CPE301 – SPRING 2024

Design Assignment 4

Student Name: Abraham Garcia
Student #: 5005262049
Student Email: garci11@unlv.nevada.edu
Primary Github address: https://github.com/SON-Abe/submission_da.git
Directory: submission_da/Design_Assignments/DA4
Video Playlist: DA4

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**

- Microchip Studio Debugger
- Microchip Studio Terminal Window
- Microchip Studio Simulator
- ATmega328PB Microcontroller

2. **INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A**

```
/*
 * ADC_example.c
 *
 * Created: 10/10/2019 10:14:02 PM
 * Author : VenkatesanMuthukumar
 */

#define F_CPU 16000000UL
#define BAUD_RATE 9600

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

void usart_init ();
void adc_init();
void timer_init();

void usart_send (unsigned char ch);

int main (void)
{
    timer_init ();
    usart_init ();
    adc_init ();

    while (1)
    {
        /*
        ADCSRA|=(1<<ADSC);  //start conversion
        while((ADCSRA&(1<<ADIF))==0);//wait for conversion to finish
        ADCSRA |= (1<<ADIF);

        int a = ADCL;
        a = a | (ADCH<<8);
        a = (a/1024.0) * 5000/10;
        usart_send((a/100)+'0');
        a = a % 100;
        usart_send((a/10)+'0');
        a = a % 10;
        //a = a * (9/5) + 32; // celsius to fahrenheit
        usart_send((a)+'0');
        usart_send('\r');

        _delay_ms(100);
        */
    }
    return 0;
}

ISR (TIMER1_OVF_vect)
{
    ADCSRA|=(1<<ADSC);  //start conversion
    while((ADCSRA&(1<<ADIF))==0);//wait for conversion to finish

    ADCSRA |= (1<<ADIF);

    int a = ADCL;
    a = a | (ADCH<<8);
    a = (a/1024.0) * 5000/10;
    usart_send((a/100)+'0');
    a = a % 100;
    usart_send((a/10)+'0');
    a = a % 10;
    //a = a * (9/5) + 32; // celsius to fahrenheit
    usart_send((a)+'0');
    usart_send('\r');
    _delay_ms(100);
    TCNT1 = 49911; // Reset timer
}
```

```c
void usart_init (void)
{
    UCSR0B = (1<<TXEN0);
    UCSR0C = (1<< UCSZ01)|(1<<UCSZ00);
    UBRR0L = F_CPU/16/BAUD_RATE-1;
}

void adc_init (void)
{
    /** Setup and enable ADC **/
    ADMUX = (0<<REFS1)|     // Reference Selection Bits
    (1<<REFS0)|     // AVcc - external cap at AREF
    (0<<ADLAR)|     // ADC Left Adjust Result
    (1<<MUX2)|      // Analog Channel Selection Bits
    (0<<MUX1)|      // ADC4 (PC4 PIN27)
    (1<<MUX0);
    ADCSRA = (1<<ADEN)|     // ADC ENable
    (0<<ADSC)|      // ADC Start Conversion
    (0<<ADATE)|     // ADC Auto Trigger Enable
    (0<<ADIF)|      // ADC Interrupt Flag
    (0<<ADIE)|      // ADC Interrupt Enable
    (1<<ADPS2)|     // ADC Prescaler Select Bits
    (0<<ADPS1)|
    (1<<ADPS0);
}

void timer_init (void)
{
    TCCR1B |= 5; //(1 << CS12) | (1 << CS10); // Sets prescaler to 1024
    TIMSK1 = (1 << TOIE1); // Enables overflow flag
    TCNT1 = 49911; // 1 second delay = (0xFFFF) - TCNT = 65535 - 15624 = 49911
    sei();
}

void usart_send (unsigned char ch)
{
    while (! (UCSR0A & (1<<UDRE0)));    //wait until UDR0 is empty
    UDR0 = ch;                          //transmit ch
}
```

