CPE301 – SPRING 2019

Design Assignment 2, Part A

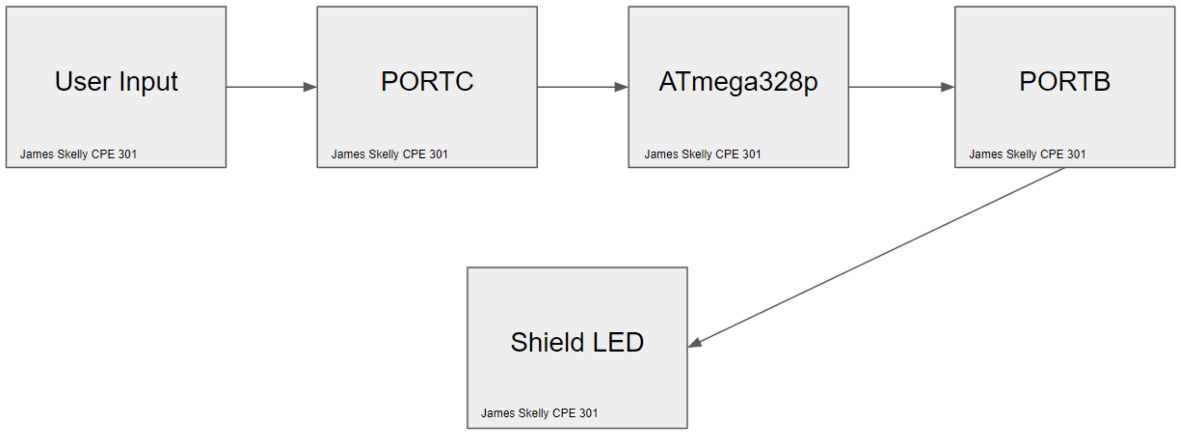
Student Name: James Skelly
Student #: 2000945485
Student Email: skellj1@unlv.nevada.edu
Primary Github address: https://github.com/skellj1/submission_da
Directory: skellj1/submission_da

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

- Components used for this assignment include Atmel Studio 7, MultiFunction Shield for microcontroller, X-plained Mini Microcontroller board with ATmega328p chip on board, and mEDBG programmer/debugger. Youtube was used to post video, my iPhone was used to record video, and the Tektronix TDS 2014 Oscilloscope was used to measure output waveforms with a 10:1 compensated oscilloscope probe.



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

Task 1 C Code

```c
/* DA2A_Task_1_C.c
 * Created: 3/1/2019 3:15:00 PM
 * Author : James Skelly */

// Design Assignment 2A:
// The goal of the assignment is use GPIO and delays:
// 1. Design a delay subroutine to generate a waveform on PORTB.2 with 60% DC and 0.725
// sec period.


#define F_CPU 16000000UL           // Set frequency of CPU to 16 MHz for delay function

#include <avr/io.h>                // Including AVR and Delay header libraries
#include <util/delay.h>

int main(void)
{
       DDRB = 0x04;                // Set data direction register bit 2 as an output

       while(1)

              PORTB = (1 << PORTB2);      // Shift a 1 into PORTB.2
              _delay_ms(435);             // Call for delay of 435 ms (0.725s * 0.6)
              PORTB = ~(1 << PORTB2);     // Shift a 0 into PORTB.2
              _delay_ms(290);             // Call for delay of 290 ms (0.725s * 0.4)

   return 0;
}
```

Task 1 Assembly Code

```
; DA2A_Task_1.asm
; Created: 3/1/2019 3:05:51 PM
; Author : James Skelly
; Replace with your application code

; Design Assignment 2A:
; The goal of the assignment is use GPIO and delays:

; 1. Design a delay subroutine to generate a waveform on PORTB.2 with 60% DC and 0.725
; sec period.

LDI R19, 0x04                   // Load immediate 0x04 into r19
LDI R20, 0x00                   // Load immediate 0x00 into R20
OUT DDRB, R19                   // Set PORTB.2 to output

main:
       OUT PORTB, R19                  // Set PORTB.2 high
       rcall delay_100ms        // Call for a delay of 435 ms
       rcall delay_100ms        //
       rcall delay_100ms        //
       rcall delay_100ms        //
       rcall delay_10ms         //
       rcall delay_10ms         //
       rcall delay_10ms         //
       rcall delay_5ms                 // End of 435 ms delay

       OUT PORTB, R20                  // Set PORTB.2 low
       rcall delay_100ms        // Call for a delay of 290 ms
       rcall delay_100ms        //
       rcall delay_10ms         //
       rcall delay_10ms         //
       rcall delay_10ms         //
       rcall delay_10ms         //
       rcall delay_10ms         //
       rcall delay_10ms         //
       rcall delay_10ms         //
       rcall delay_10ms         //
       rcall delay_10ms         // End of 290 ms delay

rjmp main                              // Jump back to the top of the loop
```

Developed Delay Subroutine for 1ms Delay

```
delay_1ms:
              push r16                    // push r16 to stack
              push r18                    // push r18 to stack
              ldi  r16, 255               // load 255 into r16
              ldi  r18, 6                 // load 6 into r18

delay_1b:                                 // label for outer loop
delay_1a:                                 // label for inner loop
              nop                         // no operation, takes 1 cycle
              nop                         //
              nop                         //
              nop                         //
              nop                         //
              nop                         //
              nop                         //
              dec r16                     // decrement the value in r16
              brne delay_1a               // if the value in r16 is not 0,
                                                 // stay in the loop

              dec r18                     // decrement the value in r18
              brne delay_1b               // if the value in r18 is not 0,
                                                 // stay in the loop

              pop r18                     // pop r18 back from the stack
              pop r16                     // pop r16 back from the stack
              ret                         // return to line below rcall
```

The code above was created to generate a subroutine that would last 1 ms. This subroutine was used as the building block for larger delays in assembly throughout this assignment. Calculating the delay generated by this subroutine was conducted by counting the number of instructions per loop and multiplying that number by the time taken to complete one cycle at 16 MHz, or 62.5 nanoseconds. Calculations led to a final generated delay of roughly 1.005ms. Adding more loops, more nop instructions, or subtracting loops and nop instructions only took the value further away from exactly 1ms. This value of 1.005 ms causes minor differences in the output waveforms between assembly and C throughout this design assignment.

