都执行。

1. 设置 MQTT 用户属性。
   ESP32-C2 MQTT 发布者：
   命令：

```
AT+MQTTUSERCFG=0,1,"publisher","espressif","123456789",0,0,""
```

   响应：

```
OK
```

   ESP32-C2 MQTT 订阅者：
   命令：

```
AT+MQTTUSERCFG=0,1,"subscriber","espressif","123456789",0,0,""
```

   响应：

```
OK
```

2. 连接 MQTT 代理。
   命令：

```
AT+MQTTCONN=0,"192.168.3.102",8883,1
```

   响应：

```
+MQTTCONNECTED:0,1,"192.168.3.102","8883","",1

OK
```

   说明：
   - 您输入的 MQTT 代理域名或 MQTT 代理 IP 地址可能与上述命令中的不同。
3. 订阅 MQTT 主题。
   ESP32-C2 MQTT 订阅者：
   命令：

```
AT+MQTTSUB=0,"topic",1
```

   响应：

```
OK
```

4. 发布 MQTT 消息（字符串）。
   ESP32-C2 MQTT 发布者：
   命令：

```
AT+MQTTPUB=0,"topic","test",1,0
```

   响应：

```
OK
```

   说明：
   - 如果 ESP32-C2 MQTT 发布者成功发布消息，以下信息将会在 ESP32-C2 MQTT 订阅者端提示。

```
+MQTTSUBRECV:0,"topic",4,test
```

5. 关闭 MQTT 连接。
   命令：

```
AT+MQTTCLEAN=0
```

   响应：

```
OK
```

4.2.2 基于 TCP 的 MQTT 连接（需要本地创建 MQTT 代理）（适用于数据量多）

以下示例同时使用两块 ESP32-C2 开发板，其中一块作为 MQTT 发布者（只作为 MQTT 发布者角色），另一块作为 MQTT 订阅者（只作为 MQTT 订阅者角色）。

示例介绍了如何基于 TCP 创建 MQTT 连接。首先您需要创建一个本地 MQTT 代理，假设 MQTT 代理的 IP 地址为 `192.168.3.102`，端口为 `8883`。

如果您发布消息的数据量相对较多，已经超过了单条 AT 指令的长度阈值 `256`，则您可以使用*AT+MQTTPUBRAW* 命令。

重要： 步骤中以 `ESP32-C2 MQTT 发布者`开头的操作只需要在 ESP32-C2 MQTT 发布者端执行即可，以 `ESP32-C2 MQTT 订阅者`开头的操作只需要在 ESP32-C2 MQTT 订阅者端执行即可。如果操作没有特别指明在哪端操作，则需要在发布者端和订阅者端都执行。

---

1. 设置 MQTT 用户属性。
   ESP32-C2 MQTT 发布者：
   命令：

```
AT+MQTTUSERCFG=0,1,"publisher","espressif","123456789",0,0,""
```

   响应：

```
OK
```

   ESP32-C2 MQTT 订阅者：
   命令：

```
AT+MQTTUSERCFG=0,1,"subscriber","espressif","123456789",0,0,""
```

   响应：

```
OK
```

2. 连接 MQTT 代理。
   命令：

```
AT+MQTTCONN=0,"192.168.3.102",8883,1
```

   响应：

```
+MQTTCONNECTED:0,1,"192.168.3.102","8883","",1

OK
```

   说明：
   - 您输入的 MQTT 代理域名或 MQTT 代理 IP 地址可能与上述命令中的不同。

3. 订阅 MQTT 主题。
   ESP32-C2 MQTT 订阅者：
   命令：

```
AT+MQTTSUB=0,"topic",1
```

   响应：

```
OK
```

4. 发布 MQTT 消息（字符串）。
   假设你想要发布消息的数据如下，长度为 427 字节。

```
{"headers": {"Accept": "application/json","Accept-Encoding": "gzip, deflate",
↪"Accept-Language": "en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7","Content-Length": "0
↪","Host": "httpbin.org","Origin": "http://httpbin.org","Referer": "http://
↪httpbin.org/","User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.
↪36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36","X-Amzn-Trace-Id
↪": "Root=1-6150581e-1ad4bd5254b4bf5218070413"}}
```

   ESP32-C2 MQTT 发布者：
   命令：

```
AT+MQTTPUBRAW=0,"topic",427,0,0
```

   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。

```
+MQTTPUB:OK
```

说明：

- AT 输出 > 字符后，数据中的特殊字符不需要转义字符进行转义，也不需要以新行结尾（CR-LF）。
- 如果 ESP32-C2 MQTT 发布者成功发布消息，以下信息将会在 ESP32-C2 MQTT 订阅者端提示。

```
+MQTTSUBRECV:0,"topic",427,{"headers": {"Accept": "application/json",
↪"Accept-Encoding": "gzip, deflate","Accept-Language": "en-US,en;q=0.9,zh-
↪CN;q=0.8,zh;q=0.7","Content-Length": "0","Host": "httpbin.org","Origin":
↪"http://httpbin.org","Referer": "http://httpbin.org/","User-Agent":
↪"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)␣
↪Chrome/91.0.4472.114 Safari/537.36","X-Amzn-Trace-Id": "Root=1-6150581e-
↪1ad4bd5254b4bf5218070413"}}
```

5. 关闭 MQTT 连接。
   命令：

```
AT+MQTTCLEAN=0
```

响应：

```
OK
```

4.2.3 基于 TLS 的 MQTT 连接（需要本地创建 MQTT 代理）

以下示例同时使用两块 ESP32-C2 开发板，其中一块作为 MQTT 发布者（只作为 MQTT 发布者角色），另一块作为 MQTT 订阅者（只作为 MQTT 订阅者角色）。

示例介绍了如何基于 TLS 创建 MQTT 连接。首先您需要创建一个本地 MQTT 代理，假设 MQTT 代理的 IP 地址为 `192.168.3.102`，端口为 `8883`。

重要： 步骤中以 `ESP32-C2 MQTT 发布者`开头的操作只需要在 ESP32-C2 MQTT 发布者端执行即可，以 `ESP32-C2 MQTT 订阅者`开头的操作只需要在 ESP32-C2 MQTT 订阅者端执行即可。如果操作没有特别指明在哪端操作，则需要在发布者端和订阅者端都执行。

---

1. 设置时区和 SNTP 服务器。
   命令：

```
AT+CIPSNTPCFG=1,8,"ntp1.aliyun.com"
```

响应：

```
OK
```

2. 查询 SNTP 时间。
   命令：

```
AT+CIPSNTPTIME?
```

响应：

```
+CIPSNTPTIME:Thu Sep  2 18:57:03 2021
OK
```

说明：

- 您的查询 SNTP 结果可能与上述响应中的不同。
- 请确保 SNTP 时间一定是真实有效的时间，不能是 1970 年及之前的时间。
- 设置时间是为了在 TLS 认证时校验证书的有效期。

3. 设置 MQTT 用户属性。
   ESP32-C2 MQTT 发布者：
   命令：

```
AT+MQTTUSERCFG=0,4,"publisher","espressif","123456789",0,0,""
```

响应：

```
OK
```

ESP32-C2 MQTT 订阅者：
命令：

```
AT+MQTTUSERCFG=0,4,"subscriber","espressif","123456789",0,0,""
```

响应：

```
OK
```

4. 设置 MQTT 连接属性。
   命令：

```
AT+MQTTCONNCFG=0,0,0,"lwtt","lwtm",0,0
```

   响应：

```
OK
```

5. 连接 MQTT 代理。
   命令：

```
AT+MQTTCONN=0,"192.168.3.102",8883,1
```

   响应：

```
+MQTTCONNECTED:0,4,"192.168.3.102","8883","",1

OK
```

   说明：
   - 您输入的 MQTT 代理域名或 MQTT 代理 IP 地址可能与上述命令中的不同。

6. 订阅 MQTT 主题。
   ESP32-C2 MQTT 订阅者：
   命令：

```
AT+MQTTSUB=0,"topic",1
```

   响应：

```
OK
```

7. 发布 MQTT 消息（字符串）。
   ESP32-C2 MQTT 发布者：
   命令：

```
AT+MQTTPUB=0,"topic","test",1,0
```

   响应：

```
OK
```

   说明：
   - 如果 ESP32-C2 MQTT 发布者成功发布消息，以下信息将会在 ESP32-C2 MQTT 订阅者端提示。

```
+MQTTSUBRECV:0,"topic",4,test
```

8. 关闭 MQTT 连接。
   命令：

```
AT+MQTTCLEAN=0
```

   响应：

```
OK
```

4.2.4 基于 WSS 的 MQTT 连接

以下示例同时使用两块 ESP32-C2 开发板，其中一块作为 MQTT 发布者（只作为 MQTT 发布者角色），另一块作为 MQTT 订阅者（只作为 MQTT 订阅者角色）。

示例介绍了如何基于 WSS 创建 MQTT 连接。MQTT 代理域名为 `test.mosquitto.org`，端口为 `8081`。

重要： 步骤中以 `ESP32-C2 MQTT 发布者`开头的操作只需要在 ESP32-C2 MQTT 发布者端执行即可，以 `ESP32-C2 MQTT 订阅者`开头的操作只需要在 ESP32-C2 MQTT 订阅者端执行即可。如果操作没有特别指明在哪端操作，则需要在发布者端和订阅者端都执行。

---

1. 设置时区和 SNTP 服务器。
   命令：

   ```
   AT+CIPSNTPCFG=1,8,"ntp1.aliyun.com"
   ```

   响应：

   ```
   OK
   ```

2. 查询 SNTP 时间。
   命令：

   ```
   AT+CIPSNTPTIME?
   ```

   响应：

   ```
   +CIPSNTPTIME:Thu Sep  2 18:57:03 2021
   OK
   ```

   说明：
   - 您的查询 SNTP 结果可能与上述响应中的不同。
   - 请确保 SNTP 时间一定是真实有效的时间，不能是 1970 年及之前的时间。
   - 设置时间是为了在 TLS 认证时校验证书的有效期。

3. 设置 MQTT 用户属性。
   ESP32-C2 MQTT 发布者：
   命令：

   ```
   AT+MQTTUSERCFG=0,7,"publisher","espressif","1234567890",0,0,""
   ```

   响应：

   ```
   OK
   ```

   ESP32-C2 MQTT 订阅者：
   命令：

   ```
   AT+MQTTUSERCFG=0,7,"subscriber","espressif","1234567890",0,0,""
   ```

   响应：

   ```
   OK
   ```

4. 连接 MQTT 代理。
   命令：

   ```
   AT+MQTTCONN=0,"test.mosquitto.org",8081,1
   ```

   响应：

```
+MQTTCONNECTED:0,7,"test.mosquitto.org","8081","/",1

OK
```

说明：

- 您输入的 MQTT 代理域名或 MQTT 代理 IP 地址可能与上述命令中的不同。

5. 订阅 MQTT 主题。
   ESP32-C2 MQTT 订阅者：
   命令：

```
AT+MQTTSUB=0,"topic",1
```

响应：

```
OK
```

6. 发布 MQTT 消息（字符串）。
   ESP32-C2 MQTT 发布者：
   命令：

```
AT+MQTTPUB=0,"topic","test",1,0
```

响应：

```
OK
```

说明：

- 如果 ESP32-C2 MQTT 发布者成功发布消息，以下信息将会在 ESP32-C2 MQTT 订阅者端提示。

```
+MQTTSUBRECV:0,"topic",4,test
```

7. 关闭 MQTT 连接。
   命令：

```
AT+MQTTCLEAN=0
```

响应：

```
OK
```

4.3 MQTT AT 连接云示例

本文档主要介绍您的设备如何通过 AT 指令对接 AWS IoT。

重要： 有关如何使用 MQTT AT 命令的详细信息，请参阅*MQTT AT 命令集*。您需要通过阅读 AWS IoT 开发指南 来熟悉 AWS IoT。

请按照以下步骤使用 ESP-AT 将您的 ESP32-C2 设备连接到 AWS IoT。

- *从 AWS IoT 获取证书以及 endpoint*
- *使用 MQTT AT 命令基于双向认证连接 AWS IoT*

4.3.1 从 AWS IoT 获取证书以及 endpoint

1. 登录您的 AWS IoT 控制台帐号，以及切换至 IoT Core services。
2. 按照 创建 AWS IoT 资源 中的说明创建 AWS IoT 策略、事物和证书。

确保您已获得以下证书和密钥文件：

- device.pem.crt（设备证书）
- private.pem.key（私有密钥）
- Amazon-root-CA-1.pem（根 CA 证书）

3. 根据文档 设置策略 获取端点 (endpoint) 以及了解如何通过证书将事物绑定到策略。
   端点的格式为 “xxx-ats.iot.us-east-2.amazonaws.com”。

备注： 强烈建议您熟悉 AWS IoT 开发人员指南 以下是本指南中值得注意的一些要点。

- AWS IoT 需要所有设备必须有事物证书、事物私钥、和根证书。
- 有关如何激活证书。
- 区域建议选择俄亥俄州 (Ohio)。

4.3.2 使用 MQTT AT 命令基于双向认证连接 AWS IoT

替换证书

打开本地 ESP-AT 工程，并执行如下操作：

- 使用 `Amazon-root-CA-1.pem` 替换 customized_partitions/raw_data/mqtt_ca/mqtt_ca.crt。
- 使用 `device.pem.crt` 替换 customized_partitions/raw_data/mqtt_cert/mqtt_client.crt。
- 使用 `private.pem.key` 替换 customized_partitions/raw_data/mqtt_key/mqtt_client.key。

编译和烧录 AT 固件

编译 ESP-AT 项目以构建 AT 固件，并将固件烧录到您的 ESP32-C2 设备。欲了解更多信息，请参阅*编译 ESP-AT 工程*。

备注： 若不想编译 ESP-AT 工程替换证书，可直接使用 AT 命令替换固件中的证书，具体请参阅*如何生成 PKI 文件*。

使用 AT 命令连接到 AWS IoT

1. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

   响应：

```
OK
```

2. 连接 AP。
   命令：

```
AT+CWJAP=<ssid>,<password>
```

   响应：

```
OK
```

3. 设置 SNTP Server。
   命令：

```
AT+CIPSNTPCFG=1,8,"pool.ntp.org"
```

响应：

```
OK
```

4. 查询 SNTP 时间。
   命令：

```
AT+CIPSNTPTIME?
```

   响应：

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

   说明：
   - 此时获得的 <asctime style time> 必须是设置时区的实时时间，否则会因为证书有效期而导致连接失败。
5. 设置 MQTT 用户属性。
   命令：

```
AT+MQTTUSERCFG=0,5,"esp32","espressif","1234567890",0,0,""
```

   响应：

```
OK
```

   说明：
   - AT+MQTTUSERCFG 中第二参数为 5，即双向认证，不可更改。
6. 连接 AWS IoT。
   命令：

```
AT+MQTTCONN=0,"<endpoint>",8883,1
```

   响应：

```
+MQTTCONNECTED:0,5,<endpoint>,"8883","",1
OK
```

   说明：
   - 请在 *<endpoint>* 参数中填写您的 <endpoint> 值。
   - 无法更改端口 8883。
7. 订阅消息。
   命令：

```
AT+MQTTSUB=0,"topic/esp32at",1
```

   响应：

```
OK
```

8. 发布消息。
   命令：

```
AT+MQTTPUB=0,"topic/esp32at","hello aws!",1,0
```

   响应：

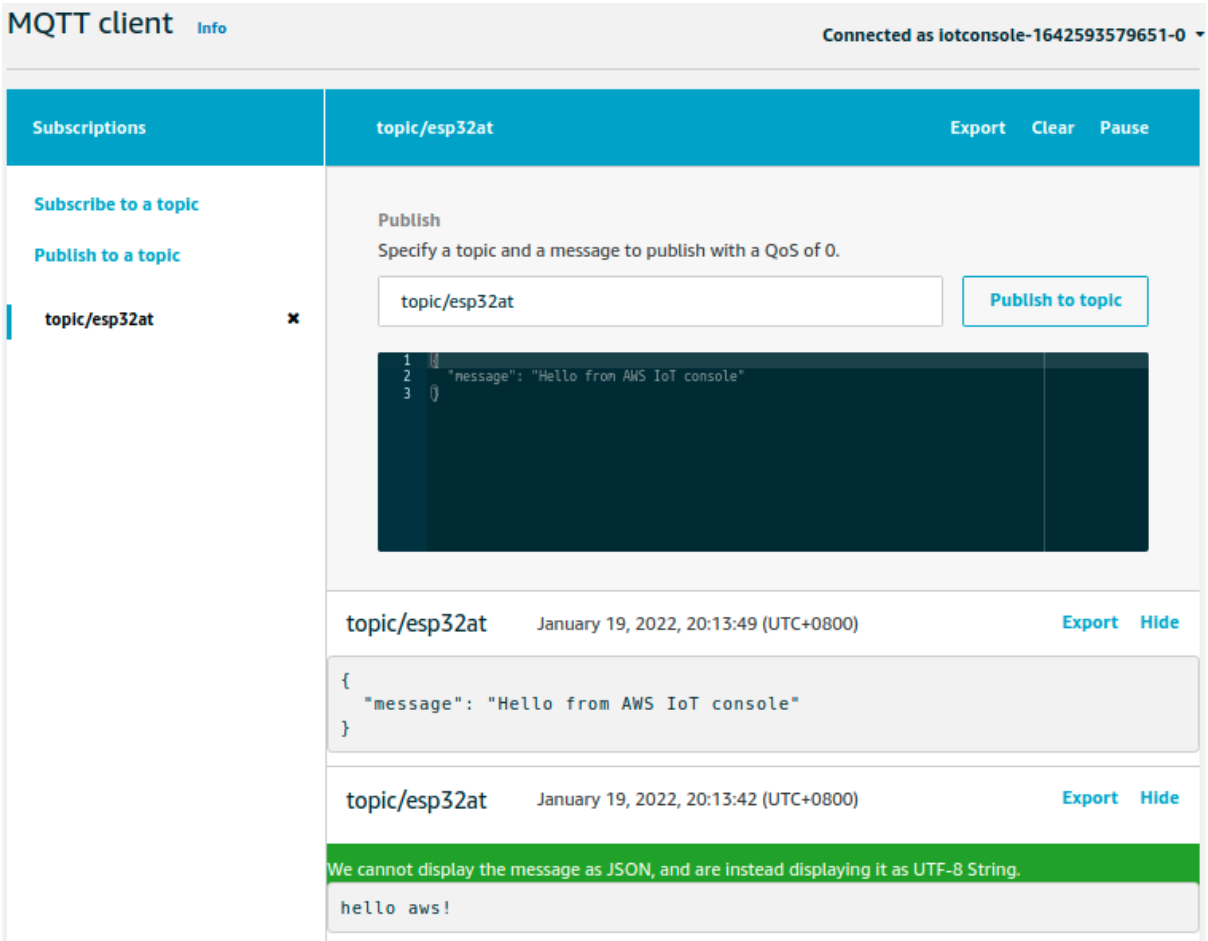
```
+MQTTSUBRECV:0,"topic/esp32at",10,hello aws!

OK
```

示例日志

正常交互日志如下：

1. ESP32 端日志

```
[20:12:43:217] AT+CWMODE=1
[20:12:43:217]
[20:12:43:217] OK
[20:12:56:684] AT+CWJAP="MERCURY_407",""
[20:12:58:234] WIFI CONNECTED
[20:12:58:937] WIFI GOT IP
[20:12:58:937]
[20:12:58:937] OK
[20:13:01:892] AT+CIPSNTPCFG=1,8,"ntp1.aliyun.com"
[20:13:01:892]
[20:13:01:892] OK
[20:13:10:854] AT+CIPSNTPTIME?
[20:13:10:854] +CIPSNTPTIME:Wed Jan 19 20:13:10 2022
[20:13:10:854] OK
[20:13:15:716] AT+MQTTUSERCFG=0,5,"esp32","espressif","1234567890",0,0,""
[20:13:15:716]
[20:13:15:716] OK
[20:13:23:030] AT+MQTTCONN=0,"a108b8vm1g634q-ats.iot.us-east-2.amazonaws.com",8883,1
[20:13:31:479] +MQTTCONNECTED:0,5,"a108b8vm1g634q-ats.iot.us-east-2.amazonaws.com","8883","",1
[20:13:31:479]
[20:13:31:479] OK
[20:13:34:275] AT+MQTTSUB=0,"topic/esp32at",1
[20:13:35:521]
[20:13:35:521] OK
[20:13:41:959] AT+MQTTPUB=0,"topic/esp32at","hello aws!",1,0
[20:13:42:422] +MQTTSUBRECV:0,"topic/esp32at",10,hello aws!
[20:13:42:422]
[20:13:42:422] OK
[20:13:49:335] +MQTTSUBRECV:0,"topic/esp32at",45,{
[20:13:49:335]   "message": "Hello from AWS IoT console"
[20:13:49:335] }
```


MQTT client
Info
Connected as iotconsole-1642593579651-0
Subscriptions
topic/esp32at
Export
Clear
Pause
Subscribe to a topic
Publish to a topic
topic/esp32at
Publish
Specify a topic and a message to publish with a QoS of 0.
topic/esp32at
Publish to topic
{
"message": "Hello from AWS IoT console"
}
topic/esp32at
January 19, 2022, 20:13:49 (UTC+0800)
Export
Hide
{
"message": "Hello from AWS IoT console"
}
topic/esp32at
January 19, 2022, 20:13:42 (UTC+0800)
Export
Hide
We cannot display the message as JSON, and are instead displaying it as UTF-8 String.
hello aws!

2. AWS 端日志

4.4 Web Server AT 示例

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：

备注： 默认的 AT 固件并不支持 AT web server 的功能，请参考 *Web 服务器 AT 命令* 启用该功能。

4.4.1 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 ESP32-C2 设备的 Wi-Fi 连接信息。您可以使用手机或电脑连接到 ESP32-C2 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 ESP32-C2 设备，然后 ESP32-C2 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 ESP32-C2 设备 首先，为了让配网设备连接 ESP32-C2 设备，ESP32-C2 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 ESP32-C2 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 ESP32-C2 设备的 AP：

图 1: 连接 ESP32-C2 AP

使用浏览器发送配网信息 在配网设备连接到 ESP32-C2 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 ESP32-C2 的 AP，则该手机不建议作为 ESP32-C2 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 ESP32-C2 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：

用户也可以点击配网页面中 SSID 输入框右方的下拉框，查看 ESP32-C2 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：

通知配网结果 配网成功后网页显示如下：

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 ESP32-C2 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：

```
+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 ESP32-C2 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。

图 2: 打开配网界面

图 3: 浏览器获取 Wi-Fi AP 列表示意图


192.168.4.1
ESP-AT
FAST_FWR310_02
配网成功, 3 秒后自动关闭网页
connect OK
开始配网

图 4: 配网成功

图 5: 配网失败

4.4.2 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，通过网页升级应用分区中的固件或者其它分区中的证书二进制固件（请参考文档*如何生成 PKI 文件* 了解更多证书信息）。

流程

- *打开 OTA 配置页面*
- *选择要更新的分区*
- *发送新版固件*
- *获取 OTA 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 ESP32-C2 模块的 AP，或者访问 OTA 配置页面的设备与 ESP32-C2 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*编译 ESP-AT 工程*) 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。

选择要更新的分区 如图，点击“Partition”下拉框来获取所有可以升级的分区：

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：

之后点击“固件升级”按钮发送新版固件。

说明 1：对于 `ota` 分区，网页会对选择的固件进行检查。固件命名的后缀必须为 `.bin`。请确保固件的大小不要超过 `partitions_at.csv` 文件中定义的 `ota` 分区大小。有关此文件的详细信息，请参考文档*如何增加一个新的模组支持*。

说明 2：对于其它分区，网页会对选择的固件进行检查。固件命名的后缀必须为 `.bin`。请确保固件的大小不要超过 `at_customize.csv` 文件中定义的分区大小。有关此文件的详细信息，请参考文档*如何自定义分区*。

获取 OTA 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
```

若接收的 OTA 固件数据失败，将提示“升级失败，请稍后重试”：

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:5       // 代表接收的 OTA 固件数据失败，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
```

说明 1：对于 `ota` 分区，需要执行*AT+RST* 重启 ESP32-C2 以应用新版固件。

说明 2：ESP32-C2 会校验接收到的 `ota` 固件内容。但不会校验接收到的其它分区固件内容，所以请确保其它分区固件内容的正确性。


ESP-AT
请选择新版本固件，然后点击升级按钮进行升级:
Partition
浏览
固件升级
当前 app 固件版本: 2.5.0
AT Core 版本: 2.5.0.0
设备配网
OTA 升级

图 6: OTA 配置页面

图 7: 获取所有可以升级的分区

图 8: 选择待发送的新版固件


ESP-AT
请选择新版本固件，然后点击升级按钮进行升级:
提示
升级成功
确定
固件升级
当前 app 固件版本: 2.5.0
AT Core 版本: 2.5.0.0
设备配网
OTA 升级

图 9: 新版固件发送成功

图 10: 新版固件发送失败

4.4.3 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 ESP32-C2 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 ESP32-C2 设备，ESP32-C2 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 *ESP32-C2* 设备参数
- 加载微信小程序
- 目标 *AP* 选择
- 执行配网

配置 ESP32-C2 设备参数 为了让小程序连接 ESP32-C2 设备，ESP32-C2 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 ESP32-C2 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
```

> **备注：** 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 ESP32-C2 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：

