# ESP32 系列教程之七：
# ESP-AT 固件的烧录与编译

Date: 2021.7.29

Version: V1.0

# 关于文档

本文档为 ESP32 教程系列，旨在为客户进行 ESP32 系列芯片开发提供环境搭建、工程示例演示等方面的参考文档及视频演示，降低 ESP32 系列芯片、模组开发的入门难度。

ESP32 教程系列文档主要参考于乐鑫官网提供的 ESP32 入门教程：https://docs.espressif.com/projects/esp-at/en/latest/Get_Started/index.html。

由于个人经验有限，文档编写过程中难免存在失误、错漏之处，欢迎广大开发爱好者对本文档提出批评建议。

版本信息：

| 日期 | 版本 | 作者 | 说明 |
| --- | --- | --- | --- |
| 2021.7.29 | V1.0 | Amiliya | 首次发布 |

文档变更通知：以启明云端官网版本为准，恕不另行通知。

意见提交邮箱：amiliya@wireless-tag.com

# 目　录

# 1 ESP-AT 固件的烧录

## 1.1 ESP-AT 简介

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

ESP-AT 是基于 ESP-IDF 或 ESP8266_RTOS_SDK 实现的软件工程。它使 ESP 模组作为从机，MCU 作为主机。MCU 发送 AT 命令给 ESP 模组，控制 ESP 模组执行不同的操作，并接收 ESP 模组返回的 AT 响应。ESP-AT 提供了大量功能不同的 AT 命令，如 Wi-Fi 命令、TCP/IP 命令、Bluetooth LE 命令、Bluetooth 命令。

