CPE301 – SPRING 2019

Design Assignment 2C

Student Name: Alex Cater

Student #: 2000583031

Student Email: cater@unlv.nevada.edu

Github address: https://github.com/katmando

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).

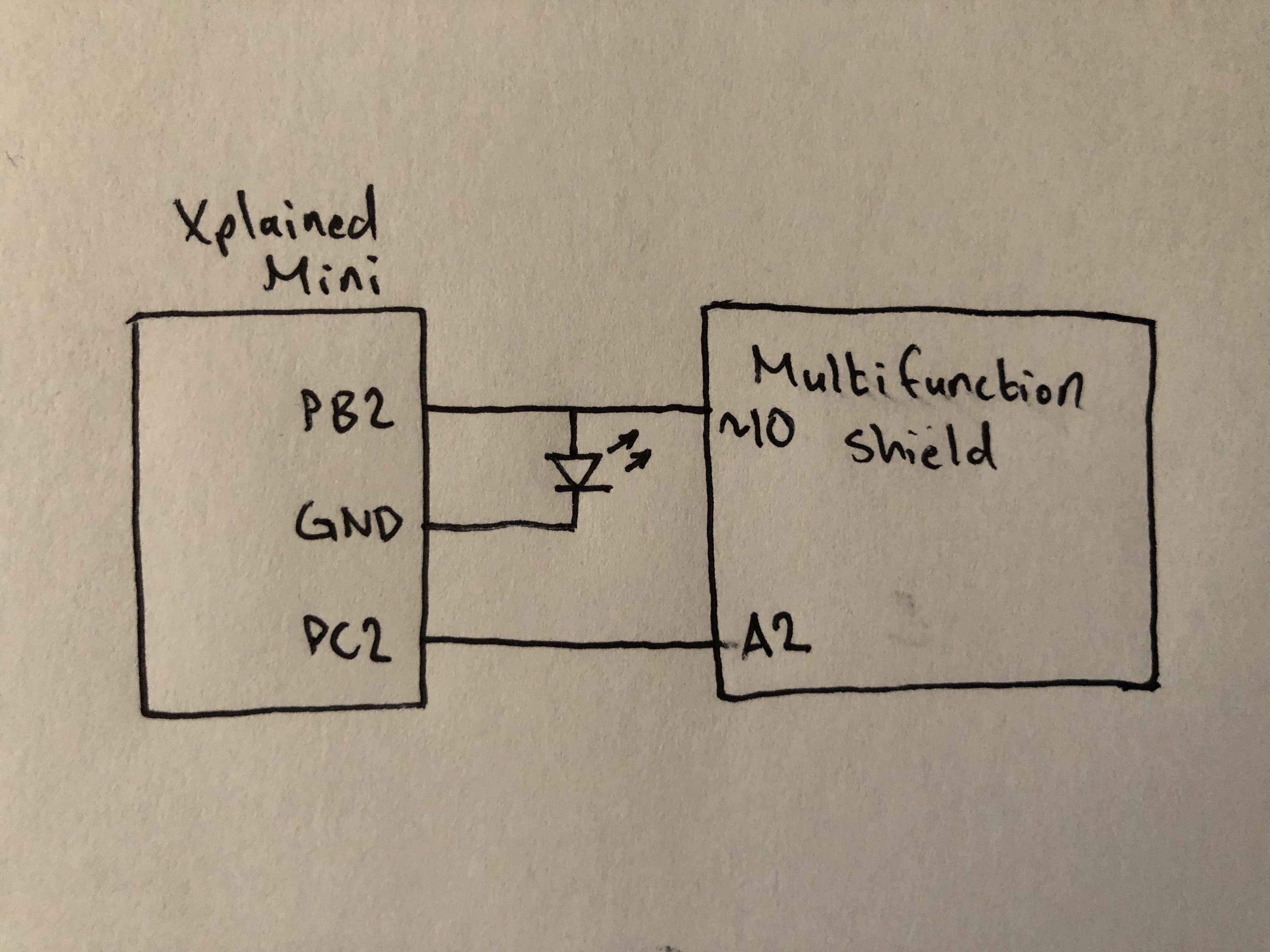
1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