打开微信小程序，进入配网界面：

图 11: 获取小程序的二维码

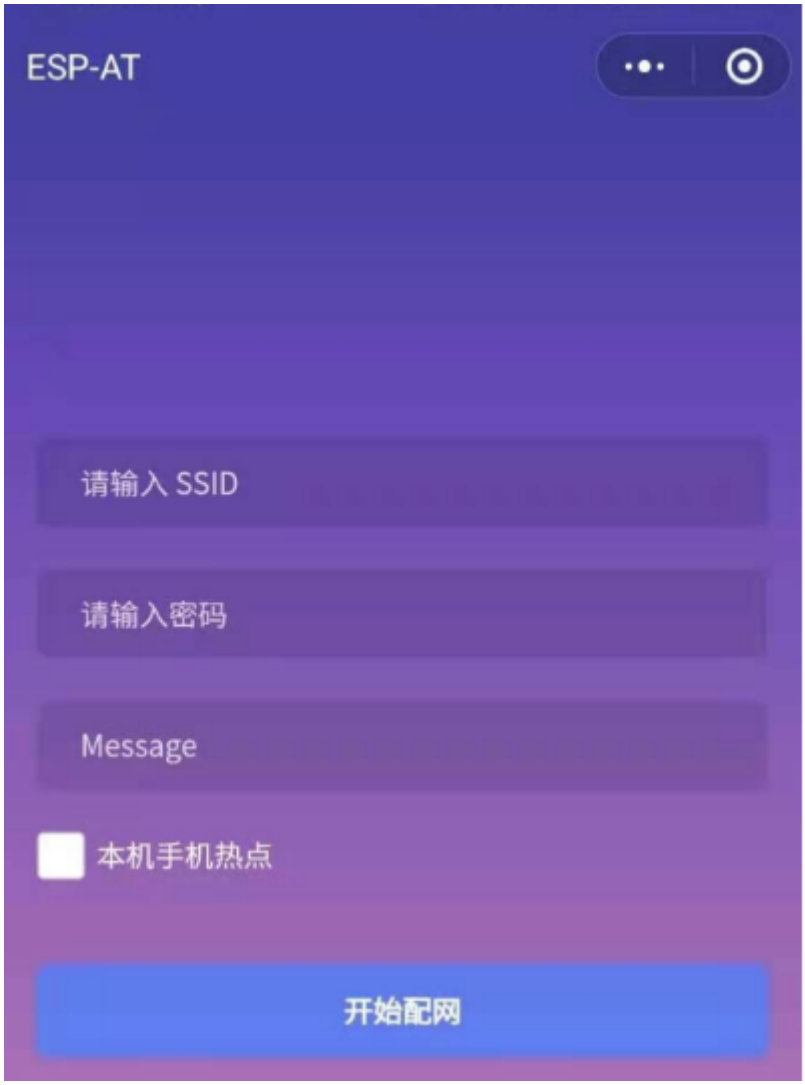


图 12: 小程序配网界面

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 13: 连接到路由器

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：

```
+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）。

备注： 要使用该功能，手机的个人热点 MAC 地址和无线局域网 MAC 地址必须确保至少前五个字节相同。

---

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”。

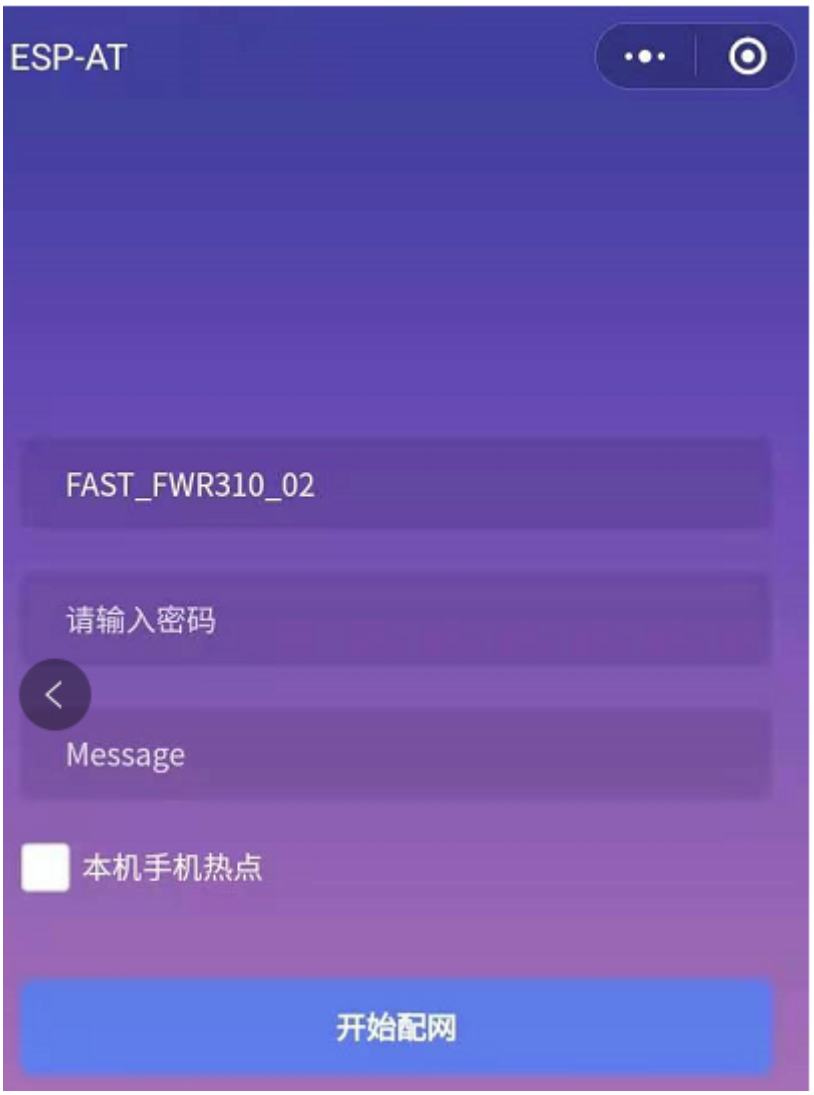


图 14: 获取路由器信息

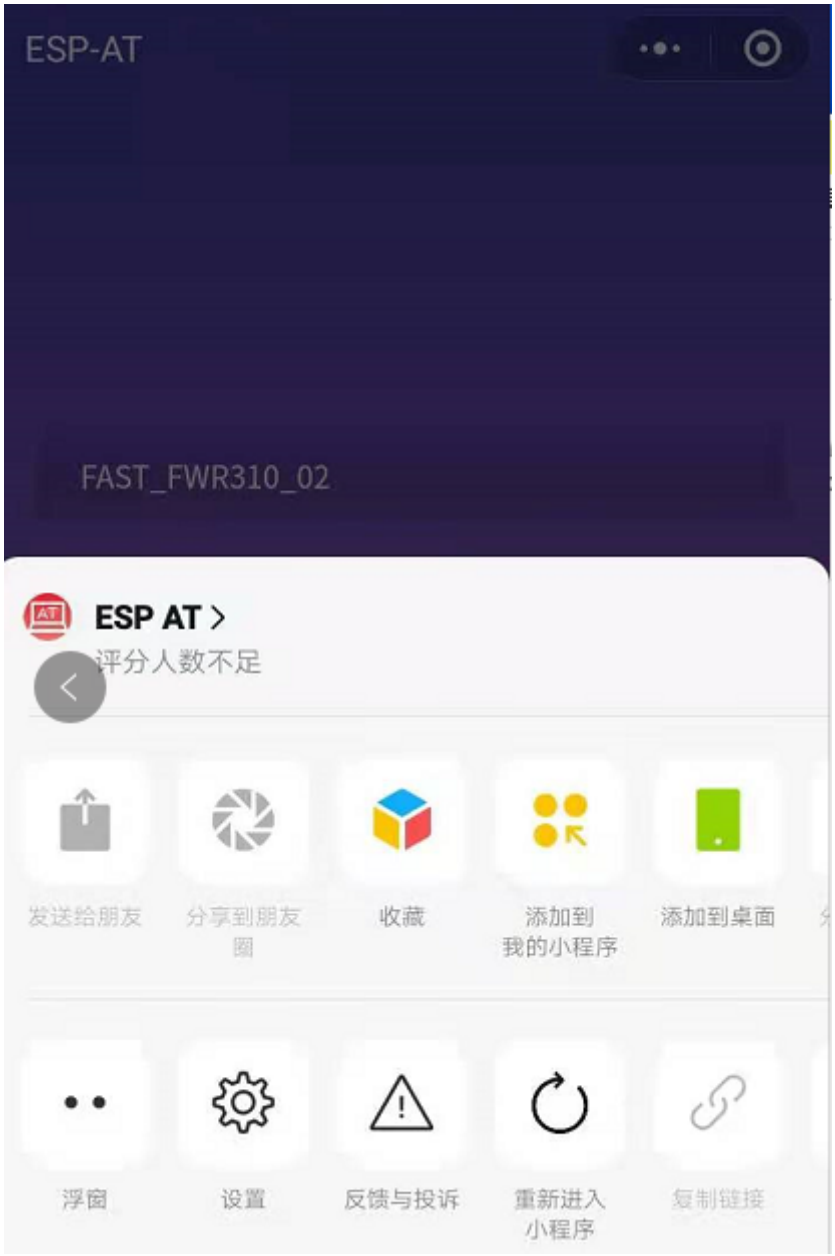

ESP-AT
FAST_FWR310_02
ESP AT >
评分人数不足
发送给朋友
分享到朋友圈
收藏
添加到我的小程序
添加到桌面
浮窗
设置
反馈与投诉
重新进入小程序
复制链接

图 15: 重新进入小程序

图 16: 通过小程序连接到路由器

图 17: 通过小程序成功连接到路由器

图 18: 通过小程序连接到路由器失败

图 19: 输入 AP 的密码

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 ESP32-C2 设备将自动扫描周围热点并发起连接。

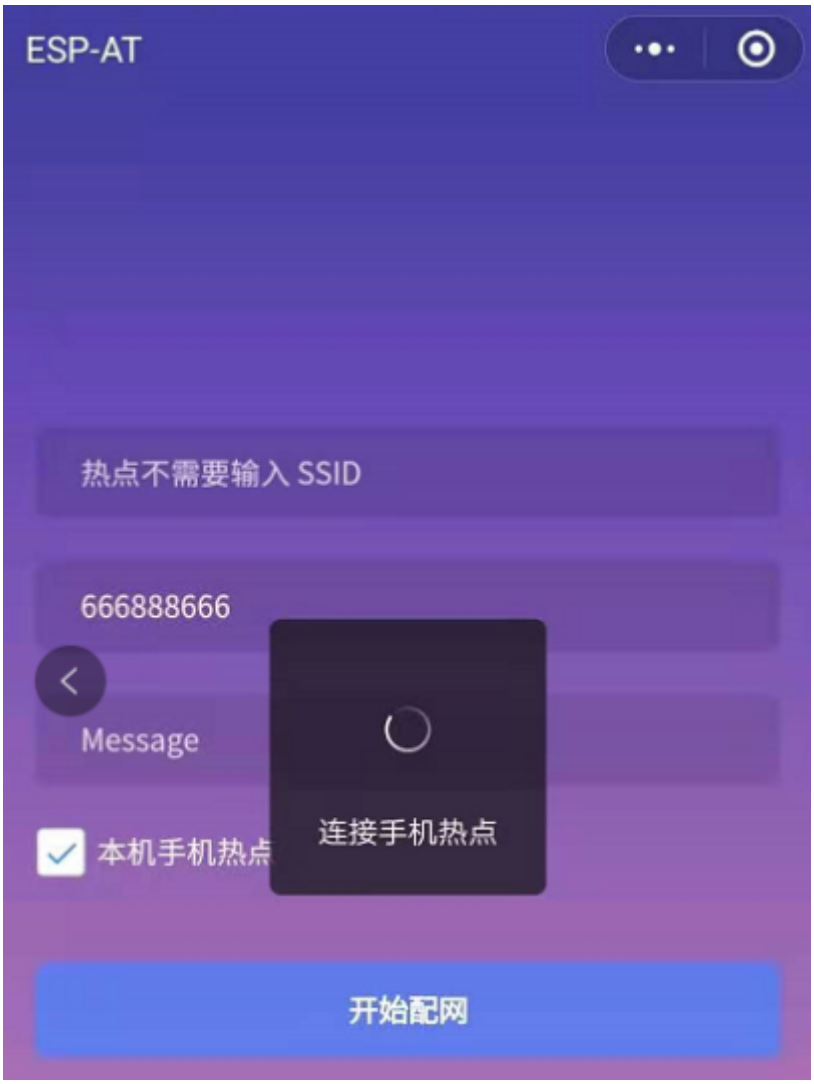


图 20: 开始连接到 AP

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 ESP32-C2 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 ESP32-C2 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 ESP32-C2 模块 AP，确认 ESP32-C2 模块的 ssid、password 已经按照上述步骤进行配置。

4.4.4 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 ESP32-C2 设备的固件升级，请参考上述*配置 ESP32-C2 设备参数* 的具体步骤完成 ESP32-C2 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。

4.4.5 ESP32-C2 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。

备注： 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（请参考编译 *ESP-AT* 工程）。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 *ESP32-C2* 设备 的具体步骤完成 ESP32-C2 模块的配置，然后连接 ESP32-C2 设备的 AP：

图 21: 连接打开 Captive Portal 功能的 AP

如上图，station 设备连接打开 Captive Portal 功能的 ESP32-C2 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。

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 ESP32-C2 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。

4.5 HTTP AT 示例

本文档主要提供在 ESP32-C2 设备上运行*HTTP AT 命令集* 命令的详细示例。

重要： 当前 ESP-AT 仅支持部分 HTTP 客户端的功能。

4.5.1 HTTP 客户端 HEAD 请求方法

该示例以 http://httpbin.org 作为 HTTP 服务器。

1. 恢复出厂设置。
   命令：

   ```
   AT+RESTORE
   ```

   响应：

   ```
   OK
   ```

2. 设置 Wi-Fi 模式为 station。
   命令：

   ```
   AT+CWMODE=1
   ```

   响应：

   ```
   OK
   ```

3. 连接路由器。
   命令：

   ```
   AT+CWJAP="espressif","1234567890"
   ```

   响应：

   ```
   WIFI CONNECTED
   WIFI GOT IP

   OK
   ```

   说明：
   - 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。

4. 发送一个 HTTP HEAD 请求。设置 `opt` 为 `1`（HEAD 方法），`url` 为 `http://httpbin.org/get`，`transport_type` 为 `1`（HTTP_TRANSPORT_OVER_TCP）。
   命令：

   ```
   AT+HTTPCLIENT=1,0,"http://httpbin.org/get",,,1
   ```

   响应：

   ```
   +HTTPCLIENT:35, Date: Sun, 26 Sep 2021 06:59:13 GMT
   +HTTPCLIENT:30, Content-Type: application/json
   +HTTPCLIENT:19, Content-Length: 329
   +HTTPCLIENT:22, Connection: keep-alive
   ```

(下页继续)

(续上页)

```
+HTTPCLIENT:23, Server: gunicorn/19.9.0
+HTTPCLIENT:30, Access-Control-Allow-Origin: *
+HTTPCLIENT:38, Access-Control-Allow-Credentials: true

OK
```

说明：

- 您获取到的 HTTP 头部信息可能与上述响应中的不同。

4.5.2 HTTP 客户端 GET 请求方法

本例以下载一个 JPG 格式的图片文件为例。图片链接为 https://www.espressif.com/sites/all/themes/espressif/images/about-us/solution-platform.jpg。

1. 恢复出厂设置。
   命令：

```
AT+RESTORE
```

   响应：

```
OK
```

2. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

   响应：

```
OK
```

3. 连接路由器。
   命令：

```
AT+CWJAP="espressif","1234567890"
```

   响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

   说明：

   - 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。

4. 发送一个 HTTP GET 请求。设置 `opt` 为 `2` （GET 方法），`url` 为 `https://www.espressif.com/sites/all/themes/espressif/images/about-us/solution-platform.jpg`，`transport_type` 为 `2` （HTTP_TRANSPORT_OVER_SSL）。
   命令：

```
AT+HTTPCLIENT=2,0,"https://www.espressif.com/sites/all/themes/espressif/images/
↪about-us/solution-platform.jpg",,,2
```

   响应：

```
+HTTPCLIENT:512,<0xff><0xd8><0xff><0xe2><0x0c>XICC_PROFILE<break>
<0x01><0x01><break>
<break>
<0x0c>HLino<0x02><0x10><break>
<break>
mntrRGB XYZ <0x07><0xce><break>
<0x02><break>
```

(下页继续)

(续上页)

```
...
+HTTPCLIENT:512,<0xeb><0xe2>v<0xcb><0x98>-<0xf8><0x8a><0xae><0xf3><0xc8><0xb6>
↪<0xa8><0x86><0x02>j<0x06><0xe2>
"<0xaa>*p<0x7f>2",h<0x12>N<0xa5><0x1e><0xd2>bp<0xea><0x1e><0xf5><0xa3>x<0xa6>J
↪<0x14>Ti<0xc3>m<0x1a>m<0x94>T<0xe1>I<0xb6><0x90><0xdc>_<0x11>QU;<0x94><0x97>
↪<0xcb><0xdd><0xc7><0xc6><0x85><0xd7>U<0x02><0xc9>W<0xa4><0xaa><0xa1><0xa1>
↪<0x08>hB<0x1a><0x10><0x86><0x84>!<0xa1><0x08>hB<0x1a><0x10><0x9b><0xb9>K
↪<0xf5>5<0x95>5-=<0x8a><0xcb><0xce><0xe0><0x91><0xf0>m<0xa9><0x04>C<0xde>k
↪<0xe7><0xc2>v<H|<0xaf><0xb8>L<0x91>=<0xda>_<0x94><0xde><0xd0><0xa9><0xc0>
↪<0xdd>8<0x9a>B<0xaa><0x1a><0x10><0x86><0x84>$<0xf4><0xd6><0xf2><0xa3><0x92>
↪<0xe7><0xa8>I<0xa3>b<0x1f>)<0xe1>z<0xc4>y<0xae><0xca><0xed><0xec><0x1e>|^
↪<0xd7>E<0xa2>_<0x13><0x9e>;{|<0xb5>Q<0x97><0xa5>P<0xdf><0xa1>#3vn<0x1b><0xc3>
↪-<0x92><0xe2>dIn<0x9c><0xb8>
<0xc7><0xa9><0xc6>(<0xe0><0xd3>i-<0x9e>@<0xbb><0xcc><0x88><0xd5>K<0xe3><0xf0>O
↪<0x9f>Km<0xb3>h<0xa8>omR<0xfe><0x8b><0xf9><0xa4><0xa6><0xff><break>
aU<0xdf><0xf3><0xa3>:A<0xe2>UG<0x04>k<0xaa>*<0xa1><0xa1><0x0b><0xca><0xec>
↪<0xd8>Q<0xfb><0xbc>yqY<0xec><0xfb>?<0x16>CM<0xf6>|}<0xae><0xf3><0x1e><0xdf>%
↪<0xf8><0xe8><0xb1>B<0x8f>[<0xb3>><0x04><0xec><0xeb>f<0x06><0x1c><0xe8><0x92>
↪<0xc9><0x8c><0xb0>I<0xd1><0x8b>%<0x99><0x04><0xd0><0xbb>s<0x8b>xj<0xe2>4f
↪<0xa0><0x8e>+E<0xda><0xab><0xc7>=<0xab><0xc7><0xb9>xz1f<0xba><0xfd>_e6<0xff>
↪<break>
(w<0xa7>b<0xe3>m<0xf0>|<0x82><0xc9><0xfb><0x8b><0xac>r<0x95><0x94><0x96><0xd9>i
↪<0xe9>RVA<0x91><0x83><0x8b>M'<0x86><0x8f><0xa3>A<0xd8><0xd8>"r"<0x8a><0xa8>
↪<0x9e>zl=<0xcd><0x16><0x07>D<0xa2><0xd0>u(<0xc2><0x8b><0x0b><0xc4><0xf1>
↪<0x87><0x9c><0x93><0x8f><0xe3><0xd5>U<0x12>]<0x8e><0x91>]<0x91><0x06>#l<0xbe>
↪<0xf4>t0?<0xd7><0x85><GEM<0xb1>%<0xee>UUT<0xe7><0xdf><0xa0><0xb9><0xce><0xe2>
↪U@<0x03><0x82>S<0xe9>*<0xa8>hB<0x1a><0x10><0xa1><0xaf>V<0x19>U<0x9d><0xb3>x
↪<0xa6><0xc7><0xe2><0x86><0x8e>d[<0x89>e<0x05>l<0x80>H<0x91>#<0xd2><0x8c>
↪<0xe1>j<0x1b>rH<0x04><0x89><0x98><0xd3>lZW]q><0xc2><';<0x93><0xb4><0xf5>&
↪<0x9d><0xa0>^Wp<0xa9>6`<0xe2>T<0xa2><0xc2><0xb1>*<0xbc><0x13><0x13><0xa0>
↪<0xc4>)<0x83><0xb6><0xbe><0x86><0xb9><0x88>-<0x1a>

OK
```

4.5.3 HTTP 客户端 POST 请求方法（适用于 POST 少量数据）

该示例以 http://httpbin.org 作为 HTTP 服务器，数据类型为 `application/json`。

1. 恢复出厂设置。
   命令：

   ```
   AT+RESTORE
   ```

   响应：

   ```
   OK
   ```

2. 设置 Wi-Fi 模式为 station。
   命令：

   ```
   AT+CWMODE=1
   ```

   响应：

   ```
   OK
   ```

3. 连接路由器。
   命令：

   ```
   AT+CWJAP="espressif","1234567890"
   ```

   响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

说明：

- 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。

4. 发送一个 HTTP POST 请求。设置 opt 为 3 （POST 方法） ，url 为 http://httpbin.org/post，content-type 为 1 （application/json） ，transport_type 为 1 (HTTP_TRANSPORT_OVER_TCP)。
   命令：

```
AT+HTTPCLIENT=3,1,"http://httpbin.org/post",,,1,"{\"form\":{\"purpose\":\"test\
↪"}}"
```

响应：

```
+HTTPCLIENT:282,{
  "args": {},
  "data": "{\"form\":{\"purpose\":\"test\"}}",
  "files": {},
  "form": {},
  "headers": {
    "Content-Length": "27",
    "Content-Type": "application/json",
    "Host": "httpbin.org",
    "User-Agent": "ESP32 HTTP Client/1.0",
    "X-Amzn-Trace-Id": "Root=
+HTTPCLIENT:173,1-61503a3f-4b16b71918855b97614c5dfb"
  },
  "json": {
    "form": {
      "purpose": "test"
    }
  },
  "origin": "20.187.154.207",
  "url": "http://httpbin.org/post"
}


OK
```

说明：

- 您获取到的结果可能与上述响应中的不同。

4.5.4 HTTP 客户端 POST 请求方法（推荐方式）

如果您 POST 的数据量相对较多，已经超过了单条 AT 指令的长度阈值 256，则建议您可以使用*AT+HTTPCPOST* 命令。

该示例以 http://httpbin.org 作为 HTTP 服务器，数据类型为 application/json。

1. 恢复出厂设置。
   命令：

```
AT+RESTORE
```

响应：

```
OK
```

2. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

响应：

```
OK
```

3. 连接路由器。
   命令：

```
AT+CWJAP="espressif","1234567890"
```

响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

说明：

- 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。

4. Post 指定长度数据。该命令设置 HTTP 头部字段数量为 2，分别是 connection 字段和 content-type 字段，connection 字段值为 keep-alive，connection 字段值为 application/json。
   假设你想要 post 的 JSON 数据如下，长度为 427 字节。

```
{"headers": {"Accept": "application/json","Accept-Encoding": "gzip, deflate",
↪"Accept-Language": "en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7","Content-Length": "0
↪","Host": "httpbin.org","Origin": "http://httpbin.org","Referer": "http://
↪httpbin.org/","User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.
↪36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36","X-Amzn-Trace-Id
↪": "Root=1-6150581e-1ad4bd5254b4bf5218070413"}}
```

命令：

```
AT+HTTPCPOST="http://httpbin.org/post",427,2,"connection: keep-alive","content-
↪type: application/json"
```

响应：

```
OK

>
```

上述响应表示 AT 已准备好接收串行数据，此时您可以输入上面提到的 427 字节长的数据，当 AT 接收到的数据长度达到 <length> 后，数据传输开始。

```
+HTTPCPOST:281,{
  "args": {},
  "data": "{\"headers\": {\"Accept\": \"application/json\",\"Accept-Encoding\
↪": \"gzip, deflate\",\"Accept-Language\": \"en-US,en;q=0.9,zh-CN;q=0.8,zh;
↪q=0.7\",\"Content-Length\": \"0\",\"Host\": \"httpbin.org\",\"Origin\": \
↪"http://httpbin.org\",\"Referer\": \"htt
+HTTPCPOST:512,p://httpbin.org/\",\"User-Agent\": \"Mozilla/5.0 (X11; Linux␣
↪x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/
↪537.36\",\"X-Amzn-Trace-Id\": \"Root=1-6150581e-1ad4bd5254b4bf5218070413\"}}
↪",
  "files": {},
  "form": {},
  "headers": {
    "Content-Length": "427",
    "Content-Type": "application/json",
    "Host": "httpbin.org",
    "User-Agent": "ESP32 HTTP Client/1.0",
    "X-Amzn-Trace-Id": "Root=1-61505e76-278b5c267aaf55842bd58b32"
  },
```

(下页继续)

(续上页)

```
  "json": {
    "headers": {

+HTTPCPOST:512,"Accept": "application/json",
      "Accept-Encoding": "gzip, deflate",
      "Accept-Language": "en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7",
      "Content-Length": "0",
      "Host": "httpbin.org",
      "Origin": "http://httpbin.org",
      "Referer": "http://httpbin.org/",
      "User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
↪ like Gecko) Chrome/91.0.4472.114 Safari/537.36",
      "X-Amzn-Trace-Id": "Root=1-6150581e-1ad4bd5254b4bf5218070413"
    }
  },
  "origin": "20.187.154
+HTTPCPOST:45,.207",
  "url": "http://httpbin.org/post"
}


SEND OK
```

说明：

- AT 输出 > 字符后，HTTP body 中的特殊字符不需要转义字符进行转义，也不需要以新行结尾（CR-LF）。

4.5.5 HTTP 客户端 PUT 请求方法（适用于无数据情况）

该示例以 http://httpbin.org 作为 HTTP 服务器。PUT 请求支持 查询字符串参数 模式。

1. 恢复出厂设置。
   命令：

```
AT+RESTORE
```

   响应：

```
OK
```

2. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

   响应：

```
OK
```

3. 连接路由器。
   命令：

```
AT+CWJAP="espressif","1234567890"
```

   响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

   说明：

   - 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。

4. 发送一个 HTTP PUT 请求。设置 opt 为 4 （PUT 方法），url 为 http://httpbin.org/put，transport_type 为 1 （HTTP_TRANSPORT_OVER_TCP）。
   命令：

```
AT+HTTPCLIENT=4,0,"http://httpbin.org/put?user=foo",,,1
```

   响应：

```
+HTTPCLIENT:282,{
  "args": {
    "user": "foo"
  },
  "data": "",
  "files": {},
  "form": {},
  "headers": {
    "Content-Length": "0",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "ESP32 HTTP Client/1.0",
    "X-Amzn-Trace-Id": "R
+HTTPCLIENT:140,oot=1-61503d41-1dd8cbe0056190f721ab1912"
  },
  "json": null,
  "origin": "20.187.154.207",
  "url": "http://httpbin.org/put?user=foo"
}


OK
```

   说明：
   - 您获取到的结果可能与上述响应中的不同。

4.5.6 HTTP 客户端 PUT 请求方法（推荐方式）

该示例以 http://httpbin.org 作为 HTTP 服务器，数据类型为 application/json。

1. 恢复出厂设置。
   命令：

```
AT+RESTORE
```

   响应：

```
OK
```

2. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

   响应：

```
OK
```

3. 连接路由器。
   命令：

```
AT+CWJAP="espressif","1234567890"
```

   响应：

```
WIFI CONNECTED
WIFI GOT IP
```

（下页继续）

(续上页)

```
OK
```

说明：

- 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。

4. PUT 指定长度数据。该命令设置 HTTP 头部字段数量为 2，分别是 connection 字段和 content-type 字段，connection 字段值为 keep-alive，connection 字段值为 application/json。
   假设你想要 put 的 JSON 数据如下，长度为 427 字节。

```
{"headers": {"Accept": "application/json","Accept-Encoding": "gzip, deflate",
↪"Accept-Language": "en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7","Content-Length": "0
↪","Host": "httpbin.org","Origin": "http://httpbin.org","Referer": "http://
↪httpbin.org/","User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.
↪36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36","X-Amzn-Trace-Id
↪": "Root=1-6150581e-1ad4bd5254b4bf5218070413"}}
```

命令：

```
AT+HTTPCPUT="http://httpbin.org/put",427,2,"connection: keep-alive","content-
↪type: application/json"
```

响应：

```
OK

>
```

上述响应表示 AT 已准备好接收串行数据，此时您可以输入上面提到的 427 字节长的数据，当 AT 接收到的数据长度达到 <length> 后，数据传输开始。

