ESP-FAQ

2023 年 08 月 22 日

Contents

1 使用说明 **3**

1.1 问题搜索 . . . 3

1.2 贡献指南 . . . 4

2 开发环境 **11**

2.1 IDE 插件 . . . 11

2.2 调试分析 . . . 12

2.3 环境搭建 . . . 18

2.4 固件更新 . . . 21

3 应用方案 **29**

3.1 安卓应用 . . . 29

3.2 AI 应用 . . . 31

3.3 AT . . . 33

3.4 音频应用框架 . . . 33

3.5 BLE Mesh 应用框架 . . . 40

3.6 摄像头应用方案 . . . 57

3.7 社区软件平台 . . . 64

3.8 ESP Matter . . . 64

3.9 ESP-NOW . . . 68

3.10 ESP RainMaker 云服务 . . . 71

3.11 苹果应用 . . . 78

3.12 第三方云服务 . . . 81

3.13 ESP-WIFI-MESH 应用框架 . . . 82

4 软件平台 **91**

4.1 蓝牙 . . . 91

4.2 以太网 . . . 108

4.3 共存 . . . 112
4.4 外设 . . . 114
4.5 协议 . . . 149
4.6 配置 . . . 166
4.7 安全 . . . 167
4.8 储存 . . . 172
4.9 系统 . . . 182
4.10 Wi-Fi . . . 205

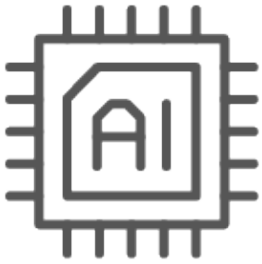
5 硬件相关 239
5.1 芯片功能对比 . . . 239
5.2 开发板使用 . . . 242
5.3 硬件设计 . . . 249
5.4 射频相关 . . . 261
5.5 工艺与 ESD 防护 . . . 264
5.6 生产测试 . . . 264

6 测试校验 267
6.1 功耗测试指南 . . . 267

7 商务常见问题 271
7.1 你们的产品通过哪些认证? . . . 271
7.2 请问贵司是否获得 ISO 质量管理体系认证? . . . 271
7.3 请问你们芯片和模组通过了 REACH、ROHS 等环保认证吗? . . . 271
7.4 你们在国内、欧洲、美国或加拿大有代理商吗? . . . 272
7.5 请问如何能成为乐鑫的代理商? . . . 272
7.6 贵司的产品信息在哪里可以查看? 量产产品有哪些? . . . 272
7.7 你们的产品有🄴期供货保证吗? . . . 272
7.8 如何查看产品的标准包装量 (SPQ) 和最小起订量 (MOQ)? . . . 272
7.9 如何购买你们的产品? . . . 272
7.10 请问批量采购价格是多少? 如何批量采购? . . . 273
7.11 请问你们各产品之间，有什么区别? 包括不同系列与型号等。 . . . 273
7.12 贵司模组本身有固件吗? 我需要定制模组/芯片烧录，请问可否在出厂时候完成? 价格如何? 交期多久? 我应如何操作? . . . 273
7.13 请问哪款产品支持 HomeKit? 如何获取 Espressif HomeKit SDK? . . . 273
7.14 请问贵司的办公地点在哪里? . . . 273
7.15 请问你们技术的联系方式是什么? . . . 274
7.16 请问如何与贵司取得联系? . . . 274
7.17 请问如何可以快速识别贵司的模组为量产产品或 NPI 新品? . . . 274

[English]

ESP-FAQ 是由乐鑫官方推出的针对常见问题的总结。在线帮助我们的用户快速检索经常问到的问题，通过简单的解释获得解答。目前常见问题的种类涵盖：开发环境、应用方案、软件平台、硬件相关和测试测试。

		
使用说明	开发环境	应用方案
		
软件平台	硬件相关	测试校验

CHAPTER 1

使用说明

[English]

本节为 ESP-FAQ 的使用说明。“问题查找”旨在帮助您快速了解该网站的搜索方法及分类框架，节省问题查找时间。同时，我们诚挚欢迎您对 ESP-FAQ 直接做出错误修复、新文档添加等优化贡献，具体操作流程可参见“文档贡献”。

1.1 问题搜索

[English]

此指南目前有以下两个部分：

- 搜索问题技巧
- 问题分类框架

1.1.1 搜索问题技巧

目前可以归纳出以下 2 种搜索问题的技巧：

- 搜索关键词
- 排除某个关键词

搜索关键词

将问题中的关键词提取出来并搜索它们，此时搜索结果会得到最匹配的结果。

比如问题为：`ESP32 的 Bluetooth LE 吞吐量是多少？`

此时搜索：`ESP32`、`BLE` 和 `吞吐量`等关键字为宜。

排除某个关键词

在搜索内容中添加一个标识符 -, 格式为：`关键词 -排除关键词`。此时搜索结果不会出现有排除关键词的结果。

比如搜索：`ESP32 -ble`。任何搜索结果中有 `ble` 的结果将会被过滤。

1.1.2 问题分类框架

在掌握上述 搜索问题技巧后，可以利用 ESP-FAQ 已经做好的分类来提取想搜索问题里的关键词并搜索。以下是此网站的框架：

- 开发环境
- 应用方案
- 软件平台
- 硬件相关
- 测试校验
- 商务问题

1.2 贡献指南

[English]

我们欢迎对 esp-faq 项目做出贡献，如修复错误，添加文档等。我们通过 Github Pull Requests 接受贡献。

1.2.1 提交流程

这一节，是对 新增问题和 修改问题两个操作的流程简要介绍，流程中涉及的环节具体要求，请点击链接查看。

针对 `git` 相关操作不做具体的介绍，可以查看 Git 相关教程。

新增问题

1. 在本地新建分支，遵循分支命名规范；
2. 在本地或者 web IDE 找到与问题类型对应的 `*.rst` 文件，根据模板格式新增问题；
3. 编辑完成后，打开预览界面查看显示结果是否符合预期，可以使用本地编译环境 编译文档，并检查生成网页是否满足；
4. 遵循提交信息规范，推送到 github 后并提交 Pull Requests；
5. 若满足上述预期，则提交合并请求；
6. 待文档所有讨论解决并成功提交 PR，即完成新增问题的流程。

修改问题

1. 在本地新建分支，遵循分支命名规范；
2. 在本地或者 web IDE 找到与问题类型对应的 `*.rst` 文件，修改期望修改的问题；
3. 编辑完成后，打开预览界面查看显示结果是否符合预期，可以使用本地编译环境 编译文档，并检查生成网页是否满足；
4. 遵循提交信息规范，推送到 github 后并提交 Pull Requests；
5. 若满足上述预期，则提交合并请求；
6. 待文档所有讨论解决并成功提交 PR，即完成修改的流程。

1.2.2 新建分支

新建分支都基于 **主分支**进行；操作时，请留意当前所在分支是否为你期望合入的分支。

操作示例:

```
git status # 查看当前分支
git checkout -b add/artificial-intelligence_camera_model # 用于新增问题 "artificial-
↪intelligence camera model"
```

1.2.3 分支命名规范

- 新增问题: `add/artificial-intelligence_{q&a}`, `{q&a}` 使用文件名的英语简要，例如新增 `artificial intelligence camera model` 问题，分支名: `add/artificial-intelligence_camera_model`。
- 修改问题: `mod/artificial-intelligence_q&a`, `q&a` 使用文件名的英语简要，例如修改 `artificial intelligence camera model` 问题，分支名: `mod/artificial-intelligence_camera_model`。

1.2.4 问题编辑规范

请按照以下格式规范规则添加或更新 Q&A：

通用规则：

- 添加新的 Q&A 时，切记需在前项问题后添加分隔符 “————————-”。

问题格式：

- 须简洁清晰地描述问题，如：
  - ESP32-S2 固件烧录时出现错误 “A fatal error occurred: Invalid head of packet (0x50)”?（**问题不清晰**）
  - ESP32-S2 固件烧录时出现错误 “A fatal error occurred: Invalid head of packet (0x50)”。如何解决?（**清晰**）
- 问题不宜过长。如描述太多，可精炼出主要的问题作为标题，并在回答的正文中详细描述问题背景及细节。

答案格式：

- 如正文中需引用代码，请使用 code 语法将其与文字隔开。
- 如某个问题的回答仅包含一句话，则使用正常段落书写即可，无需使用列表。
- 需要列举多个条目或排列顺序时，请使用列表：
  - 数字列表：有一定顺序（如操作步骤），或后文中需引用列表中的某个条目。
  - 项目符号列表：无特定顺序。
- 列表前需有介绍性文字，说明下述列表的含义或目的，且以冒号 “:” 结尾。
- 如两项条目是互为选择的关系，应使用项目符号列表罗列（非数字列表），并在段前介绍性文字中说明这二者的关系。
- 正文中（不论列表还是段落），每一行之前需空两格。
- 如某项条目后需跟注释或说明性文字，应缩进该注释，使其成为子条目。
- 关于列表中标点符号的使用，请参考 Espressif Manual of Style 中的章节 “Punctuation in Lists”。

更多关于列表格式的规则指导，请参考 Bulleted and Numbered Lists。请参阅下文文字与图片示例模版。

问题模块示例

```
--------------

乐鑫是否已有 AI 图像识别的产品？
-------------------------------------

  乐鑫已有标准开发板 ESP-EYE，主控芯片为 ESP32，可兼容 0v2640，3660，5640 等多款摄像头。
```

问题图片示例

```
--------------

curses.h: No such file or directory?
-------------------------------------------

问题截图: support ESP8266 chip, but ESP8266_RT

.. figure:: _static/application-solution/android-application/case_two_
↪kconfig_error.png
    :align: center
    :width: 900
    :height: 100

解决方法 : sudo apt-get install libncurses5-dev
```

1.2.5 本地编译环境

- 测试验证环境使用 ubuntu 或 Debian 系统，配置 python 环境为 3.7。
- 推荐使用 python 虚拟环境，或者 docker 环境。

```
# 安装 python3.7 与虚拟环境

sudo apt-get install python3.7 python3.7-venv

# 创建虚拟环境

python3.7 -m venv ~/.pyenv3_7

# 激活虚拟环境

source ~/.pyenv3_7/bin/activate

# 更新 pip

pip install --upgrade pip

# 安装 pip 组件

pip install -r docs/requirements.txt

# 编译中文版本
```

(下页继续)

(续上页)

```
cd docs/cn/ && make html && cd -

# 编译英文版本

cd docs/en/ && make html && cd -

# 退出虚拟环境

deactivate
```

1.2.6 提交信息规范

在分支上添加提交信息，以说明添加/修改/删除问题功能。每个提交都有一条消息，例如：

```
artificial-intelligence: add esp-eye support those camera models

1. esp-eye support those camera models.
```

提交信息的第一行应类似于“问题类别：添加/修复/删除/更改内容”。第一行以提交要更改的文件名的名称开头。例如：

`artificial-intelligence: esp-eye support those camera models.`

要添加有关该提交的更多详细信息，请将其放在第一行之后的提交消息中。

一个好的 git 提交消息讲述了一个为什么发生更改的故事，因此，阅读提交日志的人可以了解项目的开发。编写良好的提交信息现在看来似乎是在浪费时间，但是在将来尝试了解某些原因更改时，这对您和您的同事很有用（对我们的客户也有用）。

1.2.7 提交合并请求

一旦完成修改就可以对分支进行第一次提交，如果您需要进行更多的更改，请进行更多提交。完成您对该分支的所有提交后，提交合并请求。

我们使用 github 合并请求功能将分支合并到主分支中，步骤：

1. 将您的分支推送到 github 仓库；
2. 转到 esp-faq，然后单击“New pull request”；
3. 选择您刚创建准备合并的分支，然后填写“合并请求”详细信息。

参考：IDF 贡献代码。

提交合并请求相关规范

- Title 要求:

```
add: 简要描述
```

- Description 要求:

  分点描述该合并修改的信息。

- 示例:

Title:

```
artificial-intelligence: add esp-eye support those camera models.
```

Description:

```
1. add esp-eye support those camera models.
```

CHAPTER 2

开发环境

[English]

2.1 IDE 插件

[English]

2.1.1 Arduino IDE 如何添加 ESP32 开发板？

- 关于 Arduino-ESP32 的安装指南，请参考 arduino-ide 入门指南。
- 关于 Arduino IDE 添加开发板，请参考 arduino Cores。

2.1.2 使用 Arduino IDE 开发平台，如何读取 ESP32 出厂自带的 Wi-Fi 的 MAC 地址？

- Arduino-ESP32 开发框架为：https://github.com/espressif/arduino-esp32。
- 使用 WiFi.macAddress() 获取 ESP32 的 Wi-Fi 的 MAC 地址。
- 还可以参考 WiFiClientStaticIP 例程。

2.1.3 如何使用 Flash 下载工具将基于 Arduino 开发生成的 bin 文件烧录到 ESP32？

- 请前往 `File` -> `Preferences` -> `Show verbose output during` 勾选 `compilation`，编译成功后，会打印一条 Python 烧录命令，其中包含待烧录的 bin 文件以及对应的烧录地址。
- 在乐鑫官网的 工具 页面下载 Flash 下载工具，使用 Flash 下载工具烧录时选择 bin 文件，输入对应的烧录地址即可。

2.2 调试分析

[English]

2.2.1 ESP 设备的串口名称是什么？

串口名称通常是由操作系统指定的，不同的操作系统和设备可能会有不同的串口名称。常见如下：

- Windows 系统中串口设备名称格式是：COM*
- Linux 系统中串口设备名称格式是：/dev/ttyUSB*
- macOS 系统中串口设备名称格式是：/dev/cu.usbserial-*

2.2.2 ESP32 如何关闭默认通过 UART0 发送的调试信息？

- 一级 Bootloader 日志信息可以通过 GPIO15 接地来屏蔽。
- 二级 Bootloader 日志信息可以在 menuconfig 里的 `Bootloader config` 中进行相关配置。
- ESP-IDF 中的日志信息可以在 menuconfig 里的 `Component config` > `Log output` 中进行相关配置。

2.2.3 ESP32 如何修改默认上电 RF 校准方式？

- 上电时 RF 初始化默认采用部分校准的方案：打开 menuconfig 中 `CONFIG_ESP32_PHY_CALIBRATION_AND_DATA_STORAGE` 选项。
- 不关注上电启动时间，可修改使用上电全校准方案：关闭 menuconfig 中 `CONFIG_ESP32_PHY_CALIBRATION_AND_DATA_STORAGE` 选项。
- 建议默认使用 **部分校准**的方案，这样既可以保证上电启动的时间，也可以在业务逻辑中增加擦除 NVS 中 RF 校准信息的操作，以触发全校准的操作。

请参考 RF 校准文档 获取更多信息。

2.2.4 ESP8266 如何修改默认上电校准方式？

上电时 RF 初始化默认采用部分校准的方案。该方案中 esp_init_data_default.bin 的第 115 字节为 `0x01`，RF 初始化时间较短。如不关注上电启动时间，可修改使用上电全校准方案。

使用 NONOS SDK 及 RTOS SDK 3.0 以前的版本：

- 在 user_pre_init 或 user_rf_pre_init 函数中调用 system_phy_set_powerup_option(3)。
- 修改 phy_init_data.bin 中第 115 字节为 `0x03`。

使用 RTOS SDK 3.0 及以后版本：

- 在 menuconfig 中关闭 CONFIG_ESP_PHY_CALIBRATION_AND_DATA_STORAGE。
- 如果在 menuconfig 中开启了 CONFIG_ESP_PHY_INIT_DATA_IN_PARTITION，修改 phy_init_data.bin 中第 115 字节为 `0x03`；如果没有开启 CONFIG_ESP_PHY_INIT_DATA_IN_PARTITION，修改 phy_init_data.h 中第 115 字节为 `0x03`。

继续使用上电部分校准方案，若需在业务逻辑中增加触发全校准操作的功能：

- 使用 NONOS SDK 及 RTOS SDK 3.0 以前的版本：擦除 RF 参数区中的内容，触发全校准操作。
- 使用 RTOS SDK 3.0 及以后版本：擦除 NVS 分区中的内容，触发全校准操作。

2.2.5 ESP32 Boot 启动模式不正常如何排查？

- ESP32-WROVER 模组使用 1.8 V flash 与 PSRAM，启动状态默认为 `0x33`，下载模式 `0x23`。
- 其余模组使用 3.3 V flash 与 PSRAM，启动状态默认为 `0x13`，下载模式 `0x03`。
- 详情请参考 ESP32 系列芯片技术规格书 中的 Strapping 管脚部分。示例 `0x13` 对应如下：

管脚	GPIO12	GPIO0	GPIO2	GPIO4	GPIO15	GPIO5
电平	0	1	0	0	1	1

您也可以直接参考 Boot 模式选择文档。

2.2.6 使用 ESP32 JLINK 调试，发现会报 ERROR：No Symbols For Freertos，如何解决呢？

该错误日志不影响调试使用，解决措施可以参考 ST 论坛。

2.2.7 如何监测任务栈的剩余空间？

调用函数 `vTaskList()` 可以用于定期打印任务栈的剩余空间。详细的操作可以参考 CSDN 文档。

2.2.8 ESP32-S2 是否可以使用 JTAG 进行下载调试？

可以，详情请参考 ESP32-S2 JTAG 调试。

2.2.9 如何在不更改 menuconfig 输出级别的情况下调整日志输出？

要修改日志输出而不改变 menuconfig 的输出级别，您可以使用 `esp_log_level_set()` 函数。此函数允许您为特定模块或子系统设置日志级别，而不是更改全局日志级别。

例如，要将 network 模块的日志级别设置为 `ESP_LOG_DEBUG`，可以使用以下代码：

```
esp_log_level_set("network", ESP_LOG_DEBUG);
```

有关此功能的更多信息，请参阅 Logging library。

2.2.10 为什么 ESP8266 进入启动模式 (2,7) 并触发看🄵狗复位？

- 请确保 ESP8266 启动时，Strapping 管脚处于所需的电平。如果外部连接的外设使 Strapping 管脚进入到错误的电平，ESP8266 可能进入错误的操作模式。在无有效程序的情况下，看🄴狗计时器将复位芯片。
- 因此在设计实践中，建议仅将 Strapping 管脚用于连接高阻态外部器件的输入，这样便不会在上电时强制 Strapping 管脚为高/低电平。详情请参考 ESP8266 Boot Mode Selection。

2.2.11 ESP-WROVER-KIT 开发板 OpenOCD 错误 Error: Can't find board/esp32-wrover-kit-3.3v.cfg，如何解决？

- OpenOCD 版本为 20190313 和 20190708，请使用 `openocd -f board/esp32-wrover.cfg` 指令打开。
- OpenOCD 版本为 20191114 和 20200420（2020 以上版本），请使用 `openocd -f board/esp32-wrover-kit-3.3v.cfg` 指令打开。

2.2.12 ESP32 SPI boot 时会一直发生 RTC_WDT 复位是什么原因?

- 原因：flash 对 VDD_SDIO 上电到第一次访问之间有时间间隔要求。例如，GD 的 1.8 V Flash 要求从供电到第一次访问的时间间隔为 5 ms，而 ESP32 的时间间隔则为 1 ms 左右（XTAL 频率为 40 MHz），此时，访问 flash 会出错，接着会触发定时器看门狗或 RTC 看门狗重置，具体的重置类型取决于谁先被触发。RTC 看门狗重置的门限是 128 KB cycle，定时器看门狗重置的门限是 26 MB cycle。以 40 MHz 的 XTAL 时钟频率为例，当 RTC 慢速时钟的频率大于 192 KHz 时，会先触发 RTC 看门狗重置，反之则触发定时器看门狗重置。定时器看门狗重置时，VDD_SDIO 会持续供电，此时访问 flash 不会出现问题，芯片可以正常工作。而 RTC 看门狗重置时会停止 VDD_SDIO 供电，此时访问 flash 则会因为不满足 flash 上电到第一次访问的时间间隔而导致持续复位。
- 解决办法：当发生 RTC 看门狗重置时，VDD_SDIO 的供电停止，可以通过 VDD_SDIO 加上一个电容来保证这段时间 VDD_SDIO 的电压不会掉到 flash 能够容忍的电压以下。

2.2.13 ESP32 如何获取与解析 coredump？

- 从完整的固件中提取出 64 KB 大小的 coredump，需要先从分区表中确认 coredump 的偏移量。假设当前偏移量为 `0x3F0000`，运行如下命令读取固件：

```
python esp-idf/components/esptool_py/esptool/esptool.py -p /dev/ttyUSB* read_
↪flash 0x3f0000 0x10000  coredump.bin
```

- 使用 coredump 读取脚本将二进制的 coredump 文件转变成可读的信息。假设第一步获得的 coredump 文件为 coredump.bin，此固件对应的 elf 文件为 hello_world.elf，运行如下命令转换文件：

```
python esp-idf/components/espcoredump/espcoredump.py info_corefile -t raw -c␣
↪coredump.bin hello_world.elf
```

也可以参考 Core Dump 文档 了解更多信息。

2.2.14 ESP32、ESP8266、ESP32S2 如何做射频性能测试？

- 参见 ESP 射频测试指南。

2.2.15 Win 10 系统下识别不到设备有哪些原因？

- 请检查是否是在 Win10 Linux 虚拟子系统下识别设备。
- 如果只是在 Win10 下识别不到设备，应前往设备管理器，查看是否有对应设备，如 COM x。若没有识别到任何设备，请查看设备接线以及驱动是否正常。
- 如果是在 Linux 虚拟子系统下识别不到设备，在完成设备接线以及驱动检查后，以 VMWare 为例，前往虚拟机设置窗口里的“USB 控制器”，勾选“显示所有 USB 输入设备”。

2.2.16 ESP32 出现 Error:Core 1 paniced (Cache disabled but cache memory region accessed) 是什么原因？

问题原因：

- 在 cache 被禁用期间（例如在使用 spi_flash API 读取/写入/擦除/映射 SPI flash 的时候），发生了中断并且中断程序访问了 flash 的资源。
- 通常发生在处理程序调用了在 flash 中的程序，引用了 flash 中的常量时。值得注意的是，当在中断程序里面使用 double 类型变量时，由于 double 型变量操作的实现属于软件实现，该部分实现也被链接在了 flash 中（例如强制类型转换操作）。

解决措施：

- 给在中断中访问的函数加上 IRAM_ATTR 修饰符。
- 给在中断中访问的常量加上 DRAM_ATTR 修饰符。
- 不在中断处理程序中使用 double 类型。

您也可以参考 严重错误文档 来获取更多信息。

2.2.17 如何读取模组 Flash 型号信息？

- 乐鑫模组或芯片可通过 python 脚本 esptool 读取。

```
esptool.py --port /dev/ttyUSB* flash_id
```

2.2.18 调试 ESP-IDF 里的 Ethernet 示例，出现如下异常日志如何解决？

```
emac: Timed out waiting for PHY register 0x2 to have value 0x0243(mask␣
↪0xffff). Current value:
```

可以参考开发板的如下配置，详见开发板原理图:

- CONFIG_PHY_USE_POWER_PIN=y
- CONFIG_PHY_POWER_PIN=5

2.2.19 使用 ESP32 时出现“Brownout detector was triggered”报错，原因是什么，如何解决？

- ESP32 内置有掉电探测器，当其探测到芯片电压低于一定的预设阈值时，将重置芯片以防出现意外情况。
- 该报错信息可能会在不同场景内出现，但根本原因都在于芯片的供电电压暂时或永久性地低于掉电阈值。可通过替换电源、USB 电缆，或在模组内增加电容来解决。
- 除此之外，也可以通过配置重置掉电阈值，或禁用掉电探测功能。详细信息请参考 config-esp32-brownout-det。
- 关于 ESP32 上电、复位时序说明，详见 《ESP32 技术规格书》 。

2.2.20 导入头文件 protocol_examples_common.h 后，为什么编译时提示找不到该文件?

CHIP: ESP32

- 在工程下的 CMakeLists.txt 中添加语句 “set(EXTRA_COMPONENT_DIRS $ENV{IDF_PATH}/examples/common_components/protocol_examples_common)” 即可。
- 您也可以参考 构建系统文档 来获取更多信息。

2.2.21 使用 ESP8266 NonOS v3.0 版本的 SDK，如下报错是什么原因？

```
E:M 536    E:M 1528
```

以 E:M 开头的报错表示内存不足。

2.3 环境搭建

[English]

2.3.1 为 ESP32-S2 搭建环境时，使用 `idf.py set-target esp32s2` 指令，显示 "Error: No such command 'set-target'"，为什么？

- 因为 ESP-IDF 是从 release/v4.2 版本开始适配 ESP32-S2 的，所以如果在之前的 ESP-IDF 版本上搭建 ESP32-S2 环境，就会出现错误。例如使用指令 `idf.py set-target esp32s2` 时，会报错 "Error: No such command 'set-target'"。建议使用 ESP-IDF release/v4.2 及以后版本进行 ESP32-S2 的测试开发。更多请参考 ESP32-S2 入门指南。
- 不同 ESP-IDF 版本的 ESP 芯片支持情况，请查阅 ESP-IDF Release and SoC Compatibility

---

2.3.2 Windows 下使用 ESP-IDF Tools 2.3 工具安装 master 版本的 ESP-IDF 出现错误：Installation has failed with exit code 2，是什么原因？

此报错跟网络环境有关，代表在该网络环境下无法流畅地拉取 Github 仓库，导致电脑 SDK 下载失败。如遇到 Github 访问问题，推荐使用最新 Windows 安装工具 中的 **offline** 版本。

2.3.3 Windows 下使用 esp-idf-tools-setup-2.3.exe 搭建环境，运行 `make menuconfig` 出现如下错误：

```
-- Warning: Did not find file Compiler/-ASM Configure
-- Configuring incomplete, errors occurred!
```

出现此错误的原因是未找到编译工程。您需要将目录切换至 ESP-IDF 工程后，再运行配置、编译等指令。例如，编译 hello world 工程时，需要将目录切换至 `esp-idf/examples/get-started/hello_world`，然后再运行配置、编译等指令。

2.3.4 Windows 下使用 esp-idf-tools-setup-2.2.exe 安装过程中，出现 python 工具异常：

```
Installation has failed with exit code 1
```

1. 更新一下工具链：https://dl.espressif.com/dl/esp-idf-tools-setup-2.3.exe
2. 并且删除 idf_tools.py 中过时的选项 “–no-site-packages”

2.3.5 Windows 下安装编译环境出现 Download failed: 安全频道支持出错？

这是因为 Windows 系统已经默认不开启对 SSl3.0 的支持。

修改方法：在 控制面版中找到 Internet 选项，在其打开的窗口中选择 高级，最后在 设置中勾选 使用 SSL 3.0。

2.3.6 Windows 下执行 export.bat，提示 CMake、gdbgui 版本错误：

```
C:\Users\xxxx\.espressif\tools\cmake\3.16.4\bin
The following Python requirements are not satisfied:
gdbgui>=0.13.2.0
```

这个问题是由于上游的 gdbgui 发生了更新，从而导致与低版本的 python 不兼容。目前的解决方法是：手动修改 ESP-IDF 根目录下的 requirements.txt，找到 gdbdui 那条，修改成：gdbgui==0.13.2.0。

2.3.7 将版本从 v3.3 更新至最新版本后，使用 idf.menuconfig 及 idf.build 报错：

- 请参照 快速入门 重新搭建一下环境。
- 删除 hello_world 项目文件夹下的编译目录 build 和配置文件 sdkconfig。

2.3.8 如果同时要开发 ESP32 和 ESP8266，该怎样设置 PATH 和 IDF_PATH？

- PATH 不受影响，可以放在一起: export PATH=" $HOME/esp/xtensa-esp32-elf/bin:$HOME/esp/xtensa-lx106-elf/bin:$PATH"。
- 对于 IDF_PATH，可以在工程的 Makefile 里强制指定：

  在基于 ESP32 的工程项目里使用：IDF_PATH = $(HOME)/esp/esp-idf。在基于 ESP8266 的工程项目里使用：IDF_PATH = $(HOME)/esp/ESP8266_RTOS_SDK。

2.3.9 每一次切换项目时都需要重新调用 idf.py set-target 指令吗？

使用 idf.py build 编译项目时，target 的选择取决于：

1. 如果编译目录 build 已经生成，系统将使用上一次编译时使用的 target。该参数存储于 build 文件夹中的 CMakeCache.txt 文件内。
2. 如果还未生成编译目录，系统将检查 sdkconfig 文件，并使用其中定义的 target。
3. 如果同时存在有编译目录和 sdkconfig 文件，且其中分别定义了不同的 target，系统将报错。但该情况一般不会发生，除非在未删除编译目录的情况下手动更改了 sdkconfig 文件。
4. 如果 sdkconfig 文件或编译目录都不存在，可使用 IDF_TARGET 设置 target，作为 CMake 变量或环境变量。同样，如果该变量设置的 target 和 sdkconfig 文件或编译目录中定义的 target 不一致，系统也会报错。
5. 最后，如果上述三种途径都未定义 target，系统将使用默认值。可在 sdkconfig.defaults 中设置默认的 target 值。
6. 若未设定任何默认值，系统将使用 ESP32 进行编译。

关于是否需要多次调用 idf.py set-target：

- idf.py set-target 指令会将配置的 target 值存储于项目下的编译目录和 sdkconfig 文件中，并非存储于终端环境。因此，一旦某个项目配置完成并使用 target 编译过一次后，你切换并编译了另一项目，再次切回上一项目时，其 target 不会改变，仍为上一次为这个项目配置的值，无需再次调用 idf.py set-target 指令重设。
- 若想使项目自动编译某一默认的 target 值，请将默认值添加至项目的 sdkconfig.defaults 文件（如 CONFIG_IDF_TARGET="esp32s2"）。此后，如果项目中未存在 sdkconfig 文件和编译目录，idf.py build 将使用 sdkconfig.defaults 中定义的默认值进行编译。
- idf.py set-target 指令定义的 target 值可覆盖 sdkconfig.defaults 中配置的值。

2.3.10 如何查看当前 ESP-IDF 的版本号，是否存在记录版本号的文件？

- 命令行中获取版本号：可以通过在 IDF 环境中执行 `idf.py --version` 获取当前 IDF 版本号。
- CMake 脚本中获取版本号：可以通过变量 `${IDF_VERSION_MAJOR}.${IDF_VERSION_MINOR}.${IDF_VERSION_PATCH}` 获取当前版本号。
- 代码编译期间获取版本号：可以通过调用函数 `esp_get_idf_version` 查询，或直接使用“components/esp_common/include/esp_idf_version.h”中的版本号宏定义。

2.3.11 Windows 环境下 ESP-IDF 编译比较慢如何优化？

- 请将 ESP-IDF 源码目录以及编译器目录 `.espressif` 添加到杀毒软件的排除项。

2.3.12 是否有可以直接在 Windows 上使用的 esptool 工具？

- 可以前往 esptool ——> Releases，在下拉页面的 Asset 栏下载 Windows 版本的 esptool 工具。

2.4 固件更新

[English]

2.4.1 Host MCU 如何通过串口对 ESP32 进行烧录升级？

- 相关协议应用请参考：ESP32 串口协议；对应文档说明参见 串口协议。
- 示例实现代码请参考：esp-serial-flasher。

2.4.2 如何使用 USB 转串口工具对 ESP32 系列的模组下载固件？

USB 转串口对 ESP32 系列的模组下载固件的接线方式如下：

乐鑫模组	3V3	GND	TXD	RXD	IO0	EN
串口工具	3V3	GND	RXD	TXD	DTR	RTS

注解： ESP8266 模组需要额外将 IO15 接地。

2.4.3 macOS 与 Linux 如何烧录固件？

- 苹果系统 (macOS) 可以通过 brew 安装或 git 下载 esptool 工具烧录固件。
- Linux 系统（如 ubuntu）可以通过 apt-get 安装或 git 下载 esptool 工具烧录固件。

2.4.4 ESP32 是否支持使用 JTAG 管脚直接烧写程序？

ESP32 支持使用 JTAG 管脚 直接烧写程序，请参考文档：上传待调试的应用程序。

2.4.5 ESP_Flash_Downloader_Tool 是否支持自定义编程控制？

- ESP_Flash_Downloader_Tool GUI 工具不开源，且不支持嵌入执行脚本。
- ESP_Flash_Downloader_Tool 底层组件 esptool 开源，支持完成烧录加密等所有功能，建议基于该组件二次开发。

2.4.6 ESP32 能否通过 OTA 开启 Security Boot 功能？

- 不推荐通过 OTA 开启 Security Boot 功能，因为这样存在操作风险，并且需要多次 OTA 固件。
- Security Boot 功能存在于 Bootloader 中，需要首先更新 Bootloader 才可以启用该功能。
  1. 首先，检测目前设备的分区表是否可以存放开启 Security Boot 后的 Bootloader。
  2. 然后，更新一个支持写入 Bootloader 分区的中间固件。默认配置中无法擦写 Bootloader 分区，需要 *make menuconfig* 单独开启。
  3. 随后，将中间固件签名后 OTA 到目标设备，运行中间固件，中间固件先进行 OTA Bootloader，再 OTA 被签名的新固件。
  4. 如果在 OTA Bootloader 时出现中途断电或者断网失败重启，设备将无法启动，需要重新烧录。

2.4.7 基于 ESP-IDF v4.1 编译固件烧录到 ESP32-S2 设备的过程中遇到如下错误，该如何解决？

```
esptool.py v2.9-dev
Serial port /dev/ttyUSB0
Connecting....
Chip is ESP32S2 Beta
Features: Engineering Sample
Crystal is 40MHz
MAC: 7c:df:a1:01:b7:64
Uploading stub...
Running stub...

A fatal error occurred: Invalid head of packet (0x50)
esptool.py failed with exit code 2
```

解决方法：

如果当前使用的是 ESP32-S2 芯片而不是 ESP32-S2 Beta 芯片，需要将 ESP-IDF 升级到 v4.2 或以上。

补充说明：

- ESP-IDF v4.1 只支持 ESP32-S2 Beta，该芯片和 ESP32-S2 是不同的芯片，无法兼容。
- ESP-IDF v4.1 自带的 esptool 的版本是 v2.9-dev，也只支持 ESP32-S2 Beta。
- ESP-IDF v4.2 支持 ESP32-S2 芯片，该版本自带的 esptool 的版本是 v3.0-dev，支持 ESP32-S2。

2.4.8 如何使用 flash_download_tool 下载基于 ESP-IDF 编译的固件？

- 初次编译 ESP-IDF 工程请参考 get-started-guide。
- 以 hello-world 例程为例，运行 `idf.py build`` (支持 ESP-IDF v4.0 及以后版本，v4.0 之前版本请使用 ``make`)。编译工程后，会生成如下的 bin 文件的烧录指令提示：

```
#Project build complete. To flash, run this command:
../../../components/esptool_py/esptool/esptool.py -p (PORT) -b 921600 write_
↪flash --flash_mode dio --flash_size detect --flash_freq 40m 0x10000 build/
↪hello-world.bin  build 0x1000 build/bootloader/bootloader.bin 0x8000 build/
↪partition_table/partition-table.bin
or run 'idf.py -p PORT flash'
```

可以按照该指令提示的 bin 文件及烧录地址使用 flash_download_tool 进行烧录。

2.4.9 ESP 芯片烧录通讯协议是什么？

- ESP 烧录协议规范：Serial Protocol。
- 串口协议 Python 实现：esptool。
- 串口协议 C 语言实现：esp-serial-flasher。

2.4.10 如何对 ESP32-C3 进行固件离线烧录？

- 目前没有任何工具支持 ESP32-C3 固件离线烧录。但官方发布的 Flash 下载工具 可以直接烧录二进制固件，且支持量产烧录模式，最多支持 8 个 ESP32-C3 设备同时下载固件。
- 另外，官方也提供了用于量产生产的 治具 ，最多支持 4 个 ESP32-C3 模组同时下载固件。

2.4.11 ESP32 如何设置 Flash SPI 为 QIO 模式？

- 可前往 menuconfig，通过 `Serial flasher config` -> `Flash SPI mode` 配置端进行设置，对应 API 为 esp_image_spi_mode_t()。

2.4.12 使用 ESP8266 开发板，下载程序后，上电启动串口打印如下日志，是什么原因？

```
ets Jan  8 2013,rst cause:1, boot mode:(7,7)
waiting for host
```

- 打印 *waiting for host* 说明 Boot 模式是 SDIO 模式，表明 GPIO15 (MTDO) 被拉高，请参见 ESP8266 Boot 模式说明。

2.4.13 乐鑫模组烧录工具有哪些？

- 请前往 Flash 下载工具 下载乐鑫烧录工具。免安装 GUI 工具，仅适用于 `Windows` 环境。
- 乐鑫烧录工具 esptool 基于 *python* 编写，开放源代码，并且支持用户二次开发。

2.4.14 Flash 下载工具的工厂模式和开发者模式有什么区别？

- 工厂模式支持多通道下载，开发者模式仅支持单通道。
- 工厂模式下 bin 文件的路径是相对路径，开发者模式下的路径是绝对路径。

2.4.15 ESP32-C3 芯片可以使用 USB 进行固件下载，但在 ESP-IDF v4.3 下使用并不支持，如何使用 USB 进行固件下载？

- 需要在 ESP-IDF v4.4 以上版本下进行编译，拉取最新分支并 更新 IDF 工具 后可以正常编译并使用 USB 进行下载。使用过程请参考 usb-serial-jtag-console。

2.4.16 一拖四治具工厂模式烧写失败原因？

:CHIP: ESP32 | ESP8266 :

- 乐鑫产品启动时会通过一些发包来完成校准操作，此操作需要 3.3 V 电压并保证有 500 mA 的峰值电流。所以，在一拖多的情况下，通过连接电脑 USB 的方式来烧录时，会出现由于电脑 USB 供电不足而引起无法烧录或者烧录中断的情况，建议使用 hub 进行烧录并给 hub 供电。

2.4.17 使用 ESP32-WROVER-B 模组通过 Flash 下载工具下载 AT 固件，当完成写 Flash 后，结果显示 ERROR。但使用 ESP32-WEOVER-E 的模组下载相同的 AT 固件结果却显示正常，是什么原因？

- ESP32-WROVER-B 模组引出了 FLASH SPI 的管脚，但 ESP32-WROVER-E 模组没有引出 FLASH SPI 的管脚，请先检查 ESP32-WROVER-B 模组的 FLASH SPI 引脚是否被外部其他应用电路复用。
- ESP32-WROVER-B 的 FLASH SPI 的 CMD 引脚接 GND 会导致 Flash 无法启动，报错将打印如下日志：

```
rst:0x10 (RTCWDT_RTC_RESET),boot:0x1b (SPI_FAST_FLASH_BOOT)
flash read err, 1000
ets_main.c 371
ets Jun  8 2016 00:22:57
```

2.4.18 为什么使用 Flash 下载工具无法重新烧录已加密设备？

CHIP: ESP32 | ESP32-S2

- 当前 Flash 下载工具 不支持对已加密的设备重复加密，仅支持明文一次性加密操作。

2.4.19 基于 esptool 串口协议通过 UART 接口对 ESP32 进行刷新固件，是否可以新增一个 app 分区？

- Flash 实际的分区情况主要取决于 partition_table.bin 的数据。若可以更新 partition_table.bin，则可以重新划分 bootloader.bin、app.bin 等其他数据的存储空间，从而新增一个 app 分区。

2.4.20 使用 ESP8266 通过 Flash 下载工具下载程序固件后无程序运行日志输出，串口打印如下，是什么原因？

```
ets Jan  8
2013,rst cause:1, boot mode:(3,7)
ets_main.c
```

- 请先检查硬件接线是否正确。参见 Boot mode 接线说明。
- 请检查 bootloader.bin 的下载偏移地址是否正确，ESP8266 的 bootloader.bin 下载的偏移地址为 0x0，若此偏移地址错误将会导致 Flash 无法启动。

2.4.21 Windows7 系统 USB 驱动无法识别是什么原因？

- Windows7 系统需要手动下载并安装 USB Serial JTAG 驱动。

2.4.22 使用 ESP32-WROVER-E 模组下载程序后，上电打印日志如下，是什么原因？

```
  rst: 0x10  (RTCWDT_RTC_RESET), boot:0x37 (SPI_FLASH_BOOT)
【2020-12-11 15:51:42 049】invalrd header: 0xffffffff
  invalrd header: 0xffffffff
  invalrd header: 0xffffffff
```

- 出现如上报错日志一般情况为 GPIO12 拉高导致，ESP32-WROVER-E 模组 GPIO12 不能拉高，建议将 GPIO12 拉低测试一下。可参见 ESP32 boot log 指南。

2.4.23 使用 Flash 下载工具通过 USB 烧录 ESP32-C3 时，反复出现 8-download data fail，如何解决？

- 请先完全擦除芯片，再进行烧录
- V3.9.4 及以上版本已修复该问题

CHAPTER 3

应用方案

[English]

3.1 安卓应用

[English]

3.1.1 为什么 APP 无法扫描到 Wi-Fi 或者蓝牙信号？

- 对于 Android 6.0 之后的系统，需要申请位置权限 (android.permission.ACCESS_FINE_LOCATION)。
- 对于 Android 9.0 之后的系统，除申请位置权限外，还需打开 GPS。
- 对于 Android 12.0 之后的系统，需要申请扫描权限 (android.permission.BLUETOOTH_SCAN)。

3.1.2 为什么扫描 Wi-Fi 和蓝牙信号需要位置权限？

- 在移动设备中，当 GPS 信号不可用或不精确时，可以使用 Wi-Fi 和蓝牙信号定位用户。因此，应用程序需要首先获取位置权限，以使用这些信号来确定设备位置。
- 此外，Wi-Fi 和蓝牙信号可能也会获取附近的 Wi-Fi 网络和蓝牙设备的相关信息，如设备的 MAC 地址，用于识别特定的设备。应用程序可以通过解析 Wi-Fi 和蓝牙信息来获取用户的当前位置，Google 为了您的隐私考虑，对 Android 6.0 之后的相关 API 加入了位置权限需求。尽管一些操作系统在最新版本中引入了 MAC 地址随机化技术来保护用户的隐私，应用程序仍然需要位置权限来访问这些信号。
- 总之，为了使用 Wi-Fi 和蓝牙信号来定位用户或获取附近设备的信息，应用程序需要获得位置权限。这有助于保护用户的隐私和安全，并确保应用程序只能使用这些信号来执行预期的操作。

3.1.3 APP 需要继承第三方库中的 Application 类，但如需同时继承 MultiDexApplication 怎么办？

- 在您的 Application 类的 onCreate() 方法中调用 MultiDex.install(this) 即可。

3.1.4 APP 发送 http 请求报错是为什么？

- Android 高版本中需要使用加密请求，例如 https。若您需要发送 http 请求，在 AndroidMenifest.xml 中 application 标签下添加 android:usesCleartextTraffic=”true”即可。

3.1.5 怎么把 APP 的签名文件迁移到 pkcs12？

- keytool -importkeystore -srckeystore 源文件 -destkeystore 生成文件 -deststoretype pkcs12

3.1.6 在不安装 Android Studio 的情况下怎么查看 APP 的日志输出？

- 1. 安装 adb 工具。
- 2. 在命令终端执行下述命令。

```
pid=`adb shell ps | grep 包名 | awk '{print $2}'`
adb logcat | grep --color=auto $pid
```

3.1.7 如何在 Module 的 BuildConfig 中添加模块版本信息？

- 最新的 Android Studio 编译 Module 已经不会自动添加 VERSION_NAME 信息了，若需要该信息，可以在 Module 对应的 build.gradle 中添加以下命令：

```
android {
    defaultConfig {
        buildConfigField "String", "VERSION_NAME", "\"YOUR_MODULE_VERSION\""
    }
}
```

3.2 AI 应用

[English]

3.2.1 AI 图像识别产品可兼容哪些摄像头？

当前 ESP-EYE 主控芯片为 ESP32，可兼容 0V2640，OV3660，OV5640，OV7725 等多款摄像头。详见：esp32-camera Github。

3.2.2 ESP-WHO 支持使用 ESP-IDF 哪些版本？

请前往 ESP-WHO Github 获取最新信息。

3.2.3 请问微信小程序 ESP-EYE 有相关资料吗？

ESP-EYE demo 微信小程序的开源资料：EspEyeForWeChat。

3.2.4 esp-skainet 示例支持哪些语言呢？

目前仅支持中文和英文。

3.2.5 ESP-DL 支持哪些模型框架？

目前支持 mxnet、pytorch、tensorflow 三个平台的模型。

3.2.6 ESP-DL 支持上述三个平台 (mxnet, pytorch, and tensorflow) 的所有的模型吗？

模型中所有的算子须为 ESP-DL 所支持的算子。有关支持的算子，请参考 layer。

3.2.7 ESP-SKAINET 模型文件支持放在 SD 卡中吗？

支持放在 SD 卡中。

3.2.8 ESP-SKAINET 如何定制命令词？

定制命令词，请查看 ESP-SR GitHub。

3.2.9 如何降低 AI 语音模型的系统占用？

可以选择关闭 AEC、AE、VAD 这三个功能。

3.2.10 16 位量化模型和 8 位量化模型有什么区别？

16 位量化模型的精度更高，结果更准确。8 位量化模型更轻量化。

3.2.11 AI 语音模型如何修改麦克风通道数量？

可以在 AFE 中配置麦克风通道数和回采通道数。

3.2.12 如何拿到开发板中采集到的实际音频？

需要有 SD 卡接口，将音频文件存入到 SD 卡中。

3.2.13 有关 ESP-SR GitHub 的学习资料存放在哪里？

请参考 ESP-SR 用户指南。

3.2.14 有关 ESP-DL 的学习资料存放在哪里？

请参考 如何使用 ESP-DL 部署手势识别。

3.2.15 ESP32-S3 如何自定义英文命令词进行识别？

- 对于 MultiNet6，需要准备 `commands_en.txt` 来自定义英文命令词。对于 MultiNet5，可使用 `multinet_g2p.py` 脚本将英文命令词转换为 multinet 可以识别的音素。具体请参考 esp-sr/tool。

3.3 AT

[English]

有关 ESP-AT 的常见问题请见 ESP-AT 用户指南.

3.4 音频应用框架

[English]

3.4.1 使用 ESP-ADF 的 VOIP 功能时，手机和 ESP32 设备进行通话如何消除回音？

- 乐鑫提供基于 ESP32、ESP32-S3 芯片的回声消除 (Acoustic Echo Cancelation, AEC) 算法，可以参考 算法例程。
- 需要注意，AEC 的效果不仅仅依赖于软件参数配置和调试，还依赖于硬件设计，例如播放不能失真、录音不能有杂音以及回升参考信号没有问题等等，建议此部分设计参考乐鑫 ESP32-Lyrat-Mini 开发板 以及 ESP32-S3-Korvo-2 开发板 的设计。

3.4.2 使用 ESP32-Korvo-DU1906 开发板必须用百度云吗？

- ESP32-Korvo-DU1906 开发板例程只限于使用百度云进行测试，并且需要 Profile。
- 请联系百度云，了解相关商务接入条款，见 语音服务使用准备。

3.4.3 乐鑫官网提供的网络电话例程是否支持 RTP？

- ESP-ADF 当前默认提供的网络电话协议是基于 SIP 实现的 VoIP，协议部分有用到 RTP。
- 可使用 Espressif SDK ESP-ADF 下的 VOIP 例程。

3.4.4 ESP-ADF 中 SIP 协议是否开源？

- 目前协议未开源，以 lib 形式供外部调用。

3.4.5 ESP-ADF 例程能否实现蓝牙耳机的音量调节功能？

如：pipeline_a2dp_sink_and_hfp, pipeline_a2dp_sink_stream, pipeline_bt_sink

- 目前 ESP-ADF 还不支持 AVRCP 的调音操作，ESP-IDF release/v4.0 及以上版本已经支持了，您可以尝试使用 ESP-IDF 中的 a2dp_sink 例程 和 a2dp_source 例程 对跑。
- 后续会在 ADF 的例程中直接支持。

3.4.6 我想在 ESP32-LyraT 的 I2C 接一个传感器使用，请问有如何读取 I2C 设备数据的例程吗？

请参考 i2c 例程。

3.4.7 如何输出 32 位的 I2S 音频数据？

- 重新写一个 my_i2s_write 函数调用 i2s_write_expand，然后把 my_i2s_write 以 audio_element_set_write_cb 的形式替换 i2s_stream element 的 write 函数。

```
int my_i2s_write(audio_element_handle_t self, char *buffer, int len,␣
↪TickType_t ticks_to_wait, void *context)
{
  i2s_stream_t *i2s = (i2s_stream_t *)audio_element_getdata(self);
  size_t bytes_written = 0;
  i2s_write_expand(i2s->config.i2s_port, buffer, len, 16, 32, &bytes_written,
↪ ticks_to_wait);
  return bytes_written;
}

  i2s_stream_cfg_t i2s_writer = I2S_STREAM_CFG_DEFAULT();
  i2s_writer.type = AUDIO_STREAM_WRITER;
  i2s_writer.stack_in_ext = true;
  i2s_writer.i2s_config.sample_rate = 48000;
  i2s_writer.i2s_config.mode = I2S_MODE_MASTER | I2S_MODE_TX;
  i2s_writer.i2s_config.bits_per_sample = 32; //for cupid digital loopback
  audio_element_handle_t my_i2s = i2s_stream_init(&i2s_writer);
  audio_element_set_write_cb(my_i2s, my_i2s_write, NULL);
```

3.4.8 请问为何用 ESP-ADF 和 ESP-IDF v4.1 编译 example/get-started/play-pm3 时总是报错？

错误日志：`fatal error: audio_type_def.h: No such file or directory`

- 文件 audio_type_def.h 位于 ESP-ADF 的 esp-adf-libs 中。如果在编译过程中找不到该文件，则说明 ESP-ADF v2.4 可能未被正确检测出，特别是子模块可能尚未更新。
- 要正确检测 ESP-ADF v2.4，请按照 更新至一个稳定发布版本 中所述的步骤进行操作。
- 尝试执行以下命令并重复编译。

```
cd $ADF_PATH
git fetch
git checkout v2.4
git submodule update --init --recursive
```

3.4.9 请问官方有没有可以支持 ESP-IDF v4.4 的 ESP-ADF 版本？

ESP-ADF Release v2.4 支持 ESP-IDF v3.3，v4.1，v4.2，V4.3 和 v4.4。

3.4.10 加入 DuerOS 是否会将 ESP32-LyraT 开发板的录音功能全程占用？

目前的设计是全程占用录音数据。但是您可以通过使能 `I2S_stream` 的 `multi_output` 功能，让录音的数据通过这个通道输出到想要的地方。

3.4.11 ESP32-LyraT v4.3 不支持 DuerOS 吗？烧进去 DuerOS 固件，机器一直重启，如何解决？

- 可以将 RAM 设置为 64 M 或是自动 `Component config` -> `ESP32 Specific` -> `SPI RAM config` -> `Type of SPIRAM in use` -> `select ESP-PSRAM64`。

3.4.12 ESP-ADF 支持语音识别关键词自定义开发吗？

暂时还未开放语音训练接口，您可以直接使用免费唤醒词 “嗨，乐鑫”。如果您有定制需求，可以发送邮件至 Sales@espressif.com 咨询。

3.4.13 ESP-ADF 是否支持 ESP32-LyraTD-MSC V2.1 开发板跑 Alexa 例程？

- ESP-ADF 中还没有直接支持 Alexa 的例程，对于 Alexa 例程，请参考 esp-va-sdk。

3.4.14 ESP32 关于语音识别方面，要能本地化，能否推荐相应的开发板？

- 推荐使用 ESP32-Lyrat-Mini 开发板 或者 ESP32-S3-Korvo-2 开发板 来实现本地化。

3.4.15 ESP32 是否有同时支持 MIC 和 AUX 拾音的开发板？

- ESP32-lyraT-4.3 开发板 支持 MIC 和 AUX 拾音。

3.4.16 如何利用 ESP32-LyraT 开发板实现通话功能？

- 可参考语音通话例程 voip。

3.4.17 ESP32 系列音频开发板支持多大功率的扬声器？

- ESP32 开发板默认使用 NS4150 的 PA，一般不超过 3 W 大小。
- 如果有另外需求，可以更换 PA 设计。

3.4.18 乐鑫的语音唤醒方案对环境噪声是否有一定的要求？

- 当前乐鑫的语音方案可以满足信噪比 5 dB 以内的环境要求，对于一些固定的噪音场景还可以做到 0 dB 以内（需要针对实际产品进行优化）。

3.4.19 ESP32 的 AI 开发板上有 AUX 输入，MIC 就无法拾音了吗？

- ESP-ADF 开发框架可以选择多种方式拾音，有 MIC 输入和 Line-in。
- 拾音方式选择如下：

```
typedef enum {
  AUDIO_HAL_CODEC_MODE_ENCODE = 1, /*! <select adc */      // MIC pickup
  AUDIO_HAL_CODEC_MODE_DECODE, /*! <select dac*/
  AUDIO_HAL_CODEC_MODE_BOTH, /*! <select both adc and dac */   //  MIC +␣
↪speaker
  AUDIO_HAL_CODEC_MODE_LINE_IN, /*! <set adc channel */,             //␣
↪microphone pickup
} Audio_hal_codec_mode_t;
```

- 拾音方式配置如下：

```
audio_board_handle_t board_handle = audio_board_init();
audio_hal_ctrl_codec(board_handle->audio_hal, AUDIO_HAL_CODEC_MODE_DECODE,
↪ AUDIO_HAL_CTRL_START);      //若要 MIC 拾音，修改这个配置选项。
```

3.4.20 使用 ESP32-WROVER-B 模组 + ES8311 设计音频开发板，MCLK 时钟可选择哪些管脚？

- 硬件上 MCLK 只能使用 GPIO0、GPIO1、GPIO3 管脚，不可使用其他管脚，可阅读《ESP32 技术规格书》的 IO_MUX 表内的 CLK_OUT*，默认使用 GPIO0。
- 可参考 ESP32-LyraT-Mini 开发板的硬件原理图 设计。
- 管脚分配可参见 ESP32-LyraT-Mini V1.2 Hardware Reference。

3.4.21 ESP32-WROVER-E 模组使用一路 I2S 是否可实现同时播音和录音？

- 使用一路 I2S 可以实现同时播音和录音。可以参考 ESP32-LyraT 开发板入门指南。

3.4.22 乐鑫模块是否支持 Spotify Connect？

:CHIP: ESP32 | ESP32-S2 | ESP32-S3 :

- 当前不支持，建议考虑使用 dlna，可以达到类似的效果。

3.4.23 ESP32-Korvo-DU1906 开发板运行 korvo_du1906 例程重启，错误提示如下：Guru Meditation Error: Core 0 panic'ed (IllegalInstruction). Exception was unhandled，如何解决？

- 建议检查供电。
- 为整个系统提供电源。建议使用至少 5 V/2 A 电源适配器供电，保证供电稳定。

3.4.24 ESP-DSP fft 可以运行 4096、8192 以及更多采样吗？

- 可以，最大支持到 32 K 采样。最大值可以在 menuconfig 中配置，以 fft demo 为例，为 `idf.py menuconfig -> Component config -> DSP Library -> Maximum FFT length -> (*) 32768`。

3.4.25 ESP32 如何连接麦克风？

- 如果连接数字麦克风，可以连接 I2S 外设。
- 如果连接模拟麦克风，可以连接 ADC 外设。

3.4.26 ESP32 是否支持模拟音频或是数字音频输出？

- ESP32 支持 DAC 模拟音频输出，可以使用它播放提示音等简单音频。
- ESP32 支持 PWM 模拟音频输出，相比 DAC 效果稍好，演示代码：esp-iot-solution。
- ESP32 同时支持 I2S 数字音频输出，I2S 可配置引脚可以查看《ESP32 技术规格书》外设接口和传感器章节。

3.4.27 ESP32 芯片支持哪些音频格式？

ESP32 支持的音频格式有 MP3、AAC、FLAC、WAV、OGG、OPUS、AMR、G.711 等，可参考 ESP-ADF SDK 下的说明。

3.4.28 如何使用 ESP32 芯片解码压缩音频?

