ESP8266 SPI透传协议（单线）

Version 0.1

Espressif Systems IOT Team

Table of Contents

1. 功能综述

该协议使用ESP8266的从机模式与其他的处理器的SPI主机进行通信，连线上需要5路信号线实现该协议，除了标准SPI所需要的4路信号线外，还需要额外的1路信号用于告知主机当前从机的状态寄存器状态更新情况。

2. ESP8266 SPI从机协议格式

2.1. SPI从机时钟极性配置要求

与ESP8266SPI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34字节读/写或通过一次两字节的通信读取丛机状态寄存器的过程中，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。

2.2. SPI从机支持的通信格式

ESP8266SPI从机通信格式为命令+地址+读/写数据或命令+丛机状态值，具体：

(1) 命令：长度，8bits；主机输出从机输入（MOSI）。

其中0x02为主机发送从机接收数据，主机通过MOSI将32bytes写入从机数据缓存对应寄存器SPI_W0至SPI_W7；

而0x03为主机接收从机发送数据，将从机缓存对应寄存器SPI_FLASH_C8至SPI_FLASH_C15中的32bytes数据通过MISO发送到主机。

另外，0x04或0x05均可读取从机状态寄存器SPI_FLASH_STATUS中的低8位。

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。

(2) 地址：长度，8bits；主机输出从机输入（MOSI）。地址内容必须为0。

(3) 读/写数据：长度，256bits(32bytes)；主机输出从机输入（MOSI）对应0x02 命令或主机输入从机输出（MISO）对应0x03命令。

(4) 从机状态：长度，8bits；主机输入从机输出（MISO），使用0x04或0x05读取表示从机通信状态。

3. 从机状态定义与中断线行为

3.1. 状态定义

从机状态一共有8bits其中：

(1) wr_busy，bit0：1表示从机写缓存满，并正在处理接收数据，0表示写缓存空可以进行下一次写入操作。

(2) rd_empty，bit1：1表示从机读缓存为空，没有新数据更新，0表示读缓存已更新需要主机读取。

(3) comm_cnt，bit2-4：读写通信计数。每次从机进去SPI读/写缓存中断时，该3位计数值会加1，主机可以由此判断一次读/写数据通信是否已经被从机识别并通信完毕。

因此主机在一次读/写数据通信后，如果想要进行下一次读操作必须满足：rd_empty为0，并且comm_cnt的值为上一次通信时加1；如果想要进行下一次写操作必须满足：wr_busy为0，并且comm_cnt的值为上一次通信时加1。

3.2. GPIO0中断线行为

在从机状态寄存器产生变化时，中断线GPIO0会置1，当主机使用0x04,0x05命令读取从机状态寄存器后，中断线GPIO0会清0。

4. ESP8266 SPI从机API函数说明

注意：如果需要使用SPI带状态寄存器的单线透传协议需要在spi.h文件中配置：

```
//SPI protocol selection
#define TWO_INTR_LINE_PROTOCOL      0
#define ONE_INTR_LINE_31BYTES       0
#define ONE_INTR_LINE_WITH_STATUS   1
```

中断响应函数会采用spi_slave_isr_sta(void *para)

(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 8bits命令+8bit地址+256bits(32bytes)读/写数据。

