



# ESP8266 SPI模块使用说明

**Version 0.1**

**Espressif Systems IOT Team  
Copyright (c) 2015**



## 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。



# Table of Contents

1.	功能综述 .....	4
2.	ESP8266 SPI主机协议格式.....	4
2.1.	SPI主机支持的通信格式 .....	4
2.2.	现有API支持的SPI主机通信格式 .....	4
3.	ESP8266 SPI从机协议格式.....	4
3.1.	SPI从机时钟极性配置要求.....	4
3.2.	SPI从机支持的通信格式 .....	4
3.3.	SPI从机支持命令定义.....	5
3.4.	现有API支持的SPI从机通信格式 .....	5
4.	SPI模块API函数说明 .....	5
4.1.	SPI主机API函数说明 .....	5
4.2.	SPI从机API函数说明 .....	7



## 1. 功能综述

ESP8266 SPI模块用于与各种支持SPI协议的设备进行通信，在电气接口方面，支持SPI协议标准的4线通信（CS, SCLK, MOSI, MISO）。与普通SPI模块不同的是，ESP8266 SPI模块对SPI接口的FLASH存储器做了特殊的支持。因此ESP8266 SPI模块的主机与从机模式都有着相应的固定硬件控制协议，与之通信的SPI设备需要做对应的匹配处理。

## 2. ESP8266 SPI主机协议格式

### 2.1. SPI主机支持的通信格式

ESP8266SPI主机通信格式为命令+地址+读/写数据，具体为：

- (1) 命令：必须存在；长度，1-16bits；主机输出从机输入（MOSI）。
- (2) 地址：可选；长度，0-32bits；主机输出从机输入（MOSI）。
- (3) 数据写或读：可选；长度，0-512bits(64bytes)；主机输出从机输入（MOSI）或主机输入从机输出（MISO）。

### 2.2. 现有API支持的SPI主机通信格式

ESP8266 SPI的API函数中给出两个固定的主机初始化模式，一个模式支持大多数以字节单位的常规SPI通信，另一个模式专为驱动一种彩色LCD屏设计，该设备需要一次9位的非标准SPI通信格式，详细叙述参见4.1节。

## 3. ESP8266 SPI从机协议格式

### 3.1. SPI从机时钟极性配置要求

与ESP8266 SPI从机通信的主机设备时钟极性需配置为：空闲低电平，上升沿采样，下降沿变换数据。并且在一次16位读/写过程中，务必保持片选信号CS的低电平，如果在发送过程中CS被拉高，从机内部状态将会重置。

### 3.2. SPI从机支持的通信格式

ESP8266SPI从机通信格式与主机模式基本相同为命令+地址+读/写数据，而从机读写操作有固定硬件命令，且地址部分不能去除，具体为：

- (1) 命令：必须存在；长度，3-16bits；主机输出从机输入（MOSI）。
- (2) 地址：必须存在；长度，1-32bits；主机输出从机输入（MOSI）。



(3) 数据写或读：可选；长度，0-512bits(64bytes)；主机输出从机输入（MOSI）或主机输入从机输出（MISO）。

### 3.3. SPI从机支持命令定义

从机接收命令长度至少是3位且低3位对应有固定的硬件读写操作，具体为：

(1) 010（从机接收）：将主机发送数据通过MOSI写入从机数据缓存对应寄存器SPI\_FLASH\_C0至SPI\_FLASH\_C15。

(2) 011（从机发送）：将从机缓存对应寄存器SPI\_FLASH\_C0至SPI\_FLASH\_C15中的数据通过MISO发送到主机。

(3) 110（从机同时收发）：将从机数据缓存发送至MISO同时将MOSI上的主机数据写入数据缓存SPI\_FLASH\_C0至SPI\_FLASH\_C15。

(4) 注意：其余数值用于读写SPI从机的状态寄存器SPI\_FLASH\_STATUS，由于其通信格式与读写数据缓存不同，会造成从机读写错误，请勿使用。

### 3.4. 现有API支持的SPI从机通信格式

ESP8266 SPI的API函数中给出一个固定的从机初始化模式，该模式为兼容大多数以字节为单位的设备，将从机通信格式设定为：7bits命令+1bit地址+8bits读/写数据。这样其他SPI主机设备进行一次16bits（或两次8bits且CS需要保持低电平）的通信就可以对ESP8266的SPI从机进行一字节的读或写操作。详细参见 4.2节。

