CPE301 – SPRING 2019

Design Assignment 2, Part C

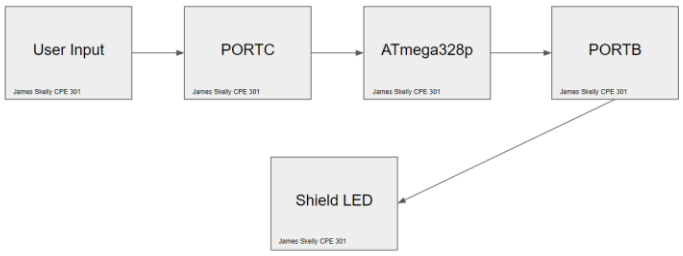
Student Name: James Skelly
Student #: 2000945485
Student Email: skellj1@unlv.nevada.edu
Primary Github address: https://github.com/skellj1/submission_da
Directory: skellj1/submission_da

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, 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. The block diagram below is specific to part B. This is because no user input is required for part A, and the block diagram would not make much sense.



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

PART 1A: Creating a waveform with 60% DC, 0.725 second period using timer0, normal mode.

```
/*
 * 2C_1A.c
 *
 * Created: 3/21/2019 11:37:08 PM
 * Author : James Skelly
 */

#define F_CPU 16000000UL
#include <avr/io.h>

int main(void)
{
	uint8_t Overflow = 0;	// declare overflow variable and initialize to 0
	DDRB = 0x04;  // set data direction reg. B to output
	TCCR0A = 0;		// normal mode operation
	TCCR0B = 0x05;	// prescaler of 1024
    TCNT0 = 0x0;	// start the timer

    while (1)
    {
		// wait for the overflow event
		while ((TIFR0 & 0x01) == 0);

		TCNT0 = 0x00; // reset timer0 to 0
		TIFR0 = 0x01; // reset overflow flag
		Overflow++;		// increment overflow variable

		if (Overflow == 18)		// if statement for 18th overflow of timer0
		{							// generate delay of 290 ms
			PORTB = 0x00;		// sets PORTB.2 high (reverse logic, LED on)
		}

		if (Overflow == 44)		// if statement for 44th overflow of timer0
		{							// generate delay of 435 ms
			PORTB = 0x04;		// sets PORTB.2 low (LED off)
			Overflow = 0;		// resets overflow counter to 0
		}
    }
}
```

PART 1B: Turn on LED for 1.25 seconds after pressing button using timer0, normal mode.

```
/*
 * 2C_1B.c
 *
 * Created: 3/21/2019 11:37:37 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)
{
	uint8_t Overflow = 0; // Declare overflow variable and initialize to 0
	DDRB = 0x04;          // Sets DDRB pin 2 to output mode
	PORTB = 0x04;         // Sets PORTB pin 2 to high
	DDRC = 0x00;          // Sets DDRC pin 2 to input mode
	PORTC =0x04;          // Sets PORTC pin 2 to high

	TCCR0A = 0x00;               // Set timer0 to operate in normal mode
	TCCR0B = 0x05;               // Set prescaler to 1024

	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.
		{
			Overflow = 0;         // reset overflow to 0 when button pressed
			TCNT0 = 0x00;         // reset timer to 0 when button pressed
			PORTB = 0x00;         // Turn on LED by setting PORTB.2 to 0.
		}



		while ((TIFR0 & 0x01) == 0); // check for overflow in overflow flag bit

			TIFR0 = 0x01;                // reset overflow flag
			Overflow++;                         // increment overflow variable

			if (Overflow == 76)          // check for 76th overflow to achieve
			                                    // desired delay of 1.250 seconds
				{
					PORTB = 0x04;         // turn the LED off after 1.250s
				}

			TCNT0 = 0;                                  // reset the timer
	}

	return 0;
}
```

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

PART 2A: Implementing Part 1A using TIMER0_OVF_vect interrupt mechanism, normal mode.

```
/*
 * 2C_2A.c
 *
 * Created: 3/21/2019 11:38:47 PM
 * Author : James Skelly
 */

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>

int main(void)
{
	DDRB = 0x04;  // set PORTB.2 to output
	PORTB = 0x04; // initially turn LED at portb.2 off
	TIMSK0 = 0x01;		// enable timer0 overflow interrupt
	TCNT0 = 0x00; // initialize timer0 to 0
	sei();				// enable interrupts

	TCCR0A = 0x00;		// set timer0 to normal mode operation
	TCCR0B = 0x05;		// set prescaler of 1024

    while (1)
    {
		// main loop
    }
}

uint8_t Overflow = 0;		// initialize overflow variable

ISR (TIMER0_OVF_vect) // interrupt subroutine highlighted to show that this is the part
                      // of the code that was modified.
{
	TCNT0 = 0;				// reset timer0 to 0 when interrupt occurs
	Overflow++;				// increment overflow variable

	if (Overflow == 18)  // if statement to check for 18th overflow of timer0
	{
		PORTB = 0x00; // turns LED at PORTB.2 on
	}

	if (Overflow == 44)  // if statement to check for 44th overflow of timer0
	{
		PORTB = 0x04; // turns LED off
		Overflow = 0; // resets overflow variable to 0
	}

}
```

PART 2B: Implementing Part 1B using TIMER0_OVF_vect interrupt mechanism, normal mode.

```
/*
 * 2C_2B.c
 * Created: 3/21/2019 11:39:09 PM
 * Author : James Skelly
 */

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>

int button = 0;  // initialize global variable for button

int main(void)
{
        DDRB = 0x04;            // Sets DDRB pin 2 to output mode
        PORTB = 0x04;           // Sets PORTB pin 2 to high
        DDRC = 0x00;            // Sets DDRC pin 2 to input mode
        PORTC = 0x04;           // Sets PORTC pin 2 to high
        TIMSK0 = 0x01;                  // enable timer0 overflow interrupt
        sei();                                  // enable interrupts
        TCCR0A = 0x00;                  // Set timer0 to operate in normal mode
        TCCR0B = 0x05;                  // Set prescaler to 1024

    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.
                {
                        button = 1;             // activates if statement inside interrupt subroutine
                }

                else
                {
                        PORTB = 0x04;           // keeps LED off until button is pressed
                        button = 0;             // does not enter subroutine if statement
                }
        }
        return 0;
}

ISR (TIMER0_OVF_vect)    // highlighted to show that this is the part of the code that was modified
{
        int overflow = 0;                       // initialize overflow variable
        int overflow_amt = 76;                  // set variable for amount required to achieve
                                                // desired delay of 1.25 seconds.
        int overflow_check = 0;                 // initialize overflow check variable

        if (button == 1)
        {
                PORTB = 0x00;                   // turn on LED if button is pressed
                TCNT0 = 0;                      // start timer0 at 0

        while(overflow < overflow_amt)          // keep looping until desired delay is reached
        {
                overflow_check = TIFR0 & 0x01;  // check for overflow
                if (overflow_check == 1)        // if overflow,
                {
                        overflow++;             // increment overflow variable
                        TIFR0 = 0x01;           // reset overflow flag
                }
        }
        overflow = 0;                                   // reset overflow variable to 0
        PORTB = 0x04;                                   // turn LED off after 1.250 s delay
        button = 0;                                     // reset button variable to 0
        }
}
```

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

PART 3A: Implementing Part 1A using TIMER0_COMPA_vect interrupt mechanism, CTC mode.