## 1.2 下载 ESP-AT 固件

ESP-AT 固件下载地址：https://www.espressif.com/zh-hans/support/download/at

## 1.3 乐鑫官方固件烧录工具

下载地址：https://www.espressif.com/zh-hans/support/download/other-tools

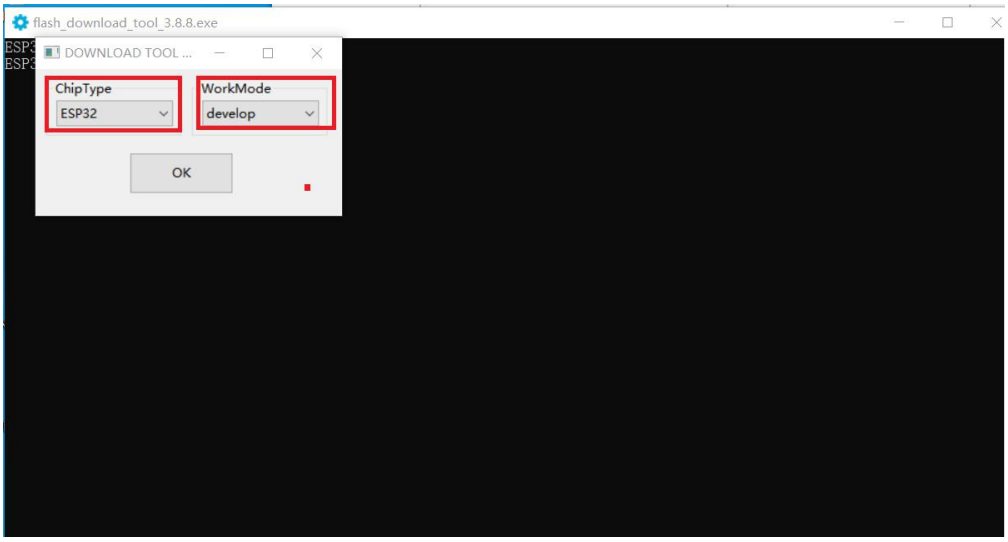

## 1.4 烧录 AT 固件

### 1.4.1 硬件连接

USB 转串口线，连接 PC 与 ESP32-WROOM 开发板

### 1.4.2 选择芯片类型和烧录模式

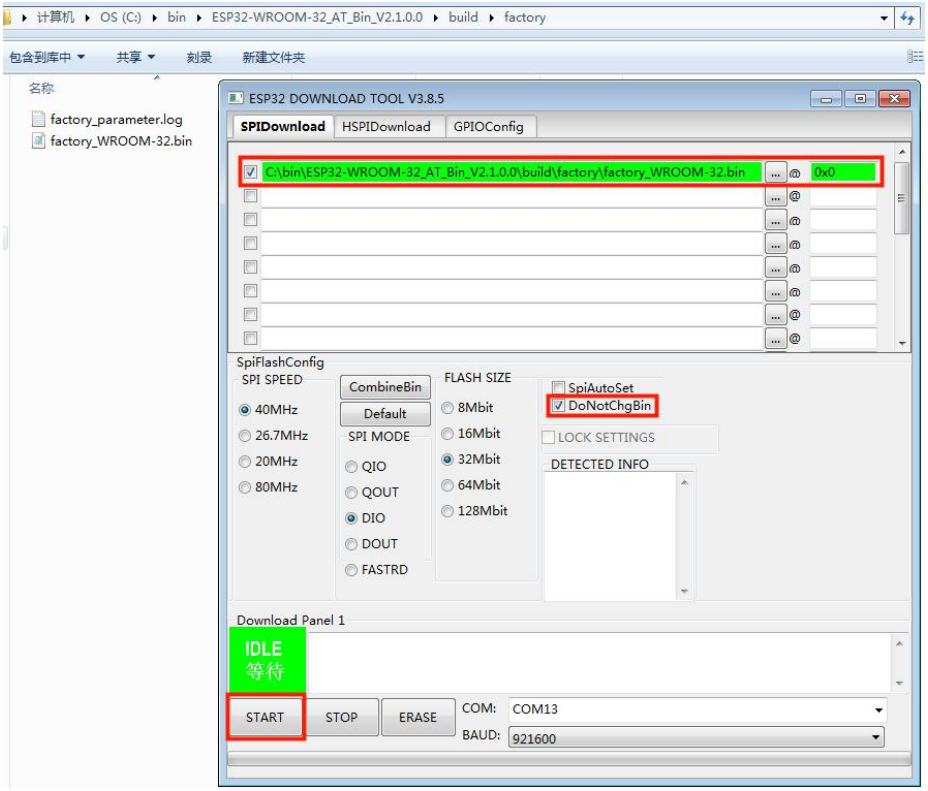
打开烧录工具，选择芯片类型和烧录模式（此处，我们选择 ESP32、Developer）