Note that for task 1 code, logic was flipped in order to see the pulse high for 435 ms, and low for 290 ms. If the logic was kept reversed, the pulses would have looked similar to those in task two, where the pulse of the LED on actually makes the waveform go low.

3. DEVELOPED/MODIFIED CODE OF TASK 2/A from TASK 1/A

Task 2 C Code

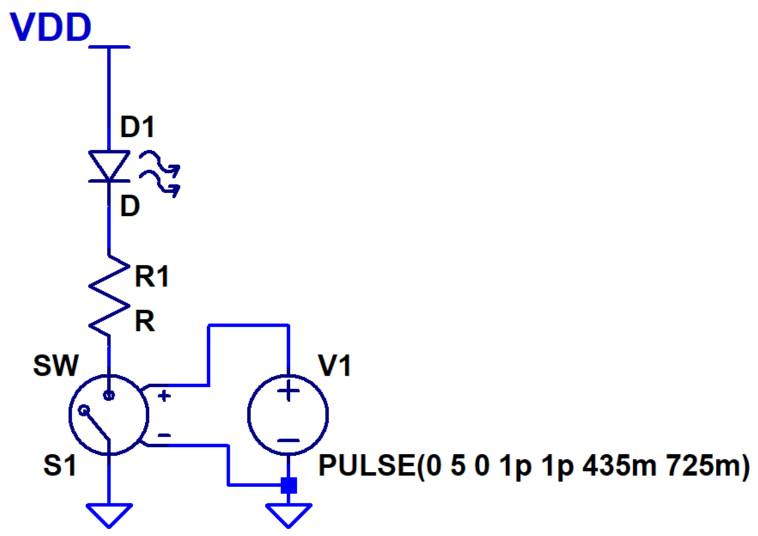
```c
/* DA2A_Task_2_C.c
 * Created: 3/1/2019 3:15:26 PM
 * Author : James Skelly */

 // 2. Connect a switch to PORTC.2 to poll for an event to turn on the LED
 // at PORTB.2 for 1.250 seconds after the event.


#define F_CPU 16000000UL    // Set frequency of CPU to 16 MHz for delay function
#include <avr/io.h>                // Including AVR and Delay header libraries
#include <util/delay.h>

int main (void)
       {

              DDRB |= (1<<2);    // Sets DDRB pin 2 to output mode by performing bitwise
                                   // OR with a 1 shifted two places to the left, and
                                   // and the value previously stored in DDRB. The
                                   // result will be stored in DDRB.

              PORTB |= (1<<2);   // Sets PORTB pin 2 to high by performing bitwise
                                   // OR with a 1 shifted two places to the left, and
                                   // and the value previously stored in PORTB. The
                                   // result will be stored in PORTB. This will initially
                                   // turn the LED off.

              DDRC &= (0 << 2);  // Sets DDRC pin 2 to input mode by performing bitwise A
                                   // AND with a 0 shifted two places to the left, and the
                                   // value previously stored in DDRC. The result will
                                   // be stored in DDRC.

              PORTC |= (1 << 2); // Sets PORTC pin 2 to high  by performing bitwise
                                   // OR with a 1 shifted two places to the left, and
                                   // and the value previously stored in PORTC. The
                                   // result will be stored in PORTC. This enables the
                                   // pull-up resistor.

              while (1)
              {
                     if (!(PINC & (1<<PINC2)))   // Check PINC1 for a value of 1,
                                                       // complemented to check for a
                                                       // zero instead.
                     {
                            PORTB &= ~(1<<2);    // Turn on LED by setting PORTB.2 to 0.
                            _delay_ms(1250);     // Call for a delay of 1.25 seconds
                     }
                     else
                     PORTB |= (1<<2);     // Turn off the LED by setting PORTB.2 to 1.
              }
              return 0;
       }
```

Task 2 Assembly Code

```
; DA2A_2_ASM.asm
; Created: 3/1/2019 3:14:30 PM
; Author : James Skelly

; 2. Connect a switch to PORTC.2 to poll for an event to turn on the LED
; at PORTB.2 for 1.250 seconds after the event.

.ORG 0

LDI R16,  0x04              ; loads 0x04 into register 16
LDI R17,  0x00              ; loads 0x04 into register 17

OUT DDRB, R16               ; sets data direction register PORTB.2 to output
OUT DDRC, R17               ; sets data direction register PORTC to all inputs

OUT PORTB,R16               ; sets PORTB.2 high
OUT PORTC, R16              ; sets port C, pin 2 high
NOP

MAIN:
       IN R18, PINC         ; reads input on pin C into register 20
       COM R18              ; invert the bits inputted to pin c
       ANDI R18, 0x04       ; bitwise AND the value in register 20 with 0x04

       CPI R18, 0x04        ; compare the value in r20 with 0x04
       BRNE MAIN

LIGHT_ON:
       LDI R21, 0xFB        ; 11111011 INTO R21
       OUT PORTB, R21       ; OUTPUT PORTB.2 LED 4 turns on
       rcall delay_1s       ; call for delay of 1.25 s
       rcall delay_100ms    ;
       rcall delay_100ms    ;
       rcall delay_50ms     ; end of 1.25 s delay
       OUT PORTB, R16       ; turn LED off
       RJMP MAIN                   ; jump back to top of code
```

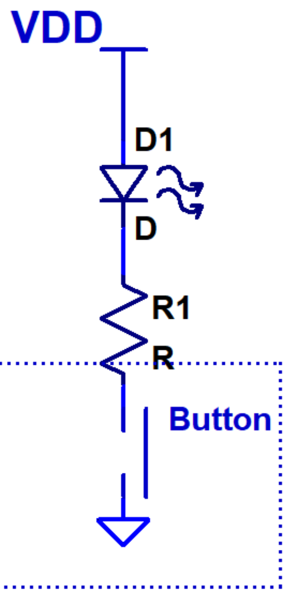
4. SCHEMATICS

Task 1 Schematic



The pulse source models the instructions given to the ATmega to output a 725ms period waveform with a 60% duty cycle.

