SPI接口

13

- SPI总线协议
- SPI控制器
- SPI应用实例

13.1 SPI总线协议

13.1.1 协议简介

SPI(Serial Peripheral Interface，串行外设接口)是一种高速的全双工、同步的通信总线。通过该总线，可以使微控制器(MCU)高速地与各种外围设备以串行方式进行通信。

SPI接口主要应用在Flash、E^2PROM、实时时钟、A/D转换器、数字信号处理器和数字信号解码器之间。

特点：

1. 采用主-从模式(Master-Slave)的控制方式。SPI规定了两个SPI设备之间通信必须由主设备(Master)来控制从设备(Slave)。
2. 采用同步方式(Synchronous)传输数据。主设备会根据将要交换的数据来产生相应的时钟脉冲，时钟脉冲组成了时钟信号。
3. 数据交换(Data Exchanges)。SPI设备间的数据传输之所以又被称为数据交换，是因为SPI协议规定一个SPI设备不能在数据通信过程中仅仅充当一个“发送者(Transmitter)”或者“接收者(Receiver)”。

13.1.2 协议内容

SPI接口的四种信号：

(1) MOSI—主器件数据输出，从器件数据输入；

(2) MISO—主器件数据输入，从器件数据输出；

(3) SPICLK—时钟信号，由主器件产生；

(4) CS—从器件使能信号，由主器件控制。

一个SPI时钟周期内，完成的操作：

1. 主机通过MOSI线发送1位数据，从机通过该线读取这1位数据。
2. 从机通过MISO线发送1位数据，主机通过该线读取这1位数据。

这些操作都是通过移位寄存器来实现的。主机和从机各有一个移位寄存器，且二者连接成环。随着时钟脉冲，数据按照从高位到低位的方式依次移出主机寄存器和从机寄存器，并且依次移入从机寄存器和主机寄存器。当寄存器中的内容全部移出时，相当于完成了两个寄存器内容的交换。