- 使用 ESP32 芯片解码压缩音频的应用可参考 esp-adf/examples/recorder 文件夹中的例程。

3.4.29 ESP-LED-Strip 对应的代码示例在哪?

- 对应的代码示例存放在 ESP-ADF 中，请参考 led_pixels 例程。

3.4.30 ESP32 是否支持在线语音识别？

支持。可参考例程: esp-adf/examples/dueros。

3.5 BLE Mesh 应用框架

[English]

3.5.1 被 Provisioner 配网到 ESP-BLE-MESH 网络中的第一个节点的单播地址是不是固定的？

`esp_ble_mesh_prov_t` 中 `prov_start_address` 的值用于设置 Provisioner 配网未配网设备的起始地址，即其首先配网的节点的单播地址。单播地址只能在初始化期间设置一次，此后不能修改。

3.5.2 手机 App 首先配置的节点的单播地址是不是固定的？

该 App 将确定单播地址，目前大多数单播地址是固定的。

3.5.3 配网过程中，认证设备共有多少种方法？提供的示例中使用了什么方法？

共有四种设备认证方法，即 No OOB、Static OOB、Output OOB 和 Input OOB。提供的示例里使用的是 No OOB。

3.5.4 配置入网前，未配网设备的广播包可以携带哪些信息？

- Device UUID
- OOB Info
- URL Hash（可选）

3.5.5 ESP-BLE-MESH 如何打印数据包？

示例使用函数 `ESP_LOG_BUFFER_HEX()` 打印信息语境，而 ESP-BLE-MESH 协议栈则使用 `bt_hex()` 打印。

3.5.6 Device UUID 可以用于设备识别吗？

可以。每个设备都有独一无二的 Device UUID，用户可以通过 Device UUID 识别设备。

3.5.7 如何知道当前 Provisioner 正在配网哪个未配网设备？

- `esp_ble_mesh_prov_t` 中 `prov_attention` 的值由 Provisioner 在配网过程中设置给未配网设备。

- 该值只能在初始化期间设置一次，此后不能修改。未配网设备加入 mesh 网络后可以用特定的方式来显示自己正在配网，比如灯光闪烁，以告知 Provisioner 其正在配网。

3.5.8 Provisioner 如何通过获取的 Composition Data 进一步配置节点？

Provisioner 通过调用 Configuration Client Model API `esp_ble_mesh_config_client_set_state()` 来进行如下配置。

- 正确设置参数 `esp_ble_mesh_cfg_client_set_state_t` 中的 `app_key_add`，将应用密钥添加到节点中。
- 正确设置参数 `esp_ble_mesh_cfg_client_set_state_t` 中的 `model_sub_add`，将订阅地址添加到节点的模型中。
- 正确设置参数 `esp_ble_mesh_cfg_client_set_state_t` 中的 `model_pub_set`，将发布地址添加到节点的模型中。

3.5.9 节点可以自己添加相应的配置吗？

本法可用于特殊情况，如测试阶段。

- 此示例展示了节点如何为自己的模型添加新的组地址。

```
esp_err_t example_add_fast_prov_group_address(uint16_t model_id, uint16_t␣
↪group_addr)
{
    const esp_ble_mesh_comp_t *comp = NULL;
    esp_ble_mesh_elem_t *element = NULL;
    esp_ble_mesh_model_t *model = NULL;
    int i, j;

    if (!ESP_BLE_MESH_ADDR_IS_GROUP(group_addr)) {
        return ESP_ERR_INVALID_ARG;
    }

    comp = esp_ble_mesh_get_composition_data();
    if (!comp) {
        return ESP_FAIL;
    }

    for (i = 0; i < comp->element_count; i++) {
        element = &comp->elements[i];
        model = esp_ble_mesh_find_sig_model(element, model_id);
        if (!model) {
            continue;
        }
        for (j = 0; j < ARRAY_SIZE(model->groups); j++) {
            if (model->groups[j] == group_addr) {
                break;
            }
        }
        if (j != ARRAY_SIZE(model->groups)) {
            ESP_LOGW(TAG, "%s: Group address already exists, element index:
↪%d", __func__, i);
            continue;
        }
        for (j = 0; j < ARRAY_SIZE(model->groups); j++) {
            if (model->groups[j] == ESP_BLE_MESH_ADDR_UNASSIGNED) {
                model->groups[j] = group_addr;
                break;
            }
        }
        if (j == ARRAY_SIZE(model->groups)) {
            ESP_LOGE(TAG, "%s: Model is full of group addresses, element␣
↪index: %d", __func__, i);
        }
    }
```

(下页继续)

(续上页)

```
    return ESP_OK;
}
```

注意：使能了节点的 NVS 存储器后，通过该方式添加的组地址以及绑定的应用密钥在设备掉电的情况下不能保存。这些配置信息只有通过 Configuration Client Model 配置时才会保存。

3.5.10 Provisioner 如何通过分组的方式控制节点？

通常而言，在 ESP-BLE-MESH 网络中实现组控制有两种方法，即组地址方法和虚拟地址方法。假设有 10 个设备，即 5 个带蓝灯的设备和 5 个带红灯的设备。

- 方案一：5 个蓝灯设备订阅一个组地址，5 个红灯设备订阅另一个组地址。Provisioner 往不同的组地址发送消息，即可实现分组控制设备。
- 方案二：5 个蓝灯设备订阅一个虚拟地址，5 个红灯设备订阅另一个虚拟地址，Provisioner 往不同的虚拟地址发送消息，即可实现分组控制设备。

3.5.11 Provisioner 如何知道网络中的某个设备是否离线？

- 节点离线通常定义为：电源故障或其他原因导致的节点无法与 mesh 网络中的其他节点正常通信的情况。
- ESP-BLE-MESH 网络中的节点间彼此不连接，它们通过广播通道进行通信。
- 此示例展示了如何通过 Provisioner 检测节点是否离线。
- 节点定期给 Provisioner 发送心跳包。如果 Provisioner 超过一定的时间未接收到心跳包，则视该节点离线。

注：心跳包的设计应该采用单包（字节数小于 11 个字节）的方式，这样收发效率会更高。

3.5.12 Provisioner 如何将节点添加至多个子网？

节点配置期间，Provisioner 可以为节点添加多个网络密钥，拥有相同网络密钥的节点属于同一子网。Provisioner 可以通过不同的网络密钥与不同子网内的节点进行通信。

3.5.13 为什么 APP 中显示的节点地址的数量比现有的节点地址更多？

每完成一次快速配网后、开始新一次快速配网前，APP 会存有上次配网的数据，因此 APP 中显示的节点地址的数量比现有的节点地址更多。

3.5.14 在 EspBleMesh App 中输入的 `count` 值有什么用途？

`count` 值为 App 提供配置的代理节点，以决定何时提前开始 Proxy 广播信息。

3.5.15 运行以下示例 fast_prov_server 的节点的 Configuration Client Model 何时开始工作？

使能了 Temporary Provisioner 功能后，Configuration Client Model 会开始工作。

3.5.16 Temporary Provisioner 功能会一直处于使能的状态吗？

节点收到打开/关闭电灯的消息后，所有节点会禁用其 Temporary Provisioner 功能并且转化为一般节点。

3.5.17 BLE MESH 打印日志 `ran out of retransmit attempts` 代表什么？

节点发送分段消息时，由于某些原因，接收端未收到完整的消息。此时，节点会重传消息。当重传次数达到最大重传数时，会出现该警告。当前最大重传数为 4。

3.5.18 BLE Mesh 打印日志 `Duplicate found in Network Message Cache` 代表什么？

当节点收到一条消息时，它会把该消息与网络缓存中存储的消息进行比较。如果在缓存中找到相同的消息，会出现该警告，这意味着之前已接受过该消息，则该消息会被丢弃。

3.5.19 BLE Mesh 打印日志 `Incomplete timer expired` 代表什么？

表示 Mesh 网络中的节点接收到了一个不完整的消息，并且在规定时间内没有接收到该消息的剩余部分。这通常是因为消息被分成了多个段 (segment)，在传输过程中丢失了其中的一部分，导致节点无法完整地接收该消息。

3.5.20 BLE Mesh 打印日志 `No free slots for new incoming segmented messages` 代表什么？

当节点没有空间来接收新的分段消息时，会出现该警告。用户可以通过配置 CONFIG_BLE_MESH_RX_SEG_MSG_COUNT 扩大空间。

3.5.21 BLE Mesh 打印日志 `No matching TX context for ack` 代表什么？

发送节点在收到一个分段 ACK 消息且没有匹配到对应的发送上下文（TX context）时，会出现该警告。这可能是因为网络中存在多个 ACK 消息。

3.5.22 BLE Mesh 打印日志 `Model not bound to AppKey 0x0000` 代表什么？

当节点发送带有模型的消息且该模型尚未绑定到索引为 0x000 的应用密钥时，会出现该警告。

3.5.23 BLE Mesh 打印日志 `Busy sending message to DST xxxx` 代表什么？

表示节点的客户端模型已将消息发送给目标节点，并且正在等待响应，用户无法将消息发送到单播地址相同的同一节点。接收到相应的响应或计时器到时后，可以发送另一条消息。

3.5.24 为什么会出现 EspBleMesh App 在快速配网期间长时间等待的情况？

快速配网期间，代理节点在配置完一个节点后会断开与 APP 的连接，待所有节点配网完成后再与 APP 重新建立连接，快速配网期间长时间等待可能是由于：

- 网络拓扑结构复杂：如果网络中节点数量较多，且拓扑结构比较复杂，Provisioner 可能需要更长的时间来扫描网络和与节点进行通信。
- 网络信号不稳定：如果网络信号不稳定，通信可能会受到干扰或丢失，从而导致 APP 等待时间变长。

- 节点响应时间较长：如果节点响应时间较长，可能会导致 Provisioner 等待超时并重新发送消息，从而导致 APP 等待时间变长。
- App 与 Provisioner 通信故障：如果 APP 与 Provisioner 之间通信故障，可能会导致 App 等待时间变长。

3.5.25 Provisoner 如何控制节点的服务器模型？

ESP-BLE-MESH 支持所有 SIG 定义的客户端模型。Provisioner 可以使用这些客户端模型控制节点的服务器模型。客户端模型分为 6 类，每类有相应的功能。

- Configuration Client Model
  - API esp_ble_mesh_config_client_get_state() 可用于获取 Configuration Server Model 的 `esp_ble_mesh_cfg_client_get_state_t` 值。
  - API esp_ble_mesh_config_client_set_state() 可用于获取 Configuration Server Model 的 `esp_ble_mesh_cfg_client_set_state_t` 值。
- Health Client Model
  - API esp_ble_mesh_health_client_get_state() 可用于获取 Health Server Model 的 `esp_ble_mesh_health_client_get_state_t` 值。
  - API esp_ble_mesh_health_client_set_state() 可用于获取 Health Server Model 的 `esp_ble_mesh_health_client_set_state_t` 值。
- Generic Client Models
  - API esp_ble_mesh_generic_client_get_state() 可用于获取 Generic Server Model 的 `esp_ble_mesh_generic_client_get_state_t` 值。
  - API esp_ble_mesh_generic_client_set_state() 可用于获取 Generic Server Model 的 `esp_ble_mesh_generic_client_set_state_t` 值。
- Lighting Client Models
  - API esp_ble_mesh_light_client_get_state() 可用于获取 Lighting Server Model 的 `esp_ble_mesh_light_client_get_state_t` 值。
  - API esp_ble_mesh_light_client_set_state() 可用于获取 Lighting Server Model 的 `esp_ble_mesh_light_client_set_state_t` 值。
- Sensor Client Models
  - API esp_ble_mesh_sensor_client_get_state() 可用于获取 Sensor Server Model 的 `esp_ble_mesh_sensor_client_get_state_t` 值。
  - API esp_ble_mesh_sensor_client_set_state() 可用于获取 Sensor Server Model 的 `esp_ble_mesh_sensor_client_set_state_t` 值。
- Time and Scenes Client Models

  - API esp_ble_mesh_time_scene_client_get_state() 可用于获取 Time and Scenes Server Model 的 `esp_ble_mesh_time_scene_client_get_state_t` 值。
  - API esp_ble_mesh_time_scene_client_set_state() 可用于获取 Time and Scenes Server Model 的 `esp_ble_mesh_time_scene_client_set_state_t` 值。

3.5.26 设备通信必须要网关吗？

- 情况 1：节点仅在 mesh 网络内通信。这种情况下，不需要网关。ESP-BLE-MESH 网络是一个泛洪的网络，网络中的消息没有固定的路径，节点与节点之间可以随意通信。
- 情况 2：如果用户想要远程控制网络，比如在到家之前打开某些节点，则需要网关。

3.5.27 Provisioner 删除网络中的节点时，需要进行哪些操作？

通常而言，Provisioner 从网络中移除节点主要涉及三个步骤：

- 首先，Provisioner 将需要移除的节点添加至“黑名单”。
- 其次，Provisioner 启动 密钥更新程序 。
- 最后，节点执行节点重置程序，切换自身身份为未配网设备。

3.5.28 在密钥更新的过程中，Provisioner 如何更新节点的网络密钥？

- 通过正确设置参数 esp_ble_mesh_cfg_client_set_state_t 中的 `net_key_update`，使用 Configuration Client Model API esp_ble_mesh_config_client_set_state()，Provisioner 更新节点的网络密钥。
- 通过正确设置参数 `esp_ble_mesh_cfg_client_set_state_t` 中的 `app_key_update`，使用 Configuration Client Model API `esp_ble_mesh_config_client_set_state()`，Provisioner 更新节点的应用密钥。

3.5.29 Provisioner 如何管理 mesh 网络中的节点？

- ESP-BLE-MESH 在示例中实现了一些基本的节点管理功能，比如 `esp_ble_mesh_store_node_info()`。

  - ESP-BLE-MESH 还提供可用于设置节点本地名称的 API esp_ble_mesh_provisioner_set_node_name() 和可用于获取节点本地名称的 API esp_ble_mesh_provisioner_get_node_name()。

3.5.30 Provisioner 想要控制节点的服务器模型时需要什么？

- Provisioner 在控制节点的服务器模型前，必须包括相应的客户端模型。
- Provisioner 应当添加本地的网络密钥和应用密钥。
  - Provisioner 调用 API `esp_ble_mesh_provisioner_add_local_net_key()` 以添加网络密钥。
  - Provisioner 调用 API `esp_ble_mesh_provisioner_add_local_app_key()` 以添加应用密钥。

- Provisioner 应当配置自己的客户端模型。

- Provisioner 调用 API esp_ble_mesh_provisioner_bind_app_key_to_local_model() 以绑定应用密钥至自己的客户端模型。

3.5.31 什么时候应该使能节点的 Relay 功能？

- 如果 mesh 网络中检测到的节点很稀疏，用户可以使能节点的 Relay 功能。
- 如果 mesh 网络中检测到的节点很密集，用户可以选择仅使能一些节点的 Relay 功能。
- 如果 mesh 网络大小未知，用户可以默认使能 Relay 功能。

3.5.32 节点包含什么样的模型？

- ESP-BLE-MESH 中，节点由一系列的模型组成，每个模型实现节点的某些功能。
- 模型分为两种，客户端模型和服务器模型。客户端模型可以获取并设置服务器模型的状态。
- 模型也可以分为 SIG 模型和自定义模型。SIG 模型的所有行为都由官方定义，而自定义模型的行为均由用户定义。

3.5.33 每个模型对应的消息格式是不是固定的？

- 消息由 opcode 和 payload 组成，通过 opcode 进行区分。
- 与模型对应的消息的类型和格式都是固定的，这意味着模型之间传输的消息是固定的。

3.5.34 节点的模型可以使用哪些函数发送消息？

- 对于客户端模型，用户可以调用 API esp_ble_mesh_client_model_send_msg() 发送消息。
- 对于服务器模型，用户可以调用 API esp_ble_mesh_server_model_send_msg() 发送消息。
- 对于发布，用户可以调用 API esp_ble_mesh_model_publish() 发布消息。

3.5.35 如何实现消息传输不丢包？

如果用户要实现消息传输不丢包，则需有应答的消息。等待应答的默认时间在 CONFIG_BLE_MESH_CLIENT_MSG_TIMEOUT 中设置。如果发送端等待应答超时，就会触发对应的超时事件。

注： API `esp_ble_mesh_client_model_send_msg()` 中可以设置应答的超时时间。如果参数 `msg_timeout` 设为 0，那么超时时间便会采用默认值（4 秒）。

3.5.36 如何发送无应答的消息？

- 对于客户端模型，用户可以调用 API `esp_ble_mesh_client_model_send_msg()` with the parameter `need_rsp` set to `false` 发送无应答消息。
- 对于服务器模型，调用 API esp_ble_mesh_server_model_send_msg() 发送的消息总是无应答的消息。

3.5.37 发送不分包消息时，最多可携带多少有效字节？

不分包消息的总有效载荷长度（可由用户设置）为 11 个八位位组，因此，如果消息的 opcode 为 2 个八位位组，则该消息可以携带 9 个八位位组的有效信息。对于 vendor 消息，由于 opcode 是 3 个八位位组，剩余的有效负载长度为 8 个八位位组。

3.5.38 什么时候应该使能节点的 Proxy 功能？

如果未配网设备将由电话配网，则未配网设备应该使能 Proxy 功能，因为当前几乎所有电话都不支持通过广播承载层发送 ESP-BLE-MESH 数据包。并且，未配网设备成功配网成为 Proxy 节点后，其会通过 GATT 承载层和广播承载层与 mesh 网络中的其他节点通信。

3.5.39 如何使用代理过滤器？

代理过滤器用于减少 Proxy Client（如手机）和 Proxy Server（如节点）之间交换的 Network PDU 的数量。另外，通过代理过滤器，Proxy Client 可以明确请求仅接收来自 Proxy Server 的某些目标地址的 mesh 消息。

3.5.40 如何实现将节点自检的信息发送出来？

推荐节点通过 Health Server Model 定期发布其自检结果。

3.5.41 Relay 节点什么时候可以中继消息？

如果要中继消息，消息需满足以下要求。

- 消息存在于 mesh 网络中。
- 消息的目的地址不是节点的单播地址。
- 消息的 TTL 值需大于 1。

3.5.42 如果一条消息分成几段，那么其他 Relay 节点是接收到一段消息就中继还是等接收到完整的数据包才中继？

Relay 节点收到其中一段消息时就中继，而非一直等到接收所有的消息。

3.5.43 设备断电后上电，如何能继续在网络中进行通讯？

在 menuconfig 中启用配置 `Store BLE Mesh Node configuration persistently`。

3.5.44 使用 Low Power 功能降低功耗的原理是什么？

- 开启无线电进行收听时，设备消耗能量。使能节点的低功耗功能后，它将在大多数时间内关闭无线电功能。
- 低功耗节点和好友节点需要合作，因此低功耗节点可以以适当或较低的频率接收消息，而无需一直收听。
- 当低功耗节点有一些新消息时，好友节点将为其存储消息。低功耗节点可以间隔固定时间轮询好友节点，以查看是否有新的消息。

3.5.45 节点间如何传输消息？

节点间传输信息的可能应用场景是，一旦烟雾警报检测到高浓度的烟雾，就会触发喷淋设备。有两种实现方法。

- 方法 1：喷淋设备订阅组地址。当烟雾警报器检测到高浓度的烟雾时，它会发布一条消息，该消息的目标地址是喷淋设备已订阅的组地址。
- 方法 2：Provisioner 可以配置喷淋设备的单播地址为烟雾报警器的地址。当检测到高浓度的烟雾时，烟雾警报器以喷淋设备的单播地址为目标地址，将消息发送到喷淋设备。

3.5.46 何时使用 IV Update 更新程序？

IV（Initialization Vector）是 BLE Mesh 网络中的一个重要参数，它用于在节点之间传递和解密消息，同时还用于识别网络中的重放攻击。当 IV 更新时，网络密钥的计算也会被更新，从而增强了网络的安全性。因此，当 IV 达到阈值时，需要更新 IV。在 BLE Mesh 网络中，IV 阈值是一个在 0 到 0xFFFF 之间的数字，它在网络初始化时被分配，通常为 0。当网络中传输的消息数量超过 IV 阈值时，就需要进行 IV 更新，IV Update 更新程序便会启用。此时，Provisioner 会向网络中的所有节点广播 IV 更新消息，然后节点会更新其 IV 和网络密钥。

3.5.47 为什么需要快速配网？

通常而言，存在少量未配网设备时，用户可以逐个配置。但是如果有大量未配网设备（比如 100 个）时，逐个配置会耗费大量时间。通过快速配网，用户可以在约 50 秒内配网 100 个未配网设备。

3.5.48 如何启用 IV Update 更新程序？

节点可以使用带有 Secure Network Beacon 的 IV Update 更新程序。

3.5.49 ESP-BLE-MESH 回调函数如何分类？

- API esp_ble_mesh_register_prov_callback() 用于注册处理配网和入网相关事件的回调函数。
- API esp_ble_mesh_register_config_client_callback() 用于注册处理 Configuration Client Model 相关事件的回调函数。
- API esp_ble_mesh_register_config_server_callback() 用于注册处理 Configuration Server Model 相关事件的回调函数。

- API esp_ble_mesh_register_health_client_callback() 用于注册处理 Health Client Model 相关事件的回调函数。
- API esp_ble_mesh_register_health_server_callback() 用于注册处理 Health Server Model 相关事件的回调函数。
- API esp_ble_mesh_register_generic_client_callback() 用于注册处理 Generic Client Models 相关事件的回调函数。
- API esp_ble_mesh_register_light_client_callback() 用于注册处理 Lighting Client Models 相关事件的回调函数。
- API esp_ble_mesh_register_sensor_client_callback() 用于注册处理 Sensor Client Model 相关事件的回调函数。
- API esp_ble_mesh_register_time_scene_client_callback() 用于注册处理 Time and Scenes Client Models 相关事件的回调函数。
- API esp_ble_mesh_register_custom_model_callback() 用于注册处理自定义模型和未实现服务器模型的相关事件的回调函数。

3.5.50 未配网设备加入 ESP-BLE-MESH 网络的流程是什么？

设备通过 Provisioner 加入 ESP-BLE-MESH 网络分为两个阶段，配网阶段和配置阶段。

- 配网阶段：为设备分配单播地址、添加网络密钥 (NetKey) 等。通过配网，设备加入 ESP-BLE-MESH 网络，身份从未配网设备变为节点。
- 配置阶段：为节点添加应用密钥 (AppKey), 并将应用密钥绑定到相应模型。配置期间，有些选项是可选的，比如为节点添加订阅地址、设置发布地址等。通过配置，该节点实际上可以向 Provisioner 发送消息，也可以接收来自 Provisioner 的消息。

3.5.51 Provisioner 的地址是否可以作为节点上报状态消息的目的地址？

Provisioner 的单播地址只能在初始化期间设置一次，此后不能更改。理论而言，只要节点知道 Provisioner 的单播地址，此地址便可用作节点上报状态消息的目的地址。节点在网络配置的过程中可以知道 Provisioner 的单播地址，因为 Provisioner 往节点发送消息时，消息的源地址就是 Provisioner 的单播地址。

订阅地址也可使用。Provisioner 订阅组地址或者虚拟地址，节点向该订阅地址发送消息。

3.5.52 如果 Provisioner 想要改变节点状态，其需满足什么条件？

- 需要有和节点的服务器模型相对应的客户端模型。
- 需要和节点有相同的、可用于加密消息的网络密钥和应用密钥。
- 需要知道节点的地址，可以是单播地址，也可以是订阅地址。

3.5.53 如何使用网络密钥和应用密钥？

- 网络密钥用于加密网络层的消息。具有相同网络密钥的节点视作在同一网络中，具有不同网络密钥的节点相互之间不能进行通信。
- 应用密钥用于加密上层传输层中的消息。如果服务器模型和客户端模型绑定的应用密钥不同，则无法实现相互通信。

3.5.54 是否可以采用固定的网络密钥或应用密钥？

- API *esp_ble_mesh_provisioner_add_local_net_key() <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/bluetooth/esp-ble-mesh.html?highlight=esp_ble_mesh_provisioner_delete_node_with_uuid#_CPPv44* 可以用来添加包含固定值或随机值的网络密钥。
- API esp_ble_mesh_provisioner_add_local_app_key() 可以用来添加包含固定值或随机值的应用密钥。

3.5.55 如何清除 ESP32 BLE 节点的组网信息？

清除节点的组网信息可以调用 esp_ble_mesh_node_local_reset()

3.5.56 如何删除某个节点的组网信息？

删除某个节点的信息可以调用 esp_ble_mesh_provisioner_delete_node_with_uuid() 或 esp_ble_mesh_provisioner_delete_node_with_addr()。

3.5.57 如果节点断电了，下次上电是否还要用手机 APP 重新组网？

可以前往 menuconfig，通过 `Component config -> Bluetooth Mesh support -> Store Bluetooth Mesh key and configuration persistently` 的选项保存配置信息，就不需要重新组网了。

3.5.58 1 号板子做 Provisioner，2、3、4 号板子做节点。组网成功后，如果 1 号板子掉电了，重新上电后还能否加入到这个 mesh 网络中？

1 号板子重新上电后，如果 net key 和 app key 没有变化，即可直接访问该网络。但是如果没有保存 mesh 网络中节点的地址，则地址将会丢失。

3.5.59 BLE_MESH 中，如果某个节点掉线了，要如何知道？

节点可以周期发布消息，你可以通过 Health model 周期发送 Heartbeat 消息，或者可以通过 vender model 周期发送自定义消息。

3.5.60 BLE_MESH 节点间如何实现以字符串的形式通信？

使用 vendor model，发送端将字符串放入 vendor message 发送，接收端接收消息后按字符串解析即可。

3.5.61 配置 ble mesh 保存节点信息时初始化 partition 失败: `BLE_MESH: Failed to init mesh partition, name ble_mesh, err 261`

如果选择 `Use a specific NVS partition for BLE Meshh` 选项，请确保 partition.csv 文件包含一个名为 `ble_mesh` 的特定分区。

3.5.62 请问如何在 provisioner 的 demo 中添加 health_mode？

进入 menuconfig，在 `Component config -> ESP BLE Mesh Support -> Support for BLE Mesh Client Models` 中勾选上 `Health Client Model`。

3.5.63 ble_mesh_fast_prov_client 当设备 provisioner 和手机当 provisioner 有什么不一样？

- ble_mesh_fast_prov_server demo 在收到 ESP_BLE_MESH_MODEL_OP_APP_KEY_ADD opcode 时，一并把 model 配置好了，而手机 Provisioner 则需要发送 ESP_BLE_MESH_MODEL_OP_MODEL_APP_BIND opcode 绑定 model APPkey，再发送 `ESP_BLE_MESH_MODEL_OP_MODEL_PUB_SET` 配置 publication。
- `ble_mesh_fast_prov_client demo` 与 `ble_mesh_fast_prov_server demo` 是我们提供的快速配网方案，实现了 100 个节点配置设备入网时间在 60 s 以内。为了实现这个功能，我们添加了一些自定义消息，用于设备间自定义信息的传递。

3.5.64 有什么工具和办法可以查看 ble_mesh 节点之间的加密消息吗？

- 数据包解密必须配置 netkey、appkey、devkey、iv index，用户可以尝试查看配置接口。
- 广播包需要 37、38、39 三通道同时抓，一般需要使用到专门的仪器。

3.5.65 app key 是否是厂家可以自己设置？Unicast address 和 app key 是否有某种关联？

app key 可以厂家自己设置，它和 Model 是绑定在一起的，和 Unicast address 没有关系。

3.5.66 如果一个节点突然掉线，那么通过 Health model 监测消息的机制，是整个 mesh 网络都要轮询的发送 Heartbeat 消息吗？

BLE MESH 网络没有建立任何连接，直接通过广播通道发送消息。用户可以向同一个节点发送心跳包进行检查。

3.5.67 主节点（代理节点）与从节点互相发送消息，可以用 client-server 模型吗？是否有提供示例？

请参见 V6.0 版本中 `ble_mesh_fast_provision/ble_mesh_fast_prov_server` 中提供的示例。

3.5.68 在 NRF 的手机 app 里，右下角“Setting”里有个“Network Key”，可以自由更改，这个修改的是指哪个 network key 呢？

- 在 NRF 的手机 app 里，右下角“Setting”里有个“Network Key”，修改它就意味着修改了 provisioner 的 Netkey，provisioner 配置其它设备入网时会把这个 netkey 分配给入网的节点。
- 如果 provisioner 拥有多个 Netkey，provisioner 在配置设备时，可以选择使用哪个 NetKey 分配给设备。provisioner 可以使用不同的 Netkey 和网络中的节点进行通讯。每个节点的 Netkey 都是 provisioner 分配的。

3.5.69 设备如何加入 BLE-Mesh 网络？

- 可以参考 ESP-BLE-MESH 快速入门。

3.5.70 Bluetooth® LE (BLE) Mesh 数据传送最大的包是多少 Bytes？

- 应用层单包最大 384 字节，底层不分包最大 11 字节。

3.5.71 能否提供通过 ESP32 BLE-Mesh 组网的例程？配置组网的 APP 可以使用什么软件？

- 可以使用例程 onoff_server，手机 APP 可以使用 nRF Mesh。
- 配网过程可参考 ESP-BLE-MESH 快速入门。

3.5.72 在 BLE-MESH 中，未配网设备默认的名称是 ESP-BLE-MESH，这个名称在哪里可以修改？

- 可以使用接口 esp_ble_mesh_set_unprovisioned_device_name(), 建议在 esp_ble_mesh_init() 后进行调用，否则还会是默认的 ESP-BLE-MESH。

3.5.73 ESP32 的 BLE-MESH 应用可以连接多少个节点设备？

- 理论上，ESP32 的 BLE-MESH 应用最大支持接入设备为 32767 个，实际应用中的接入设备数取决于内存占用情况。

3.5.74 ESP32 如何手动重置 BLE mesh 设备（不通过手机配网应用程序或配网设备）？

- 可以调用 esp_ble_mesh_node_local_reset 接口，重置 BLE Mesh 节点，擦除所有的配网信息，还需要等到重置事件到达，确认重置成功，调用后，设备需要重新配网。

3.5.75 ESP32 长时间运行 BLE MESH 程序后，发现客户端向服务器发送消息时出现分段错误，BLE MESH 打印日志 `NO multi-segment messsage contexts available`。如何解决？

- 用户可以前往 `Component config -> ESP BLE Mesh Support -> Maximum number of simultaneous outgoing segmented messages`，通过配置 BLE_MESH_TX_SEG_MSG_COUNT 来扩展空间。

3.5.76 使用 ESP32 BLE Mesh 应用，是否可以关闭网络密钥和 IV 更新？

- 不可以。网络密钥和 IV 更新必须保持开启。

3.6 摄像头应用方案

[English]

3.6.1 ESP32 系列芯片支持哪种类型的摄像头？

- 请参考 ESP32 系列支持的摄像头型号。

3.6.2 摄像头输出图像都有什么格式？

- 图像格式主要由摄像头决定，如果某个摄像头支持多个图像格式，如 RGB565、RGB888、YUV422、JPEG 等，需要通过配置摄像头的寄存器来选择输出格式。

3.6.3 摄像头支持哪些参数调整？

- 图像数据传输速度 (PCLK)、摄像头输出格式、分辨率、输出图像大小、白平衡、GAMMA 校正等摄像头自带的图像模式参数调整。

3.6.4 摄像头中 MCLK 和 PCLK 的关系是什么，两者有何区别？

- MCLK 是整个摄像头系统的主时钟，控制着整个系统的同步和频率。在摄像头芯片内部，MCLK 用于控制各个模块的时序，例如预处理器、数字信号处理器、像素数组和数据输出接口等。通常情况下，MCLK 的频率由主控芯片的系统时钟和摄像头内部的分频器共同决定，常见的频率有 6 MHz、12 MHz、24 MHz、48 MHz 等。
- PCLK 是用来控制像素输出的时钟信号。在摄像头输出图像时，每个像素的输出都需要一个时序信号，PCLK 就是用来控制这个时序的信号。具体来说，PCLK 的上升沿表示一个像素的数据已经输出，下降沿表示下一个像素的数据即将输出，这样就形成了一个像素数据的序列。
- MCLK 在摄像头内经过倍频/分频（根据摄像头配置决定）后得到 PCLK。通常情况下，PCLK 的频率是 MCLK 的一半或一半的整数倍，例如在 24 MHz 的 MCLK 下，PCLK 的频率可以为 12 MHz、6 MHz 等。

3.6.5 摄像头的 PCLK 是不是越高越好？

- 理论上，PCLK 速度越高，数据传输越快，但实际使用中，PCLK 越高也意味着对芯片的处理速度要求越高。
- 当前 ESP32 和 ESP32S2 芯片并口通信是通过 I2S 接口实现的，过高的 PCLK 会导致并口数据无法同步，出现图像抖动甚至花屏的现象。
- ESP32S3 使用独立的 LCD—CAM 接口，可以支持更高的 PCLK 频率。
- ESP32 的 PCLK 上限为 8 MHz。
- ESP32S2 的 PCLK 上限为 32 MHz。
- ESP32S3 的 PCLK 上限为 40 MHz。

3.6.6 ESP32 系列芯片支持 MIPI 接口吗？

- ESP32、ESP32S2 和 ESP32S3 均不支持，后续的芯片会支持。
- 当前 ESP32 系列的芯片支持的摄像头接口有 DVP、SPI、USB。

3.6.7 ESP32 系列芯片支持 USB2.0 接口吗？

- ESP32 和 ESP32S2 均不支持，后续的芯片会支持。

3.6.8 摄像头中 YUV/RGB 的传输速度为何会比 JPEG 慢？

- 因为 YUV/RGB 数据量比 JPEG 的数据量大。
- 例如：对于 320 × 240 的屏幕尺寸，YUV422 的输出为 153.6 K，而 JPEG 压缩后仅需约 10 K。

3.6.9 摄像头应用中，有哪些影响帧率的因素？

在摄像头应用中，影响帧率的因素主要包括：

- 分辨率：分辨率越高，每帧需要采集和传输的像素数据就越多，因此帧率就会下降。
- 图像格式：常见的图像格式包括 RGB565、RGB888、YUV422、JPEG 等，不同的图像格式在图像质量和数据压缩方面存在差异，这些差异会直接影响帧率。
- 图像处理：如果需要对每帧图像进行处理，如降噪、增强、压缩等操作，会占用更多的处理时间，降低帧率。
- 传输带宽：传输带宽越窄，每帧需要传输的数据就越少，因此帧率就会下降。
- 处理器性能：处理器性能越低，每帧需要处理的数据量就越难以承受，因此帧率就会下降。

因此，在摄像头应用中，需要根据具体的应用场景和需求，权衡这些因素，以达到最佳的帧率和图像质量。

3.6.10 摄像头运行失败如何排查？

- 无法识别摄像头型号:
- 检查管脚是否对应正确，重点关注 XCLK、SIOC、SIOD。
- XCLK 输入的时钟频率太低或摄像头供电不正常，导致摄像头无法正常运行。
- SIOC 和 SIOD 上挂载太多设备，导致轮询读到率先返回的地址 ID 不是摄像头而是其他设备。此情况建议固定摄像头 ID，以去除轮询步骤。
- 摄像头识别到了型号，没有图像显示:
- 检查摄像头数据管脚是否有信号，MCLK 是否正常输入。
- 摄像头寄存器参数配置正确。
- 摄像头图像显示不正常:
- 检查代码，查看输出格式是 RGB、YUV 还是 JPEG，是否符合接收端需要的格式。
- 尝试降低 PCLK 频率。

3.6.11 ESP32 支持传输视频流吗？

- 视频流的传输操作分为二进制传输和视频流编解码。
- 二进制传输：ESP32 支持二进制传输，此处是否支持取决于传输的网络带宽。目前 ESP32 TCP 的带宽为 20 MB/s，请参考 Wi-Fi 测试数据。
- 视频流编解码：ESP32 暂不支持视频流编解码。

3.6.12 ESP-EYE 的出厂固件在哪里？

- 请参考 ESP-EYE 的出厂固件。

3.6.13 Camera 方案相关的示例存放在哪里？

- 请参考 ESP-WHO。
- 请参考 esp-iot-solution。
- 请参考 esp-dev-kits。

3.6.14 ESP32 支持 12 位 DVP 接口的摄像头吗？

不支持，目前驱动只支持 8 位的 DVP 接口。

3.6.15 ESP32 是否支持使用不带 JEPG 编码的摄像头来获取 JPEG 图像？

- 如果摄像头本身不支持 JPEG 编码，可以参考我们提供的 esp-iot-solution/examples/camera/pic_server 例程，在 ESP32 设备上实现软件 JPEG 编码。该方法通过软件对 YUV422 或 RGB565 数据进行编码，得到 JPEG 图像。

3.6.16 ESP-EYE 上的 200 万像素的 OV2640 摄像头是否可以改成只输出 30 万像素的图片？

可以, 在初始化时通过配置 frame_size 的值来指定摄像头要输出的分辨率大小。

3.6.17 ESP32 支持全局快门的摄像头吗？

支持，目前支持的摄像头型号为 SC031GS、SC132GS，其他摄像头需要额外增加驱动支持。

3.6.18 ESP32 使用 DVP 摄像头通过 RTSP 传输 1080P 的视频可以达到多少帧？

暂未测试 1080P 的情况。目前 720P 可以达到 20 FPS.

3.6.19 ESP32-S3 只支持 MJPEG 编码，但在实现 rtsp/rtmp 推流的时候需要支持 H264/H265 格式的编码，请问是否有支持 H264/H265 格式的编码？

目前 ESP32-S3 不支持硬件加速的 H.264/H.265 编码。但是，可以使用软件编码器，例如 FFmpeg 库和 x264/x265 库，将从 OV2640 采集到的 MJPEG 帧转换为 H.264/H.265 编码帧。转换的性能取决于处理器性能，可能会影响帧率。

3.6.20 ESP32/ESP32-S3 是否有适配支持广角的摄像头？

有适配，可以参考 BF3005、OV5640 这两款摄像头。

3.6.21 ESP32-S2 从上电到显示摄像头图像需要 5 秒，是否有改善的空间？

有改善的空间，参考如下：

- 尝试去掉 `esp_camera_init()` 里的一些延时函数。
- 更改 `menuconfig -> component config -> camera configuration` 里的 sccb 的时钟频率为 400000。

3.6.22 ESP32 可以直接给 GC0308 摄像头提供 24 MHz 频率吗？

恐怕不行。经测试，ESP32 提供给 GC0308 的 XCLK 最大的稳定测试值为 20 MHz。

3.6.23 ESP32/ESP32-S3 是否支持 MMS 串流协议？

ESP32 和 ESP32-S3 本身并不直接支持 MMS 协议。MMS（Microsoft Media Server）是一种由微软开发的流媒体传输协议，主要用于 Windows Media Player 的网络流媒体播放。ESP32 和 ESP32-S3 支持的流媒体协议有 RTSP 和 SIP。如果需要将 ESP32 或 ESP32-S3 用于支持 MMS 协议的场景，可以考虑使用支持 MMS 协议的中间件或转换器。

3.6.24 使用 ESP32-S3 调试 GC2145 摄像头时，发现支持的最大分辨率为 1024x768，若是调至更大的分辨率，如 1280x720，会提示 cam_hal: EV-EOF-OVF 错误，有什么解决方法？

这种情况下，需要降低 GC2145 的 PCLK。可以尝试配置更小的 XCLK，以及调试该摄像头的 PLL 时钟系数。

3.6.25 ESP32-S3 是否支持 GB28181 协议？

ESP32-S3 本身不直接支持 GB28181 协议，但可以通过将 ESP32-S3 与外部电路和软件结合来实现该协议的支持。因为 GB28181 是一种视频监控设备之间的通信协议，可以使用 ESP32-S3 的网络功能和外部电路，例如视频编码器、音频编解码器和传感器，来实现 GB28181 的功能。同时需要进行相关的软件开发，以实现 GB28181 协议的解析和数据传输。

3.6.26 ESP32/ESP32-S2/ESP32-S3 是否有通过摄像头识别二维码的参考？

有，可以参考 ESP-WHO 里的 code recognition。

3.6.27 想为 OV5640 传感器添加 SD 卡接口和摄像头接口，但发现 ESP32 中不同外设的一些管脚存在冲突，请提供摄像头接口和 SD 卡接口的管脚。

ESP-WROVER-KIT 开发板 中有 Camera 和 SD 卡电路，可以参考 ESP-WROVER-KIT V3 入门指南的管脚配置。

3.6.28 当前适配的摄像头传感器没有适合我的需求的，能否增加一个指定型号的摄像头驱动？

可以。请先通过 技术支持 渠道与乐鑫的工程师确认需求，选定摄像头传感器的型号后，我们可以为您提供对应的摄像头传感器的驱动程序。

3.6.29 如何增加一个自定义的分辨率？

假设您需要的分辨率为 640x240，可以通过下述两种方法使用自定义分辨率：- 配置 sensor 工作在典型的分辨率 640x480 上，然后只使用其中的上半部分数据 (640x240)。- 在 esp32-camera/driver/include/sensor.h 中增加标识 FRAMESIZE_640*240，然后在 esp32-camera/driver/sensor.c 中增加该分辨率的长度与宽度的定义 {640, 240，ASPECT_RATIO_16X9}。这种方式需要 sensor 的驱动支持自定义分辨率才能正常工作。

3.6.30 如何修改摄像头传感器的寄存器配置？

假设您需要更改 OV5640 传感器的寄存器配置，可以通过下述两种方法实现：- 直接在 esp32-camera/sensors/ov5640.c 的 reset() 函数中使用 write_reg() 配置相关的寄存器。- 在应用层通过 set_reg() 函数配置相关的寄存器：

```
//初始化摄像头
esp_err_t ret = esp_camera_init(&camera_config);
sensor_t *s = esp_camera_sensor_get();
s->set_reg(s, 0xFFFA, 0xFF, 0xA1);
```

3.6.31 esp32-camera 中触发“cam_hal: EV-VSYNC-OVF”是什么原因？

这是传感器触发的帧同步信号过快导致的问题。可以按照下面的步骤进行排查：- 运行 esp-iot-solution/examples/camera/pic_server 示例。如果该示例能够正常运行，则说明该问题不是硬件问题。- 检查初始化传感器时指定的 XCLK 和分辨率的大小。分辨率变小或是 XCLK 变大，均可能导致传感器触发的帧同步信号过快。请注意，传感器使用的 XCLK 应该和当前指定的分辨率大小匹配。

3.7 社区软件平台

[English]

3.7.1 ESP32-SOLO 是否可以在 Arduino 软件上进行开发？

- 目前 ESP32-SOLO 尚不支持在 Arduino 软件上进行开发。
- 如果您非要使用 Arduino 进行构建代码，可以将 Arduino-esp32 用作 ESP-IDF 的组件 进行开发测试。

3.8 ESP Matter

[English]

3.8.1 ESP 哪些模组可以接入 Matter？

- 请参考 Espressif Matter 平台。

3.8.2 想要上手 ESP Matter，有哪些学习资料？

可以参考：- ESP Matter Github - Espressif Matter 系列博客 - Espressif Matter 方案视频 - Espressif Matter 示例视频

3.8.3 ESP Matter 是否能在 Windows 上编译开发？

- 目前无法在 Windows 上编译。同时，不建议使用虚拟机来进行 ESP Matter 开发，因为 Matter Controller 会使用蓝牙硬件，虚拟机可能会出现未知错误。建议使用 Linux 或 macOS 系统。

3.8.4 请问在哪里能找到 Matter 协议文档？

现在 Matter 的标准协议已经公开，可以在 CSA 联盟官网 中 提交申请并进行下载。

3.8.5 如何注册一个 ESP Matter 产品？

- 注册 Matter 产品需要首先成为 CSA 会员，产品在通过 Matter 认证测试之后，便可以在 CSA 中注册。

3.8.6 ESP32-C3 使用 Ubuntu 虚拟机进行 ESP Matter 开发时，发现按照 Matter 官方教程配网不成功，可能是什么原因？

不建议使用虚拟机来进行 ESP Matter 开发，因为 Matter Controller 会使用蓝牙硬件，虚拟机可能会出现未知错误。建议直接使用 Ubuntu 20.04 LTS 及以上版本的主机。

3.8.7 如何申请 Matter DAC?

有以下三种方式申请 Matter DAC:

- 与已成立的 PKI 提供商合作：许多组织已经拥有一个证书颁发机构，他们依赖该证书颁发机构来获取其公钥基础结构证书。在某些情况下，可以从此类 PKI 提供商处获得设备证明证书 (DAC)。
- 使用您自己的设备证明 PKI：许多组织已经拥有用于执行代码签名、现有设备认证要求或其他依赖非对称加密的任务的公钥基础设施，公钥通过数字证书分发。
- 与您的平台供应商（Espressif）合作：平台供应商可能使用他们的 VID/PID 在芯片或平台模块中嵌入了 DAC。CD 用于使用 dac_origin_vid/dac_origin_pid 字段重新映射 VID/PID。

3.8.8 ESP Matter 是否有测试工具/测试应用程序?

- 有，推荐使用 chip-tool。使用示例见 配置测试 chip tool。

3.8.9 Matter 配网过程中需要用到 DCL，请问 DCL 的具体功能是什么？

- Matter DCL 是一种基于区块链技术的，安全加密的分布式存储网络，允许连接标准联盟 (CSA) 和授权供应商发布其 Matter 设备信息，并允许 Matter 生态通过 DCL 客户端查询相关信息。
- 简化来说，Matter DCL 会用于设备验证及 OTA。

3.8.10 我们的产品是基于 Zigbee 的，请问该如何通过 ESP 芯片来接入 Matter？

- 基于 ZigBee 技术实现的设备并不是 Matter 标准的设备，此时需要通过 Matter Bridge 设备桥接 ZigBee 设备来接入 Matter 网络。
- Matter Bridge 设备可以使用一颗乐鑫的 Wi-Fi 芯片 + 802.15.4 芯片实现。Matter Bridge For BLE Mesh 设备可以使用一颗乐鑫的 Wi-Fi 芯片 + BLE 芯片实现，或者只用一颗 Wi-Fi + BLE combo 芯片实现。

3.8.11 Matter 是否可以对接三星的 smartthings？

- 可以对接，请参考 配置测试 smartthings 官方博客。

3.8.12 能否使用 Amazon/Google/Apple 的语音设备远程控制支持 Matter 的 ESP 设备？语音设备是否需要支持 Matter 协议？（比如：语音“Turn off the light”可以关掉家里的灯）

- 使用支持 Matter 协议的 Amazon/Google/Apple 语音设备，可以实现在手机上远程控制支持 Matter 的设备。另外，对于其他的生态系统，如果也支持 Matter 协议的生态系统，那这个生态系统的音箱等家庭中枢设备也能实现远程控制支持 Matter 的设备。
- 具体实践步骤为：基于 esp-matter SDK 来搭建 Matter 应用场景进行测试。- Google Matter 测试方法 - Apple Matter 测试方法

3.8.13 在提交 Matter 认证之前，产品是否需要先通过 Wi-Fi 认证和蓝牙 BQB 认证？

- 需要。Matter 是一种运行在 Wi-Fi、以太网、Thread 和蓝牙等其他技术上的协议。提交 Matter 认证前，要求设备已经通过传输层协议的认证。不仅要通过原有的 Wi-Fi 或 Thread 认证，并且基于 Matter 需要使用蓝牙来配网的规定，还需要过蓝牙技术联盟的 BQB 认证。

3.8.14 请问 ESP Matter 模组预先导入的 DAC (Device Attestation Certificate) 是存储在哪里的？

- ESP Matter 模组预先导入的 DAC (Device Attestation Certificate) 存储在 flash 中。在 Matter Pre-Provisioning 服务中，Matter DAC 证书预烧录在 esp_secure_cert 分区。将此分区添加至分区表中的示例如下:

```
# ESP-IDF Partition Table
# Name,          Type, SubType, Offset,  Size, Flags
esp_secure_cert, 0x3F,     ,     0xd000,  0x2000,  , # Never mark this as an␣
↪encrypted partition
```

3.8.15 ESP32 Matter 设备可以通过 BLE 来配置 Wi-Fi 吗？

- ESP32 Matter 设备可以通过 BLE 来配置 Wi-Fi，我们 esp-matter SDK 下的所有应用测试例程都是通过 BLE 进行配置的。可参考章节 2.2 Commissioning and Control 说明。

3.9 ESP-NOW

[English]

3.9.1 ESP32 ESP-Now 模式下一对一的通信速率是多少？

测试数据如下：

- 测试样板：ESP32-DevKitC V4。
- Wi-Fi 模式：station 模式。
- 物理层速率默认为 1 Mbps。
- open 环境下大约是 214 Kbps。
- 屏蔽箱内测试大约是 555 Kbps。
- 如要求更高速率，可以通过 esp_wifi_config_espnow_rate 进行配置。

3.9.2 ESP-NOW 是什么？它有哪些优势与适用场景？

- ESP-NOW 是由乐鑫定义的无连接通信协议。
- 在 ESP-NOW 中，应用程序数据被封装在各个供应商的动作帧中，在无连接的情况下，从一个 Wi-Fi 设备传输到另一个 Wi-Fi 设备。
- ESP-NOW 广泛应用于智能照明、远程控制、传感器等领域。

3.9.3 ESP-NOW 是否可以与 Wi-Fi 一起使用？

- 可以，但需要注意的是 ESP-NOW 的信道要和所连接的 AP 的信道相同。

3.9.4 如何设置 ESP-NOW 数据的发送速率？

使用 esp_wifi_config_espnow_rate() 函数进行配置即可，例如 `esp_wifi_config_espnow_rate(WIFI_IF_STA, WIFI_PHY_RATE_MCS0_LGI)`。

3.9.5 ESP-NOW 配对限制最多 20 个设备，是否有办法控制更多的设备？

使用广播包进行控制即可，目的地址包含在 payload 中，不受配对数量限制。仅需配置正确的广播地址即可。

3.9.6 ESP-NOW 最多可以控制多少个设备？

这取决于具体的通信方式：

- 如使用单播包，支持同时最多配对并控制 20 个设备。
- 如使用 ESP-NOW 加密模式，支持同时最多配对并控制 6 个设备。
- 如使用广播包，仅需配置正确的广播地址即可。控制设备的数量理论上没有上限，但需考虑设备过多时的干扰问题。

3.9.7 ESP-NOW 设备间通信需要连接路由器吗？

ESP-NOW 的交互方式为直接从设备到设备进行通信，不需要通过路由器来转发数据。

3.9.8 为什么将 ESP-NOW 每笔包的最大数据长度限制为 250 字节，这个数值可以修改吗？

- 最长长度目前不能修改。因为 ESP-NOW 使用动作帧中的供应商特定元素字段传输数据，802.11 协议规定一个供应商特定元素中的 “长度 “字段只有 1 个字节 (0xff = 255)，因此限制了正文部分 ESP-NOW 数据长度，最长为 250 字节。
- 或者您可以使用 API `esp_wifi_80211_tx()` 发送数据，使用 sniffer 模式接收数据。这样同样可以实现只工作在 Wi-Fi 层并且不使用 TCP/IP 协议栈。

3.9.9 使用 ESP-NOW 应用时，需要注意什么？

- 连接 Wi-Fi 以后不能再切换信道，只能在当前 Wi-Fi 信道收发数据。
- 如果设备进入 Modem-sleep 模式，将无法接受来自 ESP-NOW 的数据。

3.9.10 使用 ESP-NOW 方案时，如何降低功耗？

- 可使用如下方式来降低功耗：
  - 若使用 ESP-IDF v5.0 以下版本的 SDK，可以在未连接 AP 的时候，通过 esp_now_set_wake_window() 和 esp_wifi_set_connectionless_wake_interval() 这两个函数设置唤醒的窗口和间隔，节省功耗。
  - 若使用 ESP-IDF v5.0 版本或者最新的 master 版本，函数名称和含义有变化，可以在连接 AP 或者在未连接 AP 的时候，使用 esp_now_set_wake_window() 和 esp_wifi_connectionless_module_set_wake_interval() 这两个函数来设置醒来的窗口和间隔。
- 注意，需要在应用层设计时考虑发送端和接收端窗口同步的问题。芯片会在每个间隔醒来并工作设置好的窗口时间。此时，还需额外在 sdkconfig.defaults 里配置 CONFIG_ESP_WIFI_STA_DISCONNECTED_PM_ENABLE=y。

3.9.11 一对多，多对多通信除了 ESP-NOW 的设备无线通信方式，还有其他更好的方式吗？

也可以使用 SoftAP + Station 的方式实现。主设备使用 Wi-Fi SoftAP 模式，同时与多个从设备（Wi-Fi Station）建立连接。

3.9.12 ESP-NOW 应用是否支持通过每个 Wi-Fi 信道发送数据包？

支持，请参考 ESP-NOW 文档。

3.9.13 将 ESP-NOW 技术用于商业用途是否需要任何特殊程序？可以提供关于 ESP-NOW 技术的详细技术文档吗？为了评估无线通信质量，我想了解以下内容，例如 CSMA/CA、调制方式、比特率等。

- ESP-NOW 申请不需要任何特殊程序。
- 技术文档请阅读 ESP-NOW 用户指南，您可以使用 ESP-NOW SDK 示例进行测试。

- 默认的 ESP-NOW 比特率是 1 Mbps。

3.9.14 为什么使用 ESP32 测试 esp-idf/examples/wifi/espnow 例程，最多仅支持连接 7 个加密设备？

- 在 esp-now 应用中，ESP32 支持连接加密设备的数量不超过 17 个，默认值是 7 。请参见 “添加配对设备” 说明。
- 如果想要修改加密设备的数量，在 WiFi menuconfig 设置 `CONFIG_ESP_WIFI_ESPNOW_MAX_ENCRYPT_NUM`。

3.10 ESP RainMaker 云服务

[English]

3.10.1 ESP RainMaker 有哪些资料可供方案评估与验证？

请参考如下链接：

- 方案预览
- 入门开发指南
- API 指南
- RainMaker 设备固件开发仓库
- RainMaker Android App 开发仓库
- RainMaker iOS App 开发仓库
- ESP RainMaker 部署资料

3.10.2 ESP RainMaker 对接了哪些语音平台？支持哪些语音指令？

- 目前 ESP RainMaker 通过云云对接的形式完成了对 Alexa App 和 Google Home App 的支持，通过 ESP HomeKit SDK 对接了苹果的家庭 App。在上述 App 中添加设备后，即可使用对应的语音助手完成语音控制。建议使用 homekit_switch 进行测试，该 Demo 将已开关的产品形态出现在上述所有 App 中。
- Alexa 部分请参考 Alexa 所支持的语音指令。
- Google Assistant 部分请参考 Google Assistant 所支持的语音指令。
- Siri 部分请参考 Siri 所支持的语音指令。

3.10.3 RainMaker 设备的证书如何获取？是否有管理后台？

- 在方案验证阶段，您可以使用 Claiming 服务 获取证书，RainMaker 设备固件开发仓库 中提供的 Demo 都默认启用了该服务。在量产阶段，您需要首先完成私有化部署，可以咨询 乐鑫商务 来获取更为详细的批量生成证书指导及部署事宜。
- 目前 ESP RainMaker 方案提供了一个 在线网页 以查看设备信息，并支持通过该网页下发 OTA 任务及查看 ESP insight 数据。

3.10.4 ESP RainMaker 固件侧代码开发与腾讯云、阿里云的开发模式有什么不同？

- 腾讯云、阿里云通常是在控制台创建产品，随后创建设备模型；而 RainMaker 则在设备侧通过 API 创建模型，连接网络后将模型发送给云。
- 腾讯云、阿里云的连接认证阶段通常有 2 种形式：使用证书/三元组（四元组）或使用动态注册。ESP RainMaker 仅支持证书的形式。
- RainMaker 设备固件开发 SDK 基于成熟的 ESP-IDF 开发框架，对配网、OTA 升级、本地控制等功能做了整合，进行基本的配置即可使用。您不需要移植其他第三方 SDK，直接编写应用层代码即可，该部分完全开源。
- RainMaker 云中间件基于 AWS 无服务器计算 (Amazon Serverless Computing) 构建，开发者无需在云中编写任何代码、也无需进行任何配置即可使用，该部分不开源。
- RainMaker App 提供 iOS 与 Android 两个版本，App 可根据设备上报的配置自动加载 UI 及图标，同时也将完整展示设备的物模型，用户可以直接在 App 上更改这些属性。App 将协助用户完成设备的网络配置、本地发现、创建定时任务，该部分完全开源。

3.10.5 ESP RainMaker 公有云与 ESP RainMaker 私有云有什么区别？

- ESP RainMaker 是乐鑫提供的一套连接 AWS 的方案，该方案利用了 AWS 提供的一些云产品实现了设备接入、管理、维护、分析功能，乐鑫针对这个方案提供了 App、固件侧的源代码。同时为了方便您测试体验，我们部署了一套公共服务，任何开发者都可以免费使用，我们称之为 ESP RainMaker 公有云。在没有特别申明的情况下，此页面中的 ESP RainMaker 均代指公有云。
- 如果您想将 ESP RainMaker 部署在自己的 AWS 账号上，首先您需联系 乐鑫商务 以开放相关服务的访问权限，然后可参考 ESP RainMaker 部署资料 完成私有化部署。在完成私有化部署后，您将拥有自己的账户体系及管理后台。此时，我们称之为 ESP RainMaker 私有云。

3.10.6 Nova Home 跟 ESP RainMaker 有什么关系？

- Nova Home 与公有 ESP RainMaker 所使用的服务器完全一致，账户互通，不同点在于 App 端。
- Nova Home App 相对于 ESP RainMaker App，对 UI 界面及图标做了相关优化，增加了一些业务侧逻辑，目前以通用固件的形式对接了插座、开关、球泡灯等电工照明产品。如您希望接入试用，可以联系 乐鑫商务 获取更多详细信息。

3.10.7 除 Github 外是否有其他途径拉取 ESP RainMaker 源码？

- Gitee 已定期自动同步 固件开发仓库。如果您需要在 Gitee 拉取 App 源码，请联系 乐鑫商务，我们将尽快为您同步。
- 您可以使用 esp-gitee-tools 完成 RainMaker 设备固件开发仓库中子模块的拉取。

3.10.8 CLI 工具如何使用？

- 如果您调试的为公有 ESP RainMaker，可以直接使用 CLI 目录下的 rainmaker.py 脚本。
- 如果您调试的为私有 ESP RainMaker，需要将 serverconfig.py (cli/rmaker_lib/serverconfig.py) 脚本中的 HOST 替换为您服务器的 URL BASE。

3.10.9 ESP RainMaker App 执行 Claiming 时出现了错误该如何处理？

Claiming 提示非网络相关的错误时，一般为账户存在问题，例如账户被禁用、申请证书的额度已满。请联系 乐鑫商务 来获取进一步的支持。

3.10.10 ESP RainMaker 中节点、节点属性、设备、设备属性、服务、参数都是什么？有什么用？

- 节点 (node)：节点可以类比成一个产品，拥有一个 node id 作为标识符，是 ESP RainMaker 框架中的最小操作单位。
- 节点属性 (node attribute)：节点属性用来更好地描述与定义节点的功能。
- 设备 (device)：设备是用户层面可控制的具体实体，如开关、球泡灯、温度传感器、风扇。一个节点下允许挂载多个设备，此时节点将作为虚拟网关使用。
- 设备属性 (device attribute)：与节点属性类似，这些元数据用来更好地描述与定义设备的功能。

- 服务 (service)：从结构上与设备一样，主要区别在于服务不需要用户进行可见的操作，如在 OTA 升级中，就存在一些无需用户操作与管理的状态。
- 参数 (parameter)：参数用来实现设备与服务的功能，如球泡灯的电源状态、亮度、颜色，以及 OTA 过程中的状态更新。

上述这些概念可以很好地定义与描述产品的功能，与阿里云、腾讯云在控制台创建的设备模型类似。

3.10.11 ESP RainMaker 是否支持设备与设备之间的联动？

支持，在 ESP RainMaker 中称为自动触发与响应 (Automation Trigger and Actions)，但设置触发的对象为节点与节点而非设备与设备。通过 addAutomationTriggerAction 进行设置，该功能运行在云端，一旦符合预设的规则便会自动发送响应给目标节点。

3.10.12 ESP RainMaker 是否支持 App 端的消息推送？

支持，ESP RainMaker 的消息推送框架基于 GCM (Google Cloud Messaging) 与 APNs (Apple Push Notification service)。在国内建议使用 iOS 手机或装有 Google 服务框架的 Android 手机来测试。

3.10.13 ESP RainMaker 是否支持带时间戳数据的上报及后续的分析？

支持，设备支持按时间戳上报数据，云侧支持按时间点过滤并拉取数据。在 ESP RainMaker 中该数据称为时间序列数据 (Time Series data)，使用单独的 MQTT 主题上报，云端以完成集成。通过 tsdata 拉取数据，可参考 ESP RainMaker 仓库中的 温度传感器例程。

3.10.14 ESP RainMaker App 与 Nova Home App 可以从哪获取？

- iOS 手机可以在 App store 中搜索 ESP RainMaker、Nova Home 获取。
- Android 手机在 Play Store 中搜索 ESP RainMaker、Nova Home 获取。
- ESP RainMaker App Android 版本源码仓库中附带 apk 文件，请参考 ESP RainMaker App 发布版本。

如果访问上述网站困难，可联系 乐鑫商务 获取最新安装包。

3.10.15 ESP RainMaker 中节点配置信息有什么用？与参数信息的区别是什么？如何查看？

- 节点配置信息是用来描述节点的一组 JSON 格式的元数据。在 示例配置 中，*devices* 描述了每个设备的类型、参数个数及每个参数的属性等。*devices* 的数据类型为 JSON 数组，这代表一个节点下允许存在多个设备，方便实现虚拟网关功能，此处未展示的 *services* 同理。节点配置信息使用单独的 MQTT 主题 上报，设备每次连接到云都应首先上报该消息。
- 参数信息用来展示设备及服务中参数的值，值的数据类型源自节点配置消息中对该参数数据类型的配置。当云或者设备需要更新参数时，就会对该信息进行更新。通过设备固件开发 SDK 创建参数时，节点配置信息中将同步更新该参数的配置。
- 可以通过 CLI 工具查看节点配置信息与参数信息，在 CLI 中登录后使用 *getnodeconfig* 命令可获取节点配置信息，使用 *getparams* 可获取参数信息。

