```c
/*
 * Terminal_to_MAX5402.c
 *
 * Created: 3/14/2022 10:03:48 PM
 * Author : jason
 */

#define F_CPU 4000000
#define USART3_BAUD_RATE(BAUD_RATE) ((float)(F_CPU * 64 / (16 *(float)BAUD_RATE)))  //
  Calculation of baud rate from data sheet
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>


/* UART Buffer Defines */
#define USART_RX_BUFFER_SIZE 16     /* 2,4,8,16,32,64,128 or 256 bytes */
#define USART_RX_BUFFER_MASK ( USART_RX_BUFFER_SIZE - 1 )

#if ( USART_RX_BUFFER_SIZE & USART_RX_BUFFER_MASK )
#error RX buffer size is not a power of 2
#endif

/* Static Variables */
static unsigned char USART_RxBuf[USART_RX_BUFFER_SIZE];
static volatile unsigned char USART_RxHead;
static volatile unsigned char USART_RxTail;
char c;

/* Function Prototypes */
void USART3_Init( unsigned int baudrate );
unsigned char USART3_Receive( void );
void MAX5402_SPI0_write(uint8_t);


int main(void)
{
    unsigned int baudRate = 9600; //Baud rate value

    USART3_Init(baudRate);  //function to initialize for USART
    sei();   //Enable interrupts => enable USART3 interrupts
    while(1)
    {
        c = USART3_Receive();  //The data that will be received from the buffer

        //Check if there's data to write to the SPI0
        if(c != ' '){
            MAX5402_SPI0_write(c);
        }
    }
}
```

```
//Initialize USART and the SPI configuration
void USART3_Init(unsigned int baudrate){
    unsigned char x;

    PORTF_DIR |= PIN2_bm;   //Chose PF2 as the output to drive the /CS input
    PORTA_DIR |= PIN6_bm | ~(PIN5_bm) | PIN4_bm ;          //Configure as output where
       the pin level is controlled by the SPI
    SPI0_CTRLA = SPI_MASTER_bm | SPI_CLK2X_bm | SPI_ENABLE_bm;  //Enable the SPI and
      also Select the SPI master/slave operation by writing the Master/Slave Select
    //(MASTER) bit in the Control A (SPIn.CTRLA) register
    SPI0_CTRLB |= SPI_SSD_bm;   //Enable  the SPI by writing a 1 to the enable bit



    PORTB_DIR &= PIN1_bm;  //Set PB1 as the input (RX pin)
    USART3.BAUD = (uint16_t)USART3_BAUD_RATE(baudrate);   //Taken from data sheet to
      calculate baud rate
    USART3.CTRLB |= USART_RXEN_bm;  //Enable USART receiver
    USART3.CTRLA |= USART_RXCIE_bm;   //Enable the Receive complete interrupt

    //Guess set the default to being asynchronous, disable parity, 1 stop bit, 8 bits

    //Flush receive buffer
    x = 0;

    USART_RxTail = x;
    USART_RxHead = x;
}

//Interrupt service routine for the receive that will see if there's data in the
  RX_buffer
ISR(USART3_RXC_vect){
    unsigned char data;
    unsigned char tmphead;

    //Read the received data
    data = USART3_RXDATAL;

    /*Calculate the buffer index */
    tmphead = (USART_RxHead + 1) & USART_RX_BUFFER_MASK;
    USART_RxHead = tmphead;   //Store new index

    if(tmphead == USART_RxTail){

    }
    USART_RxBuf[tmphead] = data;  //Store received data in buffer
}

//Read from the RX_buffer
unsigned char USART3_Receive(void){
    unsigned char tmptail;
```

```
    while(USART_RxHead == USART_RxTail);  //Wait for incoming data

    tmptail = (USART_RxTail + 1) & USART_RX_BUFFER_MASK;  //Calculate buffer index

    USART_RxTail = tmptail;  //Store new index

    return USART_RxBuf[tmptail];    //return data

}



/*
the value to be send to the MAX5402 to set the position of the its wiper.
It is important that before this function returns, it must deselect the
MAX5402. If this is not done, there could be a subsequent SPI bus
conflict between the MAX5402 and other (future) SPI devices added to
the system.

*/
void MAX5402_SPI0_write(uint8_t data){

    PORTF_OUT &= ~PIN2_bm;   //Set to be 0 when transmission is happening

    SPI0_DATA = data;  //Write the data to the MISO input

    //Poll until the  transmit buffer register are empty
    //when they contain data that has not been moved to
    //transmit shift register
    while ((SPI0_INTFLAGS & SPI_IF_bm) == 0x00)
    {
        ;
    }

    PORTF_OUT |= PIN2_bm;//Set to be 1 when transmission is done

}


//Data received in the buffer
unsigned char DataInReceiveBuffer(void){
    /* Return 0 (FALSE) if the receive buffer is empty */
    return (USART_RxHead != USART_RxTail);
}
```