KF8F3132——SPI 主从模式样例程序

引言

本应用笔记提供了 KF8F3132—SPI 模块相关的配置信息以及如何能够快速的理解并上手使用该模块的一些配置方式。

本应用笔记须与 KF8F3132 数据手册结合使用。

寄存器

寄存器使用说明：

OSCCTL：系统控制寄存器

寄存器OSCCTL：系统频率控制寄存器(地址:2FH)

	bit7							bit0
复位值 0011 ----	CLKOE	IRCS2	IRCS1	IRCS0	-	-	-	-
	R/W	R/W	R/W	R/W	U	U	U	U

图注：R＝可读　W＝可写　P＝可编程　U＝未使用
-＝读为0　x＝状态未知

OPTR：选择寄存器

寄存器6.1： OPTR: 选择寄存器(地址: 21H)

	bit7							bit0
复位值 1111 1111	$\overline{\text{PUPH}}$	INT0SE	T0CS	T0SE	PSA	PS2	PS1	PS0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TR1：P1 口方向控制寄存器

TR1：P1口方向控制寄存器(地址：27H)

	bit7							bit0
复位值 --11 1111	-	-	TR15	TR14	TR13	TR12	TR11	TR10
	U	U	R/W	R/W	R/W	R/W	R/W	R/W

P1LR：P1 口输出锁存控制寄存器

寄存器P1LR: P1口输出锁存寄存器(地址: 47H)

	bit7							bit0
复位值 --xx xxxx	-	-	P1LR5	P1LR4	P1LR3	P1LR2	P1LR1	P1LR0
	U	U	R/W	R/W	R/W	R/W	R/W	R/W

T0：定时/计数器 1 寄存器

SSCICTL0：SSCI 控制寄存器 0

寄存器10.1: SSCICTL0: SSCI控制寄存器0(地址:128H)

	bit7							bit0
复位值 0000 0000	SSCIWCFL	SSCIOV	SSCIEN	SSCICKP	SSCI MOD3	SSCI MOD2	SSCI MOD1	SSCI MOD0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

SSCICTL1：SSCI 控制寄存器 1

寄存器10.2: SSCICTL1: SSCI控制寄存器1(地址:12AH)

	bit7							bit0
复位值	SSCI CALLEN	SSCI ACKSTA	SSCI ACKDAT	SSCI ACKEN	SSCI RCEN	STOPEN	RESTART EN	STARTEN
0000 0000	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W

SSCISTA：SSCI 状态寄存器

寄存器10.3: SSCISTA: SSCI状态寄存器(地址:12BH)

	bit7							bit0
复位值	SAMPLE	CKEGE	SSCIDA	SSCISTOP	SSCI START	SSCIRW	SSCIUA	SSCIBUF
0000 0000	R/W	R/W	R	R	R	R	R	R

SSCIADD：IIC 地址寄存器

寄存器10.5: SSCIADD: I2C地址寄存器(地址:12EH)

	bit7							bit0
复位值	SSCI ADD7	SSCI ADD6	SSCI ADD5	SSCI ADD4	SSCI ADD3	SSCI ADD2	SSCI ADD1	SSCI ADD0
0000 0000	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

