MSP430程序库<五>SPI 同步串行通信

SPI 总线系统是一种同步串行外设接口；是一种高速的，全双工，同步的通信总线，并且在芯片的管脚上只占用四根线，节约了芯片的管脚，同时为 PCB 的布局上节省空间，提供方便，正是出于这种简单易用的特性，现在越来越多的芯片集成了这种通信协议。 许多的芯片都用这种协议通信：EEPROM、Flash、实时时钟、AD 转换器、数字信号处理器等：MSP430 的 USART 模块不仅能够实现异步模式（见：MSP430程序库<二>UART 异步串口），而且支持同步串行通信（即 SPI 模式）；其 SPI 支持3线、4线操作，支持主机模式和从机模式，字符长度可以7位或8位等。由于要用 AD7708芯片完成 AD 采样，AD7708是通过 SPI 与其它设备通信的；本程序比较简化，只完成了主机模式的初始化。

硬件介绍：

SPI：SPI 是 Motorola 首先在其 MC68HCXX 系列处理器上定义的，它是一种同步的高速串行通信协议，有关 SPI 协议的详细内容，参考：SPI _互动百科。

MSP430对 SPI 的支持：当 msp430USART 模块控制器 UxCTL 的位 SYNC 置位时，USART 模块工作于同步模式，对于149即工作于 SPI 模式，若是169，USART0可以支持 I2C，可以通过另一控制位 I2C 控制，I2C 位0则工作于 SPI。在 SPI 模式下，允许单片机以确定的速率发送和接收7位或8位数据。

同步通信与异步通信类似；同步通信和异步通信寄存器资源一致，具体寄存器的不同位之间的功能存在差异；具体寄存器内容参见 TI 提供的用户指南。

USART 模块的 SPI 操作可以是3线和4线，其信号如下：
SIMO：从进主出，主机模式下，数据输出；从机模式下，数据输入。
SOMI：从出主进，主机模式下，数据输入；从机模式下，数据输出。
UCLK：USART SPI 模式时钟，信号有主机输出，从机输入。
STE：从机模式发送接收允许控制脚，用于4线模式，控制多主从系统中多个从机，避免发生冲突。具体方式如下(图截自 用户指南)：

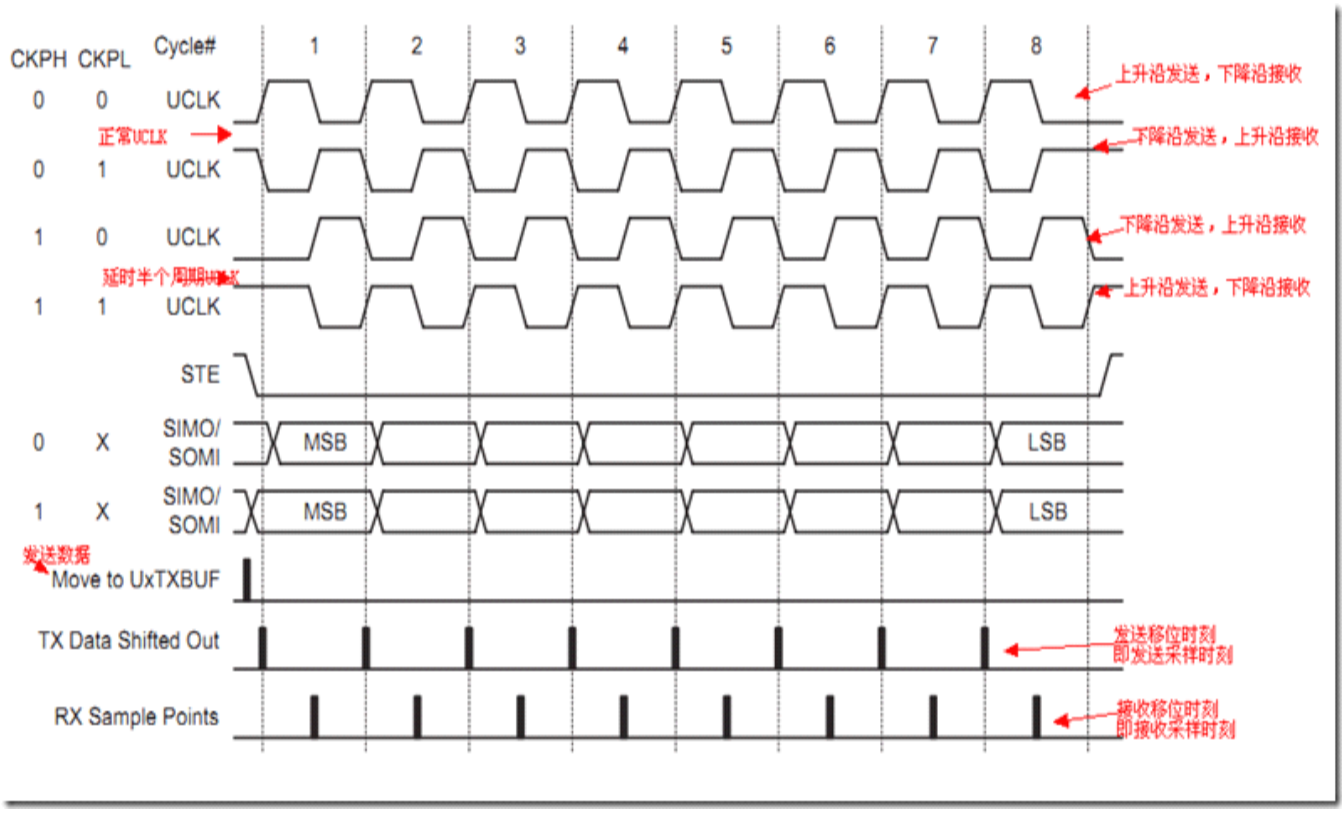
❑ STE	Slave transmit enable. Used in 4-pin mode to allow multiple masters on a single bus. Not used in 3-pin mode. 4-Pin master mode: When STE is high, SIMO and UCLK operate normally. When STE is low, SIMO and UCLK are set to the input direction. 4-pin slave mode: When STE is high, RX/TX operation of the slave is disabled and SOMI is forced to the input direction. When STE is low, RX/TX operation of the slave is enabled and SOMI operates normally.