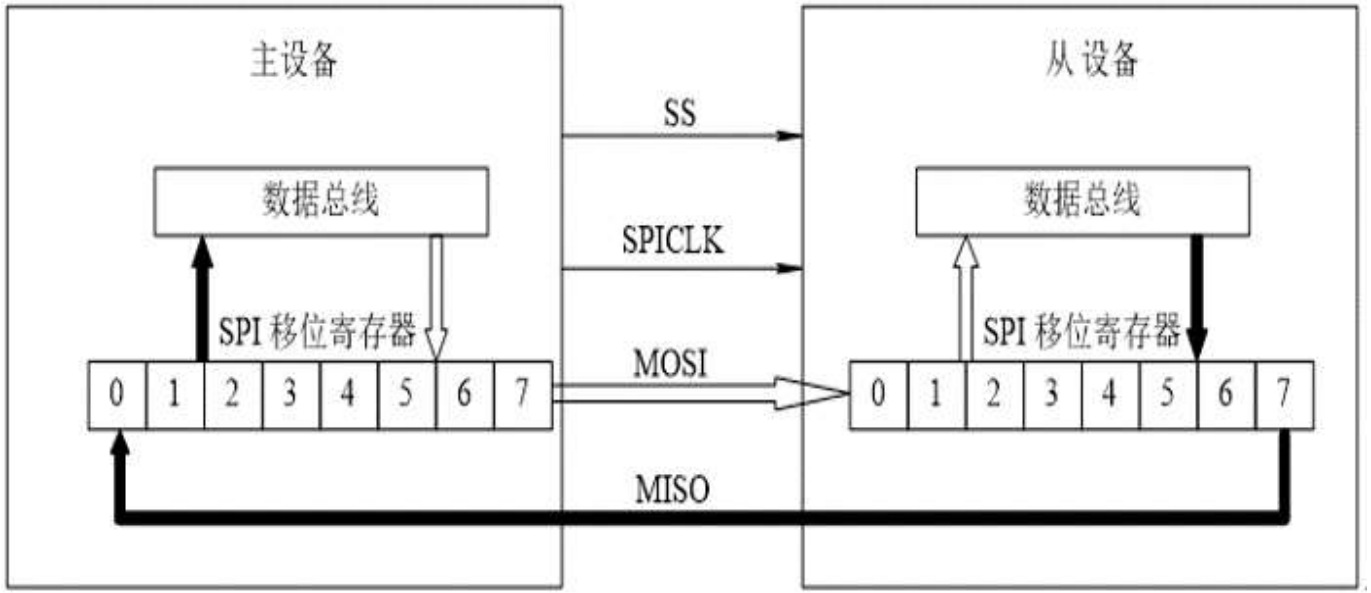
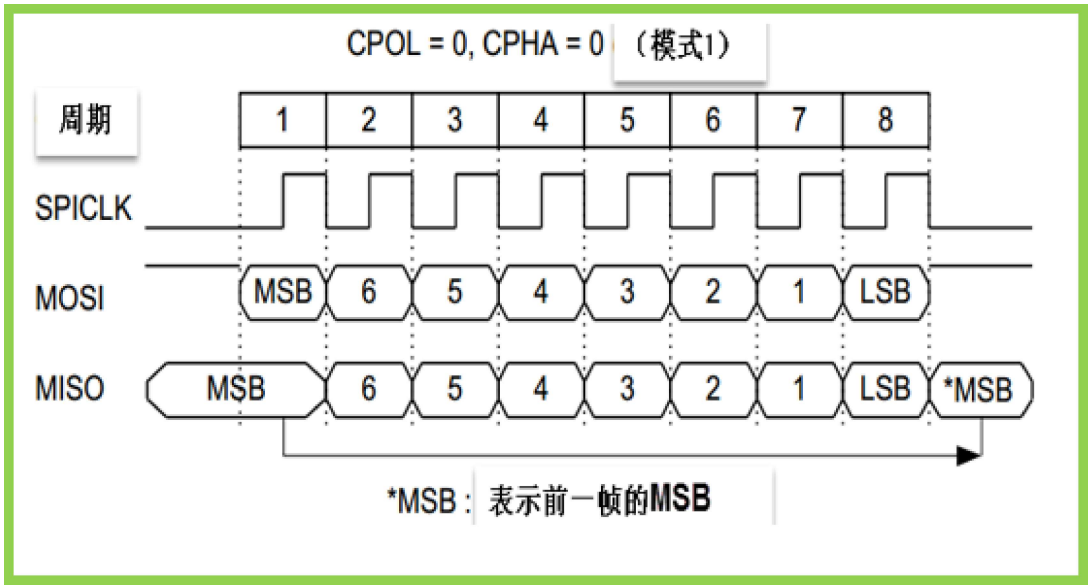