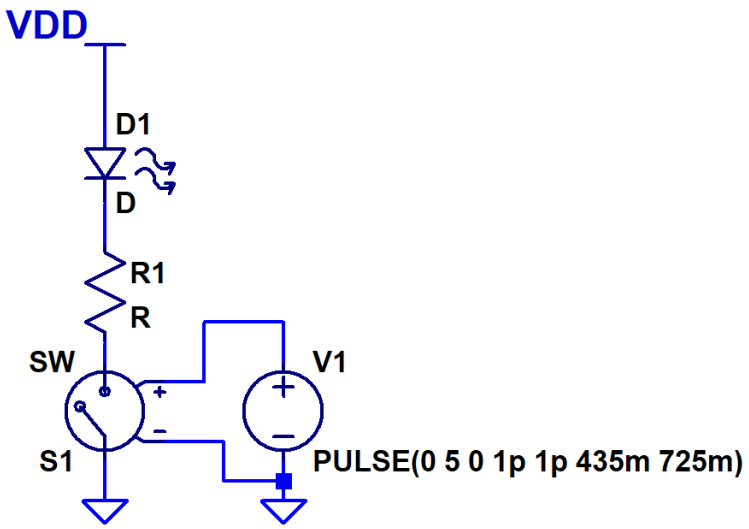
```
/*
 * 2C_3A.c
 *
 * Created: 3/21/2019 11:39:35 PM
 * Author : James Skelly
 */

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>

int main(void)
{
	DDRB = 0x04;		// set PORTB.2 to output
	PORTB = 0x04;		// turn LED initially off
	TCCR0A = 0x02;		// timer0 to operate in CTC mode
	TCCR0B = 0x05;		// set prescaler to 1024
	OCR0A = 0xFF;		// load compare register value
	TIMSK0 = 0x02;		// interrupt enabled on compare match A
	sei();			// enable interrupts

    while (1)
    {
    }
}

uint8_t Overflow = 0;			// initialize overflow variable

ISR (TIMER0_COMPA_vect)
{
	TCNT0 = 0;			// reset timer0 to 0 when interrupt occurs
	Overflow++;			// increment overflow variable

	if (Overflow == 18)		// if statement to check for 18th overflow of timer0
	{
		PORTB = 0x00;		// turns LED at PORTB.2 on
	}

	if (Overflow == 44)		// if statement to check for 44th overflow of timer0
	{
		PORTB = 0x04;		// turns LED off
		Overflow = 0;		// resets overflow variable to 0
	}
}
```

PART 3B: Implementing Part 1B using TIMER0_COMPA_vect interrupt mechanism, CTC mode.

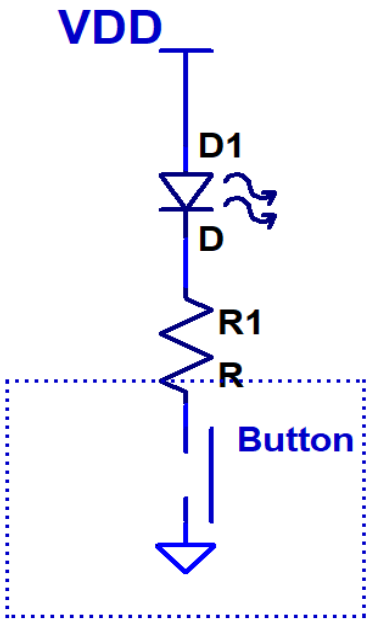
```
/*
 * 2C_3B.c
 *
 * Created: 3/21/2019 11:44:36 PM
 * Author : James Skelly
 */

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>

int button = 0;

int main(void)
{
    DDRB = 0x04;         // set PORTB.2 to output
    PORTB = 0x04;        // turn LED initially off
    DDRC = 0x00;         // set PORTC.2 to input
    PORTC = 0x04;        // set PORTC.2 initially high
    TCCR0A = 0x02;       // timer0 to operate in CTC mode
    TCCR0B = 0x05;       // set prescaler to 1024
    OCR0A = 0xFF;        // load compare register value
    TIMSK0 = 0x02;       // interrupt enabled on compare match A
    sei();               // enable interrupts

    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.
                {
                        button = 1;     // activates if statement inside interrupt subroutine
                }
                else
                {
                        PORTB = 0x04;   // keeps LED off until button is pressed
                        button = 0;       // does not enter subroutine if statement
                }
    }
        return 0;
}

ISR (TIMER0_COMPA_vect)
{
        int overflow = 0;                       // initialize overflow variable
        int overflow_amt = 76;                  // set variable for amount required to achieve
                                                // desired delay of 1.25 seconds.
        int overflow_check = 0;                 // initialize overflow check variable

        if (button == 1)
        {
                PORTB = 0x00;                           // turn on LED if button is pressed
                TCNT0 = 0;                              // start timer0 at 0

                while(overflow < overflow_amt)          // keep looping until desired delay is reached
                {
                        overflow_check = TIFR0 & 0x01;  // check for overflow
                        if (overflow_check == 1)        // if overflow,
                        {
                                overflow++;                     // increment overflow variable
                                TIFR0 = 0x01;                   // reset overflow flag
                        }
                }
                overflow = 0;                                   // reset overflow variable to 0
                PORTB = 0x04;                                   // turn LED off after 1.250 s delay
                button = 0;                                     // reset button variable to 0
        }
}
```

