

ESP8266 FAQ 文档

Status	
Current version	V0.2
Author	Fei Yu
Completion Date	2014.9.23
Reviewer	Lele Li
Completion Date	2014.9.23

☐ CONFIDENTIAL
☐ INTERNAL
☒ PUBLIC

版本信息

日期	版本	撰写人	审核人	修改说明
2014.9.9	0.1	Fei Yu		初稿

免责声明和版权公告

本文中的信息，包括供参考的 URL 地址，如有变更，恕不另行通知。

文档“按现状”提供，不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档不负任何责任，包括使用本文档内信息产生的侵犯任何专利权行为的责任。本文档在此未以禁止反言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。

Wi-Fi 联盟成员标志归 Wi-Fi 联盟所有。

文中提到的所有商标名称、商标和注册商标均属其各自所有者的财产，特此声明。

版权归© 2014 乐鑫信息技术有限公司所有。保留所有权利。

目录

版本信息.....	2
目录.....	3
1、 前言	4
2、 硬件问题	5
2.1. 工作模式	5
2.2. Flash 与内存.....	5
2.3. CPU	7
2.4. GPIO.....	7
2.5. Uart.....	9
2.6. PWM	10
2.7. 晶振	10
2.8. 硬件资料	11
2.9. WatchDog	11
2.10. ESP8266 与 ESP8089 的区别.....	11
3、 SDK 软件.....	12
3.1. 基本结构及编译	12
3.2. 系统任务	13
3.3. softAP&station	13
3.4. 系统定时器	14
3.5. 复位功能	14
3.6. 低功耗	15
3.7. 网络连接.....	16
3.8. AT 透传	16
3.9. AT 指令常见问题	17
4、 云端服务.....	18
4.1. 基本问题	18
4.2. 云端升级.....	18
5、 手机 APP.....	19
5.1. 设备离线	19

