MSP430FR5969 SPI 测试

实例代码：

```
//   MSP430F59xx Demo - eUSCI_A0, SPI 3-Wire Master Incremented Data
//
//   Description: SPI master talks to SPI slave using 3-wire mode. Incrementing
//   data is sent by the master starting at 0x01. Received data is expected to
//   be same as the previous transmission TXData = RXData-1.
//   USCI RX ISR is used to handle communication with the CPU, normally in LPM0.
//   ACLK = 32.768kHz, MCLK = SMCLK = DCO ~1MHz.  BRCLK = ACLK/2
//
//
//                  MSP430FR5969
//               -----------------
//           /|\ |              XIN|-
//            |  |                 |  32KHz Crystal
//            ---|RST          XOUT|-
//               |                 |
//               |             P2.0|-> Data Out (UCA0SIMO)
//               |                 |
//               |             P2.1|<- Data In (UCA0SOMI)
//               |                 |
//               |             P1.5|-> Serial Clock Out (UCA0CLK)
//
//   P. Thanigai
//   Texas Instruments Inc.
//   Feb 2012
//   Built with CCS V5.5
//******************************************************************************
#include <msp430.h>

volatile unsigned char RXData = 0;
volatile unsigned char TXData;

int main(void)
{
  WDTCTL = WDTPW | WDTHOLD;                 // Stop watchdog timer

  // Configure GPIO
  P1SEL1 |= BIT5;                       // USCI_A0 operation
  P2SEL1 |= BIT0 | BIT1;                    // USCI_A0 operation
  PJSEL0 |= BIT4 | BIT5;                    // For XT1

  // Disable the GPIO power-on default high-impedance mode to activate
  // previously configured port settings
  PM5CTL0 &= ~LOCKLPM5;

  // XT1 Setup
  CSCTL0_H = CSKEY >> 8;                    // Unlock CS registers
  CSCTL1 = DCOFSEL_0;                       // Set DCO to 1MHz
  CSCTL2 = SELA__LFXTCLK | SELS__DCOCLK | SELM__DCOCLK;
  CSCTL3 = DIVA__1 | DIVS__1 | DIVM__1;     // set all dividers
  CSCTL4 &= ~LFXTOFF;
  do
  {
    CSCTL5 &= ~LFXTOFFG;                    // Clear XT1 fault flag
    SFRIFG1 &= ~OFIFG;
  }while (SFRIFG1&OFIFG);                   // Test oscillator fault flag
  CSCTL0_H = 0;                             // Lock CS registers

  // Configure USCI_A0 for SPI operation
  UCA0CTLW0 = UCSWRST;                      // **Put state machine in reset**
  UCA0CTLW0 |= UCMST | UCSYNC | UCCKPL | UCMSB; // 3-pin, 8-bit SPI master
                                            // Clock polarity high, MSB
  UCA0CTLW0 |= UCSSEL__ACLK;                // ACLK
  UCA0BR0 = 0x02;                           // /2
  UCA0BR1 = 0;                              //
  UCA0MCTLW = 0;                            // No modulation
  UCA0CTLW0 &= ~UCSWRST;                    // **Initialize USCI state machine**
  UCA0IE |= UCRXIE;                         // Enable USCI_A0 RX interrupt
  TXData = 0x1;                             // Holds TX data

  while(1)
  {
    UCA0IE |= UCTXIE;
    __bis_SR_register(LPM0_bits | GIE);     // CPU off, enable interrupts
    __delay_cycles(2000);                   // Delay before next transmission
    TXData++;                               // Increment transmit data
  }
}
```

```
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=USCI_A0_VECTOR
__interrupt void USCI_A0_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(USCI_A0_VECTOR))) USCI_A0_ISR (void)
#else
#error Compiler not supported!
#endif
{
  switch(__even_in_range(UCA0IV, USCI_SPI_UCTXIFG))
  {
    case USCI_NONE: break;
    case USCI_SPI_UCRXIFG:
      RXData = UCA0RXBUF;
      UCA0IFG &= ~UCRXIFG;
      __bic_SR_register_on_exit(LPM0_bits); // Wake up to setup next TX
      break;
    case USCI_SPI_UCTXIFG:
      UCA0TXBUF = TXData;               // Transmit characters
      UCA0IE &= ~UCTXIE;
      break;
    default: break;
  }
}
```