Multifunction Shield

Atmega328PB Xplained Mini

Wire Connector

Micro USB Cable (Power Supply)



*Figure 1 – Block Diagram/Pin Connections*

1. **INITIAL/DEVELOPED CODE OF TASK 1/C**

#define *F\_CPU* 16000000UL // Used to slow delay from 16MHz to 1MHz (delay library)

#include <avr/io.h> // Standard AVR Library

#include <util/delay.h> // AVR library containing \_delay\_ms() function

#define LED 0b00000100 // Modify LED Bit (Currently: PB2)

#define SWITCH 0b00000100 // Modify SWITCH bit here (PC2 in program)

#define delay 19530 // 'delay = (16MHz\*1.250s/1024) - 1' for 1250ms delay

int main(void) {

unsigned char d\_end = delay/256; // Quotient of ‘Delay/Counter Size’

unsigned char d\_leftover = delay%256; // Remainder of ‘Delay/Counter Size’

unsigned char i = 0; // 8-bit Positive Counter 'i'

DDRB = LED; // Set PB2 as an Output

DDRC &= ~SWITCH; // Set PC2 as an Input

TCCR0A = (0<<WGM01)|(0<<WGM00); // Set WGM to Normal

TCCR0B = (0<<WGM02)|(1<<CS02)|(0<<CS01)|(1<<CS00); // Set WGM to Normal

// (Cont.),Prescaler '1024'

while (1) {

PORTB &= ~LED; // Set Output LED PB2 to 'Low'

PORTC |= SWITCH; // Activate Pull-up on PC2 (resistor connected to VCC)

if ((~PINC & SWITCH) == SWITCH) { // If SWITCH 'High'->LED 'ON' for 1250ms

PORTB |= LED; // Set Output LED PB2 to 'High'

i = 0; // Initialize Counter 'i' to zero

while (i < d\_end) { // Loop Counter 'i' until Delay is met

while((TIFR0 & 0x01) == 0); // Check if Timer0 Overflow Set

TCNT0 = 0x00; // If Overflow set, Restart Timer0

TIFR0 = 0x01; // And Reset Overflow Flag

i++; // Increment Counter 'i'

}

while (TCNT0 < d\_leftover); // If Counter 'i' at end of delay,

// then finish remainder

TCNT0 = 0x00; // Restart Period and Timer0

}

}

}

#define *F\_CPU* 16000000UL // Used to slow delay from 16MHz to 1MHz (delay library)

#include <avr/io.h> // Standard AVR Library

#include <avr/interrupt.h> // AVR library containing interrupt functions

#include <util/delay.h> // AVR library containing \_delay\_ms() function

#define LED 0b00000100 // Modify LED Bit (Currently: PB2)

#define SWITCH 0b00000100 // Modify SWITCH bit here (PC2 in program)

#define delay 19530 // 'delay = (16MHz\*1.250s/1024) - 1' for 1250ms delay

volatile unsigned char i = 0; // 8-bit Positive Counter 'i'

int main(void) {

unsigned char d\_end = delay/256; // Quotient of ‘Delay/Counter Size’

unsigned char d\_leftover = delay%256; // Remainder of ‘Delay/Counter Size’

DDRB = LED; // Set PB2 as an Output

DDRC &= ~SWITCH; // Set PC2 as an Input

TIMSK0 |= (1 << TOIE0); // Enable Timer0 Overflow Interrupt

sei(); // Enable Global Interrupts

TCCR0A = (0<<WGM01)|(0<<WGM00); // Set WGM to Normal

TCCR0B = (0<<WGM02)|(1<<CS02)|(0<<CS01)|(1<<CS00); // Set WGM to Normal

// (Cont.),Prescaler '1024'

while (1) {

PORTB &= ~LED; // Set Output LED PB2 to 'Low'

