ESP8266

应用笔记

固件下载协议

版本 1.0

关于本手册

本文介绍了 ESP8266 的固件下载协议，结构如下所示。

章	标题	内容
第 1 章	概述	介绍固件下载的硬件准备和下载流程。
第 2 章	传输协议	介绍下载固件到 Flash 时传输的数据格式。
第 3 章	固件格式	介绍在 Flash 中的固件格式。
附录 I	编程举例	介绍相关编程举例。

发布说明

日期	版本	发布说明
2016.05	V1.0	首次发布。

相关手册

请通过如下链接下载相关手册。

官网：*http://www.espressif.com/zh-hans/support/download/documents*

官方论坛：*http://bbs.espressif.com/viewtopic.php?f=67&t=225*

文档类型	文档名称
硬件指南	《ESP8266 硬件描述》
	《ESP-WROOM-02 技术规格表》
开发指南	《ESP8266 SDK 入门指南》
	《ESP8266 Non-OS SDK AT 指令集》

目录

1. 概述

1.1. 硬件准备

ESP8266 处于 UART 下载模式时，可以通过外部 MCU 将固件下载到 ESP8266。

1.1.1. 硬件设置

硬件设置如表 1-1 所示。

表 1-1. 硬件设置

设置项	值
UART 下载模式	GPIO0 和 GPIO15：低电平 GPIO2：高电平
波特率	自适应
数据位	8
停止位	1
校验位	无
流控	关闭

1.1.2. 硬件连接

硬件连接如图 1-1 所示。

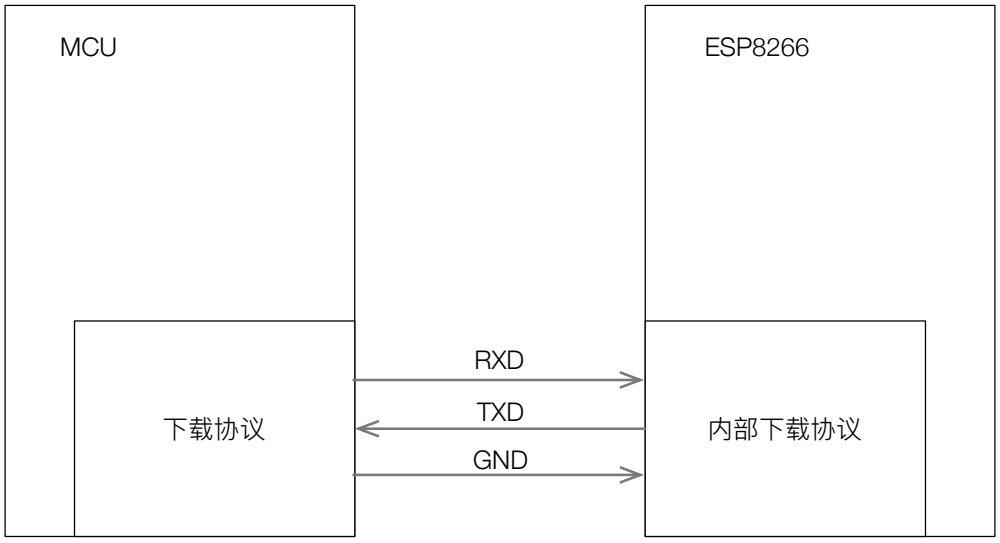

图 1-1. 硬件连接

1.2. 下载流程

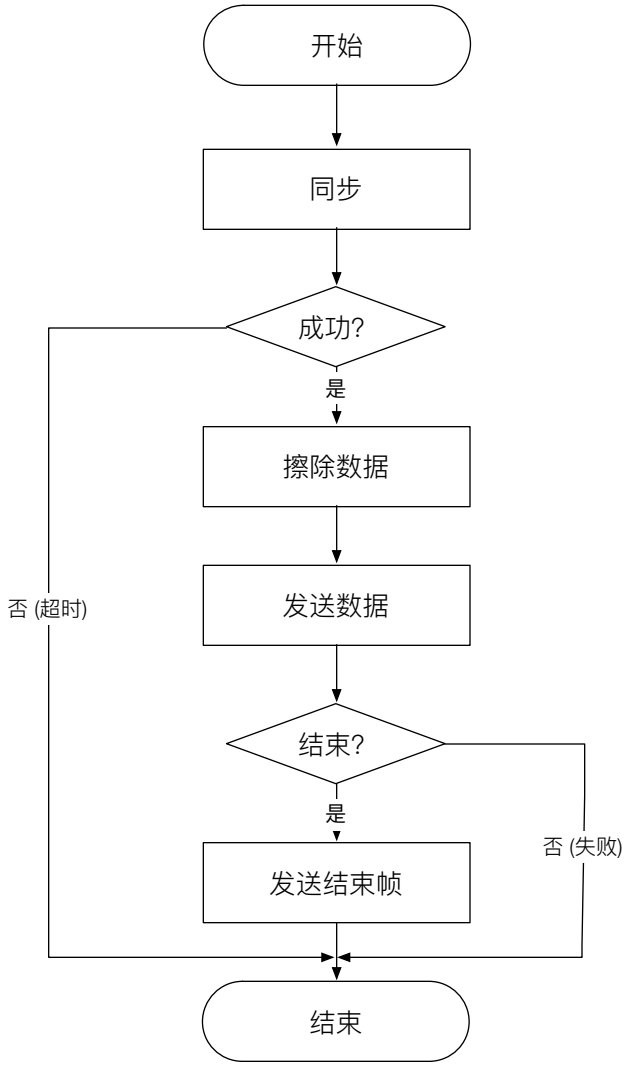

图 1-2. 下载流程

- 同步：发送同步帧同步波特率。
- 擦除数据：根据要下载的固件大小和下载地址擦除 Flash 相应的块区域。
- 发送数据：将固件封装成多帧发送给 ESP8266。
- 发送结束帧：发送下载结束帧给 ESP8266。

2. 传输协议

传输协议采用 *SLIP*（Serial Line Internet Protocol）的封装格式。

- 每个数据包都以 `0xC0` 开始和结束。
- 如果 `0xC0` 出现在数据包内部，就将 `0xC0` 替换成两个字节 `0xDB 0xDC`；如果 `0xDB` 出现在数据包内部，则替换为 `0xDB 0xDD`。
- 在数据帧里，数据包由数据头和长度不定的数据体组成，如图 2-1 所示。
