STONY BROOK UNIVERSITY
DEPARTMENT OF COMPUTER AND ELECTRICAL ENGINEERING

ESE 381.L02

Lab 6: Circular Buffers and AVR128DB48 USART Module in Asynchronous Serial (RS232) Mode

Name: Jason Tan
SBU ID #: 112319102
Due Date: March 10, 2022 by 9PM

```c
/*
 * usart3_init_test.c
 *
 * Created: 3/3/2022 12:23:17 PM
 * Author : jason
 */

#include <avr/io.h>

#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>

//Header function
void USART_sw_write(char c);
void USART3_init (uint16_t baud, uint8_t data_bits, unsigned char parity);


int main(void)
{
    /* Replace with your application code */

    uint16_t baudRate = 9600;   //For the baud rate of USART3
    uint8_t dataBits = USART_CHSIZE_8BIT_gc;    //For the (character size)  CHSIZE [2:0]
    unsigned char parity = 0x00;    //PMODE[1:0]

    USART3_init(baudRate, dataBits, parity);

    while (1)
    {
        //Send a character to the Tera Term (TX pin)
        USART_sw_write('U');
        _delay_ms(2);

    }
}

//In Laboratory 05 you were not required to organize your program using functions, even though
//this approach was discussed in class. A simple function to configure a USART might have a single parameter that specifies the desired baud rate.
//The function that you must write for this task goes further than that, it allows both baud rate and
//the frame format to be specified
void USART3_init (uint16_t baud, uint8_t data_bits, unsigned char parity){
    PORTB_DIR |= PIN0_bm;  //To transmit the data

    //Specify the baud rate value for the USART3
    USART3.BAUD = (uint16_t)USART3_BAUD_RATE(baud);
```

```
    //Initialize the data bits and the parity bits type
    USART3_CTRLC |= data_bits | parity;
    USART3.CTRLB |= USART_TXEN_bm;  //Enable USART transmitter
}


//Function to be able to transmit characters
//to the TX pin and display on the Tera Term
void USART_sw_write(char c)
{
    //Poll until the  transmit buffer register are empty
    //when they contain data that has not been moved to
    //transmit shift register
    while (!(USART3.STATUS & USART_DREIF_bm))
    {
        ;
    }

    //Load data to transmit shift register and
    //output each of the bits serially to the TXD pin
    USART3.TXDATAL = c;
}
```

```c
/*
 * usart_avr128.c
 *
 * Created: 3/3/2022 4:20:17 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 128     /* 2,4,8,16,32,64,128 or 256 bytes */
#define USART_TX_BUFFER_SIZE 128     /* 2,4,8,16,32,64,128 or 256 bytes */
#define USART_RX_BUFFER_MASK ( USART_RX_BUFFER_SIZE - 1 )
#define USART_TX_BUFFER_MASK ( USART_TX_BUFFER_SIZE - 1 )

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

#if ( USART_TX_BUFFER_SIZE & USART_TX_BUFFER_MASK )
    #error TX 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;
static unsigned char USART_TxBuf[USART_TX_BUFFER_SIZE];
static volatile unsigned char USART_TxHead;
static volatile unsigned char USART_TxTail;
volatile int flag;

/* Prototypes */
void USART3_Init( unsigned int baudrate );
unsigned char USART3_Receive( void );
void USART3_Transmit( unsigned char data );


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)
    {
```

```
        USART3_Transmit(USART3_Receive()); /* Echo the received character */
    }
}


//Initialize USART
void USART3_Init(unsigned int baudrate){
    unsigned char x;
    PORTB_DIR &= PIN1_bm;  //Set PB1 as the input (RX pin)
    PORTB.DIR |= PIN0_bm;  //Set PB0 as output (TX pin)
    USART3.BAUD = (uint16_t)USART3_BAUD_RATE(baudrate);   //Taken from data sheet to
      calculate baud rate
    USART3.CTRLB |= USART_TXEN_bm | USART_RXEN_bm;  //Enable USART transmitter and
      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;
    USART_TxTail = x;
    USART_TxHead = x;
}

//Interrupt service routine for the receive
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
}

//Interrupt service routine for the transmit
ISR(USART3_DRE_vect){
    unsigned char tmptail;

    //Check if all data is transmitted
    if(USART_TxHead != USART_TxTail){
        //Calculate buffer index
```

```
        tmptail = (USART_TxTail + 1) & USART_TX_BUFFER_MASK;
        USART_TxTail = tmptail;   //Store new index

        USART3_TXDATAL = USART_TxBuf[tmptail];  //Starts transmission
    }
    else{
        //Disable the USART data register empty interrupt enable
        USART3_CTRLA &= ~(1 << USART_DREIE_bp);  //#define USART_DREIE_bp  5  /* Data
          Register Empty Interrupt Enable bit position. */
    }
}

//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

}

//Transmit the data
void USART3_Transmit(unsigned char data){
    unsigned char tmphead;

    //Calculate buffer index
    tmphead = (USART_TxHead + 1) & USART_TX_BUFFER_MASK;  //Wait for free space in the
       buffer

    while(tmphead == USART_TxTail);

    USART_TxBuf[tmphead] = data;   //Store data in buffer
    USART_TxHead = tmphead;     //Store new index

    USART3_CTRLA |= (1<< USART_DREIE_bp);    //Enable the DREIE interrupt

}


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