```
+HTTPCPUT:281,{
  "args": {},
  "data": "{\"headers\": {\"Accept\": \"application/json\",\"Accept-Encoding\
↪": \"gzip, deflate\",\"Accept-Language\": \"en-US,en;q=0.9,zh-CN;q=0.8,zh;
↪q=0.7\",\"Content-Length\": \"0\",\"Host\": \"httpbin.org\",\"Origin\": \
↪"http://httpbin.org\",\"Referer\": \"htt
+HTTPCPUT:512,p://httpbin.org/\",\"User-Agent\": \"Mozilla/5.0 (X11; Linux x86_
↪64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.
↪36\",\"X-Amzn-Trace-Id\": \"Root=1-6150581e-1ad4bd5254b4bf5218070413\"}}",
  "files": {},
  "form": {},
  "headers": {
    "Content-Length": "427",
    "Content-Type": "application/json",
    "Host": "httpbin.org",
    "User-Agent": "ESP32 HTTP Client/1.0",
    "X-Amzn-Trace-Id": "Root=1-635f7009-681be2d5478504dc5b83624a"
  },
  "json": {
    "headers": {

+HTTPCPUT:512,"Accept": "application/json",
      "Accept-Encoding": "gzip, deflate",
      "Accept-Language": "en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7",
      "Content-Length": "0",
      "Host": "httpbin.org",
      "Origin": "http://httpbin.org",
      "Referer": "http://httpbin.org/",
      "User-Agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
↪ like Gecko) Chrome/91.0.4472.114 Safari/537.36",
      "X-Amzn-Trace-Id": "Root=1-6150581e-1ad4bd5254b4bf5218070413"
    }
```

(下页继续)

(续上页)

```
  },
  "origin": "52.246.135
+HTTPCPUT:43,.57",
  "url": "http://httpbin.org/put"
}


SEND OK
```

说明：

- AT 输出 > 字符后，HTTP body 中的特殊字符不需要转义字符进行转义，也不需要以新行结尾（CR-LF）。

4.5.7 HTTP 客户端 DELETE 请求方法

该示例以 http://httpbin.org 作为 HTTP 服务器。DELETE 方法用于删除服务器上的资源。DELETE 请求的实现依赖服务器。

1. 恢复出厂设置。
   命令：

```
AT+RESTORE
```

   响应：

```
OK
```

2. 设置 Wi-Fi 模式为 station。
   命令：

```
AT+CWMODE=1
```

   响应：

```
OK
```

3. 连接路由器。
   命令：

```
AT+CWJAP="espressif","1234567890"
```

   响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

   说明：

   - 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。

4. 发送一个 HTTP DELETE 请求。设置 opt to 5（DELETE 方法），url 为 http://httpbin.org/delete，transport_type to 1（HTTP_TRANSPORT_OVER_TCP）。
   命令：

```
AT+HTTPCLIENT=5,0,"https://httpbin.org/delete",,,1
```

   响应：

```
+HTTPCLIENT:282,{
  "args": {},
  "data": "",
  "files": {},
  "form": {},
```

(下页继续)

(续上页)

```
  "headers": {
    "Content-Length": "0",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "ESP32 HTTP Client/1.0",
    "X-Amzn-Trace-Id": "Root=1-61504289-468a41
+HTTPCLIENT:114,737b0d251672acec9d"
  },
  "json": null,
  "origin": "20.187.154.207",
  "url": "https://httpbin.org/delete"
}


OK
```

说明：

- 您获取到的结果可能与上述响应中的不同。

4.6 Sleep AT 示例

本文档简要介绍并举例说明如何在 ESP32-C2 系列产品上使用 AT 命令设置睡眠模式。

4.6.1 简介

ESP32-C2 系列采用先进的电源管理技术，可以在不同的电源模式之间切换。当前，ESP-AT 支持以下四种功耗模式（更多休眠模式请参考技术规格书）：

1. `Active` 模式：芯片射频处于工作状态。芯片可以接收、发射和侦听信号。
2. `Modem-sleep` 模式：CPU 可运行，时钟可被配置。Wi-Fi 基带、蓝牙基带和射频关闭。
3. `Light-sleep` 模式：CPU 暂停运行。RTC 存储器和外设以及 ULP 协处理器运行。任何唤醒事件（MAC、主机、RTC 定时器或外部中断）都会唤醒芯片。
4. `Deep-sleep` 模式：CPU 和大部分外设都会掉电，只有 RTC 存储器和 RTC 外设处于工作状态。

默认情况下，ESP32-C2 会在系统复位后进入 `Active` 模式。当 CPU 不需要一直工作时，例如等待外部活动唤醒时，系统可以进入低功耗模式。

ESP32-C2 的功耗，请参考 ESP32-C2 系列芯片技术规格书。

备注：

- 将分别描述在 Wi-Fi 模式和蓝牙模式下将 ESP32-C2 设置为睡眠模式。
- 在单 Wi-Fi 模式下，只有 `station` 模式支持 `Modem-sleep` 模式和 `Light-sleep` 模式。

测量方法

为避免功耗测试过程中出现一些不必要的干扰，建议使用集成芯片的乐鑫模组进行测试。

硬件连接可参考下图。（注意，图中开发板只保留了 ESP32-C2 ，外围元器件均已移除。）

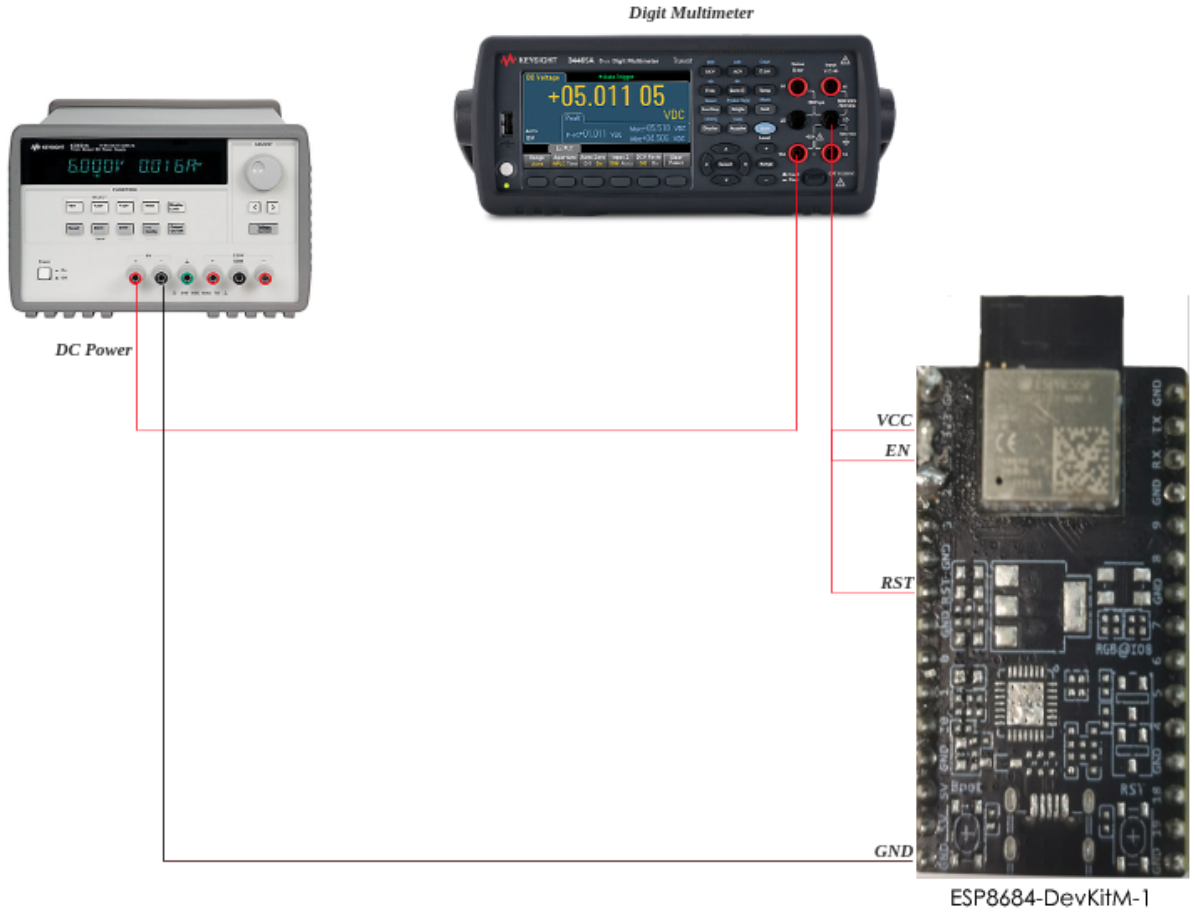


图 22: ESP32-C2 硬件连接

4.6.2 在 Wi-Fi 模式下设置为 Modem-sleep 模式

1. 设置 Wi-Fi 为 station 模式。
   命令：

```
AT+CWMODE=1
```

   响应：

```
OK
```

2. 连接路由器。
   命令：

```
AT+CWJAP="espressif","1234567890"
```

   响应：

```
WIFI CONNECTED
WIFI GOT IP

OK
```

   说明：
   - 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。
3. 设置休眠模式为 Modem-sleep 模式。

命令：

```
AT+SLEEP=1
```

响应：

```
OK
```

备注：

- RF 将根据 AP 的 DTIM 定期关闭（路由器一般设置 DTIM 为 1）。

4.6.3 在 Wi-Fi 模式下设置为 Light-sleep 模式

1. 设置 Wi-Fi 为 station 模式。
   命令：

   ```
   AT+CWMODE=1
   ```

   响应：

   ```
   OK
   ```

2. 连接路由器。设置监听间隔为 3。
   命令：

   ```
   AT+CWJAP="espressif","1234567890",,,,3
   ```

   响应：

   ```
   WIFI CONNECTED
   WIFI GOT IP

   OK
   ```

   说明：
   - 您输入的 SSID 和密码可能跟上述命令中的不同。请使用您的路由器的 SSID 和密码。
3. 设置休眠模式为 Light-sleep 模式。
   命令：

   ```
   AT+SLEEP=2
   ```

   响应：

   ```
   OK
   ```

备注：

- CPU 将会自动休眠，RF 则会根据*AT+CWJAP* 设置的监听间隔定期关闭。

4.6.4 设置为 Deep-sleep 模式

1. 设置休眠模式为 Deep-sleep 模式。设置 deep-sleep 时间为 3600000 ms。
   命令：

   ```
   AT+GSLP=3600000
   ```

   响应：

```
OK
```

说明：

- 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。
- 对于 Deep-sleep 模式，唯一的唤醒方法是定时唤醒。

Chapter 5

如何编译和开发自己的 AT 工程

5.1 编译 ESP-AT 工程

本文档详细介绍了如何编译 ESP-AT 工程，并将生成的固件烧录到 ESP32-C2 设备中。当默认的官方发布的固件 无法满足需求时，如您需要自定义*AT 端口管脚* 以及分区 等，那么就需要编译 ESP-AT 工程。

5.1.1 详细步骤

请根据下方详细步骤，完成 ESP-AT 工程的克隆、环境安装、配置、编译以及烧录。**推荐您优先选择 Linux 系统开发**。

- 第一步：*ESP-IDF* 快速入门
- 第二步：获取 *ESP-AT*
- 第三步：安装环境
- 第四步：连接设备
- 第五步：配置工程
- 第六步：编译工程
- 第七步：烧录到设备
- *build.py* 进阶用法

5.1.2 第一步：ESP-IDF 快速入门

在编译 ESP-AT 工程之前，请先学习使用 ESP-IDF，因为 ESP-AT 是基于 ESP-IDF 开发的。

请您根据 ESP-IDF v5.0 快速入门文档 的指导，完成 `hello_world` 工程的配置、编译以及下载固件至 ESP32-C2 开发板等步骤。

备注： 此步骤不是必须的，但如果您是初学者，强烈建议您完成此步骤，以熟悉 ESP-IDF 并确保顺利进行以下步骤。

完成上一步的 ESP-IDF 快速入门后，便可以开始下面的 ESP-AT 工程的编译。

5.1.3 第二步：获取 ESP-AT

编译 ESP-AT 工程需下载乐鑫提供的软件库文件 ESP-AT 仓库。

打开终端，切换到您要保存 ESP-AT 的工作目录，使用 `git clone` 命令克隆远程仓库，获取 ESP-AT 的本地副本。以下是不同操作系统的获取方式。

- Linux 或 macOS

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

- Windows
  对于 ESP32-C2 系列模组，推荐您以管理员权限运行 ESP-IDF 5.0 CMD。

```
cd %userprofile%\esp
git clone --recursive https://github.com/espressif/esp-at.git
```

如果您位于中国或访问 GitHub 有困难，也可以使用 `git clone https://gitee.com/EspressifSystems/esp-at.git` 来获取 ESP-AT，可能会更快。

ESP-AT 将下载至 Linux 和 macOS 的 `~/esp/esp-at`、Windows 的 `%userprofile%\esp\esp-at`。

备注： 在本文档中，Linux 和 macOS 操作系统中 ESP-AT 的默认安装路径为 `~/esp`；Windows 操作系统的默认路径为 `%userprofile%\esp`。您也可以将 ESP-AT 安装在任何其它路径下，但请注意在使用命令行时进行相应替换。注意，ESP-AT 不支持带有空格的路径。

5.1.4 第三步：安装环境

运行项目工具 `install` 来安装环境。此安装工具将自动安装依赖的 Python 包、ESP-IDF 仓库以及 ESP-IDF 依赖的编译器、工具等。

- Linux 或 macOS

```
./build.py install
```

- Windows

```
python build.py install
```

如果是第一次安装环境，请为 ESP32-C2 设备选择以下配置选项。

- 选择 `Platform name`，例如 ESP32-C2 系列设备选择 `PLATFORM_ESP32C2`。`Platform name` 由 factory_param_data.csv 定义。
- 选择 `Module name`，例如 ESP8684-MINI-1 4MB 模组选择 `ESP32C2-4MB`。`Module name` 由 factory_param_data.csv 定义。
- 启用或禁用 `silence mode`，启用时将删除一些日志并减少固件的大小。一般情况下请禁用。
- 如果 `build/module_info.json` 文件存在，上述三个配置选项将不会出现。因此，如果您想重新配置模组信息，请删除该文件。

5.1.5 第四步：连接设备

使用 USB 线将您的 ESP32-C2 设备连接到 PC 上，以下载固件和输出日志，详情请见*硬件连接*。注意，如果您在编译过程中不发送 AT 命令和接收 AT 响应，则不需要建立“AT 命令/响应”连接。关于更改默认端口管脚的信息请参考*如何设置 AT 端口管脚*。

5.1.6 第五步：配置工程

运行项目工具 `menuconfig` 来配置。

- Linux 或 macOS

```
./build.py menuconfig
```

- Windows

```
python build.py menuconfig
```

如果以上所有步骤都正确，则会弹出下面的菜单：

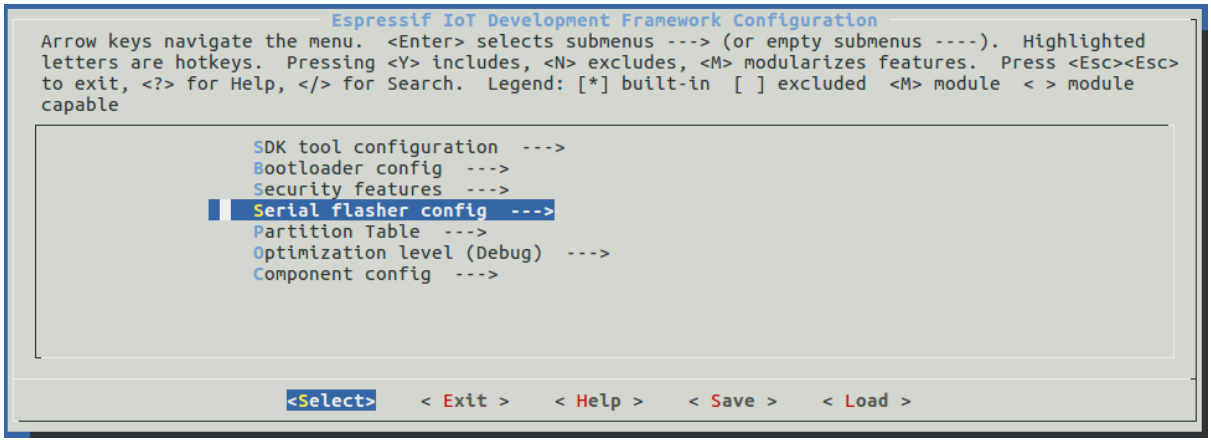


图 1: 工程配置 - 主窗口

此菜单可以用来配置每个工程，如更改 AT 端口管脚、启用经典蓝牙功能等，如果不修改配置，那么就会按照默认配置编译工程。

5.1.7 第六步：编译工程

运行以下命令编译工程。

- Linux 或 macOS

```
./build.py build
```

- Windows

```
python build.py build
```

如果启用了蓝牙功能，固件尺寸会大大增加。请确保它不超过 ota 分区的大小。

编译完成后会在 `build/factory` 路径下生成打包好的量产固件。更多信息请参见*ESP-AT 固件差异*。

5.1.8 第七步：烧录到设备

运行以下命令将生成的固件烧录到 ESP32-C2 设备上。

- Linux 或 macOS

```
./build.py -p (PORT) flash
```

- Windows

```
python build.py -p (PORT) flash
```

注意请用 ESP32-C2 设备的串口名称替换 `(PORT)`。或者按照提示信息将固件烧录到 flash 中。仍然需要注意替换 `(PORT)`。

如果 ESP-AT bin 不能启动，并且打印出 “ota data partition invalid” ，请运行 `python build.py erase_flash` 来擦除整个 flash，然后重新烧录 AT 固件。

5.1.9 build.py 进阶用法

`build.py` 脚本是基于 idf.py 封装的工具（即 `idf.py <cmd>` 功能均包含在 `build.py <cmd>` 里），您可以运行以下命令查看更多用法。

- Linux 或 macOS

```
./build.py --help
```

- Windows

```
python build.py --help
```

5.2 如何设置 AT 端口管脚

本文档介绍了如何修改 ESP32-C2 系列固件中的*AT port* 管脚。默认情况下，ESP-AT 使用两个 UART 接口作为 AT 端口：一个用于输出日志（以下称为日志端口），另一个用于发送 AT 命令和接收响应（以下称为命令端口）。

要修改 ESP32-C2 设备的 AT 端口管脚，应该：

- *克隆 ESP-AT 工程*。
- 在 menuconfig 配置工具或 factory_param_data.csv 表格中修改对应管脚。
- 编译工程，在 `build/customized_partitions/factory_param.bin` 生成新的 bin 文件。
- *将新的 bin 文件烧录进设备*。

本文档重点介绍如何修改管脚，点击上面的链接了解其它步骤的详细信息。

备注： 使用其它接口作为 AT 命令接口请参考 使用 AT SPI 接口 , AT through SPI 和 使用 AT 套接字接口 。

5.2.1 ESP32-C2 系列

ESP32-C2 AT 固件的日志端口和命令端口管脚可以自定义为其它管脚，请参阅 《ESP32-C2 技术参考手册》 查看可使用的管脚。

修改日志端口管脚

默认情况下，乐鑫提供的 ESP32-C2 AT 固件使用以下 UART0 管脚输出日志：

- TX：GPIO21
- RX：GPIO20

在编译 ESP-AT 工程时，可使用 menuconfig 配置工具将其修改为其它管脚：

- `./build.py menuconfig -> Component config -> Common ESP-related -> UART for console output`
- `./build.py menuconfig -> Component config -> Common ESP-related -> UART TX on GPIO#`
- `./build.py menuconfig -> Component config -> Common ESP-related -> UART RX on GPIO#`

修改命令端口管脚

默认情况下，UART1 用于发送 AT 命令和接收 AT 响应，其管脚定义在 factory_param_data.csv 表格中的 uart_port、uart_tx_pin、uart_rx_pin、uart_cts_pin 和 uart_rts_pin 列。

您可以直接在 factory_param_data.csv 表中修改端口管脚：

- 打开您本地的 factory_param_data.csv。
- 找到模组所在的行。
- 根据需要设置 `uart_port`。
- 根据需要设置 `uart_tx_pin` 和 `uart_rx_pin`。
- 若不需要使用硬件流控功能，请将 `uart_cts_pin` 和 `uart_rts_pin` 设置为 -1。
- 保存表格。

5.3 添加自定义 AT 命令

本文档详细介绍了如何在 ESP-AT 工程中添加用户定义的 AT 命令，以 `AT+TEST` 命令为例展示每个步骤的示例代码。

自定义一个基本的、功能良好的命令至少需要以下两个步骤。

- *定义 AT 命令*
- *注册 AT 命令*

这一步介绍新命令的运行及响应情况。

- *尝试一下吧*

其余的步骤适用于定义相对复杂的命令，可根据您的需要进行选择。

- *定义返回消息*
- *获取命令参数*
- *省略命令参数*
- *阻塞命令的执行*
- *从 AT 命令端口获取输入的数据*

AT 命令集的源代码不开源，以 库文件 的形式呈现，它也是解析自定义的 AT 命令的基础。

5.3.1 定义 AT 命令

在定义 AT 命令之前，请先决定 AT 命令的名称和类型。

命令命名规则：

- 命令应以 + 符号开头。
- 支持字母 (`A~Z`, `a~z`)、数字 (`0~9`) 及其它一些字符 (`!`、`%`、`-`、`.`、`/`、`:`、`_`)，详情请见*AT 命令分类*。

命令类型：

每条 AT 命令最多可以有四种类型：测试命令、查询命令、设置命令和执行命令，更多信息参见*AT 命令分类*。

然后，定义所需类型的命令。假设 `AT+TEST` 支持所有的四种类型，下面是定义每种类型的示例代码。

测试命令：

```
uint8_t at_test_cmd_test(uint8_t *cmd_name)
{
    uint8_t buffer[64] = {0};
```

(下页继续)

(续上页)

```
    snprintf((char *)buffer, 64, "this cmd is test cmd: %s\r\n", cmd_name);

    esp_at_port_write_data(buffer, strlen((char *)buffer));

    return ESP_AT_RESULT_CODE_OK;
}
```

查询命令：

```
uint8_t at_query_cmd_test(uint8_t *cmd_name)
{
    uint8_t buffer[64] = {0};

    snprintf((char *)buffer, 64, "this cmd is query cmd: %s\r\n", cmd_name);

    esp_at_port_write_data(buffer, strlen((char *)buffer));

    return ESP_AT_RESULT_CODE_OK;
}
```

设置命令：

```
uint8_t at_setup_cmd_test(uint8_t para_num)
{
    int32_t para_int_1 = 0;
    uint8_t *para_str_2 = NULL;
    uint8_t num_index = 0;
    uint8_t buffer[64] = {0};

    if (esp_at_get_para_as_digit(num_index++, &para_int_1) != ESP_AT_PARA_PARSE_
↪RESULT_OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    }

    if (esp_at_get_para_as_str(num_index++, &para_str_2) != ESP_AT_PARA_PARSE_
↪RESULT_OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    }

    snprintf((char *)buffer, 64, "this cmd is setup cmd and cmd num is: %u\r\n",␣
↪para_num);
    esp_at_port_write_data(buffer, strlen((char *)buffer));

    memset(buffer, 0, 64);
    snprintf((char *)buffer, 64, "first parameter is: %d\r\n", para_int_1);
    esp_at_port_write_data(buffer, strlen((char *)buffer));

    memset(buffer, 0, 64);
    snprintf((char *)buffer, 64, "second parameter is: %s\r\n", para_str_2);
    esp_at_port_write_data(buffer, strlen((char *)buffer));

    return ESP_AT_RESULT_CODE_OK;
}
```

执行命令：

```
uint8_t at_exe_cmd_test(uint8_t *cmd_name)
{
    uint8_t buffer[64] = {0};

    snprintf((char *)buffer, 64, "this cmd is execute cmd: %s\r\n", cmd_name);
```

(下页继续)

(续上页)

```
    esp_at_port_write_data(buffer, strlen((char *)buffer));

    return ESP_AT_RESULT_CODE_OK;
}
```

最后，调用*esp_at_cmd_struct* 来定义 AT 命令的名称和支持的类型，下面的示例代码定义了名称为 +TEST`` （省略了 ``AT）并支持所有四种类型的命令。

```
static esp_at_cmd_struct at_custom_cmd[] = {
    {"+TEST", at_test_cmd_test, at_query_cmd_test, at_setup_cmd_test, at_exe_cmd_
↪test},
};
```

如果不定义某个类型，将其设置为 NULL。

5.3.2 注册 AT 命令

调用 API *esp_at_custom_cmd_array_regist()* 来注册 AT 命令，以下是注册 AT+TEST 的示例代码。

```
esp_at_custom_cmd_array_regist(at_custom_cmd, sizeof(at_custom_cmd) / sizeof(at_
↪custom_cmd[0]));
```

备注： 建议把 esp_at_custom_cmd_array_regist 加入 app_main() 中的 at_custom_init()。

5.3.3 尝试一下吧

如果你已经完成了上述两个步骤，请编译 ESP-AT 工程并烧录固件，该命令即可在您的设备上正常运行。尝试运行一下吧！

下面是 AT+TEST 的运行情况。

测试命令：

```
AT+TEST=?
```

响应：

```
AT+TEST=?
this cmd is test cmd: +TEST

OK
```

查询命令：

```
AT+TEST?
```

响应：

```
AT+TEST?
this cmd is query cmd: +TEST

OK
```

设置命令：

```
AT+TEST=1,"espressif"
```

响应：

```
AT+TEST=1,"espressif"
this cmd is setup cmd and cmd num is: 2
first parameter is: 1
second parameter is: espressif

OK
```

执行命令：

```
AT+TEST
```

响应：

```
AT+TEST
this cmd is execute cmd: +TEST

OK
```

5.3.4 定义返回消息

ESP-AT 已经在*esp_at_result_code_string_index* 定义了一些返回消息，更多返回消息请参见*AT 消息*。

除了通过 return 模式返回消息，也可调用 API *esp_at_response_result()* 来返回命令执行结果。可在代码中同时使用 API 和*ESP_AT_RESULT_CODE_SEND_OK* 及*ESP_AT_RESULT_CODE_SEND_FAIL*。

例如，当使用 AT+TEST 的执行命令向服务器或 MCU 发送数据时，用*esp_at_response_result()* 来返回发送结果，用 return 模式来返回命令执行结果，示例代码如下。

```
uint8_t at_exe_cmd_test(uint8_t *cmd_name)
{
    uint8_t buffer[64] = {0};

    snprintf((char *)buffer, 64, "this cmd is execute cmd: %s\r\n", cmd_name);

    esp_at_port_write_data(buffer, strlen((char *)buffer));

    // 向服务器或 MCU 发送数据的自定义操作
    send_data_to_server();

    // 返回 SEND OK
    esp_at_response_result(ESP_AT_RESULT_CODE_SEND_OK);

    return ESP_AT_RESULT_CODE_OK;
}
```

运行命令及返回的响应：

```
AT+TEST
this cmd is execute cmd: +TEST

SEND OK

OK
```

5.3.5 获取命令参数

ESP-AT 提供以下两个 API 获取命令参数。

- *esp_at_get_para_as_digit()* 可获取数字参数。
- *esp_at_get_para_as_str()* 可获取字符串参数。

示例请见执行命令。

5.3.6 省略命令参数

本节介绍如何设置某些命令参数为可选参数。

- 省略首位或中间参数
- 省略最后一位参数

省略首位或中间参数

假设您想将 AT+TEST 命令的 <param_2> 和 <param_3> 参数设置为可选参数，其中 <param_2> 为数字参数，<param_3> 为字符串参数。