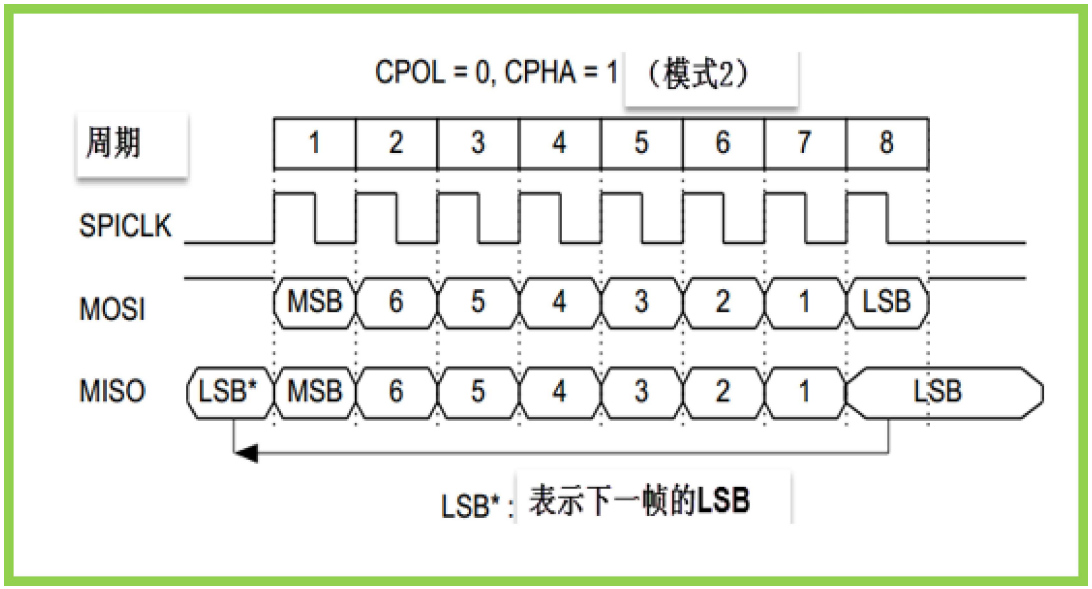


