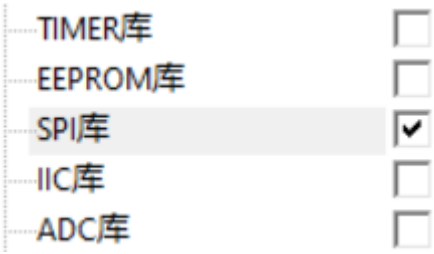
SPI库

SPI是串行外设接口（Serial Peripheral Interface）的缩写，SPI库就是关于单片机的SPI的操作库。在使用本库之前先到ecbm_core.h里使能。双击打开ecbm_core.h文件，然后进入图形化配置界面，使能SPI库。



API

spi_init

函数原型：void spi_init(void);

描述

SPI初始化函数。

输入

无

输出

无

返回值

无

参数配置

本函数初始化的参数都是由图形化配置界面来设置，双击打开spi.h文件，进入图形化配置界面。

设置说明如下：

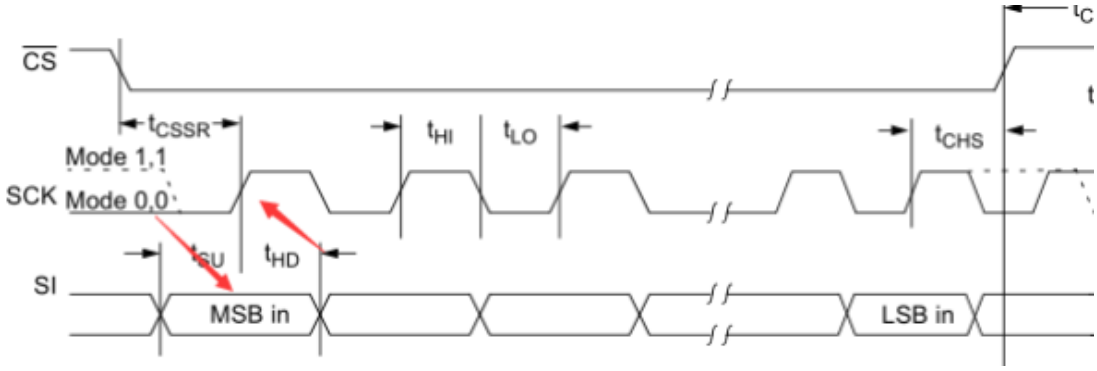
- 主/从机：用于设置SPI的是主机模式还是从机模式。目前的库只有主机发送接收函数，从机的数据处理需要自己实现了。
- SS引脚使能：这个就是SPI从机模式片选脚，如果使能的话，当单片机的该脚被拉低时，无论之前设置的是主机还是从机，都会被强制设置成从机。因此如果只当主机的话，就不要使能该功能。

- 数据收发顺序：可以先发高位或者先发低位。比如要发送0x29（二进制为0010 1001），先发高位就是按照00101001的顺序发，先发低位就是按照10010100的顺序发。选择哪个需要参考目标器件的数据手册。
- SPI时钟：通过系统时钟分频得到，分频数越小，SPI的时钟越快。但是STC8的引脚输出速度也不算太快，所以SPI的时钟也不是越快就越好，太快的话IO速度反应不过来，输出的数据就会出错。
- SPI时钟极性控制：原文标准化的描述不容易理解，我翻译一下，所谓极性就是在不通信的时候，SCLK脚为高电平还是低电平。
- SPI时钟相位控制：相位这个词一出来，说不定就会想到数学里的波形相位。但是SPI的相位没那么复杂。下面的小科普会着重说明相位和极性的关系。
- SPI输出管脚：按照选项内容和实际需求选择即可。注意选项括号里的提示，有些型号的引脚可能会不同。

小科普

在一般的SPI手册里，都会把两个极性和两个相位搭配的四种情况的时序图列出来。别说是新手了，我在会用SPI之后回来总结成库都会晕头转向的。接下来教大家一个方法：

- 先看器件是什么边沿驱动的。对于同步传输的协议来说，一定是靠时钟脚的边沿触发数据的发送或者接收，所以第一步就是看驱动的边沿是上升沿还是下降沿。好好对比下面两张图：一般而言，数据变化是需要一定时间稳定，所以就看数据的中间对应着是上升沿还是下降沿。所以不难看出图一就是上升沿，