```
AT+TEST=<param_1>[,<param_2>][,<param_3>],<param_4>
```

实现代码如下。

```
uint8_t at_setup_cmd_test(uint8_t para_num)
{
    int32_t para_int_1 = 0;
    int32_t para_int_2 = 0;
    uint8_t *para_str_3 = NULL;
    uint8_t *para_str_4 = NULL;
    uint8_t num_index = 0;
    uint8_t buffer[64] = {0};
    esp_at_para_parse_result_type parse_result = ESP_AT_PARA_PARSE_RESULT_OK;

    snprintf((char *)buffer, 64, "this cmd is setup cmd and cmd num is: %u\r\n",␣
↪para_num);
    esp_at_port_write_data(buffer, strlen((char *)buffer));

    parse_result = esp_at_get_para_as_digit(num_index++, &para_int_1);
    if (parse_result != ESP_AT_PARA_PARSE_RESULT_OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    } else {
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "first parameter is: %d\r\n", para_int_1);
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }

    parse_result = esp_at_get_para_as_digit(num_index++, &para_int_2);
    if (parse_result != ESP_AT_PARA_PARSE_RESULT_OMITTED) {
        if (parse_result != ESP_AT_PARA_PARSE_RESULT_OK) {
            return ESP_AT_RESULT_CODE_ERROR;
        } else {
            // 示例代码
            // 需要自定义操作
            memset(buffer, 0, 64);
            snprintf((char *)buffer, 64, "second parameter is: %d\r\n", para_int_
↪2);
            esp_at_port_write_data(buffer, strlen((char *)buffer));
        }
```

(下页继续)

(续上页)

```
    } else {
        // 示例代码
        // 省略第二个参数
        // 需要自定义操作
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "second parameter is omitted\r\n");
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }

    parse_result = esp_at_get_para_as_str(num_index++, &para_str_3);
    if (parse_result != ESP_AT_PARA_PARSE_RESULT_OMITTED) {
        if (parse_result != ESP_AT_PARA_PARSE_RESULT_OK) {
            return ESP_AT_RESULT_CODE_ERROR;
        } else {
            // 示例代码
            // 需自定义操作
            memset(buffer, 0, 64);
            snprintf((char *)buffer, 64, "third parameter is: %s\r\n", para_str_3);
            esp_at_port_write_data(buffer, strlen((char *)buffer));
        }
    } else {
        // 示例代码
        // 省略第三个参数
        // 需自定义操作
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "third parameter is omitted\r\n");
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }

    parse_result = esp_at_get_para_as_str(num_index++, &para_str_4);
    if (parse_result != ESP_AT_PARA_PARSE_RESULT_OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    } else {
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "fourth parameter is: %s\r\n", para_str_4);
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }

    return ESP_AT_RESULT_CODE_OK;
}
```

备注： 如果输入的字符串参数为 ""，则该参数没有被省略。

省略最后一位参数

假设 AT+TEST 命令的 <param_3> 参数为字符串参数，且为最后一位参数，您想将它设置为可选参数。

```
AT+TEST=<param_1>,<param_2>[,<param_3>]
```

则有以下两种省略情况。

- AT+TEST=<param_1>,<param_2>
- AT+TEST=<param_1>,<param_2>,

实现代码如下。

```
uint8_t at_setup_cmd_test(uint8_t para_num)
{
```

(下页继续)

(续上页)

```
    int32_t para_int_1 = 0;
    uint8_t *para_str_2 = NULL;
    uint8_t *para_str_3 = NULL;
    uint8_t num_index = 0;
    uint8_t buffer[64] = {0};
    esp_at_para_parse_result_type parse_result = ESP_AT_PARA_PARSE_RESULT_OK;

    snprintf((char *)buffer, 64, "this cmd is setup cmd and cmd num is: %u\r\n",␣
↪para_num);
    esp_at_port_write_data(buffer, strlen((char *)buffer));

    parse_result = esp_at_get_para_as_digit(num_index++, &para_int_1);
    if (parse_result != ESP_AT_PARA_PARSE_RESULT_OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    } else {
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "first parameter is: %d\r\n", para_int_1);
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }

    parse_result = esp_at_get_para_as_str(num_index++, &para_str_2);
    if (parse_result != ESP_AT_PARA_PARSE_RESULT_OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    } else {
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "second parameter is: %s\r\n", para_str_2);
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }

    if (num_index == para_num) {
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "third parameter is omitted\r\n");
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    } else {
        parse_result = esp_at_get_para_as_str(num_index++, &para_str_3);
        if (parse_result != ESP_AT_PARA_PARSE_RESULT_OMITTED) {
            if (parse_result != ESP_AT_PARA_PARSE_RESULT_OK) {
                return ESP_AT_RESULT_CODE_ERROR;
            } else {
                // 示例代码
                // 需自定义操作
                memset(buffer, 0, 64);
                snprintf((char *)buffer, 64, "third parameter is: %s\r\n", para_
↪str_3);
                esp_at_port_write_data(buffer, strlen((char *)buffer));
            }
        } else {
            // 示例代码
            // 省略第三个参数
            // 需自定义操作
            memset(buffer, 0, 64);
            snprintf((char *)buffer, 64, "third parameter is omitted\r\n");
            esp_at_port_write_data(buffer, strlen((char *)buffer));
        }
    }

    return ESP_AT_RESULT_CODE_OK;
}
```

备注： 如果输入的字符串参数为 ""，则该参数没有被省略。

5.3.7 阻塞命令的执行

有时您想阻塞某个命令的执行，等待另一个执行结果，然后系统基于这个结果可能会返回不同的值。

一般来说，这类命令需要与其它任务的结果进行同步。

推荐使用 semaphore 来同步。

示例代码如下。

```
xSemaphoreHandle at_operation_sema = NULL;

uint8_t at_exe_cmd_test(uint8_t *cmd_name)
{
    uint8_t buffer[64] = {0};

    snprintf((char *)buffer, 64, "this cmd is execute cmd: %s\r\n", cmd_name);

    esp_at_port_write_data(buffer, strlen((char *)buffer));

    // 示例代码
    // 不必在此处创建 semaphores
    at_operation_sema = xSemaphoreCreateBinary();
    assert(at_operation_sema != NULL);

    // 阻塞命令的执行
    // 等待另一个执行的结果
    // 其它任务可调用 xSemaphoreGive 来释放 semaphore
    xSemaphoreTake(at_operation_sema, portMAX_DELAY);

    return ESP_AT_RESULT_CODE_OK;
}
```

5.3.8 从 AT 命令端口获取输入的数据

ESP-AT 支持从 AT 命令端口访问输入的数据，为此提供以下两个 API。

- *esp_at_port_enter_specific()* 设置回调函数，AT 端口接收到输入的数据后，将调用该函数。
- *esp_at_port_exit_specific()* 删除由 esp_at_port_enter_specific 设置的回调函数。

获取数据的方法会根据数据长度是否被指定而有所不同。

指定长度的输入数据

假设您已经使用 <param_1> 指定了数据长度，如下所示。

```
AT+TEST=<param_1>
```

以下示例代码介绍如何从 AT 命令端口获取长度为 <param_1> 的输入数据。

```
static xSemaphoreHandle at_sync_sema = NULL;

void wait_data_callback(void)
{
    xSemaphoreGive(at_sync_sema);
}

uint8_t at_setup_cmd_test(uint8_t para_num)
{
    int32_t specified_len = 0;
```

(下页继续)

(续上页)

```
    int32_t received_len = 0;
    int32_t remain_len = 0;
    uint8_t *buf = NULL;
    uint8_t buffer[64] = {0};

    if (esp_at_get_para_as_digit(0, &specified_len) != ESP_AT_PARA_PARSE_RESULT_
↪OK) {
        return ESP_AT_RESULT_CODE_ERROR;
    }

    buf = (uint8_t *)malloc(specified_len);
    if (buf == NULL) {
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "malloc failed\r\n");
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }

    // 示例代码
    // 不必在此处创建 semaphores
    if (!at_sync_sema) {
        at_sync_sema = xSemaphoreCreateBinary();
        assert(at_sync_sema != NULL);
    }

    // 返回输入数据提示符 ">"
    esp_at_port_write_data((uint8_t *)">", strlen(">"));

    // 设置回调函数，在接收到输入数据后由 AT 端口调用
    esp_at_port_enter_specific(wait_data_callback);

    // 接收输入的数据
    while(xSemaphoreTake(at_sync_sema, portMAX_DELAY)) {
        received_len += esp_at_port_read_data(buf + received_len, specified_len -␣
↪received_len);

        if (specified_len == received_len) {
            esp_at_port_exit_specific();

            // 获取剩余输入数据的长度
            remain_len = esp_at_port_get_data_length();
            if (remain_len > 0) {
                // 示例代码
                // 如果剩余数据长度 > 0，则实际输入数据长度大于指定的接收数据长度
                // 可自定义如何处理这些剩余数据
                // 此处只是简单打印出剩余数据
                esp_at_port_recv_data_notify(remain_len, portMAX_DELAY);
            }

            // 示例代码
            // 输出接收到的数据
            memset(buffer, 0, 64);
            snprintf((char *)buffer, 64, "\r\nreceived data is: ");
            esp_at_port_write_data(buffer, strlen((char *)buffer));

            esp_at_port_write_data(buf, specified_len);

            break;
        }
    }

    free(buf);
```

(下页继续)

(续上页)

```
    return ESP_AT_RESULT_CODE_OK;
}
```

因此，如果您设置 AT+TEST=5，输入的数据为 1234567890，那么 ESP-AT 返回的结果如下所示。

```
AT+TEST=5
>67890
received data is: 12345
OK
```

未指定长度的输入数据

这种情况类似 Wi-Fi 透传模式，不指定数据长度。

```
AT+TEST
```

假设 ESP-AT 结束命令的执行并返回执行结果，示例代码如下。

```
#define BUFFER_LEN (2048)
static xSemaphoreHandle at_sync_sema = NULL;

void wait_data_callback(void)
{
    xSemaphoreGive(at_sync_sema);
}

uint8_t at_exe_cmd_test(uint8_t *cmd_name)
{
    int32_t received_len = 0;
    int32_t remain_len = 0;
    uint8_t *buf = NULL;
    uint8_t buffer[64] = {0};


    buf = (uint8_t *)malloc(BUFFER_LEN);
    if (buf == NULL) {
        memset(buffer, 0, 64);
        snprintf((char *)buffer, 64, "malloc failed\r\n");
        esp_at_port_write_data(buffer, strlen((char *)buffer));
    }

    // 示例代码
    // 不必在此处创建 semaphores
    if (!at_sync_sema) {
        at_sync_sema = xSemaphoreCreateBinary();
        assert(at_sync_sema != NULL);
    }

    // 返回输入数据提示符 ">"
    esp_at_port_write_data((uint8_t *)">", strlen(">"));

    // 设置回调函数，在接收到输入数据后由 AT 端口调用
    esp_at_port_enter_specific(wait_data_callback);

    // 接收输入的数据
    while(xSemaphoreTake(at_sync_sema, portMAX_DELAY)) {
        memset(buf, 0, BUFFER_LEN);
```

(下页继续)

(续上页)

```
        received_len = esp_at_port_read_data(buf, BUFFER_LEN);
        // 检查是否退出该模式
        // 退出条件是接收到 "+++" 字符串
        if ((received_len == 3) && (strncmp((const char *)buf, "+++", 3)) == 0) {
            esp_at_port_exit_specific();

            // 示例代码
            // 如果剩余数据长度 > 0，说明缓冲区内仍有数据需要处理
            // 可自定义如何处理剩余数据
            // 此处只是简单打印出剩余数据
            remain_len = esp_at_port_get_data_length();
            if (remain_len > 0) {
                esp_at_port_recv_data_notify(remain_len, portMAX_DELAY);
            }

            break;
        } else if (received_len > 0) {
            // 示例代码
            // 可自定义如何处理接收到的数据
            // 此处只是简单打印出接收到的数据
            memset(buffer, 0, 64);
            snprintf((char *)buffer, 64, "\r\nreceived data is: ");
            esp_at_port_write_data(buffer, strlen((char *)buffer));

            esp_at_port_write_data(buf, strlen((char *)buf));
        }
    }

    free(buf);

    return ESP_AT_RESULT_CODE_OK;
}
```

因此，如果第一个输入数据是 1234567890，第二个输入数据是 +++，那么 ESP-AT 返回结果如下所示。

```
AT+TEST
>
received data is: 1234567890
OK
```

5.4 如何提高 ESP-AT 吞吐性能

默认情况下，ESP-AT 和主机之间使用 UART 进行通信，因此最高吞吐速率不会超过其默认配置，即不会超过 115200 bps。用户如要 ESP-AT 实现较高的吞吐量，需了解本文，并做出有针对性的配置。本文以 ESP32-C2 为例，介绍如何提高 ESP-AT 吞吐性能。

备注： 本文基于 ESP-AT 处于透传模式 下，描述提高 TCP/UDP/SSL 吞吐性能的一般性方法。

用户可以选择下面其中一种方法，提高吞吐性能：

- *[简单] 快速配置*
- *[推荐] 熟悉数据流、针对性地配置*

5.4.1 [简单] 快速配置

1. 配置系统、LWIP、Wi-Fi 适用于高吞吐的参数

将下面代码段拷贝并追加至 module_config/module_esp32c2_default/sdkconfig.defaults 文件最后，其它 ESP32-C2 系列设备请修改对应文件夹下的 sdkconfig.defaults 文件。

```
# System
CONFIG_ESP_SYSTEM_EVENT_TASK_STACK_SIZE=4096
CONFIG_FREERTOS_UNICORE=n
CONFIG_FREERTOS_HZ=1000
CONFIG_ESP_DEFAULT_CPU_FREQ_MHZ_120=y
CONFIG_ESP_DEFAULT_CPU_FREQ_MHZ=120
CONFIG_ESPTOOLPY_FLASHMODE_QIO=y
CONFIG_ESPTOOLPY_FLASHFREQ_60M=y

# LWIP
CONFIG_LWIP_TCP_SND_BUF_DEFAULT=65534
CONFIG_LWIP_TCP_WND_DEFAULT=65534
CONFIG_LWIP_TCP_RECVMBOX_SIZE=12
CONFIG_LWIP_UDP_RECVMBOX_SIZE=12
CONFIG_LWIP_TCPIP_RECVMBOX_SIZE=64

# Wi-Fi
CONFIG_ESP32_WIFI_STATIC_RX_BUFFER_NUM=16
CONFIG_ESP32_WIFI_DYNAMIC_RX_BUFFER_NUM=64
CONFIG_ESP32_WIFI_DYNAMIC_TX_BUFFER_NUM=64
CONFIG_ESP32_WIFI_TX_BA_WIN=32
CONFIG_ESP32_WIFI_RX_BA_WIN=32
CONFIG_ESP32_WIFI_AMPDU_TX_ENABLED=y
CONFIG_ESP32_WIFI_AMPDU_RX_ENABLED=y
```

2. 提高 UART 缓冲区大小

将下面代码段拷贝并替换 at_uart_task.c 文件中 *uart_driver_install()* 行。

```
uart_driver_install(esp_at_uart_port, 1024 * 16, 1024 * 16, 100, &esp_at_
↪uart_queue, 0);
```

3. 删除默认配置、重新编译固件、烧录运行

```
rm -rf build sdkconfig
./build.py build
./build.py flash monitor
```

4. 透传前提高 UART 波特率

典型 AT 命令序列如下：

```
AT+CWMODE=1
AT+CWJAP="ssid","password"
AT+UART_CUR=3000000,8,1,0,3
AT+CIPSTART="TCP","192.168.105.13",3344
AT+CIPSEND
// 传输数据
```

此快速配置的方法在一定程度上可以提高吞吐，但有时可能不能达到用户的预期。另外有些配置可能不是吞吐的瓶颈，配置较高可能会牺牲内存资源或功耗等。因此，用户也可以熟悉下面推荐的方法，进行有针对性地配置。

5.4.2 [推荐] 熟悉数据流、针对性地配置

影响 ESP-AT 吞吐的因素大体描述如下图：

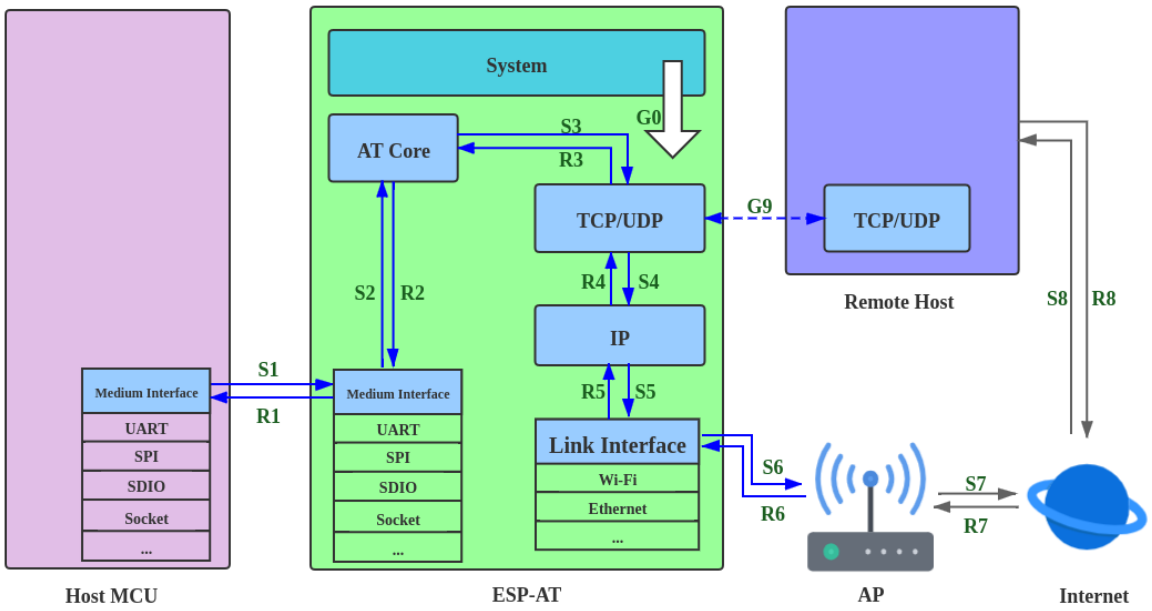


图 2: 吞吐数据流

如图中箭头所示：

- ESP-AT 发送 (TX) 的数据流为 S1 -> S2 -> S3 -> S4 -> S5 -> S6 -> S7 -> S8
- ESP-AT 接收 (RX) 的数据流为 R8 -> R7 -> R6 -> R5 -> R4 -> R3 -> R2 -> R1

吞吐的数据流类似于水流，要想提高吞吐，需要考虑在数据流速较低的节点之间进行优化，而不需要在数据流速本就达到预期的节点之间进行额外的配置，以免造成不必要的资源浪费。在实际产品中，往往只需要提高其中一条数据流吞吐即可，用户需要根据下面指导进行对应的配置即可。

备注： 下面的配置均以可用内存充足为前提的，用户可以通过*AT+SYSRAM* 命令来查询可用内存。

1. G0 吞吐优化

G0 是系统可以优化的部分，建议参考配置如下：

```
CONFIG_ESP_SYSTEM_EVENT_TASK_STACK_SIZE=4096
CONFIG_FREERTOS_UNICORE=n
CONFIG_FREERTOS_HZ=1000
CONFIG_ESP_DEFAULT_CPU_FREQ_MHZ_120=y
CONFIG_ESP_DEFAULT_CPU_FREQ_MHZ=120
CONFIG_ESPTOOLPY_FLASHMODE_QIO=y
CONFIG_ESPTOOLPY_FLASHFREQ_60M=y
```

2. S1、R1 吞吐优化

通常情况下，S1 和 R1 是 ESP-AT 吞吐高低的关键。因为默认情况下，ESP-AT 和主机之间使用 UART 进行通信，波特率为 115200，而 UART 硬件上，速率上限为 5 Mbps。因此，用户使用场景吞吐低于 5 Mbps，可以使用默认的 UART 作为和主机之间的通信介质，同时可以进行下面优化。

2.1 提高 UART 缓冲区大小

将下面代码段拷贝并替换 at_uart_task.c 文件中 *uart_driver_install()* 行。

- 提高 UART TX 吞吐

```
uart_driver_install(esp_at_uart_port, 1024 * 16, 8192, 100, &esp_at_
↪uart_queue, 0);
```

- 提高 UART RX 吞吐

```
uart_driver_install(esp_at_uart_port, 2048, 1024 * 16, 100, &esp_at_
↪uart_queue, 0);
```

- 提高 UART TX 和 RX 吞吐

```
uart_driver_install(esp_at_uart_port, 1024 * 16, 1024 * 16, 100, &esp_
↪at_uart_queue, 0);
```

2.2 透传前提高 UART 波特率

典型 AT 命令序列如下：

```
AT+CWMODE=1
AT+CWJAP="ssid","password"
AT+UART_CUR=3000000,8,1,0,3
AT+CIPSTART="TCP","192.168.105.13",3344
AT+CIPSEND
// 传输数据
```

备注： 用户需要确保主机的 UART 可以支持到这么高的速率，并且主机和 ESP-AT 之间的 UART 连线尽可能地短。

备注： 如果用户期望吞吐速率大于或接近于 5 Mbps，可以考虑使用 SPI、SDIO、Socket 等方式。具体请参考：

- Socket：Socket AT 指南

3. S2、R2、R3、S3 吞吐优化

通常情况下，S2、R2、R3、S3 不是 ESP-AT 吞吐高低的瓶颈。因为 AT core 在 UART 缓冲区和通信协议的传输层之间传递数据，仅有极少的且不耗时的应用逻辑，无需优化。

4. S4、R4、S5、R5、S6、R6 吞吐优化

ESP-AT 和主机之间使用 UART 进行通信，S4、R4、S5、R5、S6、R6 无需优化。ESP-AT 和主机之间使用其他传输介质进行通信时，S4、R4、S5、R5、S6、R6 是影响吞吐的一个因素。

S4、R4、S5、R5、S6、R6 是通信协议的传输层、网络层、和数据链路层之间的数据流。用户需要阅读 ESP-IDF 中 如何提高 Wi-Fi 性能 文档，了解原理，进行合理配置。这些配置均可以在 `./build.py menuconfig` 里进行配置。

- 优化 S4 -> S5 -> S6：发送数据方向配置
- 优化 R6 -> R5 -> R4：接收数据方向配置

5. S6、R6 吞吐优化

S6 和 R6 是通信协议的数据链路层，ESP32-C2 可以使用 Wi-Fi 或者以太网作为传输介质。Wi-Fi 除了上述介绍的优化方法之外，可能还需要用户关心：

- 提高 RF 发射功率
  默认发射功率通常不是吞吐高低的瓶颈，用户也可以通过*AT+RFPOWER* 命令查询和设置 RF 发射功率。
- 设置 802.11 b/g/n 协议
  默认 Wi-Fi 模式即为 802.11 b/g/n 协议，用户可通过*AT+CWSTAPROTO* 命令查询和设置 802.11 b/g/n 协议。配置是双向的，因此建议 AP 端 Wi-Fi 模式配置为 802.11 b/g/n 协议，频宽配置为 HT20/HT40 (20/40 MHz) 模式。

6. S7、R7、S8、R8 吞吐优化

通常情况下，S7、R7、S8、R8 不是 ESP-AT 吞吐优化的范围。因为这和实际网络带宽、网络路由、物理距离等有关。

5.5 如何生成出厂参数二进制文件

本文介绍了如何为模组生成一个自定义的 ESP-AT 出厂参数二进制文件 (factory_MODULE_NAME.bin)。例如，您可能想在 ESP-AT 固件中自定义国家代码、射频限制或 UART 管脚，则可以通过下面的两个表格定义此类参数。

- *factory_param_type.csv*
- *factory_param_data.csv*

下面介绍如何通过修改这两个表格来生成自定义的出厂参数二进制文件。

- 新增一个自定义模组
- 新增一个自定义参数
- 修改现有模组的出厂参数数据

5.5.1 factory_param_type.csv

factory_param_type.csv 表列出了您可以定义的所有参数，以及每个参数的偏移量、类型和大小，它储存在 customized_partitions/raw_data/factory_param/factory_param_type.csv。

下表提供了每个参数的信息。

param_name	说明
description	在编译 ESP-AT 工程时提示的模组附加信息。
magic_flag	该值必须是 `0xfcfc`。
version	供内部使用的出厂参数管理版本，当前版本为 3，不建议修改。
reserved1	保留。
tx_max_power	ESP32-C2 的 Wi-Fi 最大发射功率：[40,84]，详情请见 ESP32-C2 发射功率 设置范围。
uart_port	用于发送 AT 命令和接收 AT 响应的 UART 端口。
start_channel	起始 Wi-Fi 信道。
channel_num	Wi-Fi 的总信道数。
country_code	国家代码。
uart_baudrate	UART 的波特率。
uart_tx_pin	UART tx 管脚。
uart_rx_pin	UART rx 管脚。
uart_cts_pin	UART cts 管脚，不使用时请置为 -1。
uart_rts_pin	UART rts 管脚，不使用时请置为 -1。
tx_control_pin	某些开发板上电时，该 tx 管脚需要与 MCU 分开。不使用时请置为 -1。
rx_control_pin	某些开发板上电时，该 rx 管脚需要与 MCU 分开。不使用时请置为 -1。
reserved2	保留。
platform	运行当前固件的平台（也称为芯片系列）。
module_name	运行当前固件的模组。

5.5.2 factory_param_data.csv

factory_param_data.csv 表格保存了 *factory_param_type.csv* 中定义的所有参数的值，支持 ESP32-C2 系列的模组。您可通过修改表中的值来自定义出厂参数二进制文件。该表储存在 customized_partitions/raw_data/factory_param/factory_param_data.csv 中。

5.5.3 新增一个自定义模组

本节通过一个示例介绍如何在 factory_param_data.csv 中添加一个自定义模组，并为其生成出厂参数二进制文件。假设您想为一个名为 MY_MODULE 的 ESP32-C2 模组生成出厂参数二进制文件，其国家代码为 JP，Wi-Fi 信道为 1 至 14，其它参数与 PLATFORM_ESP32C2 的 ESP32C2-4MB 模组相同，可按照以下步骤进行操作。

修改 factory_param_data.csv

在 factory_param_data.csv 表中设置 MY_MODULE 的所有参数值。

首先，在表格底部插入一行，然后输入以下参数值。

- param_name: value
- platform: PLATFORM_ESP32C2
- module_name: MY_MODULE
- description: MY_DESCRIPTION
- magic_flag: 0xfcfc
- version: 3
- reserved1: 0
- tx_max_power: 78
- uart_port: 1
- start_channel: 1
- channel_num: 14
- country_code: JP
- uart_baudrate: 115200
- uart_tx_pin: 17
- uart_rx_pin: 16
- uart_cts_pin: 15
- uart_rts_pin: 14
- tx_control_pin: -1
- rx_control_pin: -1

修改后的 factory_param_data.csv 表格如下所示。

