```
/*
 * File:   newmain.c
 * Author: soehnle
 *
 * Created on 6. August 2014, 11:55
 */


#include <stdio.h>
#include <stdlib.h>
#include<xc.h>
//#include "afe_spi.h"
#include"config.h"
#include <plib/usart.h>
#define _XTAL_FREQ 8000000
int i = 0;
unsigned char UART1Config = 0, baud = 0;
unsigned char str[100];
void SetupClock(void);
void Delay1Second(void);
void SPI_init();
void writeSPI(unsigned char addr, unsigned char data1);
void writeSPI2(unsigned char addr, unsigned char data1, unsigned char
data2);
unsigned long readSPI(unsigned char addr );

typedef struct{
    unsigned char high;
    unsigned short high1;
    unsigned char middle;
    unsigned char low;
    unsigned long all;
} spiReadData;


void SPI_init()
{
    //SSPCON1
SSPCON1bits.SSPM3 = 0;                  // 0000 - SPI Master Mode clock
FOSC/4
    SSPCON1bits.SSPM2 = 0;
    SSPCON1bits.SSPM1 = 0;
    SSPCON1bits.SSPM0 = 0;
    SSPCON1bits.SSPEN = 1;                  // enable Serial Ports,
configure SDI,SDO,SCK as serial port pins
    SSPCON1bits.CKP = 0;

    //SSP1STAT
    SSPSTATbits.SMP = 0;                    // Master - input data sampled
at middle of data output time
    SSPSTATbits.CKE = 0;                    // Transmit occurs on
transition from idle to active clock state
    PIE1bits.SSPIE = 1;                     // enable MSSP Interrupt
    PIR1bits.SSPIF = 0;                     // MSSP Interrupt Flag off
```

```
    RCONbits.IPEN = 0;                      // disable priority levels on
interrupts

    ADCON0 = 0x3C;                          // disable ADC module
    ADCON1 = 0x0F;                          // disable ADC module
//    CMCONbits.CON = 0;                     // disable Comparator module
//    SPPCON = 0x00;                          // disable Streaming
Parallel Port

    // I/O
    TRISB2 = 0;                             // Output for STE
    LATB2 = 1;                              // STE high, SPI disable
    TRISB3 = 1;                             // Input for RDY
    TRISC7 = 0;                             // SDOUT
    TRISB0 = 1;                             // SDIN
    TRISB1 = 0;                             // SCLK
}

void writeSPI(unsigned char addr, unsigned char data1)
{
    volatile unsigned char dummy;

//    LATB2 = 1;                        // set high at first for properly
working
//    LATB2 = 0;                        // take STE low, SPI enabled
    // send addr
    SSPBUF = 0b00000000;           // write modified address to SSP1BUF
register
    while(!PIR1bits.SSPIF);         // wait until bus cycle complete
    PIR1bits.SSPIF = 0;
    dummy = SSPBUF;                 // Clears BF
    // send 1st byte
    SSPBUF = 0b10011000 ;                // write byte to SSP1BUF
register
    while(!PIR1bits.SSPIF);         // wait until bus cycle complete
    PIR1bits.SSPIF = 0;
   dummy = SSPBUF;                 // Clears BF

//    LATB2 = 1;                        // take STE high, SPI disabled
}
void writeSPI2(unsigned char addr, unsigned char data1, unsigned char
data2)
{
  volatile unsigned char dummy;

//    LATB2 = 1;                        // set high at first for properly
working
//    LATB2 = 0;                        // take STE low, SPI enabled
    // send addr
    SSPBUF = 0b00010000;           // write modified address to SSP1BUF
register
    while(!PIR1bits.SSPIF);         // wait until bus cycle complete
    PIR1bits.SSPIF = 0;
   dummy = SSPBUF;                 // Clears BF
    // send 1st byte
```

```
    SSPBUF = 0b01000000;                // write byte to SSP1BUF register
    while(!PIR1bits.SSPIF);        // wait until bus cycle complete
    PIR1bits.SSPIF = 0;
  dummy = SSPBUF;                // Clears BF
    // send 2nd byte
    SSPBUF = 0b00101011;                 // write byte to SSP1BUF
register
    while(!PIR1bits.SSPIF);        // wait until bus cycle complete
    PIR1bits.SSPIF = 0;
    dummy = SSPBUF;                // Clears BF

//    LATB2 = 1;                       // take STE high, SPI disabled

}

 unsigned long readSPI(unsigned char addr)
{
    volatile unsigned char dummy;
    spiReadData res;

    // set STE low
//    LATB2 = 1;                              // set high at first for
properly working
//    LATB2 = 0;                              // reset to low for SPI to
work

    // send addr
    SSPBUF = 0b10100000;                // write modified address to
SSP1BUF register
    while(!PIR1bits.SSPIF);               // wait until bus cycle
complete
    PIR1bits.SSPIF = 0;                    // clear flag
    dummy = SSPBUF;                        // clears BF

    // read 1st byte
    SSPBUF = 0x00;                         // send dummy byte
    while(!PIR1bits.SSPIF);               // wait until bus cycle
complete
    PIR1bits.SSPIF = 0;                    // clear flag
    res.high = SSPBUF;                     // return 1st byte (HIGH)

    // read 2nd byte
    SSPBUF = 0x00;                         // initiate bus cycle
    while(!PIR1bits.SSPIF);               // wait until bus cycle
complete
    PIR1bits.SSPIF = 0;                    // clear flag
    res.middle = SSPBUF;                      // return 2nd byte (LOW)

     // read 3rd byte
    SSPBUF = 0x00;                         // initiate bus cycle
    while(!PIR1bits.SSPIF);               // wait until bus cycle
complete
    PIR1bits.SSPIF = 0;                    // clear flag
    res.low = SSPBUF;                      // return 3rd byte (LOW)
```

```
    (unsigned short)res.high1 = (((short)res.middle) << 8) | res.low;
// combine 2 bytes into 1 word
(unsigned long)res.all = (((long)res.high) << 16) | res.high1;
    // set STE high
//    LATB2 = 1;                                // set STE high again to
disable SPI



    return res.all;
}


void main()
{
    unsigned long a;

    SetupClock(); // Internal Clock to 8MHz

    TRISCbits.RC6 = 1; //TX pin set as output
    TRISCbits.RC7 = 1; //RX pin set as input
    RCSTA1bits.SPEN = 1;//activate srial port
    TXSTAbits.TXEN=1;// enable transmission

    UART1Config = USART_TX_INT_OFF &
                USART_RX_INT_OFF &
                USART_ASYNCH_MODE &
                USART_EIGHT_BIT &
                USART_SYNC_MASTER & // Synchronous Master mode
                USART_CONT_RX &     // Continuous reception
                USART_BRGH_LOW ;
    baud = 12;
    OpenUSART(UART1Config,baud);
//sprintf(str," hello \r\n");
//        putsUSART(str);
//    unsigned char i;
//    i = 0;
    while(1) //infinite loop
    {

    writeSPI(0b00000000,0b10011000);
     writeSPI2(0b00010000,0b01000000,0b00101011);
     Delay1Second();

 a = readSPI(0b10100000);

   sprintf(str," %lu \r\n", a);
        putsUSART(str);

//        i++;

        Delay1Second();
    }

}
```

```
void SetupClock()
{
    OSCCONbits.IRCF0 = 1;
    OSCCONbits.IRCF1 = 1;
    OSCCONbits.IRCF2 = 1;
}

void Delay1Second()
{
    for(i=0;i<100;i++)
    {
         __delay_ms(10);
    }
}
```