3.10.16 ESP RainMaker 中设备最多能上报多大的消息？

AWS 中 MQTT 一次性最大能接收 128 KB 的数据，ESP RainMaker 中无其他限制。但需要注意的是，AWS 对于 MQTT 消息计费采用条数与大小双重规则，当消息大小每超过 5 KB 时则视为 1 条消息，以此类推，若上报 11 KB 的消息则视为 3 条消息进行计费。具体计费规则请联系 乐鑫商务 获取。

3.10.17 ESP RainMaker App 中显示设备离线总是很慢，能否加快？

设备的离线检测基于 MQTT 心跳包超时时间，默认为 120 s，即最慢能够在 180 s 检测到离线。缩短心跳包发送时间虽然能够更快的检测到设备离线，但会增加消息条数。

3.10.18 ESP RainMaker 方案适配了哪些芯片？用哪个 IDF 版本编译？是否支持其他平台的芯片？

- RainMaker 设备固件开发 SDK 目前完成了对 ESP32 系列芯片适配。
- ESP-IDF 版本需大于 v4.1，若使用 ESP32-C3 需切换到 v4.3 及以上，使用 ESP32-S3 需切换到 v4.4 及以上，使用 ESP32-C2 需切换到 v5.0 及以上。
- 支持，RainMaker 设备固件开发 SDK 提供 MQTT 适配层，需要您自行完成对接。

3.10.19 ESP RainMaker 方案中 Claiming 有 3 种形式，区别在哪？该如何选择？能否在私有部署使用中使用？

- 具体区别请查看 Claiming 实现细节。
- 对带有蓝牙功能的芯片优先选择 *Self Claiming*，其次为 *Assisted Claiming*（*Self Claiming* 最近已更改为对所有 ESP32 系列芯片开放，并非仅仅适用于 ESP32-S2）。不带蓝牙功能的芯片选择 *Self Claiming*。若 *Assisted Claiming* 与 *Self Claiming* 均无法成功，则选择 *Host Driven Claiming* 或联系 乐鑫商务 处理。
- 不可使用，详细原因请查看 为什么 Claiming 无法为私有服务器部署? 。

3.10.20 ESP RainMaker 支持哪些配网方式？这些配网如何实现？能否修改添加自己的配网逻辑？

- 目前支持蓝牙配网与 Soft Ap 配网。
- 配网方案是通过 ESP-IDF 中的 wifi_provisioning 组件实现的。运行 RainMaker 设备固件开发仓库中的例程时，还将打印二维码，二维码中的信息包含该设备支持的配网方式及加密字符串，可以使用 ESP RainMaker App 扫描读取。
- 可以添加自己的逻辑。需要注意的是，ESP RainMaker 中的配网通常指，设备连接 Wi-Fi 与完成用户绑定，无论如何自定义，都必须包含这两步。

3.10.21 ESP RainMaker App 在配网时有时会弹出配对选项，如何取消？

在 menuconfig 中将 Component config -> Wi-Fi Provisioning Manager 的下述选项关闭即可。

[ ] Enable BLE bonding

[ ] Enable BLE Secure connection flag

[ ] Force Link Encryption during characteristic Read / Write

3.10.22 ESP RainMaker 是否支持本地控制？

支持通过 Wi-Fi 进行局域网下的本地通信，设备发现基于 mDNS 服务，通过 ESP-IDF 中的 esp_local_ctrl 组件实现，RainMaker 示例均默认开启，可以通过下述方式查询当前网络下已启用本地控制的 ESP RainMaker 设备：

- Windows 平台，请先下载安装 Bonjour，随后在命令行中执行 *dns-sd -P _esp_local_ctrl._tcp*。
- Linux 平台，在命令行中执行 *avahi-browse -r _esp_local_ctrl._tcp*。

ESP RainMaker App 将实时扫描，并优先使用本地控制进行通信。更多本地控制的细节，请查阅开发指南中的 本地控制章节。

3.10.23 使用 ESP RainMaker Topic 方式进行 OTA 时，有时会报 *！The certificate is not correctly signed by the trusted CA*，这该如何处理？

请拉取最新的代码，并确认 OTA 时使用的为 最新的 OTA 服务器证书。如果确定为最新的证书则可尝试下述方案:

1. 获取 OTA 使用的 URL。您可以在云下发给设备的 OTA 消息中找到 URL，格式通常为 *https://esp-rainmaker-ota-xxxx-dev.s3.us-west-1.amazonaws.com/users/xxx/firmwar/xxx/xxxxxxxxx*。
2. 查询该链接使用的 OTA 服务器证书。您需要使用 Linux 命令行执行 *openssl s_client -showcerts -verify 5 -connect esp-rainmaker-ota-xxxxx-dev.s3.us-west-1.amazonaws.com:443 < /dev/null*。
3. 替换证书。如果一切正常，在执行完成第 2 步后，命令行中将打印多个证书，您需要选择并替换您当前使用的证书。

如果上述方案仍然无法成功，可联系 乐鑫商务 提供进一步解决方案。

3.10.24 Swagger 上提供的 RESTful API 可以在线调试吗?

可以，点击每条 API 右侧的 *Try it out* 按钮即可。需要注意的是，如果 API 带有锁的图标意味着需要 accesstoken 才能执行，您需要先使用 swaggerapis 里的 user login 进行登录，该接口将返回三组 token，随后点击页面上方的 Authorize 将 accesstoken 填入到页面中即可。

3.10.25 ESP RainMaker App 中的 UI 是如何确定的？如何自定义呢？

- ESP RainMaker App 中所有 UI 的展示是由设备上报的 *node_config* 消息决定的，字段与 UI 的映射关系请查看开发指南中的 标准参数 章节。
- 可以为每个参数根据数据类型指定不同的标准 UI，不同参数的添加顺序也将决定 App 上的显示顺序。如果需要我们支持更多不同风格的 UI，可联系 乐鑫商务 处理。

3.11 苹果应用

[English]

3.11.1 iOS 蓝牙设备名称缓存问题如何解决？

以下为 OC 和 Swift 的解决方法：

OC

```
(void)centralManager:(CBCentralManager *)central␣
↪didDiscoverPeripheral:(CBPeripheral *)peripheral
advertisementData:(NSDictionary<NSString *,id> *)advertisementData ␣
↪RSSI:(NSNumber *)RSSI{
    NSString *localName = [advertisementData objectForKey:@
↪"kCBAdvDataLocalName"];}
```

Swift

```
func centralManager(_ central: CBCentralManager, didDiscover peripheral:␣
↪CBPeripheral,
advertisementData: [String : Any], rssi RSSI: NSNumber) {
    let localName = advertisementData["kCBAdvDataLocalName"]}
```

3.11.2 阿里飞燕平台 SDk 为何报错找不到 #import <IMLDeviceCenter/IMLDeviceCenter.h> 头文件？

- 请将 ALLOW_NON_MODULAR_INCLUDES_IN_FRAMEWORK_MODULES 设为 YES。

3.11.3 iOS 13.0 及以上版本如何获取 Wi-Fi 信息？

- 在.plist 文件中开启定位权限。
- Xcode 开启 Wi-Fi 权限。
- Apple 官网申请 Identifiers Access Wi-Fi Information 权限。
- 手动请求定位权限。
- 导入系统框架 #import <SystemConfiguration/CaptiveNetwork.h> 获取 Wi-Fi 信息。

3.11.4 iOS 14.0 如何增加本地网络权限？

- 在.plist 文件中开启 NSLocalNetworkUsageDescription 权限。

3.11.5 AWS 如何生成.p12 证书？

openssl pkcs12 -export -in /Users/xxx/Desktop/awscer/73bb87b879-certificate.pem.crt -inkey /Users/xxx/Desktop/awscer/73bb87b879-private.pem.key -CAfile /Users/xxx/Desktop/awscer/AmazonRootCA1.pem -out awsiot-identity.p12

3.11.6 如何获取 AWS SDK 自带的登录注册验证码？

- 使用邮箱注册的账号在获取验证码时，由于网络原因，需要等待较长时间（大概 2 ~ 4 小时左右）才能收到。
- 点击获取验证码之后不可重复点击，否则旧验证码将失效。

3.11.7 APP 如何在后台扫描蓝牙（两种方式）？

- 第一种方式：扫描所有蓝牙设备。

```
[self.cbCentralMgr scanForPeripheralsWithServices:nil options:nil];
```

- 第二种方式：扫描指定 serviceUUID 蓝牙设备。

```
[self.cbCentralMgr scanForPeripheralsWithServices:@[[CBUUID UUIDWithString:@"
指定的 serviceUUID"]] options:nil];
```

3.11.8 如何解决 iOS 14.5 UDP 广播 sendto 返回 -1 的错误？

问题背景：

- 手机系统升级到 iOS 14.5 之后，UDP 广播发送失败。
- 项目中老版本使用 socket。
- 项目中新版本使用 CocoaAsyncSocket。
- 两种 UDP 发包方式都会报错：No route to host。

报错具体内容如下：

```
sendto: -1
client: sendto fail, but just ignore it
: No route to host
```

问题解决：

由于 192.168.0.255 广播地址只是当前本地地址，App 中需要动态改变前三段 192.168.0 本地地址，解决方法如下：

```
NSString *localInetAddr4 = [ESP_NetUtil getLocalIPv4];
NSArray *arr = [localInetAddr4 componentsSeparatedByString:@"."];
NSString *deviceAddress4 = [NSString stringWithFormat:@"%@.%@.%@.255",arr[0],
↪ arr[1], arr[2]];
```

发包过滤，只需要过滤地址最后一段是否为 255

```
bool isBroadcast = [targetHostName hasSuffix:@"255"];
```

3.11.9 如何解决 iPhone11 iOS 14.7 下载安装 App 后，点击图标，App 闪一下就回到了桌面的问题？

可以从以下四个方面进行排查：

- 项目中引入的音频动态库版本太老不兼容
- 系统 API 在 iOS 15.0 以下版本不兼容
- Xcode 版本太老
- 电脑是 M1 芯片

3.11.10 如何解决 iOS 国际化文本格式报错 *read failed: Couldn't parse property list because the input data was in an invalid format* 问题？

数据格式错误一般会有下面几种情况：

- 末尾少了分号
- 字符使用了全角字符（中文字符）
- 中间少了 =
- 少了双引号或者引号没有成对出现
- 文本中出现了不必要的特殊字符

3.12 第三方云服务

[English]

3.12.1 OTA 升级有没有相关示例参考？

请参考如下链接：

- ESP8266 OTA
- ESP32 系列 OTA

3.12.2 ESP32 如何对接天猫精灵，是否有相应的资料？

ESP32 对接天猫精灵可以使用 esp-aliyun SDK，可以参阅 ESP 设备对接阿里云指南。

3.12.3 esp-aliyun 与 esp-ali-smartliving 的区别？

- esp-aliyun 对接的是“物联网平台”。
- esp-ali-smartliving 对接“生活物联网平台”。
- 阿里云将两个平台在云端互通，使用功能上相似度较高，并可相互替代。
- 两者区别可以参见 生活物联网平台与物联网平台的区别。

3.12.4 使用乐鑫的产品连接乐鑫云平台，遇到问题如何提问与反馈？

- 如果您使用的乐鑫云平台是 ESP RainMaker，建议您直接将问题反馈至 Github。
- 如果是其他云平台，您可以将使用的云平台信息以及您的问题汇总，发送到 sales@espressif.com。

3.12.5 ESP32 与 ESP8266 可以连接 Alexa 或者 Google home 吗？

- Alexa 可以参考 aws_iot，做一些 Alexa 配置即可。
- Google home 当前没有示例，可以参考 ESP32 参考示例 esp-google-iot。

3.12.6 ESP32 + 以太网 + MQTT 方式接入阿里云，应该怎么做？

- 使用 esp-aliyun，将 Wi-Fi 初始化代码替换为 Ethernet 初始化即可。可以参考 ESP_IDF 下的 Ethernet 示例。

3.13 ESP-WIFI-MESH 应用框架

[English]

注解： 如您有新的 Wi-Fi Mesh 相关应用场景需求，我们推荐您直接使用我们新推出的 ESP-Mesh-Lite 方案，而不是 Wi-Fi Mesh。

3.13.1 ESP-WIFI-MESH 占用多大内存？是否需要外部 PSRAM ？

ESP-WIFI-MESH 内存占用约 60 KB。是否需要外部 PSRAM 取决于应用场景的复杂情况，一般性应用无需外部 PSRAM。

3.13.2 ESP-WIFI-MESH 能否批量 OTA？

ESP-WIFI-MESH 设备支持批量 OTA。OTA 方式为：根节点下载固件，然后再发至其他节点。具体示例请参考: mupgrade。

3.13.3 ESP32 支持多少设备进行 ESP-WIFI-MESH 组网？

ESP32 支持 1000 个设备进行 ESP-WIFI-MESH 组网。若要稳定连接大量设备，建议在每个 ESP-WIFI-MESH 网络下，组网设备不超过 512 台。

3.13.4 ESP32 的 ESP-WIFI-MESH Router 模式与 No Router 模式有什么区别？

- ESP-WIFI-MESH 网络的 Router 模式是根据路由器进行组网，根节点连路由器。
- No Router 模式是在无路由器的场景下进行自组网，此模式下不可与外部数据交互。

3.13.5 ESP32 的 ESP-WIFI-MESH 能否在子设备搜索不到路由器信号时完成组网？

ESP32 的 ESP-WIFI-MESH 能实现：若拥有配置相同 Wi-Fi 的 SSID，在子设备没有搜索到 Wi-Fi 时，子设备也可以连接到根节点。

3.13.6 ESP32 的 ESP-WIFI-MESH 是否可自动修复网络？

ESP32 的 ESP-WIFI-MESH 可自动修复网络，ESP-WIFI-MESH 有检测网络断线的机制。

3.13.7 使用 ESP32 的 ESP-WIFI-MESH，如何设置可以在没连接到 Wi-Fi 的情况下形成自组网？

需要指定一个设备作为 Root 节点，可参考 说明 和 示例。

3.13.8 使用 ESP32 进行 ESP-WIFI-MESH 应用，在组网自动选举根节点时，是否可以指定局部模块进行选举？

ESP32 的 ESP-WIFI-MESH 可以指定设备为子节点，它将不参与根节点的选举，可实现局部 mesh 网络进行选举。

3.13.9 使用 ESP32 进行 ESP-WIFI-MESH 应用，在无路由场景下，多个根节点之间能互发消息吗？

无路由场景下，多个根节点之间不能互发消息。

3.13.10 ESP-WIFI-MESH 可以批量 OTA 吗？

- ESP-WIFI-MESH 设备可以批量 OTA。
- OTA 的方式是根节点下载固件，然后再发至其他节点。
- 具体示例请参考 https://github.com/espressif/esp-mdf/tree/master/examples/function_demo/mupgrade。

3.13.11 如何查询 ESP-WIFI-MESH APP 端源码？

- iOS 端源码链接: https://github.com/EspressifApp/EspMeshForiOS
- 安卓端源码链接: https://github.com/EspressifApp/EspMeshForAndroid

3.13.12 ESP-WIFI-MESH 是否有无路由方案完成自组网？

Demo 中有 no-router, 以及 get-started 两个无路由方案，可以参考。

3.13.13 利用 Mwifi 自动组网后，如何获得某个节点的所有潜在父节点的信号强度 (rssi)？

- 可以通过 `mwifi_get_parent_rssi()` 获取其父节点的信号强度。
- 可以通过例程 https://github.com/espressif/esp-mdf/blob/master/examples/wireless_debug 参与获取其他结点的信号强度。

3.13.14 在 esp-mdf 的 MESH 网络内部，节点之间的通信是基于什么协议？

Mesh 网络内部，是基于数据链路层的自定义协议，即我们核心之一。有 ack 机制，但是没有超时/重传机制，如有需求可以自行在应用层添加。

3.13.15 ESP-WIFI-MESH 可以将所有的节点都连接至路由上吗？

- 只有 root 节点才可以连接上路由器，下面的子节点将会直接或者间接地连接上 root 节点，然后通过 root 节点和路由通讯。

3.13.16 ESP-WIFI-MESH 的 root 节点能否通过 4G 拨号实现联网？

功能可以实现，但目前没有专门针对该场景的应用，可参考 ESP-MDF 中 no-router，该 Demo 根节点直接通过串口和电脑通讯，可修改成将数据通过 4G 模块进行传输。

3.13.17 esp_mesh_set_parent 函数成功连接后，断开 AP，该函数会不断发起重新连接，如何设置重新连接次数？

- 如果您使用自组网方案，ESP-WIFI-MESH 默认不会重连。当断开时，您需要调用 `esp_wifi_scan_start`，获取可以连接的设备，以重新设置父节点。参见：Mesh Manual Networking Example。
- 推荐您使用自组网的方案进行开发。

3.13.18 设置按钮后报错：`phy_init: failed to load RF calibration data`。

乐鑫芯片初次上电会有 RF 自校准，并将数据存在 NVS 里。若擦除了该部分，就会出现这行打印，做全校准。

3.13.19 如何暂停/恢复 Mwifi？

使用 `mwifi_stop/mwifi_start` 暂停/开始 mesh。

3.13.20 ESP32-S 无路由 MESH 组网，APP 怎么连接 root 接口的 softAP？

MESH 的 AP 不支持非 mesh 设备接入，您可以使用一个 ESP32 作用 softAP。

3.13.21 ESP-WIFI-MESH 能连到 AP，但不能连到 AP 上的 TCP SERVER？

请参考 GitHub issue: mesh -> “with-router” example doesn’t work with espressif IDF softAP #71。

3.13.22 Mwifi 例程怎么修改网络的 AP 连接和最大层数？通信时的最大带宽和延时是多少？

- 可以通过 menuconfig 里面的配置进行修改，位于：Component config -> MDF Mwifi -> Capacity config。
- 通信性能可参考：performance。
- WIFI-MESH 带宽可通过 console_test 例程进行测试。

3.13.23 如何获得实时的传感器返回值？

由于设备端是一个 HTTP server，所以只能由 APP 发起请求，您可以采用如下两种方式获取实时数据：

- 当传感器数据变化时，通过 UDP 通知手机来主动请求数据。如果使用我们本地通信的协议，发送如下命令使 APP 主动请求设备数据：

```
data_type.protocol = MLINK_PROTO_NOTICE;
ret = mwifi_write(NULL, &data_type, "status", strlen("status"), true);
MDF_ERROR_CONTINUE(ret != MDF_OK, "<%s> mlink_handle", mdf_err_to_name(ret));
```

- 搭建一个服务器 (TCP/MQTT/HTTP server)，与服务器建立 TCP 长连接，传感器数据变化将主动上报。

3.13.24 新节点可能已经安装在设备中，且该设备已经安装在距离 ROOT 节点较远的位置，请问该节点如何加入 ESP-WIFI-MESH 网络？

- 您使用的应该是 get-started Demo。为了方便用户测试，该 Demo 是无路由的一种方案，即指定了根节点，所以在 root crash 后，其余设备将无法恢复。
- 可参考 development_kit 中的 light Demo。该 Demo 可配合 ESP-Mesh App 进行使用（Android 版可在 官网 下载，iOS 版可在 App Store 搜索 ESP-Mesh 下载测试）。
- 该 Demo 示例不指定根节点，由设备自行选举产生，需要配合路由器使用。此种方案下，如果 root 出现故障，剩余设备会自动重新完成组网并连上路由，不需要用户干预。

3.13.25 ESP-WIFI-MESH App 源码是否开放？

- 我们已经将 ESP-Mesh App 源码开放到了 GitHub 上，请参考 EspMeshForAndroid 和 EspMeshForiOS。
- 如果在使用中有任何疑问或 Bug，都可以在 GitHub 或者这里进行留言提问，我们会在第一时间进行处理。

3.13.26 Wi-Fi Mesh 数据传送最大的包为多少 Bytes？

- 最大为 1456 Bytes。

3.13.27 ESP32 的 Wi-Fi Mesh 支持 No Router 自组网吗？

- ESP32 的 Wi-Fi Mesh 支持 No Router 自组网，可参见例程 esp-mdf/examples/function_demo/mwifi/no_router。

3.13.28 ESP32 使用 Wi-Fi Mesh 时允许的最大节点层数是多少？

- 在 WiFi Mesh 网络中，可以通过 esp_mesh_set_max_layer() 函数设置网络最大层数。
- 对于树形拓扑结构，最大值为 25；对于链式拓扑结构，最大值为 1000。

3.13.29 使用 ESP32 开发板测试 esp-mdf/examples/function_demo/mwifi/router 例程，ESP32 连接路由器后，在路由器连接端显示的设备名称为“espressif”，如何修改此名称？

- 在 menuconfig → Component config → LWIP ——> (espressif) Local netif hostname 中修改设置即可。

3.13.30 Wi-Fi Mesh 可以通过 TCP Server 给特定节点发送消息吗？

- Wi-Fi Mesh 网络可在 TCP 服务器中发送数据到指定节点或组地址，可参考 demo。

3.13.31 在 ESP32 Wi-Fi Mesh 网络运行过程中，若根 (Root) 节点丢失，系统会反馈什么事件？

- 若根 (Root) 节点丢失，所有节点将会触发 MDF_EVENT_MWIFI_PARENT_DISCONNECTED (MESH_EVENT_PARENT_DISCONNECTED)，然后开始重新扫描 (Scan)，进行重新选举，直到选举出新的根 (Root) 节点。

3.13.32 使用 ESP32 进行 Wi-Fi Mesh 应用开发，目前使用的是 esp_mesh_send() 函数，发现服务器没有接收到任何数据。如何将数据从叶节点 (leaf node) 传输到外部服务器？

- esp_mesh_send() 只能用于 Wi-Fi Mesh 网络内部数据通信。
- 叶节点 (leaf node) 往外部服务器发送数据，需要通过根节点 (root node) 转发数据。
- 正确的做法是：叶节点先将数据发给根节点，根节点再把数据发给外部服务器。

3.13.33 ESP-MESH 设备组网之后如何做 OTA 升级？

- ROOT 节点可以连接服务器获取到升级 bin 文件，然后把固件通过 MAC 地址去发送给对应的模组进行 OTA 升级。
- 详情请参考 mupgrade demo。

3.13.34 是否有 ESP-MESH 灯参考设计？

- 灯的整体设计是由第三方工厂完成的，我们并没有相关原理图或者 PCB 布局。但是单从模块角度，我们只需要给芯片供电，芯片输出 PWM 控制灯的颜色或色温变化即可，并没有太复杂的设计。
- 可以参考 ESP-MDF 获取更多关于 MESH 的信息。

3.13.35 ESP-MESH 节点不做任何配置，默认是什么模式？

- 默认是 IDLE 模式。

3.13.36 ESP-MESH 启动时开启 AP+STA 模式，手机可以搜索到 AP 吗？

- 不可以，ESP-MESH 是乐鑫私有协议, 详情请参考 WIFI-MESH 介绍。

3.13.37 设备已经组网完成，新增设备是否需要全部重新扫描？

- 不需要，只需要在当前子节点中扫描，找到信号强度最好的那个节点作为它的父节点即可。

3.13.38 ESP32 作为主设备对多个从设备进行时间同步，是否可以满足误差在 2 ms 内的需求？

- 针对此应用场景，建议基于 esp-mdf 来开发，可参考 esp-mdf/examples/development_kit/light 例程。
- 使用 esp_mesh_get_tsf_time() 来实现，此精度可满足需求。

3.13.39 ESP-MESH 中如何获取节点类型？

- 可以调用 esp_mesh_get_type 接口获取。

3.13.40 ESP-Mesh 根节点通过 ethernet 向服务发消息示例？

- 请参考 root_on_ethnernet 示例。

3.13.41 esp-mesh-lite 解决方案是否支持无路由器的应用场景？

- 支持，esp-mesh-lite 解决方案支持的应用场景可参见 esp-mesh-lite 特性 说明。
- 可基于 esp-mesh-lite/examples/mesh_local_control 例程使能 `Component config > ESP Wi-Fi Mesh Lite > Enable Mesh-Lite > Mesh-Lite info configuration > [*] Join Mesh no matter whether the node is connected to router` 配置选项来测试。
- 对于无路由器的方案需要注意如下：
  - 尽量确定一个根节点，可通过 `esp_mesh_lite_set_allow_level(1)` 设置。
  - 对于其他节点，建议使用 `esp_mesh_lite_set_disallow_level(1)` 函数来禁止它们成为根节点。
  - Mesh-Lite 的应用场景下，Mesh 网络的建立需要依靠设备物理距离和 Wi-Fi 信号质量等因素，因此需要进行充分的实地测试和调试，以保证 Mesh 网络的性能和稳定性。

3.13.42 esp-wifi-mesh 已经组网时，根节点或子节点可以同时开启 Wi-Fi Scan 扫描周围可用的 AP 信息吗？

- `esp-wifi-mesh` 已经组网时，任何节点设备都不支持开启 Wi-Fi Scan 功能。

3.13.43 使用 esp-wifi-mesh router 解决方案时，如何切换新的路由器进行组网？

- 可以在 `MESH_EVENT_PARENT_DISCONNECTED` 事件后，修改如下代码：

```
mesh_router_t change_router = {
    .ssid = "TP-LINK_CSW",
    .password = "12345678",
    .ssid_len = strlen("TP-LINK_CSW"),
};
esp_mesh_set_self_organized(false, false);
esp_mesh_set_router(&change_router);
esp_mesh_set_self_organized(true, true);
```

CHAPTER 4

软件平台

[English]

4.1 蓝牙

[English]

4.1.1 移植例程 gatt_server 出现头文件不存在的错误？

移植例程 gatt_server 出现错误提示 `fatal error: esp_gap_ble_api.h: No such file or directory`，但头文件已经包含此文件。

- 检查 sdkconfig，是否未从例程中移植 sdkconfig.defaults。通常 SDK 中蓝牙默认关闭不编译，需要配置开启。
- 如果使用 cmake 需要将例程中 CMakeLists.txt 文件内的链接配置一同复制。

4.1.2 ESP32 可以支持 Bluetooth LE 5.0 吗？

ESP32 硬件不支持 Bluetooth LE 5.0，支持 Bluetooth LE 4.2。

ESP32 目前通过了 Bluetooth LE 5.0 的认证，但 Bluetooth LE 5.0 的新功能 ESP32 不支持（未来会有其它芯片支持 Bluetooth LE 5.0 全部新功能）。

4.1.3 为什么 Bluetooth® LE 开始广播后，有些手机扫描不到？

- 需确认手机是否支持 Bluetooth LE 功能：有的手机在“设置” -> “蓝牙”中只显示默认的经典蓝牙，Bluetooth LE 广播会被手机过滤掉。
- 建议使用专用的 Bluetooth LE App 来调试 Bluetooth LE 功能。例如，苹果手机可以使用 LightBlue App。
- 需确认广播包的格式符合规范，手机一般会对不符合格式的广播包进行过滤，只有格式正确的才能被显示出来。

4.1.4 ESP32 能否使用蓝牙进行 OTA？

可以使用蓝牙进行 OTA。如果是用 Bluetooth®，可以基于 bt_spp_acceptor 和 bt_spp_initiator 修改。

如果是用 Bluetooth LE，可以基于 ble_spp_server 和 ble_spp_client 修改。

4.1.5 ESP32 的蓝牙双模如何共存及使用？

ESP32 支持的双模蓝牙并没有特殊的地方，不需要做复杂的配置或调用即可使用。从开发者的🄵度来看，Bluetooth® LE 调用 Bluetooth LE 的 API，经典蓝牙调用经典蓝牙的 API。

经典蓝牙与 Bluetooth LE 共存说明可参考文档 ESP32 Bluetooth & Bluetooth LE 双模蓝牙共存说明。

4.1.6 ESP32 的 Bluetooth® LE 吞吐量是多少？

- ESP32 的 Bluetooth LE 吞吐率取决于各种因素，例如环境干扰、连接间隔、MTU 大小以及对端设备性能等等。
- ESP32 板子之间的 Bluetooth LE 通信最大吞吐量可达 700 Kbps，约 90 KB/s，具体可以参考 ESP-IDF 中的 ble_throughput example。

4.1.7 ESP32 是否支持 BT4.2 DLE (Data Length Extension)？

支持。ESP-IDF 所有版本都支持 Bluetooth® 4.2 DLE，暂无对应的 sample code，可直接调相关接口实现，参见：esp_ble_gap_set_pkt_data_len。

4.1.8 ESP32 的蓝牙和 Wi-Fi 如何共存？

在 `menuconfig` 中，有个特殊选项 `Software controls WiFi/Bluetooth coexistence`，用于通过软件来控制 ESP32 的蓝牙和 Wi-Fi 共存，可以平衡 Wi-Fi、蓝牙控制 RF 的共存需求。

- 如果使能 `Software controls WiFi/Bluetooth coexistence` 选项，Bluetooth® LE scan 间隔不应超过 `0x100 slots`（约 160 ms）。若只是 Bluetooth LE 与 Wi-Fi 共存，则开启这个选项和不开启均可正常使用。但不开启的时候需要注意 Bluetooth LE scan window 应大于 150 ms，并且 Bluetooth LE scan interval 尽量小于 500 ms。
- 若经典蓝牙与 Wi-Fi 共存，则建议开启这个选项。

4.1.9 ESP32 蓝牙的兼容性测试报告如何获取？

请联系 sales@espressif.com 获得兼容性测试报告。

4.1.10 ESP32 蓝牙的发射功率是多少？

ESP32 蓝牙的发射功率有 8 档，对应功率 -12 ~ 9 dBm，间隔 3 dBm 一档。控制器软件对发射功率进行限制，根据产品声明的对应功率等级选取档位。

4.1.11 ESP32 可以实现 Wi-Fi 和 Bluetooth® LE 桥接的功能吗？

可以实现，这个属于应用层开发：可以通过 Bluetooth LE 获取数据，由 Wi-Fi 转出去。可参考 Wi-Fi 和蓝牙共存的 demo，修改为自己的应用即可。

4.1.12 ESP32 的 Bluetooth® LE 工作电流是多少？

电流	最大值 (mA)	最小值 (mA)	平均值
Advertising: Adv Interval = 40 ms	142.1	32	42.67
Scanning: Scan Interval = 160 ms,Window = 20 ms	142.1	32	44.4
Connection(Slave): Connection Interval = 20 ms, Iatency = 0	142.1	32	42.75
Connection(Slave): Connection Interval = 80 ms, Iatency = 0	142.1	32	35.33

4.1.13 ESP32 支持哪些 Bluetooth® LE Profile？

目前支持完整的 GATT/SMP 等基础模块，支持自定义配置；已经实现的配置有 Bluetooth LE HID（设备端）、电池、DIS、BluFi（蓝牙配网）等。

4.1.14 如何使用 ESP32 蓝牙连接手机播放音乐？

用手机通过蓝牙播放音乐，ESP32 用作 A2DP Sink。A2DP Sink Demo 只是通过手机获取 SBC 编码的数据流，若要播放出声音，需要做编解码转换，通常需要编解码器、数/模转换器、扬声器等模块。

4.1.15 ESP32 的 SPP 性能如何？

使用两块 ESP32 开发板对跑 SPP，单向吞吐量量可达 1900 Kbps，约 235 KB/s，已接近规范里的理论值。

4.1.16 ESP32 的 Bluetooth® LE 传输速率最大为多少？

屏蔽箱测试 Bluetooth LE 传输速率可以达到 700 Kbps。

4.1.17 ESP32 Bluetooth® LE 如何进入 Light-sleep 模式呢？

硬件上需要外加 32 kHz 的外部晶振，否则 Light-sleep 模式不会生效。

软件上（SDK4.0 以及以上版本才会支持）在 menuconfig 中需要使能以下配置：

- Power Management:| `menuconfig > Component config > Power management > [*] Support for power management`
- Tickless Idle:| `menuconfig > Component config > FreeRTOS > [*] Tickless idle support (3) Minimum number of ticks to enter sleep mode for (NEW)`

注解： 需使能“Tickless idle”功能使 ESP32 自动进入 Light-sleep 模式。如果在 3 个节拍（默认）内无任务运行，则 FreeRTOS 将进入 Light-sleep 模式，即 100 Hz 节拍率下为 30 ms。若您希望缩短 Light-sleep 模式的持续时间，可通过将 FreeRTOS 节拍率调高来实现，如：`menuconfig > Component config > FreeRTOS > (1000) Tick rate (Hz)`。

- Configure external 32.768 kHz crystal as RTC clock source :| `menuconfig > Component config > ESP32-specific > RTC clock source (External 32 kHz crystal)[*] Additional current for external 32 kHz crystal`

注解： “additional current”选项为解决 ESP32 晶振失败的替代方案。请在您使用外部 32 kHz 晶体时使能该选项。该硬件问题将在下一个芯片版本中解决。

- Enable Bluetooth modem sleep with external 32.768 kHz crystal as low power clock :| `menuconfig > Component config > Bluetooth > Bluetooth controller > MODEM SLEEP Options > [*] Bluetooth modem sleep`

4.1.18 选择 ESP32 芯片实现蓝牙配网的方式，是否有文档可以提供参考？

蓝牙配网说明可参考 ESP32 Blufi。蓝牙配网示例可以参考 Blufi。

4.1.19 ESP32 经典蓝牙 SPP 的传输速率能达到多少？

在开放环境下，双向同时收发，实测可达到 1400+ Kbps 到 1590 Kbps（此数据仅作为参考，实际情况建议客户根据应用环境实测）。

4.1.20 ESP32 的蓝牙是否兼容 Bluetooth® ver2.1 + EDR 协议？

兼容。ESP32 的蓝牙是向下兼容的，您可以使用官方的 蓝牙示例 进行测试。

4.1.21 ESP32 支持多少蓝牙客户端连接？

Bluetooth® LE Server 最大支持 9 个客户端连接，应用中需查看配置参数 ble_max_conn。测试稳定连接为 3 个客户端。

4.1.22 ESP32 如何获取蓝牙设备的 MAC 地址？

可调用 esp_bt_dev_get_address(void); API 来获取蓝牙配置的 MAC 地址。也可以调用 esp_err_t esp_read_mac(uint8_t* mac,esp_mac_type_ttype); API 获取系统预设的分类 MAC 地址。

4.1.23 ESP32 SDK 中默认的蓝牙的发射功率是多少？

- ESP32 SDK 中默认情况下使用功率级别 5，相应的发射功率为 +3 dBm。
- ESP32 蓝牙的发射功率从 0 到 7，共有 8 个功率级别，发射功率范围从–12 dBm 到 9 dBm。功率电平每增加 1 时，发射功率增加 3 dBm。

4.1.24 ESP32 Wi-Fi Smartconfig 配网和 Bluetooth® LE Mesh 可以同时使用吗？

不推荐同时打开。

- Smartconfig 需要一直收配网数据，所以会一直占用天线，如果和 Bluetooth LE Mesh 共同使用，会导致失败率非常高。
- Bluetooth LE Mesh 可以和 BluFi 同时使用，所以推荐配网方式选择 BluFi 配网。

4.1.25 ESP32 的经典蓝牙工作电流是多少？

A2DP (Single core CPU 160 Mhz，DFS = false，commit a7a90f)

电流	最大值 (mA)	最小值 (mA)	平均值
Scanning	106.4	30.8	37.8
Sniff	107.6	31.1	32.2
Play Music	123	90.1	100.4

4.1.26 ESP32 如何修改蓝牙的发射功率？

蓝牙发射功率可通过 esp_ble_tx_power_set(); 函数进行设置，可参见 esp_bt.h。

4.1.27 ESP32 的 Bluetooth® LE 蓝牙配网兼容性如何？是否开源？

- ESP32 的蓝牙配网，简称 BluFi 配网，兼容性与 Bluetooth LE 兼容性一致，测试过苹果、华为、小米、OPPO、魅族、一加、中兴等主流品手机，兼容性良好。
- 目前 BluFi 协议及手机应用部分的代码都已经开源。

4.1.28 ESP32 运行 bt_spp_acceptor 例程时，IOS 设备无法扫描到 ESP32 设备是什么原因？

- 苹果开放的蓝牙有：A2DP、HID 的 keyboard、avrcp 以及 SPP（需要 MFI）和高端的 Bluetooth® LE 外加给予 Bluetooth LE 的 ANCS。
- 如果 IOS 设备想要和对端设备通过 SPP 通信，那么对端设备的 SPP 需要通过 MFI 认证。目前 ESP32 SPP 没有通过 MFI 认证，因此 IOS 设备无法扫描到 ESP32。

4.1.29 ESP32 Bluetooth® LE/Bluetooth® Secure Simple Pairing (SSP) 与 legacy pairing 安全性对比？

- Secure Simple Pairing (SSP) 比 legacy pairing 更加安全。
- legacy pairing 使用对称加密算法，Secure Simple Pairing (SSP) 使用的是非对称加密算法。

4.1.30 ESP32 Bluetooth® LE MTU 大小如何确定？

- ESP32 端蓝牙 Bluetooth LE 默认的 MTU 为 23 字节，最大可以设置为 517 字节。
- 手机端的 MTU 由手机端自行定义，最终通信的 MTU 选择两端 MTU 较小的那一个。

4.1.31 ESP32 Bluetooth® LE 模式下广播数据时遇到“W (17370) BT_BTM: data exceed max adv packet length”如何解决？

- 出现该警告的原因是广播的数据长度超出最大广播数据包长度限制。
- 广播有效载荷数据长度最大为 31 字节。如果超过 31 字节，那么蓝牙协议栈会丢弃一些数据，并且给出警告。
- 如果需要广播的数据长度超出最大限制，超出的数据可以放在扫描响应数据包 (scan response data) 中。

4.1.32 ESP32 Bluetooth® LE 能否同时支持主从模式，作 gatt server 的同时，也可作为 gatt client 接收其他设备的广播数据？

- 支持，可参考例程 gattc_gatts_coex。

4.1.33 ESP32 的 Bluetooth® LE 连接数 6 个以上会有哪些风险？

- 通常要根据具体的应用决定，在常规场景下，ESP32 Bluetooth LE 连接 3 个设备可以稳定通信。
- Bluetooth LE 的最大连接数未有一个准确的值，在多个 Bluetooth LE 设备同时连接的的时候，RF 是分时复用的，需要设计者保证每一个设备不会长时间占用导致其他设备超时断开。
- 连接参数里面有 connection interval、connection window、latency、timeout, 可以在 `latency` 以内的不应答，但是若超过 `timeout` 的时间，将会导致连接断开。
- 假设配置参数中 `interval` 是 100，window 是 5，Wi-Fi 关闭时，将会连接较多设备。如果用了 Wi-Fi，或者 `interval` 设置的太小，将只能连接较少设备。
- 当 Bluetooth LE 支持多个设备并发连接时，RF 的 solt 管理出错概率会增加，所以 Bluetooth LE 设备连接较多时，需要针对具体场景调试。

4.1.34 使用 ESP32 设备作为 Bluetooth® LE 主机，最大支持多少台从机设备进行连接？

- ESP32 的 Bluetooth LE 最大支持 9 台从机设备进行连接，建议连接数量不超过 3 个。
- 可通过 `menuconfig > Component config > Bluetooth > Bluetooth controller > BLE MAX Connections` 进行配置。

4.1.35 ESP32 如何通过 Bluetooth® BR/EDR 传文件？

- 可参考链接 classic bt 下的 `bt_spp_acceptor` 或者 `bt_spp_initiator` 例程。

4.1.36 ESP32 下载 ESP_SPP_SERVER 例程，如何修改蓝牙设备名称？

- 蓝牙设备名称可以通过修改 `adv` 参数实现：

```
static const uint8_t spp_adv_data[23] = {
  0x02,0x01,0x06,
  0x03,0x03,0xF0,0xAB,
  0x0F,0x09,0x45,0x53,0x50,0x5f,0x53,0x50,0x50,0x5f,0x53,0x45,0x52,0x56,0x45,
↪0x52};
```

- 第三行 0x0F 表示后续数据长度为 15，0x09 表示数据类型（固定不变），0x45 开始后续数据代表设备名称对应的 ASCII 码（默认为：BLE_SPP_SERVER)。

4.1.37 ESP32 下载 BluFi 例程进行配网，若使用 EspBluFi APP 在配网过程配置了一个错误的 Wi-Fi 从而无法连接，此时从 APP 端向设备端发送“扫描”命令后就会导致设备重启，是什么原因？

- BluFi 例程规定在 Wi-Fi 连接时不可以发送 Wi-Fi 扫描命令。
- 但可在 blufi_example_main.c 文件下的 case ESP_BLUFI_EVENT_GET_WIFI_LIST:{}; 函数的首行增加 ESP_ERROR_CHECK(esp_wifi_disconnect()); 函数来解决此问题。

4.1.38 使用 ESP32，如何指定 BLE 连接/发送在 core 0 上运行？

- ESP32 的 BLE 连接/发送目前仅支持指定在 core 1 上运行。可通过使能 `menuconfig > Component config > FreeRTOS > Run FreeRTOS only on first core` 进行设置。
- 根据此应用需求，可使用 xTaskCreatePinnedToCore() 或 xTaskCreateStaticPinnedToCore() API 来创建任务核分配。具体说明参见 core assignment。

4.1.39 ESP32 设置中文蓝牙设备名称会异常显示乱码，原因是什么？

- 这是因为此时编辑器的中文编码格式不是 UTF-8，需要把编辑器的编码格式改成 UTF-8。

4.1.40 使用 ESP32 在蓝牙通道上传分包时，一包最大传输数据长度为 253（MTU 设置为 263），导致在传输大量数据包进行多包读取时传输较慢。请问是否有 BluFi 扩展协议，可支持一包传输较大长度的数据，或者有其他解决方案可提高传输速率吗？

- 在蓝牙通道上传输大量数据包进行多包读取时传输较慢，可通过调整蓝牙连接参数来改善传输速度。
- BLE 包长度设置取决于 ESP_GATT_MAX_MTU_SIZE 设置，可参见 说明。
- 设置的 MTU Size 大小会影响数据传输率，有效的 MTU 长度需要通过 MTU 交换方式来改变默认的 MTU 的大小。最终进行 MTU 交换使用的 MTU S ize 才是作为两者通信时的 MTU Size。可查看 MTU 交换后的值是多大，例如这样的值：

```
case ESP_GATTS_MTU_EVT:
        ESP_LOGI(GATTS_TAG, "ESP_GATTS_MTU_EVT, MTU %d", param->mtu.mtu);
```

4.1.41 ESP32 经典蓝牙支持哪些 Profile？

- 目前支持 A2DP、AVRCP、SPP、HFP、HID。

4.1.42 ESP32-C3 Bluetooth® LE (BLE) 稳定连接的数目可以达到多少个？

- 建议 4 个以内。

4.1.43 BLE 中如何修改广播的时间间隔？

- 通过修改广播结构体中的 `adv_int_min` 和 `adv_int_max` 两个参数来设置。这两个分别对应了广播时间间隔的最小值和最大值。
- 广播时间间隔参数的取值范围为 0x0020 to 0x4000，默认值为 0x0800。对应的广播时间为参数值 * 0.625 ms，即广播时间间隔为 20 ms 到 10.24 s。
- 当 `adv_int_min` 和 `adv_int_max` 不同时，广播的时间间隔在两者区间内产生，当最小值和最大值设置成同一个值时，时间间隔固定为该值。

4.1.44 ESP32 经典蓝牙配对时如何使手机端输入 PIN 码？

可以通过禁用 `Secure Simple Pairing`，从而仅支持 `Legacy Pairing`。

- v3.3 到 v4.0（不包含 v4.0）：`Component config > Bluetooth > Bluedroid Enable > [*] Classic Bluetooth > [ ]Secure Simple Pairing`
- v4.0 及以上：`Component config > Bluetooth > Bluedroid Options > [ ] Secure Simple Pairing`

4.1.45 ESP32 蓝牙占用多少内存？

- 控制器：
  - BLE 单模：40 KB
  - BR/EDR 单模：65 KB
  - 双模：120 KB
- 主设备：
  - BLE GATT Client（Gatt Client 演示）：24 KB (.bss+.data) + 23 KB (heap) = 47 KB
  - BLE GATT Server（GATT Server 演示）：23 KB (.bss+.data) + 23 KB (heap) = 46 KB
  - BLE GATT Client & GATT Server: 24 KB (.bss+.data) + 24 KB (heap) = 48 KB
  - SMP: 5 KB

  - 经典蓝牙（经典蓝牙 A2DP_SINK 演示，包含 SMP/SDP/A2DP/AVRCP）：48 KB (.bss+.data) + 24 KB (heap) = 72 KB（示例运行时额外增加 13 KB）

注解： 以上堆 (Heap) 均包含任务栈 (Task Stack)，因为任务栈是从堆里分配出来的，算为堆。

- 优化 PSRAM 版本：

在 ESP-IDF V3.0 及以后，打开 `menuconfig` 里蓝牙菜单的 PSRAM 相关选项，将 Bluedroid(Host) 的部分.bss/.data 段及堆放入 PSRAM，可额外省出近 50 KB。

4.1.46 ESP32 使用 gattc_gatts_coex.c 例程测试 BLE 多连接，在 menuconfi 中将 BLE Max connection 配置选项设置为 “5”，但实际只能连 4 个设备，连接第 5 个设备的时候会报错，是什么原因？

- 请在 `menuconfig` 中将 `BT/BLE MAX ACL CONNECTIONS` 配置选项设置为 “5”。

4.1.47 ESP32-C3 BLE 同时支持主从模式吗？主、从模式连接数分别是多少？

IDF: release/v4.3, master

- ESP32-C3 同时支持主从模式，共用 8 个连接。例如，ESP32-C3 连接了 4 个 slave 设备，那么可被 8 - 4 = 4 个 master 设备连接。
- 另外，ESP32-C3 用作 slave 时，可被 8 个 master 设备连接；用作 master 时，可连接 8 个 slave 设备。

4.1.48 ESP32 经典蓝牙的 MTU Size 最大可设多大呢？

- ESP32 经典蓝牙有两种协议，分别为 A2DP 和 SPP 协议。BT A2DP 的 MTU Size 最大设置（默认）为 1008 字节，其中包头占 12 字节，应用层实际传输的数据量即为 1008 - 12 = 996（字节）；BT SPP 的 MTU Size 最大（默认）设置为 990 字节。

4.1.49 Wi-Fi 和蓝牙共存时，频繁通信出现 ELxXX error（比如 ELx200）如何解决?

CHIP: ESP32

- 该问题目前已在 commit 386a8e37f19fecc9ef62e72441e6e1272fa985b9 修补，请切换至对应的 commit 进行测试。

4.1.50 BLE 如何抓包？

- 市面上有很多工具可供选择，比如:
  - TI Packet sniffer
  - NRF Packet sniffer

4.1.51 使用 ESP32 开发板，测试好几个版本的 ESP-IDF 下的 BluFi 例程进行配网，点击配网之后都会打印如下报错，是什么原因？

```
E (117198) BT_L2CAP: l2ble_update_att_acl_pkt_num not found p_tcb
W (117198) BT_BTC: btc_blufi_send_encap wait to send blufi custom data
```

- 当出现此报错，请在 `components/bt/host/bluedroid/btc/profile/esp/blufi/blufi_prf.c` 文件下，把 `esp_ble_get_cur_sendable_packets_num(blufi_env.conn_id)` 换成 `esp_ble_get_sendable_packets_num()`。
- 此问题已经在所有分支上面进行修复，可以更新 ESP-IDF 为最新 Release 版本。

4.1.52 使用 ESP32，请问蓝牙能否使用 light-sleep 模式，并在 light-sleep 模式下保持蓝牙连接？

- ESP32 使用 light-sleep 模式，需要 ESP-IDF release/4.0 以上版本的 SDK 外加 32.768 kHz 晶振。
- light-sleep 模式下，蓝牙可以保持连接。请参考 Bluetooth modem sleep with external 32.768 kHz xtal under light sleep 指南。

4.1.53 如何修改 ESP32 的蓝牙广播名称？

- 要修改的结构体如下：

```
static uint8_t raw_adv_data[] = {

        /* flags*/

        0x02, 0x01, 0x06,

        /* tx power*/

        0x02, 0x0a, 0xeb,

        /* service uuid*/

        0x03, 0x03, 0xFF, 0x00,

        /* device name*/

        0x0f, 0x09, 'E', 'S', 'P', '_', 'G', 'A', 'T', 'T', 'S', '_', 'D','E
↪', 'M', 'O'

};
```

- 上述 `/* device name*/` 为修改项。其中 0x0f 为此字段类型加具体内容的总长度，0x09 表示此类型代指设备名。后续的‘E’，‘S’，‘P’，‘_’，‘G’，‘A’，‘T’，‘T’，‘S’，‘_’，‘D’,’ E’，‘M’，‘O’为广播设备名的 ASCII 码表达。

4.1.54 BLE 5.0 广播设置为 legacy 模式时支持最大广播长度为多少？

- 最大支持到 31-byte。

4.1.55 BLE 广播包如何设置为不可连接包?

CHIP: ESP32

- 可参考 gatt_server demo，将广播包类型 adv_type 变量修改为 ADV_TYPE_NONCONN_IND。

```
static esp_ble_adv_params_t adv_params = {
  .adv_int_min        = 0x20,
  .adv_int_max        = 0x40,
  .adv_type           = ADV_TYPE_NONCONN_IND,
  .own_addr_type      = BLE_ADDR_TYPE_PUBLIC,
  //.peer_addr            =
  //.peer_addr_type       =
  .channel_map        = ADV_CHNL_ALL,
  .adv_filter_policy = ADV_FILTER_ALLOW_SCAN_ANY_CON_ANY,
  }
```

4.1.56 怎样通过串口给 ESP32-WROOM-32D 模块直接发送蓝牙 HCI 命令?

- 请参考例程 controller_hci_uart_esp32。
- ESP32 用作 controller，其他设备作为 host，可通过 UART 给 ESP32 发送 HCI 指令。

4.1.57 ESP32 是否支持 A2DP 发送音频？

ESP32 支持 A2DP 发送音频，可参考例程 a2dp_source。

4.1.58 ESP32 Bluetooth LE 白名单最多支持多少个设备？

- 最多支持 12 个。

4.1.59 ESP32 低功耗蓝牙可以使用 PSRAM 吗？

请前往 `Component config > Bluetooth > Bluedroid Options` 开启 BT/BLE will first malloc the memory from the PSRAM 配置，即可让低功耗蓝牙使用 PSRAM。

4.1.60 使用 ESP32-C3 BLE Scan 时，是否可以设置仅扫描 Long Range 的设备？

- 可以，可基于 esp-idf/examples/bluetooth/bluedroid/ble_50/ble50_security_client 例程来测试。将 ext_scan_params 参数配置中 *.cfg_mask = ESP_BLE_GAP_EXT_SCAN_CFG_UNCODE_MASK | ESP_BLE_GAP_EXT_SCAN_CFG_CODE_MASK* 改为 *.cfg_mask = ESP_BLE_GAP_EXT_SCAN_CFG_CODE_MASK*，这样就可以仅扫描到 primary PHY 类型为 LE CODED PHY 的广播包。

4.1.61 ESP32 蓝牙设备名称长度是否有限制？

- ESP32 蓝牙设备名称长度不超过 248 字节，实际蓝牙设备名称受限于蓝牙广播数据包的长度。关于配置选项说明，请参见 CONFIG_BT_MAX_DEVICE_NAME_LEN。

4.1.62 使用 ESP32 如何设置 BLE Scan 永久扫描而不产生超时？

- 在使用 esp_ble_gap_start_scanning() 函数开始 BLE Scan 时，将 duration 参数设置为 0 即可。

4.1.63 如何通过 ESP32 获取 BLE 设备的 RSSI 的值？

- 可使用 esp_ble_gap_read_rssi() 函数来获取已连接的 BLE 设备的 RSSI 的值。
- 若需要获取周围扫描到的所有 BLE 设备的 RSSI 的值，请在 ESP_GAP_BLE_SCAN_RESULT_EVT 事件中使用 ble_scan_result_evt_param 结构体设置 RSSI 参数的打印。

4.1.64 如何增大 BLE5.0 传输距离？如何设置 BLE5.0 的 Long Range 模式？

- 在实际应用中，BLE5.0 的传输距离在 200 米左右，建议以实际测试距离为准。ESP32-S3 支持 BLE5.0 的特性，支持通过 Coded PHY（125 Kbps 和 500 Kbps）与广播扩展实现远距离 (Long Range) 通信。
- 您可以使用 125 Kbps Coded PHY 和增大发射功率 (tx_power)，来实现远距离通信。参考如下设置：

```
esp_ble_gap_ext_adv_params_t ext_adv_params_coded = {
  .type = ESP_BLE_GAP_SET_EXT_ADV_PROP_SCANNABLE,
  .interval_min = 0x50,
  .interval_max = 0x50,
  .channel_map = ADV_CHNL_ALL,
  .filter_policy = ADV_FILTER_ALLOW_SCAN_ANY_CON_ANY,
  .primary_phy = ESP_BLE_GAP_PHY_CODED,
```

(下页继续)

(续上页)

```
  .max_skip = 0,
  .secondary_phy = ESP_BLE_GAP_PHY_CODED,
  .sid = 0,
  .scan_req_notif = false,
  .own_addr_type = BLE_ADDR_TYPE_RANDOM,
  .tx_power = 18,
};
```

- BLE5.0 例程参见 ESP-IDF 里的 ble_50 示例。

4.1.65 基于 ESP32-C3 通过 esp_ble_gap_set_device_name() 更改了蓝牙名称，在 Android 设备上运行良好，并显示自定义设备名称。在 IOS 设备上，设备名称仍然是之前默认的蓝牙名称，怎样才能使它在 Apple 设备上也能正常工作？

- 此时需要使用 Raw data 的形式来创建 BLE 广播包，首先在 `menuconfig` 里使能 `CONFIG_SET_RAW_ADV_DATA` 选项（`idf.py menuconfig > Example 'GATT SERVER' Config > Use raw data for advertising packets and scan response data`），然后自定义 gatt server 示例 里的 广播包结构体 即可。
- 请使用 nRF Connect APP 进行测试。我们测试过，在 nRF connect APP 上是正常的，这种现象与 IOS APP 本身有关。

4.1.66 使用两块 ESP32 开发板测试蓝牙连接，使用 gatt_security_client 和 gatt_security_server 示例怎么设置指定密钥自动连接？

- gatt_security_client 和 gatt_security_server 示例默认设置的固定配对密钥就是 123456，参见代码 uint32_t passkey = 123456，您也可以自行设置为其他密码。
- 由于 ESP32 设备端默认没有显示器和输入键盘，因此例程将 IO 能力设置为 No output No input。您也可以参考 Gatt Security Server Example Walkthrough 来了解更多细节。
- 若要设置为可手动输入配对密钥，可将 gatt_security_server 示例中的 esp_ble_io_cap_t iocap 设置为 ESP_IO_CAP_OUT 模式，然后您可以使用 nRF Connect APP 与 BLE Server 建立连接。

4.1.67 将 gatt_security_server 设置为 ESP_IO_CAP_OUT 模式，并将 gatt_security_client 也设置为 ESP_IO_CAP_OUT 模式，然后故意设置错误的 passkey，但是还是能连接上，请问是什么原因？

- server 设置为 ESP_IO_CAP_OUT 模式时，gatt_security_client 应该设置为 ESP_IO_CAP_IN 模式。
- 同时需要在 gatt_security_client 端的 case ESP_GAP_BLE_PASSKEY_REQ_EVT 事件下，增加以下代码即可避免故意设置错误的 passkey 但是还是能连接上的情况：

```
esp_ble_passkey_reply(param->ble_security.ble_req.bd_addr, true, 123457);
```

4.1.68 ESP32-C3/ESP32-C6/ESP32-S3 是否支持蓝牙 AOA/AOD?

- ESP32-C3/ESP32-C6/ESP32-S3 均不支持蓝牙 AOA/AOD。目前，我们发布的产品都不支持蓝牙 AOA/AOD。

4.1.69 ESP32-C3 芯片使用 BLE5.0 特性支持的最大 BLE 广播数据包长是多大？

- ESP32-C3 BLE5.0 支持的最大广播数据包长为 1650 字节，可通过 esp_ble_gap_config_ext_adv_data_raw() API 进行设置。

4.1.70 ESP32 是否有 API 可用于检查设备 BLE 广播是否开始或停止？

- 如果使用的是 bluedroid 协议栈，目前没有 API 可用于检查。
- 如果使用的是 Nimble 协议栈（且使用的是 BLE 4.2 的非扩展广播），则可使用 ble_gap_adv_active API 来检查。

4.2 以太网

[English]

4.2.1 ESP32 以太网开发板示例出现“emac: Reset EMAC Timeout”有哪些原因？

此 log 为 EMAC 初始化超时，与 RMII 时钟有关，建议排查硬件问题，查看 PHY 晶振是否虚焊等。

4.2.2 ESP32 外接 LAN8720，GPIO0 对其提供 CLK，Ethernet 例程初始化出错。如何解决？

```
I (229) cpu_start: App cpu up.
I (247) heap_init: Initializing. RAM available for dynamic allocation:
I (254) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (260) heap_init: At 3FFB40A8 len 0002BF58 (175 KiB): DRAM
I (266) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (273) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (279) heap_init: At 400885D0 len 00017A30 (94 KiB): IRAM
I (285) cpu_start: Pro cpu start user code
I (303) cpu_start: Chip Revision: 1
W (303) cpu_start: Chip revision is higher than the one configured in␣
↪menuconfig. Suggest to upgrade it.
I (307) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
I (319) system_api: Base MAC address is not set, read default base MAC␣
↪address from BLK0 of EFUSE
E (1329) emac: Timed out waiting for PHY register 0x2 to have value␣
↪0x0007(mask 0xffff). Current value 0xffff
E (2329) emac: Timed out waiting for PHY register 0x3 to have value␣
↪0xc0f0(mask 0xfff0). Current value 0xffff
E (2329) emac: Initialise PHY device Timeout
ESP_ERROR_CHECK failed: esp_err_t 0xffffffff (ESP_FAIL) at 0x40084140
0x40084140: _esp_error_check_failed at /mnt/hgfs/workspace/esp32/IDF/esp-idf-
↪v3.3/components/esp32/panic.c:720

file: "/mnt/hgfs/workspace/esp32/project/ethernet/main/ethernet_example_main.
↪c" line 153
func: app_main
expression: esp_eth_enable()

ELF file SHA256:␣
↪``597d55ebf237c1cffa5f47c73148a159b22726d94a7b78100bd941d7d5fc906e``

Backtrace: 0x40083cdc:0x3ffb5e80 0x40084143:0x3ffb5ea0 0x400d32c1:0x3ffb5ec0␣
↪0x400d1742:0x3ffb5f20 0x40085d91:0x3ffb5f40
0x40083cdc: invoke_abort at /mnt/hgfs/workspace/esp32/IDF/esp-idf-v3.3/
↪components/esp32/panic.c:715
```

(下页继续)

(续上页)

```
0x40084143: _esp_error_check_failed at /mnt/hgfs/workspace/esp32/IDF/esp-idf-
↪v3.3/components/esp32/panic.c:721

0x400d32c1: app_main at /mnt/hgfs/workspace/esp32/project/ethernet/main/
↪ethernet_example_main.c:153 (discriminator 1)

0x400d1742: main_task at /mnt/hgfs/workspace/esp32/IDF/esp-idf-v3.3/
↪components/esp32/cpu_start.c:542

0x40085d91: vPortTaskWrapper at /mnt/hgfs/workspace/esp32/IDF/esp-idf-v3.3/
↪components/freertos/port.c:403


Rebooting...
```