1、前言

本文总结了一些客户常见问题。

CONFIDENTIAL

2、硬件问题

ESP8266 尺寸为5x5 mm，ESP8266 模组需要的外围器件有：10个电阻电容电感、1个无源晶振、1个 flash。工作温度范围：-40~125°C。

2.1. 工作模式

1、问：ESP8266 StandAlone mode和 SLAVE-SPI mode分别是什么？

答：ESP8266 有两种工作模式：

1) StandAlone 模式

该模式 ESP8266 作为主芯片，独立运行；

2) SIP 模式（SLAVE-SPI, Slave-SDIO）

该模式 ESP8266 作为从芯片，采用 SPI 或 SDIO 与主芯片通讯，如主芯片采用 ARM。

2.2. Flash 与内存

1、问：Flash 参考容量及型号是什么样的？

答：Flash 规格需满足以下要求：

1) SPI Flash

2) 支持 Quad SPI(推荐) 和 Dual SPI

3) 容量最好在 4Mbit 及以上，如果需要使用云端升级（OTA）或者 SSL 则 flash 容量需要 8Mbit 及以上。

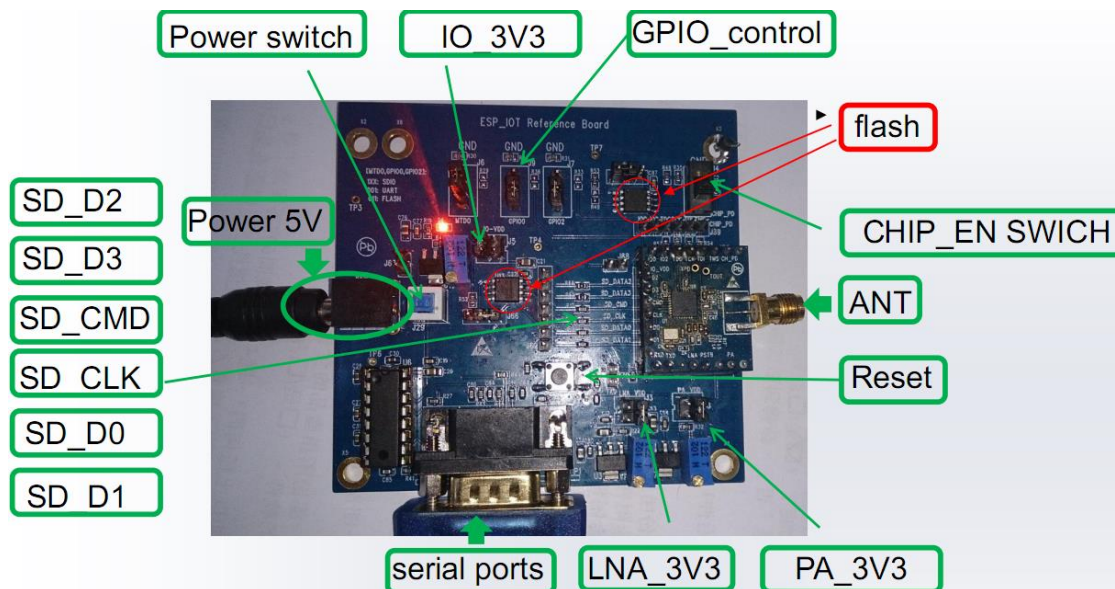
推荐型号 winbond: W25Q40CL

2、问：可用空间是怎样的？

答：

RAM size < 35kB （station模式下，连上路由后，heap+data区大致可用在 35K 左右，具体可用内存大小，可调用接口system_get_free_heap_size获得）

ROM size < 64kB (片上没有 ROM, 用户程序存放在 spi flash 中, 当前 spi flash 大小为512KB, 当前定义可用程序区 <64KB。可通过更换容量更大的spi flash, 从而可支持到4MB。)



3、

问：两个flash的分工？

答：由于ESP8266有两种工作模式，StandAlone和SPI-Slave：

PCB 板中间的 flash 用于 ESP8266 工作在 StandAlone 模式下的；

右上角的 flash 用于 ESP8266 工作在 SPI-Slave 模式下，ESP8266 作为 SPI SLAVE（或 SDIO）连接Host，同时 ESP8266 又作为 SPI master通过 HSPI 和右上角的 flash 连接。

客户实际运用时，只需根据使用ESP8266的模式，选用一种 flash 连接方式即可。

4、问：如何对 Flash 加密，保护客户软件（防抄板）？

答：

方法一：在 ESP8266 芯片中定制客户专用的 id，软件里需要传入正确的客户 id，跟芯片中的客户 id 相符，才能继续运行，否则系统 hang 住。

方法二：在使用烧录工具烧写 flash 时，先读取芯片 id（每颗芯片唯

一), 将芯片 id 存入 flash 特定地址, 在程序中读取芯片 id 和 flash 特定地址中存放的 id, 两者一致, 程序才能正常运行。

5、问: 搜索不到ESP SoftAP?

答:

- 1) 确认是否更换了flash, 请更换成我们原来Demo板上的 flash 或者替换成 W25Q 系列的Flash。
- 2) 尝试调节晶振两边的对地调节电容 (26M 晶振两边的对地调节电容在 8.2pF ~ 12pF) 。

2.3. CPU

1、问: CPU主频跑多少?

答: CPU主频支持80M和160M。

2.4. GPIO

1、问: GPIO口的输出电压电流的规格?

答: GPIO口输出电压为VDD_IO (比如 3.3V), 输出电流应该不超过20mA。

2、问: GPIO口的软件设置方式。

答:

- 1) GPIO API 请参见文档“Espressif IoT SDK 编程手册”中 5.1 GPIO接口API
- 2) 如希望提高设置 GPIO 的速率, 可以采用如下方式, 直接写寄存器控制。

```
GPIO_REG_WRITE(GPIO_ENABLE_ADDRESS, xxxx); //使能哪些 GPIO 口
```

```
GPIO_REG_WRITE(GPIO_OUT_W1TS_ADDRESS, xxxx); // 设置哪些 GPIO 口
```

```
GPIO_REG_WRITE(GPIO_OUT_W1TC_ADDRESS, xxxx); //清除哪些 GPIO 口
```