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_sta(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。用户可以修改中断服务程序实现所需通信功能，代码如下：

```
struct spi_slave_status_element
{
 uint8 wr_busy:1;
 uint8 rd_empty :1;
 uint8 comm_cnt :3;
 uint8 res :3;
};

union spi_slave_status
{
 struct spi_slave_status_element elm_value;
 uint8 byte_value;
};
void spi_slave_isr_sta(void *para)
{
 uint32 regvalue,calvalue;
 uint32 recv_data,send_data;
 union spi_slave_status spi_sta;

 if(READ_PERI_REG(0x3ff00020)&BIT4){
      //following 3 lines is to clear isr signal
          CLEAR_PERI_REG_MASK(SPI_SLAVE(SPI), 0x3ff);
          }else if(READ_PERI_REG(0x3ff00020)&BIT7){ //bit7 is for hspi isr,
          //记录中断状态
          regvalue=READ_PERI_REG(SPI_SLAVE(HSPI));
          //***********处理中断标志结束本次通行过程************//
          CLEAR_PERI_REG_MASK(SPI_SLAVE(HSPI),
                                                SPI_TRANS_DONE_EN|
                                                SPI_SLV_WR_STA_DONE_EN|
                                                SPI_SLV_RD_STA_DONE_EN|
                                                SPI_SLV_WR_BUF_DONE_EN|
                                                SPI_SLV_RD_BUF_DONE_EN);
```

```
    SET_PERI_REG_MASK(SPI_SLAVE(HSPI), SPI_SYNC_RESET);
    CLEAR_PERI_REG_MASK(SPI_SLAVE(HSPI),
                                                SPI_TRANS_DONE|
                                                SPI_SLV_WR_STA_DONE|
                                                SPI_SLV_RD_STA_DONE|
                                                SPI_SLV_WR_BUF_DONE|
                                                SPI_SLV_RD_BUF_DONE);
    SET_PERI_REG_MASK(SPI_SLAVE(HSPI),
                                                SPI_TRANS_DONE_EN|
                                                SPI_SLV_WR_STA_DONE_EN|
                                                SPI_SLV_RD_STA_DONE_EN|
                                                SPI_SLV_WR_BUF_DONE_EN|
                                                SPI_SLV_RD_BUF_DONE_EN);
    //*******************************************************//
    /****************主机写中断处理***************/
     if(regvalue&SPI_SLV_WR_BUF_DONE){
     //*****写入完成，写入忙状态位置1，通信计数器加1****//
            spi_sta.byte_value=READ_PERI_REG(SPI_STATUS(HSPI))&0xff;
            spi_sta.elm_value.wr_busy=1;
            spi_sta.elm_value.comm_cnt++;
            WRITE_PERI_REG(SPI_STATUS(HSPI), (uint32)spi_sta.byte_value);
     //*********************************************//
            //*******将寄存器接收数据搬入内存******//
            idx=0;
            while(idx<8){
                    recv_data=READ_PERI_REG(SPI_W0(HSPI)+(idx<<2));
                    //os_printf("rcv data : 0x%x \n\r",recv_data);
                    spi_data[idx<<2] = recv_data&0xff;
                    spi_data[(idx<<2)+1] = (recv_data>>8)&0xff;
                    spi_data[(idx<<2)+2] = (recv_data>>16)&0xff;
                    spi_data[(idx<<2)+3] = (recv_data>>24)&0xff;
                    idx++;
            }
            //*********************************//
            //************数据搬完，清0写入忙状态*************//
```

```
            spi_sta.byte_value=READ_PERI_REG(SPI_STATUS(HSPI))&0xff;
            spi_sta.elm_value.wr_busy=0;
            WRITE_PERI_REG(SPI_STATUS(HSPI), (uint32)spi_sta.byte_value);
            //*************************************************//
            /***测试部分，可以修改，该段程序作用是将读到数据复制到读缓存**/
            for(idx=0;idx<8;idx++)
            {
                WRITE_PERI_REG(SPI_W8(HSPI)+(idx<<2),
                    READ_PERI_REG(SPI_W0(HSPI)+(idx<<2)));
            }
            /*************************************************************/
            /***测试部分，可以修改，读缓存空状态清0，从机可以进行读取操作**/
            spi_sta.byte_value=READ_PERI_REG(SPI_STATUS(HSPI))&0xff;
            spi_sta.elm_value.rd_empty=0;
            WRITE_PERI_REG(SPI_STATUS(HSPI), (uint32)spi_sta.byte_value);
            /*****************************************/
            GPIO_OUTPUT_SET(0, 1);    //中断线置1，提醒主机读取从机状态
        /****************主机读中断处理***************/
        }else if(regvalue&SPI_SLV_RD_BUF_DONE){
            //*****读取完成，读取空状态位置1，通信计数器加1****//
            spi_sta.byte_value=READ_PERI_REG(SPI_STATUS(HSPI))&0xff;
            spi_sta.elm_value.comm_cnt++;
            spi_sta.elm_value.rd_empty=1;
            WRITE_PERI_REG(SPI_STATUS(HSPI), (uint32)spi_sta.byte_value);

            GPIO_OUTPUT_SET(0, 1);    //中断线置1，提醒主机读取从机状态
        }
        /****************主机读状态中断处理***************/
        if(regvalue&SPI_SLV_RD_STA_DONE){
            GPIO_OUTPUT_SET(0,0);    //中断线清0，主机读取状态完毕
        }
    }else if(READ_PERI_REG(0x3ff00020)&BIT9){ //bit7 is for i2s isr,

    }
}
```