```
platform,module_name,description,magic_flag,version,reserved1,tx_max_power,uart_
↪port,start_channel,channel_num,country_code,uart_baudrate,uart_tx_pin,uart_rx_
↪pin,uart_cts_pin,uart_rts_pin,tx_control_pin,rx_control_pin
PLATFORM_ESP32,WROOM-32,,0xfcfc,3,0,78,1,1,13,CN,115200,17,16,15,14,-1,-1
...
PLATFORM_ESP32C2,MY_MODULE,MY_DESCRIPTION,0xfcfc,3,0,78,1,1,14,JP,115200,17,16,15,
↪14,-1,-1
```

修改 esp_at_module_info 结构体

在 at/src/at_default_config.c 中的 esp_at_module_info 结构体中添加自定义模组的信息。

esp_at_module_info 结构体提供 OTA 升级验证 token：

```
typedef struct {
    char* module_name;
    char* ota_token;
    char* ota_ssl_token;
} esp_at_module_info_t;
```

若不想使用 OTA 功能，那么第二个参数 ota_token 和第三个参数 ota_ssl_token 应该设置为 NULL，第一个参数 module_name 必须与 factory_param_data.csv 文件中的 module_name 一致。

下面是修改后的 esp_at_module_info 结构体。

```
static const esp_at_module_info_t esp_at_module_info[] = {
#if defined(CONFIG_IDF_TARGET_ESP32)
  ...
#endif

#if defined(CONFIG_IDF_TARGET_ESP32C3)
  ...
#endif

#if defined(CONFIG_IDF_TARGET_ESP32C2)
  ...
#endif

#if defined(CONFIG_IDF_TARGET_ESP32C2)
  {"MY_MODULE",        CONFIG_ESP_AT_OTA_TOKEN_MY_MODULE,       CONFIG_ESP_AT_OTA_
↪SSL_TOKEN_MY_MODULE },      // MY_MODULE
#endif
};
```

宏 CONFIG_ESP_AT_OTA_TOKEN_MY_MODULE 和宏 CONFIG_ESP_AT_OTA_SSL_TOKEN_MY_MODULE 定义在头文件 at/private_include/at_ota_token.h 中。

```
#if defined(CONFIG_IDF_TARGET_ESP32C2)
...
#define CONFIG_ESP_AT_OTA_TOKEN_MY_MODULE        CONFIG_ESP_AT_OTA_TOKEN_DEFAULT

...
#define CONFIG_ESP_AT_OTA_SSL_TOKEN_MY_MODULE        CONFIG_ESP_AT_OTA_SSL_TOKEN_
↪DEFAULT
```

重新编译工程并选择自定义模组

在添加自定义模组信息后，根据*编译 ESP-AT 工程* 重新编译整个工程，在配置工程时选择自定义模组。

```
Platform name:
1. PLATFORM_ESP32
2. PLATFORM_ESP32C3
3. PLATFORM_ESP32C2
choose(range[1,3]):2

Module name:
...
x. MY_MODULE (description: MY_DESCRIPTION)
choose(range[1,x]):x
```

编译完成后可在 esp-at/build/customized_partitions 文件夹下找到生成的出厂参数二进制文件。

5.5.4 新增一个自定义参数

本节通过一个示例介绍如何新增一个自定义参数。假设您想为 MY_MODULE 添加参数 date，并将其设置为 20210603，可按照以下步骤进行操作。

修改 factory_param_type.csv

在 factory_param_type.csv 中定义参数 `date`。

首先，在表格的底部插入一行，然后设置参数的名称 (param_name)、偏移量 (offset)、类型 (type) 和大小 (size)。

param_name	offset	type	size
description	-1	String	0
…	…	…	…
date	88	String	9

修改 factory_param_data.csv

在 factory_param_data.csv 最后一列的后面插入一列，并命名为 `date`，然后将 `MY_MODULE` 对应的值设置为 `20210603`。

以下是修改后的 CSV 表格。

```
platform,module_name,description,magic_flag,version,reserved1,tx_max_power,uart_
↪port,start_channel,channel_num,country_code,uart_baudrate,uart_tx_pin,uart_rx_
↪pin,uart_cts_pin,uart_rts_pin,tx_control_pin,rx_control_pin,date
PLATFORM_ESP32,WROOM-32,,0xfcfc,3,0,78,1,1,13,CN,115200,17,16,15,14,-1,-1
...
PLATFORM_ESP32C2,MY_MODULE,MY_DESCRIPTION,0xfcfc,3,0,78,1,1,14,JP,115200,17,16,15,
↪14,-1,-1,20210603
```

处理自定义参数

您可以自定义函数来处理自定义的参数 `date`，以下只是简单输出参数值。

```
static void esp_at_factory_parameter_date_init(void)
{
    const esp_partition_t * partition = esp_at_custom_partition_find(0x40, 0xff,
↪"factory_param");
    char* data = NULL;
    char* str_date = NULL;

    if (!partition) {
        printf("factory_parameter partition missed\r\n");
        return;
    }

    data = (char*)malloc(ESP_AT_FACTORY_PARAMETER_SIZE); // 说明
    assert(data != NULL);
    if(esp_partition_read(partition, 0, data, ESP_AT_FACTORY_PARAMETER_SIZE) !=␣
↪ESP_OK){
        free(data);
        return;
    }
```

(下页继续)

(续上页)

```
    if ((data[0] != 0xFC) || (data[1] != 0xFC)) { // 检查 magic flag 是否为 0xfc
↪0xfc
        return;
    }

    // 示例代码
    // 可自定义如何处理参数 date
    // 此处仅简单打印 date 参数值
    str_date = &data[88];   // date 字段偏移地址
    printf("date is %s\r\n", str_date);

    free(data);

    return;
}
```

重新编译工程

参考编译 *ESP-AT* 工程 来编译整个工程。

编译完成后可在 esp-at/build/customized_partitions 文件夹下找到生成的出厂参数二进制文件。

5.5.5 修改现有模组的出厂参数数据

假设您需要修改 factory_param_data.csv 中现有的某个模组的出厂参数数据，可采用下面任意一种方法。

重新编译整个工程

打开 factory_param_data.csv 并根据需要修改参数。

重新编译 ESP-AT 工程（参考编译 *ESP-AT* 工程），出厂参数二进制文件会在 esp-at/build/customized_partitions 文件夹生成。

只编译出厂参数二进制文件

首先，克隆整个 ESP-AT 工程。

然后，前往 ESP-AT 工程的根目录，输入以下命令，并替换一些参数。

```
python tools/factory_param_generate.py --platform PLATFORM --module MODULE --
↪define_file DEFINE_FILE --module_file MODULE_FILE --bin_name BIN_NAME --log_file
↪LOG_FILE
```

- PLATFORM 替换为模组的平台，必须与 factory_param_data.csv 中 platform 的值一致。
- MODULE 替换为模组的名称，必须与 factory_param_data.csv 中 module_name 的值一致。
- DEFINE_FILE 替换为 factory_param_type.csv 的相对路径。
- MODULE_FILE 替换为 factory_param_data.csv 的相对路径。
- BIN_NAME 替换为出厂参数二进制文件名。
- LOG_FILE 储存模组名称的文件名。

以下为 MY_MODULE 的示例代码。

```
python tools/factory_param_generate.py --platform PLATFORM_ESP32C2 --module MY_
↪MODULE --define_file components/customized_partitions/raw_data/factory_param/
↪factory_param_type.csv --module_file components/customized_partitions/raw_data/
↪factory_param/factory_param_data.csv --bin_name ./factory_param.bin --log_file ./
↪factory_parameter.log
```

执行上述命令后，将在当前目录下生成以下三个文件。

- factory_param.bin
- factory_parameter.log
- factory_param_MY_MODULE.bin

将新生成的 factory_param_MY_MODULE.bin 下载到 flash 中，可使用 ESP-AT 提供的 esptool.py 进行下载，在 ESP-AT 项目的根目录下执行以下命令，并替换一些参数。

```
python esp-idf/components/esptool_py/esptool/esptool.py -p PORT -b BAUD --before␣
↪default_reset --after hard_reset --chip auto  write_flash --flash_mode dio --
↪flash_size detect --flash_freq 40m ADDRESS FILEDIRECTORY
```

- PORT 替换为端口名称。
- BAUD 替换为波特率。
- ADDRESS 替换为 flash 中开始的地址。ESP-AT 对 ADDRESS 参数有严格的要求，不同固件的出厂参数二进制文件的地址不同，请参考下面的表格。

表 1: 出厂参数二进制文件下载地址

平台	固件	地址
PLATFORM_ESP32C2	ESP32C2-4MB 固件	0x2B000

- FILEDIRECTORY 替换为出厂参数二进制文件的相对路径。

下面是将生成的出厂参数二进制文件烧录到 MY_MODULE 的命令示例。

```
python esp-idf/components/esptool_py/esptool/esptool.py -p /dev/ttyUSB0 -b 921600 -
↪-before default_reset --after hard_reset --chip auto  write_flash --flash_mode␣
↪dio --flash_size detect --flash_freq 40m 0x2B000 ./factory_param_MY_MODULE.bin
```

直接修改出厂参数二进制文件

用二进制工具打开出厂参数二进制文件，根据 factory_param_type.csv 中的参数偏移量，直接在相应位置进行修改。

将修改后的 factory_param.bin 烧录至 flash（详情请见下载指导）。

5.6 如何自定义分区

本文档介绍了如何通过修改 ESP-AT 提供的 at_customize.csv 表来自定义 ESP32-C2 设备中的分区。共有两个分区表：一级分区表和二级分区表。

一级分区表 partitions_at.csv 供系统使用，在此基础上生成 partitions_at.bin 文件。如果一级分区表出错，系统将无法启动。因此，不建议修改 partitions_at.csv。

ESP-AT 提供了二级分区表 at_customize.csv 供您存储自定义数据块。它基于一级分区表。

要修改 ESP32-C2 设备中的分区，请按照前三个步骤操作。第四部分举例说明了前三个步骤。

- *修改 at_customize.csv*
- *生成 at_customize.bin*

5.6.1 修改 at_customize.csv

请参考下表找到模组的 at_customize.csv。

表 2: at_customize.csv 路径

平台	模组	路径
ESP32-C2	ESP32C2-2MB（所有带 2 MB flash 的 ESP32C2（ESP8684）系列）	module_config/module_esp32c2-2mb/at_customize.csv
ESP32-C2	ESP32C2-4MB（所有带 4 MB flash 的 ESP32C2（ESP8684）系列）	module_config/module_esp32c2_default/at_customize.csv

然后，在修改 at_customize.csv 时遵循以下规则。

- 已定义的用户分区的 `Name` 和 `Type` 不可更改，但 `SubType`、`Offset` 和 `Size` 可以更改。
- 如果您需要添加一个新的用户分区，请先检查它是否已经在 ESP-IDF (`esp_partition.h`) 中定义。
  - 如果已定义，请保持 `Type` 值与 ESP-IDF 的相同。
  - 如果未定义，请将 `Type` 设置为 `0x40`。
- 用户分区的 `Name` 不应超过 `16` 字节。
- `at_customize` 分区的默认大小定义在 partitions_at.csv 表中，添加新用户分区时请不要超出范围。

5.6.2 生成 at_customize.bin

修改 at_customize.csv 后，您可以重新编译 ESP-AT 工程或使用 python 脚本 `gen_esp32part.py` 来生成 at_customize.bin 文件。

如果使用脚本，在 ESP-AT 工程根目录下执行以下命令，并替换 `INPUT` 和 `OUTPUT`。

```
python esp-idf/components/partition_table/gen_esp32part.py <INPUT> [OUTPUT]
```

- `INPUT` 替换为待解析的 at_customize.csv 或二进制文件的路径。
- `OUTPUT` 替换为生成的二进制或 CSV 文件的路径，如果省略，将使用标准输出。

5.6.3 烧录 at_customize.bin 至 ESP32-C2 设备

将 at_customize.bin 下载到 flash 中。关于如何将二进制文件烧录至 ESP32-C2 设备，请参考*烧录 AT 固件至设备*。下表为不同模组 at_customize.bin 文件的下载地址。

表 3: 不同模组 at_customize.bin 的下载地址

平台	模组	地址	大小
ESP32-C2	ESP32C2-2MB（所有带 2 MB flash 的 ESP32C2（ESP8684）系列）	0x1A000	0x26000
ESP32-C2	ESP32C2-4MB（所有带 4 MB flash 的 ESP32C2（ESP8684）系列）	0x1E000	0x42000

在某些情况下，必须将 at_customize.bin 下载到 flash 后才能使用一些 AT 命令：

- *AT+SYSFLASH*：查询或读写 *flash* 用户分区
- *AT+FS*：文件系统操作
- `SSL` 服务器相关命令
- `BLE` 服务器相关命令

5.6.4 示例

本节介绍如何将名为 `test` 的 4 KB 分区添加到 ESP8684-MINI-1 4MB 模组中。

首先找到 ESP8684-MINI-1 4MB 的 at_customize.csv 表，设置新分区的 `Name`、`Type`、`SubType`、`Offset` 和 `Size`。

第二步，重新编译 ESP-AT 工程，或者在 ESP-AT 根目录下执行 python 脚本生成 at_customize.bin。

```
python esp-idf/components/partition_table/gen_esp32part.py -q ./module_config/
↪module_esp32c2_default/at_customize.csv at_customize.bin
```

然后，ESP-AT 根目录中会生成 at_customize.bin。

第三步，下载 at_customize.bin 至 flash。

在 ESP-AT 工程根目录下执行以下命令，并替换 `PORT` 和 `BAUD`。

```
python esp-idf/components/esptool_py/esptool/esptool.py -p PORT -b BAUD --before␣
↪default_reset --after hard_reset --chip auto  write_flash --flash_mode dio --
↪flash_size detect --flash_freq 40m 0x1E000 ./at_customize.bin
```

- `PORT` 替换为端口名称。
- `BAUD` 替换为波特率。

5.7 如何增加一个新的模组支持

ESP-AT 工程支持多个模组，并提供了模组的配置文件：factory_param_data.csv 和 module_config。下表列出了 ESP-AT 工程支持的平台（即芯片系列）、模组以及模组配置文件的位置。

平台	模组	默认配置文件
ESP32	• WROOM-32 • PICO-D4 • SOLO-1 • MINI-1	• module_config/module_esp32_default/sdkconfig.defaults • module_config/module_esp32_default/sdkconfig_silence.defaults
ESP32	WROVER-32	• module_config/module_wrover-32/sdkconfig.defaults • module_config/module_wrover-32/sdkconfig_silence.defaults
ESP32	ESP32-D2WD	• module_config/module_esp32-d2wd/sdkconfig.defaults • module_config/module_esp32-d2wd/sdkconfig_silence.defaults
ESP32	ESP32_QCLOUD	• module_config/module_esp32_qcloud/sdkconfig.defaults • module_config/module_esp32_qcloud/sdkconfig_silence.defaults
ESP32-C2	ESP32C2-2MB（所有带 2 MB flash 的 ESP32C2（ESP8684）系列）	• module_config/module_esp32c2-2mb/sdkconfig.defaults • module_config/module_esp32c2-2mb/sdkconfig_silence.defaults
ESP32-C2	ESP32C2-4MB（所有带 4 MB flash 的 ESP32C2（ESP8684）系列）	• module_config/module_esp32c2_default/sdkconfig.defaults • module_config/module_esp32c2_default/sdkconfig_silence.defaults
ESP32-C3	MINI-1	• module_config/module_esp32c3_default/sdkconfig.defaults • module_config/module_esp32c3_default/sdkconfig_silence.defaults
ESP32-C3	ESP32C3_QCLOUD	• module_config/module_esp32c3_qcloud/sdkconfig.defaults • module_config/module_esp32c3_qcloud/sdkconfig_silence.defaults

注意:

- 当 `./build.py menuconfig` 中的 `silence mode` 为 `0` 时，对应模块的配置文件为 `sdkconfig.defaults`。
- 当 `./build.py menuconfig` 中的 `silence mode` 为 `1` 时，对应模块的配置文件为 `sdkconfig_silence.defaults`。

如果要在 ESP-AT 工程中添加对某个 ESP32-C2 模组的支持，则需要修改这些配置文件。此处的“ESP32-C2 模组”指的是：

- ESP-AT 工程暂未适配支持的模组，包括 ESP-AT 已适配相应芯片的模组，和未适配相应芯片的模组。但不建议添加后者，因为工作量巨大，此文档也不做阐述。
- ESP-AT 工程已适配支持的模组，但用户需要对其修改默认配置的。

本文档将说明如何在 ESP-AT 工程中为 ESP-AT 已支持的某款 ESP32-C2 芯片添加新的模组支持，下文中以添加对 ESP32-WROOM-32 支持为例，该模组使用 SDIO 而不是默认的 UART 接口。

5.7.1 在 factory_param_data.csv 添加模组信息

打开本地的 factory_param_data.csv ，在表格最后插入一行，根据实际需要设置相关参数。本例中，我们将 platform 设置为 PLATFORM_ESP32、module_name 设置为 WROOM32-SDIO，其他参数设置值见下表（参数含义请参考*factory_param_type.csv*）。

- platform: PLATFORM_ESP32
- module_name: WROOM32-SDIO
- description:
- magic_flag: 0xfcfc
- version: 3
- reserved1: 0
- tx_max_power: 78
- uart_port: 1
- start_channel: 1
- channel_num: 13
- country_code: CN
- uart_baudrate: -1
- uart_tx_pin: -1
- uart_rx_pin: -1
- uart_cts_pin: -1
- uart_rts_pin: -1
- tx_control_pin: -1
- rx_control_pin: -1

5.7.2 修改 esp_at_module_info 结构体

详情请参考*修改 esp_at_module_info 结构体*。

5.7.3 配置模组文件

首先，进入 module_config 文件夹，创建一个子文件夹来存放模组的配置文件（文件夹名称为小写），然后在其中加入配置文件 IDF_VERSION、at_customize.csv、partitions_at.csv、sdkconfig.defaults 以及 sdkconfig_silence.defaults。

本例中，我们复制粘贴 module_esp32_default 文件夹及其中的配置文件，并重命名为 module_wroom32-sdio。在本例中，配置文件 IDF_VERSION、at_customize.csv 和 partitions_at.csv 无需修改，我们只需修改 sdkconfig.defaults 和 sdkconfig_silence.defaults：

- 使用 module_wroom32-sdio 文件夹下的分区表，需要修改如下配置

```
CONFIG_PARTITION_TABLE_CUSTOM_FILENAME="module_config/module_wroom32-sdio/
↪partitions_at.csv"
CONFIG_PARTITION_TABLE_FILENAME="module_config/module_wroom32-sdio/partitions_
↪at.csv"
CONFIG_AT_CUSTOMIZED_PARTITION_TABLE_FILE="module_config/module_wroom32-sdio/
↪at_customize.csv"
```

- 使用 SDIO 配置，移除 UART 配置
  - 移除 UART 配置

```
CONFIG_AT_BASE_ON_UART=n
```

  - 新增 SDIO 配置

```
CONFIG_AT_BASE_ON_SDIO=y
```

完成上述步骤后，可重新编译 ESP-AT 工程生成模组固件。本例中，我们在配置工程时，应选择 PLATFORM_ESP32 和 WROOM32-SDIO 来生成模组固件。

5.8 SPI AT 指南

本文档主要介绍 SPI AT 的实现与使用，主要涉及以下几个方面：

5.8.1 简介

SPI AT 基于 AT 工程，使用 SPI 协议进行数据通信。在使用 SPI AT 时，MCU 设备作为 SPI master，ESP32-C2 设备作为 SPI slave，通信双方通过 SPI 协议实现基于 AT 命令的数据交互。

使用 SPI AT 的优势

AT 工程默认使用 UART 协议进行数据通信，但是 UART 协议在一些需要高速传输数据的应用场景并不适用，因此，使用支持更高传输速率的 SPI 协议传输数据成为一种较好的选择。

如何启用 SPI AT？

您可以通过下述步骤配置并启用 SPI AT：

1. 通过 `./build.py menuconfig -> Component config -> AT -> communicate method for AT command -> AT through HSPI` 使能 SPI AT。
2. 通过 `./build.py menuconfig -> Component config -> AT -> communicate method for AT command -> AT SPI Data Transmission Mode` 选择 SPI 数据传输模式。
3. 通过 `./build.py menuconfig -> Component config -> AT -> communicate method for AT command -> AT SPI GPIO settings` 配置 SPI 使用的 GPIO 管脚。
4. 通过 `./build.py menuconfig -> Component config -> AT -> communicate method for AT command -> AT SPI driver settings` 选择 SPI 从机的工作模式，并配置相关缓存区的大小。
5. 重新编译 `esp-at` 工程（参考*编译 ESP-AT 工程*），烧录新的固件并运行。

SPI AT 默认管脚

下表给出了不同系列的 ESP32-C2 设备使用 SPI AT 时默认的硬件管脚：

表 4: SPI AT 默认管脚

信号	GPIO 编号
SCLK	6
MISO	2
MOSI	7
CS	10
HANDSHAKE	3
GND	GND
QUADWP (qio/qout) [1]	8
QUADHD (qio/qout) [1]	9

说明 1：QUADWP 引脚和 QUADHD 引脚仅在使用 4 线 SPI 工作时使用。

您可以通过 `./build.py menuconfig > Component config > AT > communicate method for AT command > AT through HSPI > AT SPI GPIO settings`，然后编译工程来配置 SPI AT 对应的管脚（参考*编译 ESP-AT 工程*）。

5.8.2 使用 SPI AT

在使用 SPI AT 时，ESP32-C2 设备上运行的 SPI slave 工作在半双工通信模式下。

握手线 (handshake line)

SPI 是一种 master-slave 结构的外设，所有的传输均由 master 发起，slave 无法主动传输数据。但是，使用 AT 命令进行数据交互时，需要 ESP32-C2 设备主动能够主动上报一些信息。因此，我们在 SPI master 和 slave 之间添加了一个握手线，来实现 slave 主动向 master 上报信息的功能。

当 slave 需要传输数据时，将会把握手管脚主动拉高，这会在 master 侧产生一个上升沿的 GPIO 中断，master 发起与 slave 的通信，传输完成后，slave 将握手管脚拉低，结束此次通信。

使用握手线的具体方法为：

- Master 向 slave 发送 AT 数据时，使用握手线的方法为：
  1. master 向 slave 发送请求传输数据的请求，然后等待 slave 向握手线发出的允许发送数据的信号。
  2. master 检测到握手线上的 slave 发出的允许发送的信号后，开始发送数据。
  3. master 发送数据后，通知 slave 数据发送结束。
- Master 接收 slave 发送的 AT 数据时，使用握手线的方法为：
  1. slave 通过握手线通知 master 开始接收数据。
  2. master 接收数据，并在接收所有数据后，通知 slave 数据已经全部接收。

通信格式

SPI AT 的通信格式为 CMD+ADDR+DUMMY+DATA（读/写）。在使用 SPI AT 时，SPI master 使用到的一些通信报文介绍如下：

- Master 向 slave 发送数据时的通信报文：

表 5: 主机向从机发送数据

CMD（1 字节）	ADDR（1 字节）	DUMMY（1 字节）	DATA（高达 4092 字节）
0x3	0x0	0x0	data_buffer

- Master 向 slave 发送数据结束后，需要发送一条通知消息来结束本次传输，具体的通信报文为：

表 6: 主机通知从机，主机发送数据已经结束

CMD（1 字节）	ADDR（1 字节）	DUMMY（1 字节）	DATA
0x7	0x0	0x0	null

- Master 接收 slave 发送的数据时的通信报文：

表 7: 主机读取从机发送的数据

CMD（1 字节）	ADDR（1 字节）	DUMMY（1 字节）	DATA（高达 4092 字节）
0x4	0x0	0x0	data_buffer

- Master 接收 slave 发送的数据后，需要发送一条通知消息来结束本次传输，具体的通信报文为：

表 8: 主机通知从机，主机读取数据已经结束

CMD（1 字节）	ADDR（1 字节）	DUMMY（1 字节）	DATA
0x8	0x0	0x0	null

- Master 向 slave 发送请求传输指定大小数据的通信报文：

表 9: 主机向从机发送写数据请求

CMD（1 字节）	ADDR（1 字节）	DUMMY（1 字节）	DATA（4 字节）
0x1	0x0	0x0	data_info

其中 4 字节的 data_info 中包含了本次请求传输数据的数据包信息，具体格式如下：

1. Master 向 slave 发送的数据的字节数，长度 0 ~ 15 bit。
2. Master 向 slave 发送的数据包的序列号，该序列号在 master 每次发送时递增，长度 16 ~ 23 bit。
3. Magic 值，长度 24 ~ 31 bit，固定为 0xFE。

- Master 检测到握手线上有 slave 发出的信号后，需要发送一条消息查询 slave 进入接收数据的工作模式，还是进入到发送数据的工作模式，具体的通信报文为：

表 10: 主机发送请求，查询从机的可读/可写状态

CMD（1 字节）	ADDR（1 字节）	DUMMY（1 字节）	DATA（4 字节）
0x2	0x4	0x0	slave_status

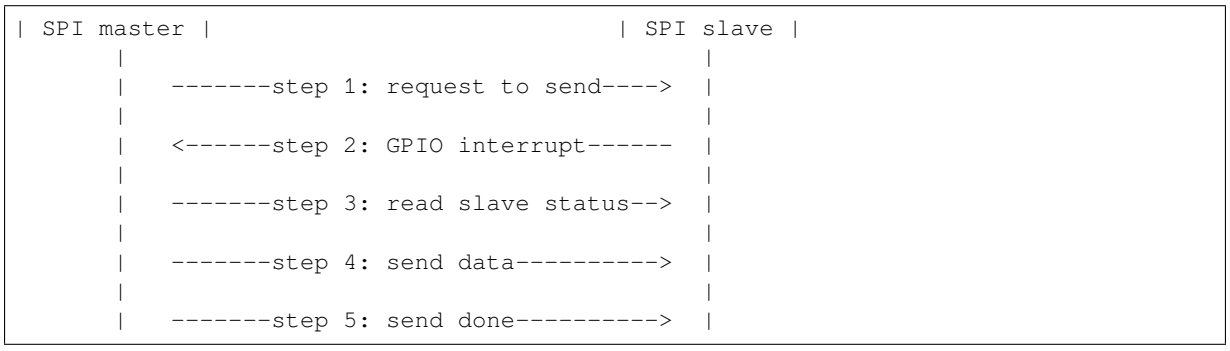
发送查询请求后，slave 返回的状态信息将存储在 4 字节的 slave_status 中，其具体的格式如下：

1. slave 需要向 master 发送的数据的字节数，长度 0 ~ 15 bit；仅当 slave 处于可读状态时，该字段数字有效。
2. 数据包序列号，长度 16 ~ 23 bit，当序列号达到最大值 0xFF 时，下一个数据包的序列号重新设置为 0x0。当 slave 处于可写状态时，该字段为 master 需向 slave 发送的下一下数据包的序列号；当 slave 处于可读状态时，该字段为 slave 向 master 发送的下一个数据包的序列号。
3. slave 的可读/可写状态，长度 24 ~ 31 bit，其中，0x1 代表可读，0x2 代表可写。

SPI AT 数据交互流程

SPI AT 数据交互流程主要分为两个方面：

- SPI master 向 slave 发送 AT 指令：



每个步骤具体的说明如下：

1. master 向 slave 发送请求向 slave 写数据的请求。
2. slave 接收 master 的发送请求，若此时 slave 允许接收数据，则向 slave_status 寄存器写入允许 master 写入的标志位，然后通过握手线触发 master 上的 GPIO 中断。
3. master 接收到中断后，读取 slave 的 slave_status 寄存器，检测到 slave 进入接收数据的状态。
4. master 开始向 slave 发送数据。
5. 发送数据结束后，master 向 slave 发送一条代表发送结束的消息。

- SPI slave 向 master 发送 AT 响应：