- 请检查 IO0 上是否有电容。作为 CLK 输出 pin 的时候最好 IO0 上没有接电容，这会影响时序。
- GPIO0 输出 RMII 时钟切记在 Kconfig 中要勾选 `CONFIG_PHY_CLOCK_GPIO0_OUT`。
- 另外，以太网部分除了可以参考 example 中的 README 讲解，也可以参阅官方文档 API 参考。

4.2.3 使用 ESP-IDF 中的 Ethernet 示例时，出现错误代码 “Timed out waiting for PHY register 0x3 to have value 0xc0f0 (mask 0xfff0). Current value 0xffff”，请问该如何解决？

- 请参考：BBS issue 与 Github issue。
- 读 PHY 寄存器值为 0xFFFF 时，通常情况可以这样排查：
  a. 检查 MDIO 和 MDC 的接线是否错误
  b. 检查 RMII 需要的 50 MHz 时钟是否正常
  c. 检查 PHY 地址是否配置正确（包括软件和硬件）
- 强烈建议检查一遍控制 PHY 地址的 strap 引脚，保证其不要悬空，**不要默认**！确保这些 strap 引脚已经被外部电阻上拉或者下拉了。
- 如果还是不够确定 PHY 地址究竟是多少，可以在软件中尝试设置 PHY 地址从 0 开始到 31，然后读取 PHY ID 寄存器，看看是否能够读到正常的数据，如果正确，记录下当前 PHY 地址。

4.2.4 使用 ESP-IDF v4.1，ESP32 Ethernet 如何设置静态 IP？

由于 ESP-IDF v4.1 以及以上版本会摒弃掉 tcp/ip 的接口，推荐使用 ESP-NETIF 的接口。

参考示例代码如下：

```
{
    ...
    esp_netif_config_t cfg = ESP_NETIF_DEFAULT_ETH();
    esp_netif_t *eth_netif = esp_netif_new(&cfg);
    // Set default handlers to process TCP/IP stuffs
    ESP_ERROR_CHECK(esp_eth_set_default_handlers(eth_netif));
    ...
    char* ip= "192.168.5.241";
    char* gateway = "192.168.5.1";
    char* netmask = "255.255.255.0";
    esp_netif_ip_info_t info_t;
    memset(&info_t, 0, sizeof(esp_netif_ip_info_t));

    if (eth_netif)
    {
        ESP_ERROR_CHECK(esp_netif_dhcpc_stop(eth_netif));
        info_t.ip.addr = esp_ip4addr_aton((const char *)ip);
        info_t.netmask.addr = esp_ip4addr_aton((const char *)netmask);
        info_t.gw.addr = esp_ip4addr_aton((const char *)gateway);
        esp_netif_set_ip_info(eth_netif, &info_t);
    }
    ...
}
```

4.2.5 ESP32-Ethernet-Kit 开发板模组替换成 ESP32-WROOM-32D 以太网功能是否存在影响？

- ESP32-Ethernet-Kit 上的 ESP32-WROVER-B 可以更换成 ESP32-WROOM-32D，且以太网功能不受影响。
- ESP32-WROOM 和 ESP32-WROVER 系列模组最大的区别是：ESP32-WROVER 带有 4 MB PSRAM，而 ESP32-WROOM 默认没有 PSRAM。请参考：
  - ESP32-WROOM-32D 技术规格书
  - ESP32-WROVER-B 技术规格书
- ESP32-WROOM 和 ESP32-WROVER 模组都使用的是 ESP32 芯片，ESP32 芯片支持以太网，详情可以参考 ESP32 技术规格书。
- 相关文档：ESP32-Ethernet-Kit 入门指南。

4.2.6 使用 ESP32 设计自行开发以太网的板子，下载官方 esp-idf/examples/ethernet 例程，报错如下，是什么原因？

```
E (5556) emac: Timed out waiting for PHY rdgister 0x2 to have value 0x0022␣
↪(mask 0xffff). Current value 0xffff
E (6556) emac: Timed out waiting for PHY register 0x3 to have value 0x1430␣
↪(mask 0xfff0). Current value 0xffff
```

- 此报错说明硬件电路有问题，RMII 时钟没有正常供给 PHY，遇到读 PHY 寄存器失败。关于 RMII 时钟部分，可参见 说明。

4.2.7 ESP32 以太网支持 MII 接口吗？

- 硬件支持，软件正在适配中，用户自行实现可参考 Ethernet doc。

4.2.8 ESP32-S2 是否可以外接以太网？

- 可以，目前 ESP-IDF 已经提供了 DM9051 模块的驱动，该模块内部集成以太网的 MAC 和 PHY 功能，可以和 MCU 之间通过 SPI 接口进行通讯。DM9051 上集成了 MAC+PHY 的模块，请参考 参考示例 以及 使用说明。

4.2.9 ESP32 是否支持 EMAC 与 SPI 以太网模块同时使用？

- 支持，ESP32 可同时使用 EMAC 和一至两个 SPI 以太网模块。可基于 esp-idf/examples/ethernet/basic 例程在 menuconfig 中同时开启 PHY 和 SPI 以太网进行测试。

4.3 共存

[English]

4.3.1 Wi-Fi 和蓝牙共存时，支持哪些共存场景？

支持的共存场景请参考 共存文档。

4.3.2 Wi-Fi 和 ESP-BLE-MESH 共存时，为什么 Wi-Fi 吞吐量很低？

未搭载 PSRAM 的 ESP32-DevKitC 开发板，Wi-Fi 和 ESP-BLE-MESH 共存可以正常运行，但是吞吐率较低。当 Wi-Fi 和 ESP-BLE-MESH 共存时，搭载 PSRAM 的 ESP32-DevKitC 速率可以稳定在 1 Mbps 以上。

应使能 menuconfig 中的一些配置来支持 PSRAM:

- `ESP32-specific --> Support for external,SPI-connected RAM --> Try to allocate memories of Wi-Fi and LWIP...`
- `Bluetooth --> Bluedriod Enable --> BT/BLE will first malloc the memory from the PSRAM`
- `Bluetooth --> Bluedriod Enable --> Use dynamic memory allocation in BT/BLE stack.`
- `Bluetooth --> Blutooth controller --> BLE full scan feature supported.`
- `Wi-Fi --> Software controls Wi-Fi/Bluetooth coexistence --> Wi-Fi`

4.3.3 ESP32 的 ESP-WIFI-MESH 和 Bluetooth® LE Mesh 可以同时支持吗？

不支持。

ESP32 的 ESP-WIFI-MESH 和 BLE 可以同时支持，或者 ESP32 Wi-Fi STA 模式和 BLE Mesh 可以同时支持。

4.3.4 ESP32 蓝牙和 Wi-Fi 能否同时使用？

ESP32 的 Wi-Fi 和蓝牙可共存，但需要分时控制，可在 menuconfig 中使能 Wi-Fi 和蓝牙共存设置。如下：

```
menuconfig -> Component config -> Wi-Fi -> Software controls WiFi/
Bluetooth coexistence (Enable)
```

4.3.5 Wi-Fi、Bluetooth® LE 和 A2DP Sink 共存，进入 Bluetooth LE 扫描的时候音频数据接收会丢失、卡顿。怎么解决？

- 使用 RingBuf 缓存音频数据
- 暂停播放音乐，并增加提示音，例如：正在扫描设备。

4.3.6 BLE adverting (Connectable) + iBeacon sending(advertising) 可以共存吗 ??

IDF: release/v4.0 以及以上版本 | CHIP: ESP32

- 硬件上还未支持，应用层可以通过定时轮询发广播包的方式来完成。

IDF: release/v4.3 以及以上版本 | CHIP: ESP32-C3|ESP32-S3

- 可以。

4.4 外设

[English]

4.4.1 模拟数字转换器 (ADC)

[English]

ESP8266 ADC 的分辨率如何？

- ESP8266 ADC 为 10 位，理论分辨率为 $2^{10} = 1024$。
- ESP8266 连接路由器后，单 STA 模式会进入 Modem-sleep 模式，导致芯片内部参考值变化，因此 ADC 测量得数据变化。
- 如果想要测量精确，可以在关闭 Wi-Fi 后，使用 `system_adc_fast_read` 函数读取。

ESP8266 如何获取 ADC 寄存器位图信息？

由于 ESP8266 ADC 是和内部 RF 电路高度集成的，所以位图和寄存器信息没有公开，如有特殊需求请联系 sales@espressif.com。

ESP32 ADC 有几个通道？采样率和有效位数是多少？

- ESP32 的 ADC 共有 18 个通道。
- 在停止 Wi-Fi 的情况且使用 ADC DMA 的情况下，采样率理论不超过 2 MHz。但实际建议使用更小的采样率。
- 在 Wi-Fi 正常工作的情况下，能达到每秒 1000 次。
- ADC 内部有效位数为 12 位。

使用 ESP8266 调用 `adc_read_fast()` API 会导致 Wi-Fi 断连吗？

- 调用 `adc_read_fast()` API 前需要将 Wi-Fi 和中断关闭，可参见此 API 的 使用说明。
- 由于 `adc_read_fast()` API 会进行连续采集，ADC 内部与 Wi-Fi RF 存在耦合部分，无法在 Wi-Fi 开启的状态下调用该函数。
- 在 Wi-Fi 开启的时候请使用 `adc_read()` API 进行 ADC 采集。为保证数据稳定，需要使用 `esp_wifi_set_ps(WIFI_PS_NONE)` 函数关闭 Wi-Fi Modem-sleep 休眠模式。

注解： ADC 采样率：在停止 Wi-Fi 的情况下，能达到每秒 100000 次。Wi-Fi 正常工作的情况下，能达到每秒 1000 次。

悬空 ADC 引脚，打印出 VDD3P3 的值为 65535，那么 VDD3P3 的电压就是 65535/1024 ≈ 63 V。这个电压值不符，是什么原因？

ADC 输入范围需要大于 0 V 小于 3.3 V（不同型号芯片上限不同），悬空测量为未定义状态。

ESP32 ADC 的输入电阻是多少？

ADC 是电容性的，可以认为电阻很大。

使用 ESP32 的 ADC 来检测电源电压，是否需要进行分压？

ESP32 的 ADC 参考电压为 1100 mV，可以通过配置内部衰减来增大 ADC 量程，量程范围可参考芯片手册 ADC 章节 如量程无法满足需求，可使用外部分压电路。

ESP32 芯片 ADC DMA 模式最高支持多大的采样频率？

理论最高支持 2 MHz 的采样频率。

使用 ESP32，在 `esp_wifi_start()` 和 `esp_wifi_stop()` 之间读取 `adc2_get_raw()` 操作失败，是什么原因？

由于 Wi-Fi 也需要使用 ADC2，且 Wi-Fi 具有更高的访问优先级。因此，在 Wi-Fi 工作期间，应用程序使用 `adc2_get_raw()` 可能读取失败，建议检测该函数的返回值，失败后重新进行一次测量。

ESP32 是否支持 ADC2 与蓝牙同时使用？

支持。

ESP32-S2 芯片 ADC DMA 模式支持的采样率范围是多大？

频率限制：611 Hz ~ 83333 Hz。

ESP32 的 ADC 支持多通道同时采样吗？

ESP32 的 ADC 不支持多通道同时采样，若使用 ADC 多通道采样，需采取轮询采样的方式来实现。

使用 ESP32-WROVER-B 模组，ESP-IDF 为 release/v4.2 版本，当 GPIO 设置为 ADC 接口后，在不进行硬件重启的情况下，再将 GPIO 设置为其他 IO 模式且其他 IO 模式不生效后，此 GPIO 无反应，请问如何释放对应的 GPIO 模式？

- 请不要将 ADC 接口设置为只输入的 GPIO。
- 结束 ADC 接口模式时，请使用 adc_digi_stop() 关闭 ADC。

ESP32 芯片的 ADC 之间的测量误差是多大？

默认情况下，ESP32 芯片 ADC 之间的测量差异是 ±6%，可参考 《ESP32 技术规格书》 。

ESP32 能同时用两个 ADC 通道来测量不同的数据吗？比如电流和电压？

使用一个 ADC 无法做到同一时刻读取多个 ADC 通道的值，可以依次轮询读取两个 ADC 通道的数据。

ESP32-S3 ADC 配置为 `ADC_ATTEN_DB_11` 时，为何测量电压无法达到标称 3100 mV？

ESP32-S3 ADC1 或 ADC2 配置为 `ADC_ATTEN_DB_11` 时，测量电压范围为 `0 ~ 3100 mV`，但部分芯片最大电压测量值小于 `3100 mV`，可使用以下两种方法来解决这个问题：

- 方案 1：避开使用边界电压值，可通过外部分压电路将输入电压维持在中间电压值附近，以获得更高的精度和一致性。
- 方案 2：使用软件 ADC 扩展量程方案 ，将最大测量电压扩展到 `3300 mV`。

我们可以使用 GPIO0 作为 ADC 管脚，同时将 ESP32 作为 Wi-Fi 热点吗？

- 当使用 Wi-Fi 时，ESP32 ADC2 管脚不能同时使用。因此，如果您在使用 Wi-Fi 时无法从 ADC2 GPIO 获取值，您可以考虑改用 ADC1 GPIO，这应该可以解决您的问题。有关详细信息，请参阅 ADC 连续模式的硬件限制 和 ADC Oneshot 模式的硬件限制。
- GPIO0、GPIO2、GPIO5、GPIO12 (MTDI) 和 GPIO15 (MTDO) 是 strapping 管脚。将 GPIO0 用于其他功能时，需要注意上电时的 GPIO 电平。如果上电时 GPIO0 为低电平，则很容易进入下载模式。有关更多信息，请参阅 ESP32 技术规格书。

使用 ESP32-S3 的 GPIO19 和 GPIO20 基于“esp-idf/examples/peripherals/adc/oneshot_read”例程测试 ADC2 功能，ADC2 衰减参数设置为 11 dB，当输入电压为 0.6 V 时，测试结果却为 1.1 V 和 2.8 V，是什么原因？

- 建议检查是否两个 ADC2 通道都进行了 adc_oneshot_config_channel() 的配置。

4.4.2 数字模拟转换器 (DAC)

[English]

ESP32-S2-Saola-1 使用 DAC 输出时，采用 3.3 V 进行供电，为什么实际测试电压只有 3.1 V？

由于存在内部压降，即使使用 3.3 V 供电，实际最大输出只有 3.2 V 左右。

4.4.3 GPIO & RTC GPIO

[English]

ESP32 管脚配置需要注意什么？

ESP32 系列模组分为 ESP32-WROOM 系列和 ESP32-WROVER 系列，GPIO 使用配置注意事项如下。

WROOM-32/32D/32U 系列共有 26 个管脚可供客户使用，注意事项如下：

- GPIO6 ~ GPIO11 被内置 flash 占用，不可用做它用；
- GPIO34、35、36 和 39 为输入管脚，不具备输出能力；
- ESP32 内置 GPIO 矩阵，部分外设接口可以配置到任意空闲管脚上。即硬件设计时，不需要严格将某些功能固定在某些管脚上。

详细信息可以参考 《ESP32 技术规格书》 中表格 9 的内容。

WROVER / WROVER-I / WROVER-B / WROVER-IB 共有 24 个管脚可供客户使用，注意事项如下：

- GPIO6 ~ GPIO11 被内置 flash 占用，不可用做它用；
- GPIO34、35、36 和 39 为输入管脚，不具备输出能力；
- WROVER 系列模组中，GPIO12 由于在模组内部被上拉，不建议用做触摸传感功能；
- ESP32 内置 GPIO 矩阵，部分外设接口可以配置到任意空闲管脚上。即硬件设计时，不需要严格将某些功能固定在某些管脚上。

详细信息可以参考 《ESP32 技术规格书》 中表格 9 的内容。

ESP32 有 3 组 UART，但下载只可使用 UART0，且管脚固定。

ESP8266 部分 GPIO 出现高电平的原因是什么？

- 根据硬件设计，部分 GPIO 存在默认上下拉状态，所以在系统初始化时，该管脚的电平状态不受程序控制，所以会出现程序在引导过程中部分 GPIO 电平不正确。
- 如果需要使用这些 GPIO，硬件上建议外接器件与默认上下拉电平一致，软件可以在引导加载程序过程中调整电平状态，软件方法也会存在短暂电平异常。

ESP32 是否可以关闭线程调度使用一个单独的 CPU 以实现 GPIO 实时控制？

- 目前 SDK 没有相关的配置选择供 CPU1 单独运行，两个核心只支持 SMP，不支持 AMP。
- 解决输出波形被打断的问题有以下解决方案:
  - 使用硬件的信号输出，选择相关数字协议实现 SPI、I2C、I2S 等，特殊用法 SPI 取信号输出线产生波形。
  - 硬件 RMT 是否可以产生想要的波形，并达到足够的长度。
  - 硬件中断中产生相应波形，需要将所有回调放入 IRAM 中。
  - 可以利用芯片中的协处理器，它可以当作无操作系统的单片机。

ESP32 GPIO 电平翻转速度是多少？

GPIO 电平翻转大约耗时 300 ns。

ESP32 当一些 RTC 外设的电源打开时（SARADC1、SARADC2、AMP、HALL 传感器），GPIO36 和 GPIO39 的数字输入会被拉低约 80 ns，如何解决？

- 对于需要精确计时和检测数字输入状态的应用，可通过软件避开以上问题：- 控制以上传感器的电源域打开时，忽略来自 GPIO36 和 GPIO39 的输入。- 通过软件实现数字输入的去抖动：在读取 GPIO36 和 GPIO39 输入状态时，可以在软件层面实现去抖动，对输入状态进行多次采样和滤波，从而减少电压短暂下降所导致的错误判断。

ESP32 如果多个 GPIO 管脚配置了沿中断，则硬件可能无法正确触发中断。如何解决？

- 请在 《ESP32 系列芯片勘误表》 中查找该问题及答案。

使用 ESP-WROOM-02D 模组，GPIO0、GPIO15、GPIO1 和 GPIO3 是否可作为普通 GPIO 使用?

- Strapping 管脚（GPIO0 和 GPIO15）和下载管脚（GPIO1 和 GPIO3）可以作为普通 GPIO 使用。
- 使用 Strapping 管脚作为普通 GPIO 使用时，在 flash 下载模式时需要注意 Strapping 管脚电平的要求。

ESP32-C3 系列芯片将 GPIO19 配置成输入下拉时，读取该 IO 口状态依旧显示高电平，但配置 ESP32-C3 的其他管脚或者其他芯片的管脚为输入下拉时，均正常显示为低电平？

- ESP32-C3 的 GPIO19 为 USB D+ 管脚，USB 管脚的上拉电阻由管脚上拉和 USB 上拉共同控制，当其中一种上拉方式为 1 时，对应的上拉电阻就会使能。
- GPIO19 是默认 USB 上拉使能的，因此配置了管脚为输入下拉后依旧是上拉使能，管脚显示高电平。
- v4.4.3 及以上版本 GPIO 驱动已经修复该问题，如果您在使用较低版本的 ESP-IDF，请直接将 `USB_SERIAL_JTAG_DP_PULLUP` 寄存器写为 0 进行配置。

使用 ESP-IDF release/v4.2 版本的 SDK，ESP32 如何设置单个 GPIO 同时作为输入/输出模式？

可使用 esp_err_t gpio_set_direction(gpio_num_t gpio_num, gpio_mode_t mode) API 来设置。

ESP-IDF 里是否能设置 GPIO 的驱动强度？

可以。请使用 API gpio_set_drive_capability 来设置 GPIO 驱动强度。

ESP32 使用 gpio_install_isr_service() 初始化新的 GPIO 中断服务时返回 *ESP_ERR_NOT_FOUND*，可能是什么原因？

这个错误通常代表 ESP32 的可用中断源不够用，此时应该同时有多个外设在同时占用中断源，可尝试减少其他组件的中断源使用个数来初始化新的 GPIO 中断。

如何获取 ESP32 RTC_GPIO 的输入电平状态？

- 可读取 RTC GPIO 对应的寄存器地址的宏来获取 RTC_GPIO 的输入电平状态，可参考 “esp-idfcomponentssocesp32includesocrtc_io_reg.h” 。
- 对应的代码参考如下：

```
uint8_t level = (uint8_t)((REG_GET_FIELD(RTC_GPIO_IN_REG, RTC_GPIO_IN_NEXT) &
↪ BIT(gpio_num)) ? 1 : 0);
```

4.4.4 I2C 驱动程序

[English]

ESP8266 是否支持 I2C 从机模式？

不支持，如果要使用此功能，推荐使用 ESP32 或者 ESP32-S2 芯片。ESP32 参考示例：i2C_self_test。

ESP8266 I2C 是软件模拟的吗？

ESP8266 I2C 是使用 GPIO 软件模拟。

当 ESP32 系列芯片 I2C 在工作时（尤其是处于快速模式时），数据线上经常出现一些尖峰，往往在第 8/9 个时钟的下降沿之后，这是否正常？

发生在 8/9 个时钟时数据线上的尖峰是由于 I2C 主从机控制权交接导致的，属于 I2C 协议里提到的正常现象。

ESP32 系列芯片作为 I2C 主机怎样才能等待从机处理数据后再接收？例如通过 `i2c_master_read_to_device` 读取数据时，需要从机接受命令后立即返回数据，但是实际的一些从机设备接收到命令后需要等待一段时间才能返回数据。

可以自行将 `i2c_master_read_device` 拆分成三部分进行实现：

1. 写命令和地址：`i2c_cmd_link_create_static` > `i2c_master_start` > `i2c_master_write_byte` > `i2c_master_cmd_begin` > `i2c_cmd_link_delete_static`

2. 延时
3. 读从机数据：`i2c_cmd_link_create_static` > `i2c_master_read` > `i2c_master_stop` > `i2c_master_cmd_begin` > `i2c_cmd_link_delete_static`

使用 ESP32 系列芯片时，能否将 GPIO32 和 GPIO33 分别配置为 I2C_SDA 和 I2C_SCL？

可以，ESP32 的 I2C 管脚可以使用任何空闲的 GPIO 进行重映射。请参阅 ESP32 技术规格书 的 4.2 小节，外设引脚配置部分。如果您不需要外部 32.768 KHz 晶振，那么您可以使用 GPIO32 和 GPIO33 作为 I2C 管脚。

4.4.5 集成电路内置音频总线 (I2S)

[English]

ESP32 是否支持使用晶振作为 I2S 的时钟源？

ESP32 不支持使用晶振作为 I2S 的时钟源，可阅读 《ESP32 技术参考手册》 来了解 I2S 的时钟源配置。

若 I2S 从设备只有 I2S_DATA、I2S_SCK 和 I2S_WS 这三根信号线，ESP32 作为 I2S 主设备时，是否支持这种连接方式？

支持，但是是否接 MCLK 要看对端编码解码芯片的要求。

ESP32-C3 的 I2S 接口是否支持 PDM RX 模式？

- 在软件驱动上，ESP32-C3 的 I2S 接口不支持使用 PDM RX 模式。与 ESP32-S3 不同，ESP32-C3 的 I2S 接口的 PDM RX 没有 PDM to PCM 格式转换的模块，这意味着读取的数据是 RAW PDM 格式，此格式的数据在大多数情况下不能直接使用。

4.4.6 LCD

[English]

ESP32 系列芯片 LCD 驱动及参考例程在哪？

- ESP 的 LCD 驱动位于 **ESP-IDF** 下的 components/esp_lcd，目前仅存在于 **release/v4.4 及以上**版本中。**esp_lcd** 能够驱动 ESP 系列芯片所支持的 **I2C**、**SPI**、**8080** 以及 **RGB** 四种接口的 LCD 屏幕，各系列芯片所支持的 LCD 接口见 ESP32 系列芯片的屏幕接口。
- 各接口的 LCD 驱动应用示例参考 ESP-IDF 下的 examples/peripherals/lcd，这些示例目前仅存在于 **release/v5.0** 及以上版本中，因为 **release/v4.4** 中 esp_lcd 的 API 名称与高版本基本一致，所以同样可以参考上述示例（两者的 API 实现上有一些区别）。
- **不推荐**使用 esp-iot-solution 中的 LCD 驱动及例程。
- RGB LCD 应用推荐使用 ESP-IDF **release/v5.1**，因为 release/v4.4 中不支持部分特性。

ESP32 系列芯片 LCD 屏幕适配情况是怎样的？

目前基于 *esp_lcd* 驱动适配过的 LCD 驱动 IC 如下：

- esp_lcd：st7789、nt35510、ssd1306
- 包管理器：gc9a01、ili9341、ili9488、ra8875、sh1107、st7796（持续更新中）

需注意，即使驱动 IC 相同，不同的屏幕往往需要不同的寄存器配置参数，而且屏幕厂商通常会给配套的配置参数（代码），因此推荐利用上面两种途径获取相似驱动 IC 的代码，根据自己屏幕的实际参数进行修改。

目前基于 *esp_lcd_touch* 驱动适配过的 Touch 驱动 IC 如下：

- 包管理器：FT5x06、GT1151、GT911、STMPE610、TT21100、XPT2046、CST816（持续更新中）

如何提高 LCD 的显示帧率？

- LCD 的实际显示帧率由“接口帧率”和“渲染帧率”共同决定，一般来说，受限于 ESP 的计算性能，“接口帧率”往往远大于“渲染帧率”，因此该问题可以表述为“如何提高 LCD 的渲染帧率”。
- 以下 ESP 配置项对帧率提升有帮助（IDF release/v5.0）：
  - CONFIG_FREERTOS_HZ=1000

  - CONFIG_ESP_DEFAULT_CPU_FREQ_MHZ_240=y
  - CONFIG_ESPTOOLPY_FLASHMODE_QIO=y
  - CONFIG_ESPTOOLPY_FLASHFREQ_120M=y [需要与 PSRAM 保持一致]
  - CONFIG_SPIRAM_MODE_OCT=y
  - CONFIG_SPIRAM_SPEED_120M=y [需要与 FLASH 保持一致]
  - CONFIG_SPIRAM_FETCH_INSTRUCTIONS=y
  - CONFIG_SPIRAM_RODATA=y
  - CONFIG_ESP32S3_DATA_CACHE_LINE_64B=y
  - CONFIG_COMPILER_OPTIMIZATION_PERF=y
- 以下 LVGL 配置项对帧率提升有帮助（LVGL v8.3）：
  - #define LV_MEM_CUSTOM 1
  - #define LV_MEMCPY_MEMSET_STD 1
  - #define LV_ATTRIBUTE_FAST_MEM IRAM_ATTR

ESP32 是否有 I2S 驱动 LCD 的参考代码？

- ESP32/ESP32-S2 使用 I2S 驱动的屏幕接口类型为 i80(8080)
- 关于例程，请参考*LCD 例程*。

ESP32 LCD 最大可以支持多大的分辨率？相应的帧率是多少？

- 可以支持多大的分辨率并没有一个“最大”的限制，由于外设接口的数据传输带宽有限，LCD 分辨率越高，接口帧率就会越低，因此需要结合两者共同进行判断。
- ESP32 LCD 在 RGB 接口下，目前测试过的最大分辨率为 800 × 480，接口帧率上限为 59 (PCLK 30 MHz)，对应 LVGL 平均帧率为 23; LVGL 平均帧率上限为 26, 对应接口帧率为 41 (PCLK 21 MHz)。

ESP32R8 如何开启 PSRAM 120M Octal(DDR)？

- ESP-IDF 需要使用 **release/v5.1** 及以上分支版本。
- 参考 文档。
- **需注意**，该特性是一种实验功能并具有以下温度风险：- 在温度高于 65°C 的情况下，即使开启 ECC 功能也无法保证正常工作。- 温度变化也可能导致访问 PSRAM/flash 时程序崩溃，具体参考 文档。

使用 ESP32-S3 测试 LVGL 例程，请问目前已经适配了哪些型号的显示触摸屏？

不推荐使用 esp-iot-solution 中的驱动和例程。关于例程，请参考*LCD 例程*。

ESP32-S3 使用 RGB 屏幕必须要外接 PSRAM 吗？

是的，而且至少是 8 线 PSRAM 并且需要配置时钟为 80 MHz 及以上，否则将导致 RGB LCD 的 PCLK 无法设置到较高的 PCLK 频率同时帧率过低。

ESP32-S3 系列的芯片支持哪些图片解码格式？

- 目前官方仅支持 JPEG 解码格式，应用例程可参考 esp-idf/examples/peripherals/lcd/tjpgd。
- 基于 LVGL 开发的话，可以支持 PNG、BMP、SJPG、GIF 图片解码格式，具体介绍见 LVGL libs。

为什么驱动 RGB LCD 屏幕时出现偏移（显示画面整体漂移）？

- **原因**
  - PCLK 设置过高，PSRAM 带宽跟不上。
  - 受写 flash 操作影响，期间 PSRAM 被禁用。
- **配置方面**
  - 提高 PSRAM 和 flash 带宽，设置 flash 为 QIO 120 M，PSRAM 为 Octal 120 M。
  - 开启 *CONFIG_COMPILER_OPTIMIZATION_PERF*。
  - 降低 data_cache_line_size 到 32 Byte。
  - 开启 *CONFIG_SPIRAM_FETCH_INSTRUCTIONS* 和 *CONFIG_SPIRAM_RODATA*。

- 开启 *CONFIG_LCD_RGB_RESTART_IN_VSYNC*，可能会导致闪花屏和降帧率，一般不推荐，可以尝试。

- **应用方面**
  - 如果需要使用 Wi-Fi 和连续写 flash 的操作，请采用 *XIP PSRAM + RGB Bounce buffer* 的方法，设置步骤如下：
    * 确认 ESP-IDF 版本为较新（> 2022.12.12）的 release/v5.0 及以上，因为旧版本不支持 *XIP PSRAM* 的功能。
    * 确认 PSRAM 配置里面是否能开启 *SPIRAM_FETCH_INSTRUCTIONS* 和 *SPIRAM_RODATA* 这两项（如果 rodata 段数据过大，会导致 PSRAM 空间不够）。
    * 确认内存（SRAM）是否有余量，大概需要占用 [10 * screen_width * 4] 字节。
    * 需要将 *Data cache line size* 设置为 64 Byte（可设置 *Data cache size* 为 32 KB 以节省内存）。
    * 如以上均符合条件，那么就可以参考 文档 修改 RGB 驱动为 *Bounce buffer* 模式。
    * 如操作 Wi-Fi 仍存在屏幕漂移问题，可以尝试关闭 PSRAM 里 CONFIG_SPIRAM_TRY_ALLOCATE_WIFI_LWIP 一项（会占用较大 SRAM）。
    * 设置后带来的影响包括：CPU 使用率升高、可能会造成中断看门狗复位、会造成较大内存开销。
  - 短时操作 flash 导致漂移的情况，如 wifi 连接等操作前后，可以在操作前调用 *esp_lcd_rgb_panel_set_pclk()* 降低 PCLK（如 6 MHz）并延时大约 20 ms（RGB 刷完一帧的时间），然后在操作结束后提高 PCLK 至原始水平，期间可能会造成短暂的闪白屏现象。
  - 使能 *esp_lcd_rgb_panel_config_t* 中的 *flags.refresh_on_demand*，通过调用 *esp_lcd_rgb_panel_refresh()* 接口手动刷屏，在保证屏幕不闪白的情况下尽量降低刷屏频率。
  - 如果无法避免，可以调用 *esp_lcd_rgb_panel_restart()* 接口重置 RGB 时序，防止永久性漂移。

为什么驱动 SPI/8080 LCD 屏幕显示 LVGL 时出现纵向错位？

如果采用 DMA 中断传输的方式，LVGL 的 `lv_disp_flush_ready` 需要在 DMA 传输结束后调用，而不是 `draw_bitmap` 后立即调用。

使用 ESP32-C3 通过 SPI 接口驱动 LCD 液晶显示屏，是否可使用 RTC_CLK 作为 SPI 时钟，让 LCD 液晶显示屏能在 Deep-sleep 模式下正常显示静态图片？

- Deep-sleep 模式：CPU 和大部分外设都会掉电，只有 RTC 存储器处于工作状态。具体请参考《ESP32-C3 技术规格书》 中关于“低功耗管理”的说明.
- ESP32-C3 的 SPI 只支持 APB_CLK 和 XTAL_CLK 两种时钟源，不支持使用 RTC_CLK。因此在 Deep-sleep 模式下，LCD 液晶屏无法显示静态图片。具体请参考 《*ESP32-C3* 技术参考手册》 > 复位和时钟 [PDF]。
- 对于 SPI 接口驱动的 LCD 屏幕，一般来说驱动 IC 内置 GRAM，不需要 ESP 持续输出 SPI 时钟的就能正常显示静态图片，只是期间画面无法更新。

使用 ILI9488 LCD 屏幕测试屏幕例程，是否支持 9-bit 总线和 18-bit 色深？

ILI9488 驱动芯片可以支持 9-bit 总线和 18-bit 色深，但目前我们的驱动只支持 8-bit 总线和 16-bit 色深。可根据 ILI9488 数据手册自行修改驱动，来实现 9-bit 总线和 18-bit 色深的支持。

4.4.7 LED PWM 控制器 (LEDC)

[English]

ESP8266 PWM 频率范围是多少呢？

ESP8266 PWM 是软件模拟的，受定时器限制 CLK 最大为 1 M。推荐频率为 1 K，也可以通过降低占空比分辨率的方式提高频率。

ESP32 GPIO 管脚输出 PWM 存在限制吗？是否可以分配至任意一个 I/O 上？

- ESP32 PWM 可通过 IO Matrix 切换至任意 GPIO 输出。除了只有输入功能的 I/O（例如：GPIO34 ~ GPIO39）之外，理论上 PWM 可以输出到任何管脚。
- 实际使用中仍会受到模组与芯片限制、模组未引出管脚或 flash 占用等情况影响。

ESP8266 NonOS SDK PWM 的变化缓慢，有哪些原因？

- 如果使用 SDK example/IOT_demo 中的渐变 API，如 `light_set_aim` 或 `light_set_aim_r` 这些 API，需要渐变的过程。
- 若需要 PWM Duty 设置后立即生效，则可以调用接口 `pwm_set_duty`，需要注意调用 `pwm_set_duty` 后要调用 `pwm_start` 此次设置才能生效。

ESP32 LEDC 递减渐变，Duty 值溢出错误，如何解决？

使用 LEDC 的过程中，应避免以下三个条件同时成立：

- LEDC 启动递减渐变功能；
- LEDC 渐变过程中 Scale 寄存器设置为 1；
- LEDC 递减渐变开始时刻或者过程中的某一时刻，Duty 值为 $2^{\text{LEDC\_HSTIMERx\_DUTY\_RES}}$ 或 $2^{\text{LEDC\_LSTIMERx\_DUTY\_RES}}$。

ESP8266 通过直接写硬件定时器 FRC1 的寄存器产生 PWM，发现初始化 Wi-Fi 时，Wi-Fi 产生的中断会干扰硬件定时器的中断，导致错误的 PWM 输出，是否可以使用 FRC2 产生 PWM？是否可以使 FRC1 的优先级高于 Wi-Fi？

不可以使用 FRC2，其被系统占用。Wi-Fi 使用 NMI 中断，其优先级高于其他普通中断，推荐使用 ESP8266 RTOS SDK 的 PWM 库，参考 ESP8266_RTOS_SDK/examples/peripherals/pwm。

使用 v3.3.3 版本 ESP-IDF 在 ESP32 设备上测试 ledc 例程，当启用了 auto light sleep，LED PWM 无输出；但不启用 auto light sleep，LED PWM 有输出。ESP-IDF 编程指南里关于 LED PWM 的说明表示 LED PWM 在 Sleep 模式下是能工作的，请问是什么原因？

- esp-idf v3.3.3 版本的 SDK 不支持 LED PWM 在 Sleep 模式下工作。请使用新版的 esp-idf（v4.0 以上版本）下的 LEDC 例程来测试，例如 esp-idf release/v4.2 版本的 SDK，且需要将 LED PWM 时钟源改为内部 RTC8M 时钟源。如下：

```
ledc_timer_config_t ledc_timer = {
      .duty_resolution = LEDC_TIMER_13_BIT,
      .freq_hz = 5000,
      .speed_mode = LEDC_LOW_SPEED_MODE,
      .timer_num = LEDC_TIMER_0,
      .clk_cfg = LEDC_USE_RTC8M_CLK,
  };
```

ESP32 PWM 支持两路死区互补输出吗？

- LEDC 不支持，MCPWM 外设支持两路死区互补输出。
- 实测 ESP32-S3 可以通过 MCPWM 产生频率 10 k、占空比精度 1 us、死区精度 100 ns 的互补输出波形。

4.4.8 电机控制脉宽调制器 (MCPWM)

[English]

ESP32 支持使用 MCPWM 的定时器来触发 AD 采样吗？

不支持。

ESP32-S3 能够产生完全互补的 PWM 吗，要求时钟精确，占空比精确，死区可调节？

实测 ESP32-S3 可以通过 MCPWM 产生频率 10 k、占空比精度 1 us、死区精度 100 ns 的互补输出波形。

4.4.9 脉冲计数器 (PCNT)

[English]

ESP8266 可以实现脉冲计数吗？

- ESP8266 未包含硬件脉冲计数模块，所以仅支持通过 GPIO 上升沿或下降沿中断实现脉冲计数。
- ESP8266 芯片中 Wi-Fi 开启后由于优先级太高可能会导致 GPIO 采样出现真空，中断采集的计数丢数据。
- 综上，在计数要求较为严格的场景推荐使用 ESP32 以及后续推出的芯片。

ESP32-S3 可以通过 PCNT 实现频率为 200 k 的低电平脉冲计数吗？

可以。

4.4.10 红外遥控接收器 (RMT)

[English]

ESP 芯片上的 RMT 外设有哪些实际的应用场景？

- 请参考 RMT 应用示例，可以实现红外遥控，LED 灯带点亮，D-shot 电机控制等。

如果要使用 RMT 功能，最推荐使用哪一款 ESP 芯片？

- 推荐 ESP32-S3，因为目前只有 ESP32-S3 这一款芯片的 RMT 带有 DMA。这样 RMT 可以避免 Wi-Fi 或蓝牙等中断的干扰。

RMT 中如何将时钟修改为 REF_TICK?

CHIP: ESP32 | ESP32-S2 | ESP32-C3

可以调用 rmt_set_source_clk 接口设置。

使用 ESP32 RMT 控制 WS2812 灯带，当与 Wi-Fi 或者蓝牙同时使用时，会出现部分数据帧异常的问题，该如何解决?

- 这个问题在非 ESP32-S3 的芯片上很难解决，因为 RMT 刷 LED（尤其是很多个 LED 的时候）严重依赖中断，且不支持 DMA，需要软件在中断切换 ping-pong buffer，如果中断没有及时响应，就会出现问题。默认情况下 (即只设置了一个存储块)，是两个灯的数据量就要进一次中断来切换 ping-pong buffer。
- 缓解思路有：
  - 对于 esp-idf release/v4.4 及之前版本，可以增大 mem_block_num，在 release/v5.0 中有进行修改，参考 Breaking Changes in Usage。
  - 将 RMT 的中断安装在特定的 CPU 核上，可以在一个 pin to core 的 task 中调用 driver install 函数，避开 Wi-Fi 或蓝牙使用的核。
  - 您也可以使用 SPI DMA 来代替 RMT 解决此问题，具体请参考 SPI DMA LED 灯带示例。

- 如果您还处于前期技术选型阶段，推荐使用 ESP32-S3 的 RMT。

ESP-IDF 里只有一个 IR NEC 示例, 如何快速实现其他红外协议的适配？

可以在参考 IR NEC 示例的基础上利用 RMT Encoder 来加速适配其他红外协议。

ESP32-S3 RMT 支持配置 4 个 RMT RX/TX channel，但为什么在实际使用 rmt_new_tx_channel 连续创建超过 2 个 RMT TX channel 时就会失败?

- 这是因为 *tx_chan_config* 结构体中配置的 *mem_block_symbols* 参数过大，ESP32-S3 上 RMT 每个专用内存块的大小为 48 字节。如果此时配置的 *mem_block_symbols* 参数超过 48，创建 TX 通道时实际上会把相邻的下一个通道对应的内存块也占用掉。因此如果您要同时创建并使用 4 个 RMT RX/TX channel，*mem_block_symbols* 参数的值不能超过 48。
- 此外，ESP32 上 RMT 每个专用内存块的大小为 64 字节。

ESP32-S3 RMT 是否能实现多个 TX Channel 的同步输出？

- 可以，请参考以下参考代码：

```
rmt_channel_handle_t tx_channels[TEST_RMT_CHANS];
rmt_sync_manager_handle_t synchro = NULL;
rmt_sync_manager_config_t synchro_config = {
  .tx_channel_array = tx_channels,
  .array_size = TEST_RMT_CHANS,
};
rmt_new_sync_manager(&synchro_config, &synchro);
for (int i = 0; i < TEST_RMT_CHANS; i++) {
  rmt_transmit(tx_channels[i], led_strip_encoders[i], leds_grb, TEST_LED_NUM␣
↪* 3, &transmit_config);
}
```

ESP32-S3 如何实现用 RMT TX 通道循环发送数据，比如进行无限循环？

- 将 *rmt_transmit_config_t::loop_count* 配置为 -1 即可无限循环传输，更多细节请参考 Initiate TX Transaction。

ESP32-S3 是否支持硬件 One-Wire？

- ESP32-S3 可以通过 RMT 外设支持 One-Wire 总线协议。具体应用可参考 "esp-idf/examples/peripherals/rmt/onewire_ds18b20" 例程。

4.4.11 安全数字输入输出 (SDIO)

[English]

SDIO 最高速度能支持到多少？

- ESP32 SDIO 时钟最高 50 MHz，最高支持 4-线模式。
- ESP32-S3 SDIO 时钟最高 80 MHz，最高支持 8-线模式。
- 实际应用速率，同时受到存储媒介读写速率影响。

ESP8266 的 SDIO 是否支持 SD 卡？

ESP8266 是 SDIO 从机，不支持 SD 卡。

ESP32 SD 卡支持的最大容量是多少？

- SD3.01 规范中 SDXC 的卡最大支持 2 TB (2048 GB) 容量。
- ESP32 的 SDMMC 主机符合 SD3.01 协议，通过该外设可以访问最多 2 TB 的区域；使用 SDSPI 驱动通过 SPI 总线访问卡时，硬件也支持访问 2 TB 的区域。
- 在软件层面上，卡能使用的空间还受文件系统的影响。

ESP32 SD 卡是否可以与 flash & PSRAM 共同使用？

- 在引脚互不占用的情况下是可以共同使用
- 需注意，当 ESP32 采用 SDMMC 主机驱动时，ESP32-WROOM 和 ESP32-WROVER 模组的 SDMMC Slot0 引脚与 flash 冲突。

使用 ESP-WROOM-S2 模组，是否支持 SDIO 作从机？

ESP-WROOM-S2 支持 SDIO 作从机。

ESP32-S2 取消了 SDIO 接口，是否还支持外接 TF 卡？

ESP32-S2 可使用 SPI2/SPI3 的接口外接 TF 卡，此时使用 TF 卡的 SPI 模式。

ESP32-S2 支持 eMMC 吗？

CHIP: ESP32-S2

不支持。

ESP32-S2 是否支持 SDIO 作从机？

ESP32-S2 没有 SDIO 接口，不支持 SDIO 作从机。

ESP32 如何开启和关闭 SDIO 从机接收数据的中断？

SDIO 从机数据接收与挂载的 buffer 状态有关，接收完之后需要调用 sdio_slave_recv_load_buf 释放 buffer，否则 SDIO 主机将无法继续向 SDIO 从机发送数据。

4.4.12 SPI 控制器

[English]

ESP-WROOM-02D 模块是否可以外接 SPI flash？

ESP-WROOM-02D 模组是一款基于 ESP8266 芯片的 Wi-Fi 模组，支持使用 SPI 接口与外部 SPI flash 设备进行通信。具体来说，ESP-WROOM-02D 模组提供了 4 个 SPI 接口引脚（GPIO12、GPIO13、GPIO14 和 GPIO15），其中 GPIO12~GPIO14 可以用作 SPI 主机接口的 MISO、MOSI、和 SCLK 引脚，而 GPIO15 则可以用作 SPI 从机接口的 CS 引脚。

要使用 ESP-WROOM-02D 模组连接外部 SPI flash 设备，需要将 SPI flash 设备的 MOSI、MISO、SCK 和 CS 引脚分别连接到 ESP-WROOM-02D 模组的 GPIO12~GPIO14 和 GPIO15 引脚上。同时，还需要在 ESP8266 的固件中正确配置和初始化 SPI 接口，以便能够正确地与外部 SPI flash 设备进行通信。

需要注意的是，外接 SPI flash 设备的型号和容量等参数需要根据具体的应用场景和需求进行选择，同时需要特别关注 SPI flash 设备的时序特性和可靠性，以保证数据传输的正确性和稳定性。此外，还需要考虑 SPI flash 设备和 ESP-WROOM-02D 模组之间的物理距离和噪声环境等因素对通信质量的影响，尽可能采取合适的措施来提高系统的可靠性和性能。

ESP-WROOM-S2 作为从机，STM32 作为 MCU ，可以使用 SPI 接口下载吗？

不可以，默认下载功能仅支持串口 UART0，固件启动后可应用中使能其他外设，在应用中自行设计支持 OTA 功能。

ESP32 中 SPI0/SPI1/HSPI/VSPI 三者有什么区别呢？

- ESP32 有 4 组 SPI，SPI0 和 SPI1 是两个外设，统称为 MSPI。其中 MSPI CS0 连接储存程序的 flash，MSPI CS1 连接 PSRAM，SPI0 和 SPI1 共用一组 GPIO 接口，默认被占用；剩下的两组 SPI2 和 SPI3 为可供客户自由使用的通用 SPI。
- HSPI 代表上述 SPI2，VSPI 代表上述 SPI3，这两组 SPI 均为通用 SPI，并且都支持 QSPI。

ESP32 使用 SPI DMA 时最大的数据传输量是 4095 字节，是因为硬件限制吗？

- 是的，这属于硬件限制。
- DMA 链表中单个节点只能挂载 4095 字节的数据，但可以通过若干节点来发送更多的数据。
- SPI 通过 DMA 链表发送数据的最大字节数还受限于硬件寄存器 *SPI_LL_DATA_MAX_BIT_LEN*（不同系列芯片的数值不同，可在 *ESP-IDF* 中搜索到），即 *'max_transfer_sz <= (SPI_LL_DATA_MAX_BIT_LEN » 3)*。

ESP32-S2 的 SPI 同时访问三个 SPI 从机设备，是否需要做信号量同步才能访问？

- 同一个 SPI 外设作为主机，一次只能与一个从机进行通信，由 CS 决定与哪个从机进行通信。如果是给 SPI 驱动挂 3 个从机设备，并与它们分别通信的话是可以的，推荐这种用法。
- 推荐只在一个任务中操作共用同一个 SPI 的设备，否则是线程不安全的，需要通过信号量同步进行通信，具体问题见 SPI Master driver-feature。

使用 ESP32 开发板基于 ESP-IDF release/v4.3 版本的 SDK 进行开发测试，软件编译报错如下，是什么原因？

```
spi_flash:Detected size(8192K) smaller than the size in the binary image␣
↪header(16384K).Probe failed.
```

- 原因是配置的 flash 大小比实际使用的 flash 大小要大，为避免误用更大的地址空间而对实际使用的 flash 大小进行检测。

SPI 从机支持最大速度是多少？

CHIP: ESP32

ESP32 作为 SPI 从机时钟最高只支持到 10 M。

使用 ESP32 作为 SPI 主机设备，在非 DMA 模式下最大可一次性传输多少字节的数据？

- 使用 ESP32 作为 SPI 主机设备，在非 DMA 模式下最大可一次性传输 64 字节的数据。
- 当传输不超过 32 比特时，可以使用 SPI Master 驱动内部的 4 字节数组作为发送数据的缓冲区，可参考 Transactions with Data Not Exceeding 32 Bits 说明。
- 当传输超过 32 比特时，需要自行创建 SPI 发送数据的缓冲区，可参考 SPI Master Transactions 说明。
- 使用 ESP32 作为 SPI 主机设备在非 DMA 模式下传输超过 32 比特的 SPI 数据，可参考例程 esp-idf/examples/peripherals/spi_slave/sender。

使用 ESP32-S3-WROOM-1 (ESP32-S3R2) 模组基于 ESP-IDF v4.4 版本的 hello-world 例程开启 PSRAM 的设置后，打印如下报错，是什么原因？

```
E (232) spiram: Virtual address not enough for PSRAM!
```

ESP32-S3R2 芯片集成了 4 线的 2 MB PSRAM，请在 menuconfig 中将 PSRAM 模式设置为 **Quad** 模式。如下：

```
menuconfig → Component config → ESP32S3 Specific → Support for
external, SPI connected RAM → SPI RAM config → Mode (QUAD/OCT)
of SPI RAM chip in use (Quad Mode PSRAM)
```

使用 ESP32-S3-WROOM-2 (ESP32-S3R8V) 模组基于 ESP-IDF v4.4 版本的 hello-world 例程开启 PSRAM 的设置后，打印如下报错，是什么原因？

```
E (453) psrm: psrm ID read error: 0x00ffff
E (454) cpu start: Failed to init external RAM!
```

ESP32-S3R8V 芯片集成了 8 线的 8 MB PSRAM，请在 menuconfig 中将 PSRAM 模式设置为 **Octal** 模式。如下：

```
menuconfig → Component config → ESP32S3 Specific → Support for
external, SPI connected RAM → SPI RAM config → Mode (QUAD/OCT)
of SPI RAM chip in use (Octal Mode PSRAM)
```

ESP8266 RTOS SDK 是否支持 SPI 全双工？

CHIP: ESP8266

不支持。因为 ESP8266 不支持 DMA，因此为了提高传输性能利用了全部 FIFO，所以只能半双工，具体的详情请参考 SPI readme。

ESP32 能支持三线 SPI 的 9 位时钟模式（即用第 1 位表示后 8 位是命令还是数据的模式）吗？

不支持，目前 ESP32 所有系列的芯片都不支持非字节对齐的数据传输，即只支持 8 位对齐的数据传输，该问题的具体说明见 Github issue。

后续新版本的 ESP32 芯片可能会支持非字节对齐的数据传输，但目前还没有具体的时间表。

将 ESP32-S2 的 GPIO35 管脚设置为 SPI 屏的 SDA 数据线后，期望的结果是空闲时 SDA 线应为低电平，写数据时应为高电平。但此时为什么一上电空闲时此管脚为高电平，写数据是低电平？如何实现我期望的结果？

请修改 spi_device_interface_config_t 结构体里的 mode 成员变量。

4.4.13 定时器

[English]

ESP8266 使用 HW 定时器中断有哪些注意事项？

- 可以参考相关 API 文档 《ESP8266 技术参考手册》。
- 如果使用 NonOS SDK，可参考 《ESP8266 Non-OS SDK API 参考》。
- 通常情况下，硬件中断需要尽快执行结束，并且将回调函数放入 IRAM 中，避免 Cache 影响。
  - RTOS SDK 需要函数去添加 IRAM_ATTR。
  - NonOS SDK 不能在函数前添加 ICACHE_FLASH_ATTR。

定时器如何设置中断优先级呢？

- esp_timer 仅在芯片为 ESP32 时可以配置中断优先级，通过在 Menuconfig 中修改配置项 CONFIG_ESP_TIMER_INTERRUPT_LEVEL 来实现。
- General Purpose Timer 可以设置中断优先级，通过修改 timer_isr_callback_add 接口的最后一个参数设置。

4.4.14 触摸传感器

[English]

使用 ESP32 做触摸相关应用时，哪里有相关资料可参考？

请参考推荐的 软硬件设计。

ESP32-S2 Touch Sensor 的防水功能是在有水时屏蔽 Touch 还是有水时仍然能识别 Touch 事件？

ESP32-S2 Touch Sensor 的防水功能可以在有少量水珠场景下正常工作。在有大面积积水（Touch 触点区域被完全覆盖）的情况下，Touch 会暂时锁定，直到积水被清除后 Touch 才能恢复正常工作。

ESP32-S2 Touch Sensor 的防水流功能在屏蔽有水流的 Touchpad 时，是否能够保持未沾水的 Pad 仍能使用？

可以，可通过软件选择具体屏蔽的通道。

是否有推荐的可以用于 Touch Sensor 测试、稳定触发 Touch Sensor 并且参数与人手触摸时参数接近的材料？

对一致性要求较高的实验，可使用手机电容笔来替代人手进行测试。

Touch Sensor 的管脚能否重映射？

不能，因为 Touch Sensor 属于模拟信号处理。

在覆盖亚克力板后，Touch Sensor 检测阈值是否需要重新设置？

需要重新设置一个阈值。

Touch Sensor 能否检测是否有亚克力板覆盖，以便在添加或移除亚克力板时，自动切换预设定的检测阈值？

暂时不能自动适应覆盖层物理参数变化所带来的影响。

ESP32 触摸屏有哪些参考驱动？

- 代码请参考 touch_panel_code。
- 文档请参考 touch_panel_doc。

4.4.15 双线汽车接口 (TWAI)

[English]

在使用 ESP32 TWAI® 控制器时有哪些注意事项？

请参考 《ESP32 系列芯片勘误表》 > 章节 ESP32 TWAI 相关问题。

4.4.16 UART 控制器

[English]

使用 ESP8266 RTOS v2.1 以及之前版本 SDK，如何将日志配置到 UART1？

在配置 UART1 初始化后，可以通过 API 切换日志输出到 UART1。

```
UART_SetPrintPort(UART1);
```

使用 ESP8266 RTOS v3.0 以及之后的 SDK，如何将日志配置到 UART1 ？

可通过 `menuconfig -> Component config -> ESP8266-specific -> UART for console output -> custom -> UART peripheral to use for console output -> UART0` 修改为 UART1 接口。

ESP32 IDF 中如何使能 UART 流控？

- 硬件流控使能：uart-flow-control。
- 软件流控使能：software-flow-control。

ESP32 使用 UART0 作为通信串口，有哪些需要注意的地方？

- 通常情况下不建议将 UART0 作为普通的通信串口，因为 UART0 为设备默认日志输出串口。
- 若 ESP32 的 UART 不够用，或者硬件设计已经不方便更改的情况下，如果您要使用 UART0 作为普通的通信串口，请参考以下建议：

软件方面：防止打印影响串口通信，默认程序中 UART0 主要有三处打印设置。

- 第一处是上电 ROM 打印，上电时可将 MTDO 管脚设为低电平屏蔽上电 ROM 打印。
- 第二处是引导加载程序日志信息输出，您可以将 `menuconfig -> Bootloader config -> Bootloader log verbosity` 设置为 `No output` 来屏蔽引导加载程序日志输出。
- 第三处是应用日志输出，您可以将 `menuconfig -> Component config -> Log output -> Default log verbosity` 设置为 `No output` 来屏蔽应用日志输出。

硬件方面：

- 在下载程序的时候，注意防止 UART0 上有其它设备，如果有其它设备可能会影响程序的下载。建议在 ESP32 和其它设备之间预留一个 0 Ω 电阻，如果下载有问题可以断开这个 0 Ω 电阻。

ESP32-SOLO-1 的 GPIO34 ~ GPIO39 是否可作为 UART 的 RX 及 TWAI® 的 RX 信号管脚？

GPIO34 ~ GPIO39 仅作为接收，可作为 UART 的 RX 及 TWAI 的 RX 信号管脚。

使用 ESP32 如何动态修改串口波特率并立即生效？

请使用 `uart_set_baudrate()` API 来修改 UART 波特率。参见 API 说明。

请问 ESP32 芯片支持 USRAT（Universal Synchronous Asynchronous Receiver Transmitter）吗？

不支持，ESP32 仅支持 UART，无法提供同步时钟。

ESP32 芯片的串口校验支持M ARK 和 SPACE 校验吗？

ESP32 芯片不支持。

ESP8266 串口的硬件 FIFO 是多大？

ESP8266 的 UART0 和 UART1 各有一个□度为 128 字节的硬件 FIFO 和读写 FIFO，且都在同一个地址操作。参见《ESP8266 技术参考手册》中“11.2. 硬件资源”章节说明。

ESP8266 的串口波特率范围是多大？

ESP8266 的串口波特率范围为 300 ~ 115200*40 bps。参见《ESP8266 技术参考手册》中的“11.3.1. 波特率”章节说明。

如何修改 UART0 的输出口？

CHIP: ESP32 | ESP32 | ESP32-C3

可以在 menuconfig 中进行设置，`idf.py menuconfig` —> `Component config` —> `Common ESP-related` -> `Channel for console output(custom UART)`。

使用 ESP8266，想把 UART0 专门用作下载，再使用 UART1 与其他芯片通信。GPIO4 和 GPIO5 能配置成 UART1 串口吗？

- 由于 UART1 的 RX D 被占用了，所以 UART1 不能与其他芯片进行通讯，但 UART1 的 TXD 管脚可用作输出日志。
- ESP8266 与其他芯片通信只能通过 UART0 的 CTS 和 RTS 管脚交换来实现，配置成 GPIO4 和 GPIO5 是无效的。
- ESP8266 与其他芯片通信可通过调用“uart_enable_swap()”函数，通过 UART0 的 CTS 和 RTS 引脚进行交换，交换为 MTCK (IO13)、MTDO (IO15) 管脚。管脚交换后 ESP8266 可通过 GPIO13（TXD）和 GPIO15（RXD）来与其他芯片进行 UART 通信。

ESP32 的 UART0 是否可以在输出日志的同时又用作接收电脑控制台的输入？

- 可以。UART0 输出日志只需要使用 TXD0 管脚，接收电脑控制台的输入只需要使用 RXD0 管脚。可基于 “esp-idf/examples/system/console/basic” 例程来测试。

4.4.17 USB

[English]

ESP32 是否支持 USB 功能？

- ESP32 不支持 USB 功能。
- ESP32-S2/S3 支持 USB2.0 Full-speed 模式。

ESP-IDF SDK USB 接口支持 HID、MSC 这些模式吗？

- ESP32S2/S3 可作为 MSC 主机，支持读写 U 盘等存储设备，可参考 esp-idf。
- ESP32S2/S3 可作为 MSC 设备，模拟 U 盘存储，可参考 esp-iot-solution。
- ESP32S2/S3 HID 主机可参考 ESP-IDF Host HID。
- ESP32S2/S3 HID 设备类可参考 ESP-IDF Device HID。

ESP32-S2 USB 接口电流稳定输出为多少？

对于 VBUS 电源线的电流输出能力，由供电决定，与 ESP32-S2 芯片无关。如果采用自供电，请参考 Self-Powered Device。

ESP32-S3 的 USB 支持 USB 主机吗？

支持，ESP32-S3 USB 主机功能与 ESP32-S2 一致。

ESP32-C3 USB 支持 USB 串口功能和 USB JTAG 功能吗？

支持，但无法自定义描述符。

ESP32-S2 和 ESP32-S3 USB 的特征是？

ESP32-S3 和 ESP32-S2 具有相同的 USB 2.0 OTG 外设，支持 full-speed 模式，可支持 USB 主机和 USB 设备功能。除此以外，ESP32-S3 还支持 USB-Serial-JTAG 专用外设，可用于固件下载和调试。

ESP32-S2 USB 主机的库和例程是否有参考？

可参考 ESP-IDF 的 USB Host 驱动。

ESP32-S2 支持的 USB 协议是 OTG 1.1，速度最高是 12 Mbps。能和 USB 2.0 设备通信吗？

USB 2.0 和 USB 1.1 full-speed 模式兼容，可以进行通信。

ESP32-S2 支持 USB 摄像头吗？

支持。ESP32-S2/ESP32-S3 USB Host UVC 示例代码请参考 usb_stream。

ESP32-S3 是否支持带有麦克风和扬声器的 USB 摄像头？

支持。ESP32-S2/ESP32-S3 USB Host UVC + UAC 示例代码请参考 usb_stream。

是否有 ESP32-S2 做 U 盘 (MSC DEVICE) 的参考示例？

请参考 usb_msc_wireless_disk demo。目前测试的平均读写速度为：读 540 KB/s，写 350 KB/s。

ESP32-C3 有 USB，是否不需要 cp2102 芯片就可以直接通过 USB 下载固件？

是的，ESP32-C3 可通过 USB 串口直接烧录程序，对应 USB 串口号 Windows 设备上显示为 COMx，Linux 设备上显示为 ttyACMx。

ESP32-C3 是否支持 USB 主机？

不支持，ESP32-C3 仅支持 USB-Serial-JTAG 功能，只能作为 USB 设备。

ESP32-C3 芯片可以使用 USB 下载固件，但在 ESP-IDF v4.3 下不支持。如何使用 USB 下载固件？

请在 ESP-IDF v4.4 及以后版本下编译，拉出最新分支并 更新 IDF 工具，然后便可正常编译并使用 USB 下载固件，使用方法请见 usb-serial-jtag-console。

ESP32-S2 是否支持 USB HID？

支持，USB HID Device 请参考 ESP-IDF Device HID。USB HID Host 请参考 ESP-IDF Host HID 例程。

测试 USB 摄像头 Wi-Fi 传输例程，日志打印如下报错，是什么原因?

```
E (1437) UVC STREAM: Configuration descriptor larger than control transfer␣
↪max length
```

此报错日志是因为 USB Camera 发送的描述符长度大于默认预设的长度（256），可以修改如下配置为 2048 进行测试：

`Component config > UVC Stream > (2048) Max control transfer data size (Bytes)`

ESP32-S3 支持 USB CDC 输出程序日志和下载固件吗？

ESP32-S3 可以用 USB CDC 输出程序日志和下载固件，但是需要开启如下配置选项：

`Component config` > `ESP System Settings` > `Channel for console output` > `USB CDC`

ESP32-S3 是否支持 USB Device 为 Class 0 的装置?