4. SCHEMATICS

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

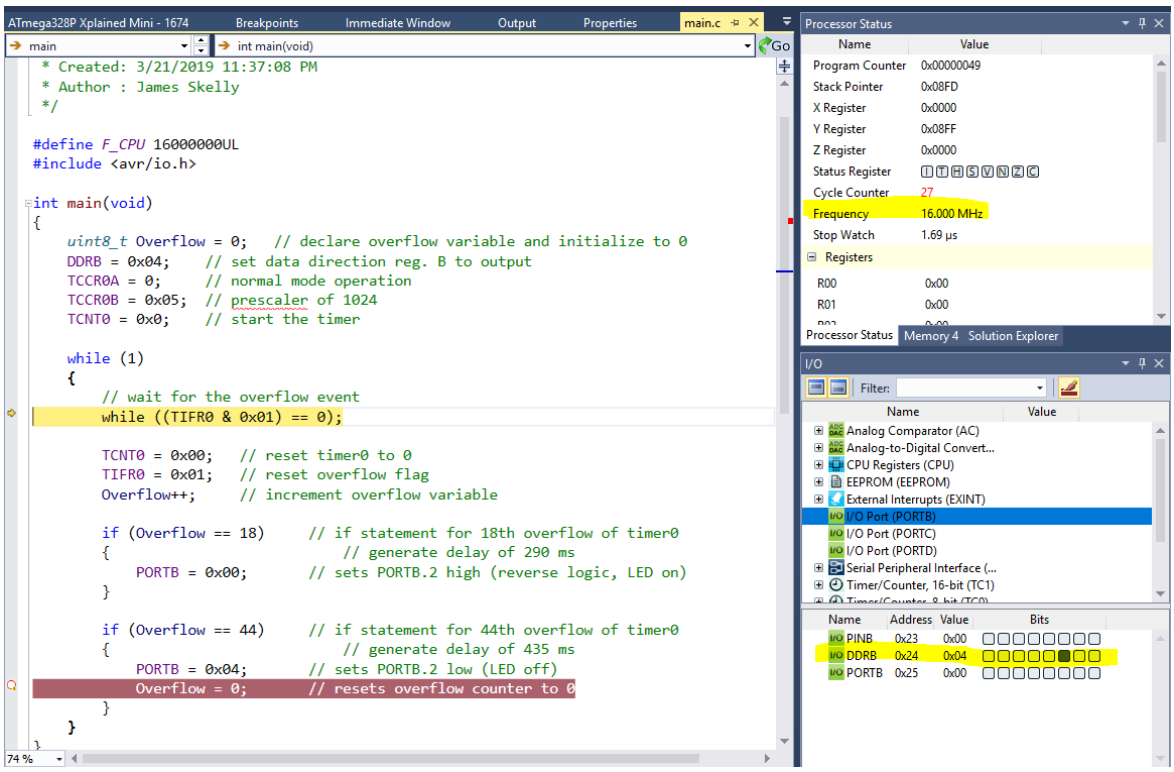


Button gets pressed, connects wire to ground. PB2 is connected to PC2 through the board, and the LED lights up for a time after the button is pressed.