## 4. SPI模块API函数说明

### 4.1. SPI主机API函数说明

#### (1) void spi\_lcd\_mode\_init(uint8 spi\_no)

功能：为驱动TM035PDZV36彩色液晶屏提供的SPI主机初始化程序。

参数说明：

uint8 spi\_no——所使用SPI模块号，只能输入SPI（0）与HSPI（1）其他输入无效。

#### (2) void spi\_lcd\_9bit\_write(uint8 spi\_no,uint8 high\_bit,uint8 low\_8bit)

功能：为驱动TM035PDZV36彩色液晶屏提供的SPI主机发送函数，该液晶模块需要一次需要9位传输。

参数说明：

uint8 spi\_no——所使用SPI模块号，只能输入SPI与HSPI其他输入无效。

uint8 high\_bit——第9位数据，0代表第九位为0，其余数据都表示第9位为1



uint8 low 8bit——低8位数据

**(3) void spi\_master\_init(uint8 spi\_no)**

功能：常规SPI主机初始化函数，波特率为CPU时钟的4分频，初始化后可以使用除 spi\_lcd\_9bit\_write 以外的所有主机函数。

### 参数说明：

uint8 spi no——所使用SPI模块号，只能输入SPI与HSPI其他输入无效。

**(4) void spi\_mast\_byte\_write(uint8 spi\_no,uint8 data)**

功能：完成基本的一字节的主机发送。

### 参数说明:

uint8 spi\_no——所使用SPI模块号，只能输入SPI与HSPI其他输入无效。

uint8 data——8位发送数据

**(5) void spi\_byte\_write\_espslave(uint8 spi\_no,uint8 data)**

**功能：**对ESP8266的SPI从机写入一字节数据。

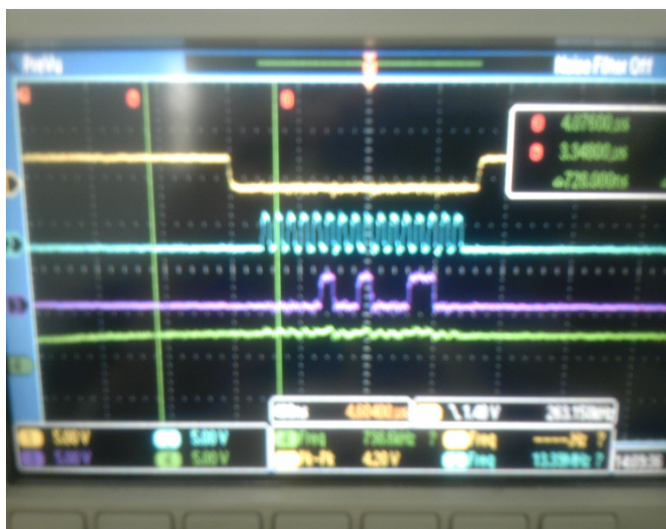
由于从机设置为：7bits命令+1bit地址+8bits数据，因此完成发送需要一次16bits的传输并且第一字节的固定为0b00000010+0（原理参见3.3）即0x04，第二字节为发送数据data。实际发送波形如图1所示。

参数说明：

uint8 spi no——所使用SPI模块号，只能输入SPI与HSPI其他输入无效。

uint8 data——8位发送数据

图1 spi byte write espslave写入ESP8266从机波形



黄线CS, 蓝线CLK, 红线MOSI, 绿线MISO



#### (6) void spi\_byte\_read\_espslave(uint8 spi\_no, uint8 \*data)

功能：对ESP8266的SPI从机读取一字节数据，也可以读取其他SPI从机设备。

由于ESP8266从机设置为：7bits命令+1bit地址+8bits数据，因此完成发送需要一次16bits的传输并且第一字节的固定为0b0000011+0（原理参见3.3）即0x06，第二字节为接收数据。

实际运行波形如图2所示。

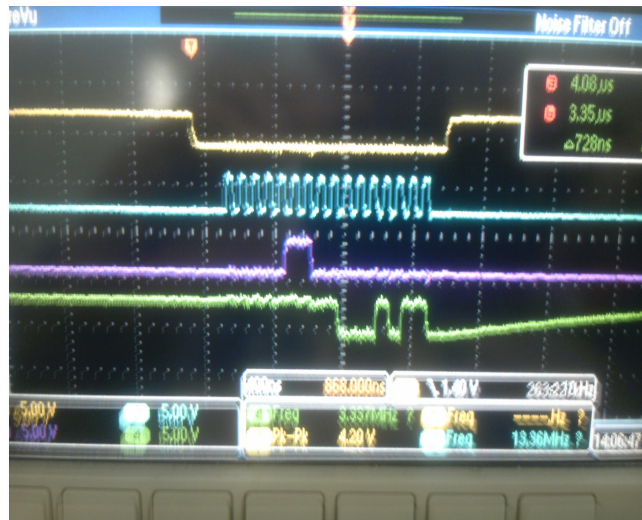
对于其他全双工SPI从机设备，需要设置从机完成一次16bits的通信，并将有效数据放置在从机发送缓存的第二个字节，该字节会被ESP8266主机接收。