- 支持，可参考示例：esp-idf/components/tinyusb/additions/src/usb_descriptors.c 。当 Class code == 00H 时，class 类别由 interface 指定。

ESP32-S3 的 USB OTG 接口可以同时使用 USB Host 和 USB Device 模式吗？

- ESP32-S3 的 USB OTG 接口不能同时使用 USB Host 和 USB Device 模式，但可以通过软件切换两种模式，分时使用。
- 如需要 USB OTG 标准协商功能，需要注意的是目前 ESP32-S3 仅硬件上支持此功能，软件协议还没有支持。

测试 esp-idf/examples/peripherals/usb/device/tusb_serial_device 例程，使用 TinyUSB 发送数据，必须要使用 tinyusb_cdcacm_write_flush 函数吗？

为了防止发送 FIFO 溢出，可以使用 `tinyusb_cdcacm_write_flush()` 函数进行刷新。但是，大量循环的刷新可能会失败，建议根据实际应用进行设置。

ESP32-S3 是否支持外接 USB hub 芯片分出两个 USB 口同时连接 USB 4G 模块和加密狗？

ESP32-S3 的 USB 接口目前不支持外接 USB hub 芯片（缺少驱动支持）。

ESP32-S2/ESP32-S3 做 UVC Host 连接部分型号的 UVC 摄像头后提示 HID_PIPI_EVENT_ERROR_OVERFLOW，什么原因？

这个错误说明选择的摄像头 Alt 接口端点 MPS 过大（ESP32-S2/ESP32-S3 最高支持 512 字节），需要确认摄像头在 USB1.1 下是否有小于等于 512 字节的接口。

ESP32-S2/ESP32-S3 是否有 USB 4G 上网方案？

有，请参考 USB CDC 4G 模组示例。

ESP32-S2/ESP32-S3 是否有 USB CDC Host 示例？

有，请参考 ESP-IDF USB CDC Host 示例 或 esp-iot-solution USB CDC Host 示例。

通过 ESP32-C3/ESP32-S3 USB Serial/JTAG Controller 功能烧录固件时发现 PC 有时识别不到 USB 串口，或者会反复看到 USB 串口识别到后又自动断开，这是什么原因？

目前 ESP32/ESP32-S2/ESP32-S3/ESP32-C3 芯片启动逻辑都是：如果不能正常启动（flash 为空，flash 里没有正确的数据/固件，flash 上电时序问题等），内部定时器会触发（一般是几秒钟）一次芯片重启。直到程序能正常启动，或者进入了下载模式，才会稳定连接不再重启，又因为芯片重启时 ESP32-S3/ESP32-C3 USB-Serial-JTAG 外设会重新初始化，所以对应的现象就是连接到 PC 以后“断断续续”（以几秒钟为周期连接、断开、连接、断开….），以下为两种解决办法：

- 芯片首次下载前或者 flash 擦除以后，手动 boot 进入下载模式，这样芯片就会稳定连接。
- 提前通过 UART 烧录能稳定运行的固件，在芯片中有稳定的固件以后，后续烧录时 “USB 断断续续识别后又断开” 的现象就不会再出现。

如果没有预留手动 boot 对应的 strap pin 测试点，则需要在初次 USB 下载时进行多次尝试。

ESP32-S2/ESP32-S3 无法达到 USB full speed 提到的最大 12 Mbps，可能是什么原因？

以 TinyUSB 协议栈为例，因为此 USB 模式没有使用 DMA，而是直接使用 CPU 轮询，每次传输都会有一些时钟时间片被浪费，所以 TinyUSB 协议栈预计只能达到 6.4 Mbps（如果采取批量传输，理论能达到 9.628 Mbps）。

如何判断 ESP32-S2/ESP32-S3 USB 有支持某款 USB 摄像头的可能性？

ESP32-S2/ESP32-S3 USB 只支持包含不大于 512 字节的 wMaxPacketSize Video Streaming 端点的 USB 摄像头，用户可直接使用 USB 摄像头 Wi-Fi 传输 例程测试，若摄像头无法支持，将会打印错误信息。

ESP32-S2/ESP32-S3 能支持最大为多大分辨率的 USB 摄像头？

- 如果不考虑本地 JPEG 解码。那主要瓶颈在 USB 吞吐率，USB 摄像头往往同步传输，因为 ESP USB 存在 FIFO 大小的限制，目前最大只能到 500 KB/s。所以假设要达到 15 帧，每帧大小只能 33 KB，具体 33 KB 能实现的最大分辨率取决于压缩率，一般可以到 480 * 320 分辨率。
- 如果考虑本地 JPEG 解码，要同时考虑这个分辨率能不能达到每秒 15 帧。

ESP32-S2/ESP32-S3 USB 做 USB CDC Device 时是否能识别到 USB 的插拔动作？

- 可以，USB device 采用 tinyusb 协议栈，包含 mount 和 umount 回调函数来反馈 USB 的插拔动作事件。
- 需要注意的是，如果该设备为自供电 USB 设备，若需要在不断电的情况检测到插拔动作，请注意预留 VBUS 检测引脚，请参考 自供电 USB 设备解决方案

ESP32-S3 USB 使能 RNDIS 和 CDC 功能后发现 PC 能识别到 COM 口，但是 COM 口的自动烧录功能失效了，是否符合预期？

- 符合预期，因为 USB 自动烧录功能通过 USB-Seial-JTAG 外设实现，但 USB RNDIS 功能通过 USB-OTG 外设来实现，USB-OTG 外设和 USB-Seial-JTAG 外设在同一时刻只有其一能工作。
- 如果应用上使用了 USB-OTG 外设，那通过 USB-Seial-JTAG 外设实现的自动烧录功能就没有了。但是可以手动进入下载模式来进行 USB 烧录。

请问 ESP32-S2/ESP32-S3 是否支持 USB CDC NCM 协议？

- 目前只支持 USB CDC ECM 协议，不支持 USB CDC NCM 协议。

将 ESP32-C3/ESP32-S3 的 USB 引脚初始化为 GPIO 或其它外设功能以后，为什么无法再通过 USB 进入固件烧录？

- ESP32-C3/ESP32-S3 的 USB 引脚可初始化为 GPIO 或其它外设引脚，但是需要注意的是，初始化完成以后，原有的 USB 下载功能将被断开，无法再通过 USB 接口自动进入下载模式，但用户可以通过手动拉低 Boot 引脚 (ESP32-C3 为 GPIO9, ESP32-S3 为 GPIO0)，手动使 ESP32-C3/ESP32-S3 进入下载模式，再通过 USB 进行下载。

将 ESP32-C3/ESP32-S3 的 USB 接口作为产品唯一的固件下载接口，有哪些注意事项?

- 禁止将 ESP32-C3（GPIO18,GPIO19）/ ESP32-S3（GPIO19,GPIO20）的 USB 引脚复用为其它外设功能。
- 如果迫不得已，应用程序中必须将 USB 引脚复用为其它功能，那硬件上必须同时引出 Boot 引脚 (ESP32-C3 为 GPIO9, ESP32-S3 为 GPIO0)，用于手动进入下载模式。

Windows 环境下使用 `idf.py -p com35 flash monitor` 命令，通过 USB 接口一键下载和打印，报错如下日志是什么原因？

- 错误日志如下：

```
Connecting...
Failed to get PID of a device on com35, using standard reset sequence.
```

- Windows 环境下配置 COM 口必须用大写，不可用小写 com。

如何为 ESP32-S 系列的产品申请 USB VID/PID？

- 如果你的软件是基于 TinyUSB 协议栈来实现的，可以使用默认的 TinyUSB PID。否则，你需要为每个 ESP32-S 系列的产品申请 USB VID/PID。详细说明请参见 “usb-pids” 。

在 Windows 环境下，使用 USB-Serial-JTAG 接口下载固件，是否可以固定 COM 口编号？

- 可以使用管理员方式打开 Windows CMD，执行以下指令来添加注册表项，以阻止依据 Serial 号递增编号，设置完成后请重启电脑使能修改：

```
REG ADD HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\usbflags\
↪303A10010101 /V IgnoreHWSerNum /t REG_BINARY /d 01
```

- 更多信息请参考 阻止 Windows 依据 USB 设备序列号递增 COM 编号。

4.4.18 其他外设

[English]

REF_TICK 时钟频率可以修改吗?

CHIP: ESP32 | ESP32-S2 | ESP32-C3

不可以修改，REF_TICK 时钟是固定的。

ESP32 是否支持 PCI-E 协议？

ESP32 不支持 PCI-E 协议。

4.5 协议

[English]

4.5.1 ESP-TLS

[English]

ESP8266 测试 RTOS SDK `mqtt/ssl_mutual_auth` 为何连接服务器失败？

- 出现 SSL 无法连接可能是由于 ESP8266 内存不足导致。
- 请使用 ESP8266-RTOS-SDK master 版本来测试此例程，master 版本支持在 menuconfig 配置端动态分配内存，可以减少峰值内存的开销，通过 menuconfig -> `Component config` -> `mbedTLS` -> (键 "Y" 使能) `Using dynamic TX /RX buffer` -> (键 "Y" 使能) `Free SSL peer certificate after its usage` -> (键 "Y" 使能) `Free certificate, key and DHM data after its usage`。

ESP HTTPS 在使用时能跳过服务器证书校验吗？

- 可以，请在 menuconfig 里使能以下选项：
  - `Menu path: (Top) -> Component config -> ESP-TLS -> Allow potentially insecure options`
  - `Menu path: (Top) -> Component config -> ESP-TLS -> Allow potentially insecure options` -> `Skip server certificate verification by default`
- 同时要确保 `esp_http_client_config_t` 结构体里不设置 `cert_pem` 成员变量。如果设置了 `cert_pem`，就仍会用这个设置的 CA 证书校验服务器证书。

- 如果要同时测试 HTTP OTA，还需要在 menuconfig 里使能 `Menu path: (Top) -> Component config -> ESP HTTPS OTA -> Allow HTTP for OTA` 选项。

如何将 ESP-TLS 中的 esp_tls_conn_read API 设置成非阻塞模式？或者有其他方式来实现非阻塞？

- 可以将 `esp_tls.h` 里 `esp_tls_cfg_t` 结构体里的 `non_block` 设置为 true 来实现非阻塞。
- 也可以调用 `esp_transport_connect_async` 来实现非阻塞。

ESP-IDF 支持的 TLS 版本有哪些？

- ESP-IDF 里推荐的 TLS 协议为 Mbed TLS 协议。
- ESP-IDF v5.0 及以上版本不再支持 SSL 3.0，TLS 1.0 和 TLS 1.1，当前支持的 TLS 版本为 TLS 1.2 和 TLS 1.3。

4.5.2 HTTP

[English]

ESP8266 支持 HTTP 服务端吗？

支持。ESP8266 在 SoftAP 和 Station 模式下都可以作服务端。

- 在 SoftAP 模式下，ESP8266 的服务端 IP 地址是 192.168.4.1。
- 如果 Station 模式，服务端的 IP 地址为路由器分配给 ESP8266 的 IP。
- 如果基于 SDK 进行二次开发，可参考相关应用示例。
- 如果使用 AT 指令，需使用 `AT+CIPSERVER` 开启服务端。

如何使用 esp_http_client 发送块 (chunked) 数据？

- 可以通过 HTTP Stream 的方式，将 `esp_http_client_open()` 的 `write_len` 参数设置为 -1，代码中会自动将 `Transfer-Encoding` 设置为 `chunked`，参考 esp_http_client.c 中的 `http_client_prepare_first_line()`。
- 可使用如下代码：

```
static void http_post_chunked_data()
{
    esp_http_client_config_t config = {
    .url = "http://httpbin.org/post",
    .method = HTTP_METHOD_POST, // This is NOT required. write_len < 0 will force␣
↪POST anyway
  };
  char buffer[MAX_HTTP_OUTPUT_BUFFER] = {0};
  esp_http_client_handle_t client = esp_http_client_init(&config);

  esp_err_t err = esp_http_client_open(client, -1); // write_len=-1 sets header
↪"Transfer-Encoding: chunked" and method to POST
  if (err != ESP_OK) {
    ESP_LOGE(TAG, "Failed to open HTTP connection: %s", esp_err_to_name(err));
    return;
  }

  // Post some data
  esp_http_client_write(client, "5", 1); // length
  esp_http_client_write(client, "\r\n", 2);
  esp_http_client_write(client, "Hello", 5); // data
  esp_http_client_write(client, "\r\n", 2);
  esp_http_client_write(client, "7", 1); // length
  esp_http_client_write(client, "\r\n", 2);
  esp_http_client_write(client, " World!", 7);  // data
  esp_http_client_write(client, "\r\n", 2);
  esp_http_client_write(client, "0", 1);  // end
  esp_http_client_write(client, "\r\n", 2);
  esp_http_client_write(client, "\r\n", 2);


  // After the POST is complete, you can examine the response as required using:
  int content_length = esp_http_client_fetch_headers(client);
  ESP_LOGI(TAG, "content_length: %d, status_code: %d", content_length, esp_http_
↪client_get_status_code(client));

  int read_len = esp_http_client_read(client, buffer, 1024);
  ESP_LOGI(TAG, "receive %d data from server: %s", read_len, buffer);
  esp_http_client_close(client);
  esp_http_client_cleanup(client);
}
```

ESP32 作为 HTTP 客户端，可以设置 cookie 的方式吗？

ESP32 本身没有直接设置 cookie 的 API，但可以通过 esp_http_client_set_header 向 HTTP 头里添加 cookie 等头数据的方式来设置 cookie。

ESP32 作为 HTTP 服务器时，如何设置可同时连接的客户端个数上限？如果客户端连接个数超出上限，会出现怎样的情况？

- 通过配置 `httpd_config_t` 结构体里的 `max_open_sockets` 即可设置同时连接的客户端最大个数。
- 如果存在客户端连接个数超出上限的情况，可以把 `httpd_config_t` 结构体里的 `lru_purge_enable` 参数设置为 true。这个参数设置为 true 的作用是如果没有可用的套接字（这个套接字由 `max_open_sockets` 决定），就会清除最少使用的那个套接字从而接受最新的套接字。

ESP32 是否有至少在 HTTP/2 上实现 gRPC 客户端的示例？

目前还没有。

在 ESP-IDF 中，如何通过 HTTP 下载文件里的某一特定段（即在头部添加 `Range:bytes` 信息）？

可以参考 esp http client 示例 里的 `http_partial_download` 函数。

4.5.3 lwIP

[English]

TCP 链接关闭后占用的相关资源何时释放？

TCP 链接关闭后占用的相关资源会在 2 MSL，即 120 s, 或者发送的 `linger/send_timeout` 超时之后释放。

ESP8266 RTOS SDK v3.2 SNTP 校准后误差会逐渐变大，如何解决？

原因是 ESP8266 系统定时器有误差，采用软件定时器，自身存在误差较大。可通过以下几种方法改善：

- 分支 v3.2 可以通过创建 task 定时重新从服务器同步时间（推荐 300 s）。
- 分支 release-v3.3 的系统时钟代码有进行重构，目前测试误差较小，并且也可以定时同步服务器时间。
- 分支 master 继承了 release-v3.3 上的代码重构，除此之外，可通过 menuconfig 配置 SNTP 同步间隔，路径如下：`Component config` > `LWIP` > `SNTP` > `Request interval to update time (ms)`。

ESP8266 是否支持设备端自发自收？

- ESP8266 设备端支持自发自收。
- 需要在 menuconfig 配置选项中把 lwIP 的 LOOPBACK 选项打开：`menuconfig` > `Component config` > `LWIP` > `Enable per-interface loopback` (键 “Y” 使能)。
- 设备端往环回地址 127.0.0.1 发送数据，设备端可以在环回地址读取到自己发送的数据。

TCP/IP 默认配置的数据包长度是多少？

请参考 `menuconfig > Component config > LWIP > TCP > Maximum Segment Size (MSS)` 配置的值。

SNTP 协议中使用 UTC 与 GMT 的方法为何获取不到目标时区的时间？

- “TZ = UTC-8” 被解释为 POSIX 时区。在 POSIX 时区格式中，这 3 个字母是时区的缩写（任意），数字是时区落后于 UTC 的小时数。
- “UTC-8” 表示时区，缩写为 “UTC”，比实际 UTC 晚 -8 小时，即 UTC + 8 小时。故 UTC+8 是比 UTC 落后 8 小时，就出现了 UTC+8 比正确的北京时间相差 16 小时的情况。

ESP32 是否有特殊的固件或者 SDK，可以不使用芯片内部的 TCP/IP 协议仅提供 AP/STA (TCP/IP bypass)，以给开发者更多的权限？

ESP-Dongle 的软件方案符合您的上述需求，请联系 商务 签署 NDA 后获取相关方案。

ESP32 & ESP8266 做 TCP server 时端口释放后如何立即被再次使用？

- 在 ESP32 和 ESP8266 上，TCP 端口关闭后并不会立即释放，而是在一定时间内处于 TIME_WAIT 状态，此时绑定与之前相同端口相同源地址的套接字会失败，这是为了确保客户端接收到服务器发送的 FIN 信号并成功关闭连接。在这个状态下，端口无法立即重新使用。需要借助套接字选项 `SO_REUSEADDR`，它的作用是允许设备绑定处于 `TIME-WAIT` 状态，端口和源地址与之前相同的 TCP 套接字。
- 故 TCP server 程序可以在调用 bind() 之前设置 `SO_REUSEADDR` 套接字选项后来绑定同样的端口。
- 也可以使用 setsockopt() 函数来设置 SO_REUSEADDR 选项。以下是一个示例代码：

```
int reuse = 1;
if (setsockopt(socket, SOL_SOCKET, SO_REUSEADDR, &reuse, sizeof(reuse)) < 0) {
  ESP_LOGE(TAG, "setsockopt(SO_REUSEADDR) failed");
  return ESP_FAIL;
}
```

在以上代码中，socket 是一个已经创建的套接字，reuse 是一个 int 类型的变量，其值为 1，表示开启 SO_REUSEADDR 选项。如果 setsockopt() 函数返回负值，表示设置失败。设置 SO_REUSEADDR 选项可以让端口在关闭后立即被再次使用，但也需要注意潜在的风险。如果在端口仍处于 TIME_WAIT 状态时，另一个连接使用该端口，可能会导致数据包错乱，因此需要根据实际情况权衡利弊。

使用 ESP32 模组下载 tcp_client 例程，通过 Wi-Fi 连接路由器，在电脑端进行 Ping 测试，偶尔出现高延时，是什么原因？

Wi-Fi 默认开启 Power Save 模式，关闭 Power Save 可降低由于 Power Save 引起的 Ping 高延时，在 esp_wifi_start() 之后调用 esp_wifi_set_ps(WIFI_PS_NONE) 来关闭 Power Save 模式。

使用 ESP-IDF 如何设置静态 IP？

可以参考 static_ip 示例。

ESP32 有没有 LTE 连接示例？

- 可以参考 ESP-IDF v4.2 及以上版本里的 examples/protocols/pppos_client 示例。
- 对于 ESP-IDF v5.0 及以上版本，请参考 *esp-protocols* 仓库下 示例。

ESP32 TCP 反复关闭并重建 socket 时会出现内存泄漏的情况 (ESP-IDF v3.3)，原因是什么？

IDF v3.3 版本，每次创建 socket 时，如果内部该 socket 数组没有分配过锁，就会给该 socket 分配锁，并且该锁在 socket 释放后并不会回收，下次分配该 socket 数组时就使用之前分配的。所以每次分配新的 socket 数组后释放，就会多一个锁的内存消耗。当每个 socket 数组都分配一遍后，就不会存在内存泄漏。

ESP32 额外开启 TCP server 后对 TCP client 的最大连接数是否有限制？

有限制，ESP32 同时存在的 socket fd 数量受限于 `LWIP_MAX_SOCKETS`，默认为 10。

使用 ESP32，lwIP 的 MTU 默认是多大？

lwIP 的 MTU 默认是 1500（固定值），不建议自行修改。

ESP32 如何增大 DNS 请求时间？

可以手动修改位于 esp-idf/components/lwip/lwip/src/include/lwip/opt.h 里的 `#define DNS_MAX_RETRIES 4`，例如将 `#define DNS_MAX_RETRIES` 的值改成 10，这样 DNS 在一个服务器上会尝试 10 次域名请求，每次请求的超时时间 (s) 是 1，1，2，3，4，5，6，7，8，9，总时间是 46 s。

连续多次创建并关闭 TCP SOCKET 后出现报错“Unable to create TCP socket: errno 23”，怎么解决？

:CHIP: ESP8266 | ESP32 | ESP32-S2 | ESP32-C3 | ESP32-S3 :

- 原因：”errno 23”代表的是 open many open files in system，由于关闭 socket 需要 2 MSL 的时间，所以调用 close 接口并不会立即关闭，导致 socket 持续累加，超过了 socket 最大支持连接数（menuconfig 中默认是 10 个，最大支持 16 个）报错。
- 解决措施：通过 setsockopt 接口设置 SO_LINGER 来调整 TCP 关闭时间，代码实现参考：

```
linger link ;
link.on_off = 1 ;
link.linger = 0 ;
setsockopt(m_sockConnect, SOL_SOCKET, SO_LINGER, (const char*)&link, sizeof(linger));
```

ESP8266 收到“tcp out of order”的报文会怎么处理？

- 如果使能 `CONFIG_LWIP_TCP_QUEUE_OOSEQ(Component config -> LWIP -> TCP -> Queue incoming out-of-order segments)`，会存储“out of order”的报文，代价是消耗内存。
- 如果该配置是未使能，收到“out of order”的报文，会丢弃数据并让对端重传。比如现在有 1、2、3、4 四包数据，ESP8266 先收到 1 然后收到 4。该配置使能时，ESP8266 会把 4 这个数据存下来，等收到 2、3 后，把这四包数据上报应用层；该配置未使能时，ESP8266 会直接丢弃 4，并让对端发送包 2，对端就会从 2 开始发送，即该情况下会增加重传。

ES32 支持 PPP 功能吗？

支持，请参考 esp_modem 示例。

ESP32 使用套接字中的 `read` 和 `recv` API 读取 4 KB 数据时，发现并不是每次都能读到 4 KB 的数据。这种情况如何解释？

- `read` 和 `recv` API 都是用来读底层缓冲区中的数据，比如底层缓冲区中有 100 字节数据，`read` 和 `recv` 传入的 `len` 大小只有 50 字节，那么 API 读到 50 字节的数据时就会返回；如果传入的 `len` 超过底层缓冲区中接收到的数据的长度，比如 200 字节，此时 API 读到 100 字节就会返回，并不会等到接收到 200 字节才返回。所以传入 4 KB 的长度的数据并不一定会返回 4KB 长度的数据，只会返回读取时底层缓冲区中有的数据。
- 如果需要每次都读取到 4 KB 的数据，建议在套接字层之上使用应用代码设计对应的逻辑，让应用代码循环读取数据直到满足 4 KB 的大小。

ESP-IDF 里目前使用的 lwIP 版本是什么？

目前使用的 lwIP 版本是 2.1.3。

在 DHCP 模式下，ESP32 申请到 IP 后，如果租期到期，会续约此 IP 还是重新申请 IP？

DHCP 模式下有两个租期，T1（租约的 1/2 时间）和 T2（租约的 7/8 时间），通常这两个租期期满后会续租同一 IP，只有当上述两个租期时间点都续租失败，才会重新申请 IP。

ESP-IDF 里使用 `setsockopt` 的 `SO_SNDBUF` 选项获取或者设置发送缓冲区大小会报错，为什么？

lwIP 默认不支持 `SO_SNDBUF` 选项，如果需要配置发送缓冲区大小可以在 menuconfig -> `Component config` -> `LWIP` -> `TCP` -> `Default send buffer size` 设置。如果需要获取或者设置接收缓冲区大小，此时需要在 menuconfig 里使能 `CONFIG_LWIP_SO_RCVBUF` 选项后才支持使用 `setsockopt` 的 `SO_RCVBUF` 选项获取或者设置接收缓冲区大小。

使用 ESP-IDF 测试发现 TCP & UDP 的网络数据延时较大，请问 TCP & UDP 协议的缓冲数据机制是什么？

- 对于 TCP，套接字选项里有 `TCP_NODELAY` 选项，可以使能该选项来禁用默认使能的 Nagle 算法，这样就不会出现本地缓存一定数据后再一起发送的情况。
- 对于 UDP，UDP 的数据交互采取直接发送的形式，如果有延迟，也是 Wi-Fi 网络环境的延迟，和 UDP 本身无关。
- 如果是网络环境较差导致 TCP 重传，重传的间隔设置过大会导致延迟高，可以尝试缩短 RTO 的值（通过修改 menuconfig 里的 `component config` -> `lwip` -> `tcp` -> `Default TCP rto time` 和 `TCP timer interval` 选项）。

ESP32 做双网卡（比如 ETH+STA）时，默认路由如何选择？

以下总结了双网卡时默认路由如何选择，以 ETH 和 STA 为例：

- 假设 ETH 和 STA 在同一个局域网：
  - 当设备访问局域网地址时，数据走最后 up 的 netif。
  - 当设备访问非局域网内地址时，数据走 `route_prio` 值大的 netif。
- 假设 ETH 和 STA 不在一个局域网，ETH 属于 192.168.3.x 网段，STA 属于 192.168.2.x 网段：

- 当设备访问 192.168.3.5 时，就会走 ETH netif。
- 当设备访问 192.168.2.5 时，就会走 STA netif。
- 当设备访问 10.10.10.10 时，就会走默认路由（`route_prio` 值大的 netif）。netif 起来后，会根据 `route_prio` 值大小设置默认路由，默认路由往往是 `route_prio` 值大的 netif。当设备访问的地址不在路由表里时，数据就会走默认路由。

ESP-IDF 里 TCP 如何开启 keepalive？

可以参考 esp_tls.c 里的使能 TCP keepalive 相关代码。

ESP-IDF 里可以在多线程里操作同一个套接字吗？

可以，在 ESP-IDF 中，多线程可以共享同一个套接字进行通信。每个线程都可以使用同一个套接字来发送和接收数据，但需要确保在访问套接字时进行线程同步，避免竞争条件和死锁等问题。一般可以使用互斥锁（mutex）来控制套接字的访问，确保每个线程访问套接字时都是互斥的，避免出现同时访问套接字导致数据混乱的情况。多线程操作同一个套接字有风险，不建议该做法。

ESP DHCP 服务器模式下，ESP 设备分配到其他设备 IP 的时间是多少？

默认为 120 s，具体见 `DHCPS_LEASE_TIME_DEF` 参数，不建议修改为太小的值。

ESP-IDF DHCP 里三个租约相关时间是指什么？具体对应代码里的什么参数？

DHCP 有租约时间 (Address Lease Time)、租约续期时 (Lease Renewal Time) 和租约重新设定的时间 (Lease Rebinding Time)，分别对应 lwIP 代码 `offered_t0_lease`、`offered_t1_renew` 和 `offered_t2_rebind`。

ESP-IDF lwIP 里每次发送数据的最大长度是多少？

如果使用套接字接口 `send`，支持最大长度有 `SSIZE_MAX` 参数决定。如果使用 `tcp_write` 函数，最大发送的长度受限于 `snd_buf`（发送缓存区长度）。`send` 接口是 lwIP 基于顺序 API 封装的套接字接口，是比 `tcp_write` 还要上层的接口，更适合于用户层开发调用。这两个 API 调用资源占用几乎没有差别。

使用 ESP-IDF 出现 lwIP 层相关问题需要更多的调试日志时，如何使能对应的调试日志打印（如 lwIP 下的 DHCP 和 IP 等）？

- 可以在 menuconfig 里使能 lwIP 相关调试日志选项，具体的选项为：menuconfig -> `Component config` -> `LWIP` -> `Enable LWIP Debug`。其中有子选项 `Enable IP debug messages`、`Enable DHCP debug messages` 等，可以按实际需要进行勾选来开启对应的调试日志。
- 如在上述 menuconfig 里没有找到想要的调试日志模块，如 UDP 模块，请首先检查 `esp-idf/components/lwip/port/esp32/include/lwipopts.h` 中是否有 `#define UDP_DEBUG`，如果有，可以手动将 `#define UDP_DEBUG LWIP_DBG_OFF` 修改为 `#define UDP_DEBUG LWIP_DBG_ON`。如果没有，可以参照 esp-idf/components/lwip/lwip/src/include/lwip/opt.h 文件下的 `#define UDP_DEBUG LWIP_DBG_OFF`，在 `esp-idf/components/lwip/port/esp32/include/lwipopts.h` 里加一行 `#define UDP_DEBUG LWIP_DBG_ON`。

ESP-IDF 中套接字阻塞和非阻塞的区别是什么？

- 对于读而言，阻塞和非阻塞的区别在于底层没有数据到达时读接口是否立刻返回。阻塞的读会一直等到读取到数据或者异常，非阻塞的读会立刻返回，无论有无数据。
- 对于写而言，阻塞和非阻塞的区别在于底层缓冲区满了后写接口是否立刻返回。阻塞的写，如果底层不可写（底层缓冲区满了或者对端没有 ack 之前发送的数据），这时候的写操作会一直阻塞，直到可写或者异常才会退出；非阻塞的写是可以写多少就写多少，无需等待底层是否可写，返回写入的长度。
- 非阻塞接口调用后不会阻塞当前进程继续执行，阻塞接口调用后会阻塞当前进程执行。

ESP32 是否支持在连上路由后使用上一次成功连接路由器时的 IP 进行通信，如果失败再重新开始认证流程，通过 DHCP 来获取新的 IP？

- 支持，可以在 menuconfig 里使能 `Component config > LWIP ->DHCP: Restore last IP obtained from DHCP server` 选项。
- 需要注意的是，此时不能用静态 IP 来代替，因为静态 IP 设置没有冲突检测，可能会导致 IP 冲突。

使用 socket 编程时，如何实现 connect 超时？

- 将 socket 设置为非阻塞模式，connect() 函数也会是非阻塞，之后通过 select() 函数设置超时时间来判断 socket 是否连接成功，详细操作可参考 “socket 连接超时设置”。

ESP32 使用 SNTP 同步实时时间时发现存在随机延时，进一步分析后发现是 IDF lwip 组件里的 SNTP_STARTUP_DELAY 默认为 1 导致的，是否有办法在不修改 IDF 组件的情况下去掉随机延时？

- 目前没有办法在不修改 IDF 组件的情况下去掉随机延时，需要手动在 lwip 组件里的 lwipopts.h 文件里加上 `#define SNTP_STARTUP_DELAY 0` 这一行代码。这个修改可以减少 SNTP 发送请求的时间，进而减少 ESP 设备上电后到连云成功的整体时间。
- 默认使能此随机延时选项的原因是：这是由 SNTP RFC 协议规定的，防止 SNTP 服务器负载太高，有个随机的 delay 值可以减少设备同时访问的数量。

IPV6 是否支持设置静态 IP？

- IPv6 的 link-local 地址是按照协议规则自动生成的，无需手动设置，所以没有类似 IPv4 一样可以设置静态地址。

4.5.4 Mbed TLS

[English]

ESP8266 OpenSSL 是否支持验证主机名？

支持，目前 ESP8266 OpenSSL 是基于 Mbed TLS 封装的接口，Mbed TLS 支持验证主机名。使用 ESP-TLS 可以根据配置切换 Mbed TLS 与 wolfSSL。

ESP32 使用 Mbed TLS 时如何优化内存？

- 可以在 menuconfig 里开启动态 buffer，具体操作为 `menuconfig -> Component config -> mbedTLS -> Using dynamic TX/RX buffer`（键"Y" 使能）。
- 同时可以使能上一步的 `Using dynamic TX/RX buffer` 里的子选项 `Free SSL peer certificate after its usage` 和 `Free certificate, key and DHM data after its usage`。
- 如果使用的是 v5.0 及以上版本，`Free SSL peer certificate after its usage` 配置功能将不存在，Mbed TLS 默认提供配置 `MBEDTLS_SSL_KEEP_PEER_CERTIFICATE`。如果您需要节省内存，可关闭 `MBEDTLS_SSL_KEEP_PEER_CERTIFICATE`，具体操作为 `menuconfig > Component config > mbedTLS > mbedTLS v3.x related > Keep peer certificate after handshake completion`（键"N" 关闭）。

基于 ESP32 模组连接 HTTPS Server, 报错如下，是什么原因？

```
free heap size: 181784 bytes
I (4285) esp_https_server: Starting server
E (4285) esp_https_server: Could not allocate memory
I (4295) example: Error starting server!
I (4295) SSDP Server: SSDP server started
free heap size: 178636 bytes
```

- 当前报错是由于内存不足导致。从日志信息来看，是使用了 esp_get_free_heap_size() API 打印了当前剩余内存，但此剩余内存包含了芯片内部 RAM 和外部 PSRAM 总容量的剩余内存。
- mbedTLS 默认使用内部 RAM 内存，可使用 esp_get_free_internal_heap_size() 获取内部剩余内存。
- 如果模组带外部 PSRAM，可将 `menuconfig > Component config > mbedTLS > Memory allocation strategy > Internal memory` 配置改为 `menuconfig > Component config > mbedTLS > Memory allocation strategy > External SPIRAM` 进行测试。

基于 ESP32 解析主机名称时，出现如下报错，是什么原因？

```
getaddrinfo() returns 202, addrinfo=0x0
```

- 当前报错日志是因为 DNS 请求超时。
- 您可用开启 Debug 等级的 DNS 日志或捕捉无线包来进一步分析。
- 可在 `esp-idf/components/lwip/lwip/src/include/lwip/opt.h` 文件中增加 `#define DNS_DEBUG LWIP_DBG_ON` 代码，然后开启 `Component config > LWIP > Enable LWIP Debug` 配置来获取 Debug 等级的 DNS 日志。

4.5.5 MQTT

[English]

如何使用 MQTT 配置服务器地址为自主云平台？

可以参考 MQTT 例程。

使用 ESP8266 release/v3.3 版本的 SDK 测试 examples/protocols/esp-mqtt/tcp 例程，配置 Wi-Fi 账号、密码，连接默认配置的服务器，出现连接失败，log 如下，是什么原因？

```
W (4211) MQTT_CLIENT: Connection refused, not authorized
I (4217) MQTT_CLIENT: Error MQTT Connected
I (4222) MQTT_CLIENT: Reconnect after 10000 ms
I (4228) MQTT_EXAMPLE: MQTT_EVENT_DISCONNECTED
I (19361) MQTT_CLIENT: Sending MQTT CONNECT message, type: 1, id: 0000
```

当出现如上报错，表示服务器拒绝了连接，原因是客户端错误的 MQTT 用户名和密码导致服务端认证没有通过。建议您确认是否使用了正确的 MQTT 用户名和密码。

ESP-IDF 中 MQTT 组件 keepalive 的默认值是多少？

默认值为 120 s，在 `mqtt_config.h` 中通过 `MQTT_KEEPALIVE_TICK` 定义。

MQTT 支持自动重连吗？

- MQTT 的自动重连由 esp_mqtt_client_config_t 中的成员变量 `disable_auto_reconnect` 控制，该变量值默认为 `false`，表示使能自动重连。
- 可以使用 `reconnect_timeout_ms` 设置重连超时时间。

ESP-IDF 支持的 MQTT 版本有哪些？

ESP-IDF 目前支持的 MQTT 版本为 MQTT 3.1，MQTT 3.1.1 和 MQTT 5.0。

ESP-IDF 里 Wi-Fi 连接断开的时候，之前 MQTT 上层协议申请的内存会自动释放吗？

- 不会自动释放，但对于用户而言不需要关心这部分内存。用户要关心的是 ESP 给用户封装的应用层。
- 对于 MQTT 应用层组件，用户初始化 MQTT 的时候会获得一个 MQTT 句柄，用户只需关心这个句柄里的内存，在不用 MQTT 的时候调用 `stop` 或者 `destroy` 释放对应 MQTT 内存即可。对于 Wi-Fi 断开与连接，用户也不需要通过 `stop` 或者 `destroy` 释放这个 MQTT 句柄然后再重新申请 MQTT 句柄。因为 MQTT 组件里有自动重连机制。

ESP32-C3 MQTT 是否能不设置对应的 `client_id` 而将 `client_id` 默认配置为空字符串？

- 可以，可通过在应用代码里设置 `set_null_client_id` 为 `true` 来实现。

使用 ESP-IDF MQTT 客户端发布 QoS 为 1 或者 2 的数据后，当 `MQTT_EVENT_PUBLISHED` 触发时是否意味着已经收到了对端合适的 ack 来证明这次发布已完成？还是仅仅只能说明成功发送了一次数据给服务器？

`MQTT_EVENT_PUBLISHED` 事件触发代表代理已确认收到客户端的发布的 QoS 为 1 或者 2 的消息，证明这次发布已经顺利完成。

ESP MQTT 客户端断开连接后，如何手动释放 MQTT 资源？

手动调用 esp_mqtt_client_destroy API 即可。

ESP32 Wi-Fi 和低功耗蓝牙共存时，MQTT keepalive 时间该如何配置？有没有什么合适的配置时间？

- 在 ESP32 中使用 Wi-Fi 和低功耗蓝牙共存时，应该合理配置 MQTT keepalive 时间。由于 Wi-Fi 和低功耗蓝牙都需要消耗系统资源，因此如果 keepalive 时间设置得太短，可能会导致系统负载过高，影响系统稳定性和性能。
- 通常情况下，建议根据实际情况设置 MQTT keepalive 时间，以确保设备在线的同时，尽可能减少系统资源的消耗。在 Wi-Fi 和低功耗蓝牙共存的情况下，可以考虑将 MQTT keepalive 时间设置为较长的时间，如可以配置为 30 s、60 s 等，这样可以减少设备与 MQTT broker 之间的通信次数，降低系统负载。
- 需要注意的是，如果 keepalive 时间设置得太长，当设备掉线时，可能需要等待较长时间才能发现设备离线，这可能会影响实时性和可靠性。因此，需要根据实际需求和系统性能来合理设置 MQTT keepalive 时间。

ESP MQTT 客户端的 disconnect 事件消息什么时候才会触发？

disconnect 消息只有在以下情况出现：

- MQTT 建立连接时，TCP 连接错误
- MQTT 建立连接时，MQTT 连接错误
- 自行主动调用了 `disconnect` 函数
- 接收或发送数据异常
- 规定时间内没收到对端 MQTT `PING RESPONSE`
- 发送 MQTT PING 请求失败
- 重新连接

ESP32 MQTT 客户端与服务器断开后会自动尝试重新连接吗？

ESP MQTT 客户端里的 `esp_mqtt_client_config_t` 结构体配置里有 `disable_auto_reconnect` 参数，可以通过配置这个参数为 `true` 或者 `false` 来决定是否需要 MQTT 自动重连，MQTT 默认会自动进行重连。

如何检测 ESP32 是否已经与 MQTT 服务器断开？

检测 ESP32 是否已经与服务器断开可以使用 MQTT 的 `PING` 机制。也就是配置 ESP-MQTT 中 `esp_mqtt_client_config_t` 结构体里的 `keepalive` 参数 `disable_keepalive` 和 `keepalive`，比如将 `disable_keepalive` 配置为 `false`（默认参数也是 `false`，即默认开启 keepalive 机制），然后配置 `keepalive` 参数为 120 s 来设置保活时间，默认为 120 s。这样 MQTT 客户端会定期发送 `PING` 来检测和 MQTT 服务器的连接是否正常。

4.5.6 其他协议

[English]

ESP32 如何优化通信延时？

- 建议关闭 Wi-Fi 休眠功能，调用 API `esp_wifi_set_ps(WIFI_PS_NONE)`。
- 建议在 menucongfig 关掉 `AMPDU` 功能。

ESP8285 是否支持 CCS (Cisco Compatible eXtensions)？

ESP8285 不支持 CCS (Cisco Compatible eXtensions)。

ESP32 是否支持 LoRa (Long Range Radio) 通信？

ESP32 自身并不支持 LoRa 通信，芯片没有集成 LoRa 协议栈与对应的射频部分。ESP32 可以外接集成 LoRa 协议的芯片，作为主控 MCU 连接 LoRa 芯片，可以实现 Wi-Fi 与 LoRa 设备的通信。

ESP32-S2 在调用 `esp_netif_t* wifiAP = esp_netif_create_default_wifi_ap()` 后通过 `esp_netif_destroy(wifiAP)` 注销会产生 12 字节的内存泄露，什么原因？

- 需要在 `esp_netif_destroy(wifiAP)` 前额外调用 `esp_wifi_clear_default_wifi_driver_and_handlers` 这样才是正确的注销流程，此时可发现内存泄露的情况已消失。
- 也可以直接调用 `esp_netif_destroy_default_wifi(wifiAP)`，该接口在 ESP-IDF v4.4 版本以上支持。

如何实现证书自动下载功能?

CHIP: ESP32

具体操作详情参考 aws 下面证书自动下载功能 。

ESP-IDF 里如何根据错误码来获取更多的调试信息？

- ESP-IDF 3.x 版本下的错误码 (errno) 列表直接存在于 IDF 中，点击 errno.h 可以进行查询。
- ESP-IDF 4.x 版本的 `errno.h` 位于编译器工具链下，比如，对于 esp-2020r3 而言，errno.h 的路径为 `/root/.espressif/tools/xtensa-esp32-elf/esp-2020r3-8.4.0/xtensa-esp32-elf/xtensa-esp32-elf/include/sys/errno.h`。

ESP8266_RTOS_SDK 是否支持 TR-069 协议？

不支持，ESP8266_RTOS_SDK 本身并不提供对 TR-069 协议的原生支持，但开发者可以根据自己的需求自行实现 TR-069 协议栈，并将其集成到 ESP8266_RTOS_SDK 中。也可以使用现有的第三方 TR-069 协议栈进行集成。总之，ESP8266_RTOS_SDK 可以支持 TR-069 协议，但需要进行自行集成实现。

ESP32 支持 SAVI 吗？

不支持，SAVI (Source Address Validation Improvements) 是通过监听控制类报文（如 ND、DHCPv6），即 CPS (Control Packet Snooping)，在接入设备上（AP 或交换机）为终端建立基于 IPv6 源地址、源 MAC 地址及接入设备端口的绑定关系，进而对通过指定端口的 IP 报文进行源地址校验。只有报文源地址与绑定表项匹配时才可以转发，保证网络上数据报文源地址真实性。这种一般针对于交换机或者企业级 AP 路由器的，是策略协议。目前 ESP32 支持 IPv6 链路本地地址、全球地址通信。

以太网和 Wi-Fi 并存时，优先使用以太网传输数据吗？

:CHIP: ESP32 :

- 先调用 esp_netif_get_route_prio 查看以太网和 Wi-Fi 的优先级，如果 Wi-Fi 的优先级高于以太网，可以通过修改 `esp_netif_t` 结构体里的 `route_prio` 来改变优先级。

4.6 配置

[English]

4.6.1 安卓 ESP-Touch 可以添加自己想要广播的数据吗（如添加设备 ID，希望 ESP32 能接收到这个 ID）？

- ESP-Touch 是一个用于在手机和 ESP8266/ESP32 之间建立 Wi-Fi 连接的通信协议，它使用一种特殊的广播方式来传输 Wi-Fi SSID 和密码信息。通常情况下，ESP-Touch 广播的数据是固定格式的，不能直接添加自定义数据。
- 如果您希望 ESP32 能够接收到设备 ID 等自定义数据，您可以考虑使用其他通信协议，例如 MQTT、HTTP 等。使用这些协议，您可以自由地定义数据格式，并在 Android 应用程序和 ESP32 之间进行通信。
- 如果您仍然希望广播自定义数据，建议使用 BluFi，这是基于 Bluetooth LE 的配网协议。请参见：
  - Android APP：https://github.com/EspressifApp/EspBlufiForAndroid。
  - iOS APP：https://github.com/EspressifApp/EspBlufiForiOS。

4.7 安全

[English]

4.7.1 ESP8266 的固件是否能被读取？

ESP8266 固件放置在外部 flash，数据可被外部读取。并且 ESP8266 不支持 flash 加密，所有数据均为明文。

4.7.2 ESP8285 是否可以固件加密？

- ESP8285 芯片自身未有固件加密功能，无法完成固件加密。
- ESP32 及 ESP32-S2 芯片支持固件加密功能，用户可以考虑使用这两款芯片。
- 如仍需使用 ESP8285，用户可以选择外置加密芯片完成部分数据加密。

4.7.3 secure boot v1 和 secure boot v2 有什么区别？

secure boot v2 相较于 secure boot v1 主要做了以下方面的改进：- bootloader 和 app 使用相同的签名格式。- bootloader 和 app 使用统一的签名密钥。

当前，仅 ESP32 v3.0 以下版本推荐使用 secure boot v1，ESP32 v3.0 及以上版本、ESP32-C3、ESP32-S2 和 ESP32-S3 推荐使用 secure boot v2。

4.7.4 开启 secure boot 后，为什么编译报错缺少文件？

错误日志：/Makefile.projbuild:7：/f/ESP32Root/secure_boot_signing_key.pem。

报错原因：security boot 是固件签名校验的功能，该功能需要生成密钥对。- 启用 secure boot v1 时生成密钥对的方法请参考 secure boot v1 生成密钥。- 启用 secure boot v2 时生成密钥对的方法请参考 secure boot v2 生成密钥。

4.7.5 模组使能 secure boot 后是否可以再次烧录？

- 若 secure boot v1 配置为 one-time，则仅支持烧录一次，不可以再重新烧录 bootloader 固件。
- 若 secure boot v1 配置为 reflashable，则可以重新烧录 bootloader 固件。
- secure boot v2 允许重新烧录 bootloader 和 app 固件。

4.7.6 模组使能 flash encrypted，重新烧录后出现 `flash read error` 错误。如何解决？

模组开启加密功能后将不支持明文固件烧录，常见的错误请参考 重新烧录可能出现的错误。可以通过 espefuse 脚本关闭加密后再次烧录明文数据，或者参考 flash 加密示例，将加密后的固件数据烧录到设备上。

注解： flash encrypted 加密开关存在次数限制。

4.7.7 ESP32 打开 flash 加密和 secure boot 后，如何关闭？

- 如果您使用的是 one-time flash (Release) 模式，那么 flash 加密和 secure boot 都是不能关闭的。
- 如果您使用的是 reflashable (Development (NOT SECURE)) 模式，那么 flash 加密可以关闭，请参见 关闭 Flash 加密；secure boot 不能关闭。

4.7.8 ESP32 保护固件安全的方式有哪些？

- ESP32 支持 flash 加密与 secure boot。
- flash 加密参考文档：flash 加密。
- 安全引导参考文档：secure boot。
- 安全引导 V2 参考文档：v3.0 版本芯片的安全引导 V2。

4.7.9 ESP32 启动 flash 加密后进行 GDB 调试，为何会不断复位重启？

- ESP32 启动了 flash 加密或 secure Boot 后，默认将会禁用 JTAG 调试，请参见 注意事项和补充内容。
- 可以通过 esptool 工具包中的 `espefuse.py summary` 脚本指令读取当前芯片 JTAG 状态。

4.7.10 ESP32 芯片如何开启 Flash 加密？

- 在 ESP-IDF 编译配置中，通过 menuconfig 或 idf.py menuconfig 中的 `Security features -> Enable flash encryption on boot (READ DOCS FIRST)` 修改。
- 请参见 Flash 加密说明。

4.7.11 ESP32 的 GPIO0 拉低后无法进入下载模式，日志打印“download mode is disable”是什么原因？

- ESP32 芯片上电打印“download mode is disable”日志，说明该芯片的 UART 下载模式 (UART download mode) 已被禁用。您可以检查该芯片 eFuse 中的 `UART_DOWNLOAD_DIS` 位，查看该模式是否被禁用。
- 注意，启用 flash 加密的量产模式后，UART 下载模式将默认被禁用。更多信息，请参考 UART ROM download mode。

4.7.12 在 Arduino 开发环境中使用 ESP32 能开启 secure boot 功能吗？

- 不能，如果要使用 Arduino 进行开发，开启 secure boot 功能的唯一方法是将 Arduino 作为 IDF 组件使用。

4.7.13 secure boot 和 flash 加密的使用场景有哪些？

- 启用 secure boot 后，设备将仅加载运行指定密钥签名后的固件。因此，启用 secure boot 可以避免设备加载非法的固件、防止对设备刷写未经授权的固件。
- 启用 flash 加密后，flash 上存储固件的分区以及被标识为“encrypeted”的分区中的数据将被加密。因此，启用 flash 加密可以避免 flash 上的数据被非法查看，并且从 flash 上拷贝的固件数据无法应用到其他设备上。

4.7.14 secure boot 和 flash 加密中涉及的存储在 eFuse 数据有哪些？

- secure boot v1 中使用的存储在 eFuse 数据请参考 secure boot v1 efuses。
- secure boot v2 中使用的存储在 eFuse 数据请参考 secure boot v2 efuses。
- flash 加密中使用的存储在 eFuse 数据请参考 flash 加密 efuses。

4.7.15 启用 secure boot 失败，提示 “Checksum failure”，怎么解决？

- 启用 secure boot 后，bootloader.bin 的大小将增大，请检查引导加载程序分区的大小是否足够存放编译得到的 bootloader.bin。更多说明请参考 引导加载程序大小。

4.7.16 启用 NVS 加密失败，提示 `nvs: Failed to read NVS security cfg: [0x1117] (ESP_ERR_NVS_CORRUPT_KEY_PART)`，怎么解决？

- 启用 NVS 加密前，建议先使用烧录工具擦除一次 flash，然后烧录包含使能 NVS 加密的固件。

4.7.17 启用 flash 加密后，提示 `esp_image: image at 0x520000 has invalid magic byte (nothing flashed here)`，怎么解决？

- 启用 flash 加密后，将尝试对所有 app 类型的分区的数据进行加密，当 app 分区中没有存储对应的 app 固件时，将提示该 log。您可以在启用 flash 加密时对所有 app 类型的分区烧录预编译的 app 固件来避免出现这种警告。

4.7.18 使能 `CONFIG_EFUSE_VIRTUAL` 选项后，开启 flash 加密，为何相关数据未被加密？

- Virtual eFuses 功能目前仅仅用于测试 eFuse 数据的更新，启用该功能后，flash 加密功能并未完全开启。

4.7.19 可以向一个未使能 flash 加密的设备中通过 OTA 更新一个使能了 flash 加密的 app 固件吗？

- 可以，请在编译时取消选中 `Check Flash Encryption enabled on app startup`。

4.7.20 如何撤销 secure boot 的 key？

- 撤销 secure boot key 的操作是在 `new_app.bin` 固件中完成的。首先 `new_app.bin` 必须附带两个签名。然后，下发 `new_app.bin` 到设备上。最后，当旧的签名校验通过后，通过 `new_app.bin` 中的 `esp_ota_revoke_secure_boot_public_key()` 执行撤销旧 key 的操作。注意，如果您使用了 OTA 回滚方案，请在 `esp_ota_mark_app_valid_cancel_rollback()` 返回 `ESP_OK` 后再调用 `esp_ota_revoke_secure_boot_public_key()`。更多说明请参考 Key Revocation。

4.7.21 启用 secure boot 或者 flash 加密（开发模式）后，无法烧录新固件，提示 `Failed to enter Flash download mode`，怎么解决？

- 这种提示通常代表您使用的烧录命令不正确。请使用 `idf.py` 脚本执行 `idf.py bootloader`、`idf.py app` 命令编译 `bootloader.bin`、`app.bin`。然后根据编译后的提示使用 `idf.py` 执行烧录命令。如果还不能烧录程序，请使用 `espefuse.py -p PORT summary` 命令查看当前设备的 eFuse，并检查 `flash download mode` 是否是 enable 状态。

4.7.22 在配置了 ESP-IDF 环境的终端里输入 `espefuse.py read_protect_efuse BLOCK3` 指令对 Efuse BLOCK3 进行读保护后，再输入 `esp_efuse_read_block()` 读取 Efuse BLOCK3 的数据，数据全为 0x00，是什么原因？

- Efuse BLOCK3 被读保护之后就不能再被读取了。

4.7.23 如何通过预烧录 eFuse 的方式使能 secure boot 或者 flash 加密？

默认情况下，可以通过向设备中烧录使能了 secure boot 或者 flash 加密的固件来启用 secure boot 或者 flash 加密。用户也可以通过下述两种通过预烧录 eFuse 的方式使能 secure boot 或者 flash 加密：- flash_download_tool 在使能 secure boot 或者 flash 加密时会自动预烧录 eFuse. - 通过使用 espsecure.py 和 espefuse.py 来生成密钥以及烧录对应的 eFuse 存储块。

4.7.24 启用 Secure Boot 后，使用 `idf.py build` 命令无法烧录新的 bootloader.bin？

启用 Secure Boot 后，请使用 `idf.py bootloader` 命令编译新的 bootloader.bin。然后通过 `idf.py -p (PORT) bootloader-flash` 命令烧录新的 bootloader.bin。

4.7.25 启用 Secure Boot 或者 flash 加密后，如何查看设备中关于安全特性的信息？

请使用 `esptool.py --no-stub get_security_info` 命令查看设备的安全信息。

4.7.26 启用 Secure Boot 或者 flash 加密后，OTA 时应该注意什么？

- 启用 Secure Boot 后，你必须对 OTA 要使用的新固件进行签名，否则新固件无法被应用到设备上；
- 启用 flash 加密后，在生成新固件时，请保持使能 flash 加密的选项。

4.8 储存

[English]

4.8.1 FAT 文件系统

[English]

意外的断电导致 FatFs 文件系统损坏如何改善？

因为 FatFs 设计不支持 write transactions，因此意外断电可能会导致分区错误，并且无法通过简单修改 FatFs 来修复这个问题。目前建议：可以通过创建两个相同的 FatFs 分区进行双备份来从应用层避免该问题，也可以选择使用具有更高安全性的文件系统，如：LittleFS 和 SafeFAT （收费）。

如何制作并烧录一个 FatFs 文件系统的镜像？

ESP-IDF 中未提供相关工具，需要借助第三方工具，完整示例过程如下：

- 第一步：使用 mkfatfs 工具从一个指定文件夹创建镜像，从 file_image 文件夹创建大小为 1048576 Byte、名为 fat_img.bin 的镜像：

```
./mkfatfs -c file_image -s 1048576 ./fat_img.bin
```

- 第二步：烧录镜像到 0x110000 地址：

```
esptool.py -p /dev/ttyUSB1 -b 460800 --before default_reset --after hard_
↪reset write_flash --flash_mode dio --flash_size detect --flash_freq 80m␣
↪0x110000 ~/Desktop/fat_img.bin;
```

- 第三步：在程序中挂载：

```
static void initialize_filesystem() {
  static wl_handle_t
  wl_handle = WL_INVALID_HANDLE;
  const esp_vfs_fat_mount_config_t
  mount_config = { .max_files = 10, };
  ESP_LOGI(TAG, "Mounting FATfilesystem");
  esp_err_t err = esp_vfs_fat_spiflash_mount("/spiflash", "storage", &mount_
↪config, &wl_handle);
  if (err != ESP_OK) {
      ESP_LOGE(TAG, "Failed to mount FATFS (%s)", esp_err_to_name(err));
      return;
  }
}
```

注解： 这里烧录的地址一定要是分区表里 FatFs 挂载时对应分区的地址，创建的镜像需要与分区表中设置的大小一致。menuconfig 中的 `Component config -> Wear Levelling -> Wear Levelling library sector size` 需要设置为 512，否则将导致挂载失败。

FATFS 与 SPIFFS 这两种文件系统有何差异，我们怎么选择？

请参考 文件系统。

FatFs 支持的最大容量是多少？

由于 Windows 系统的限制，目前 FatFs 普遍最大只在 32 GB 的存储设备上使用。大于 32 GB 的存储设备默认会使用 exFAT 等文件系统。

使用 FAT 文件系统时，文件名稍微长一点的文件无法打开，该如何处理？

- 可以在 `menuconfig -> Component config -> FAT Filesystem support -> Long filename support` 中进行修改，选择 ``Long filename buffer in heap`` 或 `Long filename buffer on stack` 配置项。然后可以在 `Component config -> FAT Filesystem support -> Max long filename length` 中修改最大的文件名长度。

使用 ext_flash_fatfs 示例测试，分区表中将 fatffs 分区设置小于 512 KB 时，会报 `vfs_fat_spiflash:f_mks failed(14),config:Failed to mount FATFS(ESP_FAIL)` 错误，如何解决？