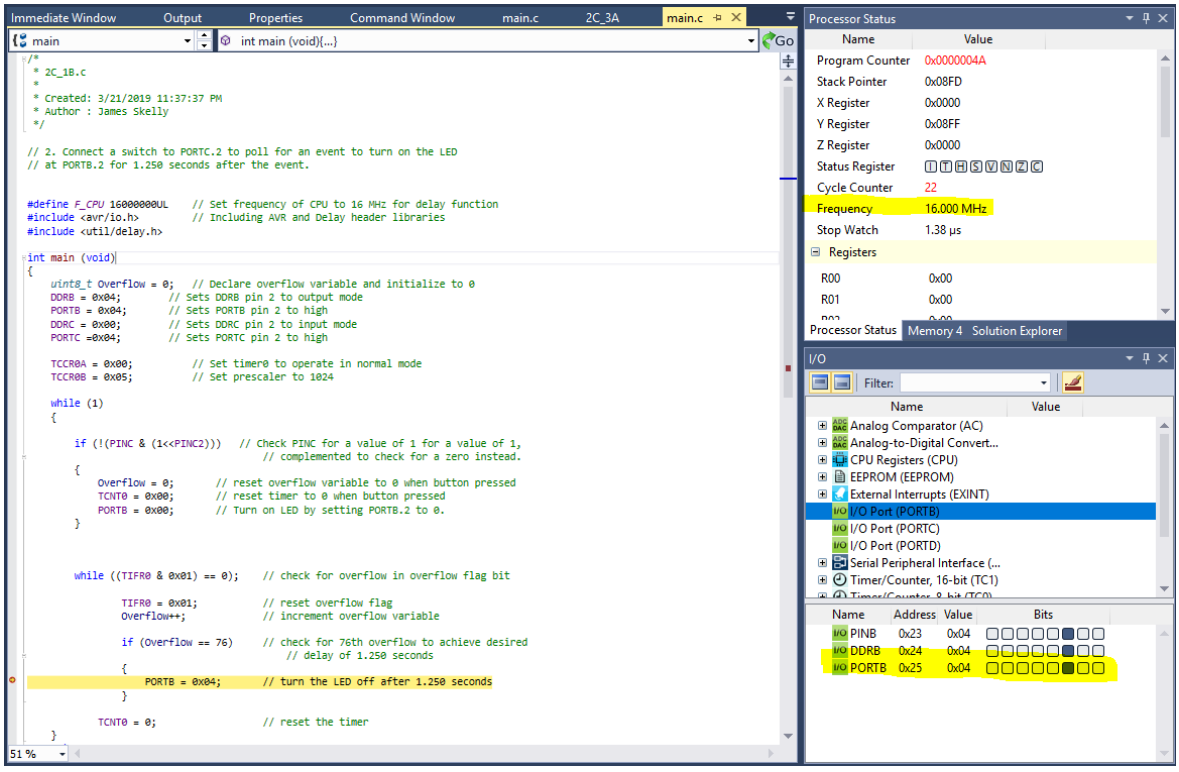


5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

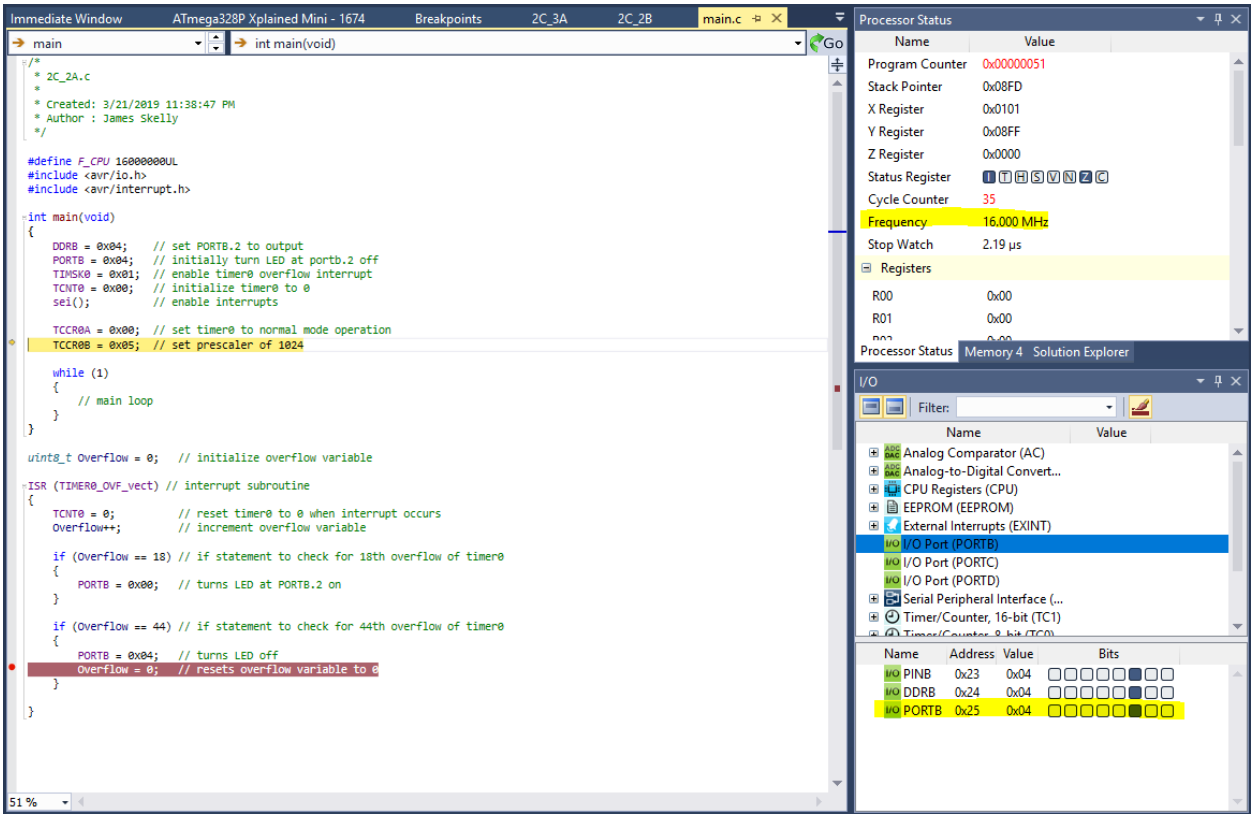
PART 1A



PART 1B



PART 2A



```
/*
 * 2C_2A.c
 *
 * Created: 3/21/2019 11:38:47 PM
 * Author : James Skelly
 */

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>

int main(void)
{
    DDRB = 0x04;    // set PORTB.2 to output
    PORTB = 0x04;   // initially turn LED at portb.2 off
    TIMSK0 = 0x01;  // enable timer0 overflow interrupt
    TCNT0 = 0x00;   // initialize timer0 to 0
    sei();          // enable interrupts

    TCCR0A = 0x00;  // set timer0 to normal mode operation
    TCCR0B = 0x05;  // set prescaler of 1024

    while (1)
    {
        // main loop
    }
}

uint8_t Overflow = 0;   // initialize overflow variable

ISR (TIMER0_OVF_vect) // interrupt subroutine
{
    TCNT0 = 0;          // reset timer0 to 0 when interrupt occurs
    Overflow++;         // increment overflow variable

    if (Overflow == 18) // if statement to check for 18th overflow of timer0
    {
        PORTB = 0x00;   // turns LED at PORTB.2 on
    }

    if (Overflow == 44) // if statement to check for 44th overflow of timer0
    {
        PORTB = 0x04;   // turns LED off
        Overflow = 0;   // resets overflow variable to 0
    }

}
```

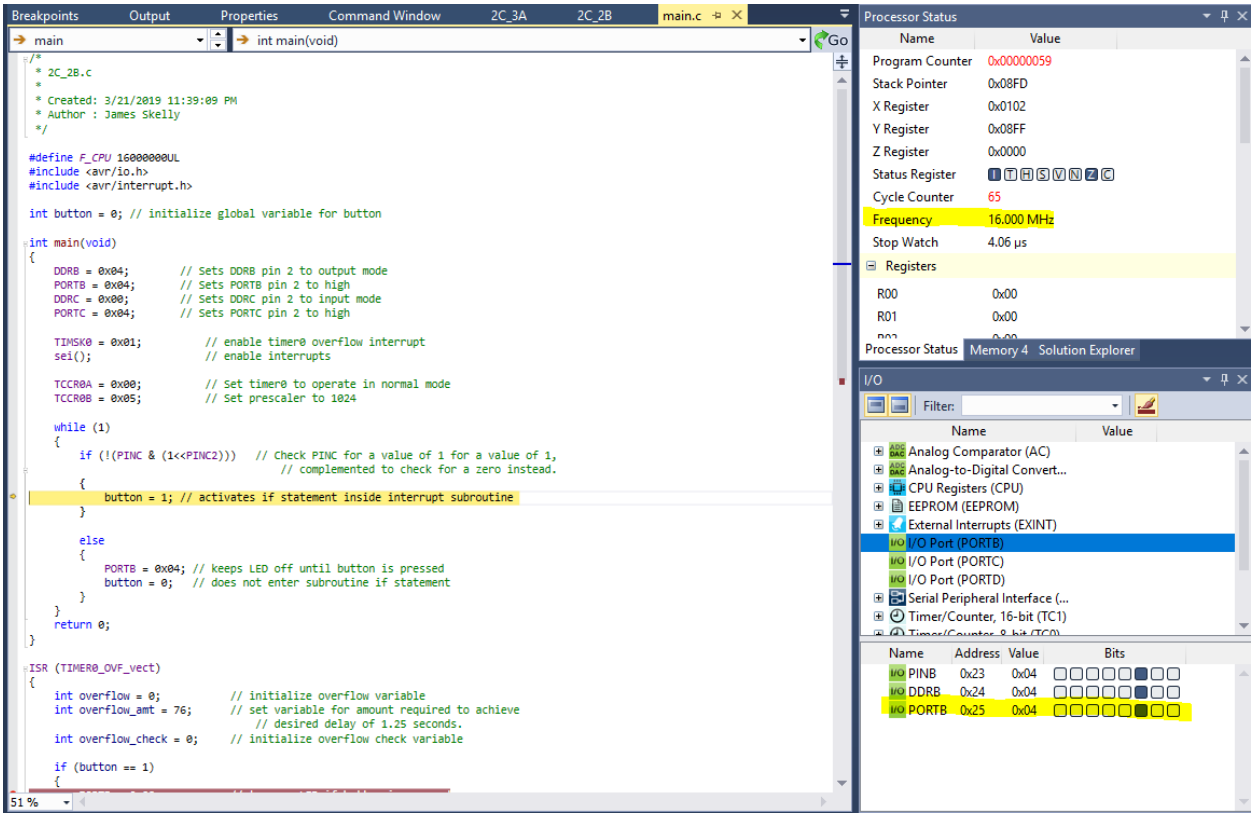
Processor Status

Name	Value
Program Counter	0x00000051
Stack Pointer	0x08FD
X Register	0x0101
Y Register	0x08FF
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	35
Frequency	16.000 MHz
Stop Watch	2.19 µs
Registers	
R00	0x00
R01	0x00