图13.1 SPI数据传输示意图

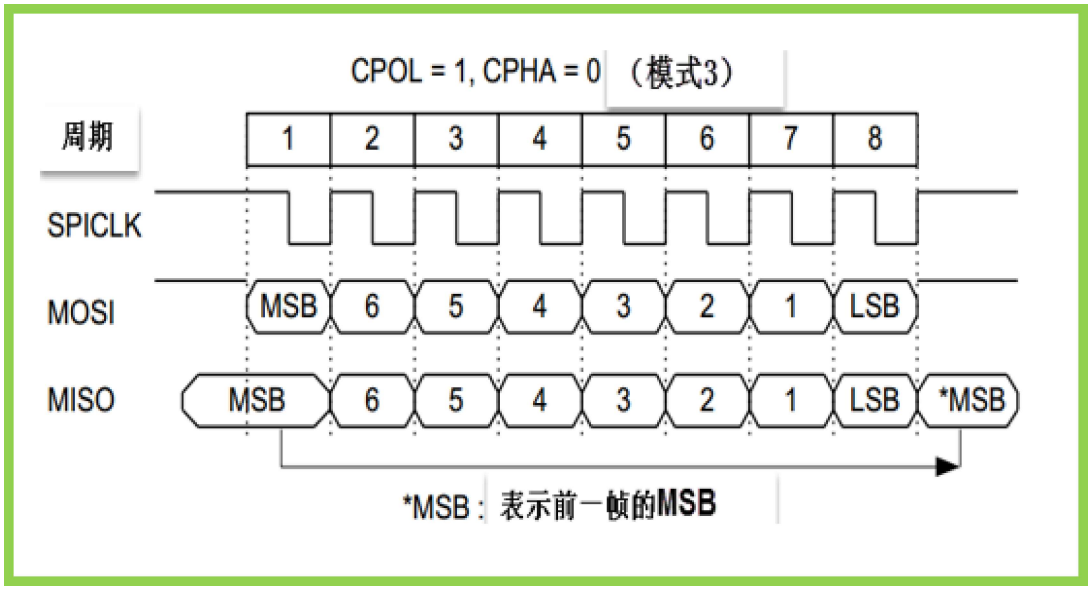
SPI的四种工作模式：由CPOL、CPHA两个控制位进行选择



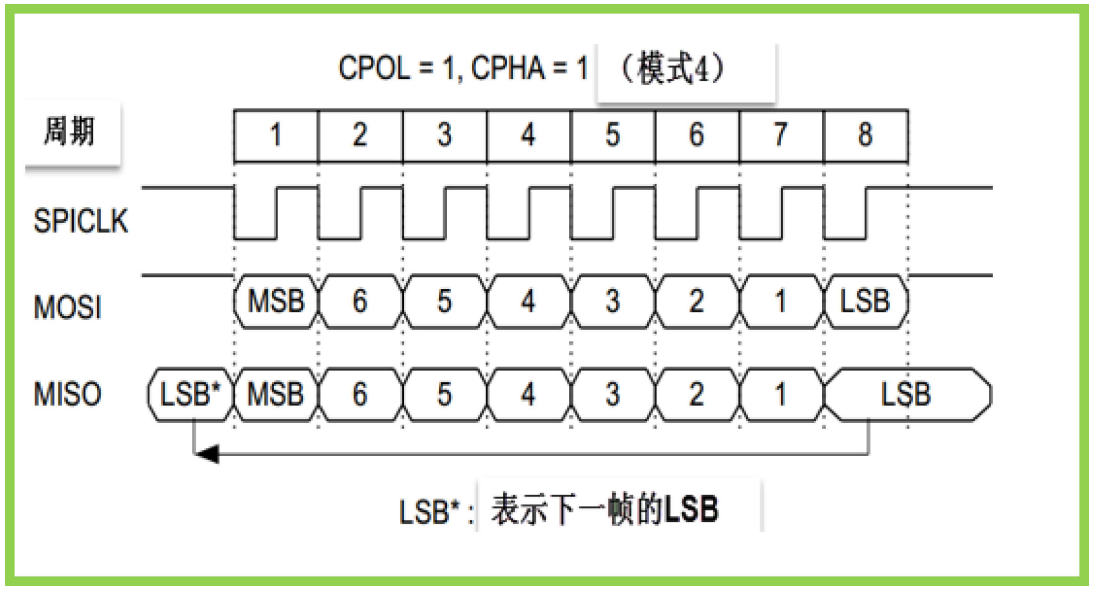
模式1


CPOL = 0, CPHA = 1 （模式2）
周期
1
2
3
4
5
6
7
8
SPICLK
MOSI
MSB
6
5
4
3
2
1
LSB
MISO
LSB*
MSB
6
5
4
3
2
1
LSB
LSB*：表示下一帧的LSB

模式2


CPOL = 1, CPHA = 0 （模式3）
周期
1
2
3
4
5
6
7
8
SPICLK
MOSI
MSB
6
5
4
3
2
1
LSB
MISO
MSB
6
5
4
3
2
1
LSB
*MSB
*MSB：表示前一帧的MSB

模式3


CPOL = 1, CPHA = 1 （模式4）
周期
1
2
3
4
5
6
7
8
SPICLK
MOSI
MSB
6
5
4
3
2
1
LSB
MISO
LSB*
MSB
6
5
4
3
2
1
LSB
LSB* : 表示下一帧的LSB

模式4

总结：

模式	CPOL 和 CPHA	第一个数据输出	其他位数据输出	数据采样
1	CPOL = 0，CPHA = 0	第一个 SPICLK 上升沿前	SPICLK 下降沿	SPICLK 上升沿
2	CPOL = 0，CPHA = 1	第一个 SPICLK 上升沿	SPICLK 上升沿	SPICLK 下降沿
3	CPOL = 1，CPHA = 0	第一个 SPICLK 下降沿前	SPICLK 上升沿	SPICLK 下降沿
4	CPOL = 1，CPHA = 1	第一个 SPICLK 下降沿	SPICLK 下降沿	SPICLK 上升沿

表13.1 SPI四种工作模式时序表

13.2 SPI控制器

13.2.1 Exynos 4412的SPI控制寄存器

该控制寄存器的主要特性如下：

- 全双工通信方式；
- 具有8/16/32位移位寄存器用于收/发；
- 支持8位、16位、32位总线接口；
- 支持摩托罗拉SPI协议和美国国家半导体公司的Microwire串行接口(SPI的精简接口)；