SSCIBUFR：SSCI 数据接收/发送缓冲寄存器

位操作使用说明：

8 位单片机支持对寄存器的位进行直接的操作，因此在使用的过程中不仅可以通过给寄存器赋值来达到想要的配置，同时还可以直接对位进行操作来达到需要的配置。

以下是对程序中使用到的位进行说明：

T0IF:T0 中断标志位

SSCIPIN：SSCI 模块功能引脚切换位

SSCIEN：同步串行端口使能位

STARTEN：启动条件使能位

SSCIIF：SSCI 中断标志位

STOPEN：停止条件使能位

SSCIBUF：缓冲器满状态位

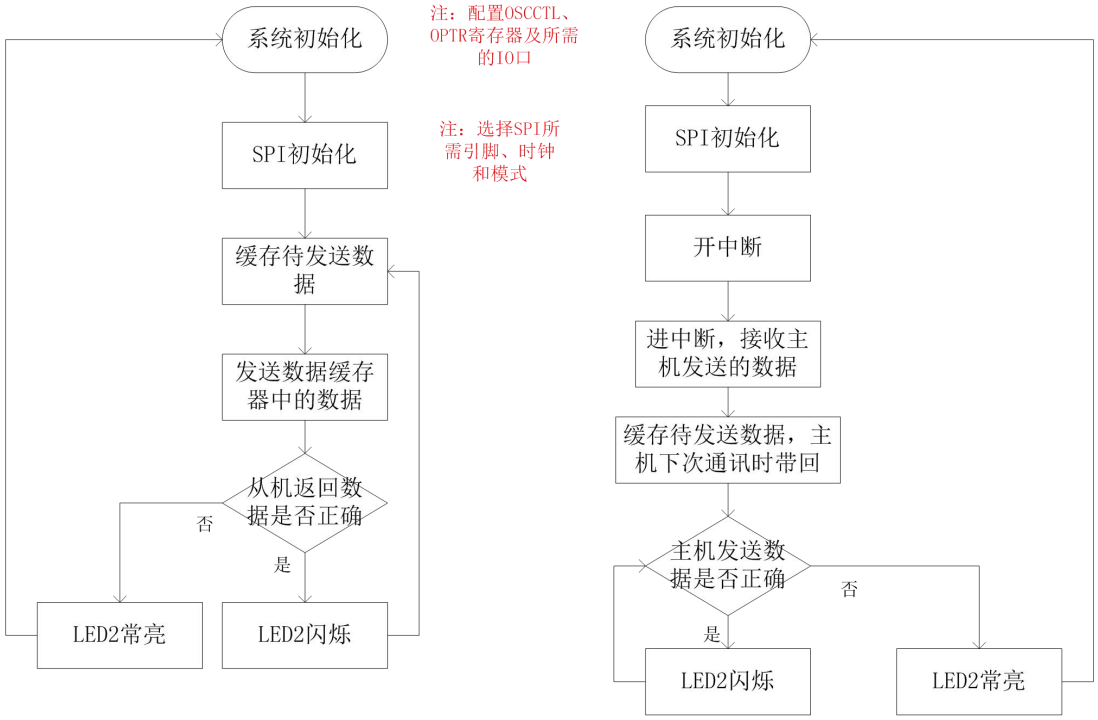
SSCIACKSTA：应答状态位

SSCIRCEN：接受使能位

SSCIACKDAT：应答数据位

SSCIACKEN：应答序列使能位

SPI 主从模式样例程序框图



注：流程图中的注释没有详细说明需要操作的寄存器，因为寄存器较多，用户可在后边的详细程序中查看，具体的寄存器配置。

SPI 主从模式样例简述：

开发环境：ChipON IDE

功能简述：使用 SSCI 的 SPI 功能实现多字节连续的收发实验。

硬件连接：实验时，需要用 2 块 KF8F3132 开发板。

主控机的 P13（SDI）——从机 P17（SDO）

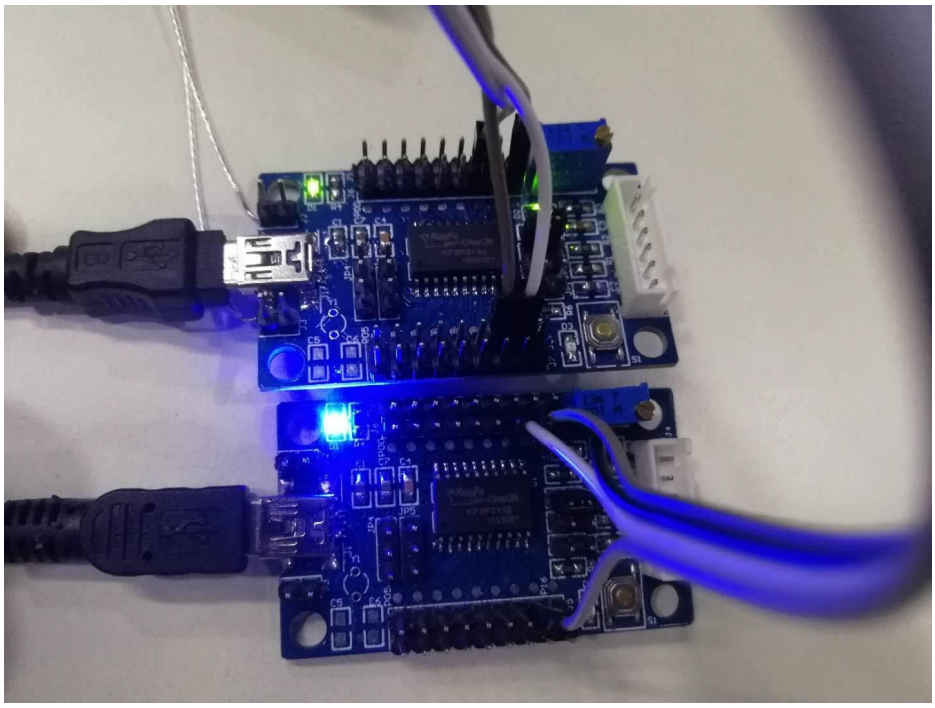
主控机的 P17（SDO）——从机 P13（SDI）

主控机的 P15（SCK）——从机 P15（SCK）

主控机的 P14——从机 P23（SS）

LED2 的 JP2 用跳线连接。

下面为主机从机的硬件连接实物图：



SPI 主机样例程序：

MCU 初始化：

```
void Init_mcu()
{
      OSCCTL =0x70;//系统时钟设为16M

      TR1= 0x08; //P13设置为输入
      P1LR =0x04; //P12输出为高

      OPTR =0x03;//T0 16分频，T1的单数计时周期是4us
}
```

SPI 模块初始化：

```
void Init_SPI()
{
      SSCIPIN = 0;          //SDI=P13，SCk=P15
      SDOPIN = 0;           //SDO=P17，主机的SS脚配置为IO口