I/O

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

PART 2B



```
/*
 * 2C_2B.c
 *
 * Created: 3/21/2019 11:39:09 PM
 * Author : James Skelly
 */

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>

int button = 0; // initialize global variable for button

int main(void)
{
    DDRB = 0x04;        // Sets DDRB pin 2 to output mode
    PORTB = 0x04;       // Sets PORTB pin 2 to high
    DDRC = 0x00;        // Sets DDRC pin 2 to input mode
    PORTC = 0x04;       // Sets PORTC pin 2 to high

    TIMSK0 = 0x01;          // enable timer0 overflow interrupt
    sei();                  // enable interrupts

    TCCR0A = 0x00;          // Set timer0 to operate in normal mode
    TCCR0B = 0x05;          // Set prescaler to 1024

    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.
        {
            button = 1; // activates if statement inside interrupt subroutine
        }

        else
        {
            PORTB = 0x04; // keeps LED off until button is pressed
            button = 0;   // does not enter subroutine if statement
        }
    }
    return 0;
}

ISR (TIMER0_OVF_vect)
{
    int overflow = 0;           // initialize overflow variable
    int overflow_amt = 76;      // set variable for amount required to achieve
                                    // desired delay of 1.25 seconds.
    int overflow_check = 0;     // initialize overflow check variable

    if (button == 1)
    {
```

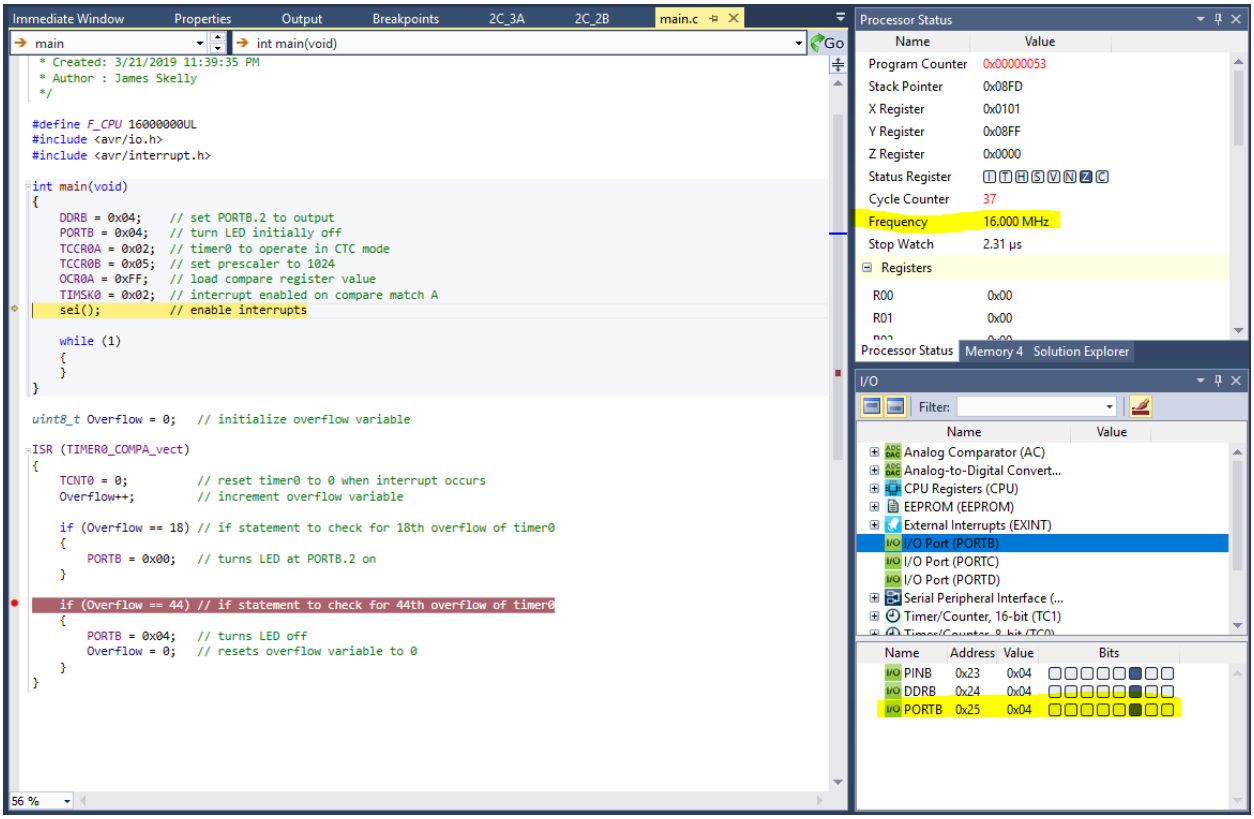
Processor Status

Name	Value
Program Counter	0x00000059
Stack Pointer	0x08FD
X Register	0x0102
Y Register	0x08FF
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	65
Frequency	16.000 MHz
Stop Watch	4.06 µs
Registers	
R00	0x00
R01	0x00