- FAT 分区的最小扇区数是 128，所以文件系统的最小尺寸是 128*4+4*4=528 KB，额外的 4 个扇区需要用于磨损均衡信息，所以至少需要保证 fatffs 分区 size 不小于 528 KB。

4.8.2 非易失性存储 (NVS)

[English]

若每分钟保存或者更新数据到 flash 中，ESP32 设备的 NVS 能否满足该需求？

根据 NVS 说明，NVS 库在其操作中主要使用两个实体：页面和条目。逻辑页面对应 flash 的一个物理扇区。假设 flash 扇区大小为 4096 字节，每个页面可容纳 126 个条目（每个条目大小为 32 字节），页面的其余部分用于页头部（32 字节）和条目状态位图（32 字节）。每个扇区的典型 flash 寿命为 100 k 个擦除周期。假设期待设备的运行时间为 10 年，每分钟写入 flash 的数据大小为 4 字节，并且不使用 flash 加密，计算 flash 写操作的次数为：60×24×365×10=5256000。这样，在 NVS 中会导致不超过 42 k 个擦除周期 (5256000/126)，而 42 k < 100 k，因此，即使在没有多扇区影响

的情况下也可以支持。在实际使用中，分配给 NVS 的大小一般为多个扇区，NVS 会在多扇区之间分配擦除周期，那么每个扇区的擦除周期的次数必然小于 42 k。

因此，NVS 可以满足该擦写需求。

NVS 是否具有磨损均衡？

是的，NVS（Non-Volatile Storage）具有磨损均衡（wear leveling）功能。在使用 flash 存储数据时，由于 flash 的写入和擦除次数有限，会导致一些存储块的寿命比其他块更短，从而影响整个存储器的寿命。为了解决这个问题，NVS 使用的不是 ESP-IDF 中的 wear_levelling 组件，而是在其内部实现的一种擦写平衡机制，将数据均匀地分布在 flash 的各个存储块中，使得每个存储块的使用次数尽可能相等，从而延长整个 flash 的寿命。

NVS 扇区是否会因写入时意外断电而损坏？

NVS 设计之初就具有抵抗意外断电的能力，因此不会损坏。

配置好的 Wi-Fi SSID 和 PASSWORD 在 ESP 系列开发板上重新上电后是否会消失，需要重新输入吗？

- 默认会存在 NVS 里，不会因为掉电而消失，您也可以通过 `esp_wifi_set_storage()` 设置，此时分为两种情况：
  - 如果想要实现掉电保存 Wi-Fi SSID 和 PSAAWORD，可通过调用 `esp_wifi_set_storage(WIFI_STORAGE_FLASH)` 将 Wi-Fi 信息存储在 flash 内。
  - 如果想要实现掉电不保存 Wi-Fi SSID 和 PASSWORD 的操作，可通过调用 `esp_wifi_set_storage(WIFI_STORAGE_RAM)` 将 Wi-Fi 信息存储在 RAM 内。

如何实现存储的用户数据支持掉电保存，不被 OTA 擦除，且支持重写或改写？

- 根据需求，需要使用非易失性存储，存储数据的区域只有 eFuse 或 flash。考虑到需要修改，只能使用 flash。推荐使用 NVS 或 MFG 机制。请参考：
  - MFG 量产程序
  - NVS 分区生成程序

4.8.3 PSRAM

[English]

使用 ESP32 模组，如何查看模组的 PSRAM 的大小？

对于使用 ESP32 模组的情况，可以使用 ESP-IDF 中的 esp_spiram_get_size() 函数来获取模组的 PSRAM 大小。该函数会返回 PSRAM 的总大小（单位为字节），可以用于进行内存分配和管理等操作。

以下是获取 PSRAM 大小的示例代码：

```
size_t psram_size = esp_spiram_get_size();
printf("PSRAM size: %d bytes\n", psram_size);
```

注意，该函数需要在使用 PSRAM 之前调用，以确保正确获取 PSRAM 的大小。另外，需要先在 `make menuconfig` 中配置开启 PSRAM 功能，以便正确启用和配置 PSRAM。此外，PSRAM 的大小可通过 bootloader 的 log 信息来查看。

ESP32 外接 PSRAM 后，如何更改 PSRAM 的 clock 来源？

在 menuconfig 中修改。具体位置：menuconfig -> Component config -> ESP32-specific -> SPI RAM config。

ESP32 模组挂载 8 MB PSRAM, 为何实际映射的只有 4 MB？

- 使用 ESP32 芯片建议搭配使用官方 ESP-PSRAM 芯片。
- 片外 RAM 最大可映射 4 MB (0x3F80_0000 ~ 0x3FBF_FFFF) 到数据地址空间，可参考 ESP32 技术规格书 中 3.1.4 节存储器映射的说明。
- 对于 8 MB PSRAM，可参考例程 himem 访问其余的 4 MB 空间。

使用 ESP32 开发板，上面用了官方 PSRAM 芯片 PSRAM64H，当更换了另一个型号的 PSRAM 芯片后，运行 ESP-IDF 的例程并开启 PSRAM 配置，却无法正常识别，是什么原因？

- 更换 PSRAM 芯片的型号，需要在 menuconfig -> Component config -> ESP32-specific -> Support for external, SPI-connected RAM -> SPI RAM config -> Type of SPI RAM chip in use 中修改相应配置选项。
- 若更换的 PSRAM 芯片型号在 menuconfig 中没有相应的配置选项，则需要自行加入 PSRAM 芯片的驱动。

使用 ESP32-WROOM-32E 模组下载 hello-world 例程，打印如下报错，是什么原因？

```
E (225) psram: PSRAM ID read error: 0xffffffff
E (225) spiram: SPI RAM enabled but initialization failed. Bailing out.
```

报错原因是：软件上开启了 PSRAM (`Component config > ESP32-specific > Support for external, SPI-connected RAM`) 的设置，但硬件上没有 PSRAM 的支持。

ESP32 支持 16 MB 的 External Flash 和 8 MB 的 External PSRAM 共存吗？

ESP32 可以支持 16 MB 的 External Flash 和 8 MB 的 External PSRAM 共存使用。

4.8.4 SD/SDIO/MMC 驱动

[English]

ESP8266 是否可以搭配 TF 卡使用？

不建议这么使用。

- 虽然硬件上是可以连接的（通过 SPI 与 TF 卡通信），但是因为 ESP8266 的资源有限，根据不同的应用场景，很可能会出现内存不足等情况。所以不建议 ESP8266 搭配 TF 卡使用。
- 如果您只需要单 Wi-Fi 模组，并且要连接 TF 卡，建议使用 ESP32-S2 芯片。

ESP32-S3 支持的 EMMC 最大容量是多少？

2 TB 是 SD 和 eMMC 协议的限制。ESP32-S3 没有具体的最大容量限制。但如果你的应用是建立在文件系统之上的，最大容量也可能受到文件系统的限制。

ESP32 外接 eMMC 卡时，是否支持 DDR52/HS200/HS400/SDR52 模式？

- 在频率为 40 MHz 时，支持 DDR52 模式，其他模式不支持。支持的模式参见 Supported Speed Modes 说明。

4.8.5 SPI Flash

[English]

ESP32 flash 空间与使用要求是什么？

外部 flash 可以同时映射到 CPU 指令和只读数据空间。外部 flash 最大可支持 16 MB。

- 当映射到 CPU 指令空间时，一次最多可映射 11 MB + 248 KB。如果一次映射超过 3 MB + 248 KB，则 Cache 性能可能由于 CPU 的推测性读取而降低。
- 当映射到只读数据空间时，一次最多可以映射 4 MB。支持 8-bit、16-bit 和 32-bit 读取。
- 需要在编写代码时指定 flash 分区表，即将 flash 划分为不同的分区，如 app 分区、data 分区、OTA 分区等，并指定每个分区的大小和偏移地址
- 在使用 flash 时，需要注意 flash 的使用寿命问题。由于 flash 的擦写次数有限，需要合理规划和管理 flash 的使用，例如使用 wear leveling 等技术来延长 flash 的寿命。
- 在使用 flash 时，需要注意 flash 写操作会占用 CPU 时间，可能会影响系统的响应速度，因此需要尽量减少 flash 写操作的频率。

ESP8266 的模组，有哪些扇区可以自主使用？

- SDK rel3.0 之前的版本，除 bootloader 与 app bin 外还会在设置的 flash 大小的尾部会保留扇区如下：1 个存放系统信息，1 个存储 OTA 信息，1 个存放 RF 校准信息。
- SDK rel3.0 及其以后的版本使用 partition_table 来管理 flash，除该文件自身与 bootloader 外，其余 bin 文件均在 partition_table 标注。

ESP8266 如何读取 flash 数据？

- 可使用 ESP8266-RTOS-SDK 下的脚本工具读 flash ，读 flash 方式如下：
  - 安装 python 环境以及工具依赖的包文件；
  - 进入 ESP8266_RTOS_SDK/components/esptool_py/esptool 路径下；
  - 执行 `python esptool.py --chip esp8266 --port /dev/ttyUSB0 --baud 115200 read_flash 0x0 0x400000 esp8266.bin`。该命令中 “esp8266.bin” 为自定义名称，读取到的 flash 数据将会生成名为 “esp8266.bin” 的文件，命令中 “/dev/ttyUSB0” 为 linux 环境中的串口号，其他环境以及系统中会有不同。

不同的 ESP32 模组为何出现 flash 擦除时间不一致？

- 不同的 ESP32 模组可能具有不同的 flash 芯片或 flash 控制器，这些硬件组件可能会对擦除时间产生影响，部分型号的 flash 进行擦除时没有空块跳过的机制，所以耗时较长。具体而言，不同的 flash 芯片可能具有不同的擦除时间，例如 SPI flash 和 QSPI flash 具有不同的擦除时间，即使是同一类型的 flash 芯片也可能存在不同的擦除时间，这取决于其封装和生产批次等因素。此外，不同的 flash 控制器的设计和性能也可能对擦除时间产生影响。因此，不同的 ESP32 模组可能会使用不同的 flash 芯片或 flash 控制器，从而导致擦除时间的不一致。

使用 ESP32-S3-WROOM-2-32R8V 模组，设置 flash SPI mode 为 QIO 模式，固件运行时打印如下错误是什么原因？

```
E (47) qio_mode: Failed to set QIE bit, not enabling QIO mode
```

ESP32-S3-WROOM-2-32R8V 模组使用的是 32 MB Octal SPI flash 和 8 MB Octal SPI RAM。请在配置选项中开启 Octal SPI flash 和 Octal PSRAM 的设置：

- `(Top) > Serial flasher config > [*] Enable Octal Flash > Flash SPI mode (OPI)`
- `(Top) > Component config > ESP PSRAM > Support for external, SPI-connected RAM > SPI RAM config > Mode (QUAD/OCT) of SPI RAM chip in use`

如何判断 ESP-IDF 是否已经支持某款 flash？

- 可以参考 Optional features for flash 来进一步了解 ESP-IDF 支持的 flash 信息，需要注意的是，此文档仅说明 ESP-IDF 代码已支持这些 flash 的特性，并不是乐鑫认证的稳定 flash 列表。
- 如需要 flash 选型的进一步支持，可以联系 乐鑫商务。

基于 ESP32-S3 芯片外接 SPI flash，支持单次写入的数据量是多大？

- ESP32-S3 硬件限制外接 SPI flash 只允许单次写入最大 64 字节数据。

4.8.6 SPIFFS 文件系统

[English]

SPIFFS 支持磁盘加密吗？

CHIP: ESP32, ESP32S2, ESP32S3, ESP32C3

- SPIFFS 不提供原生的磁盘加密功能，但 SPIFFS 建立在 flash 上，可以使用 flash 加密对该部分的数据进行加密。可以使用标准的加密库，如 mbedtls 或 OpenSSL，来加密和解密 SPIFFS 中的文件。在写入文件时，先将数据进行加密，然后再写入 SPIFFS。在读取文件时，先从 SPIFFS 读取数据，然后再使用相应的解密算法进行解密。

如何将 ESP32 设备的 key 和 certs 存储到 SPIFFS 中呢？

可将文件生成 SPIFFS 镜像后烧录到对应分区，可参考 SPIFFS 文件系统。

ESP32 是否可以在外挂的 SPI flash 中挂载 SPIFFS 文件系统分区？

在 ESP-IDF v4.0 以及之后的版本添加了该功能。需要注意的是，当挂载了两个分区时，不能多个任务同时向一个分区写入文件。

4.8.7 其他存储相关

[English]

ESP32 是否可以使用 LittleFS 文件系统？

目前 ESP-IDF 未包含 LittleFS，存在第三方移植组件 esp_littlefs，可直接在 ESP-IDF 中使用。匹配 LittleFS 文件系统镜像的工具为 mklittlefs。

ESP32 如何查看芯片内存（例如：DRAM、IRAM、rodata）使用情况？

可以在工程终端目录下输入 `idf.py size-components` 指令来查看静态存储空间使用估算情况。如需查询运行时内存动态申请信息，请使用 heap_caps_get_info 查询。

ESP8266 用户可用的 RTC RAM 是多大？

- ESP8266 用户可用的 RTC RAM 为 0x200。可参见 esp8266.ld 文件说明。

如何使能 exFAT？

CHIP: ESP32

- 需要在代码中将 #define FF_FS_EXFAT 0 修改为 #define FF_FS_EXFAT 1 , 具体的请参考 ffconf.h。

ESP32 分区表中的分区数量有限制吗？

- 分区表的长度为 0xC00 字节（最多可以保存 95 条分区表条目）。参考链接 分区表 的说明。

ESP32 如何读取芯片剩余内存？

- 可通过 esp_get_free_heap_size() 来读取芯片 RAM 剩余内存。

在 ESP-IDF 下使用 xTaskCreateStatic() 需要注意什么？

- 可以参考 xTaskCreateStatic() 说明。

4.9 系统

[English]

4.9.1 如果我的应用不需要看🄵狗，如何关闭看🄵狗？

ESP-IDF 中主要有两种看门狗：任务看门狗和中断看门狗。您可以在配置菜单中关闭这两种看门狗。更多细节请参考 Watchdogs。

4.9.2 RTOS SDK 和 Non-OS SDK 有何区别？

主要差异点如下：

Non-OS SDK

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

RTOS SDK

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

- RTOS 版本兼容 Non-OS SDK 中的 Wi-Fi 接口、SmartConfig 接口、Sniffer 相关接口、系统接口、定时器接口、FOTA 接口和外围驱动接口，不支持 AT 实现。

4.9.3 ESP8266 启动时 LOG 输出 ets_main.c 有哪些原因？

ESP8266 启动时打印 `ets_main.c`，表示没有可运行的程序区，无法运行；遇到这种问题时，请检查烧录时的 bin 文件和烧录地址是否正确。

4.9.4 ESP8266 编译 Non-OS SDK 时 IRAM_ATTR 错误是什么原因？

如果需要在 IRAM 中执行功能，就不需要加 `ICACHE_FLASH_ATTR` 的宏，那么该功能就是放在 IRAM 中执行。

4.9.5 ESP8266 main 函数在哪里？

ESP8266 用户 SDK 部分没有 main 函数。main 函数处于一级 bootloader 并且固化在芯片 ROM 中，用于引导二级 bootloader。二级 bootloader 入口函数为 ets_main，启动后会加载用户应用中的 user_init，引导至用户程序。

4.9.6 ESP8266 partition-tables 特殊注意点？

ESP8266 partition-tables 相对 ESP32 对于 ota 分区有一定的特殊要求，这是由于 ESP8266 cache 特性导致。详情参考 ESP8266 partition-tables 偏移与空间。

4.9.7 应用层与底层的 bin 文件可以分开编译码？

不支持分开编译。

4.9.8 ESP32 模组 flash 使用 80 Mhz 有什么注意事项吗？

乐鑫模组发售前已经过稳定性测试，测试可以支持 80 MHz 频率。根据稳定性测试数据，80 MHz 的频率不会影响使用寿命和稳定性。

4.9.9 ESP32 系统软件复位 API？

软件复位 API：`esp_restart()`。

4.9.10 使用 esp-idf 按汇编操作寄存器，是否涉及到调用不可编辑的库文件？

- 如果单纯使用读写寄存器指令或者汇编指令，不存在调用库文件的问题。如果调用 esp-idf 预制的函数，则可能会遇到调用 lib 函数的情况。
- 不推荐在 ESP32 中使用汇编操作，如果部分场景想要提高速度，可以读写寄存器来完成部分操作。

4.9.11 使用 ESP-IDF 测试程序，如何设置可在单核模组上下载程序？

程序编译时，使用 make menuconfig 指令进入配置界面，进行如下配置，可在单核模组上下载程序；在配置界面中，按键 Y 为启动，N 为关闭。

- `Component config --> FreeRTOS --> Run FreeRTOS only on first core`

4.9.12 使用 esp-idf，如何使能 ESP32 的双核模式？

esp-idf 一般情况下默认配置的是双核模式，您可以在 menuconfig 中进行单双核的修改：menuconfig -> Component config -> FreeRTOS -> Run。

FreeRTOS only on first core 使能即为单核，未使能默认双核。

4.9.13 使用 ESP32-D0WD 芯片是否可以存储用户程序？

不可以，用户程序必须使用外挂 Flash 进行存储，片上 ROM 不能存储用户程序。ROM 内存放的程序为芯片一级 bootloader，为了保护出厂程序不被破坏，该区域为只读存储。

4.9.14 ESP32 进入低功耗模式时，PSRAM 中的数据会丢失吗？

- Modem-sleep/Light-sleep 模式时，PSRAM 中的数据不会丢失。
- Deep-sleep 模式时，CPU 和大部分外设都会掉电，PSRAM 的数据会丢失。

4.9.15 请问 ESP32 CPU 系统时间是否由系统滴答时钟生成？精度如何？

CPU 系统时间是由 esp_timer 内部的 64 位硬件定时器 CONFIG_ESP_TIMER_IMPL 产生的，是微秒级的时间分辨率。参见 高精度时钟说明。

4.9.16 ESP32 的 flash 和 psram 的时钟频率如何修改？

在 menuconfig 中修改。- flash 时钟频率：menuconfig -> Serial flasher config -> Flash SPI speed。- PSRAM 时钟频率： Component config -> ESP32-specific -> SPI RAM config -> Set RAM clock speed。

4.9.17 使用 ESP32-SOLO-1 模组，esp-idf 如何设置可在单核模组上运行？

使用 menuconfig 指令进入配置界面，Component config –> FreeRTOS –> Run FreeRTOS only on first core（启动此选项）可在单核模组上运行下载。

4.9.18 esp-idf 是否可以配置 time_t 为 64 bit ？(现在是 32 bit)

当前暂时不支持，预计在 release/v4.2 或更高版本种支持。如果配置支持 time_t 64 bit 自定义工具链，可以使能 make menuconfig 中 SDK tool configuration -> SDK_TOOLCHAIN_SUPPORTS_TIME_WIDE_64_BITS 。

4.9.19 固件如何区分主芯片是 ESP8285 还是 ESP8266？

通常使用外部工具 esptool 来读取芯片类型。可以在固件中根据 python 代码示例，读取芯片对应寄存器位，并进计算判断得出。

```
def get_efuses(self):
# Return the 128 bits of ESP8266 efuse as a single Python integer
return (self.read_reg(0x3ff0005c) << 96 | self.read_reg(0x3ff00058) << 64 |␣
↪self.read_reg(0x3ff00054) << 32 | self.read_reg(0x3ff00050))

def get_chip_description(self):
  efuses = self.get_efuses()
  is_8285 = (efuses & ((1 << 4) | 1 << 80)) != 0  # One or the other efuse␣
↪bit is set for ESP8285
  return "ESP8285" if is_8285 else "ESP8266EX"
```

4.9.20 ESP32 能否以动态库的方式加载库文件运行？

ESP32 不支持动态库的方式加载库文件，只支持□态库。

4.9.21 ESP32 如何减小系统对 IRAM 内存的占用？

- 请将 menuconfig > Component config > LWIP > Enable LWIP IRAM optimization (键 N 禁用) 配置选项禁用。
- 请更改 menuconfig > Compiler option > Optimization Level > Optimize for size (-Os) 中的配置选项。
- 请将 menuconfig > Component config > wifi 中的配置选项中的 WiFi IRAM speed optimization (N) 和 WiFi RX IRAM speed optimization (N) 配置选项禁用。
- 更多细节请参考 Minimizing RAM Usage。

ESP32 芯片低电压复位阈值是多少?

- 欠压复位电压阈值范围在 2.43V ~ 2.80 V 之间，可在 menuconfig -> Component config -> ESP32-specific -> Brownout voltage level 中进行设置。

4.9.22 ESP32 light sleep 例程为何会自动唤醒？

light sleep 例程下，默认使用了两种唤醒方式，如下：.. code-block:: c

esp_sleep_enable_timer_wakeup(2000000); // 2 秒自动唤醒 esp_sleep_enable_gpio_wakeup(); // GPIO 唤醒

GPIO 唤醒默认是 GPIO0 低有效唤醒，GPIO0 拉低则为唤醒状态，GPIO0 释放则自动进入 light sleep 模式。若需要长时间保存 light sleep 模式，可以将 2 秒自动唤醒屏蔽，仅开启 GPIO 唤醒。

4.9.23 ESP32 deep_sleep 例程测试，为何当 const int wakeup_time_sec = 3600 时，程序 crash 出现死循环？

- 程序 crash 原因是 int 类型参数 *wakeup_time_sec* 在 wakeup_time_sec * 1000000 在运算时溢出。

```
const uint64_t wakeup_time_sec = 3600;
printf("Enabling timer wakeup, %lldn",wakeuo_time_sec);
```

4.9.24 ESP32 有几种系统复位方式？

ESP32 有多种系统复位方式，包括以下几种：

- 软件复位（Software Reset）：在应用程序中可以通过调用 esp_restart() 函数来进行软件复位。
- 外部复位（External Reset）：通过外部硬件电路，例如按下 RESET 按键、供电电压不稳定等，触发 ESP32 的复位。
- 硬件看门狗复位（Hardware Watchdog Reset）：当 ESP32 在运行时发生死锁或其他异常情况时，硬件看门狗模块会自动触发复位。
- 欠压复位（Brownout Reset）：当系统电压不稳定或电源电压不足时，ESP32 内置的电源管理模块会自动触发复位。
- 异常复位（Exception Reset）：当 ESP32 在运行时发生 CPU 异常，例如访问非法内存、运行非法指令等，会触发异常复位。
- JTAG 复位（JTAG Reset）：当 ESP32 使用 JTAG 调试器进行调试时，可以通过 JTAG 复位信号进行复位操作。
- 更多说明参见 ESP32 技术规格书 4.1.2 复位源章节。

4.9.25 ESP8266-NONOS-V3.0 版本的 SDK，报错如下，是什么原因？

```
E:M 536
E:M 1528
```

- 导致出现 E:M 开头的 LOG 是由于剩余内存不足。

4.9.26 ESP32 是否可以完整使用 8 MB PSRAM 内存？

- ESP32 可完整使用 8 MB PSRAM 内存。
- 由于 cache 最大映射空间为 4 MB，所以仅支持 4 MB psram 映射使用，剩余空间可以使用 API 操作使用。
- 参考示例 himem。

4.9.27 ESP8266 AT 连接 AP 后，系统默认进入 modem-sleep，但电流未明显下降有哪些原因？

- AT 固件连接 AP 后，ESP8266 会进入自动 modem-sleep 模式，功耗大约会在 15mA ~ 70mA 之间波动。
- 如果功耗并没有在 15mA ~ 70mA 之间波动，在示波器中未呈现波形的电流，有以下建议：- 擦除设备 flash 后，重新烧录 AT 固件。- 抓取网络包分析，是否在当前的网络环境中，是否有频繁发送广播包的设备，可换一个网络环境的路由器（AP）进行测试。

4.9.28 ESP32 是否可以永久更改 MAC 地址？

- 芯片自带的 MAC 地址无法修改。efuse 中支持用户写入自己的 MAC 地址。
- 在固件中调用 api 可以获取定制 MAC 地址，并且可以设置到系统中替代默认地址。
- 配置参考：mac-address。
- 另外，Espressif 提供在芯片出厂之前，烧录用户提供的 MAC 地址服务。如有需要，可发送邮件至 sales@espressif.com

---

4.9.29 ESP8266 进行 ota 升级时如何校验 all.bin 为非法文件？

问题背景：

- all.bin: 由 bootloader.bin，partition.bin 和 app.bin 合并生成。
- ota.bin: 用于 ota 升级的目标 bin 文件。

使用 simple_ota_example 进行 OTA 升级时，误从服务器上下载 all.bin, 写入 ota 分区之后，设备会出现反复重启的现象。

原因分析：

代码中未对 all.bin 进行校验，导致将非法的 bin 文件写入 ota 分区。

解决方案：

通过打开 sha256 校验判断 all.bin 为非法 bin 文件，配置路径如下：Component config > App update > [*] Check APP binary data hash after downloading.

4.9.30 IDF 版本更新后，更新说明在哪里？

可以在 Github release note 查看相关说明。

4.9.31 ESP8266 是否有详细的寄存器手册？

请参考 8266 TRM appendix。

4.9.32 ESP32 开启 Secure Boot 后无法正常启动, 出现如下报错，是什么原因？

```
csum err:0x9a!=0x5f
ets_main.c 371
ets Jun  8 2016 00:22:57
rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4
load:0x3fff0034,len:9372
load:0x40078000,len:19636
```

可能是因为开启 Secure Boot 后 Bootloader 会变大，烧录固件时 Bin 文件产生了覆盖。可以查询 Secure Boot 后 Bootloader 的大小，比如可以尝试把分区表的偏移量增大为 0xF000 。

4.9.33 ESP8266 如何在设备软重启的情况下保留数据？

- 如果写入或者修改的次数不频繁，可以使用 Flash 来存储数据，该区域相对于内存较大，并且容易调整。
- 若数据较小，可以使用 RTC Memory 内存来存储相关数据。示例：Rel 2.1 的分支中 esp_system.h 中的接口（详细阅读使用说明）system_rtc_mem_read。
- 如果以上两者都无法满足需求，也可以选择外挂的 RTC 内存，可以使用 I2C 与 SPI 进行交互。
- 通常在写入频率不高的情况下建议写入 Flash ，因为该方法在硬断电时数据仍然正常。

4.9.34 ESP8266 有哪些定时器可用？

- ESP8266 有一个硬件定时器，可以产生中断，在 NONOS SDK 与 RTOS SDK 调用 API 略有不同。
- 软件定时器：- NONOS 中 API os_timer 是 DSR 处理，不能产生中断，但是可以产生任务，任务会按照普通等级排队。- RTOS 中可以使用 freertos 中的软件定时器，使用方式更加灵活。

4.9.35 ESP8266 的看🄵狗是什么作用？

- 为了提供系统稳定性，以应对多冲突的操作环境，ESP8266 集成了 2 级看🄵狗机制，包括软件看🄵狗和硬件看🄵狗。
- 默认 2 个看🄵狗都是打开的，HW WDT 始终在运行，并且如果未重置 HW WDT 计时器，则会在大约 6 秒钟后重置 MCU。
- SW WDT 大约在 1.5 秒左右将 MCU 复位。您可以启用/禁用 SW WDT，但不能启用/禁用 HW WDT。因为必须重置 SW WDT 后才能同时重置 HW WDT。
- 可通过修改 *make menuconfig -> Component config -> Common ESP-related* 里的 *Invoke panic handler on Task Watchdog timeout* 等来配置看门狗。

4.9.36 ESP8266 user_init 内有那些注意事项？

- wifi_set_ip_info、wifi_set_macaddr 仅在 user_init 中调用生效，其他地方调用不生效。
- system_timer_reinit 建议在 user_init 中调用，否则调用后，需要重新 arm 所有 timer。
- wifi_station_set_config 如果在 user_init 中调用，底层会自动连接对应路由，不需要再调用 wifi_station_connect 来进行连接。否则，需要调用 wifi_station_connect 进行连接。

- wifi_station_set_auto_connect 设置上电启动时是否自动连接已记录的路由；例如，关闭自动连接功能，如果在 user_init 中调用，则当前这次上电就不会自动连接路由，如果在其他位置调用，则下次上电启动不会自动连接路由。

4.9.37 ESP32 同时开启“Enable debug tracing of PM using GPIOs”和“Allow .bss segment placed in external memory”后为何会导致系统不停重启？

- “Enable debug tracing of PM using GPIOs”配置选项是在 GDB 调试时需要打开的，不可与“Allow .bss segment placed in external memory”配置选项同时使用。
- 因为“Enable debug tracing of PM using GPIOs”默认使用的是 GPIO16 与 GPIO17，与 PSRAM 接口（默认也是 GPIO16 和 GPIO17）冲突。

4.9.38 ESP32 IDF v3.3 版本 bootloader 运行 v3.1 版本 APP bin，程序为何会触发 RTCWDT_RTC_RESET ？

- 在 v3.3 的 bootloader 中会开启 WDT 看门狗，且在应用程序 (app) 运行时关闭 WDT 看门狗。
- 但 v3.1 的 bootloader 没有开启 WDT 看门狗，所以应用程序 (app) 没有 WDT 看门狗的机制，进而导致 v3.3 的 bootloader 引导 v3.1 的应用程序 (app) 会触发 WDT 看门狗复位。
- 可以通过在 menuconfig 中不使能 BOOTLOADER_WDT_ENABLE，关闭 v3.3 版本 bootloader 中 WDT 看门狗开启。

4.9.39 ESP32 芯片出厂是否有唯一的 chip_id ？

- ESP32 芯片未未烧录唯一 chip_id，但设备默认烧录有全球唯一 MAC 地址，可以读取 MAC 地址替代 chip_id。

4.9.40 ESP8266 rst cause 如何查看？

- 请参考 ESP8266 异常重启原因。

4.9.41 ESP32 编译生成的 bin 文件大小如何优化？

在 ESP32 编译生成的 bin 文件中，通常包括应用程序代码、分区表、ESP-IDF 固件和其他数据。为了优化 bin 文件的大小，可以采取以下几种方法：

- 配置编译选项：可配置 GCC 编译优化，操作步骤 `idf.py menuconfig` > `Compiler options` > `Optimization level (Optimize for size(-Os))`。
- 优化代码：可以对应用程序代码进行优化，例如采用更高效的算法和数据结构、精简代码逻辑和流程、提高代码复用率，调整 log 等级，减少不必要的 log 打印，减少代码文件大小。
- 需要注意的是，在进行 bin 文件大小优化时，需要权衡优化效果和程序功能，避免优化过度导致程序异常或功能不完整。建议在进行 bin 文件大小优化时参考官方文档和示例，并遵循相关规定和标准。

更多细节请参考 Minimizing Binary Size。

4.9.42 ESP32 是否有系统重新启动的 API ？

- 系统重新启动的 API 可使用 esp_restart()，相关说明可 参见 。

4.9.43 ESP32 异常 log *invalid header: 0xffffffff*

- ESP32 芯片打印该异常 log 通常有如下几种情况：
- 芯片上下电时序不正确，芯片部分区域未完全复位。
- Flash 中的固件出现异常，例如未烧录完整固件。
- Flash 器件损坏，无法读取正确数据。
- 芯片自身 cache 被关闭或者损坏，无法读取固件数据。

4.9.44 ESP8266 deep sleep 定时唤醒机制是什么？

- 在 Deep-sleep 状态下，将 GPIO16 (XPD_DCDC) 连接至 EXT_RSTB , 计时到达睡眠时间后，GPIO16 输出低电平给 EXT_RSTB 管脚，芯片被复位唤醒。

4.9.45 ESP32 使用 heap_caps_get_free_size 获取 RAM 约 300 KB，为何与手册 520 K 存在差异？

- 是因为内存在系统启动时预分配给各个功能模块使用，系统启动后剩余内存约 300 KB。
- 如果剩余内存不足，可以选用带 PSRAM 模组，将内存分配在 PSRAM 中。

4.9.46 ESP32 & ESP8266 如何通过局域网的 APP 进行 OTA 升级？

- 局域网内 APP 设备可以配置开启 http 服务，将提供的固件下载链接通过其他方法（udp，coap，mqtt 等）发送至设备。
- 设备通过传统 URL OTA 方法即可完成 OTA 更新，示例已在 SDK 中提供。

4.9.47 ESP32 如何修改日志输出串口使用的 GPIO？

- 配置 `menuconfig` > `Component Config` > `ESP System Settings` > `Channel for console output` > `Custom UART`，选择自定义 UART 管脚。
- 返回上一层，会看到出现 `UART TX on GPIO#` 和 `UART RX on GPIO#` 的选项，通过修改这两个选项可以更改日志输出串口使用的 GPIO。

4.9.48 ESP8266 使用 MQTT ssl_mutual_auth 通讯，在 OTA 时出现如下报错：

- 0x7f00 此报错是由于 内存不足 导致，建议使用 http 方式 OTA 。

4.9.49 ESP32 配置 menuconfig –> Component config 中有 NVS 选项，为何配置项目为空？

- menuconfig –> Component config 中的 NVS 选项是配置 NVS 加密功能的，该功能的前提是开启 Flash 加密。
- menuconfig –> security feaures –> enable flash encryption on boot 配置选项后，便可以看到 NVS 的配置选项。

4.9.50 ESP32 上电或 Deep-sleep 醒来后，会随机发生一次看🄵狗复位？

- 芯片上电的看🄵狗复位无法使用软件绕过，但复位后 ESP32 正常启动。
- Deep-sleep 醒来后的看🄵狗复位在 ESP-IDF V1.0 及更高版本中自动绕过。
- Deep-sleep 醒来后，CPU 可以立即执行 RTC fast memory 中的一段程序。RTC fast memory 中的这段程序通过清除 cache MMU 的非法访问标志从而绕过 Deep-sleep 醒来后的看🄵狗复位，具体为：
  - 将 DPORT_PRO_CACHE_CTRL1_REG 寄存器的 PRO_CACHE_MMU_IA_CLR 比特置 1。
  - 将该比特清零。

4.9.51 ESP32 CPU 使用 cache 访问外部 SRAM 时，如果这些操作需要 CPU 同时处理，可能会发生读写错误？

- 这个问题无法使用软件自动绕过。
- 对于版本 0 ESP32，CPU 使用 cache 访问外部 SRAM 时，只能够进行单向操作，即只能够单纯的进行写 SRAM 操作，或者单纯的进行读 SRAM 操作，不能交替操作。
- 使用 MEMW 指令：在读操作之后，加上 `__asm__("MEMW")` 指令，然后在 CPU 流水线被清空前再发起写操作。

4.9.52 ESP32 CPU 访问外设时，如果连续不间断地通过 DPORT 写同一个地址，为何会出现数据丢失的现象？

- 此问题在 ESP-IDF V1.0 及更高版本中自动绕过。
- 当连续写同一个地址（即类似 FIFO 的地址）时，使用 AHB 地址而不是 DPORT 地址。（对于其他类型的寄存器写入，使用 DPORT 地址可能写性能更好。）

寄存器名称	DPORT 地址	AHB（安全）地址
UART_FIFO_REG	0x3FF40000	0x60000000
UART1_FIFO_REG	0x3FF50000	0x60010000
UART2_FIFO_REG	0x3FF6E000	0x6002E000
I2S0_FIFO_RD_REG	0x3FF4F004	0x6000F004
I2S1_FIFO_RD_REG	0x3FF6D004	0x6002D004
GPIO_OUT_REG	0x3FF44004	0x60004004
GPIO_OUT_W1TC_REG	0x3FF4400c	0x6000400c
GPIO_OUT1_REG	0x3FF44010	0x60004010
GPIO_OUT1_W1TS_REG	0x3FF44014	0x60004014
GPIO_OUT1_W1TC_REG	0x3FF44018	0x60004018
GPIO_ENABLE_REG	0x3FF44020	0x60004020
GPIO_ENABLE_W1TS_REG	0x3FF44024	0x60004024
GPIO_ENABLE_W1TC_REG	0x3FF44028	0x60004028
GPIO_ENABLE1_REG	0x3FF4402c	0x6000402c
GPIO_ENABLE1_W1TS_REG	0x3FF44030	0x60004030

4.9.53 ESP32 CPU 频率从 240 MHz 直接切换到 80/160 MHz 会卡死，如何解决？

- 建议使用以下两种模式：

  (1) 2 MHz <-> 40 MHz <-> 80 MHz <-> 160 MHz

  (2) 2 MHz <->40 MHz <->240 MHz

- 此问题已在芯片版本 1 中修复。

4.9.54 ESP32 同时有 GPIO 和 RTC_GPIO 功能的 pad 的上拉下拉电阻只能由 RTC_GPIO 的上拉下拉寄存器控制，如何解决？

- ESP-IDF V2.1 及更高版本的 GPIO 驱动自动绕过此问题。
- GPIO 和 RTC_GPIO 都使用 RTC_GPIO 寄存器。

4.9.55 ESP32 由于 flash 启动的速度慢于芯片读取 flash 的速度，芯片上电或 Deep-sleep 醒来后，会随机发生一次看🄵狗复位，如何解决？

- 更换更快的 flash，要求 flash 上电到可读的时间小于 800 µs。这种方法可以绕过芯片上电和 Deep-sleep 醒来时的看🄵狗复位。
- Deep-sleep 醒来后的看🄵狗复位问题在 ESP-IDF V2.0 及更高版本中自动绕过（延迟时间可以根据需要配置）。具体方式是从 Deep-sleep 醒来后首先读取 RTC fast memory 中的指令，等待一段时间，然后再读取 flash。

4.9.56 ESP32 CPU 在访问外部 SRAM 时会小概率发生读写错误, 如何解决？

- x>=y 时，在 store.x 和 load.y 之间插入 4 个 nop 指令。
- x<y 时，在 store.x 和 load.y 之间插入 memw 指令。

4.9.57 ESP32 双核情况下，一个 CPU 的总线在读 A 地址空间，而另一个 CPU 的总线在读 B 地址空间，读 B 地址空间的 CPU 可能会发生错误如何解决？

- 一个 CPU 在读 A 地址空间时，通过加锁和中断的方式来避免另一个 CPU 发起对 B 地址空间的读操作。
- 一个 CPU 在读 A 地址空间之前，加一个此 CPU 读 B 地址空间（非 FIFO 地址空间，如 0x3FF40078）操作，并且要保证读 B 地址空间操作和读 A 地址空间操作是原子的。

4.9.58 ESP32 CPU 通过读取 INTERRUPT_REG 寄存器来复位 CAN 控制器的中断信号。如果在同一个 APB 时钟周期内 CAN 控制器刚好产生发送中断信号，则发送中断信号丢失，如何解决？

数据等待发送完成期间（即发送请求已发起），每一次读取 INTERRUPT_REG 后，用户都应检查 `STATUS_TRANSMIT_BUFFER` 位。如果 `STATUS_TRANSMIT_BUFFER` 置位而 `CAN_TRANSMIT_INT_ST` 没有置位，则说明发送中断信号丢失。在 ESP32 中，可以通过读取 INTERRUPT_REG 寄存器来复位 CAN 控制器的中断信号。但是如果在同一个 APB 时钟周期内 CAN 控制器产生发送中断信号，该中断信号可能会丢失，因为 ESP32 在该时钟周期内读取该寄存器时可能已经被清除。为了解决这个问题，可以使用以下方法：

- 添加延时：在读取 INTERRUPT_REG 寄存器之前，可以添加一定的延时，以确保 CAN 控制器的中断信号已经被清除。可以通过试验和调整来确定适当的延时时间。

- 使用中断处理程序：可以使用中断处理程序来处理 CAN 控制器的中断信号，并避免在同一个 APB 时钟周期内读取 INTERRUPT_REG 寄存器。中断处理程序可以及时响应 CAN 控制器的中断信号，保证信号不会丢失。
- 使用其他寄存器：可以使用其他寄存器来复位 CAN 控制器的中断信号，以避免在同一个 APB 时钟周期内读取 INTERRUPT_REG 寄存器。例如，可以使用 CANCTRL 寄存器或 ERRCNT 寄存器等。

需要注意的是，在使用以上方法时，需要根据具体应用场景和需求来选择和实现。同时，也需要对软件和硬件进行充分的测试和验证，以确保系统的可靠性和稳定性。在 ESP32 中复位 CAN 控制器的中断信号时，需要注意避免中断信号丢失的问题，以保证系统的正常运行。

4.9.59 ESP32 v3.0 芯片，当程序同时满足下列条件时，会出现 live lock（活锁）现象，导致 CPU 一直处于访存状态，不能继续执行指令，请问如何解决？

发生 live lock 时，软件主动或被动识别并解锁 cache line 竞争，之后两个核按队列节拍分时完成各自的 cache 操作，解锁 live lock。详细过程如下：

- 当两个核执行的指令均不在代码临界区中时出现 live lock，系统各类型中断会主动解除 cache line 竞争，解锁 live lock；
- 当两个核执行的指令位于代码临界区中时出现 live lock，在临界区中，系统会屏蔽 3 级及以下中断。因此软件预先为两个核各设置一个高优先级（4 级或 5 级）中断，将它们绑定到同一个定时器，并选择合适的定时器超时阈值。由于 live lock 产生的定时器超时中断会迫使两个核均进入高优先级中断处理程序，从而释放两个核的 IBUS 以达到解锁 live lock 目的。解锁过程通过 3 个阶段完成：
  a. 第 1 阶段两个核均进行等待以清空 CPU write buffer；
  b. 第 2 阶段一个核 (Core 0) 等待，另一个核 (Core 1) 执行；
  c. 第 3 阶段 Core 1 等待，Core 0 执行。

4.9.60 ESP32 CPU 访问 0x3FF0_0000 ~ 0x3FF1_EFFF 与 0x3FF4_0000 ~ 0x3FF7_FFFF 两段地址空间存在限制，如何解决？

- 落在这两段地址空间的 CPU 访问均需要在对应的操作前加入 “MEMW” 指令。即在 C/C++ 中，软件访问这两段地址内的寄存器时需要加上 “volatile” 属性。

4.9.61 ESP32 如何关闭程序 LOG 输出？

- 关闭 bootloader 日志：`menuconfig > bootloader config > bootloader log verbosity` 选定为 `No output`。
- 关闭程序日志：`menuconfig > Component config > log output > Default log verbosity` 选定为 `No output`。
- 在 ESP-IDF release/v4.3 及之前的版本中关闭 UART0 输出日志：`menuconfig > Component Config > Common ESP-related > Channel for console output > None`。
- 在 ESP-IDF release/v4.4 及之后的版本中关闭 UART0 输出日志：`Component config > ESP System Settings > Channel for console output > None`。

4.9.62 ESP8266 在 Deep sleep 模式下，保存在 RTC Memory 里的数据是否可运行？

- ESP8266 在 Deep sleep 模式下只有 RTC 定时器继续工作，保存在 RTC Memory 里的数据在 Deep sleep 模式下不会运行，只能保持数据不会丢失。但是，当 ESP8266 掉电后，保存在 RTC memory 里的数据无法保存。

4.9.63 ESP32 的 NVS 的 Key 的最大长度为多大？

- ESP32 的 NVS 的 Key 最大长度为 15 个字符，且无法更改 Key 的最大长度。可参见 键值对 说明。
- 可使用 nvs_set_str() 的 value 来存数据。

4.9.64 ESP-IDF release/v4.2 里的 cJSON 支持 uint64_t 的数据解析吗？

- 不支持。cJSON 库解析长整形有限制，最长只有 Double 类型。

4.9.65 未启用 Flash 加密的 ESP32 可以进行 GDB 调试，但启动 Flash 加密后进行 GDB 调试时，设备一直重启，是什么原因？

- 启用 Flash 加密或安全启动 (secure boot) 后，将默认禁用 JTAG 调试功能，更多信息请参考 JTAG 与闪存加密和安全引导。

4.9.66 ESP32 使用手机热点进行 OTA 固件下载时，关闭流量开关几秒后再次打开会出现程序一直卡死在 OTA 里的情况（使用路由器时插拔 wan 口网线同理），是什么原因？

- 这是协议的正常现象。如果使用 *esp_https_ota* 组件进行 OTA，可以设置网络超时时间 *http_config->timeout_ms* 为 10 ~ 30 秒（不建议太小），使能 *http_config->keep_alive_enable* 来检测链路是否异常。
- 对于用户自行实现的 OTA 模块，按照上述思路，通过 *select* 机制添加读取超时或者使能 TCP Keep-alive 链路检测机制。

4.9.67 ESP32-C3 在 Deep-Sleep 模式下可以通过哪些 GPIO 进行唤醒？

- ESP32-C3 仅有 VDD3P3_RTC 域中的管脚 (GPIO0 ~ GPIO5) 可用于将芯片从 Deep-sleep 唤醒。请阅读 '《ESP32-C3 技术参考手册》<https://www.espressif.com/sites/default/files/documentation/esp32-c3_technical_reference_manual_cn.pdf>'_ 中 "5.9.1 GPIO 管脚供电" 章节的说明。

4.9.68 使用 ESP-WROOM-02D 模组，电池供电，在低电量（模组勉强启动）的时候，频繁格式化读写 flash 有什么风险吗？

在低电量情况下频繁进行格式化和读写 flash 存储器可能会有一些风险。它在低电量情况下可能无法正常工作或容易发生错误，如果在这种情况下频繁进行格式化和读写 flash 存储器，可能会导致以下风险：

- 数据丢失或损坏：在低电量情况下，flash 存储器可能无法正常写入数据。如果频繁进行格式化和读写操作，可能会导致数据丢失或损坏。
- 模组崩溃或损坏：低电量情况下频繁进行格式化和读写 flash 存储器会消耗模组的电量，可能会导致模组崩溃或损坏。

因此，建议在低电量情况下尽量减少对 flash 存储器的访问和操作，避免频繁进行格式化和读写操作。如果需要进行格式化和读写操作，应确保模组有足够的电量，并在操作前先备份数据以防止数据丢失。此外，建议使用低功耗模式和优化代码以尽可能减少电量消耗。

4.9.69 ESP32 如何查看线程使用过的最大栈大小？

- 可以调用 UBaseType_t uxTaskGetStackHighWaterMark(TaskHandle_t xTask) 函数来查看。该函数可以返回任务启动后的最小剩余堆栈空间。

4.9.70 使用 ESP32 时打印“SW_CPU_RESET”日志是什么原因？

在 ESP32 上，打印出“SW_CPU_RESET”日志通常是由于程序异常终止导致的。ESP32 内置了两个处理器内核，即主核和辅助核。在某些情况下，如果程序在主核上执行，并且出现了一些异常情况，例如访问非法地址或发生未处理的中断，可能会导致主核进入异常状态并重新启动。当这种情况发生时，ESP32 会在串行终端（UART）上打印“SW_CPU_RESET”日志。此外，使用 ESP-IDF 开发应用程序时，也可能会在应用程序中调用 esp_restart() 函数来重新启动 ESP32。在这种情况下，ESP32 也会在串行终端上打印“SW_CPU_RESET”日志。需要注意的是，出现“SW_CPU_RESET”日志并不一定意味着程序有问题或 ESP32 硬件有故障。它可能只是由于某些异常情况导致的正常现象。但是，如果程序频繁出现异常并重新启动，需要进行调试和排除问题。可以通过检查程序日志和硬件设备状态来确定问题的原因。

4.9.71 使用 ESP32 时，单独测试 NVS 发现占用内存很大，是什么原因？

- 请检查分区表设置，建议将分区表中的 NVS 数据分区设置小一些来测试，NVS 数据分区设置越大占用内存越多。

4.9.72 如何修改模块的系统时间?

CHIP: ESP32 | ESP32 | ESP32-C3

- 可以使用 c 语言 `time()` 接口来设置系统时间。

4.9.73 OTA 升级过程中 esp_ota_end 返回 ESP_ERR_OTA_VALIDATE_FAILED 报错，如何排查这类问题?

CHIP: ESP32

- 一般是由于下载的固件内容有误导致的，可以通过 esptool 中的 read_flash 指令 dump 出模组中的内容，然后再用 Beyond Compare 工具对这 2 个 bin 文件进行 16 进制对比，看 bin 文件哪里下载有误。

4.9.74 ESP8266-RTOS-SDK 如何将数据存储在 RTC memory 中？

- 将数据存储在 RTC memory 中的定义方式如下：
- 可参见 esp_attr.h 文件说明。

4.9.75 在 Deep-sleep 模式唤醒后，ESP8266 是从哪里启动的？

- ESP8266 在 Deepsleep 模式唤醒后，设备将从 user_init 启动。请参见 ‘ esp_deep_sleep() <https://docs.espressif.com/projects/esp8266-rtos-sdk/en/latest/api-reference/system/sleep_modes.html?highlight=deep#_CPPv414esp_deep_sleep8uint64_t>‘__ 说明。

4.9.76 RTC 时钟什么时候会被重置？

- 除上电复位外的任何睡眠或者复位方式都不会重置 RTC 时钟。

4.9.77 ESP32 使用 AT+GSLP 指令进入 Deep-sleep 模式后，是否可通过拉低 EN 进行唤醒？

- 使用 AT+GSLP 指令进入 Deep-sleep 模式后，可以通过拉低 EN 唤醒，但不推荐此做法。
- Deep-sleep 模式可通过 RTC_GPIO 来唤醒。请参见 《ESP32 技术参考手册》 。

4.9.78 当多个线程要使用 ESP32 的看门狗时，是否每个线程都要开启看门狗？

- 当多个线程要使用看门狗时，每个线程都要开启看门狗。可参见 任务看门狗说明。

4.9.79 使用 ESP8266-RTOS-SDK release/v3.3，如何进入 Light-sleep 模式？

- 先设置 Light-sleep 模式的唤醒模式，可参考 ESP8266_RTOS_SDK/components/esp8266/include/esp_sleep.h。
- 然后使用 esp_light_sleep_start() API 进入 Light-sleep 模式。
- 程序实现逻辑可以参考 esp-idf/examples/system/light_sleep/main/light_sleep_example_main.c 例程。
- ESP8266-RTOS-SDK 关于 Sleep 模式的 API 说明请阅读 Sleep modes API Reference。

4.9.80 ESP8266 在 Deep sleep 模式下如何唤醒？

- ESP8266 在 Deep sleep 模式下只能通过 RTC Timer 进行唤醒，定时时长为用户通过函数 esp_deep_sleep 设置的时间，且硬件上需要把 GPIO16 (XPD_DCDC) 通过 0 欧姆电阻连接到 EXT_RSTB，以支持 Deep Sleep 模式唤醒。请参见 相应 API 唤醒说明。

4.9.81 使用 ESP32-WROVER 模组，休眠时存在电池抖动或异常掉电上电导致死机无法唤醒的问题，是什么原因？

- 应用场景：休眠的时候电流大概是 12 uA, 当拔电池或震动摇晃产品的时候会造成掉电，但是电容里还有电，ESP32 从 3.3 V 放电到 0 V 的过程中，再上电恢复 3.3 V 会导致 ESP32 无法唤醒。
- 请检查芯片 VCC 与 EN 是否满足上电时序要求。
- 在使用 ESP32-WROVER 模组进行休眠时，如果存在电源电压不稳定或异常掉电的情况，可能会导致芯片的电源管理单元出现问题，导致无法正常唤醒。
- 可以考虑添加复位芯片保证时序正常。
- ESP32 上电、复位时序说明，详见 《ESP32 技术规格书》 。

4.9.82 如何烧录自定义 mac 地址？

- 可以先了解 ESP 模块 mac 的机制，请参考 Mac 地址介绍。目前烧录自定义 mac 地址有 2 种方案：
  - 方案 1：直接烧到 efuse blk3 中，可以保证不被修改；
  - 方案 2：存储到 flash 中。不推荐将 mac 地址存放在默认 nvs 分区中，建议创建一块自定义的 nvs 分区用来存储自定义的 Mac 地址。关于自定义 mac 地址的使用，可以参考 base_mac_address。

4.9.83 ESP32 在使用 esp_timer 时，出现网络通信或者蓝牙通信异常，是什么原因？

- esp_timer 是高精度的硬件定时器组件，后台一些组件也使用 esp_timer 完成一些系统任务。在使用 esp_timer 时，请不要在该定时器的回调函数中使用延时、阻塞类的 API，应尽可能地保证回调函数能够快速地被执行结束，以免影响系统其他组件的功能。
- 如您需要的定时精度不是太高，请使用 FreeRTOS 中的定时器组件 xTimer。

4.9.84 使用 ESP32，请问 ULP 里面用 jump 跳转到一个函数，是否有返回的指令？

目前 ULP CPU 指令列表以及说明参见 这里。返回指令通常使用一个通用寄存器备份 PC 地址，用于后续跳回，由于目前 ULP 只有 4 个通用寄存器，所以需要合理使用。

4.9.85 如何调整编译的警告级别？

编译工程时，发现一些警告被视为错误，导致编译失败，如下：

```
error: format '%d' expects argument of type 'int *', but argument 3 has type
↪'uint32_t *' {aka 'long unsigned int *'} [-Werror=format=]
```

针对于上述错误，用户可以在组件级别（在组件 CMakeLists.txt 中）或项目级别（在项目 CMakeLists.txt 中）修改编译标志，这两种方式的效果大致相同。

- 要修改特定组件的编译标志，请使用标准 CMake 函数 `target_compile_options`。请参考 组件编译控制。组件级别的 `target_compile_options` 示例请见 CMakeLists.txt#L3。
- 要修改整个项目的编译标志，请使用标准 CMake 函数 `add_compile_options` 或 IDF 特定函数 `idf_build_set_property` 来设置 `COMPILE_OPTIONS` 属性。请参考 覆盖默认的构建规范。

4.9.86 基于 ESP-IDF SDK 编译固件时，会包含 IDF_PATH 的信息和存储编译时间，导致编译的 bin 不一样。如何删除这些信息？

- 如果是 v5.0 及以上版本的 SDK，可以开启 `CONFIG_APP_REPRODUCIBLE_BUILD` 配置选项，开启后，使用 ESP-IDF 构建的应用程序不依赖于构建环境。应用程序的.elf 文件和.bin 文件都保持完全相同，即使以下变量发生变化：
  - 项目所在目录
  - ESP-IDF 所在目录（IDF_PATH）
  - 构建时间

  详情参见 Reproducible Builds 说明。
- 如果是 v5.0 以下版本的 SDK，可以关闭 `CONFIG_APP_COMPILE_TIME_DATE=n` 配置，来删除编译时间戳信息，并且开启 `COMPILER_HIDE_PATHS_MACROS=y` 配置来隐藏 IDF_PATH。

4.9.87 使用 ESP32-S3-DevKitM-1 开发板下载官方 hello_world 例程后出现如下报错，是什么原因？

```
ESP-ROM:esp32s3-20210327
Build:Mar 27 2021
rst:0x7 (TG0WDT_SYS_RST),boot:0x8 (SPI_FAST_FLASH_BOOT)
Saved PC:0x40043ac8
Invalid chip id. Expected 9 read 4. Bootloader for wrong chip?
ets_main.c 329
```

- 当前报错可能与开发板上的芯片版本或 esp-idf SDK 的软件版本不是正式量产版本有关。芯片 (ROM) 引导加载程序预期芯片 ID 为 9，这是芯片的量产版本（不是测试版本）。然而，在二级引导加载程序标头中，它看到了芯片 ID 为 4，这是 beta 版本的芯片。可参考 '这里 <https://github.com/espressif/esp-idf/issues/7960>'_ 的说明：
- 可以通过 esptool.py chip_id 命令来查询芯片的实际版本。如果芯片版本是量产版本，那么该报错与所使用的 esp-idf SDK 版本有关。ESP32-S3 系列的产品请使用 esp-idf release/v4.4 及以上版本的软件环境。

4.9.88 请问 ESP32 芯片的内部 150 kHz 的 RTC 时钟精度是多大？

- ESP32 芯片内部 150 kHz 的 RTC 时钟精度为 ±5%。

4.9.89 ESP32-D0WDR2-V3 芯片支持的 esp-idf SDK 的版本有哪些？

- 支持 IDF 版本是：v4.4.1、v4.3.3、v4.2.3、v4.1.3。

4.9.90 基于 ESP32 芯片测试 OTA 应用，是否可以删除分区表里默认存在的 factory 分区，将 OTA_0 分区的地址设置为 0x10000?

- 可以省略 factory 分区并将 OTA_0 分区的地址设置为 0x10000，需要注意任何 app 类型的分区的偏移地址必须要与 0x10000 (64K) 对齐。

4.9.91 为什么使用 espefuse.py burn_key 命令无法烧录 ESP32-C3 eFuse 的 BLOCK3？

- espefuse.py burn_key 命令只能向类型为 KEY_DATA 的 eFuse 块烧录数据，但是默认情况下 ESP32-C3 的 BLOCK3 为 USR_DATA 类型。
- 可以通过 espefuse.py burn_block_data 命令向 USR_DATA 类型的 eFuse 块烧录数据。

4.9.92 基于 esp-idf SDK 运行固件后打印如下报错是什么原因？

```
***ERROR*** A stack overflow in task sys_evt has been detected.
```

- 当前报错是由于 system_event 任务堆栈不足导致的，可尝试增大 Component config > ESP System Setting > Event loop task stack size 设置来进行测试。不过出现溢出是在 system_event 事件中处理了太多的逻辑，这种行为本身是不提倡的，可能会导致后序事件无法及时抛出来。我们建议是通过队列或者其他操作将这个事件抛给其他任务处理。

4.10 Wi-Fi

[English]

4.10.1 ESP32 和 ESP8266 是否支持中文 SSID？

ESP32 和 ESP8266 均支持中文 SSID，但需要使用相应的库和设置。需要注意的是，由于中文字符占用的字节数不同，因此使用中文 SSID 时需要特殊处理。

对于 ESP32，可以使用 ESP-IDF 提供的 Wi-Fi 相关 API。在连接 AP 时，可以使用 esp_wifi_set_config() 函数设置 Wi-Fi 配置，其中的 ssid 参数可以设置为中文字符串。例如：

```
wifi_config_t wifi_config = {
  .sta = {
      .ssid = "你好，世界",
      .password = "password123",
  },
};
ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_STA, &wifi_config));
```

4.10.2 [Scan] ESP32 扫描一次需要花多长时间？

扫描花费的总时间取决于:

- 是被动扫描还是主动扫描，默认为主动扫描。
- 每个信道停留的时间，默认主动扫描为 120 ms，被动扫描为 360 ms。
- 国家码与配置的信道范围，默认为 1~13 信道。
- 是快速扫描还是全信道扫描，默认为快速扫描。
- Station 模式还是 Station-AP 模式，当前是否有连接。
- 默认情况下，1~11 信道为主动扫描，12🄵13 信道为被动扫描。
  - 在 Station 模式没有连接的情况下，全信道扫描总时间为：11*120 + 2*360 = 2040 ms；
  - 在 Station 模式有连接，或者 Station-AP 模式下，全信道扫描总时间为：11*120 + 2*360 + 13*30 = 2430 ms。

4.10.3 [Scan] 乐鑫是否支持 boundary scans(边界扫描)？

ESP32 不支持 boundary scan。

4.10.4 Wi-Fi 信道是什么？可以自行选择信道吗？

信道指的是 Wi-Fi 使用的指定频段中特定频率的波段。不同国家地区使用的信道数是不同的。例如在北美，Wi-Fi 信道范围是 1 到 11，而在欧洲，Wi-Fi 信道范围是 1 到 13。用户可以参考 ESP8266 Wi-Fi 信道选择指南。

4.10.5 [LWIP] 使用 ESP-IDF v4.1，ESP32 用作 SoftAP 模式时如何设置 IP 地址?