第一句只要调用一次；后两句控制输出高电平或低电平。

3、问：GPIO口使用示例。

答：以下简单罗列出了几个应用中的 GPIO 口的功能描述。

a) wifi 开关应用：

MTDI---控制继电器，可高低电平控制；

MTCK--- 指示 wifi 工作状态；

MTMS---接复位按键（长按 5s 可完成复位功能）。

b) 温湿度传感器应用：

MTMS---I2C_SCL；

GPIO2---I2C_SCK；

MTCK---复位按键（按住复位按键重新上电，可完成复位）；

GPIO0---wifi 工作状态指示灯；

MTDI--- 与服务器通信的指示灯；

U0RXD---Button，暂未定义功能；

MTDO---LED，暂未定义功能。

c) 智能灯应用：

MTMS---红外接收；

三路 PWM 输出：

MTDI---红色灯控制；

MTDO---绿色灯控制；

MTCK---蓝色灯控制。

d) 2 UART：

UART0: U0RXD+U0TXD---通信

UART1: GPIO2(TXD)---打印

目前uart0可以用来收发用户自己的数据包，uart1用作打印信息。

2.5. Uart

1、问：ESP8266 有几个 uart ？

答：ESP8266 有两个 uart，其中 uart0 有 TX、RX，可做数据传输；uart1 仅有 TX，可以做串口调试信息打印。

2、问：目前支持的波特率范围？

答：110~460800

3、问：怎样修改 uart 波特率？

答：下载请将波特率设为 115200,修改波特率需要做如下操作：

a) uart.c 中最后一句注释掉；

```
void ICACHE_FLASH_ATTR
uart_init(UartBaudRate uart0_br, UartBaudRate uart1_br)
{
    // rom use 74880 baud_rate, here reinitialize
    UartDev.baut_rate = uart0_br;
    uart_config(UART0);
    UartDev.baut_rate = uart1_br;
    uart_config(UART1);
    ETS_UART_INTR_ENABLE();

    // install uart1 putc callback
    //os_install_putc1((void *)uart1_write_char);
}
```

b) user_main.c 中需添加 uart_init(115200,115200);

```
void user_init(void)
{
    #if ESP_PLATFORM
        user_esp_platform_init();
    #endif

    uart_init(115200,115200);

    user_devicefind_init();

    user_webserver_init(SERVER_PORT);
}
```

4、问：串口 log 乱码？

答：关于波特率打印出现乱码，有两种方法来解决：

（1）使用USB转TTL的串口线：例如CP2102，只需接GND RXD TXD。

（2）可能是串口线不支持 74880 这个波特率的打印，请修改波特率为 115200。修改方法如 UART 问题1。

5、问：初始波特率为什么是 74880？

答：这是由于一些最初的设计原因，在板子上电初始跑boot rom的一段log需要在74880的波特率下正常打印。跑到用户程序区后，波特率改为115200后可正常打印的。

2.6. PWM

1、问：ESP8266 支持几路PWM？

答：目前支持4路PWM，硬件实际是1路PWM，由软件实现为4路PWM。

2、问：ESP8266 PWM 频率多少？

答：默认100-500Hz。10kHz也行 但分辨率到不了1/256。

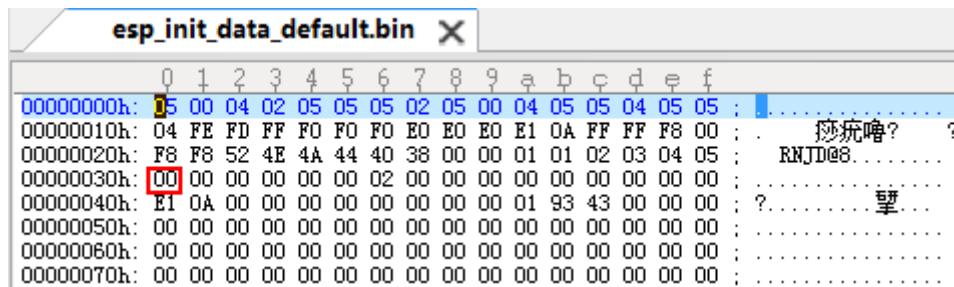
2.7. 晶振

1、问：晶振的要求？

答：要求晶振本身的频偏要在 $\pm 10\text{ppm}$ 以内。

2、问：Demo 使用26M 晶振，如需更换成 40M 晶振，firmware 是否要修改？

答：请将 esp_init_data_default.bin 中下图红色标注的地方改为 00（40M 晶振）。01 表示 26M 晶振。注意更换为 40M 晶振后，两边的对地调节电容需要更改（应该是 18PF）。（26M 晶振两边的对地调节电容在 8.2pF ~ 12pF）



2.8. 硬件资料

1、问：硬件资料使用什么工具打开？

答：电路原理图：orcad 16.6

PCB: Pads Layout 9.5

2.9. WatchDog

1、问：ESP8266有watchdog吗？

答：有。程序异常跑飞的情况下，会自动reset。

2.10. ESP8266 与 ESP8089 的区别

1、问：ESP8266、ESP8089两者的区别？

答：ESP8266 是一颗为物联网定义的产品，特征是它能从 flash boot-up。

与 ESP8089 最大的区别就是：ESP8266 能直接跑 TCP/IP 和应用程序，自带协议（TCP/IP等），ESP8266 底层是有 MAC 的，而 ESP8089 没有，ESP8089 的 MAC 由主IC 实现。

