Judah Ben-Eliezer

112352727
3/18/2021

Prelab 6:

Software Implementation of a UART Transmitter and Receiver and Use of a Saleae Logic Analyzer and Tera Term Terminal Emulator

```c
/*
 * asynchronous_sw_send.c
 *
 * Created: 3/18/2021 4:31:07 PM
 * Author : Judah Ben-Eliezer
 */

#define BAUD_RATE 4800UL                                                    //  baud rate.
#define F_CPU 4000000UL                                                     // 4 MHz clock.
#include <avr/io.h>
#include <util/delay.h>

void USART_sw_write(char);                                                  //  declaration for write function.

int main(void)
{
    while (1)
    {
        USART_sw_write("A");                                                //  write character.
        _delay_ms(1);                                                       //  delay 1 ms.
    }
}

void USART_sw_write(char c) {
    PORTB.DIRSET = PIN0_bm;                                                 //  set PB0 as output.
    uint8_t d;                                                              //  bit time.
    if (BAUD_RATE == 4800L) {
        d = 48;
    } else if (BAUD_RATE == 9600L) {
        d = 99;
    } else if (BAUD_RATE == 19200L) {
        d = 201;
    } else return 0x00;

    uint8_t data = (uint8_t) c;

    PORTB_OUT = 0x00 | PIN0_bm;                                             //  send start bit.
    _delay_us(d);                                                           //  delay for bit time.

    uint8_t i;
```

```
41      for (i = 0; i < 8; ++i) {
42          PORTB_OUT = data | PIN0_bm;                                    //  ↵
              send lsb of data.
43          data >>= data;                                                 //  ↵
              shift data right.
44          _delay_us(d);                                                  //  ↵
              delay for bit time.
45      }
46
47      PORTB_OUT = PIN0_bm;                                               //  ↵
          send end bit.
48      _delay_us(d);                                                      //  ↵
          delay for bit time.
49  }
50
51
```

```c
/*
 * asynchronous_sw_read.c
 *
 * Created: 3/18/2021 5:47:32 PM
 * Author : Judah Ben-Eliezer
 */

#define BAUD_RATE 4800L                                                 // baud rate.
#define F_CPU 4000000UL                                                 // 4 MHz clock.
#include <avr/io.h>
#include <util/delay.h>

uint8_t USART_sw_read();                                                //  declaration for read function.

char c;                                                                 //  global char variable.

int main(void)
{
    while (1)
    {
        c = (char) USART_sw_read();                                     // read UART input to c.
    }
}

uint8_t USART_sw_read() {
    PORTB.DIRCLR = PIN1_bm;                                             // PB1 set as input.

    uint8_t d;                                                          // bit time.
    if (BAUD_RATE == 4800L) {
        d = 48;
    } else if (BAUD_RATE == 9600L) {
        d = 99;
    } else if (BAUD_RATE == 19200L) {
        d = 201;
    } else return 0x00;

    uint8_t data = 0;
    uint8_t reading = 1;

    while (reading == 1) {
        while ((PORTB_IN & PIN1_bm) == 1) {}                                //  wait for falling edge.
```

```
42          _delay_us(d/2);
43          if ((PORTB_IN & PIN1_bm) != 0) continue;                    //  ↪
              check for false start.
44          _delay_us(d);                                               //  ↪
              delay for bit time.
45
46          uint8_t i = 0;
47          for (i; i < 8; ++i) {
48              data >>= data | ((PORTB_IN | PIN1_bm) << 6);             //  ↪
                  read little endian input.
49              _delay_us(d);                                           //  ↪
                  delay for bit time.
50          }
51          reading = 0;
52      }
53      return data;
54  }
55
```

```c
/*
 * asynchronous_sw_read_interrupt.c
 *
 * Created: 3/18/2021 6:03:31 PM
 * Author : Judah Ben-Eliezer
 */

#define BAUD_RATE 4800UL                                    //  baud rate.
#define F_CPU 4000000UL                                     //  4MHz clock.

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

uint8_t USART_sw_read();                                    //  declaration for read function.

int main(void)
{
    PORTB.DIRCLR = PIN1_bm;                                 //  set PB1 as input.
    PORTB.PIN1CTRL |= PORT_ISC_FALLING_gc;                  //  enable interrupt on falling edge.
    sei();                                                  //  enable global interrupts.

    while (1)
    {
        asm volatile ("nop");                               //  nop to avoid optimization deletion.
    }
}

ISR (PORTB_PORT_vect) {
    c = USART_sw_read();                                    //  call USART_sw_read.
    PORTB.INTFLAGS |= PIN1_bm;                              //  clear interupt.
}

uint8_t USART_sw_read() {

    uint8_t d;                                              //  bit time.
    if (BAUD_RATE == 4800UL) {
        d = 48;
    } else if (BAUD_RATE == 9600UL) {
```

```
40          d = 99;
41      } else if (BAUD_RATE == 19200UL) {
42          d = 201;
43      } else return 0x00;
44
45      uint8_t data = 0;
46
47      _delay_us(d/2);
48      if ((PORTB_IN & PIN1_bm) != 0) return 0x00;                    //  ↵
          check for false start.
49      _delay_us(d);                                                  //  ↵
          delay for bit time.
50
51      uint8_t i;
52      for (i = 0; i < 8; ++i) {
53          data >>= data | ((PORTB_IN | PIN1_bm) << 6);               //  ↵
              read little endian input into data.
54          _delay_us(d);                                              //  ↵
              delay for bit time.
55      }
56
57      return data;
58  }
59
60
```

```c
/*
 * interrupt_echo.c
 *
 * Created: 3/18/2021 7:00:09 PM
 * Author : Judah Ben-Eliezer
 */

#define BAUD_RATE 4800UL                                                    //  ↵
  baud rate.
#define F_CPU 4000000UL                                                     //  ↵
  clock at 4 MHz.

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

uint8_t USART_sw_read();                                                    //  ↵
  read function declaration.
void USART_sw_write(char);                                                  //  ↵
  write function declaration.

char c;

int main(void)
{
    PORTB.DIRCLR = PIN1_bm;                                                 //  ↵
      set PB1 as input.
    PORTB.PIN1CTRL |= PORT_ISC_FALLING_gc;                                  //  ↵
      enable interrupt on falling edge of PB1.
    sei();                                                                  //  ↵
      enable global interrupts.