- 支持两个独立的收/发FIFO；
- 支持主机模式和从机模式；
- 支持无发送操作时接收数据操作；
- 最大收/发频率高达50 MHz。

13.2.2 时钟源选择

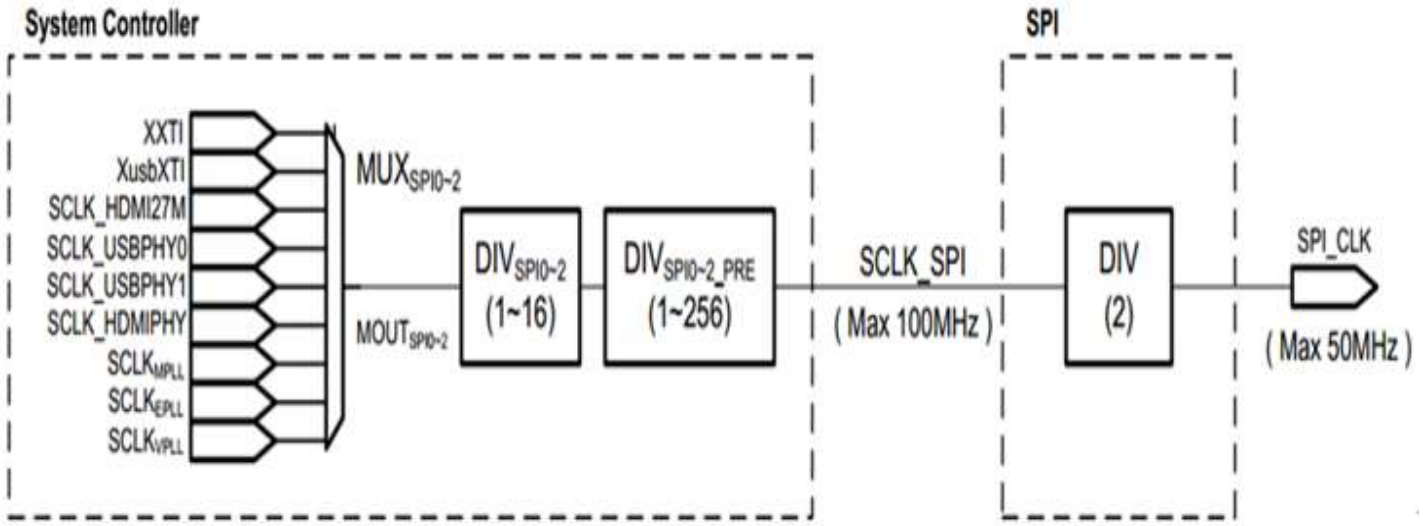


图13.3 SPI时钟源框图

Exynos 4412为SPI提供了9种不同的时钟源：XXTI、XusbXTI、SCLK_HDMI27M、SCLK_USBPHY0、SCLK_USBPHY1、SCLK_HDMIPHY、$SCLK_{MPLL}$、$SCLK_{EPLL}$、$SCLK_{VPLL}$，具体时钟源可参考第6章进行选择配置。选择好时钟源后，可以通过寄存器CLK_DIV_PERIL0和CLK_DIV_PERIL1的相关位域设置分频系数$DIV_{SPI0\sim2}$和$DIV_{SPI0\sim2_PRE}$进行两次分频，这样得到最大100 MHz的SCLK_SPI信号，再经2分频后，最终得到供SPI工作的时钟信号SPI_CLK。

13.2.3 SPI相关寄存器

寄存器名称	功　　能	复位值	寄存器名称	功　　能	复位值
CH_CFGn	配置 SPI 寄存器	0x0	SPI_RX_DATAn	存储接收到的数据	0x0
MODE_CFGn	控制 FIFO	0x0	PACKET_CNT_REGn	指定包的数目	0x0
CS_REGn	选择从设备	0x1	PENDING_CLR_REGn	清除中断挂起	0x0
SPI_INT_ENn	中断使能	0x0	SWAP_CFGn	配置 SWAP	0x0
SPI_STATUSn	指示 SPI 状态	0x0	FB_CLK_SELn	选择反馈时钟	0x0
SPI_TX_DATAn	存储将要发送的数据	0x0			

表13.3 SPI相关寄存器(n = 0～2)

1．SPI配置寄存器CH_CFGn(n = 0～2)

名　称	位域	类型	功 能 描 述	复位值
HIGH_SPEED_EN	[6]	RW	从机模式下 Tx 输出时间控制位。 0＝禁止；1＝使能(输出时间为 SPI_CLK/2)	0
SW_RST	[5]	RW	软件复位	0
SLAVE	[4]	RW	主从模式选择位。0＝主机模式；1＝从机模式	0
CPOL	[3]	RW	CLK 时钟线初始状态位。0＝高电平；1＝低电平	0
CPHA	[2]	RW	线上相位传输方式选择位。0＝方式 A；1＝方式 B	0
RX_CH_ON	[1]	RW	SPI 接收通道(Rx)使能位。0＝禁止；1＝使能	0
TX_CH_ON	[0]	RW	SPI 发送通道(Tx)使能位。0＝禁止；1＝使能	0

表13.4 SPI配置寄存器CH_CFGn (n = 0～2)

2．SPI模式配置寄存器MODE_CFGn(n = 0～2)

名　称	位域	类型	功 能 描 述	复位值
CH_WIDTH	[30:29]	RW	通道宽度选择位。 00=字节；01=半字；10=字；11=保留	0
TRAILING_CNT	[28:19]	RW	接收 FIFO 中最后写入字节的个数	0
BUS_WIDTH	[18:17]	RW	总线宽度选择位。 00=字节；01=半字；10=字；11=保留	0
RX_RDY_LVL	[16:11]	RW	在 INT 模式下 Rx FIFO 触发水平。 Port 0：触发水平(字节数) =4×N； Port 1、2：触发水平(字节数)=N(N=RX_RDY_LVL)	0

表13.5 模式配置寄存器MODE_CFGn (n = 0～2)

3．SPI状态寄存器CS_REGn(N = 0～2)