ESP8266 可以作为主IC，让客户进行二次开发；ESP8089 只能作为其他 Host 的从IC，无法做二次开发。

3、SDK 软件

3.1. 基本结构及编译

1、问：ESP8266有操作系统吗？

答：ESP8266目前没有操作系统，后期可能会采用FreeRTOS。

2、问：ESP8266 有main吗？

答：ESP8266 没有main，程序入口为 user_init。

3、问：如何定义一个函数，或者const的全局变量，放到flash里面动态加载运行，还是常驻RAM里面运行。

答：函数若添加 ICACHE_FLASH_ATTR，会动态加载；不添加，则常驻 iRAM。

Const 常驻 dRAM。

4、问：运行程序时出现 exception 如何查证？

答：请在路径 \esp_iot_sdk\bin 的 eagle.app.v6.S 中搜索 exception 报错的地址（形如 0x40XXXXX），定位问题。

5、问：编译时 make 无反应？

答：请注意是否有将 esp_iot_sdk_v0.8.4\examples\IoT_Demo 文件夹下的所有文件拷贝到 APP 文件夹中。

6、问：master-device-key.bin 一定要烧录吗？

答：master-device-key.bin的实际名称是长度为40字节的16进制字符串，在 Espressif website: <http://iot.espressif.cn/#/> 上申请，如果客户不用 Espressif 云端服务（例如客户自建服务器），则无需申请、烧录。

7、问：哪些接口需要在 user_init 中调用，否则容易出现問題，或者不生效？

答：

- (1) `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。
- (2) `system_timer_reinit` 建议在 `user_init` 中调用，否则调用后，需要重新 arm 所有 timer。
- (3) `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。

3.2. 系统任务

1、问：请问如何编写任务才比较合理？

答：参考文档“Espressif IoT SDK 编程手册”，其中有简单举例，`system_os_task` 与 `system_os_post`

3.3. softAP&station

1、问：ESP8266 station 和 softAP 的 mac 地址是否相同？

答：不相同。ESP8266 有两套 MAC，因此可以支持 softAP + station 共存的模式。

2、问：现在最多可以记录多少个 wifi 的ssid和password？

答：最多记录 5 个 AP 的信息

3、问：ESP8266 softAP 可连接几个station？

答：由于内存限制，ESP8266 softAP 最多可连接 4 个 station

4、问：ESP8266 `wifi_softap_set_config` 这类设置 wifi 属性的接口不生效？

答：需要在 `user_init` 中调用；否则，需要重启之后生效，即调用 `system_restart`

5、问：搜索不到ESP SoftAP？

答：

（1）确认是否更换了flash，请更换成我们原来Demo板上的 flash 或者替换成 W25Q 系列的Flash。

（2）尝试调节晶振两边的对地调节电容（26M 晶振两边的对地调节电容在 8.2pF ~ 12pF）。

3.4. 系统定时器

1、问：如何使用 `os_timer_arm_us`，以及定时时间范围？

答：使能us级定时器，在`user_config.h`中 `#define USE_US_TIMER`，并在`user_init`中调用`system_timer_reinit()`，此时可以同时使用 `os_timer_arm_us` 和 `os_timer_arm`