由于 ESP-IDF v4.1 以及以上版本会摒弃掉 TCP/IP 的接口，推荐使用 ESP-NETIF 的接口.

参考示例代码如下:

```
{
    ...
    esp_netif_t *ap_netif = esp_netif_create_default_wifi_ap();
    char* ip= "192.168.5.241";
    char* gateway = "192.168.5.1";
    char* netmask = "255.255.255.0";
```

(下页继续)

(续上页)

```
    esp_netif_ip_info_t info_t;
    memset(&info_t, 0, sizeof(esp_netif_ip_info_t));

    if (ap_netif)
    {
        ESP_ERROR_CHECK(esp_netif_dhcps_stop(ap_netif));
        info_t.ip.addr = esp_ip4addr_aton((const char *)ip);
        info_t.netmask.addr = esp_ip4addr_aton((const char *)netmask);
        info_t.gw.addr = esp_ip4addr_aton((const char *)gateway);
        esp_netif_set_ip_info(ap_netif, &info_t);
        ESP_ERROR_CHECK(esp_netif_dhcps_start(ap_netif));
    }
    ...
}
```

4.10.6 [LWIP] ESP32 Station 模式，如何设置静态 IP？

由于 v4.2 以及以上版本会摒弃掉 TCP/IP 的接口，推荐使用 ESP-NETIF 的接口. 参考示例代码如下：

```
esp_netif_t *sta_netif = esp_netif_create_default_wifi_sta();
if (sta_netif)
{
    esp_netif_ip_info_t info_t = {0};
    esp_netif_dhcpc_stop(sta_netif);

    info_t.ip.addr = ESP_IP4TOADDR(192, 168, 3, 23);
    info_t.gw.addr = ESP_IP4TOADDR(192, 168, 3, 1);
    info_t.netmask.addr = ESP_IP4TOADDR(255, 255, 255, 0);
    esp_netif_set_ip_info(sta_netif, &info_t);
}
esp_netif_dns_info_t dns_info = {0};
```

4.10.7 [LWIP] ESP-IDF 里如何设置 DHCP Server 的 Option 内容？

由于 v4.1 以及以上版本会摒弃掉 tcp/ip 的接口，推荐使用 ESP-NETIF 的接口。DHCP Client 设置方法也可以参考本示例。参考示例代码如下：

```
// 创建 softap 的 netif 句柄
esp_netif_t *ap_netif = esp_netif_create_default_wifi_ap();

// ESP_NETIF_IP_ADDRESS_LEASE_TIME, DHCP Option 51, 设置分发的 IP 地址有效时间
uint32_t dhcps_lease_time = 60; // 单位是分钟
ESP_ERROR_CHECK(esp_netif_dhcps_option(ap_netif,ESP_NETIF_OP_SET,ESP_NETIF_
↪IP_ADDRESS_LEASE_TIME,&dhcps_lease_time,sizeof(dhcps_lease_time)));

// ESP_NETIF_DOMAIN_NAME_SERVER , DHCP Option 6, 设置 DNS SERVER
// 设置 DNS 之前先要设置本地主 DNS
esp_netif_dns_info_t dns_info = {0};
dns_info.ip.u_addr.ip4.addr = ESP_IP4TOADDR(8,8,8,8);
ESP_ERROR_CHECK(esp_netif_set_dns_info(ap_netif,ESP_NETIF_DNS_MAIN,&dns_
↪info));

uint8_t dns_offer = 1; // 传入 1 使修改的 DNS 生效，如果是 0，那么用 softap 的 gw␣
↪ip 作为 DNS server (默认是 0)
ESP_ERROR_CHECK(esp_netif_dhcps_option(ap_netif,ESP_NETIF_OP_SET,ESP_NETIF_
↪DOMAIN_NAME_SERVER,&dns_offer,sizeof(dns_offer)));

// ESP_NETIF_ROUTER_SOLICITATION_ADDRESS, DHCP Option 3 Router, 传入 0 使␣
↪DHCP Option 3(Router) 不出现 (默认为 1)
uint8_t router_enable = 0;
ESP_ERROR_CHECK(esp_netif_dhcps_option(ap_netif,ESP_NETIF_OP_SET,ESP_NETIF_
↪ROUTER_SOLICITATION_ADDRESS,&router_enable, sizeof(router_enable)));

// ESP_NETIF_SUBNET_MASK, DHCP Option 1, 设置子网掩码
// 通过 ESP_NETIF_SUBNET_MASK 设置子网掩码无效， 请通过 esp_netif_set_ip_info 修改
```

4.10.8 [Performance] 如何测试 Wi-Fi 模组的通信速率？

可以使用 ESP-IDF 里的 iperf 示例进行测试。

4.10.9 [LWIP] ESP8266 SoftAP 默认使用哪个网段？

ESP8266 SoftAP + Station 模式下, 连接的 192.168.4.X 网段时，为什么会失败?

- ESP8266 SoftAP 默认使用网段 192.168.4.*，IP 地址是 192.168.4.1。ESP8266 如果要连接 192.168.4.X 的路由时，不能分辨是要连接自己本身的 SoftAp 还是外部路由，所以会造成错误。

4.10.10 [Connect] ESP8266 SoftAP 模式支持几个设备？

ESP8266 SoftAP 模式最多可以支持八个设备连接。这是由于 ESP8266 芯片在 SoftAP 模式下使用的 NAT（网络地址转换）机制只支持最多八个设备的连接。但需要注意的是，每个连接的设备会占用一定的带宽和资源，因此我们推荐连接四个设备，因为连接过多设备可能会影响 Wi-Fi 模组的性能和稳定性。

4.10.11 ESP8266/ESP32/ESP32-S2/S3/C2/C3 是否支持 web/softAP 配网？

支持。

- ESP8266 请参考此示例 ESP8266 softap_prov;
- ESP32/ESP32-S2/S3/C2/C3 请参考此示例 ESP32/ESP32-S2/S3/C2/C3 wifi_prov_mgr。

4.10.12 [Connect] ESP8266 和 ESP32 作为 softap 模式如何隐藏 SSID？

要隐藏 ESP8266 或 ESP32 作为 SoftAP 模式下的 SSID，可以通过以下方法实现：

- 调用 esp_wifi_set_config() 来配置 SoftAP 模式下的 SSID，密码以及是否隐藏。例如，以下代码设置 SSID 为 “MySoftAP”，密码为 “MyPassword”，函数中使用.ssid_hidden = 1 选项来隐藏 SSID：

```
wifi_config_t config = {
  .ap = {
    .ssid = "MySoftAP",
    .ssid_len = strlen("MySoftAP"),
    .password = "MyPassword",
    .max_connection = 4,
    .authmode = WIFI_AUTH_WPA_WPA2_PSK
    .ssid_hidden = 1
  },
```

(下页继续)

(续上页)

```
};
esp_wifi_set_config(WIFI_IF_AP, &config);
```

配置完后调用 esp_wifi_start() 启动 Wi-Fi。

4.10.13 esp_wifi_802.11_tx 接口中的 buffer 参数中包括 FCS 吗？

不包括，FCS 帧是硬件自动生成的。

4.10.14 ESP-WROOM-32D 支持的 Wi-Fi 频段信息和功率表分别是什么？

Wi-Fi 频段是 2412 ~ 2484 MHz，软件里可配置可用信道和对应的工作频率。功率表有默认值，也可支持软件配置。详细指导请参考 《ESP32 Phy Init Bin 重要参数配置说明》 。

4.10.15 ESP32 Wi-Fi RF 功率最高值是多少？

ESP32 的 Wi-Fi RF（无线电频率）功率输出最高可以配置为 20 dBm。请注意，最大功率输出水平可能会因不同的国家/地区和规定而有所不同。在使用 ESP32 时，请确保您遵守当地的规定和法规，以确保合法和安全使用。另外，高功率输出也会对电池寿命和 Wi-Fi 信号稳定性产生影响，因此在选择功率输出水平时，需要根据具体的应用场景和要求进行权衡和选择。

4.10.16 ESP32 如何调整 Wi-Fi 的发射功率？

- 可通过 menuconfig 配置 `Component config > PHY > Max Wi-Fi TX power(dBm)` 来调整 Wi-Fi 的发射功率，最大是 20 dBm。
- 或者使用 API *esp_err_t esp_wifi_set_max_tx_power(int8_t power);* 设置调整。

4.10.17 [Connect] ESP32 AP 模式最多支持多少设备连接？

ESP32 AP 模式，最多可配置为支持 10 个设备连接，默认配置为支持 4 设备。

4.10.18 [Connect] Wi-Fi 模组如何通过 RSSI 数值划分信号强度等级？

我们没有对 RSSI 信号强度进行等级划分。如果您需要标准进行划分，可以参考安卓系统的计算方法。

```
@UnsupportedAppUsage
private static final int MIN_RSSI = -100;

/** Anything better than or equal to this will show the max bars. */
@UnsupportedAppUsage
private static final int MAX_RSSI = -55;

public static int calculateSignalLevel(int rssi, int numLevels) {
  if(rssi <= MIN_RSSI) {
    return 0;
  } else if (rssi >= MAX_RSSI) {
    return numLevels - 1;
  } else {
    float inputRange = (MAX_RSSI -MIN_RSSI);
    float outputRange = (numLevels - 1);
    return (int)((float)(rssi - MIN_RSSI) * outputRange / inputRange);
  }
}
```

4.10.19 [Connect] ESP32 做 soft-AP 时为什么会把 STA 踢掉？

- 默认情况下连续 5 min 收不到 STA 发过来的数据包就会把 STA 踢掉。该时间可以通过 esp_wifi_set_inactive_time 进行修改。
- 注: esp_wifi_set_inactive_time 新增的 API。
  - master commit: `63b566eb27da187c13f9b6ef707ab3315da24c9d`
  - 4.2 commit: `d0dae5426380f771b0e192d8ccb051ce5308485e`
  - 4.1 commit: `445635fe45b7205497ad81289c5a808156a43539`
  - 4.0 commit: `0a8abf6ffececa37538f7293063dc0b50c72082a`
  - 3.3 commit: `908938bc3cd917edec2ed37a709a153182d511da`

4.10.20 [Connect] ESP32 进行 Wi-Fi 连接时，如何通过错误码判断失败原因？

ESP-IDF v4.0 及以上版本可参考如下代码获取 Wi-Fi 连接失败的原因：

```
if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_DISCONNECTED) {
  wifi_event_sta_disconnected_t *sta_disconnect_evt = (wifi_event_sta_
↪disconnected_t*)event_data;
  ESP_LOGI(TAG, "wifi disconnect reason:%d", sta_disconnect_evt->reason);
  esp_wifi_connect();
  xEventGroupClearBits(s_wifi_event_group, CONNECTED_BIT);
}
```

当回调函数接收到 WIFI_EVENT_STA_DISCONNECTED 事件时，可以通过结构体 wifi_event_sta_disconnected_t 的变量 reason 获取到失败原因。

- WIFI_REASON_AUTH_EXPIRE 在连接的 auth 阶段，STA 发送了 auth，但在规定时间内未收到 AP 的 auth 回复，有较低概率会出现。
- WIFI_REASON_AUTH_LEAVE 通常是由 AP 因为某种原因断开了 STA 连接，reason code 是由 AP 发过来的。
- WIFI_REASON_4WAY_HANDSHAKE_TIMEOUT 或者 WIFI_REASON_HANDSHAKE_TIMEOUT 失败原因为密码错误。

  其中，WIFI_REASON_4WAY_HANDSHAKE_TIMEOUT 为标准通用的错误码，而 WIFI_REASON_HANDSHAKE_TIMEOUT 为自定义错误码。两者区别在于 WIFI_REASON_4WAY_HANDSHAKE_TIMEOUT 为路由器在密码错误时告知设备，产生的错误，WIFI_REASON_HANDSHAKE_TIMEOUT 为路由器在密码错误时不告知设备，由设备本身超时机制产生的错误。
- WIFI_REASON_CONNECTION_FAIL 扫描阶段返回的错误码，主要是由于 STA 扫描到了匹配的 AP，但是这个 AP 在黑名单里。AP 在黑名单里面的原因是上次 AP 主动踢掉了 STA，或者 STA 连接 AP 的过程中失败了。

4.10.21 ESP32 系列芯片每次连接服务器都会执行域名解析吗？

在协议栈内，域名会通过 DNS 进行解析，解析后的数据会在时效内进行缓存。缓存时间基于从 DNS 服务器获取的 TTL 数据，该数据是配置域名时填入的参数，通常为 10 分钟。

4.10.22 [Connect] Wi-Fi Log 中状态机切换后面数字的含义？

eg: run -> init (fc0)，fc0 含义为 STA 收到了 deauth 帧，reason 为密码错误。

- c0 代表收到的帧类型（00 代表超时）
- f 代表 reason

帧类型: [a0 disassoc]、[b0 auth]、[c0 deauth]。

4.10.23 [Connect] bcn_timeout, ap_probe_send_start 是什么意思？

在规定时间内（ESP32 默认 6 s，即 60 个 Beacon Interval），STA 未收到 Beacon 帧。造成该现象可能有:

- 内存不足。"ESP32_WIFI_MGMT_SBUF_NUM" 不够 (log 中会打出 "esf_buf: t=8, l=beacon_len, …" 这样的 Error)。内存不够，可在收到 disconnect event 时打出 heap 大小来排查。
- AP 未发出 beacon。可通过抓包 AP 的 beacon 来排查。
- Rssi 值太低。在复杂环境下 Rssi 值较低时，可能导致 STA 收不到 beacon，可通过调用 `esp_wifi_sta_get_ap_info` 获取 Rssi 值来排查。
- 硬件原因。收包性能差。

出现 bcn_timeout 时，STA 会尝试发送 5 次 Probe Request，如果 AP 回 Probe Reponse，就保持连接；如果 AP 未回复，STA 发送 Disconnect 事件，并断开连接。

4.10.24 [Connect] Wi-Fi 连接断开后如何重连？

收到 `WIFI_EVENT_STA_DISCONNECTED` 之后调用 esp_wifi_connect。

4.10.25 [Connect] ESP32 作为 station 时什么时候会把 SoftAP 踢掉？

默认情况下 6 s 未收到 AP 的 beacon 就会把 AP 踢掉。该时间可以通过 esp_wifi_set_inactive_time 进行修改。

4.10.26 [Scan] 为什么有时候扫描不到 AP？

ESP32 和 ESP8266 扫描不到 AP 的原因可能有很多，以下是一些常见的原因和解决方法：

- AP 距离过远或信号质量差：ESP32 和 ESP8266 的 Wi-Fi 功能只能在一定范围内工作。如果 AP 距离过远或 Wi-Fi 信号质量太差，ESP32 和 ESP8266 可能无法扫描到 AP。可以尝试将 ESP32 或 ESP8266 靠近 AP，或者使用信号增强器来增强 AP 信号强度。
- AP 的 SSID 隐藏：一些 AP 可能隐藏其 SSID，这意味着它不会被广播到附近的设备。在这种情况下，ESP32 和 ESP8266 无法扫描到 AP。要解决这个问题，您可以手动输入 AP 的 SSID 和密码进行连接。
- AP 已满载或故障：如果 AP 已满载或故障，它可能无法处理新的连接请求，这会导致 ESP32 和 ESP8266 无法连接到 AP。您可以尝试等待一段时间，然后再次扫描 AP。
- ESP32 或 ESP8266 的软件问题：有时候，ESP32 或 ESP8266 的软件可能会出现问题，导致无法正确扫描 AP。在这种情况下，您可以尝试重置 ESP32 或 ESP8266，并重新启动 Wi-Fi 功能。如果问题仍然存在，您可能需要更新 ESP32 或 ESP8266 的固件。
- 其他因素：其他因素，如无线干扰、安全设置、网络配置等，也可能会影响 ESP32 或 ESP8266 的 Wi-Fi 功能。在这种情况下，您需要仔细检查 Wi-Fi 环境并进行相应的设置。

4.10.27 [Scan] 最多能够扫描多少个 AP？

能够扫描到的 AP 最大个数没有限制，取决于扫描时周边 AP 的数目与扫描参数的配置，比如每个信道停留的时间，停留时间越长越可能找到全部的 AP。

4.10.28 [Scan] 连接时周围存在多个相同 ssid/password 时能否选出最佳 AP 连接？

默认情况下为 WIFI_FAST_SCAN, 总是连接第一个扫描到的 AP。如果要连接最佳 AP，需要在设置 station 时将 scan_method 配置成 WIFI_ALL_CHANNEL_SCAN，同时配置 sort_method 来决定选择 RSSI 最强或者是最安全的 AP。

4.10.29 [Scan] wifi_sta_config_t 中 scan_method 怎么配置？全信道扫描和快速扫描的区别在哪里？

全信道扫描和快速扫描是用在连接前寻找合适 AP 所需要的，scan_method 设定了 fast_scan，可以配合 threshold 来过滤信号或加密方式不强的 AP。

- 选择了 fast_scan 会在扫描到第一个匹配的 AP 的情况下停止扫描，然后进行连接，节省连接的时间。

- 选择了 all_channel_scan 的时候扫描会进行全信道扫描，然后根据 sort_method 中设定的排序方法，存储四个信号最好或者加密方式最安全的 AP，等到扫描结束后选择其中信号最好或者加密方式最安全的 AP 进行连接。

4.10.30 [LWIP] 如何获取 socket 的错误码？

- ESP-IDF v4.0 版本以上 (含 v4.0) 标准的做法是 socket API 返回失败后直接通过 *errno* 的值来获取错误码。
- ESP-IDF v4.0 版本以下标准的做法是 socket API 返回失败后调用 *getsockopt(sockfd, SOL_SOCKET, SO_ERROR, ⋯)* 的方式获取错误码，否则当多个 socket 并行操作的时候可能会获取到不正确的错误码。

4.10.31 [LWIP] 默认 TCP keep-alive 时间为多少？

默认情况下，如果连续两个小时收不到任何 TCP 报文，会每隔 75 秒发送一个 TCP keep-alive 报文，连续发送 9 个 tcp keep-alive 报文，如果依然收不到对方发过来的任何报文 LWIP 会断开 TCP 连接。

Keep-alive 可通过 socket option 进行配置。

4.10.32 [LWIP] TCP 重传间隔？

ESP32 作为发送方时，TCP 协议的重传间隔初始值为 3 秒，如果接收方没有发送 ACK 消息，则会依据 Jacoboson 算法决定下次重传间隔, 即指数级地增加重传间隔时间，一般是按照 2、4、8、16、32 秒逐渐增加。这个重传间隔时间不是固定的，TCP 协议的实现者可以通过调整一些参数，如超时时间、滑动窗口大小等来影响重传间隔的计算。

4.10.33 [LWIP] 最多能够创建多少个 socket ？

最多 32 个，默认为 10 个。

4.10.34 [Sleep] ESP32 有哪几种休眠方式及其区别是什么？

- 一共有三种休眠方式: Modem sleep, Light sleep 和 Deep sleep。
  - Modem sleep: WiFi 协议规定的 station WMM 休眠方式 (station 发送 NULL 数据帧通知 AP 休眠或醒来)，station 连接上 AP 之后自动开启，进入休眠状态后关闭射频模块，休眠期间保持和 AP 的连接，station 断开连接后 modem sleep 不工作。ESP32 modem sleep 进入休眠状态后还可以选择降低 CPU 时钟频率，进一步降低电流。
  - Light sleep: 基于 modem sleep 的 station 休眠方式，和 modem sleep 的不同之处在于进入休眠状态后不仅关闭射频模块，还暂停 CPU，退出休眠状态后 CPU 从断点处继续运行。
  - Deep sleep: 非 WiFi 协议规定的休眠方式，进入休眠状态后关闭除 RTC 模块外的所有其他模块，退出休眠状态后整个系统重新运行 (类似于系统重启)，休眠期间不能保持和 AP 的连接。

4.10.35 [Sleep] ESP32 modem sleep 动态调频功能在哪打开？

在 `menuconfig > Component Config > Power Management > Support for power management > Enable dynamic frequency scaling (DFS) at startup` 中打开。

4.10.36 [Sleep] ESP32 modem sleep 降频功能最低能降到多少？

目前 CPU 时钟最低能降到 40 MHz。

4.10.37 [Sleep] ESP32 modem sleep 平均电流大小影响因素？

ESP32 的 modem sleep 是通过设定一个唤醒周期，每个周期开始时打开芯片的射频进行通信其余时间关闭射频来降低功耗。

该模式下平均电流的大小受多种因素影响，下面列举了一些主要的影响因素：

- 唤醒周期：如果设定的唤醒周期越短，则单位时间内芯片唤醒的越频繁，平均电流也会相应增大。
- 信号质量：如果 Wi-Fi 信号质量较差，芯片会不断尝试重新连接或发送数据，或者改用较大发射功率的通信协议进行数据通信，这些都会导致平均电流增大。
- 硬件配置：芯片的硬件配置也会对功耗产生影响，如 CPU 单核还是双核、CPU 时钟频率、CPU 空闲时间比、电源电压、是否外接晶振等因素都会对平均电流大小产生影响。
- 其他因素：例如测试路由器发送 beacon 时间点是否准确，是否发送过多的广播包，芯片本身是否有外设模块工作等

4.10.38 [Sleep] 为什么测到的 modem sleep 平均电流偏高？

- 测试过程中有较多的 Wi-Fi 数据收发。数据收发越多，进入休眠状态的机会越少，平均电流就越高。
- 测试用的路由器发送 beacon 时间点不准确。Station 需要定时醒来监听 beacon，若 beacon 时间点不准确，station 会等待较长时间，进入休眠状态的时间就越少，平均电流就越高。
- 测试过程中有外设模块在工作，请关闭外设模块再进行测试。
- 开启了 station + softap 模式，modem sleep 只在 station only 模式下才会降低电流。

4.10.39 [Sleep] 为什么测到的 light sleep 平均电流偏高？

除了上述四个原因之外还可能是：

- 应用层代码在不停地运行，CPU 没有机会暂停。
- 应用层使用了 ets timer 或者 esp timer，且 timer 的超时时间间隔较短，CPU 没有机会暂停。

4.10.40 [Sleep] ESP32 有哪几种 Wi-Fi 节能模式及其区别？

ESP32 的节能模式一共有三种类型：modem 最小节能模式、modem 最大节能模式、以及不节能模式。

- modem 最小节能模式：该模式为默认模式。在该模式下，ESP32 从 Light-sleep 中醒来收 beacon 的时间间隔由路由器端的 DTIM 决定，为 (DTIM * 102.4) ms，即假如路由器的 DTIM 为 1，则每隔 100 ms ESP32 会醒来进行一次收包。
- modem 最大节能模式：在该模式下，ESP32 从 Light-sleep 中醒来收 beacon 的时间间隔由 `wifi_sta_config_t` 这个结构体中的 `listen_interval` 参数决定，为 (listen interval * 102.4) ms，即假如路由器的 DTIM 为 1，而 listen_interval = 10，则每隔 1 s ESP32 会醒来进行一次收包。
- 不节能模式：不进行节能处理。

4.10.41 ESP8266 是否支持 802.11k/v/r 协议？

当前只支持 802.11k 和 802.11v，可参考示例 roaming。

4.10.42 ESP32 Wi-Fi 支持相同的 SSID 不同的 AP 之间漫游吗？

支持，当前支持 802.11k 和 802.11v 协议，请参考示例 roaming。

4.10.43 [Connect] NONOS_SDK *2.1.0* 升级到 *2.2.2* 后，连接时间变长？

请升级到 NONOS_SDK *master* 版本，该版本中解决了 CCMP 加密与某些 AP 不兼容的问题。

4.10.44 ESP32 如何收发 Wi-Fi 802.11 数据包？

- 可以通过如下 API 进行 802.11 数据包收发：

```
esp_err_t esp_wifi_80211_tx(wifi_interface_t ifx, const void *buffer, int␣
↪len, bool en_sys_seq);
esp_wifi_set_promiscuous_rx_cb(wifi_sniffer_cb);
```

- 上述 API 在 MDF 项目中有用到，可以参考：mconfig_chain。

4.10.45 [Connect] ESP32 系列 & ESP8266 路由器连接失败有哪些可能原因？

- 检查配置中的 SSID 与 Password 是否正确。
- 不建议使用中文 SSID，可能存在不同中文编码带来的异常。
- 需要注意 bssid_set 的设置，如果不需要指定路由的 MAC 地址，那么需配置 stationConf.bssid_set = 0。
- wifi_config_t wifi_config 建议使用静态变量 *static* 来定义。

4.10.46 [Connect] ESP8266 有那些配网方式？

- SmartConfig 模式：一键配置方式，设备在 sniffer 模式扫描特征包的方式。
- SoftAP 模式：设备开启 SoftAP，手机连接 SoftAP 后建立稳定的 TCP/UDP 连接后，发送 SSID 和密码。
- WPS 模式：此方式需要设备中增加按键；或连接到设备的 SoftAP 后使用手机软件控制开启 WPS。

4.10.47 [Connect] SmartConfig 配网 Wi-Fi 参数信息有哪些要求？

SmartConfig 是一种通过局域网广播方式配置 Wi-Fi 参数的方案，用户可以通过使用配套的 APP 将 Wi-Fi 账号和密码发送给设备。下面是 SmartConfig 配网 Wi-Fi 参数信息的要求：

- SSID 名称：支持中英文和数字字符，长度不超过 32 个字节。
- Wi-Fi 密码：8-64 个字符，区分大小写。
- Wi-Fi 安全加密方式：目前 SmartConfig 支持的加密方式有：WPA、WPA2 和 WEP，不支持开放式无加密方式。

4.10.48 [Connect] ESP8266 Wi-Fi 是否支持 WPA2 企业级加密？

- 支持。请参考示例 wpa2_enterprise。
- 可使用 FreeRADIUS 服务搭建 RADIUS 服务器，请参考 FreeRADIUS。

4.10.49 [Connect] ESP32 保持 Wi-Fi 连接的低功耗模式有哪些？

- 在保存 Wi-Fi 连接的场景中，芯片会在 Active 和 Modem-sleep 模式之间自动切换，功耗也会在两种模式间变化。
- ESP32 支持在 light sleep 下 Wi-Fi 保活，自动唤醒间隔由 DTIM 参数决定。
- 例程参见：ESP-IDF - > examples - > wifi - > power_save。

4.10.50 乐鑫芯片是否支持 WPA3？

- ESP32 系列：esp-idf 从 release/v4.1 版本开始支持 WPA3，默认使能，可在 menuconfig > Component config > Wi-Fi 中配置。
- ESP8266：ESP8266_RTOS_SDK 的 release/v3.4 分支开始支持 WPA3，默认使能，可在 menuconfig > Component config > Wi-Fi 中配置。

4.10.51 [Connect] 当环境内存在多个相同 SSID 时，设备如何连接？

- 设备会连接优先扫描到的 AP 设备。
- 如果想要根据信号质量等排序，可以使用 Scan 方法自主筛选。
- 如果想要连接指定 AP, 可以在连接参数中填入 BSSID 信息。

4.10.52 [Connect] ESP8266 有中继器方案吗？

- 乐鑫官方未推出中继类应用方案。
- 社区中有相关中继的应用，可以在 github 中查询，中继速率建议基于实际测试。

4.10.53 ESP32 数据帧和管理帧的重传次数是多少？是否可以配置？

重传次数是 31 次，不可以配置。

4.10.54 ESP32 如何自定义 hostname？

- 以 ESP-IDF v4.2 为例，可以在 menuconfig > Component Config > LWIP > Local netif hostname，然后输入指定的 hostname 即可。
- 不同的版本在命名上可能略有区别。

4.10.55 如何获取 802.11 无线数据包？

- 可以参考 ESP-IDF 编程文档中的 Wireshark 使用指南 。
- 需要注意的是，所使用的无线网卡需要支持 Monitor 模式。

4.10.56 ESP32 Wi-Fi 支持 PMF(Protected Management Frames) 和 PFS(Perfect Forward Secrecy) 吗？

WPA2/WPA3 中均支持 PMF，WPA3 中支持 PFS。

4.10.57 ESP8266 在使用 esptouch v2 出现 AES PN 错误 log？

- ESP8266 收到路由器重传了好几次的包会报这个错误，但是不影响使用。

4.10.58 ESP32 WFA 认证支持多播吗？

- 不支持，建议参考 ASD-1148 方式测试。

4.10.59 使用 ESP32，是否可以在建立热点之前，先扫描所有的 AP 以及所占用的信道，从中选择一个占用最小最干净的信道来建立自己的 AP 呢？

- 可以在建立热点之前，先扫描所有的 AP 以及所占用的信道，参考 API esp_wifi_scan_get_ap_records。
- 不能自动选择最干净的信道来建立自己的 AP，需要自定义信道选择算法。

4.10.60 使用 ESP32，ESP-IDF 版本为 release/v3.3，Wi-Fi Scan 时，当有多个相同的 SSID 时，获取的列表中有多个重复的 SSID，是否有 API 进行过滤，只保留一个 SSID？

- 不能对重复 SSID 进行过滤。因为 SSID 重复不代表是同一个路由器，扫描到的 SSID 相同的路由器的 BSSID 是不同的。

4.10.61 ESP8266 是否支持 EDCF (AC) 方案？

当前最新 master 版本的 ESP8266-RTOS-SDK 支持 EDCF (AC) 应用，但没有应用实例。您可以在 `menuconfig > Component config -> Wi-Fi` 配置中开启 Wi-Fi QoS 配置，以获得支持。

4.10.62 使用 master 版本的 ESP8266-RTOS-SDK，开启 Wi-Fi Qos 应用获得 EDCF 的支持，请问 ESP8266 是如何决定哪个数据包应该分配到 EDCF AC 类别的?

- 可以通过设置 IPH_TOS_SET(iphdr, tos) 来确定。

4.10.63 使用 ESP32，在不考虑内存与功耗的情况下，如何配置最大 Wi-Fi 传输速度与稳定性呢？

- 如需配置最大 Wi-Fi 传输速度与稳定性，请参考 ESP-IDF 编程指南中 如何提高 Wi-Fi 性能，在 menuconfig 中设置相关配置参数即可。配置选项路径可在 menuconfig 界面中，通过“/”来搜索。最优配置参数需根据实际当前的环境进行测试。

4.10.64 ESP8266 作为 Wi-Fi SoftAP 模式，最多支持多少个 Station 设备连接？

- ESP8266 最多支持 8 个 Station 设备连接。

4.10.65 使用 ESP32 设备作为 Station 模式，如何获取 CSI 数据?

- 通过调用“esp_wifi_set_csi_rx_cb()”可获取 CSI 数据。参见 API 说明。
- 具体使用方法参见 Espressif CSI 示例

4.10.66 ESP32 在 AP + STA 模式连接 Wi-Fi 后，任意开启关闭 AP 模式是否会影响 Wi-Fi 连接？

- ESP32 在 AP + STA 双模式下进行 Wi-Fi 连接后，可以任意开启关闭 AP 模式，不影响 Wi-Fi 连接。

4.10.67 ESP32 使用 release/v3.3 版本的 ESP-IDF 进行开发，只需要蓝牙功能，如何通过软件关闭 Wi-Fi 功能？

- 调用 esp_wifi_stop() 可关闭 Wi-Fi 功能。API 说明参见 esp_err_t esp_wifi_stop(void)。
- 若需要回收 Wi-Fi 占用的资源，则还需要调用 esp_wifi_deinit()，API 说明请参见 esp_err_t esp_wifi_deinit(void)。
- 以下是一个简单的示例代码：

```
#include "esp_wifi.h"
#include "esp_bt.h"

void app_main()
{
```

(下页继续)

(续上页)

```
  // 关闭 Wi-Fi 功能
  esp_wifi_stop();

  // 初始化蓝牙功能
  esp_bt_controller_config_t bt_cfg = BT_CONTROLLER_INIT_CONFIG_DEFAULT();
  esp_bt_controller_init(&bt_cfg);
  esp_bt_controller_enable(ESP_BT_MODE_BTDM);

  // ...
}
```

在这个示例中，先调用 esp_wifi_stop() 函数关闭 Wi-Fi，然后再初始化蓝牙功能。需要注意的是，一旦关闭了 Wi-Fi 功能，就无法再使用 Wi-Fi 相关的 API 了。

4.10.68 使用 ESP-IDF 开发，esp_wifi_80211_tx() 接口只能发送数据包，是否有对应的接收函数接口？

- 接收数据包是使用回调的方法， 如下：

```
esp_wifi_set_promiscuous_rx_cb(wifi_sniffer_cb);
esp_wifi_set_promiscuous(true);
```

- 另一个开源项目中有用到该方法，可参考 esp-mdf。

4.10.69 esptouch 配网失败概率较高的原因有哪些？

CHIP: ESP32, ESP32S2, ESP32S3, ESP32C3, ESP8266

- 手机连接的热点使用人数较多。
- 手机连接的热点信号质量较差。
- 路由器不转发组播数据。
- 路由器开启了双频合一，手机连接到 5G 频段。

4.10.70 ESP32 使用 Wi-Fi 时 IRAM 不足，如何优化？

- 可以在 menuconfig 里关闭 `WIFI_IRAM_OPT`、`WIFI_RX_IRAM_OPT` 以及 `LWIP_IRAM_OPTIMIZATION` 来优化 IRAM 空间，但这样会降低 Wi-Fi 的性能。

4.10.71 ESP32 如何测试 Wi-Fi 传输距离？

- 可以使用 iperf 示例 并配置为 iperf UDP 模式，然后不断地拉开 ESP 设备，检测在怎样的距离 Wi-Fi 数据传输速率会降至 0。

4.10.72 ESP32 使用 Wi-Fi 通信时 MTU 的长度最大能设置多大，需要在哪进行设置？

- 利用 Wi-Fi 通信时，MTU 的长度最大只能设置为 1500。可通过 LwIP 组件中的 `netif > mtu` 来修改该数值，不过不建议进行修改。

4.10.73 ESP32 模组挂机测试有时会打印类似如下 log，代表什么含义？

log 信息如下：

```
[21-01-27_14:53:56]I (81447377) wifi:new:<7,0>, old:<7,2>, ap:<255,255>, sta:
↪<7,0>, prof:1
[21-01-27_14:53:57]I (81448397) wifi:new:<7,2>, old:<7,0>, ap:<255,255>, sta:
↪<7,2>, prof:1
[21-01-27_14:53:58]I (81449417) wifi:new:<7,0>, old:<7,2>, ap:<255,255>, sta:
↪<7,0>, prof:1
[21-01-27_14:53:59]I (81450337) wifi:new:<7,2>, old:<7,0>, ap:<255,255>, sta:
↪<7,2>, prof:1
```

- 其中，`new` 后的数值表示当前主次信道；`old` 后的数值表示上次主次信道；`ap` 后的数值表示当前 ESP32 AP 的主次信道，若没有使能 softAP 对应的值就是 255；`sta` 后的数值表示当前 ESP32 sta 的主次信道；`prof` 是 nvs 里面存储的 ESP32 softAP 的信道。
- 有关次信道代表的数值，请参考 wifi_second_chan_t。
- 上述 log 信息表示路由器在 HT20 和 HT40 minus 之间切换，可以检查下路由器的 Wi-Fi 频宽设置。

4.10.74 ESP32 在 AP + STA 模式下，如何关闭 AP 模式?

- 关闭 AP 模式通过 esp_wifi_set_mode(wifi_mode_t mode); 函数来设置。
- 调用 esp_wifi_set_mode(WIFI_MODE_STA); 即可。

4.10.75 ESP32 使用 Wi-Fi 的功能后，是否 ADC2 的所有通道都不能使用了？

- ESP32 在使用 Wi-Fi 的情况下，没有被 Wi-Fi 占用的 ADC2 的引脚可以做普通 GPIO 使用。可参考官方 ADC 说明。

4.10.76 Wi-Fi 模块如何设置国家码？

CHIP: ESP8266 | ESP32 | ESP32 | ESP32-C3

- 可以通过调用 esp_wifi_set_country 接口设置国家码。

4.10.77 当 ESP32 用作 SoftAP 连接苹果手机时，手机提示”低安全性 WPA/WPA2(TKIP) 并不安全。如果这是您的无线局域网，请配置路由器以使用 WPA2(AES) 或 WPA3 安全类型“，该如何解决？

IDF: release/v4.0 及以上

- 可以参考下面的代码进行设置:

```
wifi_config_t wifi_config = {
    .ap = {
        .ssid = EXAMPLE_ESP_WIFI_SSID,
        .ssid_len = strlen(EXAMPLE_ESP_WIFI_SSID),
        .channel = EXAMPLE_ESP_WIFI_CHANNEL,
        .password = EXAMPLE_ESP_WIFI_PASS,
        .max_connection = EXAMPLE_MAX_STA_CONN,
        .authmode = WIFI_AUTH_WPA2_PSK,
        .pairwise_cipher = WIFI_CIPHER_TYPE_CCMP
    },
};
```

- WIFI_AUTH_WPA2_PSK 是 AES，也叫 CCMP。WIFI_AUTH_WPA_PSK 是 TKIP。WIFI_AUTH_WPA_WPA2_PSK 是 TKIP+CCMP。

4.10.78 ESP32 的 Wi-Fi 模块仅支持 2.4 GHz 频率的带宽，如果在进行连网配置时使用 2.4G 和 5G 多频合一的路由器，Wi-Fi 能否配网成功？

- 路由器设置为多频合一的模式（一个 Wi-Fi 账号同时支持 2.4 GHz 和 5 GHz），ESP32 设备可以正常连接 Wi-Fi。

4.10.79 ESP32 用作 AP 模式时如何获取连接进来的 station 的 RSSI？

- 可以调用接口 esp_wifi_ap_get_sta_list，参考如下代码：

```
{
    wifi_sta_list_t wifi_sta_list;
    esp_wifi_ap_get_sta_list(&wifi_sta_list);
    for (int i = 0; i < wifi_sta_list.num; i++) {
        printf("mac address: %02x:%02x:%02x:%02x:%02x:%02x\t rssi:%d\n",wifi_sta_
↪list.sta[i].mac[0], wifi_sta_list.sta[i].mac[1],wifi_sta_list.sta[i].mac[2],
                  wifi_sta_list.sta[i].mac[3],wifi_sta_list.sta[i].mac[4],wifi_
↪sta_list.sta[i].mac[5],wifi_sta_list.sta[i].rssi);
    }
}
```

- `esp_wifi_ap_get_sta_list` API 获取到的 RSSI 为一段时间内的平均值，不是实时的 RSSI。之前的 RSSI 权重为 13，新的 RSSI 的权重为 3。在 >= 100ms 时更新 RSSI，更新时需要使用旧的 rssi_avg：`rssi_avg = rssi_avg*13/16 + new_rssi * 3/16`。

4.10.80 ESP32 支持 FTM(Fine Timing Measurement) 吗？

- 不支持，FTM 需要硬件支持，ESP32 没有对应的硬件。
- 当前 ESP32-S2 和 ESP32-C3 在硬件上支持 FTM。
- ESP-IDF v4.3-beta1 开始支持 FTM。
- 关于 FTM 的更多内容以及例程，请参考 FTM。

4.10.81 当 ESP32 设置为 STA+AP 共存时，能否指定通过 STA 或者 AP 接口发送数据？

问题背景：

ESP32 作为 AP 默认的网段是 192.168.4.x，作为 STA 连接的路由器网段也在 192.168.4.x，PC 连接到该路由器并创建 tcp server，此时 ESP32 作 tcp client 无法建立到 PC 的 tcp 连接。

解决方案：

- ESP32 可以指定通过 STA 或者 AP 接口发送数据，可参考例程 tcp_client_multi_net。该例程中同时使用了 Ethernet 接口和 STA 接口，可以指定接口发送数据。
- 有两种方式将 socket 绑定到某个接口：
  - 使用 netif name (使用 socket 选项 SO_BINDTODEVICE)
  - 使用 netif local IP address (通过 esp_netif_get_ip_info() 获取接口 IP，调用 bind() 绑定)

注解：

- 绑定 STA 接口可以建立 ESP32 和 PC 的 tcp 连接，绑定 AP 接口无法建立 ESP32 和 PC 的 tcp 连接；
- 默认情况下可以建立 ESP32 到手机的 tcp 连接 (手机作为 STA 接入 ESP32)。

4.10.82 ESP8266 wpa2_enterprise 如何开启 Wi-Fi 调试功能?

- 使用 idf.py menuconfig 开启 menuconfig 配置，然后配置以下参数：

```
menuconfig==>Component config ==>Wi-Fi ==>
[*]Enable WiFi debug log ==>The DEBUG level is enabled (Verbose)
[*]WiFi debug log submodule
[*] scan
[*] NET80211
[*] wpa
[*] wpa2_enterprise

menuconfig==>Component config ==>Supplicant ==>
[*] Print debug messages from WPA Supplicant
```

4.10.83 Wi-Fi 信号格数有对应标准吗？

CHIP: ESP8266 | ESP32 | ESP32 | ESP32-C3

- 对于 Wi-Fi 信号格数并没有对应的标准，可以根据接收到的 RSSI 进行折算，比如接收到的 RSSI 范围是 [0,-96]，如果要求信号强度的格数为 5 格，那 [0~-20] 就为满格，以此类推。

4.10.84 WFA 漏洞修复最新情况？

CHIP: ESP32 | ESP32-S2 | ESP32-C3 | ESP8266

- 详情请参考乐鑫官网上 Wi-Fi 安全公告。

4.10.85 Wi-Fi 连接失败时产生的错误码代表什么？

CHIP: ESP32

- Wi-Fi 连接过程中出错都会让状态转移到 init，并且 log 里会有 16 进制数表示，例如 wifi:state, auth-> init(200)。前两位表示原因，后两位表示收到或者发送的管理帧的类型代码。常见的帧类型代码有 00 (什么都没收到，表示超时)、A0（disassoc）、B0（auth）和 C0（deauth）。
- 前两位表示的原因可以从 WiFi Reason Code 里查看。后两位可以直接从管理帧代码里查看。

4.10.86 使用 ESP32 Release/v3.3 版本的 SDK 下载 Station 例程，无法连接不加密的 Wi-Fi，是什么原因？

- 例程下默认是连接加密模式的 AP，如下设置：

```
.threshold.authmode = WIFI_AUTH_WPA2_PSK,
```

- 若连接不加密的 AP，需将以下参数改为 0，

```
.threshold.authmode = 0,
```

- AP 模式选择说明可参见 esp_wifi_types。

4.10.87 ESP32-S2 芯片，Wi-Fi 通信的物理层最大速率是多少？

- ESP32-S2 Wi-Fi 通信的物理层最大速率为 150 Mbps。

4.10.88 ESP 模块是否支持 EAP-FAST?

:CHIP: ESP32 | ESP32-S2 | ESP32-C3 :

- 支持，请参考 wifi_eap_fast demo。

4.10.89 ESP 模块支持 WiFi NAN (Neighbor Awareness Networking) 协议吗？

CHIP: ESP8266 | ESP32 | ESP32-C3 | ESP32-S2 | ESP32-S3

- 不支持。

4.10.90 使用 ESP32，ESP-IDF 版本为 release/v3.3，配置路由器时，是否有 API 可以直接判断输入的密码不正确？

- 没有 API 可直接判断密码错误，依据 Wi-Fi 协议标准，当密码出错时，路由器并不会明确告诉 station 四次握手是由于密码出错了。正常情况下获取密码是 4 个包（1/4 帧、2/4 帧、3/4 帧、4/4 帧），当密码正确时 AP 会发送 3/4 帧，而当密码错误时 AP 不会发送 3/4 帧而是会重发 1/4 帧。 但是当 AP 发送了 3/4 帧，但由于某种原因而在空气中丢掉时，AP 也会重发 1/4 帧。 因此，对于 station 来说，无法准确区分这两种情况，最终都是上报 204 错误，或者 14 错误。
- 可参考 Wi-Fi 原因代码。

4.10.91 基于 ESP-IDF v4.4 版本的 SDK 测试 ESP32 的 Station 例程，如何支持 WPA3 加密模式？

- 开启 `menuconfig → Component config → Wi-Fi → Enable WPA3-Personal` 的配置；
- 在应用程序中设置 `pmf_cfg` 里 `capable = true`；
- 可参考 Wi-Fi Security 说明。

4.10.92 ESP32 如何加快 Wi-Fi 的连接速度？

如下措施均可以加快 ESP32 的 Wi-Fi 连接速度：

- 设置 CPU 频率到最大，可以加快密钥计算速度。除此外还可以设置 FLASH 参数为 `QIO`、`80 MHz`，代价是增加功耗。
- 关闭 `CONFIG_LWIP_DHCP_DOES_ARP_CHECK`，可以大幅降低获取 IP 的时间，代价是不检查局域网中是否有 IP 地址冲突。
- 打开 `CONFIG_LWIP_DHCP_RESTORE_LAST_IP`，保存上次获得的 IP 地址，dhcp start 时直接发送 dhcp request，省去 dhcp discover 过程。
- 固定扫描信道。

4.10.93 ESP32 WPA2 企业级认证是否支持 Cisco CCKM 模式？

- 目前不支持该模式，虽然 esp_wifi_driver.h 中的枚举有 WPA2_AUTH_CCKM，但是目前不支持。

4.10.94 使用 wpa2_enterprise（EAP-TLS 方式），客户端证书最大支持长度是多少？

- 最大支持 4 KB。

4.10.95 ESP8089 是否支持 Wi-Fi Direct 模式？

- ESP8089 支持 Wi-Fi Direct 模式，但 ESP8089 只能使用默认的固定的程序，无法进行二次开发。

4.10.96 环境中有很多 AP，ESP32 如何连接 RSSI 不低于配置阈值的 AP?

- 在 ESP32 staion 模式下，有一个 wifi_sta_config_t 的结构体，下面有 2 个变量，分别是 sort_method 和 threshold 变量，通过给这两个变量赋值来设置 RSSI 阈值。

4.10.97 ESP32 Wi-Fi 出现信标丢失 (beacon lost) 且在 6 秒钟之后给 AP 发 5 个探测请求 (probe request)，此时 AP 没回应就会导致断开连接，这个 6 秒钟可以配置吗？

用 API esp_wifi_set_inactive_time 即可配置。

4.10.98 ESP32 Wi-Fi 可以使用 PSRAM 吗？

- 关于 Wi-Fi 使用 PSRAM 的信息，请参考 使用 PSRAM。

4.10.99 [Connect] ESP32 系列产品如何从软件、硬件方面来排查 Wi-Fi 连不上路由器的问题？

可以按以下步骤来排查问题：

- 首先通过 Wi-Fi 错误码 判断可能的失败原因。
- 然后，当在 ESP32 连接不上路由器时，尝试连接其他设备到该路由器来定位是路由器还是 ESP32 问题：
  - 如手机也无法连上路由器，请排查路由器是否存在问题。
  - 如手机可以正常连上路由器，请排查 ESP32 是否存在问题。
- 排查路由器问题的步骤：
  - 查看路由器是否处于断电重启的阶段，在此阶段将无法正常连接此路由器，需要等待一段时间至路由器初始化完成后才能正常连接。
  - 查看配置的 SSID 和密码是否与路由器一致。
  - 查看在配置路由器为 OPEN 模式后是否能正常连上。
  - 查看是否能正常连上其他路由器。
- 排查 ESP32 问题的步骤：
  - 排查 ESP32 硬件部分：
    * 查看是否是特定的 ESP32 才会出现此问题，如仅有固定的少许 ESP32 出现此问题，统计出现问题的 ESP32 的概率并比较它们和正常 ESP32 的硬件差异。
  - 排查 ESP32 软件部分：
    * 查看使用 ESP-IDF 里的 station 示例 是否能正常连上 Wi-Fi，此处示例里默认存在重连机制，可以同步观察在几次重连后是否能正常连上 Wi-Fi。

* 查看配置的 SSID 和密码是否与路由器一致。
* 查看在配置路由器为 OPEN 模式后是否能正常连上。
* 查看在 Wi-Fi 连接前的代码逻辑里额外调用 API `esp_wifi_set_ps(WIFI_PS_NONE)` 后是否能正常连上 Wi-Fi。

- 如进行上述所有步骤仍然没有定位到问题，建议进行 Wi-Fi 抓包来进一步分析，可参考 乐鑫 Wireshark 使用指南。

4.10.100 ESP32 连上路由器后会每 5 分钟会反复打印几次 W (798209) wifi:<ba-add>idx:0 (ifx:0, f0:2f:74:9b:20:78), tid:0, ssn:154, winSize:64 与 W (798216) wifi:<ba-del>idx 并明显发现 ESP32 的功耗增大，这是什么原因？

- 首先此日志往往没有问题，这里是 Wi-Fi 块确认机制的相关日志，`ba-add` 表示 ESP32 收到路由器的添加块确认请求帧，`ba-del` 表示 ESP32 收到路由器的删除块确认请求帧。打印频繁说明路由器一直在发包。
- 如果是每五分钟定期观察到此日志，往往是路由器在进行组秘钥更新，可以通过以下步骤来进一步验证：
  - 在 wpa_supplicant_process_1_of_2() 里进行日志打印来确认是不是每 5 分钟调用了此函数来配合路由器每五分钟进行组秘钥更新。
  - 查看路由器的 Wi-Fi 配置界面是否存在 组秘钥更新时间选项并被配置为 5 分钟。

4.10.101 ESP32 使用函数 esp_wifi_config_80211_tx_rate() 为何无法固定 Wi-Fi 发送速率来保持稳定传输？

- esp_wifi_config_80211_tx_rate() 函数用来配置 esp_wifi_80211_tx() 这个函数的发送速率。
- 如要设置并固定 Wi-Fi 的发送速率，请使用函数 esp_wifi_internal_set_fix_rate。

4.10.102 ESP32 做 station 连接路由器时发现没有正常获取到 IP，如何调试？

- 打开 lwIP 里 DHCP 的调试日志，在 ESP-IDF menuconfig 配置 `Component config > LWIP > Enable LWIP Debug(Y)` 和 `Component config -> LWIP > Enable DHCP debug messages(Y)`。
- 早期 IDF 版本没有上述选项时，请参考 lwipopts.h 里的 806 到 807 行，将这两行代码里的 `LWIP_DBG_OFF` 都改成 `LWIP_DBG_ON`，如下所示。

```
#define DHCP_DEBUG          LWIP_DBG_ON
#define LWIP_DEBUG          LWIP_DBG_ON
```

4.10.103 ESP32 做 softAP 时发现连接它的 station 没有正常获取到 IP，如何调试？

请将 dhcpserver.c 中的 `#define DHCPS_DEBUG 0` 修改为 `#define DHCPS_DEBUG 1`，即可打开 lwIP 里 DHCP 的调试日志进调试。

4.10.104 在 ESP-IDF menuconfig 配置 `Component config > PHY > Max Wi-Fi TX power(dBm)` 来调整 Wi-Fi 发射功率后实际功率如何？比如设置 17 dBm 时实际最大发射功率是多少？

- 对于 ESP32，此时的实际最大发射功率为 16 dbm，具体请参考 esp_wifi_set_max_tx_power() 函数描述的映射规则。
- 对于 ESP32-C3，在 menuconfig 中配置的最大发射功率值即为实际最大功率值。

4.10.105 ESP-IDF 目前支持连接 UTF-8 编码的中文 SSID 路由器，是否有方法连接到编码为 GB2312 的中文 SSID 路由器？

此时让 ESP 设备端的编码方式和路由器保持一致即可，比如这种情况下让 ESP 设备端也采用基于 GB2312 编码的中文 SSID。

4.10.106 ESP32 在连接上路由器后发现在空闲状态下功耗偏高，大约有 60 mA 的平均电流，如何排查？

- 此时建议进行 Wi-Fi 抓包来进一步分析，可参考 乐鑫 Wireshark 使用指南。抓包后查看设备发送的 NULL data 包里是否包含 `NULL(1)`，其中若每 10 秒发送一次 `NULL(1)` 则说明是和路由器在进行保活交互。
- 也可以查看 Wi-Fi 抓包结果里的 beacon 包中 `TIM(Traffic Indication Map)` 字段，如果 `Traffic Indication` 等于 1，说明存在广播包缓存 (Group Frames Buffered)，ESP32 在此时会打开 RF，导致功耗增高。

4.10.107 当 ESP 终端产品需要销往全球时，对应的 Wi-Fi 国家码要如何配置？

- 需要在不同国家的产品中，设置不同的 Wi-Fi 国家码。
- 默认的国家码配置可以用于大多数国家，但不能兼容一些特殊情况。默认的国家码为 `CHINA {.cc="CN", .schan=1, .nchan=13, policy=WIFI_COUNTRY_POLICY_AUTO}`，在 ESP-IDF v5.0 后，默认为 `"01" (world safe mode) {.cc="01", .schan=1, .nchan=11, .policy=WIFI_COUNTRY_POLICY_AUTO}`）。由于 12 和 13 信道默认为被动扫描，所以不会违反大多数国家的法规。同时 ESP 终端产品连上路由器后国家码会自动根据路由器改变。断开路由器后，会自动配置为默认的国家码。

注解:

- 此时可能存在一个问题：如果路由器隐藏了 SSID，且于 12 或 13 信道，ESP 终端产品就扫描不到路由器。此时需要设置 `policy=WIFI_COUNTRY_POLICY_MANUAL` 来让 ESP 终端产品在 12、13 信道进行主动扫描。
- 对于其他特殊的国家，比如日本支持 1-14 信道，14 信道只支持 802.11b。ESP 终端产品在默认配置下，无法连接 14 信道的路由器。

4.10.108 进行 iperf 测试时发现一段时间后速率会下降甚至中断发射，这是什么原因，需要如何解决？

- 可能原因：
  - 网络环境不好
  - 电脑或手机与 ESP32-S2 或 ESP32-S3 softAP 的兼容性问题，导致断线或者吞吐速率下降。
- 解决方法：
  - 针对第一种情况，尝试更换网络环境或者在屏蔽箱里进行测试。
  - 针对第二种情况，关闭 `menuconfig > Component config > Wi-Fi > WiFi AMPDU RX` 选项，如果还存在断线现象，关闭 `menuconfig > Component config > Wi-Fi > WiFi AMPDU TX` 选项。

注解:

- AMPDU 代表聚合 MAC 协议数据单元，是 IEEE 802.11n 标准中用来提高网络吞吐量的技术。
- 关闭 `WiFi AMPDU RX` 表示不支持接收 AMPDU 包，此时会影响设备的 RX 性能。
- 关闭 `WiFi AMPDU TX` 表示不支持发送 AMPDU 包，此时会影响设备的 TX 性能。

4.10.109 基于 ESP-IDF v5.0 版本的 SDK 创建 ESP32-S3 设备作为 Wi-Fi AP 模式，当手机连接上 AP 后，会频繁打印如下日志，是什么原因？

```
 (13964) wifi:<ba-del>idx
 (13964) wifi:<ba-add>idx:2 (ifx:1, 48:2c:a0:7b:4e:ba), tid:0, ssn:5,␣
↪winSize:64
```

打印此日志是因为一直在创建、删除 A-MPDU，此打印只是辅助作用，不影响通信。若需要屏蔽该日志，可以在 Wi-Fi 初始化程序之前加上如下代码进行测试：

```
esp_log_level_set("wifi", ESP_LOG_ERROR);
```

4.10.110 ESP32 的网口 (LAN8720) 与 Wi-Fi (Wifi-AP) 能否共存？

可以共存的。将两个连接的检测事件写成一个就可以实现共存。

4.10.111 ESP32 在弱网环境或干扰环境下，Wi-Fi 连上以后获取 IP 地址比较慢如何优化？

- 可以在 Wi-Fi start 之后先关闭 Modem-sleep *esp_wifi_set_ps(WIFI_PS_NONE);*，在获取到 *IP_EVENT_STA_GOT_IP* 事件后再开启 Modem-sleep。
- 对于断开重连情况，可以在连接之前先主动关闭 Modem-sleep, 获取到 *IP_EVENT_STA_GOT_IP* 事件后再开启 Modem-sleep。
- 注意：该优化对于 Wi-Fi/BT 共存场景不适用。

4.10.112 ESP32/ESP32-S2/ESP32-S3 工作在 SoftAP 模式时，与其他厂商手机、PC 等进行通信时容易出现断连该如何优化？

建议关闭 menuconfig 里的 `WiFi AMPDU RX` 和 `WiFi AMPDU TX` 选项。

4.10.113 ESP32 Wi-Fi TX power 的取值范围是多少？

- ESP32 Wi-Fi TX power 的取值范围为 2-20 dBm。在 ESP-IDF 中，可以使用函数 `esp_wifi_set_max_tx_power()` 设置 TX power 的最大值，同时也可以使用 `esp_wifi_get_max_tx_power()` 函数获取当前系统所支持的最大 TX power 值。
- 需要注意的是，设置 TX power 过高可能会影响系统的稳定性和电池寿命，同时也可能违反国家和地区的无线电规定，因此应该谨慎使用。详细请参考 esp_wifi_set_max_tx_power API。

4.10.114 使用 ESP32 时如何获取 Wi-Fi RSSI 值？

在 ESP-IDF release/v4.1 中，当 ESP32 作为 station 使用时，要获取连接到的 AP 的 RSSI，可以使用以下代码示例：

```
wifi_ap_record_t ap_info;
if (esp_wifi_sta_get_ap_info(&ap_info) == ESP_OK) {
  int rssi = ap_info.rssi;
  // 处理 rssi
}
```

`wifi_ap_record_t` 结构体中包含了连接到的 AP 的信息，包括 SSID 、BSSID 、频道、加密方式等，RSSI 字段则表示 AP 的 RSSI 值。调用 `esp_wifi_sta_get_ap_info()` 函数即可获取该结构体信息。API 说明参见 esp_err_t esp_wifi_sta_get_ap_info(wifi_ap_record_t *ap_info)。

4.10.115 ESP32 支持 WPA3 企业版吗？

- ESP32 支持 WPA/WPA2/WPA3/WPA2-Enterprise/WPA3-Enterprise/WAPI/WPS 和 DPP Wi-Fi 功能。有关信息，请参考 'ESP32 Wi-Fi 功能列表 <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/wifi.html#esp32-wi-fi-feature-list>'__。
- 在 esp-idf release/v5.0 版本 SDK 中，我们提供了 wifi_enterprise 示例。在 ESP-IDF 中，支持设置 WPA3-Enterprise 模式进行测试。可通过如下步骤进行配置 `idf.py menuconfig > Example Configuration > Enterprise configuration to be used > WPA3_ENT`。

4.10.116 ESP 模组支持 WAPI (Wireless LAN Authentication and Privacy Infrastructure) 功能吗？

- 支持，请参考 WIFI_AUTH_WAPI_PSK。

4.10.117 使用 ESP32 作为 Wi-Fi Station 连接路由器，如何增加扫描路由器的时间？

- 在 ESP32 中，默认情况下 1 ~ 11 信道为主动扫描，12 ~ 13 信道为被动扫描。主动扫描和被动扫描所需时间不同，详情可参考 Wi-Fi 扫描配置。主动扫描的默认时间是每个信道 120 ms，被动扫描为每个信道 360 ms。如果希望增加扫描时间，可在 `esp_wifi_start()` 函数之前，调用如下函数来增加扫描路由器的时间：

```
extern void scan_set_act_duration(uint32_t min, uint32_t max);
extern void scan_set_pas_duration(uint32_t time);
scan_set_act_duration(50, 500);
scan_set_pas_duration(500);
```

