STONY BROOK UNIVERSITY
DEPARTMENT OF COMPUTER AND ELECTRICAL ENGINEERING

ESE 381.L02

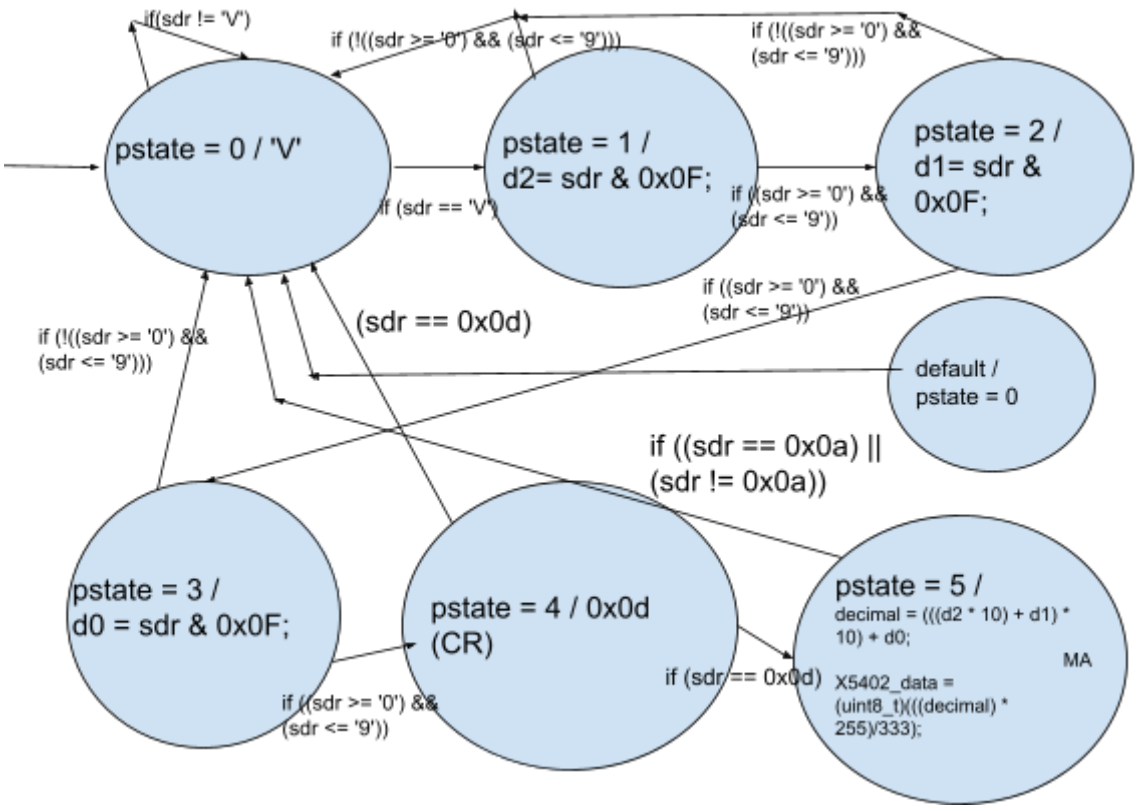
Lab 7: Asynchronous Serially Controlled Digital Potentiometer with SPI Interface