四线主机模式：STE 为高电平，SIMO 和 UCLK 操作正常；STE 为低电平，SIMO 和 UCLK 被置为输入方向，主机控制权让出。
四线从机模式：STE 为高电平，从机的发送和接收无效，且把 SOMI 置为输入方向；STE

为低电平，发送接收正常，SOMI 也为正常输出。

USART 模块串行时钟极性和相位设置：

USART 的时钟 UCLK 的极性和相位由位于 UxTCTL 寄存器的 CKPH 和 CKPL 位控制，具体如下图：在程序中，我分别称之为，时钟模式0、时钟模式1、时钟模式2、时钟模式3。



USART 的波特率产生，SPI 不同于异步通信：异步通信由 UxBR1\UxBR0\UxMCTL 三个寄存器控制，以产生标准频率；而同步模式，主从设备用同一个时钟，不再需要产生标准时钟，故而不再用 UxMCTL 寄存器，设其值为0.

其他的，与异步通信基本一致，这里不再细说。具体参考用户指南。

程序实现：

程序和异步通信方式类似：首先是初始化函数，然后是读取数据、写入数据函数。此程序采用和我之前的 UART 程序库类似的结构，写入数据后进入低功耗等待中断，判断标志位进行写入数据和读取数据。

这里函数只实现430的主机模式，如需从机模式可以仿照我的程序，进行简化实现。

由于，我即将使用的 SPI 设备（AD7708）不是字符型设备，这里不再实现写入字符串函数，也不再移植 printf 和 scanf 函数，如若需要可以自己添加，printf 和 scanf 的移植参考：MSP430程序库<四>printf 和 scanf 函数移植

初始化函数：SpiMasterInit，实现主机模式的初始化工作，函数内容如下：

```
charSpiMasterInit(longbaud,chardataBits,charmode,charclkMode)
{
 long intbrclk;                    //波特率发生器时钟频率

UxCTL |= SWRST;                      //初始
```

```
 //反馈选择位，为1，发送的数被自己接收，用于测试，正常使用时注释掉
 //UxCTL |= LISTEN;

UxCTL |= SYNC + MM;                 //SPI 主机模式

 //时钟源设置
UxTCTL &=~ (SSEL0+SSEL1);          //清除之前的时钟设置
if(baud<=16364)                    //
   {
    UxTCTL |= SSEL0;                //ACLK，降低功耗
  brclk = 32768;                   //波特率发生器时钟频率=ACLK(32768)
   }
 else
   {
    UxTCTL |= SSEL1;                //SMCLK，保证速度
  brclk = 1000000;                 //波特率发生器时钟频率=SMCLK(1MHz)
   }

 //-----------------------设置波特率-------------------------
if(baud < 300||baud > 115200)    //波特率超出范围
   {
        return0;
    }
 //设置波特率寄存器
intfen = brclk / baud;           //分频系数
if(fen<2)return(0);              //分频系数必须大于2
else
   {
        UxBR0 = fen / 256;
        UxBR1 = fen % 256;
    }

 //-----------------------设置数据位-------------------------
   switch(dataBits)
    {
     case7:case'7': UxCTL &=~ CHAR; break;       //7位数据
    case8:case'8': UxCTL |= CHAR;   break;       //8位数据
    default:         return(0);                  //参数错误
   }
 //-----------------------设置模式--------------------------
   switch(mode)
    {
     case3:case'3': UxTCTL |= STC;   USPI3ON;     break;  //三线模式
```

```
      case4:case'4': UxTCTL &=~ STC; USPI4ON;     break;  //四线模式
      default:       return(0);                          //参数错误
    }

  //-----------------------设置 UCLK 模式----------------------
    switch(clkMode)
     {
      case0:case'0': UxTCTL &=~ CKPH; UxTCTL &=~ CKPL;    break;  //模
式0
      case1:case'1': UxTCTL &=~ CKPH; UxTCTL |= CKPL;     break;  //模式
1
      case2:case'2': UxTCTL |= CKPH;  UxTCTL &=~ CKPL;    break;  //模式
2
      case3:case'3': UxTCTL |= CKPH;  UxTCTL |= CKPL;     break;  //模式3
      default:       return(0);                           //参数错误
    }

   UxME |= USPIEx;              //模块使能

  UCTL0 &= ~SWRST;              // Initialize USART state machine

  UxIE |= URXIEx + UTXIEx;     // Enable USART0 RX interrupt

  return(1);                   //设置成功
  }
```