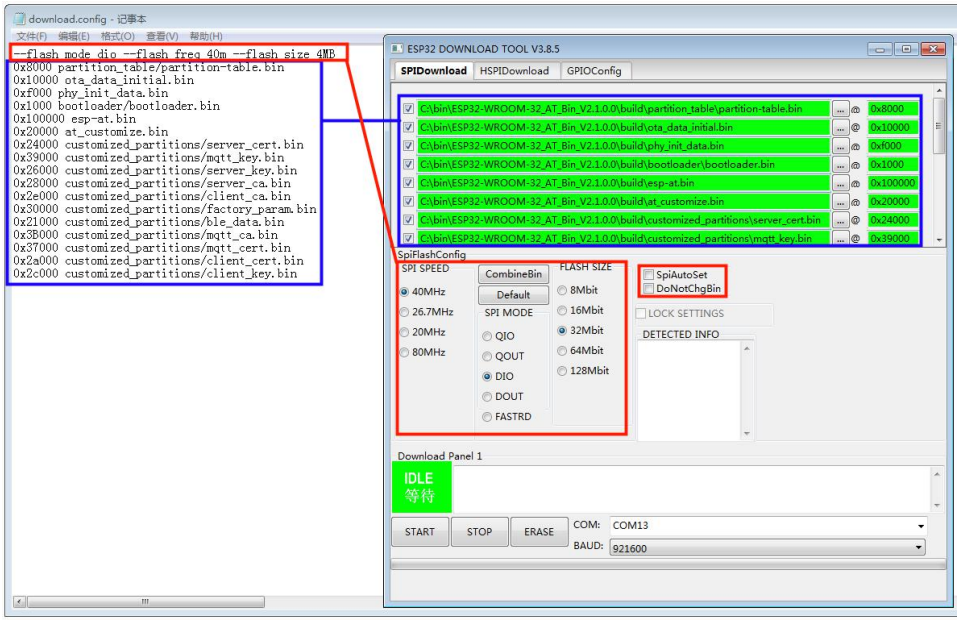

## 1.4.3 烧录固件

选择下列两种方式中的一种烧录固件：，

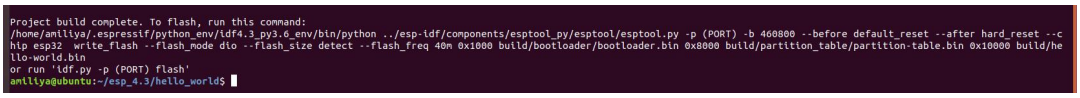
（1）直接下载打包好的量产固件（factory 文件夹下）至 0x0 地址：勾选 “DoNotChgBin”，使用量产固件的默认配置，选择端口号和波特率，开始烧录。




（2）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”：


（3）其它 ESP32 系列工程可参照上述方式进行烧录




工程编译后出现的烧录信息

# 2 编译 ESP-AT 固件

## 2.1 前提条件

对 AT 固件进行编译开发的前提条件必须是已完成 esp-idf 环境的搭建且最好能确保 hello_world 工程的正常编译与烧录。环境搭建过程可参考：ESP32 系列教程之二：Linux 搭建 esp-idf 环境。

以下 AT 固件的编译环境为 Linux 下 ESP-IDF(V4.2)环境，所使用模组型号为 ESP32 系列的 ESP32-WROOM-32D，此次编译实现功能为将官方 AT 固件的命令端口（uart1）与日志端口（uart0）合为同一端口（日志端口（uart0））。

## 2.2 克隆并编译项目

（1）克隆项目

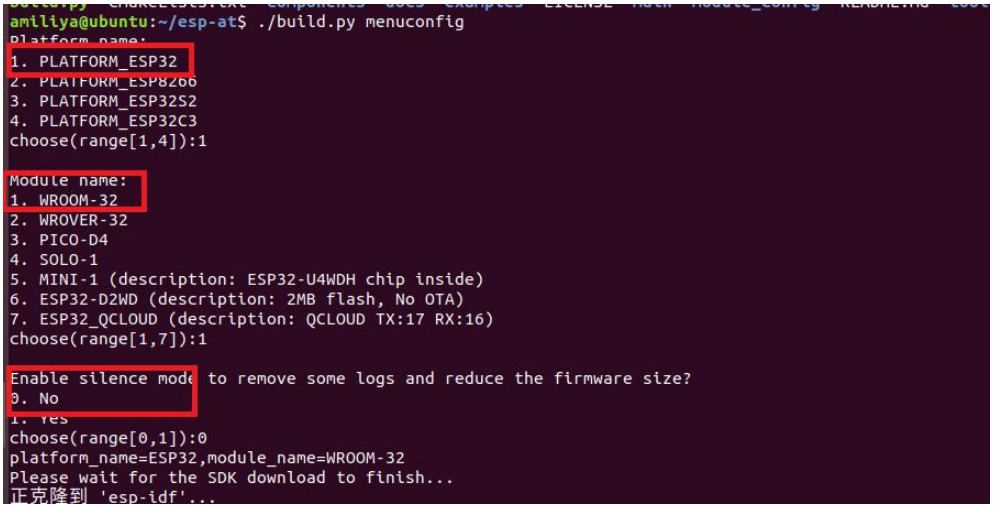
```
cd ~/esp

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