Name: Jason Tan
SBU ID #: 112319102
Due Date: March 24, 2022 by 9PM

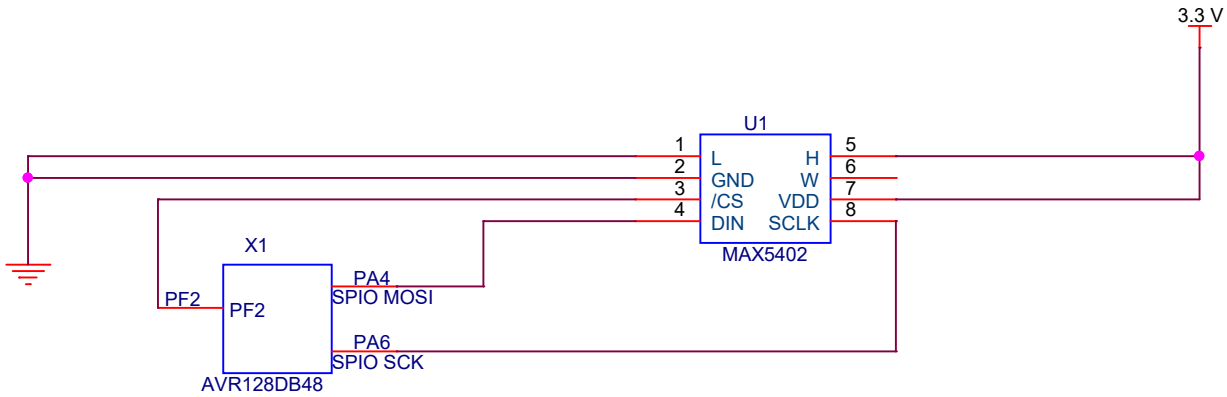
Verification Strategy

Task 1:

The way that I will verify for task 1 is I will use the Saleae logic analyzer and place the three probes of the Salae into PA4 (MOSI), PA6 (SCK), and PF2 (SS) to check on whether the SPI is working as expected. After wiring the connection to the MAX5402, I will check on the W (wiper) pin to see if there is a ramp which would mean that the value is being incremented by 1 from 0 until to 255 as expected to see if the SPI operation works as intended when verifying. This is what is expected according to one of the laboratory 7 lecture slides showing the professor's ramp function



Task 3 **state diagram for the finite state machine (FSM) according to given case statement**