```
| SPI master |                                  | SPI slave |
       |                                          |
       |   <------step 1: GPIO interrupt------   |
       |                                          |
       |   -------step 2: read slave status-->   |
       |                                          |
       |   <------step 3: send data------------  |
       |                                          |
       |   -------step 4: receive done------->   |
```

每个步骤具体的说明如下：

1. slave 向 slave_status 寄存器写入允许 master 读取来自 slave 的数据的标志位，然后通过握手线触发 master 上的 GPIO 中断。
2. master 接收到中断后，读取 slave 的 slave_status 寄存器，检测到 slave 进入发送数据的状态。
3. master 开始接收来自 slave 的数据。
4. 数据接收完毕后，master 向 slave 发送一条代表接收数据结束的消息。

说明 1. 为了方便理解，我们还以发送 AT 命令为例，提供了通信涉及的所有交互流程和逻辑分析仪数据，请参考 at_spi_master/spi/esp32_c_series/README.md。

SPI AT 对应的 SPI master 侧示例代码

SPI AT 本身是作为 SPI slave 使用的，使用 SPI master 与 SPI slave 进行通信的示例代码请参考 at_spi_master/spi/esp32_c_series。

说明 1. 在使用 MCU 开发之前，强烈建议使用 ESP32-C3 或者 ESP32 模拟 MCU 作为 SPI master 来运行此示例，以方便在出现问题时更容易调试问题。

5.8.3 SPI AT 速率

测试说明

- 使用 ESP32 或者 ES32-C3 开发板作为 SPI master，运行 ESP-AT 中的 at_spi_master/spi/esp32_c_series 目录的代码。其软硬件配置如下：

1. 硬件：CPU 工作频率设置为 240 MHz，flash SPI mode 配置为 QIO 模式，flash 频率设置为 40 MHz。
2. 软件：基于 ESP-IDF v4.3 的编译环境，并将示例代码中的 streambuffer 的大小调整为 8192 字节。

- 使用 ESP32-C2 作为 SPI slave，编译并烧录 SPI AT 固件（参考*编译 ESP-AT 工程*），并将 ESP32-C2 配置工作在 TCP 透传模式。其软硬件配置如下：

1. 硬件：CPU 工作频率设置为 160 MHz。
2. 软件：SPI-AT 的实现代码中，将 streambuffer 的大小设置为 8192 字节，并使用 ESP-IDF 下的 example/wifi/iperf 中的 sdkconfig.defaults.esp32c3 中的相关配置参数。

测试结果

下表显示了我们在屏蔽箱中得到的通信速率结果：

表 11: SPI AT Wi-Fi TCP 通信速率

Clock	SPI mode	master->slave	slave->master
10 M	Standard	0.95 MByte/s	1.00 MByte/s
10 M	Dual	1.37 MByte/s	1.29 MByte/s
10 M	Quad	1.43 MByte/s	1.31 MByte/s
20 M	Standard	1.41 MByte/s	1.30 MByte/s
20 M	Dual	1.39 MByte/s	1.30 MByte/s
20 M	Quad	1.39 MByte/s	1.30 MByte/s
40 M	Standard	1.37 MByte/s	1.30 MByte/s
40 M	Dual	1.40 MByte/s	1.31 MByte/s
40 M	Quad	1.48 MByte/s	1.31 MByte/s

说明 1：当 SPI 的时钟频率较高时，受限于上层网络组件的限制，使用 Dual 或者 Quad 工作模式的通信速率想比较于 Standard 模式并未显著改善。

说明 2：更多关于 SPI 通信的介绍请参考对应模组的 技术参考手册。

5.9 如何实现 OTA 升级

本文档指导如何为 ESP32-C2 系列模块实现 OTA 升级。目前，ESP-AT 针对不同场景提供了以下三种 OTA 命令。您可以根据自己的需求选择适合的 OTA 命令。

1. *AT+USEROTA*
2. *AT+CIUPDATE*
3. *AT+WEBSERVER*

文档结构如下所示：

- *OTA 命令对比及应用场景*
- *使用 ESP-AT OTA 命令执行 OTA 升级*

5.9.1 OTA 命令对比及应用场景

AT+USEROTA

此命令通过 URL 实现 OTA 升级。您可以升级到放置在 HTTP 服务器上的固件。目前该命令仅支持应用分区升级。请参考*AT+USEROTA* 获取更多信息。

由于此命令属于用户自定义命令，您可以通过修改 at/src/at_user_cmd.c 源代码来实现该命令。

此命令的应用场景如下：

1. **您有属于自己的 HTTP 服务器**。
2. **您必须指定 URL**。

重要:

- 如果您升级的固件为非乐鑫正式发布的固件，在升级完成后，您可能无法使用 AT+CIUPDATE 命令升级，除非您按照*使用 AT+CIUPDATE 进行 OTA 升级* 创建了自己的设备。

AT+CIUPDATE

此命令使用 iot.espressif.cn 作为默认 HTTP 服务器。该命令不仅可以升级应用程序分区，还可以升级 `at_customize.csv` 中定义的用户自定义分区。如果您使用的是乐鑫发布的版本，该命令将只会升级

到乐鑫发布的版本。请参考*AT+CIUPDATE* 获取更多信息。

使用该命令升级自定义 bin 文件，请选择以下方式之一。

1. **将 iot.espressif.cn 替换为您自己的 HTTP 服务器，并实现交互流程**。如何实现自己的 AT+CIUPDATE 命令，请参考 at/src/at_ota_cmd.c。
2. **在 iot.espressif.cn 上创建一个设备，并在其上上传 bin 文件。（前提是模组中运行的固件已经对应您在乐鑫服务器上创建的设备。）**请参考*使用 AT+CIUPDATE 进行 OTA 升级* 获取更多信息。

此命令的应用场景如下：

1. **您只使用乐鑫发布的固件，只想升级到乐鑫发布的固件。**
2. **您希望升级自定义的 bin 文件，但没有自己的 HTTP 服务器。**
3. **您有自己的 HTTP 服务器。除了升级应用程序分区外，还希望升级在 at_customize.csv 文件中定义的用户自定义分区。**

AT+WEBSERVER

此命令通过浏览器或微信小程序升级 AT 固件。目前，该命令仅提供升级应用程序分区的功能。在开始升级之前，请启用 web 服务器命令并提前将 AT 固件复制到电脑或者手机上。您可以参考*AT+WEBSERVER* 或者*Web Server AT 示例* 获取更多信息。

为了实现您自己的 HTML 页面，请参考示例 fs_image/index.html。为了实现您自己的 AT+WEBSERVER 命令，请参考示例 at/src/at_web_server_cmd.c。

此命令的应用场景如下：

1. **您需要更方便快捷的 OTA 升级，不依赖于网络状态。**

重要：

- 如果您升级的固件为非乐鑫正式发布的固件，在升级完成后，您可能无法使用 AT+CIUPDATE 命令升级，除非您按照*使用 AT+CIUPDATE 进行 OTA 升级* 创建了自己的设备。

5.9.2 使用 ESP-AT OTA 命令执行 OTA 升级

使用 AT+USEROTA 进行 OTA 升级

请参考*AT+USEROTA* 获取更多信息。

使用 AT+CIUPDATE 进行 OTA 升级

通过*AT+CIUPDATE* 命令升级自定义的 bin 文件，首先要做的就是将 bin 文件上传到 iot.espressif.cn 并且获取到 **token** 值。以下步骤描述了如何在 iot.espressif.cn 上创建设备并上传 bin 文件。

1. 打开网站 http://iot.espressif.cn 或者 https://iot.espressif.cn。
2. 点击网页右上角的“Join”，输入您的名字，邮箱地址和密码。

备注：

- 当前 `Join` 功能暂不对新用户开放。如果您想使用该功能，请联系 乐鑫。

---

3. 点击网页左上角的“Device”，然后点击“Create”来创建一个设备。
4. 当设备创建成功后会生成一个密钥，如下图所示：
5. 使用该密钥来编译您的 OTA bin 文件。配置 AT OTA token 密钥的过程如下如所示：

   备注： 如果使用 SSL OTA，选项“The SSL token for AT OTA”也需要配置。

图 3: 打开 iot.espressif.cn 网站

图 4: 加入 iot.espressif.cn

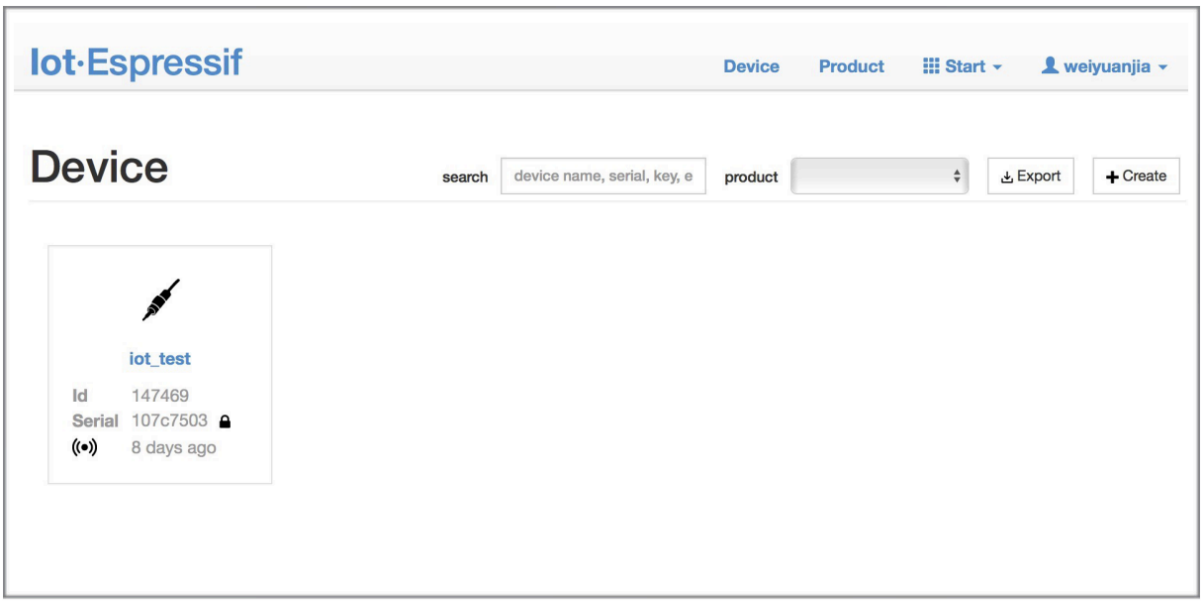


图 5: 点击“Device”

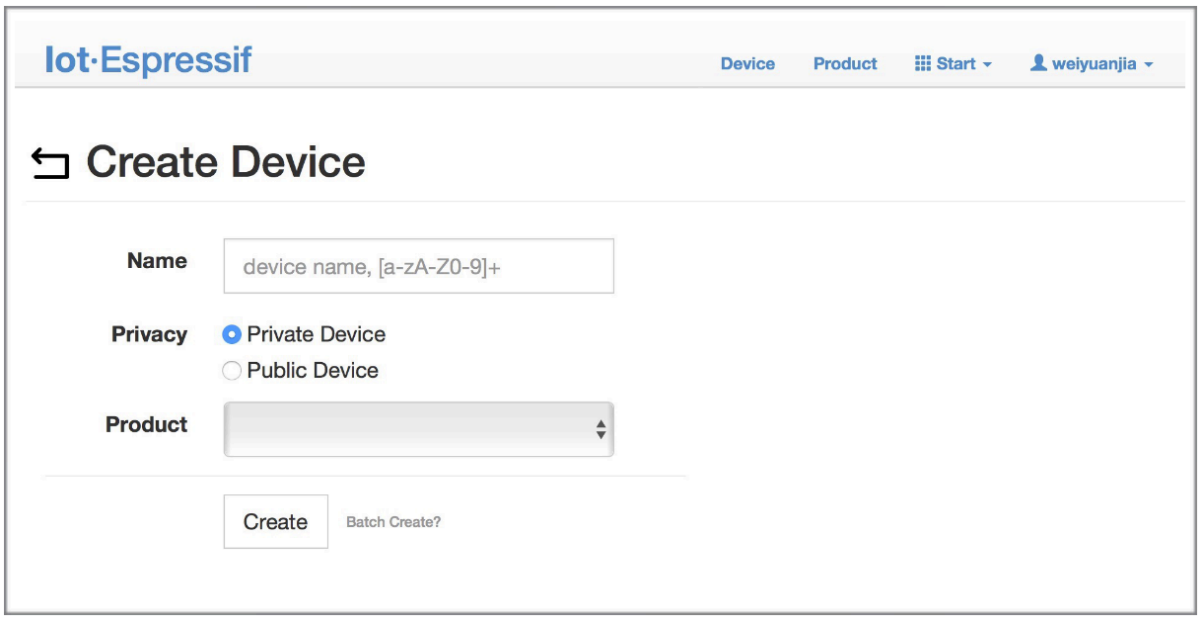


图 6: 点击“Create”

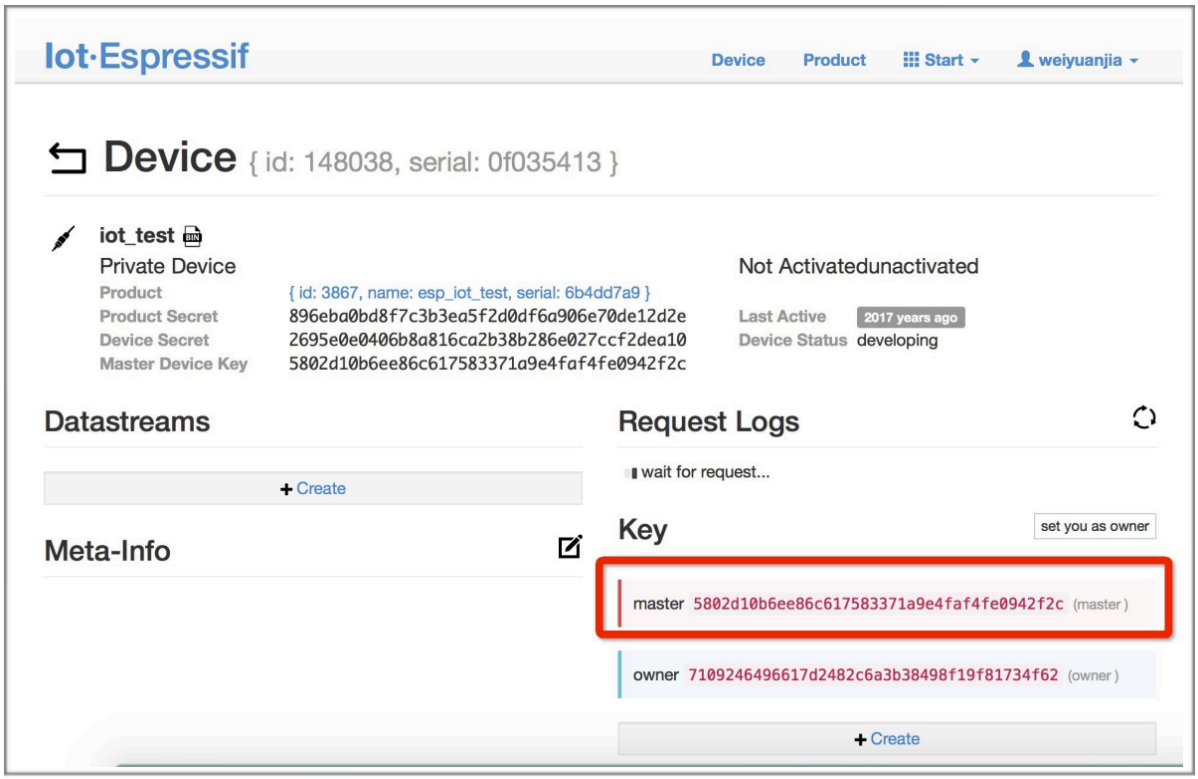


图 7: 生成一个密钥

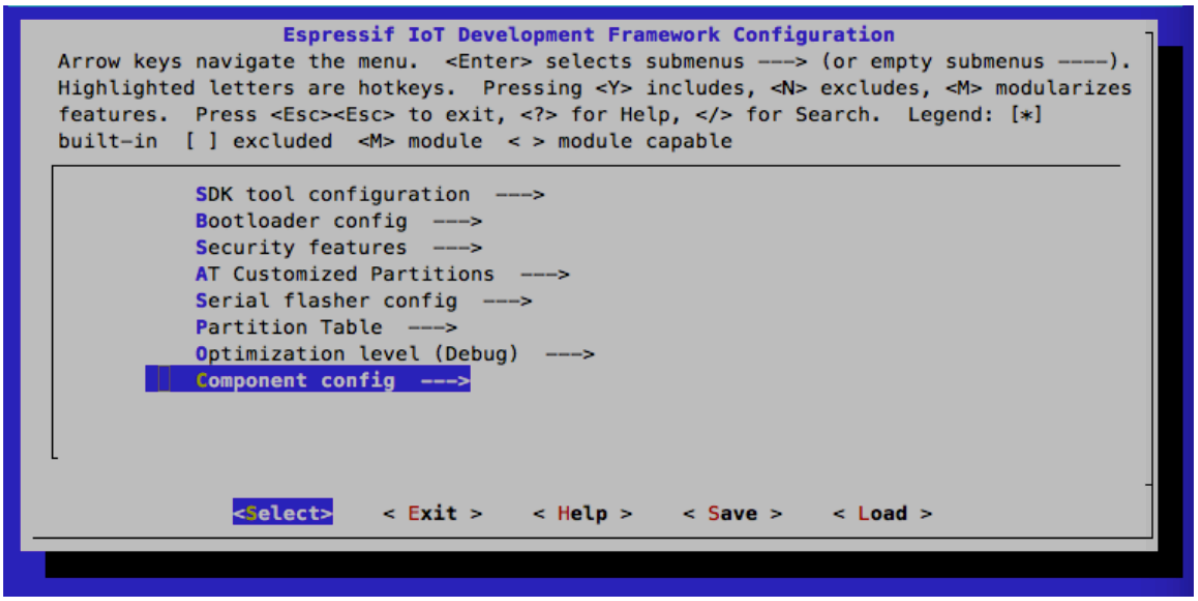


图 8: 配置 AT OTA token 密钥 - 步骤 1

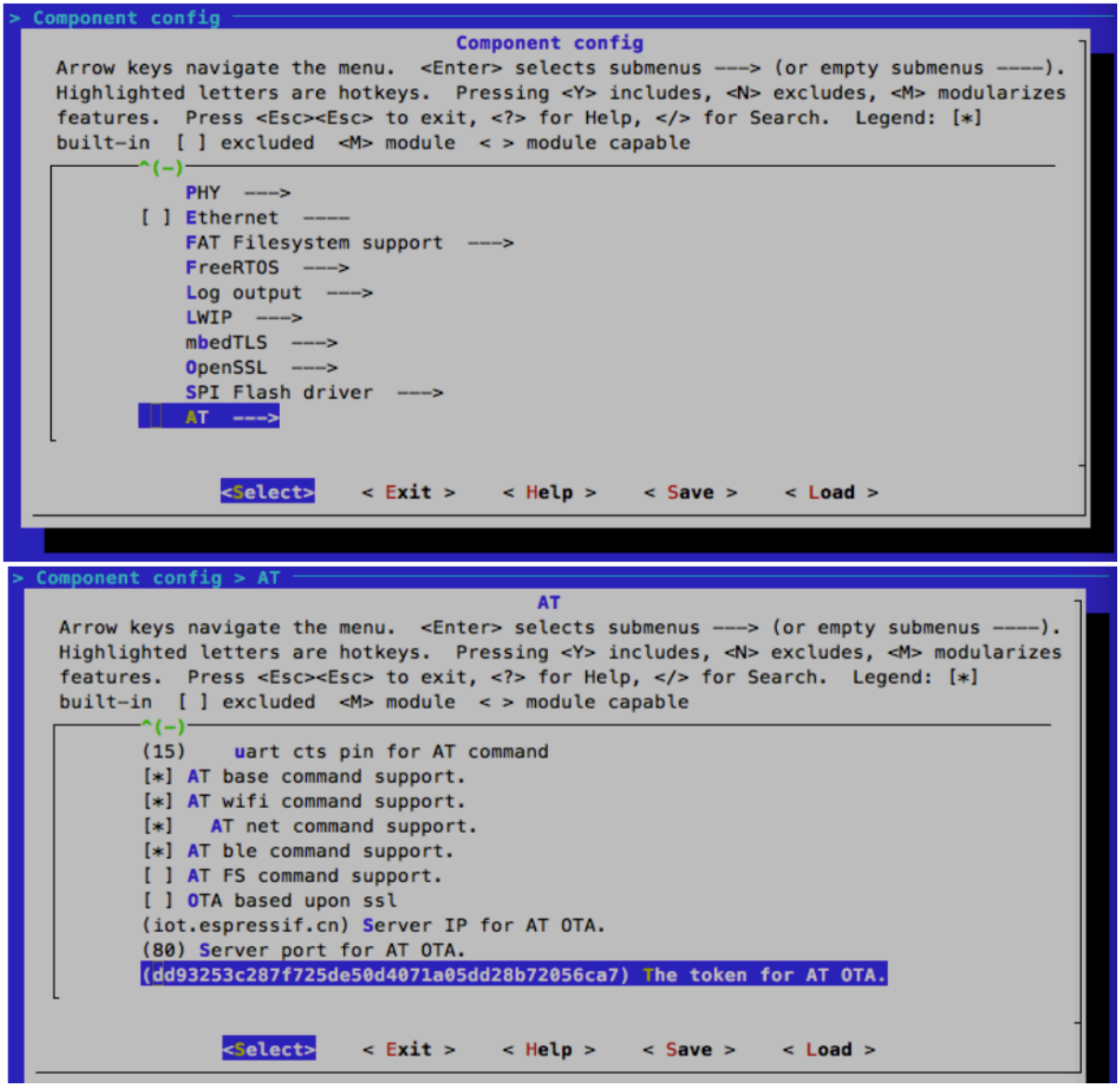


图 9: 配置 AT OTA token 密钥 - 步骤 2 和 3

6. 点击 “Product” 进入网页，如下如所示。单击创建的设备，在 “ROM Deploy” 下输入版本和 corename。将步骤 5 中的 bin 文件重命令为 “ota.bin” 并保存。

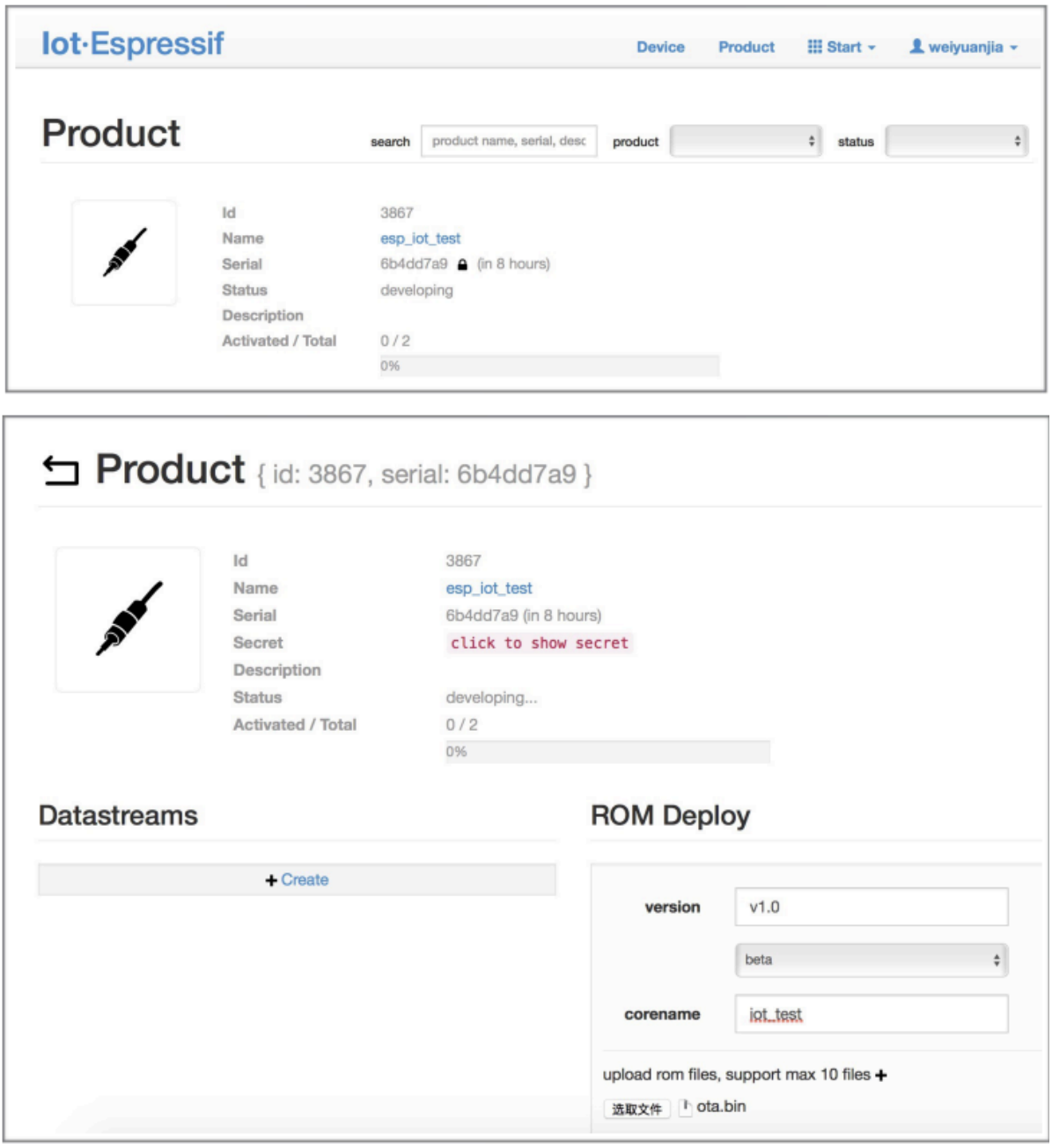


图 10: 输入版本和 corename

备注:

- 如果您想要升级 `at_customize.csv` 中定义的用户自定义分区，只需将 `ota.bin` 替换为用户自定义分区的 bin 文件即可。
- 对于 `corename` 字段，此字段仅仅用于帮助您区分 bin 文件。

7. 单击 ota.bin 将其保存为当前版本。
8. 在设备上运行*AT+USEROTA* 命令。如果网络已连接，将开始 OTA 升级。

重要:

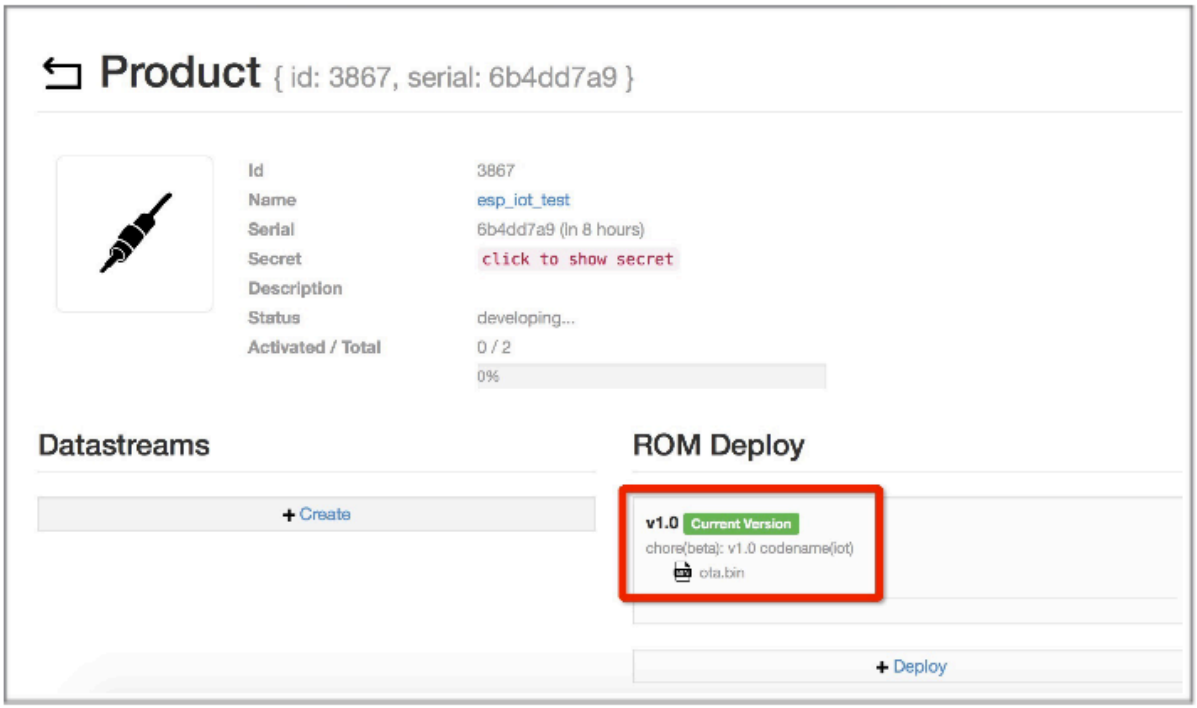


图 11: 保存当前版本的 ota.bin

- 设置上传到 iot.espressif.cn 的 bin 文件名称时，请遵循以下规则：
  - 如果升级 `app` 分区，请将 bin 文件名设置为 `ota.bin`。
  - 如果升级用户自定义的分区，请将 bin 文件名设置为 `at_customize.csv` 中的 `Name` 字段。例如，如果升级 `factory_Param` 分区，请将其设置为 `factory_ param.bin`。
- ESP-AT 将新固件存储在备用 OTA 分区中。这样，即使 OTA 由于意外原因失败，原始 ESP-AT 固件也能正常运行。但对于自定义分区，由于 ESP-AT 没有备份措施，请小心升级。
- **如果您打算从一开始只升级自定制的 bin 文件，那么在发布初始版本时，就应该将 OTA token 设置为自己的 token 值。**

使用 AT+WEBSERVER 进行 OTA 升级

请参考*AT+ WEBSERVER* 和*Web Server AT 示例* 获取更多信息。

5.10 如何更新 ESP-IDF 版本

ESP-AT 固件基于乐鑫物联网开发框架（ESP-IDF），每个版本的 ESP-AT 固件对应某个特定的 ESP-IDF 版本。强烈建议使用 ESP-AT 工程默认的 ESP-IDF 版本，**不建议**更新 ESP-IDF 版本，因为 libesp32c2_at_core.a 底层的 ESP-IDF 版本与 ESP-AT 工程的 IDF 版本不一致可能会导致固件的错误操作。

但是，在某些特殊情况下，ESP-IDF 的小版本更新也可能适用于 ESP-AT 工程。如果您需要更新，本文档可作为参考。

ESP-AT 固件对应的 ESP-IDF 版本记录在 `IDF_VERSION` 文件中，这些文件分布在 module_config 文件夹下的不同模组文件夹中。该文件描述了模组固件所基于的 ESP-IDF 的分支、提交 ID 和仓库地址。例如，`PLATFORM_ESP32C2` 平台的 `undefined` 模组的 `IDF_VERSION` 位于 module_config/module_esp32c2_default/IDF_VERSION。

如果您想为 ESP-AT 固件更新 ESP-IDF 版本，请按照以下步骤操作。

- 找到模组的 IDF_VERSION 文件。
- 根据需要更新其中的分支、提交 ID 和仓库地址。
- 删除 esp-at 根目录下原有的 esp-idf，以便下次编译时首先克隆 IDF_VERSION 中指定的 ESP-IDF 版本。
- 重新编译 ESP-AT 工程。

注意，当 ESP-AT 版本和 ESP-IDF 版本不匹配，编译时会报如下错误。