名 称	位域	类型	功 能 描 述	复位值
TX_DONE	[25]	R	移位寄存器中传送完毕标志。 0＝其他情况；1＝传送开始后，Tx FIFO 和移位寄存器为空	0
TRAILING_BYTE	[24]	R	Trailing 为 0 标志	0
RX_FIFO_LVL	[23:15]	R	Rx FIFO 中数据水平。Port 0：0～256 B；Port 1、2：0～64 B	0
TX_FIFO_LVL	[14:6]	R	Tx FIFO 中数据水平。Port 0：0～256 B；Port 1、2：0～64 B	0
RX_OVERRUN	[5]	R	Rx FIFO 溢出错误。0＝无错误；1＝溢出错误	0
RX_UNDERRUN	[4]	R	Rx FIFO 数据缺失错误。0＝无错误；1＝数据缺失错误	0
TX_OVERRUN	[3]	R	Tx FIFO 溢出错误。0＝无错误；1＝溢出错误	0
TX_UNDERRUN	[2]	R	Tx FIFO 数据缺失错误。0＝无错误；1＝数据缺失错误	0
RX_FIFO_RDY	[1]	R	0＝FIFO 里的数据低于触发水平；1＝FIFO 里的数据超过触发水平	0
TX_FIFO_RDY	[0]	R	0＝FIFO 里的数据超过触发水平；1＝FIFO 里的数据低于触发水平	0

表13.6 SPI状态寄存器CS_REGn (n = 0～2)

4．SPI发送数据寄存器SPI_TX_DATAn(n = 0～2)

名称	位域	类型	功 能 描 述	复位值
TX_DATA	[31:0]	W	存储将要发送的数据	0

表13.7 SPI发送数据寄存器SPI_TX_DATAn (n = 0～2)

5．SPI接收数据寄存器SPI_RX_DATAn(n = 0～2)

名称	位域	类型	功 能 描 述	复位值
RX_DATA	[31:0]	W	存储接收到的数据	0

表13.8 SPI接收数据寄存器SPI_RX_DATAn (n = 0～2)

SPI控制器编程模型如下：

(1) 设置时钟源并配置分频值等参数；

(2) 软复位后，设置SPI配置寄存器；

(3) 设置反馈时钟选择寄存器；

(4) 设置SPI模式寄存器；

(5) 设置SPI中断使能(INT_EN)寄存器；

(6) 如果需要的话，设置SPI帧个数(PACKET_CNT_REG)寄存器；

(7) 设置从机选择寄存器；

(8) 设置nSSout为低电平，开始收/发操作。

13.3 SPI应用实例

本节给出一种通过SPI信号转CAN总线信号的实例。

Microchip公司的MCP2515是一款局域网络(Controller Area Network，CAN)协议控制器，完全支持CAN V2.0B技术规范。

本实例通过SPI控制操作CAN总线控制器，使其工作在回环模式。

1. 相关寄存器结构体的定义：

```
typedef struct {
                    unsigned int CH_CFG0;
                    unsigned int CLK_CFG0;
                    unsigned int MODE_CFG0;
                    unsigned int CS_REG0;
                    unsigned int SPI_INT_EN0;
                    unsigned int SPI _STATUS0;
                    unsigned int SPI _TX_DATA0;
                    unsigned int SPI _ RX_DATA0;
                    unsigned int PACKET_CNT_REG0;
                    unsigned int PENDING_CLR_REG0;
                    unsigned int SWAP_CFG0;
                    unsigned int FB _CKK_SEL0;
                    } spi0;
#define SPI0 (*(volatile spi0*) 0x 13920000
```

(2) 延时函数、片选从机芯片函数和取消片选芯片函数：

```
void delay(int times)
{
  volatile int i, j;
  for(j = 0;j<times;j++)
    for(i = 0;i<1000;i++);
}
void disable_chip(void)
{
  SPI0.CS_REG0 |= 0X1;// CS_REG0[0]=1，禁止从机芯片
  delay(1);
}
void enable_chip(void)
{
  SPI0.CS_REG0 &= ~0X1;  // CS_REG0[0]=0，使能从机芯片
  delay(1);
}
```

(3) 软件复位代码：

```
void soft_reset(void)
{
  SPI0.CH_CFG0 |= (0X1<<5); // CH_CFG0[5]=1，激活软件复位
  delay(1);
  SPI0.CH_CFG0 &= ~(0X1<<5);       // CH_CFG0[5]=0，禁止软件复位
}
```