```
amiliya@ubuntu:~$ mkdir esp
amiliya@ubuntu:~$ cd esp
amiliya@ubuntu:~/esp$ git clone --recursive https://github.com.cnpmjs.org/espressif/esp-at.git
正克隆到 'esp-at'...
remote: Enumerating objects: 10117, done.
remote: Counting objects: 100% (1891/1891), done.
remote: Compressing objects: 100% (944/944), done.
remote: Total 10117 (delta 1294), reused 1473 (delta 932), pack-reused 8226
接收对象中: 100% (10117/10117), 101.97 MiB | 10.61 MiB/s, 完成.
处理 delta 中: 100% (5941/5941), 完成.
```

（2）选择芯片类型

```
cd esp-at

./build.py menuconfig
```

选择具体的芯片类型、芯片型号、是否启用 `silence mode`（一般为禁用）。

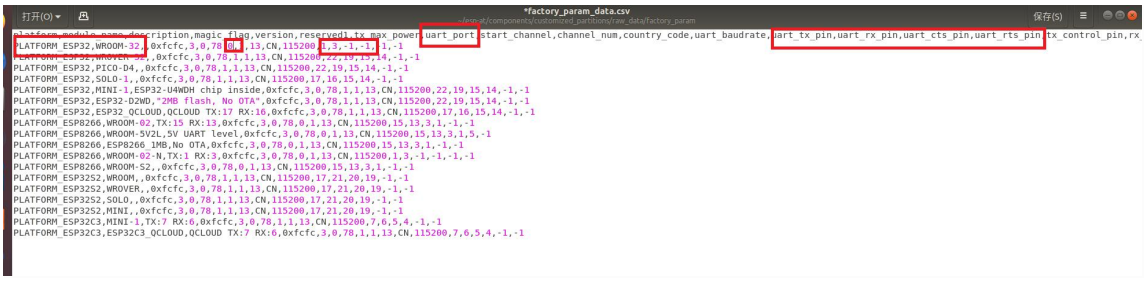



等待 esp-idf 克隆完成，若存在子模组未克隆可参考 4.2.2.2 子模组缺失

（3）修改 AT 端口引脚（可选）

在 esp-at 目录下，执行：

```
sudo gedit components/customized_partitions/raw_data/factory_param/factory_param_data.csv
```



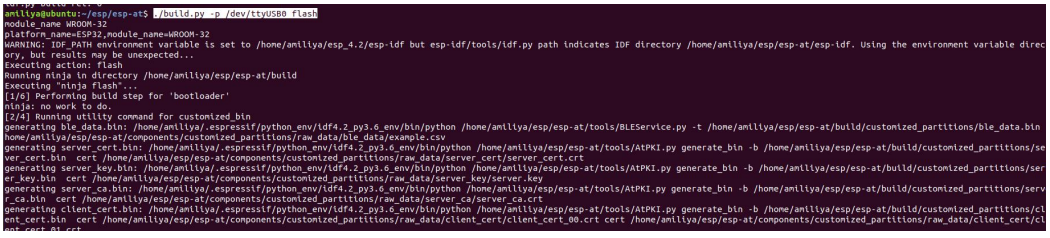

在文档中找到相对应的芯片型号、uart_port，uart_tx_pin，uart_rx_pin，uart_cts_pin,uart_rts_pin 进行修改，例如模组 ESP32-WROOM-32D 的日志端口为（uart0，TX：GPIO1，RX：GPIO3），因此将 AT 端口相对应的数字（uart_port:1;uart_tx_pin:17;uart_rx_pin:16）修改为（0；1；3）；如果不使用硬件流控制功能，则可以设置 uart_cts_pin 和 uart_rts_pin 为-1，也可以不做改变。

保存文件。

## 2.3 烧录固件

在 esp-at 目录下：