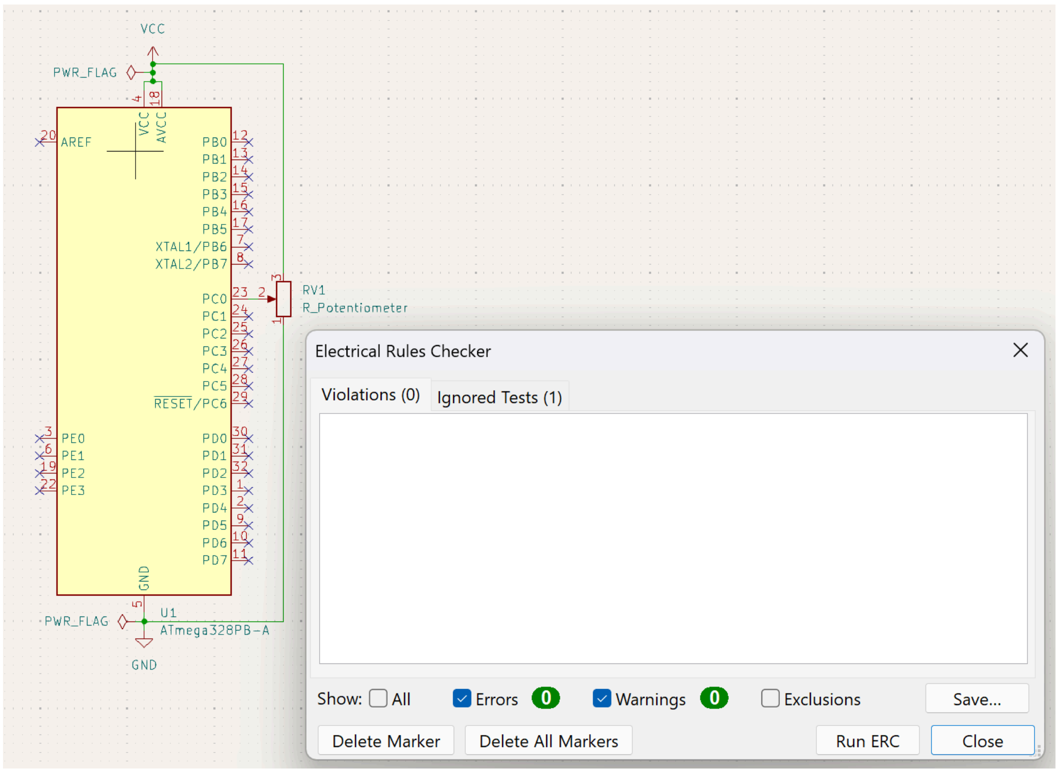
```
void usart_print(char* str)
{
    int i = 0;
    while(str[i] != 0)
    usart_send(str[i]);
}
```