```
Please wait for the update to complete, which will take some time
```

5.11 ESP-AT 固件差异

本文档比较了同一 ESP32-C2 系列的 AT 固件在支持的命令集、硬件、模组方面的差异。

5.11.1 ESP32-C2 系列

本节介绍以下 ESP32-C2 系列 AT 固件的区别：

- ESP32C2-2MB_AT_Bin（本节简称为 **ESP32C2-2MB Bin**）
- ESP32C2-4MB_AT_Bin（本节简称为 **ESP32C2-4MB Bin**）

支持的命令集

下表列出了官方适配的 ESP32-C2 系列 AT 固件默认支持哪些命令集（用 ✅ 表示）、默认不支持但可以在配置和编译 ESP-AT 工程后支持的命令集（用 ✅ 表示）、完全不支持的命令集（用 ❌ 表示），下表没有列出的命令集也为完全不支持的命令集。正式发布的固件见*AT 固件*，已适配但未发布的模组固件，需要自行编译。自行编译的固件无法从乐鑫官方服务器进行 OTA 升级。

命令集	ESP32C2-2MB Bin	ESP32C2-4MB Bin
base	✅	✅
user	✅	✅
wifi	✅	✅
net	✅	✅
MDNS	✅	✅
WPS	✅	✅
smartconfig	✅	✅
ping	✅	✅
MQTT	✅	✅
http	✅	✅
FS	✅	✅
driver	✅	✅
WPA2	✅	✅
WEB	✅	✅
OTA	✅	✅
blufi	✅	✅

硬件差异

硬件	ESP32C2-2MB	ESP32C2-4MB
Flash	2 MB	4 MB
PSRAM	✗	✗
UART 管脚[1]	TX: 7 RX: 6 CTS: 19 RTS: 20	TX: 7 RX: 6 CTS: 5 RTS: 4

支持的模组

下表列出了 ESP32-C2 系列 AT 固件支持的模组或芯片。

模组/芯片	ESP32C2-2MB Bin	ESP32C2-4MB Bin
ESP8684 MINI 系列	✓	✓
ESP8684 WROOM 系列	✓	✓

5.12 如何从 GitHub 下载最新临时版本 AT 固件

由于 ESP-AT 在 GitHub 上启用 CI（持续集成），因此每次代码被推送到 GitHub 都会生成 ESP-AT 固件的临时版本。

> **注意：AT 固件的临时版本仅用于测试，乐鑫不对其负责**，您需要自行测试功能。
>
> **请保存好下载的固件以及下载链接**，用于后续可能的问题定位。

以下步骤指导您如何从 GitHub 下载最新临时版本 AT 固件。

1. 登录您的 GitHub 账号
   在开始之前，**请先登录您的 GitHub 账号**，因为下载固件需要登录权限。
2. 打开网页 https://github.com/espressif/esp-at
3. 点击“Actions”进入“Actions”页面
4. 点击“Branch”选择指定分支
   默认为 master 分支。如果您想下载指定分支的临时固件，点击“Branch”进入指定分支的 workflow 页面。
5. 点击最新的 workflow 进入 workflow 页面
6. 将页面滚动到最后，在 `Artifacts` 页面中选择相对应的模组
7. 点击下载模组的最新临时版本 AT 固件

> **备注：** 如果您在 `Artifacts` 页面中没有找到对应的模组，请参考*ESP-AT 固件差异* 选择类似固件进行下载。

[1] UART 管脚可自定义，详情请参考*如何设置 AT 端口管脚*。

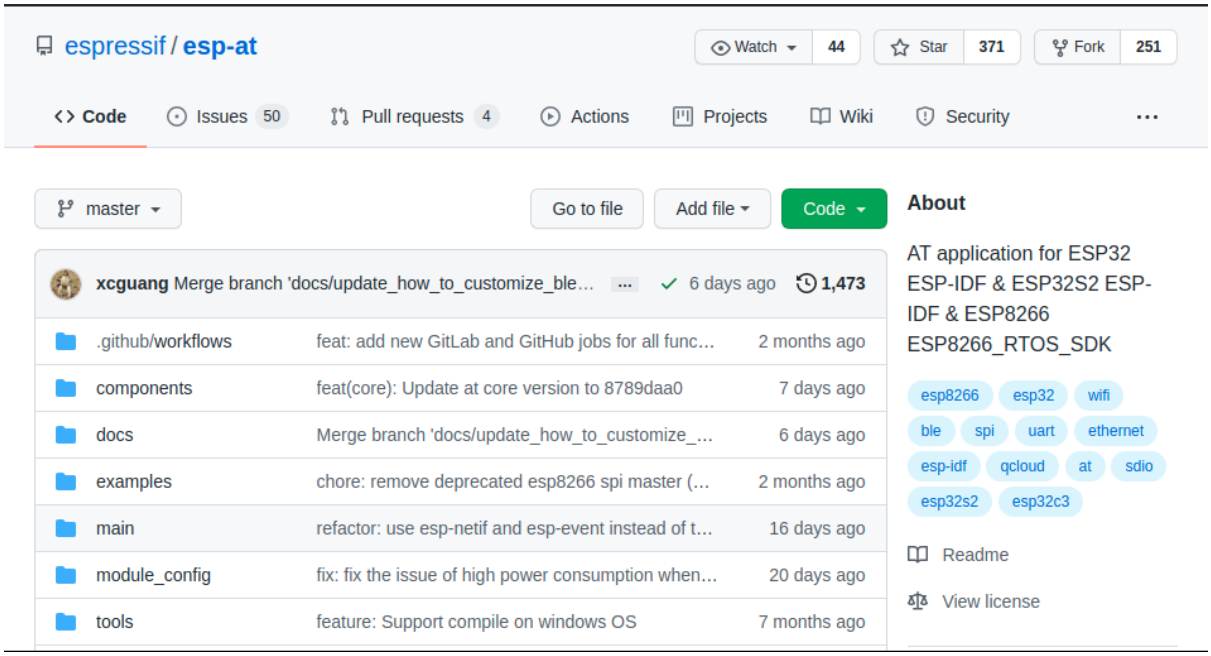


图 12: ESP-AT GitHub 官方页面

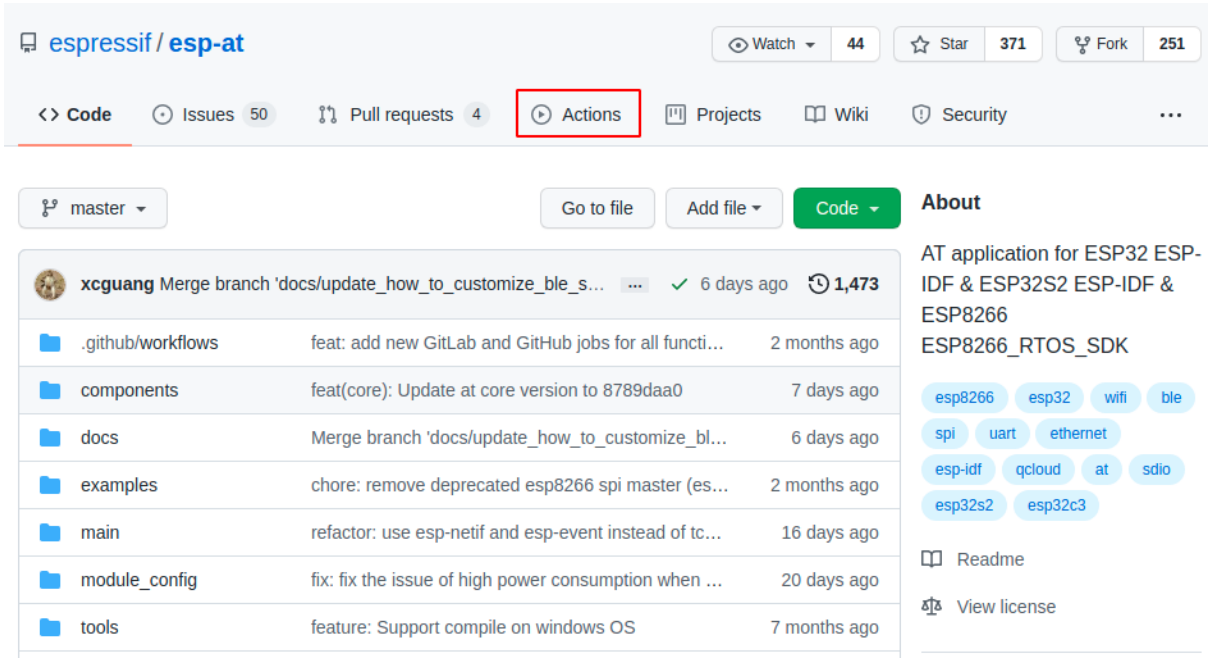


图 13: 点击 Actions

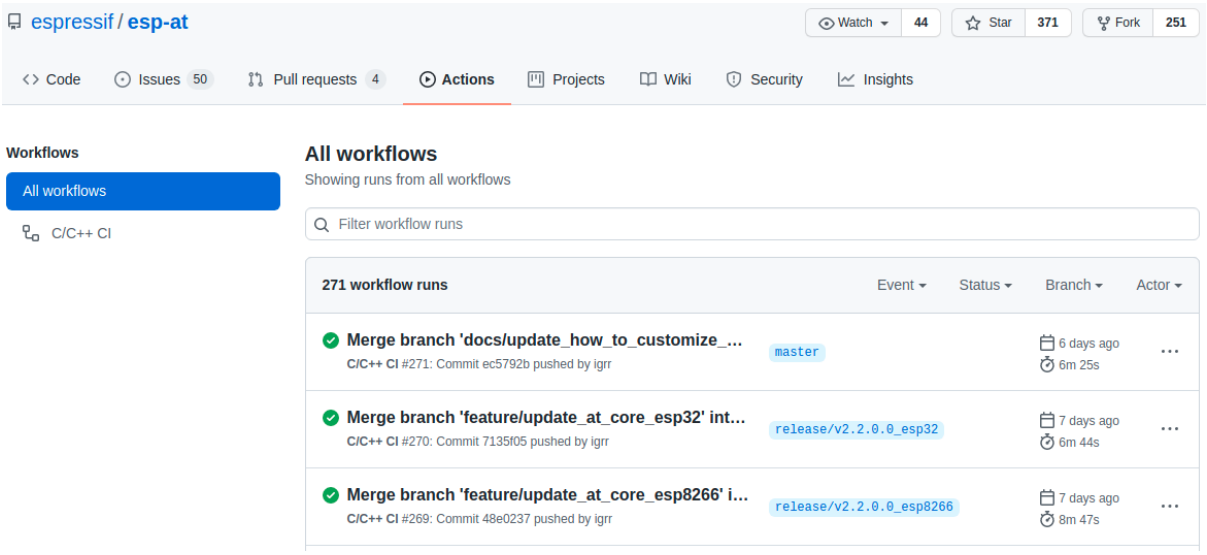


图 14: Actions 页面

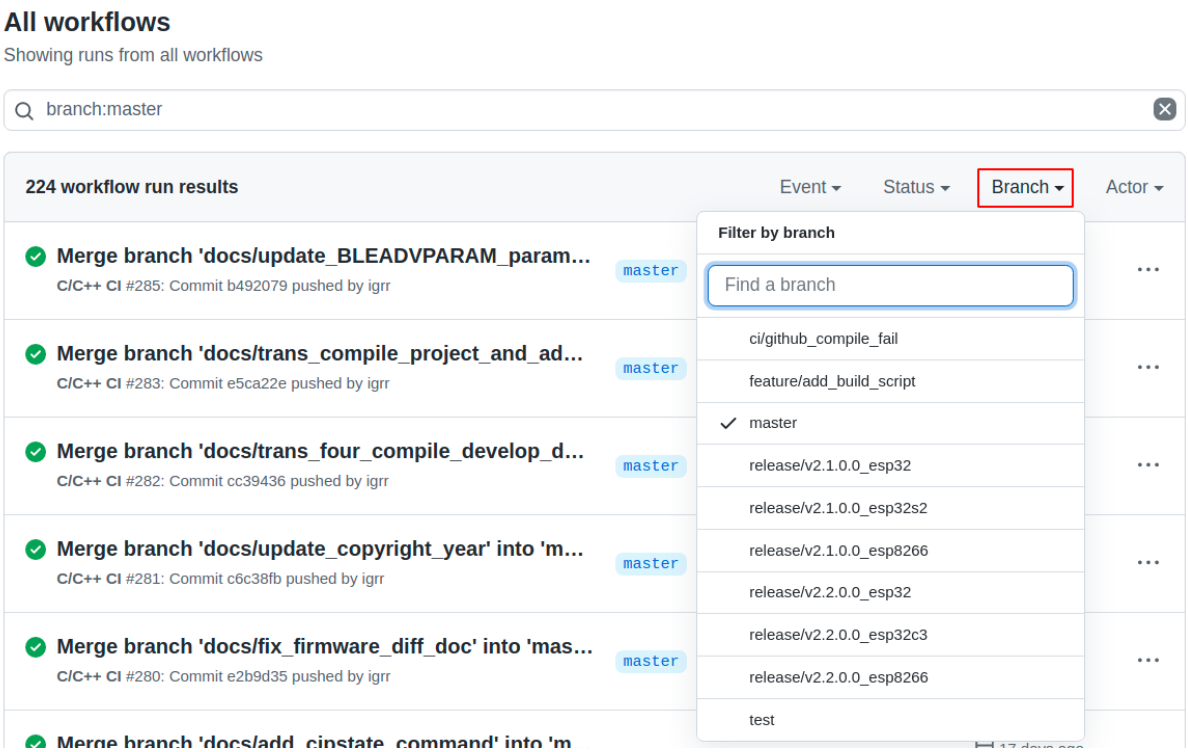


图 15: 点击 Branch

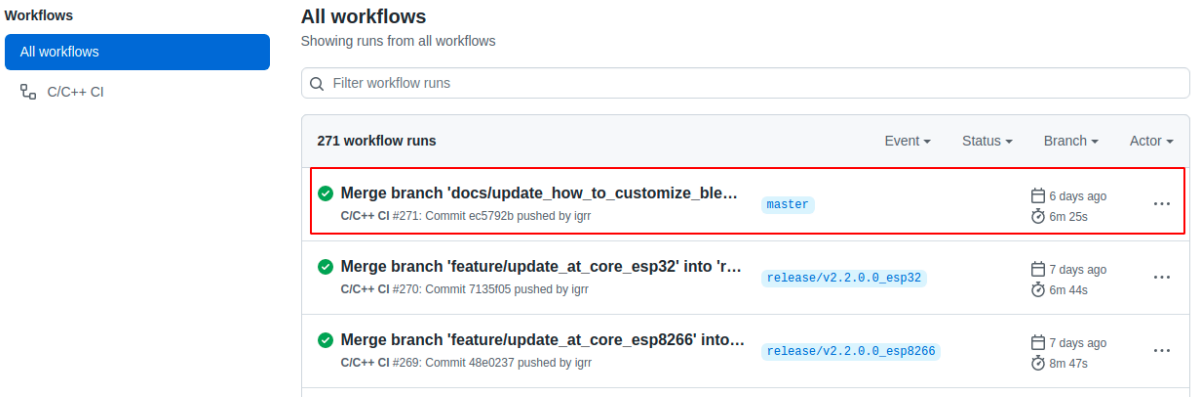


图 16: 点击最新的 Workflow

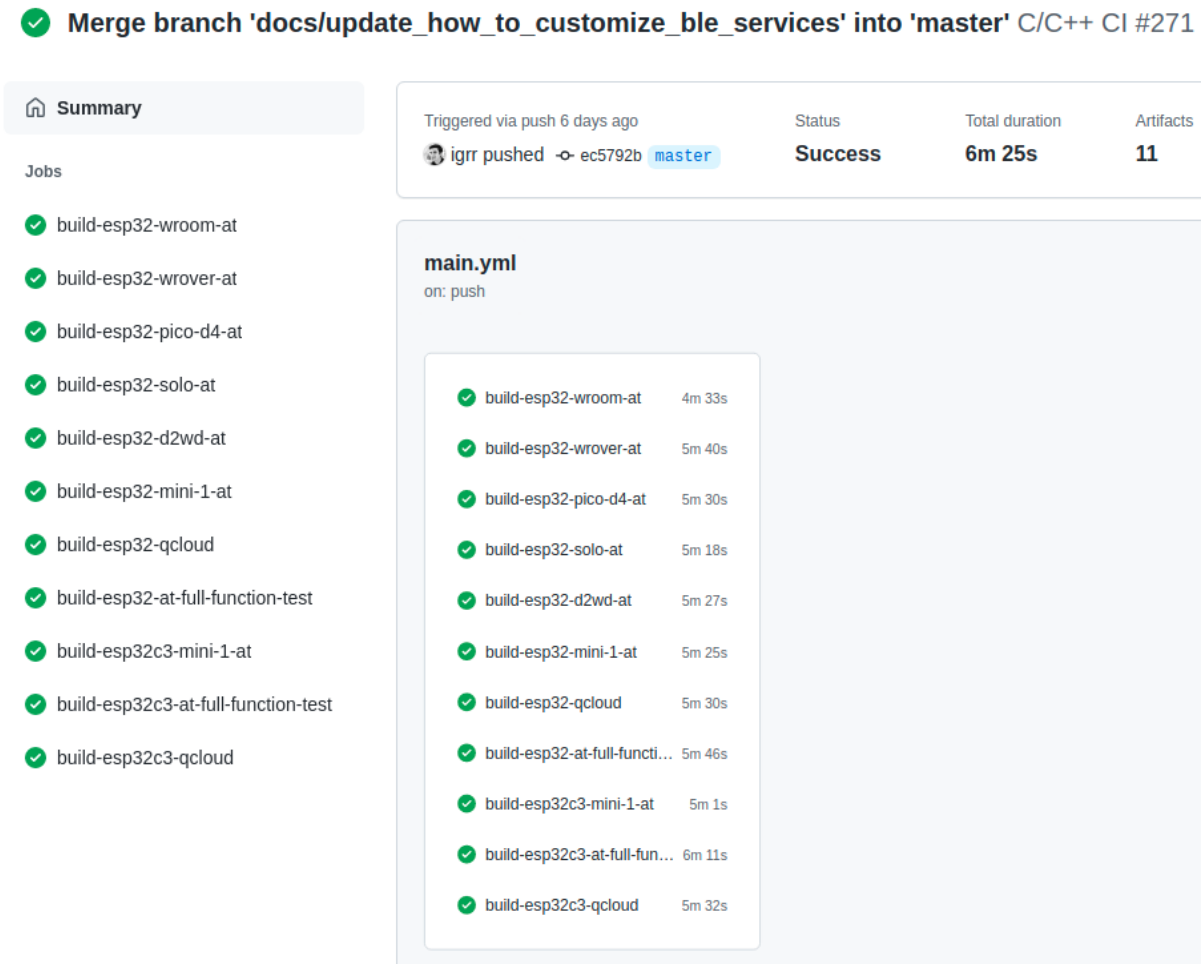


图 17: 最新的 Workflow 页面

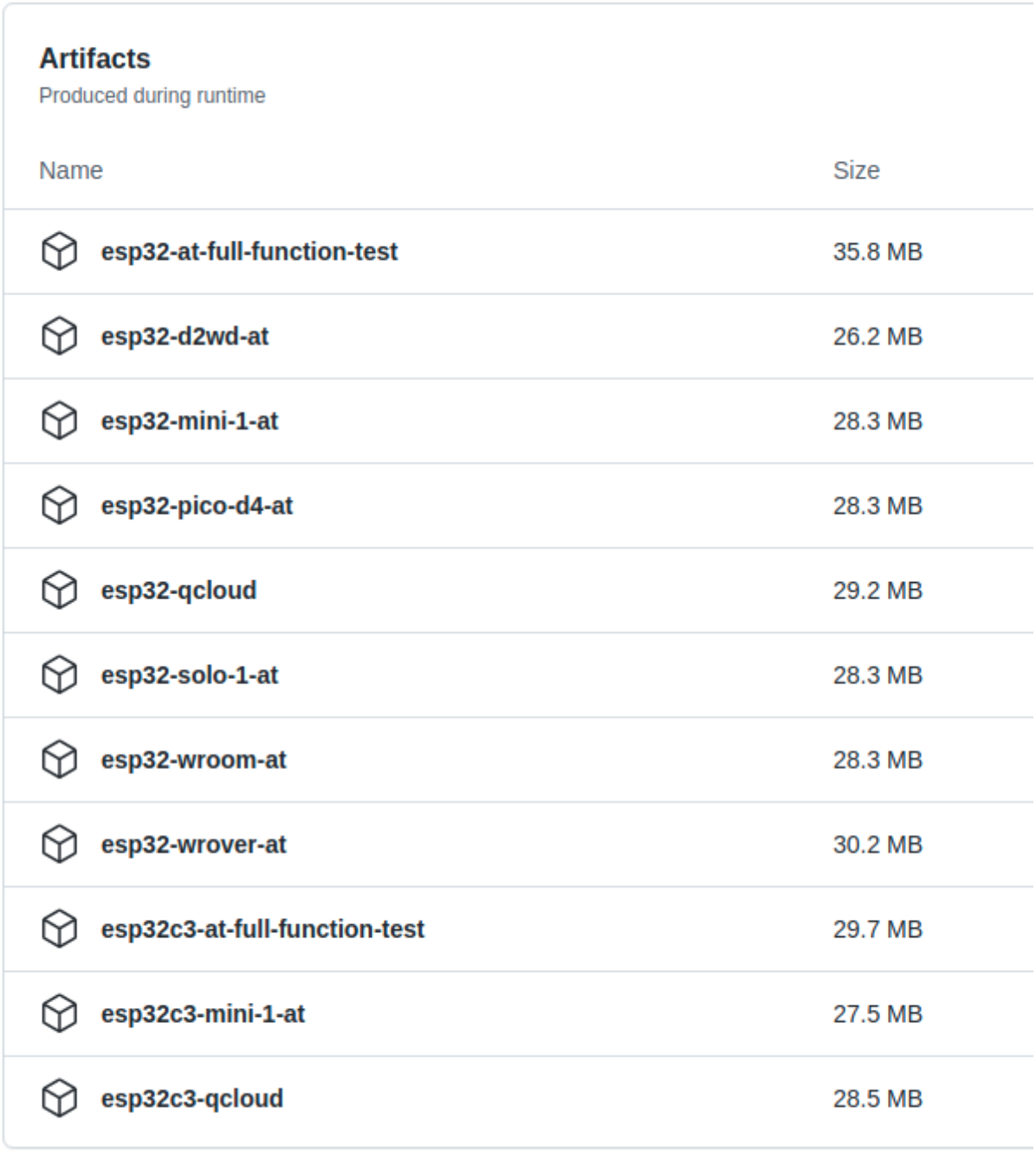


图 18: Artifacts 页面

5.13 如何生成 PKI 文件

ESP-AT 提供 python 脚本 `AtPKI.py` 将 SSL 服务器证书文件，SSL 客户端证书文件，MQTT 证书文件和 WPA2 证书文件（包括 CA 证书，`cert` 证书和私钥）转换为二进制文件。

5.13.1 证书二进制文件格式

除了将证书文件转换为二进制文件外，脚本 `AtPKI.py` 还会向二进制文件中添加一些额外信息。

转换单个证书文件时，将会在文件头中以小端模式添加 12 字节数据，并在文件末尾进行 4 字节对齐。

字段	描述
magic code（2 字节）	0xF1 0xF1
列表大小（2 字节）	证书文件的数量
长度（4 字节）	剩余文件长度 = 总的文件大小 - magic code 大小 - 列表大小 - 长度大小
类型（1 字节）	ca: 0x01 certificate: 0x02 key: 0x03
ID（1 字节）	用于匹配证书文件
内容长度（2 字节）	转换为二进制文件的证书文件的大小

以生成证书二进制文件 小节中生成的 `client_cert.bin` 文件为例，二进制文件开头的小端模式 12 字节数据如下所示：

字段	描述
magic code（2 字节）	0xF1 0xF1
列表大小（2 字节）	0x02 0x00
长度（4 字节）	0x20 0x09 0x00 0x00
类型（1 字节）	0x02
ID（1 字节）	0x00
内容长度（2 字节）	0x8C 0x04

转换多个证书文件时，脚本 `AtPKI.py` 还会在除第一个证书文件外的每个证书文件的开头插入 4 字节数据。文件开头的小端模式 4 字节数据如下：

字段	描述
类型（1 字节）	ca: 0x01 certificate: 0x02 key: 0x03
ID（1 字节）	用于匹配证书文件
内容长度（2 字节）	转换为二进制文件的证书文件的大小

以生成证书二进制文件 小节中生成的 client_cert.bin 文件为例，文件开头的小端模式 4 字节数据如下所示：

字段	描述
类型（1 字节）	certificate: 0x02
ID（1 字节）	0x01
内容长度（2 字节）	0x8C 0x04

5.13.2 生成证书二进制文件

请选择以下方式之一生成证书二进制文件。

- 脚本生成
- 编译期间生成

脚本生成

脚本 AtPKI.py 的路径是 tools/AtPKI.py。您可以通过 -h 选项获取脚本的帮助信息。您也可以直接通过以下命令生成二进制文件。