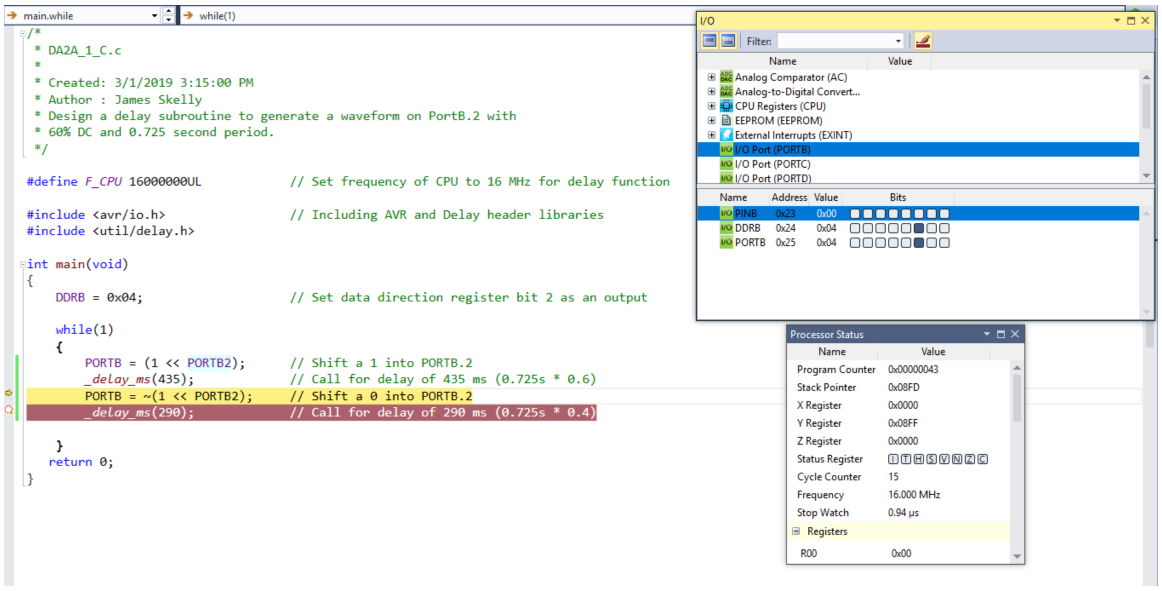
Task 2 Schematic



Button gets pressed, connects wire to ground. PB2 is connected to PC2 through the board, and the LED lights up for a time after the button is pressed.

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

Task 1 C

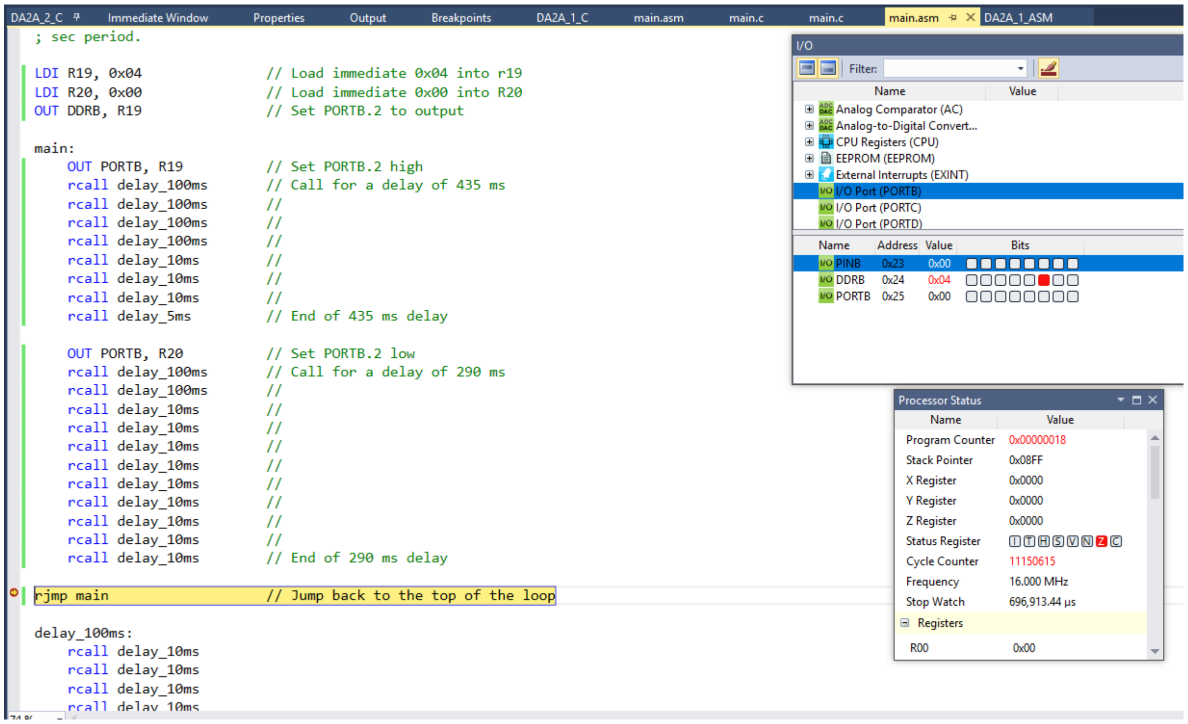


```
/*
 * DA2A_1_C.c
 *
 * Created: 3/1/2019 3:15:00 PM
 * Author : James Skelly
 * Design a delay subroutine to generate a waveform on PortB.2 with
 * 60% DC and 0.725 second period.
 */

#define F_CPU 16000000UL           // Set frequency of CPU to 16 MHz for delay function

#include <avr/io.h>                // Including AVR and Delay header libraries
#include <util/delay.h>

int main(void)
{
    DDRB = 0x04;                   // Set data direction register bit 2 as an output

    while(1)
    {
        PORTB = (1 << PORTB2);     // Shift a 1 into PORTB.2
        _delay_ms(435);            // Call for delay of 435 ms (0.725s * 0.6)
        PORTB = ~(1 << PORTB2);    // Shift a 0 into PORTB.2
        _delay_ms(290);            // Call for delay of 290 ms (0.725s * 0.4)

    }
   return 0;
}
```

```
Name     Address  Value  Bits
PINB     0x23     0x00   ☐☐☐☐☐☐☐☐
DDRB     0x24     0x04   ☐☐☐☐☐■☐☐
PORTB    0x25     0x04   ☐☐☐☐☐■☐☐
```

```
Processor Status
Name              Value
Program Counter   0x00000043
Stack Pointer     0x08FD
X Register        0x0000
Y Register        0x08FF
Z Register        0x0000
Status Register   ITHSVNZC
Cycle Counter     15
Frequency         16.000 MHz
Stop Watch        0.94 µs
Registers
R00               0x00
```