    while (1)
    {
        asm volatile ("nop");                                               //  ↵
          nop to avoid optimization deletion of while loop.
    }
}

ISR (PORTB_PORT_vect) {
    c = USART_sw_read();                                                    //  ↵
      call USART_sw_read.
    USART_sw_write(c - 0x20);                                               //  ↵
      write uppercase c.
    PORTB.INTFLAGS |= PIN1_bm;                                              //  ↵
      clear interrupt.
}

uint8_t USART_sw_read() {
```

```
39
40      uint8_t d;                                                      //  ↵
          bit time.
41      if (BAUD_RATE == 4800UL) {
42          d = 48;
43      } else if (BAUD_RATE == 9600UL) {
44          d = 99;
45      } else if (BAUD_RATE == 19200UL) {
46          d = 201;
47      } else return 0x00;
48
49      uint8_t data = 0;
50
51      _delay_us(d/2);
52      if ((PORTB_IN & PIN1_bm) != 0) return 0x00;                     //  ↵
          check for false start.
53      _delay_us(d);                                                   //  ↵
          delay for bit time.
54
55      uint8_t i;
56      for (i = 0; i < 8; ++i) {
57          data >>= data | ((PORTB_IN | PIN1_bm) << 6);                //  ↵
            read little endian input into data.
58          _delay_us(d);                                               //  ↵
            delay for bit time.
59      }
60
61      return data;
62  }
63
64  void USART_sw_write(char c) {
65      PORTB.DIRSET = PIN0_bm;                                         //  ↵
          set PB0 as output.
66      uint8_t d;                                                      //  ↵
          bit time.
67      if (BAUD_RATE == 4800L) {
68          d = 48;
69      } else if (BAUD_RATE == 9600L) {
70          d = 99;
71      } else if (BAUD_RATE == 19200L) {
72          d = 201;
73      } else return;
74
75      uint8_t data = (uint8_t) c;
76
77      PORTB_OUT = 0x00 | PIN0_bm;                                     //  ↵
          send start bit.
78      _delay_us(d);                                                   //  ↵
          delay for bit time.
```

```
79
80      uint8_t i;
81      for (i = 0; i < 8; ++i) {
82          PORTB_OUT = data | PIN0_bm;                                    //  ↪
              send lsb of data.
83          data >>= data;                                                 //  ↪
              shift data right.
84          _delay_us(d);                                                  //  ↪
              delay for bit time.
85      }
86
87      PORTB_OUT = PIN0_bm;                                               //  ↪
          send end bit.
88      _delay_us(d);                                                      //  ↪
          delay for bit time.
89  }
90
91
```

```c
/*
 * interrupt_echo_line.c
 *
 * Created: 3/18/2021 7:33:19 PM
 * Author : Judah Ben-Eliezer
 */

#define BAUD_RATE 4800UL                                                    //  ↵
  baud rate.
#define F_CPU 4000000UL                                                     //  ↵
  clock at 4 MHz.

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

uint8_t USART_sw_read();                                                    //  ↵
  read function declaration.
void USART_sw_write(char);                                                  //  ↵
  write function declaration.

char c[80];                                                                 //  ↵
  buffer.
uint8_t i = 0;

int main(void)
{
    PORTB.DIRCLR = PIN1_bm;                                                 //  ↵
      set PB1 as input.
    PORTB.PIN1CTRL |= PORT_ISC_FALLING_gc;                                  //  ↵
      enable interrupt on falling edge of PB1.
    sei();                                                                  //  ↵
      enable global interrupts.

    while (1)
    {
        asm volatile ("nop");                                               //  ↵
          nop to avoid optimization deletion of while loop.
    }
}

ISR (PORTB_PORT_vect) {
    c[i++] = USART_sw_read();                                               //  ↵
      call USART_sw_read.
    if (c[i] == 0x0D) {                                                     //  ↵
      check for CR.
        uint8_t j;
        for (j = 0; j <= i; ++j) {
            USART_sw_write(c[j]);                                           //  ↵
```

```
                write line.
39          }
40          i = 0;                                                  //  ↵
              clear buffer
41      }
42      PORTB.INTFLAGS |= PIN1_bm;                                  //  ↵
          clear interrupt.
43  }
44
45  uint8_t USART_sw_read() {
46
47      uint8_t d;                                                  //  ↵
          bit time.
48      if (BAUD_RATE == 4800UL) {
49          d = 48;
50      } else if (BAUD_RATE == 9600UL) {
51          d = 99;
52      } else if (BAUD_RATE == 19200UL) {
53          d = 201;
54      } else return 0x00;
55
56      uint8_t data = 0;
57
58      _delay_us(d/2);
59      if ((PORTB_IN & PIN1_bm) != 0) return 0x00;                 //  ↵
          check for false start.
60      _delay_us(d);                                               //  ↵
          delay for bit time.
61
62      uint8_t i;
63      for (i = 0; i < 8; ++i) {
64          data >>= data | ((PORTB_IN | PIN1_bm) << 6);            //  ↵
              read little endian input into data.
65          _delay_us(d);                                           //  ↵
              delay for bit time.
66      }
67
68      return data;
69  }
70
71  void USART_sw_write(char c) {
72      PORTB.DIRSET = PIN0_bm;                                     //  ↵
          set PB0 as output.
73      uint8_t d;                                                  //  ↵
          bit time.
74      if (BAUD_RATE == 4800L) {
75          d = 48;
76      } else if (BAUD_RATE == 9600L) {
77          d = 99;
```

```
78      } else if (BAUD_RATE == 19200L) {
79          d = 201;
80      } else return;
81
82      uint8_t data = (uint8_t) c;
83
84      PORTB_OUT = 0x00 | PIN0_bm;                                    //
          send start bit.
85      _delay_us(d);                                                  //
          delay for bit time.
86
87      uint8_t i;
88      for (i = 0; i < 8; ++i) {
89          PORTB_OUT = data | PIN0_bm;                                //
              send lsb of data.
90          data >>= data;                                             //
              shift data right.
91          _delay_us(d);                                              //
              delay for bit time.
92      }
93
94      PORTB_OUT = PIN0_bm;                                           //
          send end bit.
95      _delay_us(d);                                                  //
          delay for bit time.
96  }
97
98
```