```
./build.py -p /dev/ttyUSB0 flash
```

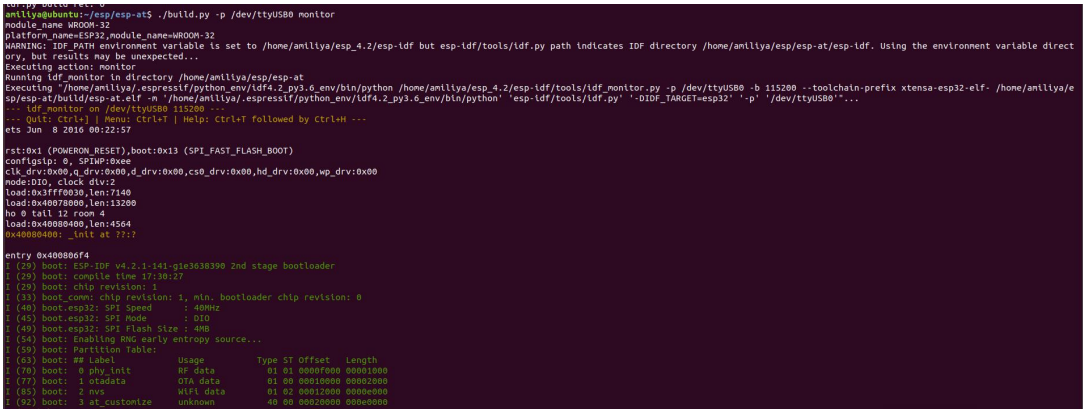


```
Hash of data verified.
Compressed 1176 bytes to 905...
Writing at 0x00028000... (100 %)
Wrote 1176 bytes (905 compressed) at 0x00028000 in 0.1 seconds (effective 128.8 kbit/s)...
Hash of data verified.
Compressed 2344 bytes to 1487...
Writing at 0x0002a000... (100 %)
Wrote 2344 bytes (1487 compressed) at 0x0002a000 in 0.1 seconds (effective 201.4 kbit/s)...
Hash of data verified.
Compressed 3368 bytes to 2526...
Writing at 0x0002c000... (100 %)
Wrote 3368 bytes (2526 compressed) at 0x0002c000 in 0.1 seconds (effective 243.3 kbit/s)...
Hash of data verified.
Compressed 2344 bytes to 1499...
Writing at 0x0002e000... (100 %)
Wrote 2344 bytes (1499 compressed) at 0x0002e000 in 0.1 seconds (effective 225.6 kbit/s)...
Hash of data verified.
Compressed 1168 bytes to 895...
Writing at 0x00037000... (100 %)
Wrote 1168 bytes (895 compressed) at 0x00037000 in 0.1 seconds (effective 121.0 kbit/s)...
Hash of data verified.
Compressed 1692 bytes to 1322...
Writing at 0x00039000... (100 %)
Wrote 1692 bytes (1322 compressed) at 0x00039000 in 0.1 seconds (effective 169.8 kbit/s)...
Hash of data verified.
Compressed 1172 bytes to 914...
Writing at 0x0003b000... (100 %)
Wrote 1172 bytes (914 compressed) at 0x0003b000 in 0.1 seconds (effective 133.4 kbit/s)...
Hash of data verified.
Compressed 4096 bytes to 80...
Writing at 0x00030000... (100 %)
Wrote 4096 bytes (80 compressed) at 0x00030000 in 0.1 seconds (effective 559.9 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
Done
idf.py build ret: 0
```

烧录成功！

## 2.4 监视工程

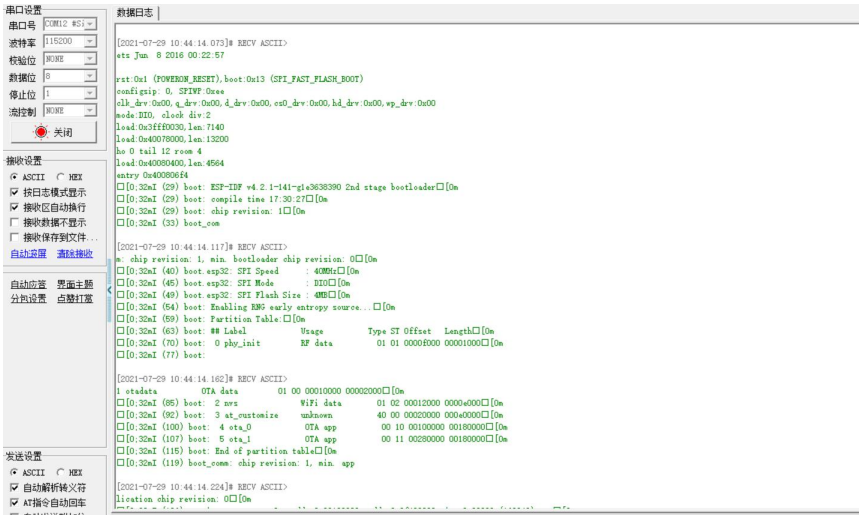
```
./build.py -p /dev/ttyUSB0 flash
```




## 2.5 AT 测试

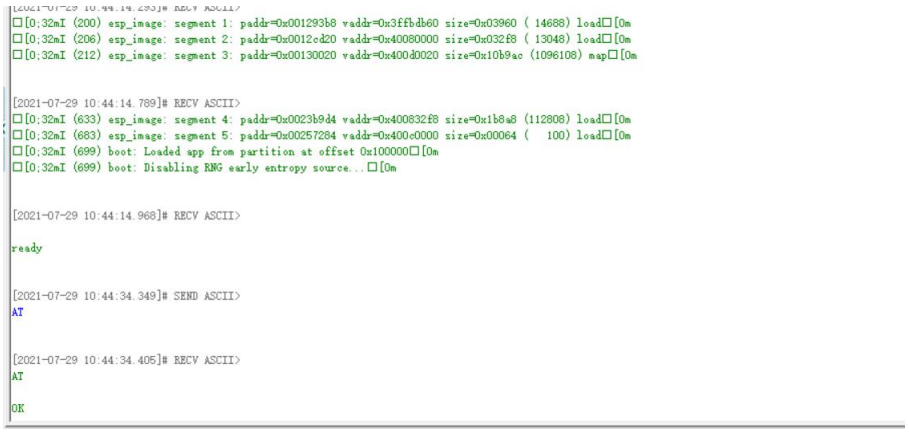
打开串口调试工具，选择对应串口号，波特率 115200（未修改）

串口工具打印以下启动信息：