Task 1 Assembly



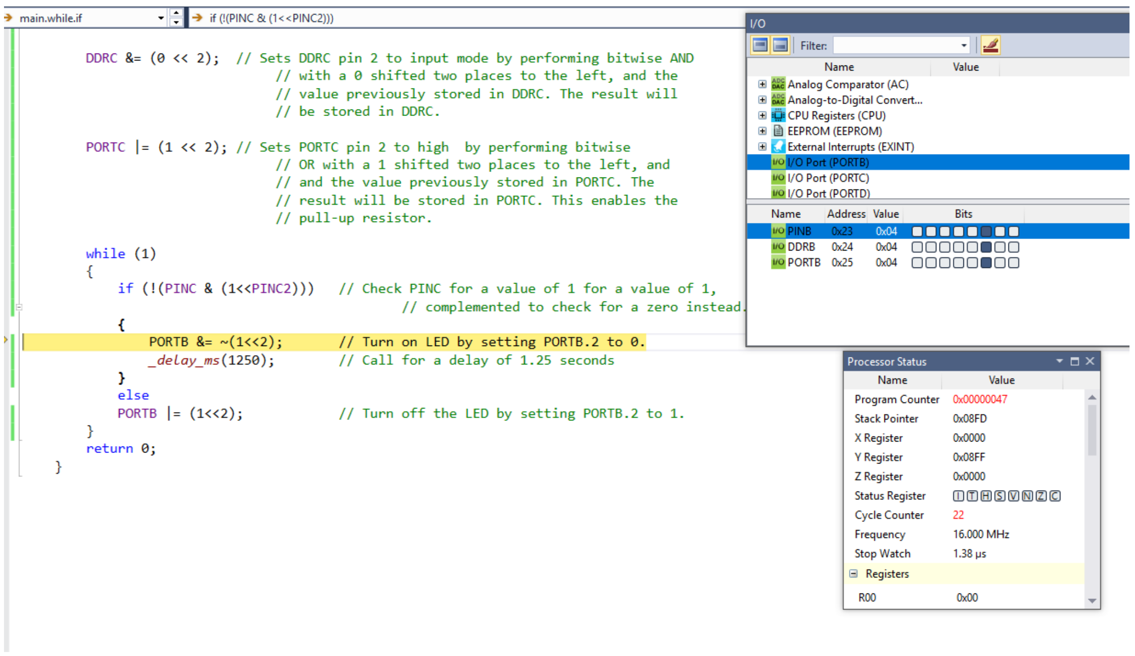
```
; sec period.

LDI R19, 0x04              // Load immediate 0x04 into r19
LDI R20, 0x00              // Load immediate 0x00 into R20
OUT DDRB, R19              // Set PORTB.2 to output

main:
    OUT PORTB, R19         // Set PORTB.2 high
    rcall delay_100ms      // Call for a delay of 435 ms
    rcall delay_100ms      //
    rcall delay_100ms      //
    rcall delay_100ms      //
    rcall delay_10ms       //
    rcall delay_10ms       //
    rcall delay_10ms       //
    rcall delay_5ms        // End of 435 ms delay

    OUT PORTB, R20         // Set PORTB.2 low
    rcall delay_100ms      // Call for a delay of 290 ms
    rcall delay_100ms      //
    rcall delay_10ms       //
    rcall delay_10ms       //
    rcall delay_10ms       //
    rcall delay_10ms       //
    rcall delay_10ms       //
    rcall delay_10ms       //
    rcall delay_10ms       //
    rcall delay_10ms       //
    rcall delay_10ms       // End of 290 ms delay

rjmp main                  // Jump back to the top of the loop

delay_100ms:
    rcall delay_10ms
    rcall delay_10ms
    rcall delay_10ms
    rcall delay_10ms
```

```
Name     Address  Value  Bits
PINB     0x23     0x00   ☐☐☐☐☐☐☐☐
DDRB     0x24     0x04   ☐☐☐☐☐■☐☐
PORTB    0x25     0x00   ☐☐☐☐☐☐☐☐
```

```
Processor Status
Name              Value
Program Counter   0x00000018
Stack Pointer     0x08FF
X Register        0x0000
Y Register        0x0000
Z Register        0x0000
Status Register   ITHSVNZC
Cycle Counter     11150615
Frequency         16.000 MHz
Stop Watch        696,913.44 µs
Registers
R00               0x00
```

Here we see the stopwatch measure 696 ms, which is exactly the period of the waveform measured by the oscilloscope.

