CPE301 – SPRING 2019

Design Assignment 3A

Student Name: Rocky Yasuaki Gonzalez
Student #: 5003229733
Student Email: gonzar14@unlv.nevada.edu
Primary Github address: https://github.com/rockyg1995/ihswppdar.git
Directory: C:\Users\rocky\Documents\CpE 301+L - Embedded Systems Design\CpE 301\Repository\DesignAssignments\DA3A

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

Atmega328PB Xplained Mini
Micro USB Cable (Power Supply)

Block Diagram N/A (Everything done through Atmega328PB Xplained Mini)

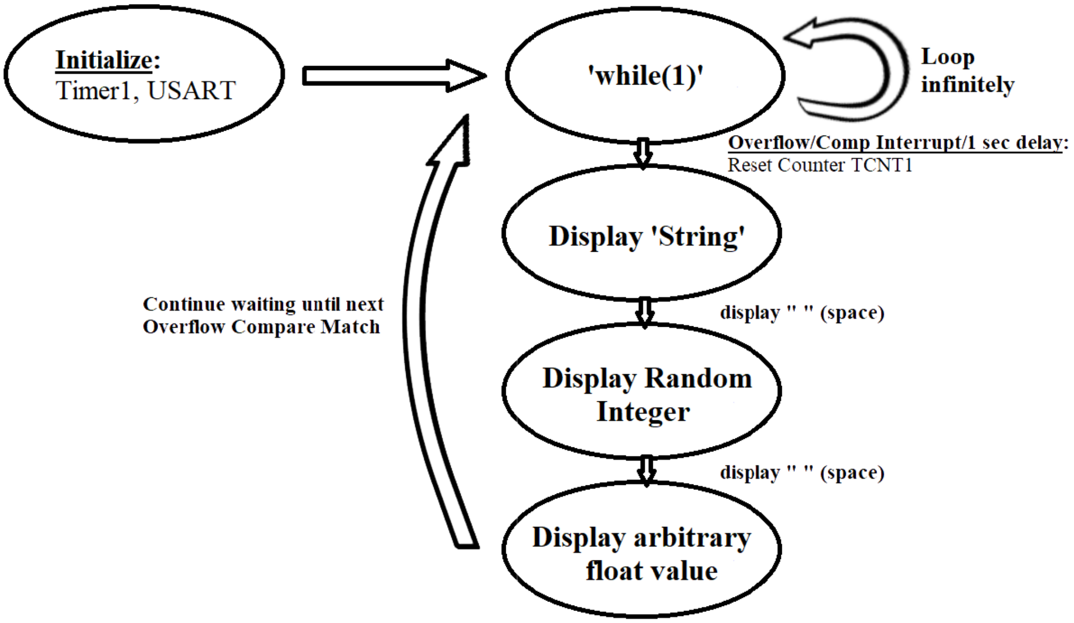


Figure 1 – Flow Chart for Coding Algorithm

2. INITIAL/DEVELOPED CODE OF TASK

```
/*
 * DA3A.c
 *
 * Created: 3/25/2019 12:05:50 AM
 * Author: rocky
 */

#define F_CPU 16000000UL    // Frequency of Xplained Mini (16MHz)
#include <avr/io.h>         // Standard AVR Library
#include <stdio.h>          // AVR library containing printf functions
#include <avr/interrupt.h>  // AVR library containing interrupt functions
#include <util/delay.h>     // AVR library containing _delay_ms() function

#define BAUDRATE 9600                           // Baudrate in Bits per second (bps)
#define BAUD_PRESCALLER ((F_CPU / (BAUDRATE * 16UL)) - 1)     // Baudrate Prescaler UBRR0
#define MAX_1s 15624                            // 1s delay: OCR1A = (16MHz*1000ms/prescaler) -1

//Declaration of our functions
void Timer1_init(void);                         // Function to initialize Timer1
void USART_init(void);                          // Function to initialize USART

unsigned char USART_receive(void);              // Function to receive Serial data from UDR0
void USART_send( unsigned char data);           // Function to send individual char into UDR0
```

```c
void USART_putstring(char* StringPtr);    // Function to break string into chars and send

char String[] = "DA3A";     // 'String[]' is an array treated as a string when between " "
char outs[20];              // 'outs[]' to store integer/float values into array of chars
char t[10];                 // 't[]' use
unsigned int rand_int;      // contains integer representing random value
float rand_float;           // contains float representing arbitrary random value

int main(void) {

       TIMSK1 |= (1 << OCIE1A);           // Set Interrupt on Compare Match
       sei();                             // Enable Global Interrupts
       Timer1_init();                     // Call the Timer1 initialization code
       USART_init();                      // Call the USART initialization code

       while (1) {                        // Infinite loop
              if (TCNT1 == OCR1A) {       // Display a String when Timer1 Matches OCR1A
                     rand_int = rand();   // Randomize Integer Value