- 所有多字节字段的存储模式均为小端模式。

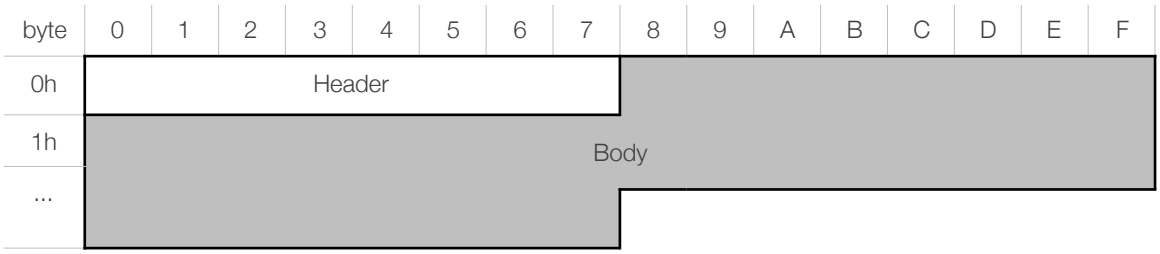

图 2-1. 数据包格式

> 📖 **说明：**
> 数据头 *Data size* 统计的是不替换前的长度。

2.1. 数据头

数据头的格式如表 2-1 所示。

表 2-1. 数据头格式

数据类型	字节	请求	应答
Type	0	始终为 `0x00`。	始终为 `0x01`。
Command	1	操作代码详细信息请参考表 2-2。	
Data size	2~3	数据体的大小。	

数据类型	字节	请求	应答
Checksum/ Response	4~7	payload（数据体中 16 字节之后的固件数据）的异或校验。 Checksum 计算方法请参考“附录 – 编程举例”。	响应数据。
Body	8~n	取决于操作。	
Status	8	-	状态标志，成功（0）或失败（1）。
Error	9	-	成功（null）或失败（错误码）。

表 2-2. 操作代码

代码	名称	说明
02	Flash DownLoad Start	擦除 Flash 中的数据。 • **Word0**：擦除扇区的数量，每个扇区 4096 字节。 • **Word1**：发送数据包的数量。 • **Word2**：发送数据包的大小，如 `0x400`。 • **Word3**：偏移地址。 **说明：** 关于擦除数据的代码示例请参考“附录 – 编程举例”。
03	File Packet Send	发送数据。 • **Word0**：发送数据包的大小（填 `0x400`）。 • **Word1**：发送数据包的序列号。 • **Word2**：`0x0` • **Word3**：`0x0`
04	Flash DownLoad Stop	停止发送数据。

08	Sync Frame Send	`sync_frame[36] = { 0x07, 0x07, 0x12, 0x20,`
		`0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55,`
		`0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55,`
		`0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55,`
		`0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55, 0x55 };`