（1）未定义 `USE_US_TIMER` 时：

每秒钟的 tick 数是 $80000000/256=312500$

因此有效时间是 $(2^{31} - 1)/312500 = 6871.947670s$

所以 `os_timer_arm()` 的时间参数范围是0-6871947ms

`os_timer_arm_us`不可用

（2）定义 `USE_US_TIMER` 时：

每秒钟的 tick 数是 $80000000/16=5000000$

因此有效时间是 $(2^{31} - 1)/5000000 = 429.4967294s$

所以 `os_timer_arm()` 的时间参数范围是 0-429496ms

`os_timer_arm_us` 的时间参数范围是 0-429496729us

3.5. 复位功能

1、问：复位后恢复的出场设置参数是什么？

答：system_restore 将 wifi 相关参数复位，即擦了路由信息以及恢复了 softap 默认名称

3.6. 低功耗

1、问：低功耗说明。

答：如下图，其中的“可唤醒”指数据来了自动唤醒。

模式	Modem-Sleep	Light-Sleep	Deep-Sleep
动作	关闭 WiFi Modem 电路； CPU 和其他外设正常运行。	关闭 WiFi Modem 电路、晶振和 PLL ； CPU 和其他外设处于时钟暂停待机状态。	仅 RTC 电路工作，关闭其他电路，芯片处于极低功耗待机状态。
电流	10~20mA	0.5mA	10~20uA
唤醒	可唤醒	可唤醒	无法唤醒，设备依照设定，定时醒来。
应用场景	用于 CPU 需要一直工作的场景。 如 PWM 或 I2S 应用等。 如果没有数据传输，可根据 802.11 标准（如 U-APSD），关闭 WiFi Modem 电路来省电。 例如，在 DTIM3 时，每 sleep 300ms，醒来 3ms 接收 AP 的 Beacon 包等，整体平均电流约 15mA。	用于 CPU 可暂停的应用。 如 WiFi 开关。 如果没有数据传输，可根据 802.11 标准（如 U-APSD），关闭 WiFi Modem 电路，并暂停 CPU 来省电。 例如，在 DTIM3 时，每 sleep 300ms，醒来 3ms 接收 AP 的 Beacon 包等，则整体平均电流约 0.9mA。	用于不需一直保持 WiFi 连接，很长时间才发送一次数据包的应用。 如每 100 秒测量一次温度的传感器。 例如，每 300S 醒来后需 0.3~1s 连上 AP 发送数据，则整体平均电流可远小于 1mA。

2、问：ESP8266 低功耗注意事项？

答：ESP8266 低功耗只针对 station 模式，对于 softAP 则没有低功耗。

目前实现的低功耗为 modem sleep 模式和 deep sleep 模式，modem sleep 目前没有接口控制，自动开启，依据所连AP（路由器）的设置支持DTIM；deep sleep的接口为 system_deep_sleep，设置每休眠多久醒来一次。

3.7. 网络连接

1、 问：最多可以建立几个网络连接？

答：TCP 连接最多可以建立 5 个；UDP 连接个数无限制。

2、 问：ESP8266 作为 tcp client，某些工具作为 tcp server，连接断开后会重连。

答：使用 TCP&UDPDebug 这个工具做server 即可。这个问题是因为某些网络调试工具，在断开连接时没有实现完整的 TCP 退出流程，如果不走完整的 TCP 退出流程，ESP8266 会判断为异常断开，对TCP server 进行重连。

3、 问：ESP8266 是否支持 iperf，如何测试网络吞吐量？

答：ESP8266 不支持使用 iperf，可以参照文档“Espressif IoT SDK 编程手册”中的 espconn 系列接口自行编写程序测试。另，我司内部测试过网络吞吐量，可发邮件向我司申请提供网络吞吐量的测试结果。

3.8. AT 透传

1、 问：ESP8266 AT 透传的通信能力？

答：在硬件流控情况下，数据不丢失，速度11kbits/s,基本达到串口极限

2、 问：使用工具测试，无法退出透传？

答：注意，如果直接用键盘打字输入 +++ ，有可能时间太慢，不被认为是连续的三个+，建议使用如下工具：



字符串输入框：+++

发送新行：不要勾选

点击发送

3.9. AT 指令常见问题

1、问：AT+CIPCLOSE 无法成功断开？

答：使用 TCP&UDPDebug 这个工具做server 即可。这个问题是因为某些网络调试工具，在断开连接时没有实现完整的 TCP 退出流程，如果不走完整的 TCP 退出流程，ESP8266 会判断为异常断开，对TCP server 进行重连。

2、问：AT 测试指令的用途？

答：测试指令是提供给用户作为指令格式帮助，但是帮助格式很占内存所以没有写完整，有些指令没有写测试指令，有些指令只留了OK作为响应。测试指令不会对模块进行操作。

3、问：AT+CIPSEND 发送数据时，提示 busy。

答：在 AT+CIPSEND 发送数据时，回车符号也会被认为是一个数据字节，若发送字节数，超过设置字节数，会提示 busy，并将多余数据丢弃，不影响设定字节范围内的数据发送。

4、 云端服务

有关 Espressif 服务器的所有 API，可在网站查询：<http://iot.espressif.cn/#/api/>

4.1. 基本问题

1、问：网站使用简介，及 master-device-key.bin 如何申请？

答：参考文档“Espressif Cloud Introduction”，推荐使用chrome浏览器下载 master-device-key.bin。

4.2. 云端升级

1、问：固件云端升级成功之后，重启自动运行新固件吗？

答：要调用 system_upgrade_reboot；不调用就不会切换

2、问：客户自建服务器，怎样进行云端升级？

答：客户可以自建服务器，参照文档“云端升级实现方案”第5章 软件实现，调用其中的接口，通过http url，实现云端升级。若用 https url 的话，需要服务器支持Maximum Fragment Length Negotiation，否则内存占用太大。

5、手机 APP

5.1. 设备离线

1、问：更换AP 或者使用3G网络，设备显示离线？

答：请登陆 Espressif 服务器，查证烧录到设备中的 `master-device-key.bin` 所属的产品，是否创建了数据模型。如何创建数据模型，请参考文档“Espressif Cloud Introduction”。

CONFIDENTIAL