3. **DEVELOPED/MODIFIED CODE OF TASK 3/A**

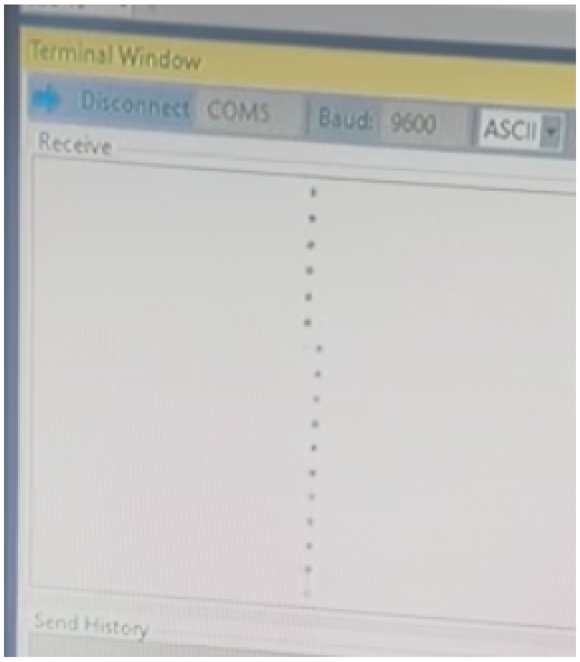
```
#define F_CPU 16000000UL                    //CPU FREQUENCY
#define BAUDRATE 9600                       //BAUDRATE
#define BAUD_PRESCALE (((F_CPU / (BAUDRATE * 16UL))) - 1)   //BAUD_PRESCALE
FORMULA

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

char str[51];                           //str char array of length 51

int adc_init(void)
{
    ADMUX |= (1 << REFS0);                  //Set ADC reference voltage to
AVCC
    ADCSRA = (1 << ADEN) | (1 << ADATE) | (1 << ADIE) | (1 << ADPS2) | (1
<< ADPS1) | (1 << ADPS0); //Enable ADC, Auto Trigger, Interrupt, Set ADC
prescale
    ADCSRB = (1 << ADTS2) | (1 << ADTS1);  //SET ADC AUTO TRIGGER TO
TIMER1 OVF
    while (ADCSRA & (1 << ADSC));         //WAIT TILL ADC CONVERSION FINISH
    return ADC;                             //RETURN ADC
}

void timer_init (void)
{
    TCCR1B |= (1 << CS11);                      //SET TIMER1 PRESCALE TO 8
    TIMSK1 = (1 << TOIE1);                      //TIMER1 OVERFLOW INTERRUPT
ENABLED
    TCNT1 = 1999;                               //SET TCNT1 TO 1999 FOR 10ms
```

```c
    sei();                                      //ENABLE GLOBAL VARIABLES
}

void usart_init(void)
{
    UBRR0H = (uint8_t) (BAUD_PRESCALE >> 8);    //LOAD UBRR0 HIGH 8 BITS
    UBRR0L = (uint8_t) (BAUD_PRESCALE);         //LOAD UBBR0 LOW BITS
    UCSR0B = (1 << RXEN0) | (1 << TXEN0);       //USART TRANSMITTER AND
RECEIVER ENABLED
    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);     //8 DATA BITS, 1 STOP BIT,
NO PARITY
}

void usart_send (unsigned char ch)
{
    while (! (UCSR0A & (1<<UDRE0)));        //WAIT TILL UDR0 IS EMPTY
    UDR0 = ch;                              //TRANSMIT CHARACTER
}

void usart_print(char* ChArrPtr)
{
    while ((*ChArrPtr) != '\0')             //WHILE CHAR ARRAY ISNT EMPTY
    {
        while (! (UCSR0A & (1 << UDRE0)));  //WAIT TILL UDR0 IS EMPTY
        UDR0 = *ChArrPtr;                   //TRANSMIT CHAR
        ChArrPtr++;                         //MOVE TO NEXT CHAR IN ARRAY
    }
}

ISR (TIMER1_OVF_vect)
{
    ADCSRA |= (1 << ADSC);                  //START ADC CONVERSION
    while ((ADCSRA & (1 << ADIF)) == 0);    //WAIT TILL CONVERSION FINSHES
    ADCSRA |= (1 << ADIF);                  //CLEAR ADC INTERRUPT FLAG

    int ADC_VALUE = ADCL;                   //ADD ADCL TO ADC_VALUE LOW
    ADC_VALUE = ADC_VALUE | (ADCH << 8);    //ADD ADCH TO ADC_VALUE HIGH

    int8_t i = ADC / 20.48;                 //INDEX FORMULA
    str[i] = '*';                           //STORE * IN STR[I]
    usart_print(str);                       //TRANSMIT STRING TO USART
    usart_print("\n");                      //NEWLINE

    str[i] = ' ';                           //CLEAR STR[I]
    TCNT1 = 1999;                           //RESET TCNT1
}

int main(void)
{
    memset(str, ' ', 49);               //INITIALIZE OUTPUT WITH SPACES
    str[50] = '\0';                     //NULL TERMINATE OUTPUT

    adc_init();                             //INITIALIZE ADC
    timer_init();                           //INITIALIZE TIMER1
    usart_init();                           //INITIALIZE USART

    while(1);
    return 0;
}
```

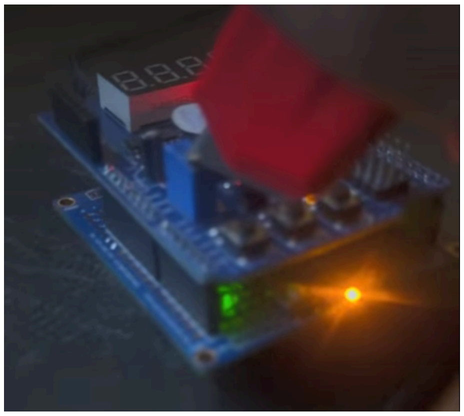
4. SCHEMATICS



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



6. **SCREENSHOT OF EACH DEMO (BOARD SETUP)**



7. **VIDEO LINKS OF EACH DEMO**

DA4

8. **GITHUB LINK OF THIS DA**

DA4

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

"This assignment submission is my own, original work".
Abraham Garcia