(4) 指定地址字节读的实现：图13.4是MCP2515芯片手册给出的读时序图。

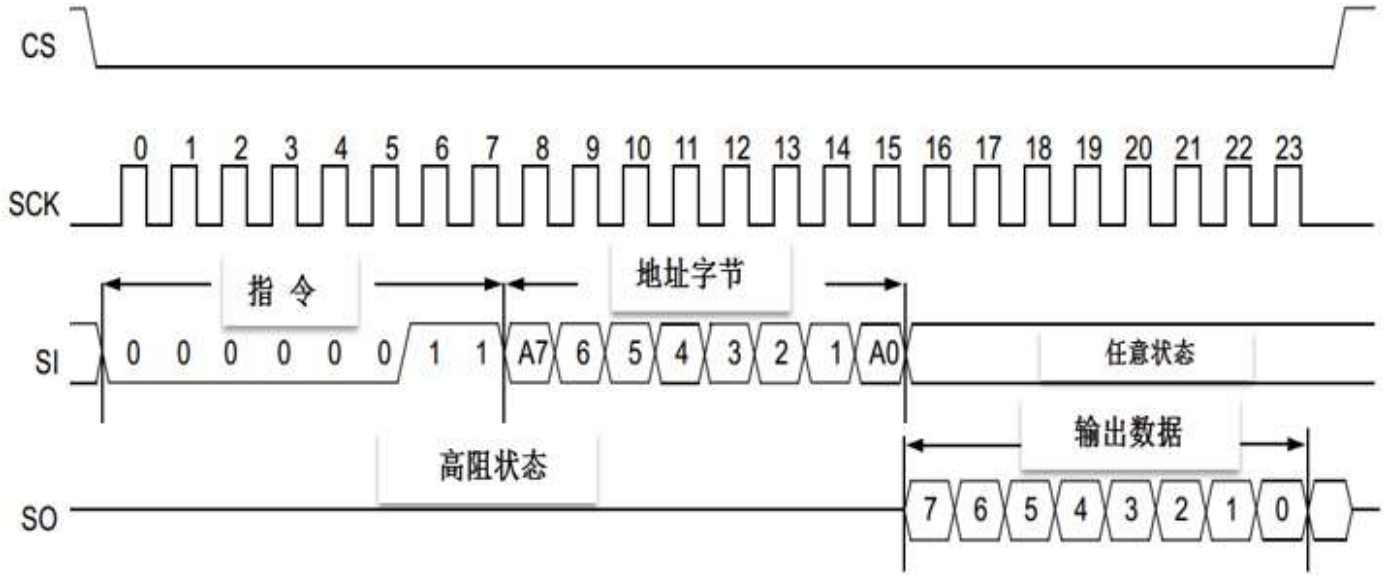


图13.4 SPI读时序图

```
void send_byte(unsigned char data) //向SPI总线发送一个字节
{
        SPI0.CH_REG0 |= 0X1; // CH_REG0[0]=1，使能Tx通道
        delay(1);
        SPI0.SPI_TX_DATA0 = data;
        while(!(SPI0.SPI_STATUS0 & (0x1<<25))); //等待发送数据完毕
        SPI0.CH_REG0 &= ~0X1; //CH_REG0[0]=0，禁止Tx通道
}
unsigned char recv_byte( ) //从SPI总线读取一个字节
{
        unsigned char data;
        SPI0.CH_REG0 |= (0X1<<1); //CH_REG0[1]=1，使能Rx通道
        delay(1);
        data = SPI0.SPI_RX_DATA0;
        delay(1);
        SPI0.CH_REG0 &= ~(0X1<<1); //CH_REG0[1]=0，禁止Rx通道
        return data;
}
```

/*功能：从指定地址起始的寄存器读取数据；

*输入：Addr为要读取地址寄存器的地址；

返回值：从地址中读取的数值/

```
unsigned char read_byte_fromAddr(unsigned char Addr)
{
        unsigned char ret;
        enable_chip( ); //CS_REG0[0]= 0;
        send_byte(0x03); //发送读命令
        send_byte(Addr); //发送地址
        ret = recv_byte( ); //接收数据
        disable_chip( ); //CS_REG0[0] = 1
        return(ret);
}
```