4.10.118 ESP32 是否支持 LDPC？

- 支持。不需要额外的配置或调用，已经在驱动中实现了。

CHAPTER 5

硬件相关

[English]

5.1 芯片功能对比

[English]

5.1.1 请从编程开发方式、性能表现、功耗表现等方面列举一下 ESP32 单核与双核的区别？

ESP32 单核与双核的主要差异是双核多了一个独立核心，可以把一些实时性高的操作放在该独立核心上。

- 单核与双核的编程方式一致，不过单核芯片需要配置 FreeRTOS 运行在单核上，双核芯片则无需此步骤。配置路径：`make menuconfig` > `Component config` > `FreeRTOS` > `[*] Run FreeRTOS only on first core`。
- 性能表现仅在高负载运算时有差异（例如 AI 算法，高实时性中断），其余使用上无明显差异。
- 功耗仅在 Modem-sleep 模式下会有细微差别，此时双核芯片的功耗略高于单核芯片。详情可参考 《ESP32 技术参考手册》 。

5.1.2 ESP32 芯片版本 v3.0 在软硬件使用上和之前版本的芯片有什么区别呢？

- 软件上使用无区别，是兼容之前的固件的，硬件上修复了一些 bug。
- 具体的设计变化可以参考文档 《ESP32 芯片版本 v3.0 使用指南》 。

5.1.3 ESP32 的 GPIO34 ~ GPIO39 管脚是否只能设置为输入模式？

- ESP32 的 GPIO34 ~ GPIO39 只能设置为输入模式，没有软件上拉或下拉功能，不能设置为输出模式。
- 详情可参见 外设说明。

5.1.4 ESP32 有适配 Linux 平台驱动吗？

有适配，请参考 esp-hosted 示例。

注解： 该示例适配 802.3 协议，并不是 802.11 协议。

5.1.5 模组屏蔽盖上的二维码扫描数据如何解读？

- 若二维码扫描后读取数据为 0920118CAAB5D2B7B4，那么其中 09 为工厂代码，20 为 20** 生产年份（本示例中为 2020 年），11 为本年第几周生产，后 12 位 8CAAB5D2B7B4 为设备 MAC 地址。关于此信息的最新说明，请参考 《模组包装信息》 。

5.1.6 ESP32 的 VDD3P3_RTC 是否支持单独电池供电？

- ESP32 内部 RTC 域不可以独立工作，需要主 CPU 参与配置，单独使用电池供电时依然无法抵御突然掉电的情况。
- 如果需要系统掉电时时钟信息保存，可以添加外部 RTC 时钟芯片。

5.1.7 ESP32-PICO-D4 和 ESP32-PICO-V3 以及 ESP32-PICO-V3-02 有什么区别？

- 内置芯片版本：ESP32-PICO-V3 和 ESP32-PICO-V3-02 的核心是 ESP32 (ECO V3) 芯片，ESP32-PICO-D4 的核心是 ESP32 (ECO V1) 芯片。
- 封装尺寸：ESP32-PICO-D4 和 ESP32-PICO-V3 的尺寸为 7 × 7 × 0.94 (mm)，ESP32-PICO-V3-02 的尺寸为 7 × 7 × 1.11 (mm)。
- 内置 flash：ESP32-PICO-D4 和 ESP32-PICO-V3 集成了 4 MB 的 SPI Flash，而 ESP32-PICO-V3-02 集成了 8 MB flash 和 2 MB PSRAM。
- 管脚差异：可以参考 《ESP32-PICO-V3-02 技术规格书》 中的“与 ESP32-PICO-V3 和 ESP32-PICO-D4 兼容性”章节。

5.1.8 ESP8266 是否支持 32 MHz 晶振频率？

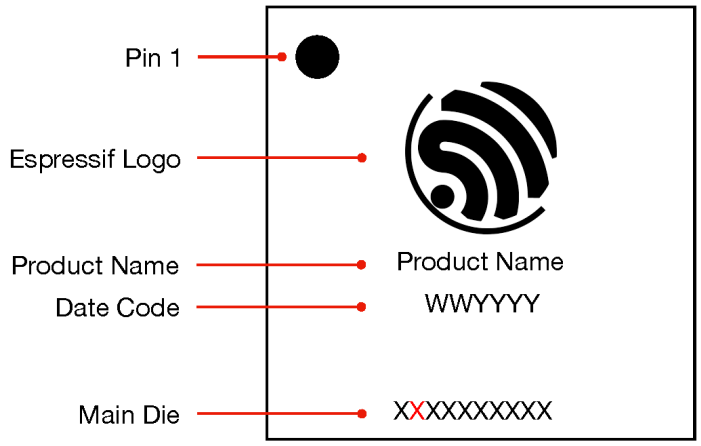
- 不支持，ESP8266 支持 26 MHz 和 40 MHz 晶振频率，推荐用 26 MHz。

5.1.9 ESP 系列产品是否支持 Zephyr？

- ESP 系列产品对 Zephyr 的支持可以参考 乐鑫对 Zephyr 的最新支持，目前仅适配了部分功能模块，后续会进一步更新。如果您有相关的功能需求，可以先在 Zephyr Github issue 上查询或者提问。
- 也可以从 Zephyr 官方文档 的 XTENSA Boards 和 RISCV Boards 找到 ESP 产品的相关资料。

5.1.10 如何通过芯片丝印，识别芯片版本？

芯片版本可通过查看芯片丝印最下方 main die 行的第二位进行识别。



乐鑫各系列芯片的芯片版本与 main die 第二位字符一一对应，具体如下：

芯片系列	芯片版本	丝印标记
ESP32	v0.0	A
	v1.0	B
	v1.1	F
	v3.0	E
	v3.1	G
ESP32-S2	v0.0	A
	v1.0	B
ESP32-C3	v0.0	A
	v0.1	B
	v0.2	C
	v0.3	D
	v0.4	E
ESP32-S3	v0.0	A
	v0.1	B
	v0.2	C
ESP32-C2/ESP8684	v0.0	A
	v1.0	AA
	v1.1	B
	v1.2	C
ESP32-C6	v0.0	A
ESP32-H2	v0.0	A
	v0.1	B

- 有关不同芯片版本的详细差异，请从乐鑫 文档页面 查看对应芯片系列的勘误表文档。
- 有关芯片丝印的完整介绍，请参考 《乐鑫芯片包装信息》 。

5.2 开发板使用

[English]

5.2.1 ESP32-Korvo V1.1 开发板是否集成 LED driver 芯片？

乐鑫出厂的 ESP32-Korvo V1.1 开发板中集成了 LED driver 芯片。

5.2.2 ESP-EYE 开发板运行发热过高如何改善？

- 降低功耗：如果摄像头并非实时开启，Wi-Fi 可以周期传输，空闲时间可以进入休眠模式以降低功耗。
- 增大散热面积：可以通过在 ESP32 芯片上方增加散热片实现。

5.2.3 若开发板不使用 USB 供电，应如何使用管脚供电？

- 第一种方法：”3V3 连接 3V3” + “GND 连接 GND”（如果开发板存在非 3.3 V 供电的器件，则该器件将无法使用）。
- 第二种方法：”5V 连接 5V” + “GND 连接 GND”。

注解： 供电电流需要满足 500 mA。

5.2.4 ESP8266 连接手机热点，出现如下报错，有哪些原因？

```
wifi: state : 0 -> 2 (b0)
wifi: state : 2 -> 0 (200)
simple wifi : Disconnect reason : 2
```

- 请检查外接天线状态、路由器是否存在以及 SSID 是否正确。

5.2.5 开发板 ESP32-Korvo-DU1906 中 DU1906 芯片音频数据通过什么协议与 ESP32 交互？

- 开发板 ESP32-Korvo-DU1906 中，DU1906 的音频数据通过 SPI 传给 ESP32。

5.2.6 是否有支持 POE 供电的以太网开发板？

- ESP32-Ethernet-Kit 是支持 POE 供电的以太网开发板。

5.2.7 为什么在 ESP32S2-Kaluga 中 Audio 和 Camera 不能共用？

- Camera 和 Audio 都使用了 I2S 接口进行通讯，而 ESP32S2 只有 1 路 I2S。
- 并且 Camera 使用并行 I2S，而 Audio Codec 使用串行 I2S，这两种模式不能一起工作，因此不能通过 CS 线来分时复用，必须重新配置。

5.2.8 ESP32S2-Kaluga 是否支持蓝牙语音？

- Kaluga 使用的芯片为 ESP32S2，该芯片不支持蓝牙功能，您可以选择使用 ESP32 的开发板或芯片进行相关功能的开发。

5.2.9 ESP32S2-Kaluga 上的 Codec 芯片是否自带功放？最大支持外接多大功率的喇叭？

- Kaluga 使用的 Codec 芯片内部未集成功放，Codec 芯片仅可以驱动低阻抗耳机。
- 因此，开发板集成了一颗型号为 NS415 的 PA，最大可输出功率为 3 W（典型值），推荐使用 5 V 4 欧的扬声器。

5.2.10 ESP32-S2-Kaluga-1 是否有并行 LCD 接口？如果有，硬件上需要如何使用？

- ESP32-S2-Kaluga-1 上有并行 LCD 接口，但是目前暂时没有提供配套的屏幕，硬件设计可以参考 ESP32-S2-Kaluga-1 硬件设计指南。

5.2.11 ESP32S2-Kaluga 是否支持 USB 调试？

- Kaluga 所有版本均内置了 FT2232HL，该芯片自带 USB - JTAG Bridge 以使用 JTAG 调试功能。
- 您需要切换拨码开关来连接 JTAG 相关引脚，在 ESP-IDF menuconfig 中打开 JTAG 调试功能，并且确保 eFuse 没有禁用 JTAG 调试。
- 安装 OpenOCD 以支持 JTAG 调试之后，便可以使用 `idf.py openocd gdb` 、`idf.py openocd gdbgui` 等命令进行调试。

5.2.12 ESP32S2-Kaluga 中为什么需要多加一块数字 MIC 芯片，不可以直接使用 ADC 采集吗？

- Kaluga 可以直接使用内部或外部的 ADC 采集模拟麦克风的信号，但是可能需要自行设计相关电路。
- Kaluga LyraT V1.2 贴了数字麦和模拟麦两种 MIC，Kaluga LyraT V1.3 只贴了模拟麦。使用两种麦克风是为了方便用户对不同种类的麦克风进行评估。
- 数字麦克风引脚直接与 ESP32 管脚连接，通过 I2S 进行通讯。
- 模拟麦克风连接到了 Audio Codec IC，由 Codec IC 内部的 ADC 进行采样，并通过 Codec IC 的 I2S 接口进行通讯。
- Kaluga 使用的 Codec IC 同时支持音频的编码和解码，您可以同时使用音频采集和播放功能，无需使用额外的 ADC 及相关的转换调理电路。

5.2.13 ESP32S2-Kaluga 中的 speaker 与 Audio_Out 接口是否支持同时输出？

- ESP32S2-Kaluga 中的 speaker 与 Audio_Out 接口可以同时输出。
- 如果您使用模拟麦克风，那么您只需要将麦克风的音频 PA 连接至 Codec IC，便可以使用 I2S 与 Codec 进行全双工通讯，同时进行音频采集和播放。
- 如果您使用数字麦克风，那么您只需要将数字麦克风和 Codec IC 连接至 ESP32S2 的 I2S 相关引脚，便可以使用 I2S 进行全双工通讯。

5.2.14 ESP32S2-Kaluga-V1.2 中的 I2C *FPC* CNN 接口如何使用？是否有相关的例程？

- 该 FPC 可供用户在自行开发产品时，通过使用 Kaluga 底板进行功能评估而无需预先设计主控板，方便进行功能测试，因此没有相关例程提供。

5.2.15 ESP32S2-Kaluga-V1.2 中的 4.3 inch *LCD* FPC_CNN 接口是否为并口 LCD 接口？

- 是的，该 FPC 接口可以用于驱动并口的屏幕。

注解:

- 该 FPC 默认未贴，需要用户自行焊接。
- 由于并口会占用大量的 IO 口，因此，音频板和摄像头的功能都会无法使用，或者需要分时复用。
- 目前暂未提供基于并口的 Kaluga LCD 例程，用户可能需要自行实现其驱动。

5.2.16 ESP32S2-Kaluga-V1.2 PCB 上有很多没有焊接元件的地方，是否是运送过程中丢失？

- Kaluga 的每个版本上都有一些元件位的焊盘上无元件的情况，这些是处于未来的升级而预留的位置。
- 例如并口屏的 FPC 接口，由于目前暂未使用，因此没有贴。同理，音频板上的 ES7210 也没有贴。

5.2.17 ESP32-S2-Kaluga-V1.2 开发板配有摄像头，是否有摄像头的例程可以提供？

- ESP32-S2-Kaluga-V1.2 开发板例程代码。
- ESP32-S2-Kaluga-V1.2 开发板摄像头例程。

5.2.18 ESP32 DevKitc 开发板 LED 灯不亮，设备管理器也无法找到该设备，可能是什么原因导致的？

- 检查供电是否正常：插上 USB 线之后供电，用万用表测试引脚 VCC 和 GND 是否有电压。
- 检查是否为特定开发板故障：检查其他的 ESP32 DevKitc 开发板设备用该 USB 线是否正常。
- 若尝试上述方法后仍无法找到原因，可以通过 USB 转 TTL 设备去接线，只需接 ESP32 DevKitc 的 VCC、GND、TXD 引脚，测试是否为芯片问题，用串口助手查看是否能够打印日志。
- 如果可以，请测试串口驱动芯片是否有电压，可以参考 esp32-devkitc 原理图。

5.2.19 文档中有提到 EN 按键，但在购买的开发板上没有找到该按键？

建议检查开发板是否有 Reset 按键，由于 EN 常用做复位功能，部分开发板丝印会标记为 Reset 按键。

5.2.20 使用 ESP32 开发板，连接 Windows 电脑后未在设备管理器中找到串口，有哪些原因？

使用 ESP32 开发板连接到 Windows 电脑后，如果在设备管理器中未找到串口，可能是以下几个原因：

- 未安装驱动程序：在使用 ESP32 开发板连接 Windows 电脑前，需要安装驱动程序。如果没有安装驱动程序或者驱动程序安装不正确，开发板将无法被识别为串口设备。下载安装 FT232R USB UART 驱动。
- USB 线松动或损坏：如果 USB 线松动或损坏，可能会导致开发板无法被正确识别。用户可以更换 USB 线或者检查 USB 线是否插紧，确保 USB 线与电脑之间的连接正常。
- 开发板故障：如果以上两种情况都不存在，那么可能是开发板本身存在故障。用户可以尝试使用其他 USB 端口或者其他电脑进行连接测试，或者进行开发板的硬件检测和维修。

需要注意的是，在进行开发板连接测试时，需要确认开发板的串口设置和驱动程序设置是否正确。有些开发板需要在串口设置中手动选择正确的端口号和波特率，才能正确连接到电脑。同时，一些驱动程序也需要手动设置端口号和波特率，确保与开发板设置一致。

5.2.21 使用 ESP32-LyraT-V4.3 音频开发板，长按 Boot 按键也很难进入下载模式，是什么原因？

- 正确的做法是：长按 Boot 按键，然后按 RST 按键（此时 Boot 按键不松开），然后松开 RST 按键（此时 Boot 依然不松开），当进入下载模式开始下载后，即可松开 Boot 键。

5.2.22 使用 ESP-WROOM-02D 模组，复位信号持续多久后模组会进入复位状态？

- 当输入电平低于 0.6 V 并持续 200 µs 以上时，ESP-WROOM-02D 模组会重启。

5.2.23 ESP32-LyraT-Mini 开发板的原理图中将 ES8311 codec 芯片的模拟量输出连接到了 ES7243 ADC 芯片的输入，这样做的目的是什么？

- AEC 回声参考信号的硬件回采电路将 Codec（ES8311）的 DAC 输出同时传输给喇叭 PA 和 ADC（ES7243）AINLP/N，随后将采集的信号送回 ESP32，用做 AEC 回声消除算法的参考信号。

5.2.24 使用 ESP32-mini-1 模组，串口上电打印日志如下，是什么原因？

```
rst:0x10 (RTCWDT_RTC_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
invalid header: 0xffffffff
invalid header: 0xffffffff
invalid header: 0xffffffff
invalid header: 0xffffffff
invalid header: 0xffffffff
invalid header: 0xffffffff
invalid header: 0xffffffff
invalid header: 0xffffffff
ets Jul 29 2019 12:21:46
```

- ESP32-MINI-1 模组打印如上日志是因为 Flash 没有程序。

5.2.25 ESP32-S3-DevKitC-1 开发板的 RGB LED 连接的是哪个 GPIO?

- ESP32-S3-DevKitC-1 v1.0 版本开发板的 RGB LED 连接的是 GPIO48。
- ESP32-S3-DevKitC-1 v1.1 版本开发板的 RGB LED 连接的是 GPIO38。
- ESP32-S3-DevKitC-1 v1.1 版本开发板将 RGB LED 管脚改为 GPIO38 是因为 ESP32-S3R8V 芯片 的 VDD_SPI 电压已设置为 1.8 V。所以，不同于其他 GPIO，该芯片在 VDD_SPI 电源域中的 GPIO47 和 GPIO48 的工作电压也为 1.8 V。

5.3 硬件设计

[English]

5.3.1 ESP32 中 I2S 信号管脚过于分散，是否可以配置集中一些，例如配置到 GPIO5、GPIO18、GPIO23、GPIO19、GPIO22 或者 GPIO25、GPIO26、GPIO32、GPIO33 管脚上？

- 所有 I2S 的 I/O 均可任意分配。需要注意的是，有的 I/O 只能作为输入，请参考 《ESP32 技术规格书》的外设管脚分配章节和附录中的 IO_MUX 管脚清单。

5.3.2 ESP32 在 Light-sleep 模式下如何避免 VDD3P3_RTC 管脚的电压掉电？

ESP32 进入 Light-sleep 模式后，RTC 掉电会使 VDD3P3_RTC 管脚对应的 GPIO 的电平被拉低，从而导致外部 RTC 或者其他外设无法正常工作。可以采取以下两种方式解决该问题：

- 设置 RTC 硬件电压控制寄存器 RTC_CNTL_REG 来控制电压。具体来说，需要将 RTC_CNTL_REG 寄存器中的 FORCE_PU 和 FORCE_PD 位设置为 1，即 `RTC_CNTL_REG |= RTC_CNTL_FORCE_PU_M | RTC_CNTL_FORCE_PD_M;`。
- 使用 GPIO 保持管脚。ESP32 的 Light-sleep 模式支持 GPIO 保持功能，可以将某些 GPIO 管脚设置为保持管脚，保持在系统进入低功耗模式时的电平状态。具体来说，可以将 VDD3P3_RTC 管脚设置为保持管脚，以保持其电压。相关代码片段如下：

```
esp_sleep_pd_config(ESP_PD_DOMAIN_RTC_PERIPH, ESP_PD_OPTION_ON);
esp_sleep_enable_gpio_wakeup();
gpio_hold_en(GPIO_NUM_X);
```

其中，ESP_PD_DOMAIN_RTC_PERIPH 表示 RTC 子系统的电源域，ESP_PD_OPTION_ON 表示使能电源域，gpio_hold_en() 函数可以将指定的 GPIO 管脚设置为保持管脚。将 VDD3P3_RTC 管脚设置为保持管脚之后，即使系统进入 Light-sleep 模式，该管脚的电压也会保持。

需要注意的是，使用 GPIO 保持功能会增加系统的功耗，因此需要根据具体的应用场景来选择合适的方案。如果只需要保持 RTC 硬件的电源供应，可以使用第一种方法；如果需要保持其他外设的电源供应，可以使用第二种方法。

5.3.3 配置 ESP32 管脚有什么注意事项？

- 大部分数字外设可以通过 GPIO 交换矩阵配置到任意管脚。SDIO、SPI 高速以及模拟类相关功能只能通过 IO MUX 切换使用。
- 管脚使用注意事项可参考 GPIO & RTC GPIO 说明。

注解:

- Strapping 管脚的默认电平，详情参考 《ESP32 技术规格书》 。
- GPIO34 ~ GPIO39（用作输入 IO，并且无上下拉功能）。
- GPIO6 ~ GPIO11 由 flash 管脚占用。
- GPIO1 和 GPIO3 是 UART0 的 TX 和 RX 管脚，无法进行配置。
- 对于带有 PSRAM 的模组，GPIO16 和 GPIO17 由 PSRAM 占用。

5.3.4 乐鑫芯片 GPIO 最大承载电压是多少？

- GPIO 最大耐压设计为 3.6 V。超出部分建议从硬件设计上补充分压电路，否则会造成 GPIO 损坏。

5.3.5 ESP8266 电压电流需求？

- ESP8266 的数字部分的电压范围是 1.8 V ~ 3.3 V。
- 模拟部分的工作电压是 3.0 V ~ 3.6 V，最低 2.7 V。
- 模拟电源峰值 350 mA。
- 数字电源峰值 200 mA。

注解: 选择的 SPI flash 工作电压也需要与 GPIO 的电压匹配。CHIP_EN 工作在 3.0 V ~ 3.6 V，使用 1.8 V GPIO 控制时需要注意电平转换。

5.3.6 乐鑫 Wi-Fi 模组是否有单面板 PCB 的方案？

- ESP32 属于无线模块，射频性能对于 PCB 材质有较高的要求。我们测试过 4 层与 2 层的方案，但未测试过单层的设计。
- 在此不建议使用单层板子的方案，建议产品 PCB 可以使用单层板，贴装我们的模组。单层板子的模组，射频性能无法预估。
- 为保证良好的 RF 性能，我们建议使用 4 层板设计。

5.3.7 使用电池为 ESP8266 供电有哪些注意事项？

- ESP8266 电压范围为 3.0 V ~ 3.6 V，两节 AA 电池可以给 ESP8266 供电。需要注意电池压降是否满足芯片电压范围。
- 锂电池电压范围超过模组要求，并且放电时压降较大，不适合直接给 ESP8266 供电。
- 推荐电池使用 DC/DC 或 LDO 升降压后给 ESP8266 供电，并且注意电源芯片压差要求。

5.3.8 如何获取 ESP32 系列芯片 footprint？

可以在 模组设计 中下载芯片对应的模组参考设计，里面有管脚封装设计。

5.3.9 使用 ESP32-S2 芯片，用了 DVP camera 接口后还能接入语音吗？

ESP32-S2 的 LCD 接口、DVP camera 接口和 I2S 接口共用一套硬件资源，只能同时支持其中一个。

5.3.10 使用 ESP32 模块，使用 GPIO0 和 GPIO4 作为 I2C 信号接口，需要注意什么？

作为 I2C 信号接口时，GPIO0 需要上拉。烧写时，只要保证上电时 GPIO0 能拉低，然后便可释放。GPIO0 无需一直拉低，只有下载的时候需要拉低。

5.3.11 ESP32 的外接 flash 占用了 GPIO6 ~ GPIO11，这 6 个 GPIO 是否还能作为 SPI 来使用？

ESP32 的外接 flash 占用了 GPIO6 ~ GPIO11，这 6 个 GPIO 就不能再作为 SPI 来使用了。

5.3.12 使用 ESP8285 芯片时，是否需要连接外部晶振？

ESP8285 芯片内部无晶振，需要连接外部晶振。

5.3.13 ESP32-D2WD 外接 PSRAM 的参考设计？

建议参考 《ESP32-PICO-D4 技术规格书》 的外围设计原理图章节。

注解： ESP32-D2WD 是 1.8 V flash，所以 VDD_SDIO 需要加电阻和电容，并且连接 1.8 V PSRAM。

5.3.14 ESP32 是否可以用 PWM 或 DAC 来播放音乐？

ESP32 可以用 PWM 或 DAC 来播放音乐，推荐用于提示音播放，可基于 esp-adf/examples/player/pipeline_play_mp3_with_dac_or_pwm 例程进行测试。

5.3.15 为什么 ESP32 模组和 ESP32 芯片的建议工作电压范围不一样？

- 因为它们的工作环境和使用方式不同。- ESP32 芯片是一颗裸片，需要在电路板上加上外围电路才能正常工作。ESP32 芯片的建议工作电压范围为 2.3 V 至 3.6 V，是根据 ESP32 芯片本身的电气参数来决定的。在这个电压范围内，ESP32 芯片能够正常工作，并且可以提供最佳的性能和功耗表现。- ESP32 模组则是已经封装好的整个电路模块，通常会加上稳压电路、外部晶振、外部天线等外围电路，以及其他外设芯片（如 flash、RAM）等，可以直接使用。由于模组上的电路已经经过优化和调试，因此建议工作电压范围会更窄一些。例如，ESP32-WROOM-32 模组的建议工作电压范围就为 3.0 V 至 3.6 V。此外，由于模组要考虑 flash 的电压，所以 ESP32 模组的建议工作电压会更高一些。
- 在使用这些芯片和模组时，需要根据具体情况选择合适的电源和外围电路，以确保它们能够正常工作。
- 更多信息，请对比模组和芯片的 技术规格书。

5.3.16 自主设计模组 flash 擦除速度相比乐鑫模组较慢有哪些原因？

- 由于不同厂家 flash 器件存在差异，擦除扇区部分所需时间也各不相同，该时间差异属于正常现象。
- 如果希望擦除速度较快，可以测试不同厂家 flash 的擦除速度进行综合评估。

5.3.17 ESP8266 为何上电瞬间会电流较大？

- ESP8266 的 RF 和数字电路具有极高的集成度，上电后 RF 自校准，在校准时会需要大电流。
- 模拟部分电路最大的极限电流可能达到 500 mA，数字电路部分最大电流达到 200 mA。
- 常规应用时，平均电流约 100 mA。
- 综上，ESP8266 电源设计需要满足 500 mA 电流。

5.3.18 ESP32 以太网 RMII 时钟选择有哪些？

- 硬件设计上建议使用 GPIO0 作为 RMII 时钟输入的管脚，请注意 GPIO0 在芯片上电时不可为低电平。
- 详细说明请阅读 配置 MAC 与 PHY 指南。

5.3.19 ESP32-LyraT 开发板扬声器接口规格？

- 接口使用 PH-2A 规格连接器。

5.3.20 基于 ESP32 设计的模组，哪些管脚无法被用户使用？

- ESP32-WROOM 系列模组，GPIO6 ~ GPIO11 为 flash 管脚，作为 flash 通信使用，不可被用户使用。
- ESP32-WROVER 系列模组，GPIO16 和 GPIO17 被模组 PSRAM 占用，不可被用户使用。
- 此外，ESP32 有 5 个 Strapping 管脚，在使用时需要额外注意，具体细节请参考 《ESP32 技术规格书》。

5.3.21 ESP32 如何使用管脚复位芯片？

- ESP32 的复位可使用 CHIP_PU 管脚。当 CHIP_PU 为低电平时，复位电平 (VIL_nRST) 要求足够低，并且持续一段时间。注意：该管脚不可浮空。可参见 《ESP32 硬件设计指南》 中的 复位章节。

5.3.22 ESP8266 供电设计需要注意哪些问题？

- 如果是使用 LDO 变压，请确保输入电压在 (2.7 V ~ 3.6 V) 和输出电流（大于 500 mA）要足够大。
- 电源轨去耦电容器必须接近 ESP8266 摆放，等效电阻要足够低。
- ESP8266 不能直连 5 V，仅支持 3.3 V，电压范围 2.7 V ~ 3.6 V。
- 如果是通过 DC-DC 给 ESP8266 供电，必要时要加上 LC 滤波电路。
- 可参考 《ESP8266 硬件设计指南》 中的 电源章节。

5.3.23 ESP8266 使用 TOUT 管脚做 ADC 采样时，超过 0 V ~ 1.0 V 是否会损坏管脚？

- 输入电压在芯片管脚电压范围内均不会损坏管脚（默认为 0 V ~ 3.6 V）。
- 超过采样阈值将会影响采集的数据结果，导致数据结果异常。

5.3.24 使用板载天线的模组，对 PCB 和外壳设计有哪些要求？

- 如产品采用模组进行 on-board 设计，则需注意考虑模组在底板的布局，应尽可能地减小底板对模组 PCB 天线性能的影响。
- 条件允许的情况下，建议将模组 PCB 天线区域延伸出底板板框外，并将模组尽可能地靠近底板板边放置，使天线的馈点距离板边最近。
- 请确保模块不被任何金属的外壳包裹，模块 PCB 天线区域及外扩 15 mm 区域需净空（严禁铺铜、走线、摆放元件）。
- 具体说明请阅读对应模组的 硬件设计指南。

5.3.25 使用 ESP32 GPIO34 ~ GPIO39 是否可作为 UART 的 RX ？

- GPIO34 ~ GPIO39 作为接收使用，可应用 UART 的 RX。

5.3.26 ESP32 模组外接 32 kHz 晶振参考设计？

- 请参考 《ESP32 硬件设计指南》 中的 *RTC* 时钟（可选）章节。

5.3.27 ESP32 模组 flash 是否支持 80 MHz 的 QIO 模式？

- ESP32 模组可以同时支持 flash 模式，QIO 和 flash 速度为 80 MHz。
- 使用 QIO 模式建议使用在二级 Bootloader 中开启，因为部分 flash 状态寄存器默认 QE 未置 1。

5.3.28 如何配置 ESP32 以太网的 RMII 同步时钟？

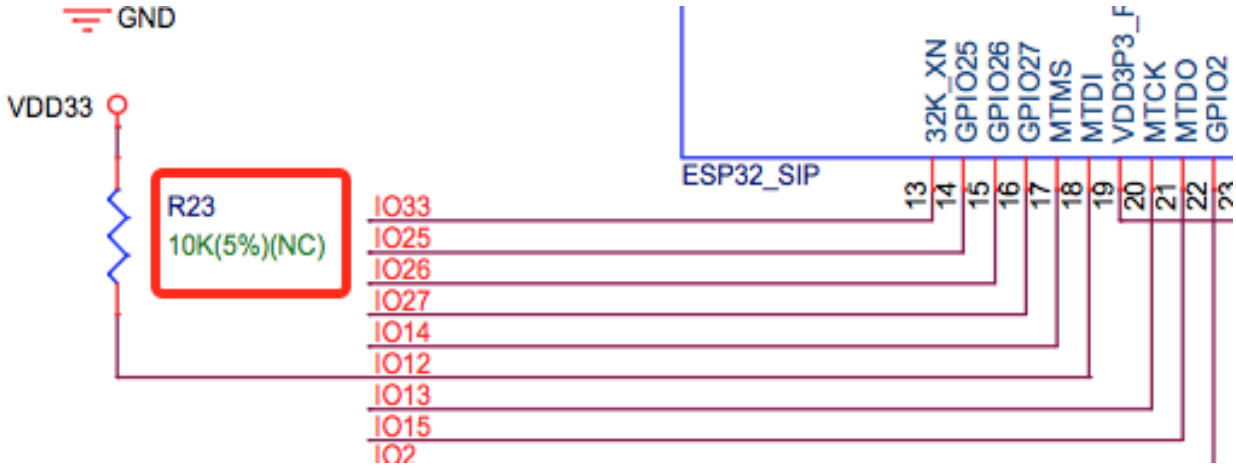
- 请下载 esp-idf/examples/ethernet/basic 例程进行测试。
- IP101 PHY 芯片在 GPIO0 输出 CLK 时会出现网络不稳定的现象，所以推荐 PHY 外接 50 MHz 晶振，GPIO0 作为输入。
- 由于 GPIO0 的特殊性，所以需要配置 IO 控制 PHY 的使能管脚。
- 请阅读 以太网文档。
- 可参考 SCH_ESP32-ETHERNET-KIT 原理图设计。

5.3.29 使用 ESP8266 芯片如何进行硬件复位？硬件复位信号是低电平有效还是高电平有效？复位的条件是什么？

- ESP8266 的 Pin32 EXT_RSTB 为复位管脚。此管脚内部有上拉电阻，低电平有效。为防止外界干扰引起的重启，建议 EXT_RSTB 的走线尽量短，并在 EXT_RSTB 管脚处增加一个 RC 电路。
- ESP8266 的 CHIP_EN 管脚也可作为硬件复位管脚，当使用 CHIP_EN 管脚作为复位管脚时，复位信号是低电平有效。复位条件为当输入电平低于 0.6 V 并持续 200 µs 以上时，ESP8266 会复位重启。我们推荐使用 CHIP_EN 管脚进行芯片复位。可参考 《ESP8266 硬件设计指南》 中的 复位章节。

5.3.30 乐鑫原理图中的 NC 缩写是什么意思？

- NC 是 No Component 的缩写，即不上件。如下图所示，上拉电阻标有 NC，即表示该上拉电阻不上件。



5.3.31 如何在 ESP32-S2 中使用多天线？

- ESP32-S2 的多天线使用和 ESP32 类似，可以参考 《ESP32-WROOM-DA 技术规格书》 中的多天线使用。
- 《ESP-IDF 编程指南》 中提供了详细的操作说明。
- 使用时添加一个 RF 开关，通过开关选择具体工作的天线。

5.3.32 ESP32-C3F SPI CS0 是否需要外接 10 kΩ 上拉电阻？

CHIP: ESP32-C3F

- ESP32-C3F 的 SPI 控制器支持软件可编程的 CS (Chip Select) 管脚，不需要外接 10 kΩ 上拉电阻。- 在 ESP32-C3F 中，可以通过在 SPI 控制器配置中设置 CS 管脚为任意 GPIO 引脚，并在代码中通过设置 GPIO 状态来控制 CS 管脚的电平。当 SPI 总线空闲时，CS 管脚会自动被拉高至 GPIO 引脚的默认状态，不需要外接上拉电阻。
- 需要注意的是，在使用软件可编程的 CS 管脚时，需要在 SPI 总线传输前手动将 CS 管脚拉低，以选择目标设备，并在传输完成后将 CS 管脚拉高，以释放设备。同时，还需要根据实际情况调整 CS 管脚的电平和状态，以确保 SPI 总线的稳定性和可靠性。

5.3.33 ESP-Skainet 有语音识别硬件设计参考吗？

- 请参考 ESP32-Korvo V1.1 用户指南。

5.3.34 硬件上是否有必要接 32 kHz 的 RTC 晶振？

CHIP: ESP32 | ESP32-C3 | ESP32-S3

- 外接 32 KHz 晶振主要是用于 Bluetooth LE Light-sleep 计时，所以应用场景中不使用 Bluetooth LE Light-sleep 时不需要外接。

5.3.35 使用 ESP32-MINI-1 模组，是否可提供 Altium Designer 的元件库？

- 我们的硬件原理图是在 PADS 中开发设计的，在《ESP32-MINI-1 参考设计》中有一个 ASC 的文件，可在 Altium Designer 里转换打开。
- 更多型号的模组的硬件设计资料可在 技术文档 中获取。

5.3.36 ESP8266 的 UART0 的输入电压能由 3.3 V 改为 1.8 V 吗？

- UART0 的电源域是 VDDPST，VDDPST 理论上可以到 1.8 V，所以 UART0 理论上可以改成 1.8 V。

5.3.37 ESP8266 UART0 的电平是由 VDD 决定的，还是由 VDDPST 决定的？

- ESP8266 UART0 的电平是由 VDDPST（硬件电源域）决定的，数字电源电压都是由 VDDPST 决定的。

5.3.38 ESP32-D2WD 芯片外接 PSRAM 软件配置注意事项是什么？

- 需要在 menuconfig 中使能 `CPU frequece 240 Mhz` 和 `RTC clock 80 Mhz`，具体配置如下：
  - `menuconfig > Serial flasher config > Flash SPI Speed (80 Mhz)`
  - `Component config > CPU frequency (240 Mhz)`
  - `Component config > ESP32 specific > [*]Support for external, SPI-connected RAM`
  - `Component config > ESP32 specific > SPI RAM config > Set RAM clock speed (80 Mhz clock speed)`

5.3.39 ESP32 芯片当 VDD 供电从 0 V 慢慢升到 3.3 V 时，芯片为何无法正常启动？

- 出现此问题是由于芯片上电时序不满足要求，时序要求当 VDD 达到 2.3 V 时，EN 电压不应超过 0.6 V。
- 但 VDD 上电时间过慢时，芯片 EN 端的 RC 电路将 EN 延时的功能就丧失了。
- 可以调整 RC 电路，增加电容，调整电阻，或是使用 Reset 芯片管控 EN 状态。
- 建议检测到供给 ESP32 的电压低于 2.3 V 时将 ESP32 的 EN 脚拉低。
- ESP32 上电时序说明参见 《ESP32 技术规格书》 。

5.3.40 使用 ESP32-WROOM-32D 模组，是否可以使用 GPIO12 用作其他功能？

- GPIO12 为 Strapping 管脚，控制 SPI flash 的启动电压。ESP32-WROOM-32D 模组的 SPI flash 启动电压为 3.3 V，因此在上电启动时 GPIO12 需要拉低。
- 若需要使用 GPIO12 用作其他功能，请使用 esptool 工具通过 espefuse.py set_flash_voltage 3.3V 命令将 VDD_SDIO 固定为 3.3 V。
- 硬件上可以将 VDD_SDIO 直接连到 3.3 V 上，这样就不用再烧录 eFuse。
- 在量产阶段，也可以直接将 flash 下载工具里 “config/esp32/utility.confgi” 文件下 ESP32_EFUSE_CONFIG 的默认配置选项修改为 config_voltage = 3.3 V 来下载固件。

5.3.41 ESP32-WROOM-32D 模组的外接 flash，是否可以不使用 GPIO6 ~ GPIO11 的接口？

- ESP32 共有 3 组 SPI（SPI、HSPI 和 VSPI）接口，可以通过 SPI0/1 (HSPI/VSPI) 总线访问外部 flash。但接到其他脚（GPIO6 ~ GPIO11 以外的 GPIO）的外接 flash 不能跑程序，只能接收数据作存储。需要跑程序的 flash 只能接在 GPIO6 ~ GPIO11 接口上。

5.3.42 ESP32 芯片设计模组，PCB 板是否需要加屏蔽盖？

- 是否需要加屏蔽盖取决于具体的应用场景和要求。- 在一些高要求的应用场景，例如无线通讯干扰环境较严峻、电磁兼容性（EMC）测试要求较高等情况下，加装屏蔽盖可以有效地减少外界干扰和 PCB 板上的互相干扰，提高系统的稳定性和可靠性。此时，屏蔽盖应该采用导电材料，并接地处理，以确保其有效性。- 另一方面，如果应用场景较为简单，如无线通讯干扰较小，EMC 要求不高等情况下，加装屏蔽盖的效果可能不是很明显，且可能增加系统成本和复杂度。- 如果板子还有其他信号干扰，比如 2G、3G、4G 或者 Wi-Fi、Bluetooth、Zigbee 等等建议加上屏蔽盖。

5.3.43 ESP32 的 I2S 的 CLK 管脚必须使用 GPIO0、GPIO1 或 GPIO3 吗？

- MCLK 管脚必须使用 GPIO0、GPIO1 或 GPIO3 管脚。其他的时钟管脚可以使用任意的 GPIO。注意，由于 GPIO0 为 Strapping 管脚，一般不推荐用作其他功能。

5.3.44 ESP32-U4WDH 芯片是否支持外接 PSRAM 芯片？

- ESP32-U4WDH 芯片支持外接 PSRAM 芯片，但仅支持乐鑫发布的 ESP-PSRAMXXH 芯片，不支持使用第三方 PSRAM 芯片。
- 硬件设计上，除了 CS 管脚外，其他所有管脚都可以与 Flash 复用，更多指南请参考 《ESP32 硬件设计指南》 。
- 另外，PCB 设计时请注意 PSRAM 的 GND 到 ESP32-U4WDH 的 GND 要尽量短，否则可能会影响信号质量。

5.3.45 ESP32 芯片是否支持使用 SPI0/SPI1 接口外接 SD NAND flash 来存储程序固件（而不是使用默认的 NOR flash）？

- ESP32 芯片不支持使用 SPI0/SPI1（连接程序 flash）接口来外接 SD NAND flash 芯片。
- 如果要存储外部数据，建议使用 ESP32 的 SPI2、SPI3 或 SDIO 接口来外接 NAND SD 芯片。
- SPI2 和 SPI3 可以使用任意 GPIO，但 SDIO 接口则只能使用指定接口，详细说明请见 《ESP32 技术规格书》 中的 外设管脚分配章节。

5.3.46 是否支持基于 ESP32-S3R8 芯片外挂第二个 PSRAM 芯片？

- 不支持。原因如下：
  - PSRAM 芯片与 MSPI 总线相连。MSPI 外设只有两个 CS 信号，一个与 flash 相连，另一个则与 PSRAM 相连。
  - CPU 通过 cache 和 MSPI 访问外部存储器。GPSPI 外设是不能被 cache 访问的。

5.3.47 能否提供 ESP32-S3-WROOM-1 模组的 3D 模型和 Footprint 文件？

- 可在 espressif/kicad-libraries 库中获取模组的 3D 模型和 Footprint 文件。

5.3.48 ESP32/ESP32-S2/ESP32-C3/ESP32-S3 是否支持单独给 RTC 电源域供电来保持芯片低功耗工作？

不支持。以 ESP32 为例，详细信息后续会更新到 ESP32 硬件设计指南 里的 RTC 章节。

5.3.49 有哪些提高 EMC 性能的方法?

- 在硬件层面，可以采取以下措施来提高 PCB 板的 EMC 性能：
  - 采用四层板设计的 EMC 性能优于两层板的硬件设计。
  - 对电源电路增加滤波电路。
  - 在天线电路中添加防静电或磁珠。
  - 在 SPI Flash 电路上增加一个 0 欧姆串联电阻，以降低驱动电流，减少对射频的干扰，调整时序，更好地屏蔽干扰。
  - 尽量保持 GND 完整。
  - 更多硬件设计建议可参考 《ESP 硬件设计指南》 。

5.3.50 ESP32-S3 U0TXD 为什么要预留 499 Ω 的电阻?

- U0TXD 预留 499 Ω 电阻是用于抑制 80 MHz 谐波。详细信息请参考 《ESP32-S3 硬件设计指南》 。

5.3.51 如何在硬件上校准 ESP32-S3 ADC？

- ESP32-S3 已经在芯片内部进行了 ADC 的硬件校准。ESP32-S3 ADC 对噪声敏感，可能导致 ADC 读数出现较大差异。根据使用场景，您可能需要将旁路电容（例如 100 nF 陶瓷电容）连接到使用的 ADC 输入焊盘，以最大限度地减小噪声。此外，还可以使用多重采样来进一步减轻噪声的影响。

5.3.52 如何基于 ESP32 系列芯片设计自动下载电路？

- 可以参考 ESP32-DevKitC 开发板原理图 中自动下载电路的硬件设计。

5.3.53 在 ESP8266 芯片上应该使用哪种晶振？

- ESP8266 芯片需要使用 26 MHz 的晶振来启动芯片。选用的晶振自身精度需在 ±10 PPM。详情请参见《ESP8266 硬件设计指南》。

5.3.54 ESP32-C2、ESP32-C3 和 ESP32-C6 芯片是否支持外接 PSRAM 芯片？

- ESP32-C2、ESP32-C3 和 ESP32-C6 芯片均不支持外接 PSRAM 芯片。

5.3.55 ESP32-C3 采用电池供电时，当供电电压逐渐下降，比如将电池放完电后再充电，ESP32-C3 可能会无法启动。此时，只能首先断开电池与 ESP32-C3 的连接，再重新连接充好电的电池，或者尝试在 3.3 V 引脚和 EN 引脚间连接一个稳压二极管才能让芯片正常启动，出现这种情况的根本原因是什么？有没有最佳的解决办法？

- 根本原因：ESP32-C3 芯片重新上电和复位时，CHIP_EN 管脚需要满足 ESP32-C3 芯片规格书 或 ESP32-C3 硬件设计指南 里的上电时序图要求和说明。如果电池放电和上电比较缓慢，ESP32-C3 可能无法充分复位，从而导致芯片部分单元处于不确定状态。
- 解决办法：目前，若使用电池供电或储能类应用系统，可以通过调整不同 RC 器件值、使用两个电阻分压控制，或使用一颗较为常见的复位芯片来解决此问题。有关 RC 器件值、相关电阻的详细信息，请参见 ESP32-C3 系列芯片硬件设计指南。

5.4 射频相关

[English]

5.4.1 ESP32 模组在 2.8 V 电源下运行，射频性能会有下降吗？

射频会不稳定。建议按照相应 模组技术规格书 中说明的建议工作电压范围提供电压。

5.4.2 乐鑫芯片支持的调制方式有哪些？

- ESP8266 芯片支持的调制方式有：BPSK/QPSK/16QAM/64QAM/DBPSK/DQPSK/CCK。
- ESP32 芯片支持的调制方式有：BPSK/QPSK/16QAM/64QAM/DBPSK/DQPSK/CCK/GFSK Π/4-DQPSK 8-DPSK。
- ESP32-S2 芯片支持的调制方式有：BPSK/QPSK/16QAM/64QAM/DBPSK/DQPSK/CCK。
- ESP32-C3 芯片支持的调制方式有：BPSK/QPSK/16QAM/64QAM/DBPSK/DQPSK/CCK/GFSK。
- ESP32-S3 芯片支持的调制方式有：BPSK/QPSK/16QAM/64QAM/DBPSK/DQPSK/CCK/GFSK。

5.4.3 如何获取乐鑫产品的 RF 相关的信息（如天线描述、天线辐射图等）用于认证？

请联系 商务 获取相关信息。

5.4.4 ESP32 使用 RF Test Tool 时为什么在高温 80 °C 下运行会自动降低发射功率？

- ESP32 的定频测试固件默认不开启温度补偿，当温度较高时功率会变低，如果需要打开温度补偿就需要通过默认的 log 串口向 ESP32 发送 `txpwr_track_en 1 1 0`。

5.4.5 ESP32-WROVER-E 模组如何提高 Wi-Fi 信号的接收距离和强度？(应用场景：Wi-Fi 探针)

- 软件上可以通过 API esp_wifi_set_max_tx_power() 设置最大发射功率。也可通过 menuconfig 进行配置：`Component config > PHY > Max Wi-Fi TX power(dBm)`，默认发射功率为最大值 20 dBm。
- 若发射功率已设置为最大值，可从天线和接收设备方面进行提升：
  - 可以考虑调整模组的摆放方向，使天线较强的辐射方向指向接收设备，使辐射距离最远。
  - 模组的天线附近没有金属或遮挡物，天线背面没有 PCB，Wi-Fi 信号未受整机其它信号干扰。
  - 如果 PCB 天线效果不好，可以考虑更换使用 IE 系列模组，使用定向增益更高的外置 IPEX 天线。
  - 接收设备也可以增加天线辐射效率。

5.4.6 如何烧写 phy_init 数据到 flash 中？

:CHIP: ESP32 :

- 可以通过 power limit tool 烧写。下载 ESP_RF_TEST Tool，解压完成后，打开 EspRFTest-Tool_vx.x_Manual.exe，点击 `help` > `Tool help` > `PowerLimitTool help` 查看详细的操作步骤。

5.4.7 客户自研产品如何优化二次谐波等杂散？

二次谐波主要来源于射频链路辐射和 PA 电源辐射，同时易受到客户底板（板子尺寸）及产品整机影响。因此建议：

- 在射频匹配中使用一个接近 2.4 pF 大小的对地电容，以优化射频链路上的杂散辐射。
- 在 PA 电源 (芯片 3、4 管脚) 入口增加一个串联电感，以减少 PA 电源的杂散辐射。

5.4.8 80 MHz 倍频杂散较差该如何解决？

若 80 MHz 倍频杂散超标，如 160 MHz、240 MHz、320 MHz 等均比较高，可在发送数据 (TXD) 串口线路中串联一个阻值约为 470 Ω 的电阻，即可有效抑制 80 MHz 倍频杂散。

5.4.9 使用乐鑫外接天线的模组，是否需要手动进行功率校准？

不需要。使用外接天线时，请确保天线连接正常的情况下再给模组供电。然后模组会进行自校准，其中包含功率校准。

5.4.10 使用“RF 测试工具”设置 Wi-Fi “TX continues”模式下，默认设置的“default”等级的占空比是多大？

- 默认设置的“default”等级的占空比是 98% 以上，且不可修改。

5.5 工艺与 ESD 防护

[English]

5.5.1 ESP32 ESD 测试注意事项有哪些？

- ESP32 的 ESD（Electrostatic Discharge，静电放电）测试是为了确保 ESP32 设备在遭受静电放电时具有足够的耐受能力。注意事项如下：- ESD 测试应该在 ESD 实验室或者 ESD 防护区域进行，这些地方需要具备良好的接地保护和静电放电保护设施。- 在进行 ESD 测试时，应该使用符合国际标准的 ESD 测试设备，包括 ESD 发生器和 ESD 接地垫等，以确保测试结果的准确性。- 在进行 ESD 测试时，需要稳定的 3.3 V 电压，EN trace 如果太长，容易导致重启。- 应对 ESP32 设备进行多次测试，以验证其耐受能力的可靠性，并对测试结果进行记录和分析。- 如模组无反应，请确认测试的空气放电或接触放电的具体电压。

5.6 生产测试

[English]

5.6.1 为什么部分模组使用 DIO/DOUT 时可以正常下载固件，但使用 QOUT/QIO 时程序无法正常运行？

- 首先需要确认模组内 flash 支持哪些模式，以及模组设计的走线是否满足模式需要。
- 其次检测 flash 状态寄存器的 QE 位，该 bit 位控制 flash 是否使能四线模式。
- 不同的乐鑫芯片/模组采用 flash 厂家不同，部分厂商 flash 默认 QE 关闭。需要通过实际测试判断是否支持四线模式。
- 当 ROM 引导二级引导加载程序时，如果配置的参数使用 QIO 方式读取，会因 QE 关闭而二次读取失败。
- 建议模组使用 DIO 模式烧录，在 menuconfig 中配置 QIO 模式，该配置会在二级引导加载程序中配置 QE 位使能，进而引导 app bin 使用四线模式。

5.6.2 如何获取产测工具？

CHIP: ESP32 | ESP8266

- 请点击 产测工具 进行下载。

5.6.3 ESP32 使用 `esptool.py burn_custom_mac` 命令写入用户自定义 MAC 地址，为什么通过 `esptool.py read_mac` 读到的还是出厂默认的 MAC 地址？

- `esptool.py read_mac` 命令默认只能读到出厂写在 eFuse BLOCK0 的 MAC 地址，而使用 `esptool.py burn_custom_mac` 命令写入的用户自定义的 MAC 地址是写到 eFuse BLOCK3 中，可以使用 `espefuse.py get_custom_mac` 命令来查询写入 eFuse BLOCK3 中的 MAC 地址。
- 更多信息可以参考 esptool 文档。

5.6.4 ESP32-WROVER-E (16 MB flash）模组使用 Flash 下载工具下载多个单独的 bin 文件成功，但下载合并的固件 (12 MB) 失败，是什么原因？

由于合并后的固件绝大部分都是 “0xFF”，压缩率比较高，同样长度的压缩数据解压后的数据量比较大，导致烧录时间久出现烧录超时（默认 7 秒）报错。可以关闭 Flash 下载工具里的 `configure > esp32 > spi_download` 文件里的压缩配置选项来下载合并的固件，如下：

```
compress = False
no_compress = True
```

CHAPTER 6

测试校验

[English]

6.1 功耗测试指南

[English]

6.1.1 ESP32 从 Deep-sleep 中唤醒时为何表现为重启的状态？

进入 Deep-sleep 后，数字内核会断电，CPU 内容丢失，唤醒后需要重新引导固件并加载至内存。在 Deep-sleep 时 RTC 内存保持供电，可以将需要保留的应用信息保存其中，在唤醒后进行加载保留信息。

6.1.2 ESP32 的休眠方式有哪几种？有什么区别？

有 Modem-sleep、Light-sleep 和 Deep-sleep 三种休眠方式。

- Modem-sleep 模式：CPU 正常工作，可以对时钟进行配置。设备作为 station 连接上 AP 之后自动开启，进入休眠状态后关闭射频模块，休眠期间保持和 AP 的连接。如果与 AP 断开连接，ESP32 将无法在 Wi-Fi Modem-sleep 模式下正常运行。ESP32 进入 Modem-sleep 模式后，还可以选择降低 CPU 时钟频率，进一步降低电流。

- Light-sleep 模式：CPU 暂停工作，数字内核时钟受限。与 Modem-sleep 模式的不同之处在于，进入休眠状态后，不仅 RF 模块关闭，CPU 和部分系统时钟也将暂停。退出休眠状态后，CPU 继续运行。
- Deep-sleep 模式：数字内核断电，CPU 内容丢失。进入休眠状态后，关闭除 RTC 模块外的所有其他模块；退出休眠状态后，整个系统重新运行（类似于系统重启）。休眠期间，AP 连接断开。

对应的休眠功耗可参考 ESP32 规格书 中的 *表 8：不同功耗模式下的功耗*。

6.1.3 ESP32 Deep-sleep 可以通过任意 RTC_GPIO 唤醒吗？

可以，RTC_GPIO 管脚配置可以参考 《ESP32 技术规格书》 中的 *管脚定义 > 管脚描述*章节。

6.1.4 ESP8266 的 CHIP_PU 管脚为低电平时，芯片的功耗是多少？

- CHIP_PU 管脚即模组 EN 管脚，管脚设为低电平时，芯片的功耗约为 0.5 μA。
- 在 《ESP8266 技术规格书》 > *功能描述 > 低功耗管理 > 表 3-4. 不同功耗模式下的功耗*中，功耗模式为**关闭**的一栏即代表 CHIP_PU 管脚拉低的状态。

6.1.5 ESP32 进入 Light-sleep 时，仅配置 GPIO 唤醒而不配置定时器唤醒时，底电流为什么会升高？

- 默认情况下，调用函数 esp_light_sleep_start 后不会断电 flash，这是为了防止当设备刚进入休眠又立刻被唤醒时，如果 flash 尚未完全断电又重新上电可能会导致的错误。
- 有关此问题的详细信息，以及如何优化这种场景下的电流功耗，请参考《ESP-IDF 编程指南》中的 Power-down of Flash 小节。

6.1.6 在 ESP32 的 Deep-sleep 模式下，使用内部 150 KHz 的 RTC 时钟或使用外部 32 KHz，哪个功耗更大？

- 若 RTC 时钟源选择的是外部无源 32 kHz 晶振，则功耗没有区别。
- 若硬件上外接了外部有源 32 kHz 晶振，无论选择何者作为 RTC 时钟源，功耗都会上升 50~100 μA。

6.1.7 如果通过降低 CPU 主频的方法减少功耗，为了保证射频模块的正常运行对 CPU 主频有什么要求？

CPU 主频至少需要 80 Mhz 才能保证射频模块的正常工作。

CHAPTER 7

商务常见问题

[English]

7.1 你们的产品通过哪些认证？

您可以在 证书 页面查阅并下载证书文件。

7.2 请问贵司是否获得 ISO 质量管理体系认证？

我司已通过 ISO9001:2015 质量管理体系认证。

7.3 请问你们芯片和模组通过了 REACH、ROHS 等环保认证吗？

我司芯片和模组符合 REACH、RoHS、Prop65 等多项环保认证标准。您可以联络 乐鑫商务 获取并查看证书。

7.4 你们在国内、欧洲、美国或加拿大有代理商吗？

请联系 乐鑫商务 获得您所需信息，我们的工作人员会尽快与您取得联系。

7.5 请问如何能成为乐鑫的代理商？

如果您有意向成为我们的代理商，请将贵司的公司介绍发至此邮箱: sales@espressif.com 。

7.6 贵司的产品信息在哪里可以查看？量产产品有哪些？

感谢您对乐鑫的关注。查看所有产品的基本信息请点击 这里 。查看所有产品的技术文文档请点击 这里 。

7.7 你们的产品有🄵期供货保证吗？

乐鑫承诺旗下产品的供货期至少为 12 年，具体信息请点击 这里 。

7.8 如何查看产品的标准包装量 (SPQ) 和最小起订量 (MOQ)？

各个产品的标准包装量 (SPQ) 和最小起订量 (MOQ) 请点击 这里 参阅《乐鑫产品订购信息》。

7.9 如何购买你们的产品？

若您需要批量采购，请联系 乐鑫商务 ，我们的工作人员会尽快与您取得联系。若您需要少量样品，请 点此 查看购买渠道。

7.10 请问批量采购价格是多少？如何批量采购？

请您将需求提交至 商务问题 ，我们的工作人员会尽快联络您。

7.11 请问你们各产品之间，有什么区别？包括不同系列与型号等。

您可以点击 这里 查看所有产品的选型介绍。同时欢迎您联络我司 商务工作人员 获取详细解答。

7.12 贵司模组本身有固件吗？我需要定制模组/芯片烧录，请问可否在出厂时候完成？价格如何？交期多久？我应如何操作？

乐鑫开发了一套 AT 指令集，方便客户简单快速地使用 AT 指令来控制芯片。乐鑫大部分模组出厂时带有标准 AT 固件。具体更多信息请联系 乐鑫商务 。我们的工作人员会尽快与您取得联系。此外，为简化和缩短客户的制造过程，乐鑫为您提供贴心的定制生产服务。您可以进入 生产服务 查看我司可完成的出厂烧录项目。具体更多信息请联系 乐鑫商务 ，我们的商务支持将尽快协助处理并帮您完成定制。

7.13 请问哪款产品支持 HomeKit？如何获取 Espressif HomeKit SDK？

您可以查阅 Espressif HomeKit SDK 概览 ，Espressif HomeKit SDK 仅提供给已获得 MFi 许可的用户。获取 SDK 时请您务必提供六位数的 Account Number。

7.14 请问贵司的办公地点在哪里？

乐鑫科技（股票代码：688018）是一家全球化的无晶圆厂半导体公司，成立于 2008 年，在中国、捷克、印度、新加坡和巴西均设有办公地，团队来自 30 多个国家和地区。点此查看 乐鑫全球办公室 。

7.15 请问你们技术的联系方式是什么？

请进入 这里 ，告诉我们您遇到的问题或困惑。我们会协助您予以解决。

7.16 请问如何与贵司取得联系？

为了解您的具体问题及需求，请您进入 这里 提交需求信息，我司会尽快与您取得联络。

7.17 请问如何可以快速识别贵司的模组为量产产品或 NPI 新品？

乐鑫通用模组的屏蔽盖上印有产品信息丝印，在丝印的左下角您可以看到一串总长 4 位至 9 位的规格标识码，有关模组丝印的详细说明您也可以进入 这里 查看了解。如果规格标识码的前 2 位显示为 XX 或者 Mn（n 为数字），代表为乐鑫量产产品；如果规格标识码的前 2 位显示不是 XX 和 Mn，代表您拿到的是 NPI 新品。

若您拿到乐鑫 NPI 新品进行开发，非常感谢您成为 NPI 新品体验者。NPI 新品可能在产品硬件状态和软件兼容性还没有达到量产最终状态，在使用过程中会有一些特殊性，具体 NPI 新品使用细节您可以联系 乐鑫技术支持团队。