I/O

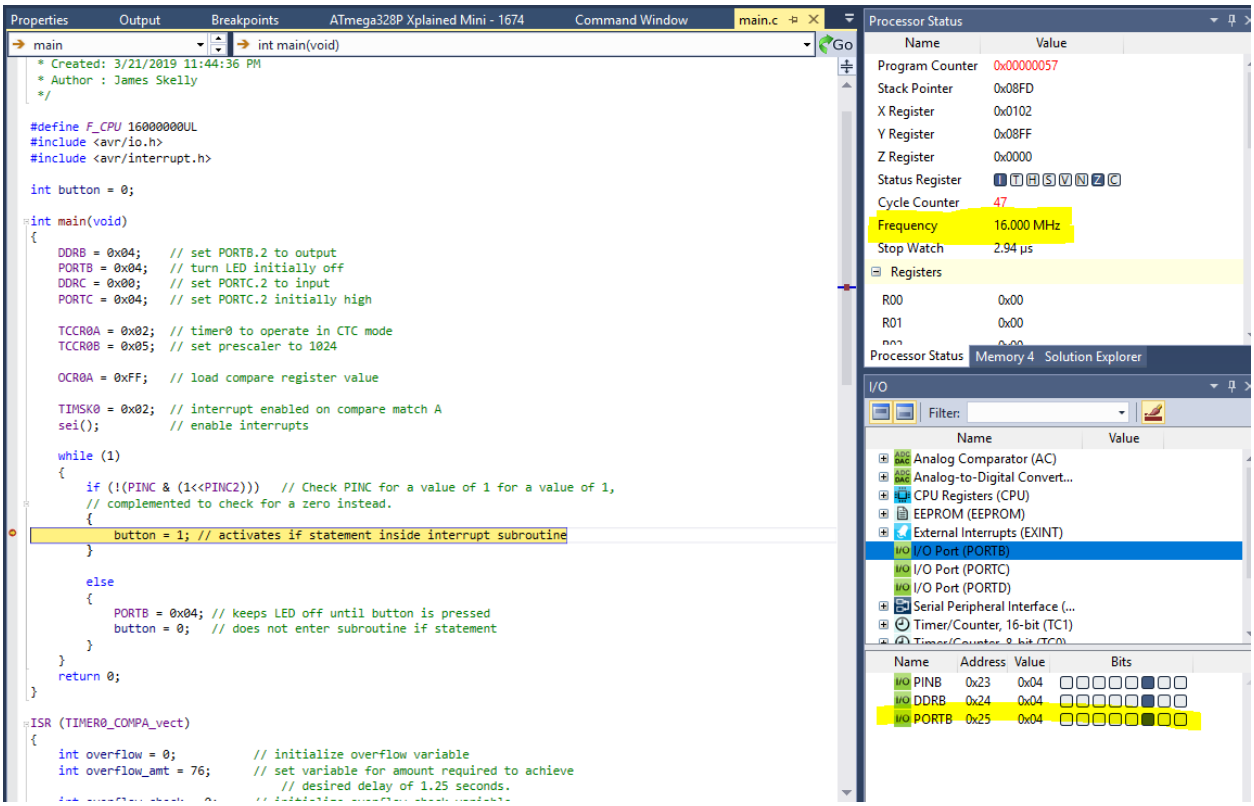
Name	Address	Value	Bits
PINB	0x23	0x04	
DDRB	0x24	0x04	
PORTB	0x25	0x04	

PART 3A



```
 * Created: 3/21/2019 11:39:35 PM
 * Author : James Skelly
 */

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>

int main(void)
{
    DDRB = 0x04;    // set PORTB.2 to output
    PORTB = 0x04;   // turn LED initially off
    TCCR0A = 0x02;  // timer0 to operate in CTC mode
    TCCR0B = 0x05;  // set prescaler to 1024
    OCR0A = 0xFF;   // load compare register value
    TIMSK0 = 0x02;  // interrupt enabled on compare match A
    sei();          // enable interrupts

    while (1)
    {
    }
}

uint8_t Overflow = 0;   // initialize overflow variable

ISR (TIMER0_COMPA_vect)
{
    TCNT0 = 0;          // reset timer0 to 0 when interrupt occurs
    Overflow++;         // increment overflow variable

    if (Overflow == 18) // if statement to check for 18th overflow of timer0
    {
        PORTB = 0x00;   // turns LED at PORTB.2 on
    }

    if (Overflow == 44) // if statement to check for 44th overflow of timer0
    {
        PORTB = 0x04;   // turns LED off
        Overflow = 0;   // resets overflow variable to 0
    }
}
```

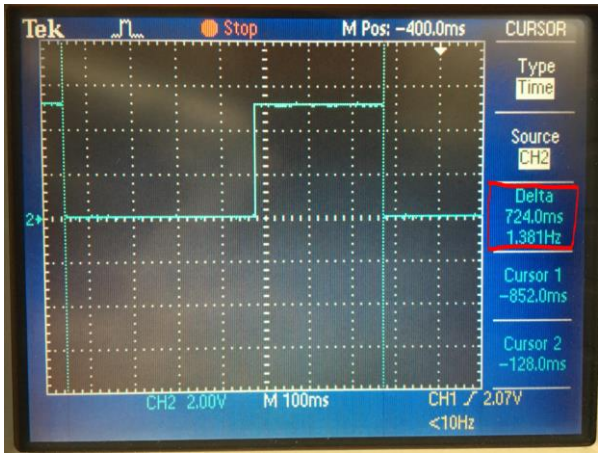
PART 3B

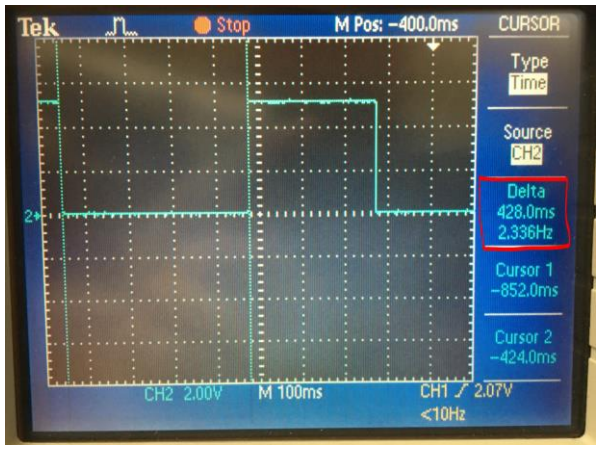


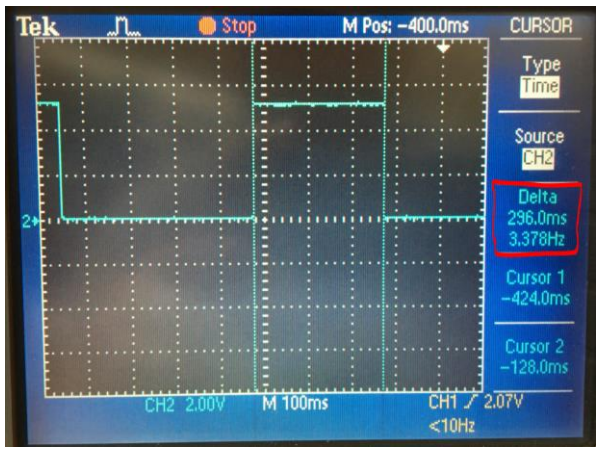
```
 * Created: 3/21/2019 11:44:36 PM
 * Author : James Skelly
 */

#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>

int button = 0;

int main(void)
{
    DDRB = 0x04;    // set PORTB.2 to output
    PORTB = 0x04;   // turn LED initially off
    DDRC = 0x00;    // set PORTC.2 to input
    PORTC = 0x04;   // set PORTC.2 initially high

    TCCR0A = 0x02;  // timer0 to operate in CTC mode
    TCCR0B = 0x05;  // set prescaler to 1024

    OCR0A = 0xFF;   // load compare register value

    TIMSK0 = 0x02;  // interrupt enabled on compare match A
    sei();          // enable interrupts

    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.
        {
            button = 1; // activates if statement inside interrupt subroutine
        }

        else
        {
            PORTB = 0x04; // keeps LED off until button is pressed
            button = 0;   // does not enter subroutine if statement
        }
    }
    return 0;
}

ISR (TIMER0_COMPA_vect)
{
    int overflow = 0;           // initialize overflow variable
    int overflow_amt = 76;      // set variable for amount required to achieve
                                // desired delay of 1.25 seconds.
```