分析：直接分析SPI部分，其实这篇的话也是对我自己一些不理解的进行提问，希望能得到解答。

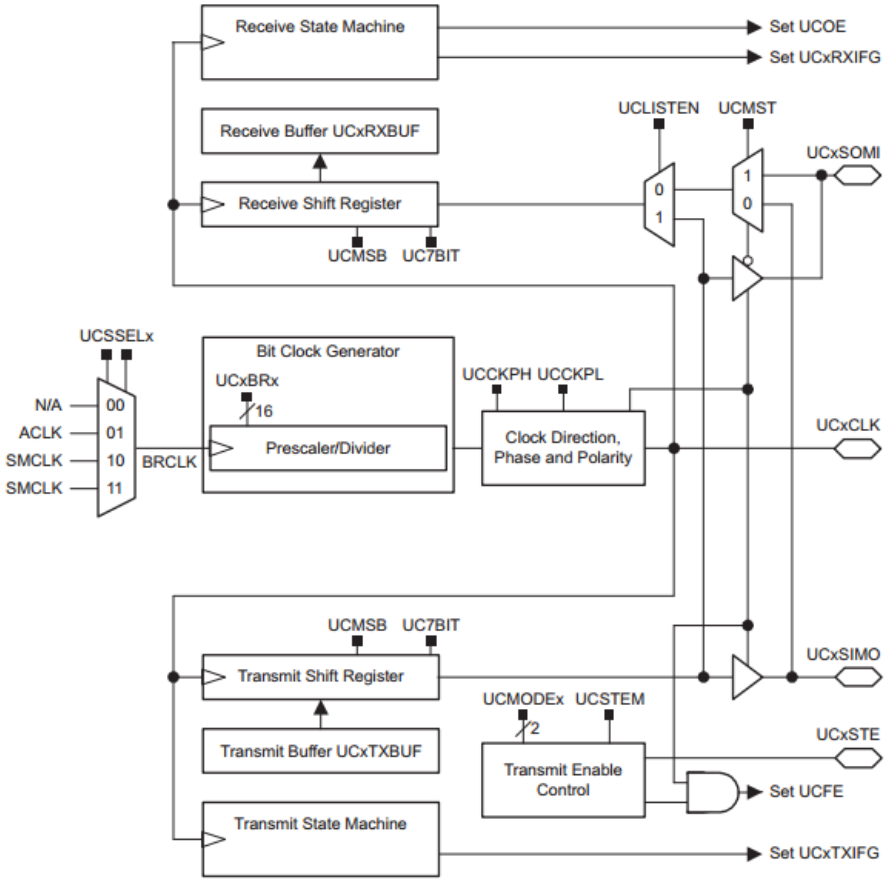

Figure 25-1. eUSCI Block Diagram – SPI Mode

1、配置USCI_A0为I2C模式。

（1）在配置或者重新配置eUSCI_A模块时应该先将UCSWRST置1以避免出现配置错误，配置完之后释放复位。

```
UCA0CTLW0 = UCSWRST;                      // **Put state machine in reset**
UCA0CTLW0 &= ~UCSWRST;                    // **Initialize USCI state machine**
```