                     rand_float = 4.716;  // Arbitrary Float Value chosen
                     USART_putstring(String);    // Pass 'String' to send serial of chars
                     USART_send(' ');            // Puts blank space into terminal monitor
                     snprintf(outs,sizeof(outs),"%3d", rand_int);     // Stores integer
                                                 // 'rand_int' into the string 'outs'
                     USART_putstring(outs);      // Pass 'outs' to to send serial of chars
                     USART_send(' ');            // Puts blank space into terminal monitor
                     dtostrf(rand_float, 3, 5, t);      // Store float readable to
                                                        // 'snprintf'
                     snprintf(outs,sizeof(outs),"%s", t);             // Stores 't' (float
                                                 // 'rand_float') into the string 'outs'
                     USART_putstring(outs);      // Pass 'outs' to send serial of chars
                     USART_send('\n');           // Puts new line into terminal monitor
              }
       }
       return 0;
}
//------------------------------------------------------------------------
void Timer1_init(void) {           // Function to Initialize Timer1 properties
       OCR1A = MAX_1s;             // Let the OCR1A be the Max Value used for 1s delay
       TCCR1A = (0<<COM1A1)|(0<<COM1A0);  // Normal Operation, Disconnected OC1A COM1A1:0
       TCCR1A = (0<<COM1B1)|(0<<COM1B0);  // Normal Operation, Disconnected OC1B COM1B1:0
       TCCR1A = (0<<WGM11)|(0<<WGM10);           // Set Timer1 to CTC Mode 'WGM1:0'
       TCCR1B = (0<<WGM13)|(1<<WGM12);           // Set Timer1 to CTC Mode 'WGM3:2'
       TCCR1B = (1<<CS12)|(0<<CS11)|(1<<CS10);   // Set Clock Select for Prescaler '1024'
}

ISR(TIMER1_COMPA_vect) {    // Function to Truncate Timer1 to delay properly for 1 second
       TCNT1 = 0;           // Restart Timer1 from the Beginning
       return;              // Resume code from where interrupt left off
}
//------------------------------------------------------------------------
void USART_init(void) {                   // Function to Initialize USART properties
       UBRR0H = (uint8_t)(BAUD_PRESCALLER >> 8);// Store Upper Baudrate into UBRR0H
       UBRR0L = (uint8_t)(BAUD_PRESCALLER);      // Store Lower Baudrate into UBRR0L
       UCSR0B = (1 << RXEN0) | (1 << TXEN0);     // Enable Receiver and Enable Transmitter
       UCSR0C = (3 << UCSZ00);                   // Set UCSZ02:1 as 8-bit character data
}

unsigned char USART_receive(void) {       // Function to receive ASCII value from UDR0
```