Task 2 C



main.while.if | if (!(PINC & (1<<PINC2)))

```c
        DDRC &= (0 << 2);  // Sets DDRC pin 2 to input mode by performing bitwise AND
                           // with a 0 shifted two places to the left, and the
                           // value previously stored in DDRC. The result will
                           // be stored in DDRC.

        PORTC |= (1 << 2); // Sets PORTC pin 2 to high  by performing bitwise
                           // OR with a 1 shifted two places to the left, and
                           // and the value previously stored in PORTC. The
                           // result will be stored in PORTC. This enables the
                           // pull-up resistor.

        while (1)
        {
            if (!(PINC & (1<<PINC2)))    // Check PINC for a value of 1 for a value of 1,
                                              // complemented to check for a zero instead.
            {
                PORTB &= ~(1<<2);        // Turn on LED by setting PORTB.2 to 0.
                _delay_ms(1250);         // Call for a delay of 1.25 seconds
            }
            else
            PORTB |= (1<<2);             // Turn off the LED by setting PORTB.2 to 1.
        }
        return 0;
    }
```

I/O

Name	Value
Analog Comparator (AC)	
Analog-to-Digital Convert...	
CPU Registers (CPU)	
EEPROM (EEPROM)	
External Interrupts (EXINT)	
I/O Port (PORTB)	
I/O Port (PORTC)	
I/O Port (PORTD)	

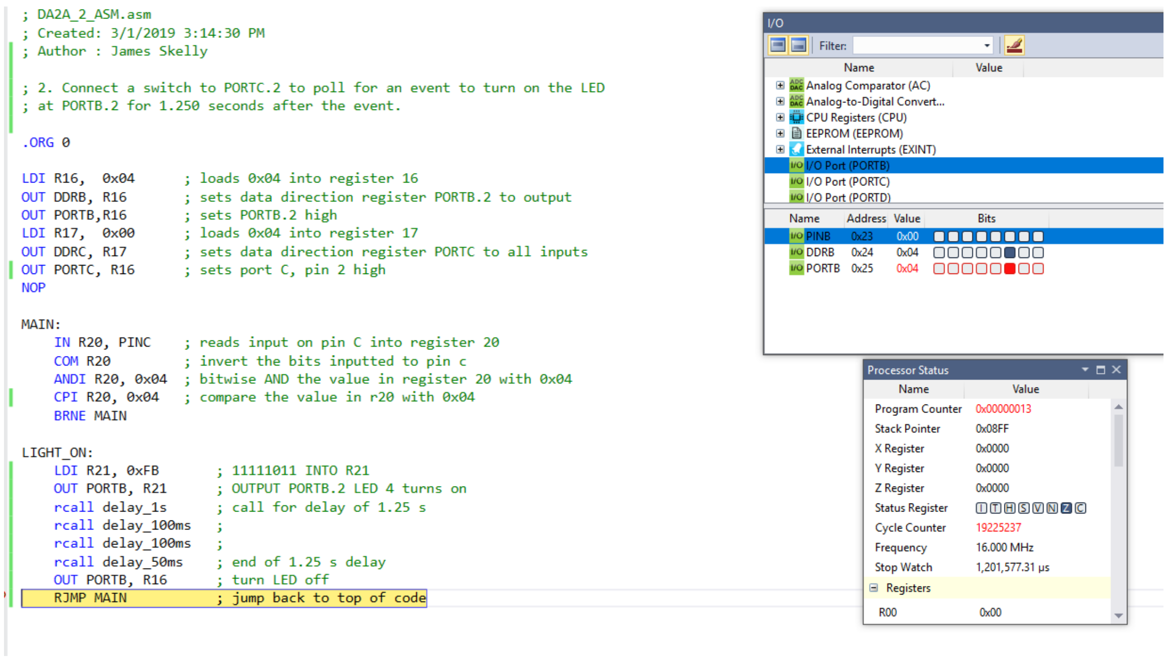
Name	Address	Value	Bits
PINB	0x23	0x04	☐☐☐☐☐■☐☐
DDRB	0x24	0x04	☐☐☐☐☐■☐☐
PORTB	0x25	0x04	☐☐☐☐☐■☐☐

Processor Status

Name	Value
Program Counter	0x00000047
Stack Pointer	0x08FD
X Register	0x0000
Y Register	0x08FF
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	22
Frequency	16.000 MHz
Stop Watch	1.38 µs
Registers	
R00	0x00

When programming in C, far less instructions are required to perform tasks, so our stop watch only records a total time of 1.38 microseconds.

Task 2 Assembly



```asm
; DA2A_2_ASM.asm
; Created: 3/1/2019 3:14:30 PM
; Author : James Skelly

; 2. Connect a switch to PORTC.2 to poll for an event to turn on the LED
; at PORTB.2 for 1.250 seconds after the event.

.ORG 0

LDI R16,  0x04        ; loads 0x04 into register 16
OUT DDRB, R16         ; sets data direction register PORTB.2 to output
OUT PORTB,R16         ; sets PORTB.2 high
LDI R17,  0x00        ; loads 0x04 into register 17
OUT DDRC, R17         ; sets data direction register PORTC to all inputs
OUT PORTC, R16        ; sets port C, pin 2 high
NOP

MAIN:
    IN R20, PINC      ; reads input on pin C into register 20
    COM R20           ; invert the bits inputted to pin c
    ANDI R20, 0x04    ; bitwise AND the value in register 20 with 0x04
    CPI R20, 0x04     ; compare the value in r20 with 0x04
    BRNE MAIN

LIGHT_ON:
    LDI R21, 0xFB           ; 11111011 INTO R21
    OUT PORTB, R21          ; OUTPUT PORTB.2 LED 4 turns on
    rcall delay_1s          ; call for delay of 1.25 s
    rcall delay_100ms       ;
    rcall delay_100ms       ;
    rcall delay_50ms        ; end of 1.25 s delay
    OUT PORTB, R16          ; turn LED off
    RJMP MAIN               ; jump back to top of code
```