2.2. 数据体

数据体格式如图 2-2 所示。

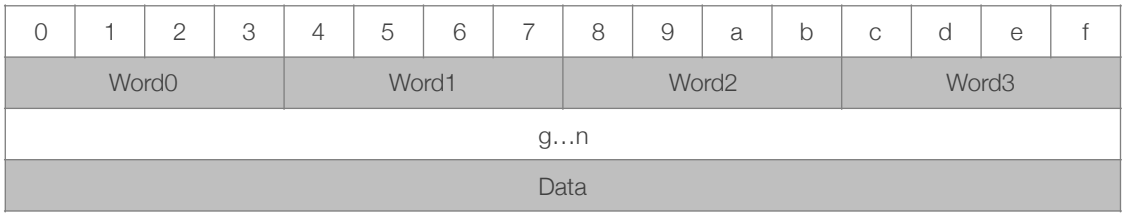

图 2-2. 数据体格式

数据体前 16 个字节（Word0~Word3）是描述数据体的，不同的数据指令，其描述也不同。

3. 固件格式

固件包含文件头和数目可变的数据块（数据块的大小可能不同），如图 3-1 所示。所有多字节字段的存储模式均为小端模式。

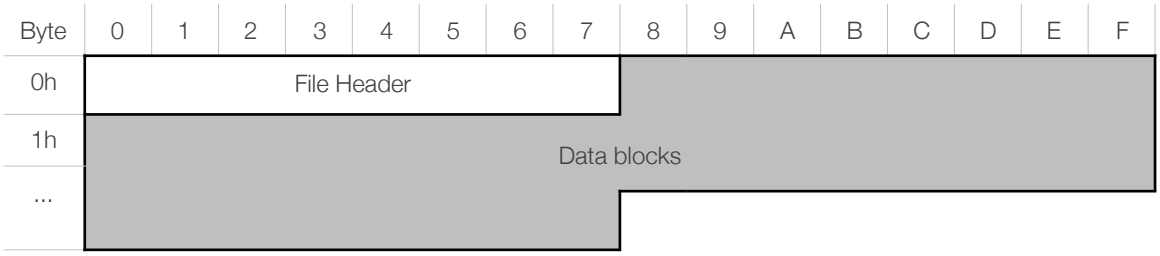

图 3-1. 固件格式

文件头格式如表 3-1 所示。

表 3-1. 固件格式说明

字节	数据类型	说明
0	Magic Code	值始终为 `0XE9`。
1	Block Number	数据块的数量。
2	SPI Mode	SPI 的工作模式。 • `0x00`：QIO 模式 • `0x01`：QOut 模式 • `0x02`：DIO 模式 • `0x03`：DOut 模式
3	SPI Flash Info	SPI Flash 的大小和频率。 高 4 位：`0x0` = 512 kB；`0x1` = 256 kB；`0x2` = 1 MB；`0x3` = 2 MB；`0x4` = 4 MB 低 4 位：`0x0` = 40 MHz；`0x1` = 26 MHz；`0x2` = 20 MHz；`0XF` = 80 MHz
4~7	Entry Address	CPU 入口地址。

I. 附录 – 编程举例

I.I. checksum

```
uint32_t espcomm_calc_checksum(unsigned char *data, uint16_t
data_size)
{
    uint16_t cnt;
    uint32_t result;
    result = 0xEF;
    for(cnt = 0; cnt < data_size; cnt++)
    {
        result ^= data[cnt];
    }
    return result;
}
```

I.II. erase flash

```
#define  BLOCKSIZE_FLASH  0x400
#define  FLASH_DOWNLOAD_BEGIN  0x02
uint32 flash_packet[];
//uint32_t size:firmware real size,  uint32_t address: download
offset address
int erase_flash(uint32_t size, uint32_t address)
{
const int sector_size = 4096;
const int sectors_per_block  = 16;
const int first_sector_index = address / sector_size;
const int total_sector_count = ((size % sector_size) == 0) ?
                                (size / sector_size) : (size /
sector_size + 1);
const int max_head_sector_count  = sectors_per_block -
(first_sector_index % sectors_per_block);
const int head_sector_count = (max_head_sector_count >
total_sector_count) ?
```

```
                                    total_sector_count :
max_head_sector_count;
// SPIEraseArea function in the esp8266 ROM has a bug which causes
extra area to be erased.
// If the address range to be erased crosses the block boundary,
// then extra head_sector_count sectors are erased.
// If the address range doesn't cross the block boundary,
// then extra total_sector_count sectors are erased.
const int adjusted_sector_count = (total_sector_count > 2 *
head_sector_count) ?
                                    (total_sector_count -
head_sector_count):
                                    (total_sector_count + 1) / 2;
erase_size = adjusted_sector_count * sector_size;


flash_packet[0] = erase_size;
flash_packet[1] = (size + BLOCKSIZE_FLASH - 1) / BLOCKSIZE_FLASH;
flash_packet[2] = BLOCKSIZE_FLASH;
flash_packet[3] = address;
espcomm_send_command(FLASH_DOWNLOAD_BEGIN, (unsigned char*)
&flash_packet, 16);
}
```

I.III. 参考资料

(1) igrr/esptool-ck url：*https://github.com/igrr/esptool-ck*

(2) themadinventor/esptool url：*https://github.com/themadinventor/esptool*

乐鑫 IOT 团队

www.espressif.com