      SSCIEN=0;                     //关闭SPI模块，初始化完之后再打开

      SSCICTL0=0X02;                //SPI 主控模式，时钟 = 工作时钟/64
      SSCICTL1=0X00;
      SSCISTA=0X00;

      MODE4//SPI的模式配置
      SSCIEN=1;
}
```

发送单字节（接收从机反馈字节）：

```
uchar SPI_one(uchar d)
{
      SSCIBUFR=d; //将待发送的字节放入发送缓存中
      while(!SSCIIF); //等待发送完毕后SSCIIF置位，
      SSCIIF=0;
      return SSCIBUFR;//返回接收的字节
}
```

延时函数 1：

```
void Delay_ms(uint j)
{
      uint k=0;
      for(k=0;k<j;k++)
      {
            T0 =6;
            T0IF=0;
            while(!T0IF);
      }
}
```

延时函数 2：

```
void Delay_50us(uint j)
{
      uint k=0;
      for(k=0;k<j;k++)
      {
            T0 =244;
            T0IF=0;
            while(!T0IF);
      }
}
```

连续发送多个字节：

```
void SPI_mul(uchar cnt)
{
      uchar i=0;
      for(i=0;i<10;i++)//清空接收缓存
            Reciv_BUF[i]=0;

      for(i=0;i<cnt;i++)
      {
            Delay_50us(1);//延时一段时间，给接收端留下足够的处理时间
            Reciv_BUF[i]=SPI_one(Send_BUF[i]);//发送数据后，读取接收的数据
      }

      for(i=0;i<10;i++)//清空Send_BUF
            Send_BUF[i]=0;
}
```