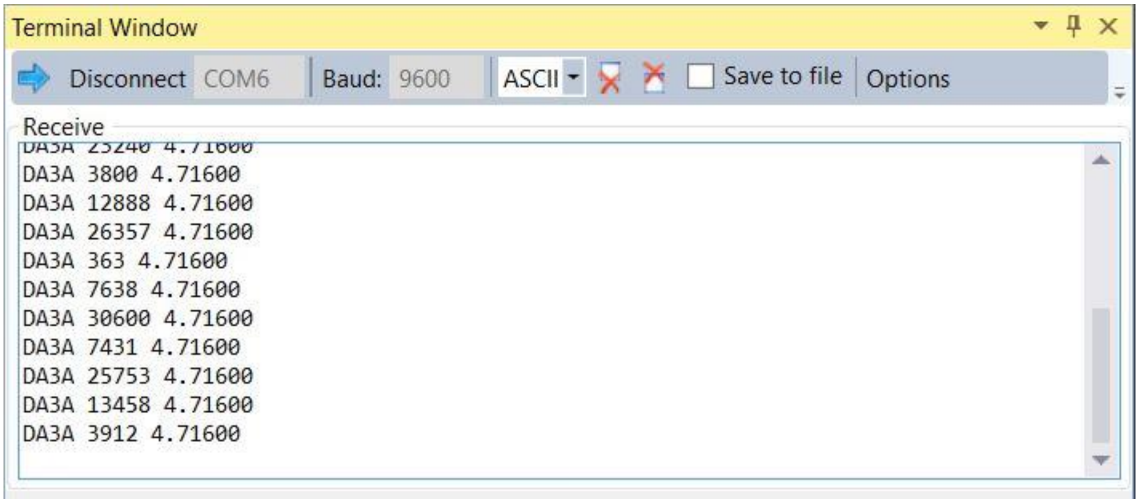
```
        while (!(UCSR0A & (1 << RXC0)));   // Keep Checking until RXC0 is 'High' to break
        return UDR0;                       // Return received serial into unsigned char
}

void USART_send(unsigned char data) {     // Function to transmit ASCII value into UDR0
        while (!(UCSR0A & (1 << UDRE0))); // Check until UDRE0 data register 'High'
        UDR0 = data;                       // Store unsigned char serial data into UDR0
}

void USART_putstring(char* StringPtr) {   // Function to break string into chars, and send
        while (*StringPtr != 0x00) {       // Keep Looping until String Completed (null)
                USART_send(*StringPtr);    // Send char value pointed by the string pointer
                StringPtr++;               // Increment pointer to next char array location
        }
}
//------------------------------------------------------------------------
```

3. SCHEMATICS

N/A (Everything done through Atmega328PB Xplained Mini)

4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

Since the Atmega328P/PB Xplained Mini have a built in AVR RX/TX module, we can send our serial data into a terminal which is able to display the string of chars, a random integer, and arbitrarily chosen float as shown in *Figure 3*:



COM6 Baud: 9600

```
DA3A 23240 4.71600
DA3A 3800 4.71600
DA3A 12888 4.71600
DA3A 26357 4.71600
DA3A 363 4.71600
DA3A 7638 4.71600
DA3A 30600 4.71600
DA3A 7431 4.71600
DA3A 25753 4.71600
DA3A 13458 4.71600
DA3A 3912 4.71600
```

Figure 2 – Output Terminal of Serial Data

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

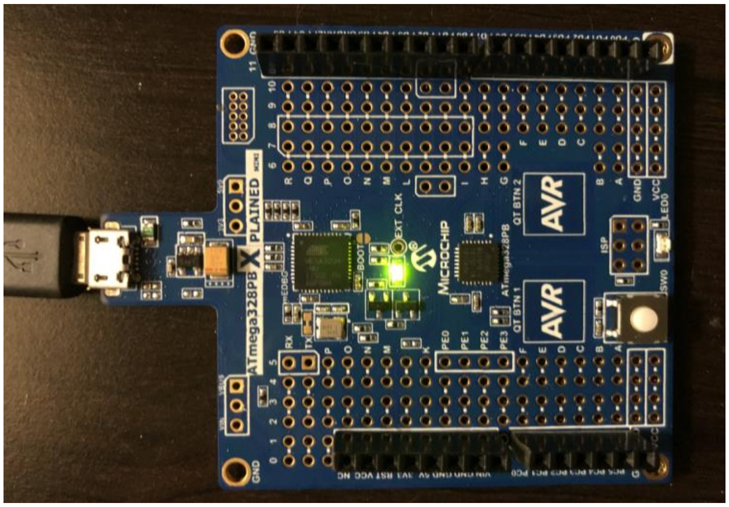


Figure 3 – Xplained Mini only for Serial Communication

6. VIDEO LINKS OF EACH DEMO

N/A (Not required for assignment)

7. GITHUB LINK OF THIS DA

https://github.com/rockyg1995/ihswppdar/tree/master/DesignAssignments/DA3A

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"This assignment submission is my own, original work".
Rocky Gonzalez