程序注释已经比较详细，这里不再细说；如果要改为从机模式，把时钟设置和波特率设置去掉应该就可以了。

发送函数和接收函数：

```
  voidSpiWriteDat(charc)
  {
   while(TxFlag==0) SpiLpm();  // 等待上一字节发完，并休眠
  TxFlag=0;                    //
    UxTXBUF=c;
  }
  charSpiReadDat()
  {
   while(RxFlag==0) SpiLpm(); // 收到一字节?
    RxFlag=0;
     return(UxRXBUF);
  }
```

发送和接收函数和异步通信里面的几乎一样，如果标志位为0，则等待改变为1，然后写入或读出；标志位在中断函数里被更改；中断函数如下：

```
  #pragmavector=USARTxRX_VECTOR
```

```
__interrupt voidUartRx()
{
    RxFlag=1;
    __low_power_mode_off_on_exit();
}
#pragmavector=USARTxTX_VECTOR
__interrupt voidUartTx ()
{
    TxFlag=1;
    __low_power_mode_off_on_exit();
}
```

中断里面仅仅置标志位后，就退出低功耗；退出后即写入或者读取数据。

读取或写入函数调用的 SpiLpm 函数：

```
voidSpiLpm()
{
 if(UxTCTL&SSEL0) LPM3;  //若以 ACLK 作时钟，进入 LPM3休眠(仅打开 ACLK)
else              LPM0;  //若以 SMCLK 作时钟，进入 LPM0休眠(不关闭 SMCLK)
}
```

根据不同情况进入低功耗，如果单片机其他地方不允许进入低功耗，可以更改这个函数。

程序部分就这么多了。需要的函数在头文件里面声明，方便使用。

使用示例：

程序使用方式和之前的程序库相同，加入 c 文件，包含 h 文件，调用初始化函数后即可掉用程序库中的函数。

```
#include"msp430x16x.h"   //430寄存器头文件
#include"Spi.h"          //串口通讯程序库头文件

voidmain()
{
 // Stop watchdog timer to prevent time out reset
   WDTCTL = WDTPW + WDTHOLD;

    ClkInit();
 // 主机模式，波特率25000,8位数据位，三线模式，时钟模式0（具体见 spi.c）
   SpiMasterInit(25000,8,3,0);
    _EINT();


 while(1)                    //串口测试
   {
       SpiWriteDat(0X20);
       chara = SpiReadDat();
    }
```

```
}
```

这里只是一个简单的使用示例，详细的使用，将会在下一篇给出，下一篇：MSP430程序库<六>通过 SPI 操作 AD7708；将会使用今天的程序库，完成 SPI 的通信部分。

注意事项：

SPI 是全双工通信，每次写入（发送）8位/7位数据的同时，430的 SPI 主模块都会在发送后半个时钟周期读取采样的0/1信号，存入接收缓冲寄存器，所以，每次的写入，均有数据读取，但不一定是从设备发送回来的，这个地方在使用430主机模式的时候必须注意，很容易出错（我也是在调试 AD7708的时候才注意到这个地方的）；SPI 的函数已经添加 SpiWriteData 函数，这个函数会在发送的同时返回发送完成半个时钟周期后的接收到的数据，方便使用；不建议使用前面的发送和读取函数，很容易出错；建议使用刚添加的这个函数，程序库已经更新，可以重新下载。函数 SpiWriteData：

```
charSpiWriteData(charc)
{
    SpiWriteDat(c);
    returnSpiReadDat();
}
```

发送后读取即可，程序比较简单。

新的示例程序：

```
voidmain()
{
 // Stop watchdog timer to prevent time out reset
   WDTCTL = WDTPW + WDTHOLD;

    ClkInit();
 // 主机模式，波特率25000,8位数据位，三线模式，时钟模式0（具体见 spi.c）
   SpiMasterInit(25000,8,3,0);
    _EINT();


 while(1)                          //串口测试
   {
     SpiWriteData(0X20);       //只写入
    chara = SpiWriteData(0xff);     //只读取
   }
}
```

详细示例，参参考下一篇：MSP430程序库<六>通过 SPI 操作 AD7708

SPI 的通信部分完成。

相关文章及附件下载：http://www.ideyi.org/bbs/article_1077_370082.html