I/O

Name	Value
Analog Comparator (AC)	
Analog-to-Digital Convert...	
CPU Registers (CPU)	
EEPROM (EEPROM)	
External Interrupts (EXINT)	
I/O Port (PORTB)	
I/O Port (PORTC)	
I/O Port (PORTD)	

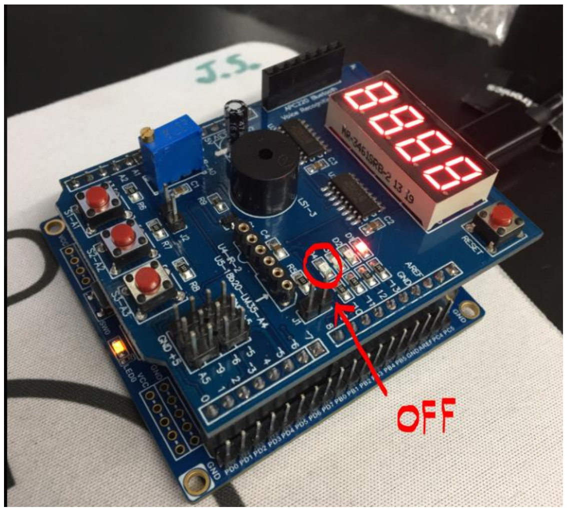
Name	Address	Value	Bits
PINB	0x23	0x00	☐☐☐☐☐☐☐☐
DDRB	0x24	0x04	☐☐☐☐☐■☐☐
PORTB	0x25	0x04	☐☐☐☐☐■☐☐

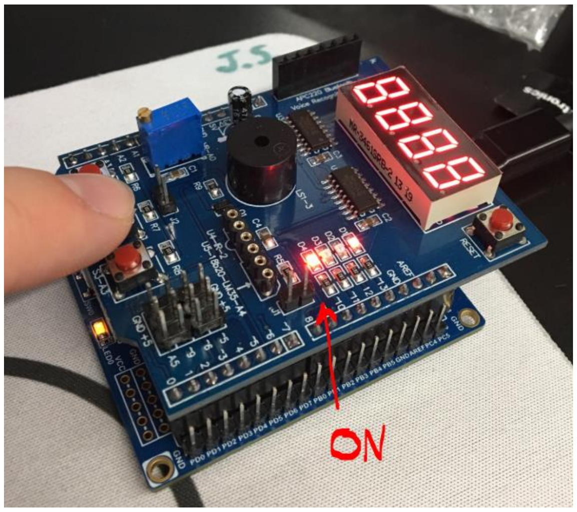
Processor Status

Name	Value
Program Counter	0x00000013
Stack Pointer	0x08FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	19225237
Frequency	16.000 MHz
Stop Watch	1,201,577.31 µs
Registers	
R00	0x00

We see here the stop watch measured 1.201 seconds in simulation. This is very similar to our experimental value.

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)



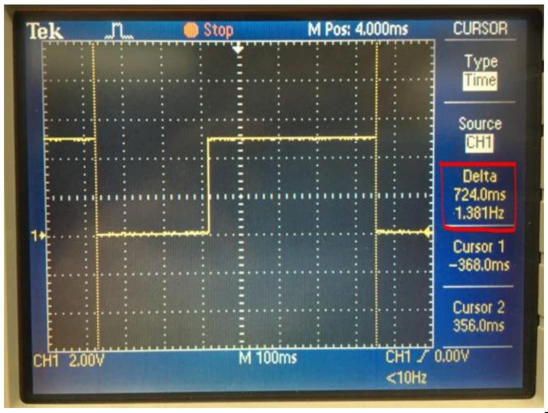


Task 1: Design a delay subroutine to generate a waveform on PORTB.2 with 60% DC and 0.725 second period.

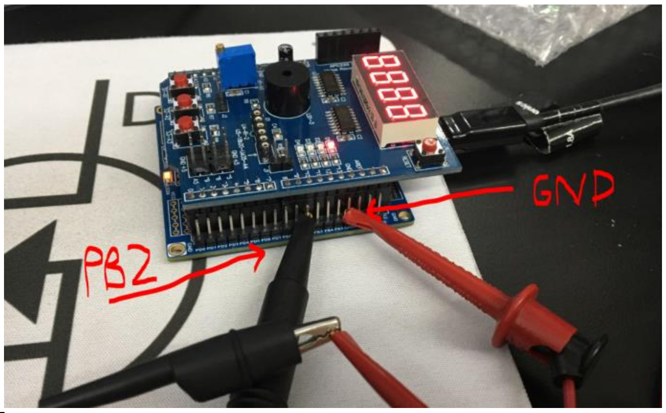
TASK 1 C Code SCOPE OUTPUT

- **Note that for parts 1 and 2, Assembly and C, the same board setup shown below was used. A 10:1 compensated scope probe was used to probe the signal on PORTB.2 and made visible on the oscilloscope for analysis using cursors.**

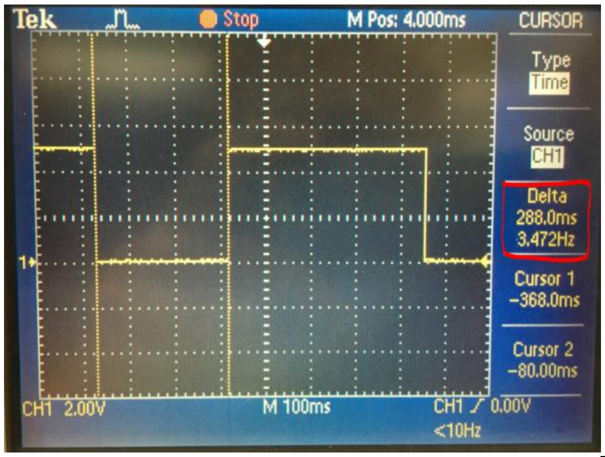
Period of Waveform is 0.724 seconds.