（2）设置模式。

Table 25-3. UCAxCTLW0 Register Description

Bit	Field	Type	Reset	Description
15	UCCKPH	RW	0h	Clock phase select 0b = Data is changed on the first UCLK edge and captured on the following edge. 1b = Data is captured on the first UCLK edge and changed on the following edge.
14	UCCKPL	RW	0h	Clock polarity select 0b = The inactive state is low. 1b = The inactive state is high.
13	UCMSB	RW	0h	MSB first select. Controls the direction of the receive and transmit shift register. 0b = LSB first 1b = MSB first
12	UC7BIT	RW	0h	Character length. Selects 7-bit or 8-bit character length. 0b = 8-bit data 1b = 7-bit data
11	UCMST	RW	0h	Master mode select 0b = Slave mode 1b = Master mode
10-9	UCMODEx	RW	0h	eUSCI mode. The UCMODEx bits select the synchronous mode when UCSYNC = 1. 00b = 3-pin SPI 01b = 4-pin SPI with UCxSTE active high: Slave enabled when UCxSTE = 1 10b = 4-pin SPI with UCxSTE active low: Slave enabled when UCxSTE = 0 11b = Reserved
8	UCSYNC	RW	0h	Synchronous mode enable 0b = Asynchronous mode 1b = Synchronous mode
7-6	UCSSELx	RW	0h	eUSCI clock source select. These bits select the BRCLK source clock. 00b = UCxCLK in slave mode. Do not use in master mode. 01b = ACLK in master mode. Do not use in slave mode. 10b = SMCLK in master mode. Do not use in slave mode. 11b = SMCLK in master mode. Do not use in slave mode.
5-2	Reserved	R	0h	Reserved
1	UCSTEM	RW	0h	STE mode select in master mode. This byte is ignored in slave or 3-wire mode. 0b = STE pin is used to prevent conflicts with other masters 1b = STE pin is used to generate the enable signal for a 4-wire slave
0	UCSWRST	RW	1h	Software reset enable 0b = Disabled. eUSCI reset released for operation. 1b = Enabled. eUSCI logic held in reset state.

```
UCA0CTLW0 |= UCMST | UCSYNC | UCCKPL | UCMSB;     // 3-pin, 8-bit SPI master
                                                  // Clock polarity high, MSB
```

其中，UCMST = 0x0800，UCSYNC = 0x0100, UCCKPL = 0x4000, UCMSB = 0x2000。

这里我们会发现，eUSCI_A在UART和SPI两种模式下，UCAxCTLW0 寄存器是不一样的。我有一个疑问：

eUSCI_A在UART和SPI两种模式是通过哪里切换的，只要PxSEL设置了两个引脚就是UART模式，四个引脚就自动进入SPI模式？

（3）选择eUSCI_A时钟源

```
UCA0CTLW0 |= UCSSEL__ACLK;              // ACLK
```

其中，UCSSEL__ACLK = 0x0040。

（4）设置设置比特率。

25.4.2 UCAxBRW Register

eUSCI_Ax Bit Rate Control Register 1

Figure 25-6. UCAxBRW Register

15	14	13	12	11	10	9	8
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw
7	6	5	4	3	2	1	0
UCBRx							
rw	rw	rw	rw	rw	rw	rw	rw

Can be modified only when UCSWRST = 1.

Table 25-4. UCAxBRW Register Description

Bit	Field	Type	Reset	Description
15-0	UCBRx	RW	0h	Bit clock prescaler setting. $f_{BitClock} = f_{BRCLK} / UCBRx$ If UCBRx = 0, $f_{BitClock} = f_{BRCLK}$

```
UCA0BR0 = 0x02;                 // /2
UCA0BR1 = 0;                    //
```

fBitClock = fBRCLK/UCBRx

（5）我在SPI的模式说明里发现没有对UCA0MCTLW这个寄存器的说明，是不是和UART模式时的寄存器一模一样。用来配置波特率参数？希望有知道的帮忙解答下这条代码的作用：

```
UCA0MCTLW = 0;                  // No modulation
```

（6）使能中断

25.4.6 UCAxIE Register

eUSCI_Ax Interrupt Enable Register

Figure 25-10. UCAxIE Register

15	14	13	12	11	10	9	8
Reserved							
r0	r0	r0	r0	r0	r0	r0	r0
7	6	5	4	3	2	1	0
Reserved						UCTXIE	UCRXIE
r-0	r-0	r-0	r-0	r-0	r-0	rw-0	rw-0

Table 25-8. UCAxIE Register Description

Bit	Field	Type	Reset	Description
15-2	Reserved	R	0h	Reserved
1	UCTXIE	RW	0h	Transmit interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled
0	UCRXIE	RW	0h	Receive interrupt enable 0b = Interrupt disabled 1b = Interrupt enabled

```
UCA0IE |= UCRXIE;                    // Enable USCI_A0 RX interrupt
```

2、处理函数（参考UART和I2C，这里不详述）

3、对于中断子程序（参考UART和I2C，这里不详述）