主函数：

```
void main()
{
	uchar k=0, Flag=0;
	Init_mcu();//初始化时钟和IO口
	Init_SPI();//初始化SPI外设
	while(1)
	{
		for(k=0;k<10;k++)//填写待发送的数据到Send_BUF
		{
			Send_BUF[k]=k;
		}

		SS=0;//使能从机
		SPI_mul(10);//发送Send_BUF中的10个数据
		SS=1;//复位从机

		for(k=0;k<10;k++)//验证从机发来的数据是否正确
		{
			if(k+10 == Reciv_BUF[k])
				Flag=1;
			else
			{
				Flag=0;
				break;
			}
		}

		if(Flag)//如果从机发来的数据正确LED2闪烁
		{
			Flag=0;
			Delay_ms(100);
			LED2 =0;
			Delay_ms(100);
			LED2 =1;
		}

	}

}
```

SPI 从机样例程序：

MCU 初始化：

```
void Init_mcu()
{
      OSCCTL =0x70;//系统时钟配置为16M

      TR1= 0x28;//将P13和P15配置为输入模式
      P1LR =0x04;//LED2初始化为熄灭状态

      TR23 =1;//SS配置为输入模式

      OPTR =0x03;//T0 16分频，单计数周期是4us
}
```

SPI 模块初始化：

```
void Init_SPI()
{
      SSCIPIN = 0;          //SDI=P13，SCk=P15
      SDOPIN = 0;           //SDO=P17
      SSPIN = 0;            //SS=P23

      //SPI初始化
      SSCIEN=0;                     //关闭SPI模块，初始化完之后再打开

      SSCICTL0=0X04;                //SPI 从动模式，时钟 = SCK 引脚
      SSCICTL1=0X00;
      SSCISTA=0X00;

      MODE4//配置SPI工作于模式4
      SSCIEN=1;

      SSCIIF=0;                     //清零SPI中断标志
      SSCIIE=1;                     //打开SPI接收中断

     //中断初始化
      PUIE=1;                 //使能所有未屏蔽的外设中断
     AIE = 1;           //打开总中断
}
```

延时函数：

```
void Delay_ms(uint j)
{
      uint k=0;
      for(k=0;k<j;k++)
      {
            T0 =6;
            T0IF=0;
            while(!T0IF);
      }
}
```

中断函数：

```
void int_fun0() __interrupt (0)
{
     if(SSCIIF)//接收中断需要一定的时间处理数据，主机的发送速度不能太快
     {
          SSCIIF=0;
          if(0==SSCIBUFR)//如果收到的是第一个字节，初始化接收计数和发送计数
          {
               Reciv_cnt =0;
               Send_cnt=1;
          }

          Reciv_BUF[Reciv_cnt++] =SSCIBUFR;//读出接受的数据，并将它存到缓存中
          if(Reciv_cnt>9)
               Reciv_cnt=0;

          SSCIBUFR= Send_BUF[Send_cnt++];//
将待发送的数据填写到SSCIBUFR中，等待下次主机通讯时带回
          if(Send_cnt>9)
               Send_cnt=0;
     }
}
```

主函数：

```
void main()
{
     uchar i=0, Flag=0;
     Init_mcu();//初始化时钟和IO口
     Init_SPI();//初始化SPI外设
     for(i=0;i<10;i++)//填写待发送数据的缓存
     {
          Send_BUF[i]= i+10;
     }

     while(1)
     {
          for(i=0;i<10;i++)//验证接收的数据是否正确
          {
               if(i== Reciv_BUF[i])
                    Flag=1;
               else
               {
                    Flag=0;
                    break;
               }
          }

          if(Flag)//如果接受的数据正确则LED2闪烁
          {
               Flag=0;
               Delay_ms(100);
               LED2 =0;
               Delay_ms(100);
               LED2 =1;
          }
     }

}
```

注意事项：

1、因为由主控制器控制 SCK 信号，所以它可以在任意时刻启动数据传输。

2、在从动模式下，外部时钟由 SCK 引脚上的外部时钟源提供。

3、当 SPI 处于从动模式，并且 SS 引脚控制使能时，如果 SS 引脚置为 VDD 电平将使 SPI 模块复位。

4、主控模式下，进入休眠模式后所有模块的时钟都停振，在器件被唤醒前，发送/接收也将保持