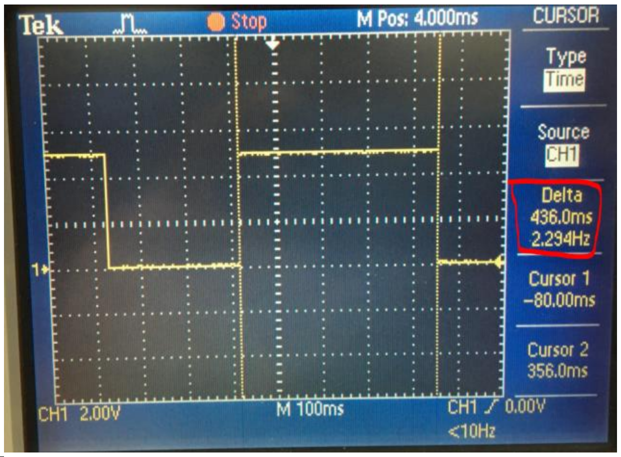


Board Setup with PB2 and GND Connected to Probe



Time low should be (0.725 s * 0.4 = 290 ms). **Time high should be (0.725s * 0.6 = 435 ms)**



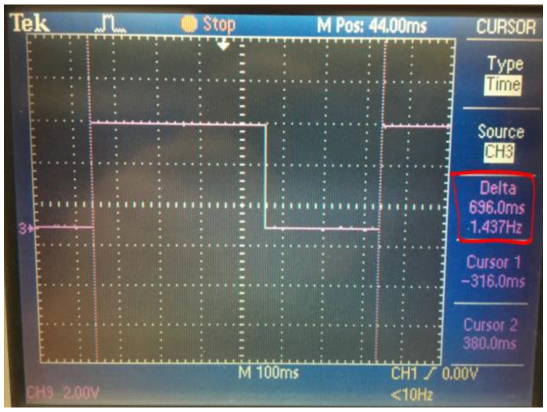


The C code for task 1 gave very accurate output waveforms, with the time high and the period varying from the expected values by only 1 millisecond, and the time low varying by 2 milliseconds.

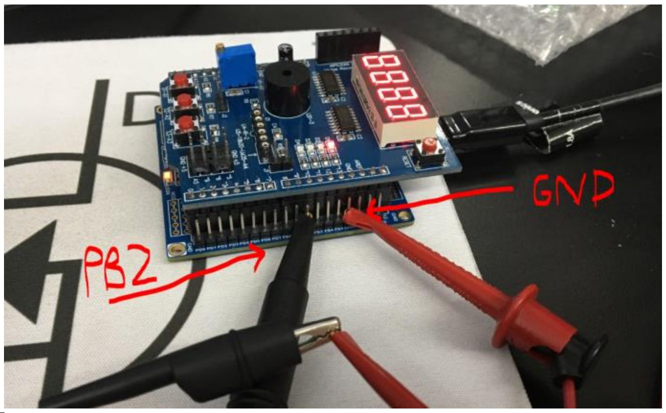
<u>TASK 1 Assembly Code SCOPE OUTPUT</u>

- **In the waveforms below, we will see that there are small differences between our calculated or expected values and the actual time high, time low, and period of the signal. That is because in Assembly, without using timers, generating a delay for an exact amount of time is not easy. Had I used the internal timers of the ATmega328p, I could have generated a more accurate signal.**

Period of Waveform is 0.696 seconds.

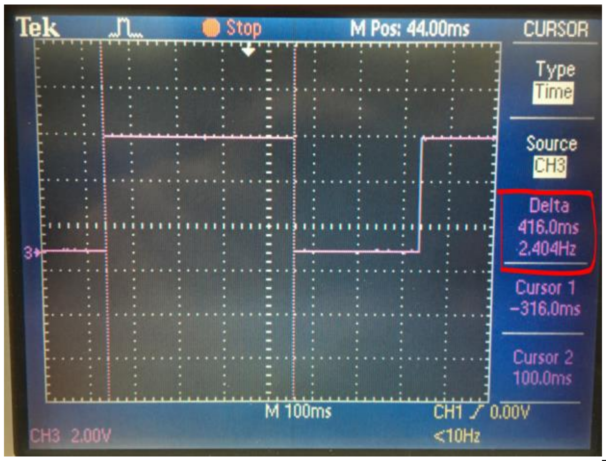


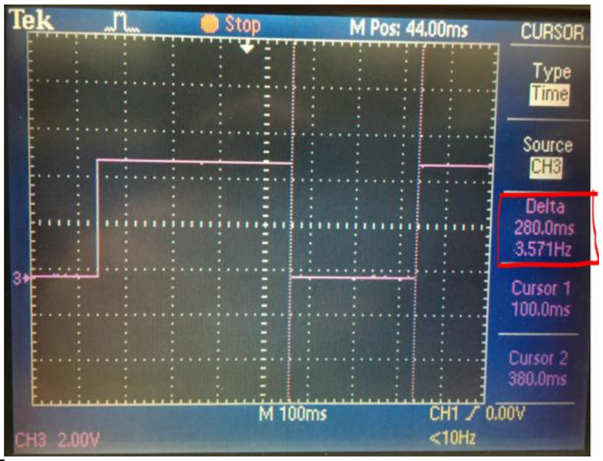
Board Setup with PB2 and GND Connected to Probe



Time low SHOULD BE (0.725 s * 0.4 = 290 ms).