Title	Lab 7 Schematic	
Size A	Document Number <Doc>	Rev <RevCo
Date: Wednesday, March 16, 2022	Sheet 1 of 1	

```c
/*
 * MAX5402_verify.c
 *
 * Created: 3/14/2022 9:53:50 PM
 * Author : jason
 */


#include <avr/io.h>
#define F_CPU 4000000
#include <util/delay.h>

//All the header funnctions
void MAX5402_SPI0_init(void);
void MAX5402_SPI0_write(uint8_t);




int main(void)
{
    uint8_t i = -1;
    MAX5402_SPI0_init();  //Initialize the SPI0 module
    while (1)
    {
            if(i < 256){
                MAX5402_SPI0_write(i++);      //Write the data to the SPI of the
                  incremented value

            }
            else{
                //Reset the value to 0 if trying to exceed the value of 255
                i = 0;
                MAX5402_SPI0_write(i);
            }
        //  _delay_ms(1);
        //}
    }
}


//initializes the AVR128DB48's SPI0 module to communicate with the
//MAX5402. SPI0 must be configured in the Normal mode (no buffering).
//Thus, this function must check that the previous transfer is complete
//after writing each byte of data and then deselect the MAX5402 before
//returning. Note that there are two variations of the INTFLAGS register.
// One for use in Normal mode and the other for use in Buffered mode
void MAX5402_SPI0_init(void){

    PORTF_DIR |= PIN2_bm;   //Chose PF2 as the output to drive the /SS input
    PORTA_DIR |= PIN6_bm | PIN4_bm ;          //Configure PA6 and PA4 output where the
       pin level is controlled by the SPI
```

```
    PORTA_DIR &= ~PIN5_bm;   //Configure PA5 as the input

     //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.CTRLA |= SPI_ENABLE_bm | SPI_MASTER_bm | SPI_PRESC_DIV4_gc;

    SPI0_CTRLB |= SPI_SSD_bm;   //Disable the Slave select since we don't need the
      actual /SS


    PORTF_OUT |= PIN2_bm;   //Initialize the /SS to be 1 meaning no data to be sent
      yet


}




//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;

    //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;

}
```

```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);
}
```

```
/*
 * ASCII_str_to_MAX5402.c
 *
 * Created: 3/18/2022 9:10:56 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


//The switch statement labeled FSM creates a FSM to parse the command
//string received in Task 3. You will have to analyze its operation
//to answer some of the questions for this laboratory.

//You will need to include the following declarations in your code
//as global variables. Accordingly, place the outside of all functions.
unsigned char sdr;  //serial data received
uint8_t MAX5402_data;   //data to be written to MAX5402
uint8_t pstate = 0; //present state
uint8_t d2, d1, d0; //digits of the decimal value received
uint32_t decimal;       //binary value equal to decimal value received

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

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

int i= 1;  //To loop through the RX array to get the character that will be parsed
int j = 0;  //Number of iterations to get the 6 ASCII values after user entered the desired format
char c;
int receiveFlag = 0;
```

```
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)
    {
//      _delay_us(10000);
        //_delay_us(10100);
        if(!(VPORTB_IN & PIN2_bm) || (VPORTC_IN & PIN2_bm)){
            //Delay loop to get all the characters entered by user from 'V' to 0x0A
            for(int k= 0 ; k < 10000; k++){
                asm volatile("nop");
            }
        //  asm volatile("nop");
        //  asm volatile("nop");
        //  asm volatile("nop");
        //  asm volatile("nop");
        //  asm volatile("nop");

            //Loop to get the characters that will be parsed from the RX_buffer
            while(j < 6){
                if(i < 16){
                    FSMFunction(USART_RxBuf[i]);
                    i++;
                }
                else{
                    i = 0;
                    FSMFunction(USART_RxBuf[i]);
                    i++;
                }
                j++;
            }
            j = 0;
        }
        //c = USART3_Receive();

    //  if((USART3_STATUS & USART_RXCIF_bm)){
        //  FSMFunction(USART3_Receive());
    //      asm volatile("nop");
    //  }

        PORTC_OUT &= ~PIN2_bm;


        //Poll to ensure there is data to read from the receive buffer
        //while(!(USART3_STATUS & USART_RXCIF_bm)){}
            //Check if there's data to write to the SPI0
//      while(!(USART3_STATUS & USART_RXCIF_bm)){}
```

```
    //  }
    }
}


//Initialize USART and also 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

    PORTC_DIR |= PIN2_bm;  //Dummy pin to know when there's data to be received to be
      sent to the SPI

    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 receiving data from the Termite 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){

    }
    PORTC_OUT |= PIN2_bm;
    USART_RxBuf[tmphead] = data;  //Store received data in buffer
```

```
}


//The function for doing the FSM and parsing the characters in the correct format
  entered by the user
void FSMFunction(uint8_t sdr1){
    switch (pstate)
    {
        case 0:
            //Read the received data
            //sdr = USART3_RXDATAL;
            if (sdr1 == 'V'){
                pstate = 1;
            }
            else
            pstate = 0;
            break;



        case 1:

            if ((sdr1 >= '0') && (sdr1 <= '9'))
            {
                d2 = sdr1 & 0x0F;
                pstate = 2;
            }
            else
                pstate = 0;
                break;



        case 2:
            if ((sdr1 >= '0') && (sdr1 <= '9'))
            {
                d1 = sdr1 & 0x0F;
                pstate = 3;
            }
            else
                pstate = '0';
            break;

        case 3:
            if ((sdr1 >= '0') && (sdr1 <= '9'))
            {
                d0 = sdr1 & 0x0F;
                pstate = 4;
            }
            else
                pstate = 0;
            break;
```

```
        case 4:
            if (sdr1 == 0x0d)
                pstate = 5;
            else
                pstate = 0;
            break;

        case 5:
            if (sdr1 == 0x0a)
            {
                pstate = 0;
                decimal = (((d2 * 10) + d1) * 10) + d0;
                MAX5402_data = (uint8_t)(((decimal) * 255)/333);
                MAX5402_SPI0_write(MAX5402_data);  //Send the compute the binary value
                   to be sent to the
                //MAX5402 to output this voltage value
            }
            else
                pstate = 0;
            break;

        default:
            pstate = 0;
    }
}



//Read and write function
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

    //Poll until the  transmit buffer register are empty
    //when they contain data that has not been moved to
    //transmit shift register


    SPI0_DATA = data;

    //Poll to check if the sending of Serial data to the slave is done for the SPI
    while ((SPI0_INTFLAGS & SPI_IF_bm) == 0x00)
    {
        ;
    }

    PORTF_OUT |= PIN2_bm;

}


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