Verification Strategy:

For part 1:
Verification of transmission is viable with either the Saleae logic analyzer or the Tera Term dumb terminal emulator. To verify transmission, I will have the transmitter send a continuous string of A's, and verify that the Tera Term is receiving them all. To make sure that none are missed, I will either increase the delay so as to check each transmission or open it with the Saleae logic analyzer to look at the waveform generated and measure the accuracy of the bit times.

For part 2:
For verification of reading UART data, I will set up a watch on the data variable inside the read function. Then I will set up a breakpoint on the statement at the end of the read function so I can check the value being returned. The rest is trivial, but I can also verify that the global character variable is being set correctly using the watch window. By sending data from the Tera Term. I can verify correct operation.

For part 3:
Similar to part 2, except I will set a breakpoint inside the ISR as well to make sure it is being called.

For part 4:
The verification for part 4 will be simple. Using the Tera Term, I will be able to type each character of the alphabet and make sure that I receive its uppercase equivalent. For non-alphabetic characters, the behavior of the program is undefined, as I just have it subtract 0x20 from the lowercase character for capitalization.

For part 5:
Similar to part 4, the Tera Term will be used to verify that I receive the same line that I send. I am not checking for overflow, but if I set the terminal to only 80 characters that won't be an issue. I am not adding a LF character to the string I send, so it will erase the line above each time, but if I want to change that it would be trivial. I will basically just compare input data to output data for verification of correct operation.