Time high SHOULD BE (0.725s * 0.6 = 435 ms)





Calculating an estimate for the time taken to complete my delay_1ms subroutine using a 62.5 ns time per cycle at 16 MHz yielded that my delay_1ms subroutine was taking about 1.005 ms to execute. With millions of cycles running to generate larger delays, this 5 microsecond overshoot is what is causing the variations between my waveforms and my calculations for expected time high, time low, and period.

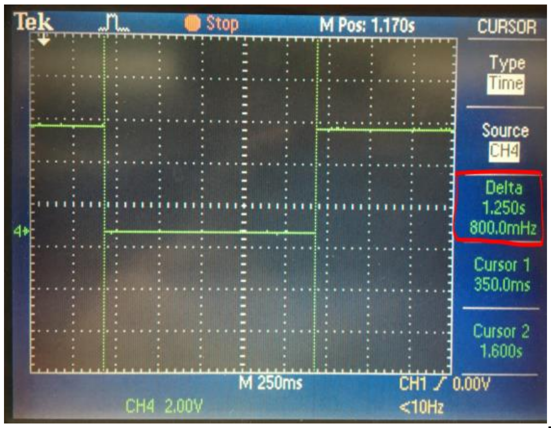
Task 2: Connect a switch to PORTC.2 to poll for an event to turn on the LED at PORTB.2 for 1.250 seconds after the event.

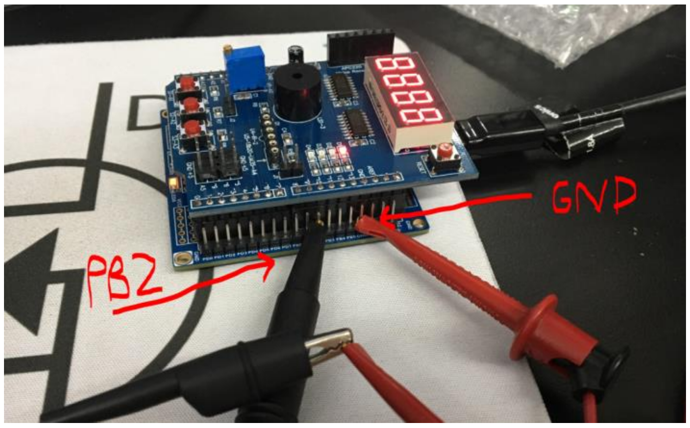
TASK 2 C Code SCOPE OUTPUT

- **The C code for part 2 worked perfectly, outputting a pulse low (reverse logic) to turn on the LED for exactly 1.250 seconds. The scope was adjusted to have a high seconds per division measurement so that the pulse could be captured and measured.**

Width of Pulse is 1.250 seconds.

Board Setup with PB2 and GND Connected to Probe

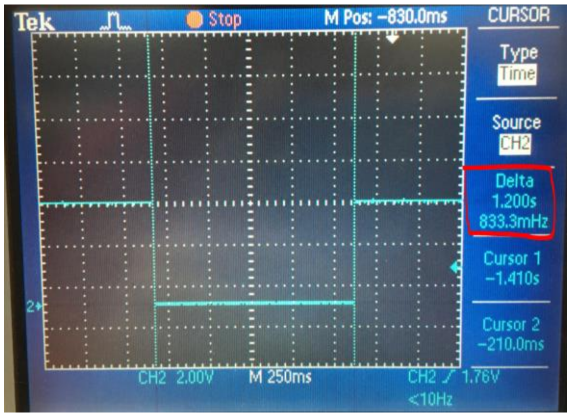




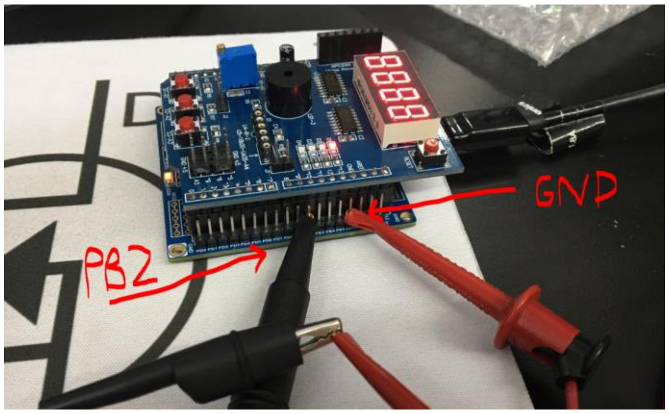
__TASK 2 Assembly SCOPE OUTPUT__

- **The Assembly code for part 2 yielded a small error of 50 ms. The oscilloscope shows 1.200 seconds between cursors, while our expected value was 1.250s. As was mentioned previously, this error is caused by the small overshoot for the time taken to complete the delay_1ms subroutine.**

Width of Pulse is 1.200 seconds.



Board Setup with PB2 and GND Connected to Probe



7. VIDEO LINKS OF EACH DEMO

Link to my youtube video library:
https://www.youtube.com/channel/UC4DhXgCIBkllvnwG6NH4sDA?view_as=subscriber

TASK 1 C Code: https://www.youtube.com/watch?v=yClkI5S3_iQ&t=18s
TASK 2 C Code: https://www.youtube.com/watch?v=EpZwVWvA-EU

TASK 1 Assembly: https://www.youtube.com/watch?v=-tadpfDRS0s&t=3s
TASK 2 Assembly: https://www.youtube.com/watch?v=ORVSoxsDOTU

8. GITHUB LINK OF THIS DA

https://github.com/skellj1/submission_da

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

"*This assignment submission is my own, original work*".
James W. Skelly