参数说明：

uint8 spi\_no——所使用SPI模块号，只能输入SPI与HSPI其他输入无效。

uint8\* data——8位接收数据所在内存地址

图2 spi\_byte\_read\_espslave读取ESP8266从机波形



黄线CS，蓝线CLK，红线MOSI，绿线MISO

## 4.2. SPI主机API函数说明

### (1) void spi\_slave\_init(uint8 spi\_no)

功能：SPI从机模式初始化，将IO口配置为SPI模式，启用SPI传输中断，并注册 spi\_slave\_isr\_handler 函数。

通信格式设定为 7bits命令+1bit地址+8bits读/写数据。命令与地址组成高8位，且地址位必须为0，根据3.3节叙述，支持：0x04主机写从机读，0x06主机读从机写，0x0c主从同时读写，三个主机命令。

通信波形参见图1，2。

参数：

spi\_no: SPI模块的序号，ESP8266处理器有两组功能相同的SPI模块，分别为SPI和HSPI





可选配的值: SPI或HSPI

## (2) spi\_slave\_isr\_handler(void \*para)

功能与触发条件: spi中断处理函数, 主机如正确进行了传输操作(读或写从机), 中断就会触发。

代码:

```
//0x3ff00020 is isr flag register, bit4 is for spi isr,
if(READ_PERI_REG(0x3ff00020)&BIT4){
    //following 3 lines is to close spi isr enable
    regvalue=READ_PERI_REG(SPI_FLASH_SLAVE(SPI));
    regvalue&=~(0x3ff);
    WRITE_PERI_REG(SPI_FLASH_SLAVE(SPI),regvalue);
    //os_printf("SPI ISR is trigged\n"); //debug code
}else if(READ_PERI_REG(0x3ff00020)&BIT7){ //bit7 is for hspi isr,
    //following 3 lines is to clear hspi isr signal
    regvalue=READ_PERI_REG(SPI_FLASH_SLAVE(HSPI));
    regvalue&=~(0x1f);
    WRITE_PERI_REG(SPI_FLASH_SLAVE(HSPI),regvalue);
    //when master command is write slave 0x04,
    //recieved data will be occur in register SPI_FLASH_C0's low 8 bit,
    //also if master command is read slave 0x06,
    //the low 8bit data in register SPI_FLASH_C0 will transmit to master,
    //so prepare the transmit data in SPI_FLASH_C0' low 8bit,
    //if a slave transmission needs
    recv_data=(uint8)READ_PERI_REG(SPI_FLASH_C0(HSPI));
    /*put user code here*/
    //    os_printf("recv data is %08x\n", recv_data); //debug code
}else if(READ_PERI_REG(0x3ff00020)&BIT9){ //bit9 is for i2s isr,
}
```

代码说明: 由于SPI用于读写程序存储flash芯片, 因此使用HSPI进行通信。对于ESP8266处理器而言应该有多个设备公用此中断函数, 包括SPI模块, HSPI模块, I2S模块, 寄存器 0x3ff00020的第4位、第7位、第9位分别对应。

SPI模块会频繁触发传输中断, 因此需要关闭其5个中断源使能, 对应代码如下:





```
regvalue=READ_PERI_REG(SPI_FLASH_SLAVE(SPI));  
regvalue&=~(0x3ff);  
WRITE_PERI_REG(SPI_FLASH_SLAVE(SPI),regvalue);
```

而在HSPI触发的情况下，需要软件清零其5个中断源以避免反复触发中断函数。代码对应如下：

```
regvalue=READ_PERI_REG(SPI_FLASH_SLAVE(HSPI));  
regvalue&=~(0x1f);  
WRITE_PERI_REG(SPI_FLASH_SLAVE(HSPI),regvalue);
```

而接收和发送数据都共用一个寄存器SPI\_FLASH\_C0。其中读出寄存器对应代码：

```
recv_data=(uint8)READ_PERI_REG(SPI_FLASH_C0(HSPI));
```

recv\_data为全局变量。在该语句之后可以加入用户自定义处理程序。

注意：中断程序不适宜执行时间过长的处理代码，因为长时间的中断程序会使看门狗定时器无法正常清零，造成处理器意外重启。