```
[2021-07-29 10:44:14.162]# RECV ASCII>
1 otadata          OTA data         01 00 00010000 00002000[0m
[0;32mI (85) boot:  2 nvs              WiFi data        01 02 00012000 0000e000[0m
[0;32mI (92) boot:  3 at_customize     unknown          40 00 00020000 000e0000[0m
[0;32mI (100) boot:  4 ota_0            OTA app          00 10 00100000 00180000[0m
[0;32mI (107) boot:  5 ota_1            OTA app          00 11 00280000 00180000[0m
[0;32mI (115) boot: End of partition table[0m
[0;32mI (119) boot_comm: chip revision: 1, min. app

[2021-07-29 10:44:14.224]# RECV ASCII>
lication chip revision: 0[0m
[0;32mI (126) esp_image: segment 0: paddr=0x00100020 vaddr=0x3f400020 size=0x29390 (168848) map[0m


[2021-07-29 10:44:14.293]# RECV ASCII>
[0;32mI (200) esp_image: segment 1: paddr=0x001293b8 vaddr=0x3ffbdb60 size=0x03960 ( 14688) load[0m
[0;32mI (206) esp_image: segment 2: paddr=0x0012cd20 vaddr=0x40080000 size=0x032f8 ( 13048) load[0m
[0;32mI (212) esp_image: segment 3: paddr=0x00130020 vaddr=0x400d0020 size=0x10b9ac (1096108) map[0m


[2021-07-29 10:44:14.789]# RECV ASCII>
[0;32mI (633) esp_image: segment 4: paddr=0x0023b9d4 vaddr=0x400832f8 size=0x1b8a8 (112808) load[0m
[0;32mI (683) esp_image: segment 5: paddr=0x00257284 vaddr=0x400c0000 size=0x00064 (   100) load[0m
[0;32mI (699) boot: Loaded app from partition at offset 0x100000[0m
[0;32mI (699) boot: Disabling RNG early entropy source...[0m


