```c
/*
 * interrupt_echo_line.c
 *
 * Created: 2/21/2022 10:48:11 PM
 * Author : jason
 */

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

#define BAUD_RATE 9600  //baud rate value 1


//header functions that will be in use
uint8_t UART_sw_read();
void UART_sw_write(char);
void sendString(char*);

//global character variable to store the recent data
char tempReceiveData;
uint8_t startBit = 1;
uint8_t rxData;


//Interrupt function executes when Receive Complete Interrupt is enabled or set
ISR(PORTB_PORT_vect){
    char receivedData[80];
    uint8_t i = 0;
    while(i < 80){
        tempReceiveData = (char)UART_sw_read();   //Execute the sw read function
        if(tempReceiveData == 0x0D){
            break;               //terminate the loop if the carriage character 'CR'
              is returned
        }
        else{
            receivedData[i] = tempReceiveData;     //send character into the array
        }
        i++;
    }
    sendString(receivedData);       //Execute the function to send
    PORTB_INTFLAGS |= PORT_INT1_bm;       //Clear the interrupt for
}


int main(void)
{
    //unsigned char receiveData;
    //USART Full Duplex Initialization
    PORTB.DIR = PIN0_bm | ~PIN1_bm;   //Set  PB1 as input to receiving data and PB0 as
```

```
       the output for transmitting data

    PORTB_PIN1CTRL = PORT_ISC_FALLING_gc;   //Edge trigger at the falling edge for an
      interrupt
    sei();    //Enables global interrupts

    while(1){
        //a nop so while loop is not optimized away
        asm volatile ("nop");
    }

}


void sendString(char* sendLine){
    for(uint8_t i = 0; i < strlen(sendLine); i++){
        UART_sw_write(sendLine[i]); //write the entire message to a line
    }
    UART_sw_write("\r\n");   //To pass into a next line
}


void UART_sw_write(char c){
    uint8_t tempData = (uint8_t) c;   // pass in to the tempData variable
    PORTB_OUT &= ~PIN0_bm; // Send the start bit for PB0


    //Set the bit times for sending the data
    if(BAUD_RATE == 4800){
        _delay_us(208.3);
    }
    else if(BAUD_RATE == 9600){
        _delay_us(104.2);
    }
    else if(BAUD_RATE == 19200){
        _delay_us(52.1);
    }


    //Do some kind of a loop to shift right to PB0 to put into that pin output
    for(uint8_t i = 0; i < 8; i++){
        PORTB_OUT = tempData & 0x01;   //Mask to get the LSB to pass to PB0
        tempData >>= 1;         //shift right by 1

        //Set the bit times for sending the data
        if(BAUD_RATE == 4800){
            _delay_us(208.3);
        }
        else if(BAUD_RATE == 9600){
            _delay_us(104.2);
        }
        else if(BAUD_RATE == 19200){
```

```
            _delay_us(52.1);
        }

    }

    PORTB_OUT = PIN0_bm;   //Send the stop bit

    //Set the bit times for sending the data
    if(BAUD_RATE == 4800){
        _delay_us(208.3);
    }
    else if(BAUD_RATE == 9600){
        _delay_us(104.2);
    }
    else if(BAUD_RATE == 19200){
        _delay_us(52.1);
    }
}




/*
The USART receiver samples the RX line to detect and
interpret the received data. The RX pin must be configured
as an input.

? The receiver accepts data when its hardware detects a valid
start bit. The data is shifted into the Receive Shift register
until the first Stop bit is received.

? The contents of the Receive Shift register is loaded into the
receive buffer and the Receive Complete Interrupt Flag (the
RXCIF bit in the USARTn.STATUS register) is set.

? The RXDATA registers are the part of the double-buffered
RX buffer that can be read by the application software when
RXCIF is set.

*/

uint8_t UART_sw_read(){
    uint8_t tempStoreReceiveData;
    uint8_t ReceiverData [8];

    startBit = 0;       //Start having the data being received by setting it to 0
    //Sample at falling edge when RX pin is 0 and check start bit is 0
    while(startBit == 0){

        //Purpose is to check the false start at half the start bit
        if(BAUD_RATE == 4800){
            _delay_us(208.3/2);
```

```
        }
        if(BAUD_RATE == 9600){
            _delay_us(104.2/2);
        }
        if(BAUD_RATE == 19200){
            _delay_us(52.1/2);
        }

        //Check in the middle of bit time if PB1 is actually 0 or not. If PB1 is not
          0, else go back in the beginning.
        if(startBit != 0){
            continue;
        }

        //False start to check the conditions if true or just wait till next start bit
        while(startBit == 0){

            //Bit time to be able to read data from the RX pin
            if(BAUD_RATE == 4800){
                _delay_us(208.3);
            }
            if(BAUD_RATE == 9600){
                _delay_us(104.2);
            }
            if(BAUD_RATE == 19200){
                _delay_us(52.1);
            }


            //Bit time for corresponding baud ratings
            for(uint8_t i = 0; i < 8; i++){

                //Try to read each bit that is stored into PB1 to receive
                tempStoreReceiveData = (PORTB_IN & 0x02);

                //Align so that the bit data fetched from PB1 only exist in bit 0
                  position
                tempStoreReceiveData >>= 1;

                //Mask to only care about the one bit bit every time the bit from PB1
                  is read
                ReceiverData[i] = (tempStoreReceiveData & 0x01);


                //Bit time to receive each bit that is transmitted
                if(USART3.BAUD == 4800){
                    _delay_us(208.3);
                }
                if(USART3.BAUD == 9600){
                    _delay_us(104.2);
                }
                if(USART3.BAUD == 19200){
```

```
                    _delay_us(52.1);
                }
            }

            //Store all of the data bits read from RX pin into RX output
            rxData = ReceiverData[0] << 0 | ReceiverData[1] << 1 | ReceiverData[2] << 
              2 |
            ReceiverData[3] << 3 | ReceiverData[4] << 4 | ReceiverData[5] << 5 |
            ReceiverData[6] << 6 | ReceiverData[7] << 7;

            startBit = 1;
        }
        startBit = 1;
    }
    startBit = 1;
    return (char) rxData;
}
```