```
python <SCRIPT_PATH> generate_bin [-b OUTPUT_BIN_NAME] <PKI_LIST> <source_file>
```

- SCRIPT_PATH：脚本路径。如果您正处于 “tools” 目录下，则 SCRIPT_PATH 为 AtPKI.py。
- OUTPUT_BIN_NAME：要保存生成的目标文件名（目标文件名的绝对地址或者相对地址）；如果 -b OUTPUT_BIN_NAME 被省略，则脚本将会在当前目录下生成 PKI.bin 文件。
- PKI_LIST：必须等于 ca、cert、key 中的一个。
- source_file：您想转换的证书源文件（证书源文件的绝对地址或者相对地址）。

以 ESP-AT 的 SSL 客户端证书文件为例，您可以执行下面的命令在 tools 目录下生成 client_cert.bin 二进制文件。

```
python AtPKI.py generate_bin -b ./client_cert.bin  cert ../components/customized_
↪partitions/raw_data/client_cert/client_cert_00.crt cert ../components/customized_
↪partitions/raw_data/client_cert/client_cert_01.crt
```

编译期间生成

ESP-AT 中证书文件的存储路径为 components/customized_partitions/raw_data。

以 ESP-AT 的 SSL 客户端证书文件为例。如果您想生成自己的 SSL 客户端证书二进制文件，您必须将目录 client_ca 下的 CA 证书替换为自己的 CA 证书，将目录 client_cert 目录下的 cert 证书替换为自己的 cert 证书，将 client_ key 目录下的私钥替换为自己的私钥。

如果您有多套证书文件，请按照您的证书链依次放置于对应的目录下。建议您可以将文件名以数字结尾以确保证书文件的解析顺序。

替换完成之后，您可以参考编译 *ESP-AT* 工程 来编译 ESP-AT 工程。

5.13.3 下载或者更新证书二进制文件

脚本 AtPKI.py 仅仅负责将证书文件转换为二进制文件。您可以通过以下方式将二进制文件烧录到 flash 中：

通过烧录工具烧录

- Windows
  请下载 Windows Flash 下载工具。
  请参考 zip 文件夹中 `readme.pdf` 或者 `doc` 目录获取更多有关该工具的信息。
- Linux or macOS
  请使用 esptool.py。
  您可以在 ESP-AT 根目录下执行下面的命令烧录二进制文件。

```
esptool.py --chip auto --port PORTNAME --baud 921600 --before default_reset --
↪after hard_reset write_flash -z --flash_mode dio --flash_freq 40m --flash_
↪size 4MB ADDRESS FILEDIRECTORY
```

将 `PORTNAME` 替换为您的串口。将 `ADDRESS` 替换为烧录地址。将 `FILEDIRECTORY` 替换为二进制文件所在的目录。

通过命令更新

- *AT+SYSFLASH* 命令
  以 `ESP32-C2` 模组为例，您可以执行以下命令来更新 `client_cert` 分区。请参考*AT+SYSFLASH* 获取更多信息。
  1. 查询 flash 用户分区
     命令：

```
AT+SYSFLASH?
```

     响应：

```
+SYSFLASH:"server_cert",64,1,0x1f000,0x2000
+SYSFLASH:"server_key",64,2,0x21000,0x2000
+SYSFLASH:"server_ca",64,3,0x23000,0x2000
+SYSFLASH:"client_cert",64,4,0x25000,0x2000
+SYSFLASH:"client_key",64,5,0x27000,0x2000
+SYSFLASH:"client_ca",64,6,0x29000,0x2000
+SYSFLASH:"factory_param",64,7,0x2b000,0x1000
+SYSFLASH:"wpa2_cert",64,8,0x2c000,0x2000
+SYSFLASH:"wpa2_key",64,9,0x2e000,0x2000
+SYSFLASH:"wpa2_ca",64,10,0x30000,0x2000
+SYSFLASH:"mqtt_cert",64,11,0x32000,0x2000
+SYSFLASH:"mqtt_key",64,12,0x34000,0x2000
+SYSFLASH:"mqtt_ca",64,13,0x36000,0x2000
+SYSFLASH:"ble_data",64,14,0x38000,0x6000
+SYSFLASH:"fatfs",1,129,0x47000,0x19000

OK
```

  2. 擦除 `client_cert` 分区
     命令：

```
AT+SYSFLASH=0,"client_cert"
```

     响应：

```
OK
```

  3. 更新 `client_cert` 分区
     命令：

```
AT+SYSFLASH=1,"client_cert",0,2344
```

     响应：

```
>
```

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。当写入操作完成之后，系统会提示以下信息。

```
OK
```

- *AT+CIUPDATE* 命令
  例如，您可以执行以下命令来更新 client_ca 分区。请参考*AT+CIUPDATE* 获取更多信息。

重要： 如果您想通过这种方式更新 client_ca 分区，您必须实现自己的 OTA 设备，请参考文档*如何实现 OTA 升级*。

```
AT+CIUPDATE=1,"v2.2.0.0","client_ca"
```

备注： 您必须确保烧录的地址是正确的，否则 ESP-AT 固件可能不能工作。查看烧录地址的最简单方法是执行命令 **AT+SYSFLASH?**。

5.14 AT API Reference

5.14.1 Header File

- components/at/include/esp_at_core.h

5.14.2 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.

参数

- **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.

参数

- **para_index** –the index of parameter
- **value** –the value parsed

返回

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

esp_at_para_parse_result_type **esp_at_get_para_as_float**(int32_t para_index, float *value)

Parse float parameter from command string.

参数

- **para_index** –the index of parameter
- **value** –the value parsed

返回

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

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.

参数

- **para_index** –the index of parameter
- **result** –the pointer that point to the result.

返回

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

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.

参数 **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.

参数

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

返回

- true : succeed
- false : fail

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,

参数

- **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,

参数 **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,

参数 **ops** –custom operate functions set

uint32_t **esp_at_get_version**(void)

get at module version number,

返回 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,

参数 **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,

参数

- **data** –data buffer to be written
- **len** –data length

返回

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

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,

参数

- **data** –data buffer to be written
- **len** –data length

返回

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

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

read data from device,

参数

- **data** –data buffer
- **len** –data length

返回

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

bool **esp_at_port_wait_write_complete**(int32_t timeout_msec)

wait for transmitting data completely to peer device,

参数 **timeout_msec** –timeout time,The unit is millisecond.

返回

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

int32_t **esp_at_port_get_data_length**(void)

get the length of the data received,

返回

- >= 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.

参数 **terminator** –the line terminator

返回

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

uint8_t ***esp_at_custom_cmd_line_terminator_get**(void)

Get AT command line terminator,by default, the return string is "\r\n".

返回 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.

参数

- **type** –the type of the partition
- **subtype** –the subtype of the partition
- **label** –Partition label

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

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

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

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

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

参数 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.

void **at_handle_result_code**(*esp_at_result_code_string_index* code, void *pbuf)

5.14.3 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

5.14.4 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

5.14.5 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.

5.14.6 Enumerations

enum **esp_at_status_type**
esp_at_status some custom function interacting with AT
Values:

enumerator **ESP_AT_STATUS_NORMAL**
Normal mode.Now mcu can send AT command

enumerator **ESP_AT_STATUS_TRANSMIT**
Transparent Transmition mode

enum **at_sleep_mode_t**
Values:

enumerator **AT_DISABLE_SLEEP**

enumerator **AT_MIN_MODEM_SLEEP**

enumerator **AT_LIGHT_SLEEP**

enumerator **AT_MAX_MODEM_SLEEP**

enumerator **AT_SLEEP_MAX**

enum **esp_at_module**

module number,Now just AT module

Values:

enumerator **ESP_AT_MODULE_NUM**

AT module

enum **esp_at_error_code**

subcategory number

Values:

enumerator **ESP_AT_SUB_OK**

OK

enumerator **ESP_AT_SUB_COMMON_ERROR**

reserved

enumerator **ESP_AT_SUB_NO_TERMINATOR**

terminator character not found (“\r\n” expected)

enumerator **ESP_AT_SUB_NO_AT**

Starting “AT” not found (or at, At or aT entered)

enumerator **ESP_AT_SUB_PARA_LENGTH_MISMATCH**

parameter length mismatch

enumerator **ESP_AT_SUB_PARA_TYPE_MISMATCH**

parameter type mismatch

enumerator **ESP_AT_SUB_PARA_NUM_MISMATCH**

parameter number mismatch

enumerator **ESP_AT_SUB_PARA_INVALID**

the parameter is invalid

enumerator **ESP_AT_SUB_PARA_PARSE_FAIL**

parse parameter fail

enumerator **ESP_AT_SUB_UNSUPPORT_CMD**

the command is not supported

enumerator **ESP_AT_SUB_CMD_EXEC_FAIL**
the command execution failed

enumerator **ESP_AT_SUB_CMD_PROCESSING**
processing of previous command is in progress

enumerator **ESP_AT_SUB_CMD_OP_ERROR**
the command operation type is error

enum **esp_at_para_parse_result_type**
the result of AT parse
Values:

enumerator **ESP_AT_PARA_PARSE_RESULT_FAIL**
parse fail,Maybe the type of parameter is mismatched,or out of range

enumerator **ESP_AT_PARA_PARSE_RESULT_OK**
Successful

enumerator **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:

enumerator **ESP_AT_RESULT_CODE_OK**
“OK”

enumerator **ESP_AT_RESULT_CODE_ERROR**
“ERROR”

enumerator **ESP_AT_RESULT_CODE_FAIL**
“ERROR”

enumerator **ESP_AT_RESULT_CODE_SEND_OK**
“SEND OK”

enumerator **ESP_AT_RESULT_CODE_SEND_FAIL**
“SEND FAIL”

enumerator **ESP_AT_RESULT_CODE_IGNORE**
response nothing, just change internal status

enumerator **ESP_AT_RESULT_CODE_PROCESS_DONE**
response nothing, just change internal status

enumerator **ESP_AT_RESULT_CODE_OK_AND_INPUT_PROMPT**

enumerator **ESP_AT_RESULT_CODE_MAX**

5.14.7 Header File

- components/at/include/esp_at.h

5.14.8 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

void **esp_at_main_preprocess**(void)

some workarounds for esp-at project

5.14.9 Macros

ESP_AT_PORT_TX_WAIT_MS_MAX

ESP_AT_FACTORY_PARAMETER_SIZE

Chapter 6

AT FAQ

6.1 AT 固件

6.1.1 我的模组没有官方发布的固件，如何获取适用的固件？

如果*AT 固件* 章节中没有发布相关固件，您可考虑以下选择：

- 使用相同硬件配置的模组的固件（点击*ESP-AT 固件差异*。
- 为你的模组*创建出厂参数二进制文件* 并*自行编译固件*。

6.1.2 如何获取 AT 固件源码？

ESP-AT 固件部分开源，开源仓库参考 esp-at。

6.1.3 官网上放置的 AT 固件如何下载？

- 烧录工具请下载 Flash 下载工具。
- 烧录地址请参考*AT 下载指南*。

6.1.4 如何整合 ESP-AT 编译出来的所有 bin 文件？

可以使用 Flash 下载工具 的 combine 按钮进行整合。

6.1.5 模组出厂 AT 固件是否支持流控？

- 该模组支持硬件流控，但是不支持软件流控。
- 对于是否开启硬件流控，您可以通过串口命令*AT+UART_CUR* 或者*AT+UART_DEF* 进行修改。
- *硬件接线参考*。

6.2 AT 命令与响应

6.2.1 AT 提示 busy 是什么原因？

- 提示“busy”表示正在处理前一条命令，无法响应当前输入。因为 AT 命令的处理是线性的，只有处理完前一条命令后，才能接收下一条命令。
- 当有多余的不可□字符输入时，系统也会提示“busy”或“ERROR”，因为任何串口的输入，均被认为是命令输入。
  - 串口输入 AT+GMR (换行符 CR LF) (空格符)，由于 AT+GMR (换行符 CR LF) 已经是一条完整的 AT 命令了，系统会执行该命令。此时如果系统尚未完成 AT+GMR 操作，就收到了后面的空格符，将被认为是新的命令输入，系统提示“busy”。但如果是系统已经完成了 AT+GMR 操作，再收到后面的空格符，空格符将被认为是一条错误的命令，系统提示“ERROR”。
  - MCU 发送 AT+CIPSEND 后，收到 busy p.. 响应，MCU 需要重新发送数据。因为 busy p.. 代表上一条命令正在执行，当前输入无效。建议等 AT 上一条命令响应后，MCU 再重新发送新命令。

6.2.2 AT 固件，上电后发送第一个命令总是会返回下面的信息，为什么？

```
ERR CODE:0x010b0000
busy p...
```

- 此信息代表的是”正在处理上一条命令”。
- 一般情况下只会显示”busy p…”，显示 ERR CODE 是因为打开了错误代码提示。
- 如果是上电的第一条命令就返回了这个错误码信息，可能的原因是：这条命令后面多跟了换行符/空格/其他符号，或者连续发送了两个或多个 AT 命令。

6.2.3 在不同模组上的默认 AT 固件支持哪些命令，以及哪些命令从哪个版本开始支持？

- 如果您想了解 ESP-AT 在不同模组上默认固件都支持哪些命令，您可以参考*ESP-AT 固件差异*。
- 如果您想查找某个命令从哪个版本开始支持，以及各个版本上修复了哪些问题，您可以参考 release notes。

6.2.4 主 MCU 给 ESP32-C2 设备发 AT 命令无返回，是什么原因？

当主 MCU 给 ESP32-C2 设备发送 AT 命令后需要添加结束符号，在程序中的写法为：" ATrn"。可参见*检查 AT 固件是否烧录成功*。

6.2.5 ESP-AT 命令是否支持 ESP-WIFI-MESH？

ESP-AT 当前不支持 ESP-WIFI-MESH。

6.2.6 AT 是否支持 websocket 命令？

- 默认命令不支持 websocket 命令。
- 可通过自定义命令实现，代码参考 websocket，以及*添加自定义 AT 命令*。

6.2.7 是否有 AT 命令连接阿里云以及腾讯云示例？

若使用*通用 AT 固件*，可参考以下示例：

- 阿里云应用参考：AT+MQTT aliyun。
- 腾讯云应用参考：AT+MQTT QCloud。

6.2.8 AT 命令是否可以设置低功耗蓝牙发射功率？

可以。ESP32-C2 的 Wi-Fi 和 Bluetooth LE 共用一根天线，可使用*AT+RFPOWER* 命令设置。

6.2.9 如何支持那些默认固件不支持但可以在配置和编译 ESP-AT 工程后支持的命令？

例如在 ESP32-C2 系列支持连接 WPA2 企业级路由器功能，需编译时在 menuconfig 中开启该功能 `./build.py menuconfig > Component config > AT > [*]AT WPA2 Enterprise command support`。

6.2.10 AT 命令中特殊字符如何处理？

可以参考*AT 命令分类* 章节中的转义字符语法。

6.2.11 AT 命令中串口波特率是否可以修改？(默认：115200)

AT 命令串口的波特率是可以修改的。

- 第一种方法，您可以通过串口命令*AT+UART_CUR* 或*AT+UART_DEF*。
- 第二种方法，您可以重新编译 AT 固件，编译介绍：*如何编译 AT 工程* 与*修改 UART 波特率配置*。

6.2.12 ESP32-C2 使用 AT 指令进入透传模式，如果连接的热点断开，ESP32-C2 能否给出相应的提示信息？

- 可以通过命令*AT+SYSMSG* 进行配置，可设置 AT+SYSMSG=4，如果连接的热点断开，串口会上报“WIFI DISCONNECTrn”。
- 需要注意的是，该命令在 AT v2.1.0 之后添加，v2.1.0 及之前的版本无法使用该命令。

6.3 硬件

6.3.1 在不同模组上的 AT 固件要求芯片 flash 多大？

- 对于 ESP32-C2 系列模组，您可以参考*ESP-AT 固件差异*。

6.3.2 AT 固件如何查看 error log？

- ESP32-C2 在 download port 查看 error log，默认 UART0 为 GPIO21、GPIO20。
- 详情可以参阅*硬件连接*。

6.3.3 AT 在 ESP32-C2 模组上的 UART1 通信管脚与 ESP32-C2 模组的 datasheet 默认 UART1 管脚不一致？

- ESP32-C2 支持 IO 矩阵变换，在编译 ESP-AT 的时候，可以在 menuconfig 中通过软件配置修改 UART1 的管脚配置，所以就会出现和 datasheet 管脚不一致的情况。
- 管脚详情可以参阅 factory_param_data.csv。

6.4 性能

6.4.1 AT Wi-Fi 连接耗时多少？

- 在办公室场景下，AT Wi-Fi 连接耗时实测为 5 秒。但在实际使用中，Wi-Fi 连接时间取决于路由器性能，网络环境，模块天线性能等多个条件。
- 可以通过*AT+CWJAP* 的 **<jap_timeout>** 参数，来设置最大超时时间。

6.4.2 ESP-AT 固件中 TCP 发送窗口大小是否可以修改？

- TCP 发送窗口当前无法通过命令修改，需要配置和编译 ESP-AT 工程生成新的固件。
- 可以重新配置 menuconfig 参数，`Component config > LWIP > TCP > Default send buffer size`。

6.4.3 ESP32-C2 AT 吞吐量如何测试及优化？

- AT 吞吐量测试的影响因素较多，建议使用 esp-idf 中的 iperf 示例进行测试（用 AT 测试时，请使用透传方式，并将数据量调整为 1460 字节连续发送）。
- 若测试速率不满足需求，您可以参考*如何提高 ESP-AT 吞吐性能* 来提高速率。

6.5 其他

6.5.1 乐鑫芯片可以通过哪些接口来传输 AT 命令？

- ESP32-C2 支持 UART 接口通信。
- AT 默认固件是使用 UART 接口来传输。用户如果需要使用 SDIO 或者 SPI 接口进行通信，可以基于 ESP-AT 配置编译，详情请见编译和开发。
- 更多资料请参考 使用 AT SDIO 接口 ， 使用 AT SPI 接口 ， 或 使用 AT 套接字接口 。

6.5.2 ESP32-C2 AT 如何指定 TLS 协议版本？

编译 ESP-AT 工程时，可以在./build.py menuconfig -> Component config -> mbedTLS 目录下，可以将不需要的版本关闭使能。

6.5.3 AT 固件如何修改 TCP 连接数？

- 目前 AT 默认固件的 TCP 最大连接数为 5。
- ESP32-C2 AT 最大支持 16 个 TCP 连接，可以在 menuconfig 中进行配置，配置方法如下：
  - ./build.py menuconfig—> Component config—> AT—> (16)AT socket maximum connection number
  - ./build.py menuconfig—> LWIP—> (16)Max number of open sockets

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 AT stands for "attention".
AT 是 attention 的缩写。

AT command port The port that is used to send AT commands and receive responses. More details are in the *AT port* introduction.

AT 命令端口 也称为 AT 命令口，用于发送 AT 命令和接收响应的端口。更多介绍请参考*AT 端口*。

AT log port The port that is used to output log. More details are in the *AT port* introduction.

AT 日志端口 也称为 AT 日志口，用于输出 AT 日志的端口。更多介绍请参考*AT 端口*。

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 如何设置 *AT* 端口管脚 for how to customize them.

AT 端口 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，参考如何设置 *AT* 端口管脚 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

Command Mode Default operating mode of AT. In the command mode, any character received by the AT command port will be treated as an AT command, and AT returns the command execution result to the AT command port. AT enters *Data Mode* from *Command Mode* in the following cases.

- After sending the *AT+CIPSEND* set command successfully and returns >.
- After sending the *AT+CIPSEND* execute command successfully and returns >.
- After sending the *AT+CIPSENDL* set command successfully and returns >.
- After sending the *AT+CIPSENDEX* set command successfully and returns >.
- After sending the *AT+SAVETRANSLINK* set command successfully and sending the *AT+RST* command and restart the module.

In the data mode, send the +++ command, AT will exit from *Data Mode* and enter the *Command Mode*.

命令模式 AT 的默认工作模式。在命令模式下，AT 命令端口收到的任何字符都会被当作 AT 命令进行处理，同时 AT 会在命令端口回复命令执行结果。AT 在下列情况下，会从命令模式 进入数据模式。

- 发送*AT+CIPSEND* 设置命令成功，回复 > 之后
- 发送*AT+CIPSEND* 执行命令成功，回复 > 之后
- 发送*AT+CIPSENDL* 设置命令成功，回复 > 之后
- 发送*AT+CIPSENDEX* 设置命令成功，回复 > 之后
- 发送*AT+SAVETRANSLINK* 设置命令成功，再发送 (*AT+RST*) 命令，模组重启之后

在数据模式下，发送+++ 命令，会从数据模式 退出，进入命令模式。

Data Mode In the data mode, any character received by the AT command port will be treated as data (except for special +++) instead of the AT command, and these data will be sent to the opposite end without modification. AT enters *Data Mode* from *Command Mode* in the following cases.

- After sending the *AT+CIPSEND* set command successfully and returns >.
- After sending the *AT+CIPSEND* execute command successfully and returns >.
- After sending the *AT+CIPSENDL* set command successfully and returns >.
- After sending the *AT+CIPSENDEX* set command successfully and returns >.
- After sending the *AT+SAVETRANSLINK* set command successfully and sending the *AT+RST* command and restart the module.

In the data mode, send the +++ command, AT will exit from *Data Mode* and enter the *Command Mode*.

数据模式 在数据模式下，AT 命令端口收到的任何字符都会被当作数据（除了特殊的+++），而不是 AT 命令，这些数据会无修改的发往对端。AT 在下列情况下，会从命令模式 进入数据模式。

- 发送*AT+CIPSEND* 设置命令成功，回复 > 之后
- 发送*AT+CIPSEND* 执行命令成功，回复 > 之后
- 发送*AT+CIPSENDL* 设置命令成功，回复 > 之后
- 发送*AT+CIPSENDEX* 设置命令成功，回复 > 之后
- 发送*AT+SAVETRANSLINK* 设置命令成功，再发送*AT+RST* 命令，模组重启之后

在数据模式下，发送+++ 命令，会从数据模式 退出，进入命令模式。

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

LwIP A Lightweight TCP/IP stack
一个轻量级的 TCP/IP 协议栈

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, ESP32-C2 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，ESP32-C2 不会重连，并提示连接断开。

更多介绍请参考*Transmission Mode Shift Diagram*。

OWE Opportunistic Wireless Encryption. OWE is a Wi-Fi standard which ensures that the communication between each pair of endpoints is protected from other endpoints.

More details are in Wikipedia.

机会性无线加密。OWE 是一种 Wi-Fi 标准，它确保每对端点之间的通信受到保护，不受其他端点的影响。

更多介绍请参考 维基百科。

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, ESP32-C2 (as client) will keep trying to reconnect until +++ is input to exit the passthrough transmission; ESP32-C2 (as server) will shutdown the old connection and listen new connection 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，ESP32-C2 作为客户端时，会不停地尝试重连，此时单独输入+++ 退出透传，则停止重连；ESP32-C2 作为服务器时，会关闭连接同时监听新的连接，此时单独输入+++ 退出透传。

更多介绍请参考*Transmission Mode Shift Diagram*。

Transmission Mode Shift Diagram

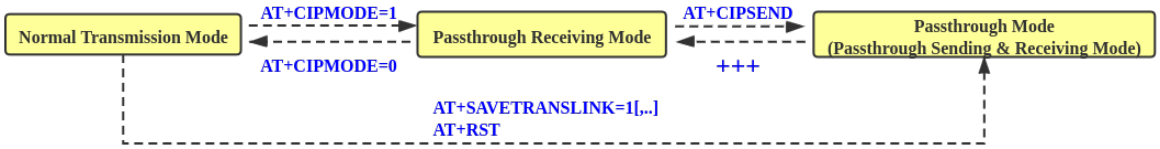


图 1: Transmission Mode Shift Diagram

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。

PKI A public key infrastructure (PKI) is a set of roles, policies, hardware, software and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption.
More details are in Public Key Infrastructure.
公开密钥基础建设。公开密钥基础建设（PKI）是一组由硬件、软件、参与者、管理政策与流程组成的基础架构，其目的在于创造、管理、分配、使用、存储以及撤销数字证书。
更多介绍请参考 公开密钥基础建设。

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

Chapter 8

关于 ESP-AT

这是 ESP-AT 的文档，ESP-AT 是乐鑫开发的可与乐鑫产品快速简单交互的解决方案。

乐鑫 Wi-Fi 和蓝牙芯片可以用作附加模块，可以完美集成在其他新产品或现有产品上，提供无线通讯功能。为降低客户开发成本，乐鑫开发了*AT 固件* 和各类*AT 命令*，方便客户简单快速地使用 AT 命令来控制芯片。

图 1: ESP-AT 方案

乐鑫提供的 AT 固件具有以下特色，利于芯片快速集成到应用中：

- 内置 TCP/IP 堆栈和数据缓冲
- 能便捷地集成到资源受限的主机平台中
- 主机对指令的回应易于解析
- 用户可自定义 AT 命令

- genindex

图 2: ESP-AT 命令

索引

G

H

I

L

M

N

O

P

Q

R

S

T

U

W