[2021-07-29 10:44:14.968]# RECV ASCII>

ready
```

发送 AT 指令：




AT 命令端口修改成功！

（注：AT 命令端口与日志端口重合后会影响日志的输出打印，故非必要不建议这么做！）

# 3、 参考视频

（注：为保证视频质量，演示视频并未录制声音，同时对于一些长时间下载过程进行了缩略录制，视频仅作为本文档参考文件。）

## 3.1 烧录工具的使用

ESP32系列教程（12）：固件烧录工

## 3.2 AT 固件的编译

ESP32系列教程（13）：AT固件的编

# 4 后　记

## 4.1 相关资源

（1）《官方 AT 固件》

（2）《官方下载工具》

（3）《AT 命令集》

（4）《AT 固件的编译与开发》

（5）《ESP32 技术规格书》

（6）《ESP32 技术参考手册》

（7）《ESP32 硬件设计指南》

## 4.2 常见问题

此章节整理了一些固件下载及 AT 编译过程中的常见问题以供大家参考，其它相关问题可提交至意见提交邮箱。

### 4.2.1 固件下载

#### 4.2.1.1 硬件连接

除了开发板以外，对需要从引脚接线出来的芯片或模组下载固件时首先需要前往芯片或模组相对应的技术规格书查看相应管脚的功能，了解如何使芯片或模组处于固件下载模式。

以 ESP32-WRROM-32D 模组为例，查询 ESP32 技术规格书，存在以下信息：

配置 Strapping 管脚的详细启动模式请参阅表 3 。

**表 3: Strapping 管脚**

| 内置 LDO (VDD_SDIO) 电压 | | | |
|---|---|---|---|
| 管脚 | 默认 | 3.3 V | 1.8 V |
| MTDI | 下拉 | 0 | 1 |
| 系统启动模式 | | | |
| 管脚 | 默认 | SPI 启动模式 | 下载启动模式 |
| GPIO0 | 上拉 | 1 | 0 |
| GPIO2 | 下拉 | 无关项 | 0 |
| 系统启动过程中，控制 U0TXD 打印 | | | |
| 管脚 | 默认 | U0TXD 正常打印 | U0TXD 上电不打印 |
| MTDO | 上拉 | 1 | 0 |
| SDIO 从机信号输入输出时序 | | | |
| 管脚 | 默认 | 下降沿采样<br>下降沿输出 | 下降沿采样<br>上升沿输出 | 上升沿采样<br>下降沿输出 | 上升沿采样<br>上升沿输出 |
| MTDO | 上拉 | 0 | 0 | 1 | 1 |
| GPIO5 | 上拉 | 0 | 1 | 0 | 1 |

由此可知，使用接线方式对 ESP32-WRROM-32D 模组下载固件时除了将 VCC、GND、TX、RX 等引脚接线出来，还必须将 GPIO0 引脚接地才能让模组进入固件下载模式。其它系列芯片与此类似（如 C3 则是 IO9），可查询相应技术规格书获取相应管脚信息。

#### 4.2.1.2 烧录工具

不同类型的芯片需要在烧录工具中选择好对应的烧录模式，因此使用新类型芯片时需关注官网的烧录工具是否有更新。如只有最新版（V3.8.8）烧录工具支持 ESP32C3 的烧录。

### 4.2.2 AT 固件编译

#### 4.2.2.1 克隆 AT 项目

当使用 2.2 克隆并编译项目的克隆指令克隆下来的 AT 项目无法正常进行编译时，可以在命令中使用`-b`指定下载特定的可正常编译的版本，如：

```
git clone  -b release/v2.2.0.0_esp32 --recursive https://github.com.cnpmjs.org/espressif/esp-at.git
```

版本信息可在 `https://github.com/espressif/esp-at.git` 查看。

#### 4.2.2.2 子模组缺失

克隆 esp-idf 时可能会有子模组未克隆完全导致无法启动 menuconfig 以及无法正常编译等问题，在 esp-idf 文件夹下使用以下命令下载缺失子模组：

```
git submodule update --init --recursive
```

若命令长时间无响应可取消后重试，此命令也可用于检测子模组是否下载完全（下载完全则直接返回命令行），也可进入缺失子模组的目录下单独克隆该子模组。