图二就是下降沿。

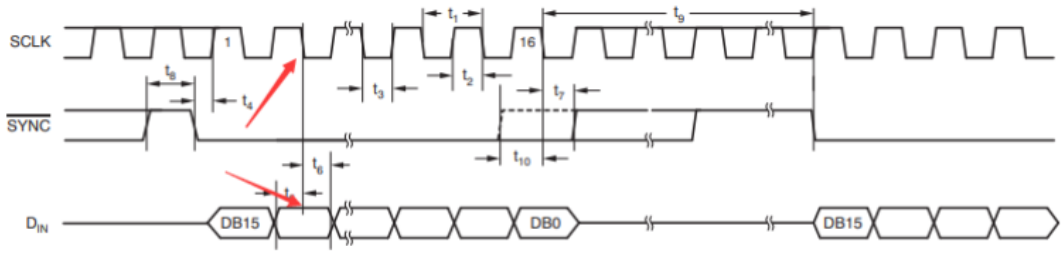


Figure 1. Serial Write Operation

- 然后就是选定时钟线的闲时电平，就是在不通信的时候，IO口保持的电平值。这个主要看器件手册的要求，但如果没有特殊要求，那么随便选一个就行。于是现在有两个属性是确定下来了：一个是触发边沿，一个是闲时电平。
- 基于你对闲时电平的选择，就能立马对选项【SPI时钟极性控制】做出选择！
- 到这里，应该就快理解相位了吧。假如你选择了闲时电平为低电平。而器件要求的触发边沿为下降沿，那么好好想一想：一个低电平的后面是不可能有下降沿的对吧，因为下降沿的定义就是高电平跳转到低电平的那个瞬间。于是乎为了触发数据传送，就必须先拉高时钟线（这是空闲状态到工作状态的第一个边沿）然后再拉低产生下降沿（这是空闲状态到工作状态的第二个边沿）。看看括号里的提示，现在应该明白选项【SPI时钟相位控制】的意思了吧，这里就应该选择“在时钟变化的第二个边沿”。同理，如果闲时电平为高电平，器件要求下降沿触发。那么从空闲状态转到工作转态时，时钟线可以立马拉低产生下降沿，所以就选择“在时钟变化的第一个边沿”。

调用例程

```
#include "ecbm_core.h"//加载库函数的头文件。
void main(){//main函数，必须的。
    system_init();//系统初始化函数。
    spi_init();//初始化spi。
    while(1){
    }
}
```

注意事项

1. SPI的原理简单，但是细节很多，最好要对照手册确认模式都选择正确。

spi_set_pin

函数原型：void spi_set_pin(u8 group);

描述

SPI的引脚设置函数。

输入

- group：引脚所在的分组。

输出

无

返回值

无

分组定义

基本宏定义的名字就说明了SPI将会用到哪些IO了：

- SPI_PIN_P12_P13_P14_P15。
- SPI_PIN_P22_P23_P24_P25。
- SPI_PIN_P74_P75_P76_P77。
- SPI_PIN_P35_P34_P33_P32。
- SPI_PIN_P54_P40_P41_P43。
- SPI_PIN_P55_P54_P33_P32。
- SPI_PIN_P54_P13_P14_P15。

在调用之前，请确认当前的型号确实有这些脚。前期确认一遍，不会耽误太多时间。

调用例程

```
if(run_mode==1){//当运行模式为1的时候，
    spi_set_pin(SPI_PIN_P12_P13_P14_P15);//控制P1连接的SPI器件。
}else{//在其他模式下，
    spi_set_pin(SPI_PIN_P22_P23_P24_P25);//控制P2连接的SPI器件。
}
```

注意事项

无

spi_send

函数原型：u8 spi_send(u8 dat);

描述

SPI发送接收函数。

输入

- dat：要发送的数据。

输出

无

返回值

- 接收到的数据

调用例程

既发送也接收：

```
#include "ecbm_core.h"//加载库函数的头文件。
u8 dat_in,dat_out;//两个缓存。
void main(){//main函数，必须的。
    system_init();//系统初始化函数。
    spi_init();//初始化spi。
    while(1){
        if(RI){//当接收到串口信息时。
            RI=0;//清除接收标志位。
            dat_in=SBUF;//把串口收到的数据保存下来。
            dat_out=spi_send(dat_in);//发送该数据，同时接收SPI返回的数据。
            SBUF=dat_out;//将SPI返回的数据发送到串口。
        }
    }
}
```

只发送：

```
#include "ecbm_core.h"//加载库函数的头文件。
void main(){//main函数，必须的。
    system_init();//系统初始化函数。
    spi_init();//初始化spi。
    while(1){
        spi_send(0x55);//发送该数据0x55。
        delay_ms(500);//每隔500mS发送一次。
    }
}
```

只接收：

```
#include "ecbm_core.h"//加载库函数的头文件。
u8 dat_out;//缓存
void main(){//main函数，必须的。
    system_init();//系统初始化函数。
    spi_init();//初始化spi。
    while(1){
        dat_out=spi_send(0xFF);//接收数据。
        delay_ms(500);//每隔500mS接收一次。
    }
}
```

注意事项

1. SPI协议就是有发送有接收，且发送和接收都是同时发生的。因此在只接收的情况下，也必须要发个0xFF才能接收到数据。

优化建议

本库比较简单，只有3个函数，所以没有可优化的地方