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

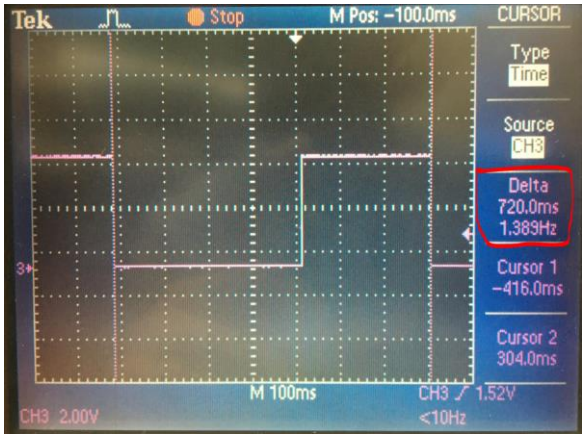
Task 1A

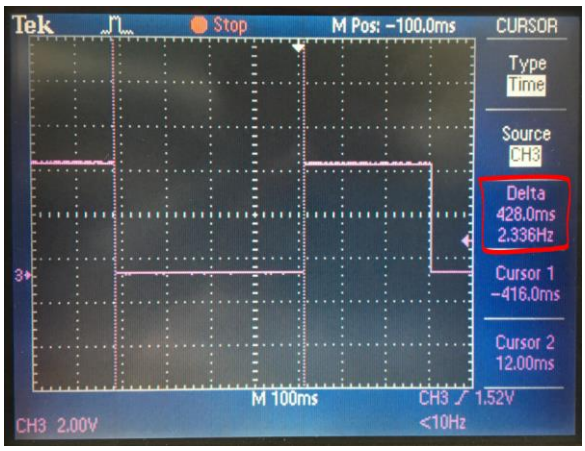


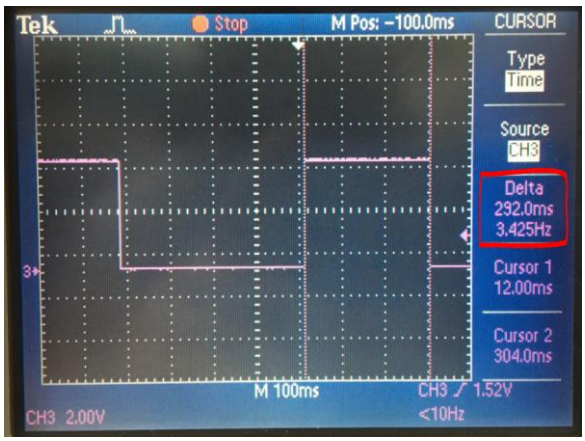




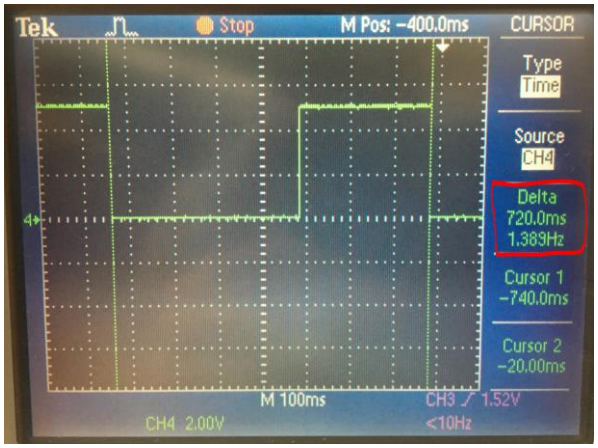
Task 2A

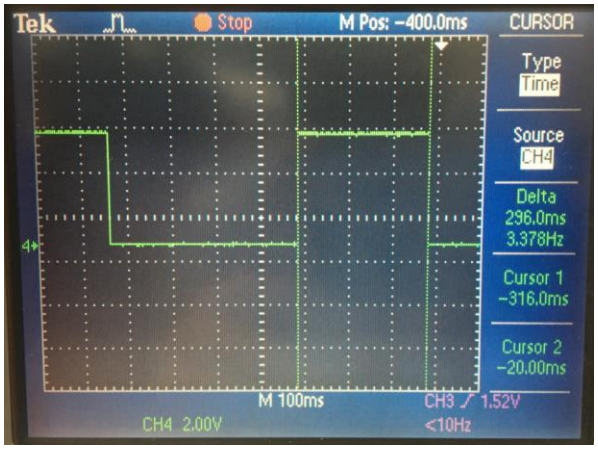

Tek
Stop
M Pos: -100.0ms
CURSOR
Type
Time
Source
CH3
Delta
720.0ms
1.389Hz
Cursor 1
-416.0ms
Cursor 2
304.0ms
M 100ms
CH3 ⁄ 1.52V
CH3 2.00V
<10Hz

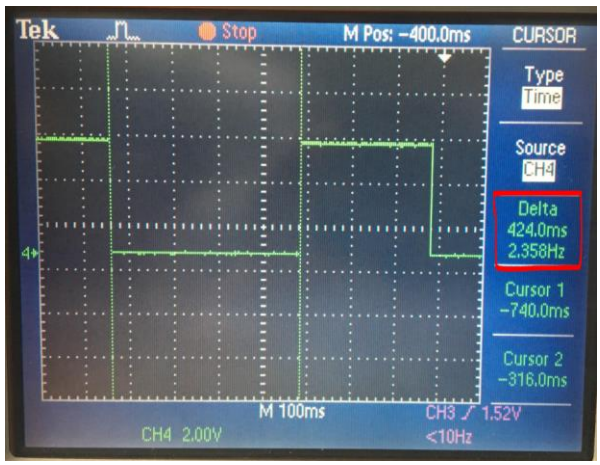

Tek
Stop
M Pos: -100.0ms
CURSOR
Type
Time
Source
CH3
Delta
428.0ms
2.336Hz
Cursor 1
-416.0ms
Cursor 2
12.00ms
M 100ms
CH3 ⁄ 1.52V
CH3 2.00V
<10Hz


Tek
Stop
M Pos: -100.0ms
CURSOR
Type
Time
Source
CH3
Delta
292.0ms
3.425Hz
Cursor 1
12.00ms
Cursor 2
304.0ms
M 100ms
CH3 ⁄ 1.52V
CH3 2.00V
<10Hz