(5) 指定地址字节写的实现：图13.5是MCP2515芯片手册给出的写时序图。

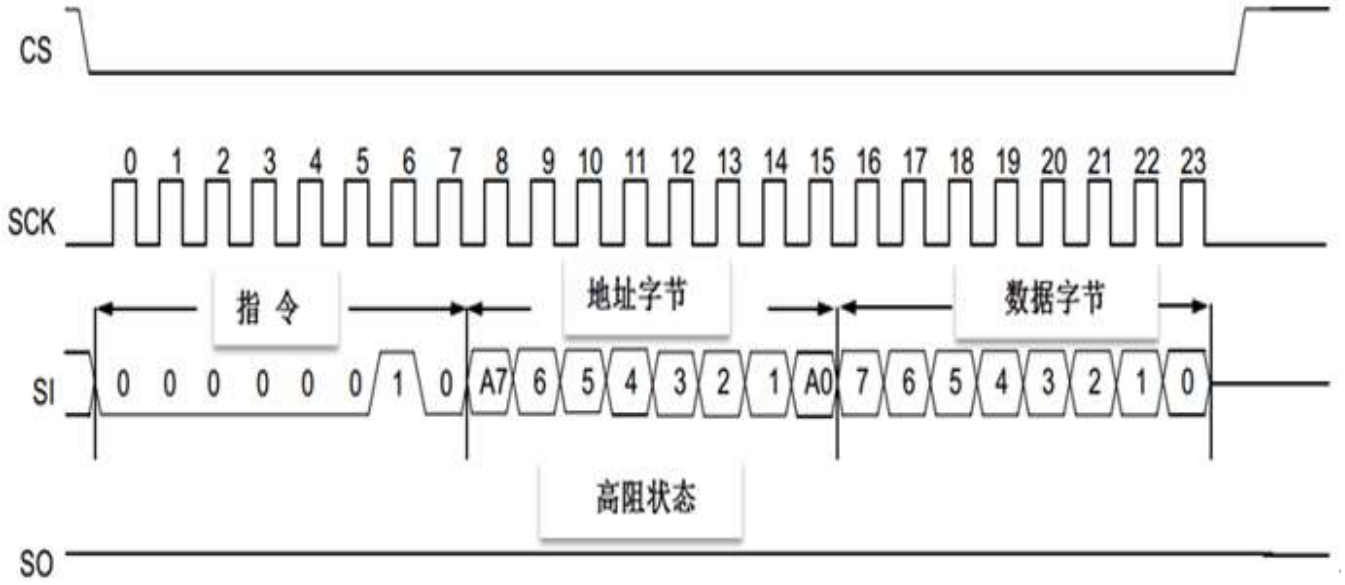


图13.5 SPI写时序图

/*功能：将数据写入到指定地址起始的寄存器；

*输入：addr为寄存器的地址；

data：向寄存器写入的数据/

```
unsigned char write_byte_toaddr(unsigned char addr, unsigned char data)

{
        enable_chip( );
        send_byte(0x02);
        send_byte(addr);
        send_byte(data);
        disable_chip( );
}
```

(7) 主程序：在Exynos 4412片内SPI控制器的操作下，实现了SoC与CAN控制器MCP2515的通信，将MCP2515配置成回环模式，实现数据的自发自收。

```
#include "exynos4412.h"
unsigned char src[ ] = "This is a example for spi";
unsigned char dst[100];
int main( )
{
        unsigned char buff[8]; //存放状态字
        unsigned char  ret;
        volatile int i = 0;
        uart0_init( );
        GPB.GPBCON = (GPB.GPBCON & ~0xffff) | 0x2222; //设置IO引脚位SPI0
模式
        //设置SPI时钟位PCLK，使能时钟spi clock = pclk/10 =6.6 MHz
        SPI0.CLK_DIV_PERIL1 = (0x1<<8)|(4<<0);      // pclk/(1+1)/(4+1)

        soft_reset( ); //软件复位SPI控制器
        SPI0.CH_CFG0 &= ~(0x1f); //主机模式，CPOL = 0，CPHA = 0
        SPI0.MODE_CFG0 &= ~((0x3<<17) | (0x3<<29)); //总线宽度8 bit，通
道宽度8 bit
        SPI0.CS_REG0 &= ~(0x01<<1); //选择手动选择芯片
        delay(10);
        Init_can( );
```

```
while(1)
{
    printf("\nplease input 8 bytes\n\r");
    for(i=0;i<8;i++) src[i] = getc( );
    can_sent(src);
    delay(100);
    ret = can_receive(dst);
    printf("src=");
    for(i=0;i<8;i++) printf("%x”,src[i] ");
    printf("\n\r");
    printf("dst");
    for(i=0;i<8;i++) printf("%x",dst[6+i] ");
    printf("\n\r");
}
    }
```

问题与思考：

1. SPI总线和I^2C总线的区别是什么？
2. 试编写一个实现读/写SPI功能的程序。
3. 试编写SPI用于DMA通信的程序。