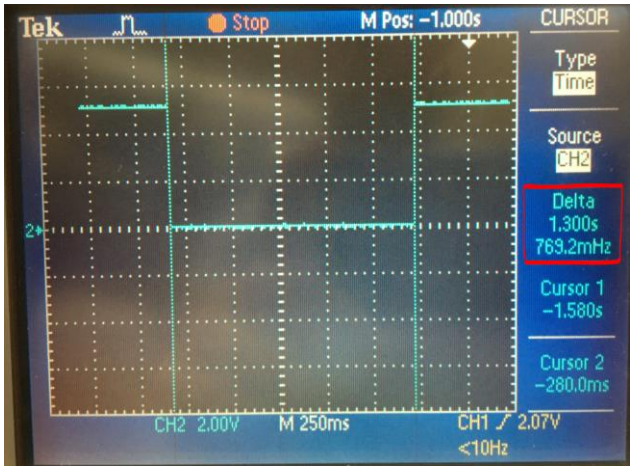
Task 3A


Tek
Stop
M Pos: -400.0ms
CURSOR
Type
Time
Source
CH4
Delta
720.0ms
1.389Hz
Cursor 1
-740.0ms
Cursor 2
-20.00ms
M 100ms
CH3 ∕ 1.52V
CH4 2.00V
<10Hz

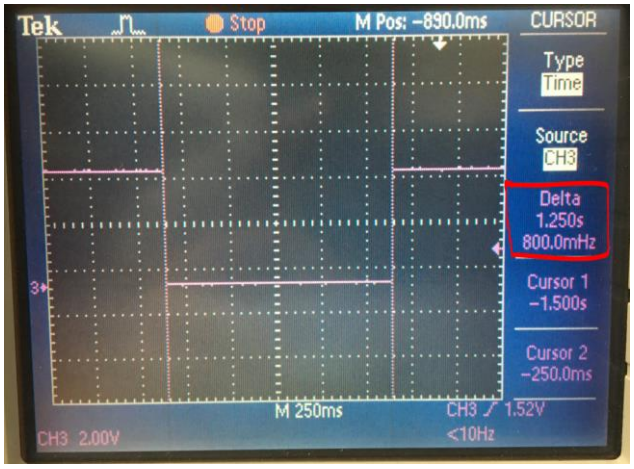

Tek
Stop
M Pos: -400.0ms
CURSOR
Type
Time
Source
CH4
Delta
296.0ms
3.378Hz
Cursor 1
-316.0ms
Cursor 2
-20.00ms
M 100ms
CH3 ∕ 1.52V
CH4 2.00V
<10Hz


Tek
Stop
M Pos: -400.0ms
CURSOR
Type
Time
Source
CH4
Delta
424.0ms
2.358Hz
Cursor 1
-740.0ms
Cursor 2
-316.0ms
M 100ms
CH3 ∕ 1.52V
CH4 2.00V
<10Hz

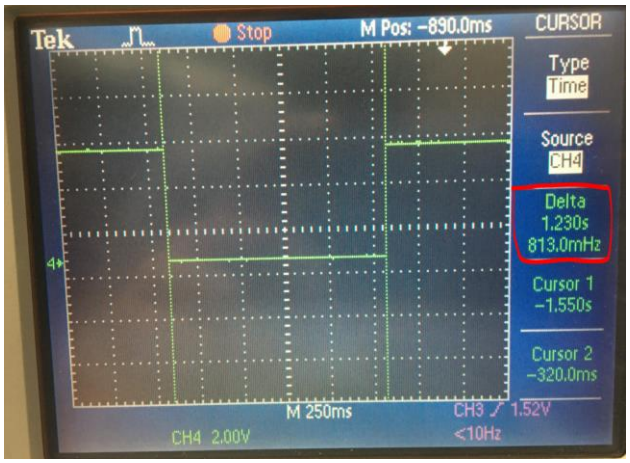
Task 1B



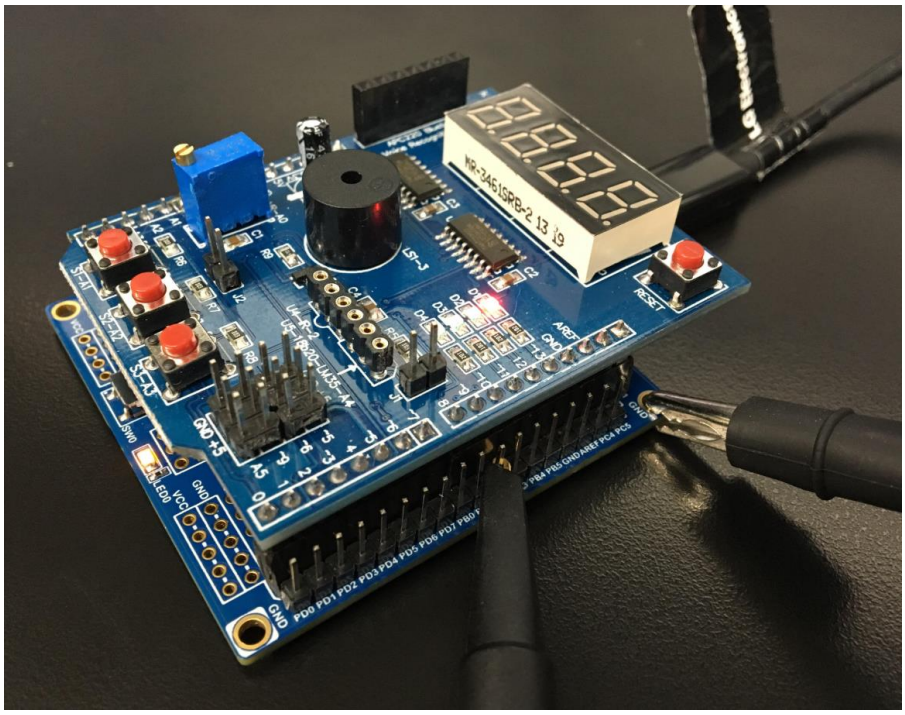
Task 2B



Task 3B



The same board set up, shown below, was used for every part of the design assignment.



7. VIDEO LINKS OF EACH DEMO

DA2C Part 1A
https://www.youtube.com/watch?v=YTjuXlmjI6k

DA2C Part 1B
https://www.youtube.com/watch?v=pFA_oJKaz0U

DA2C Part 2A
https://www.youtube.com/watch?v=D1bpT4Hpnfg

DA2C Part 2B
https://www.youtube.com/watch?v=MDWpwpgFVWA

DA2C Part 3A
https://www.youtube.com/watch?v=uaK8-E41RxU

DA2C Part 3B
https://www.youtube.com/watch?v=77TYOUd_ZcU

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