PORTC |= SWITCH; // Activate Pull-up on PC2 (resistor connected to VCC)

if ((~PINC & SWITCH) == SWITCH) { // If SWITCH 'High'->LED 'ON' for 1250ms

PORTB |= LED; // Set Output LED PB2 to 'High'

i = 0; // Initialize Counter 'i' to zero

while (i < d\_end); // Loop Counter 'i' until Delay is met

while (TCNT0 < d\_leftover); // If Counter 'i' at end of delay, then

// finish remainder

TCNT0 = 0x00; // Restart Period and Timer0

}

}

}

ISR(TIMER0\_OVF\_vect) {

i++; // Increment Counter 'i'

return; // Resume code from where interrupt left off

}

#define *F\_CPU* 16000000UL // Used to slow delay from 16MHz to 1MHz (delay library)

#include <avr/io.h> // Standard AVR Library

#include <avr/interrupt.h> // AVR library containing interrupt functions

#include <util/delay.h> // AVR library containing \_delay\_ms() function

#define LED 0b00000100 // Modify LED Bit (Currently: PB2)

#define SWITCH 0b00000100 // Modify SWITCH bit here (PC2 in program)

#define delay 19530 // 'delay = (16MHz\*1.250s/1024) - 1' for 1250ms delay

volatile unsigned char i = 0; // 8-bit Positive Counter 'i'

int main(void) {

unsigned char d\_end = delay/256; // Quotient of ‘Delay/Counter Size’ unsigned char d\_leftover = delay%256; // Remainder of ‘Delay/Counter Size’

DDRB = LED; // Set PB2 as an Output

DDRC &= ~SWITCH; // Set PC2 as an Input

OCR0A = 0xFF; // Load Compare Register Value

TIMSK0 |= (1 << OCIE0A); // Set Interrupt on Compare Match

sei(); // Enable Global Interrupts

TCCR0A = (0<<COM0A1)|(0<<COM0A0); // Set Compare Output Mode

TCCR0A = (0<<WGM01)|(0<<WGM00); // Set WGM to Normal

TCCR0B = (0<<WGM02)|(1<<CS02)|(0<<CS01)|(1<<CS00); // Set WGM to Normal

// (Cont.),Prescaler '1024'

while (1) {

PORTB &= ~LED; // Set Output LED PB2 to 'Low'

PORTC |= SWITCH; // Activate Pull-up on PC2 (resistor connected to VCC)

if ((~PINC & SWITCH) == SWITCH) { // If SWITCH 'High'->LED 'ON' for 1250ms

PORTB |= LED; // Set Output LED PB2 to 'High'

i = 0; // Initialize Counter 'i' to zero

while (i < d\_end); // Loop Counter 'i' until Delay is met

while (TCNT0 < d\_leftover); // If Counter 'i' at end of delay, then

// finish remainder

TCNT0 = 0x00; // Restart Period and Timer0

}

}

}

ISR(TIMER0\_COMPA\_vect) {

i++; // Increment Counter 'i'

return; // Resume code from where interrupt left off

}

1. **INITIAL/DEVELOPED CODE OF TASK 2/C**

#define *F\_CPU* 16000000UL // Used to slow delay from 16MHz to 1MHz (delay library)

#include <avr/io.h> // Standard AVR Library

#include <util/delay.h> // AVR library containing \_delay\_ms() function

#define LED 0b00000100 // Modify LED Bit (Currently: PB2)

#define SWITCH 0b00000100 // Modify SWITCH bit here (PC2 in program)

#define delay 19530 // 'delay = (16MHz\*1.250s/1024) - 1' for 1250ms delay

int main(void) {

unsigned char d\_end = delay/256; // Quotient of ‘Delay/Counter Size’

unsigned char d\_leftover = delay%256; // Remainder of ‘Delay/Counter Size’

unsigned char i = 0; // 8-bit Positive Counter 'i'

DDRB = LED; // Set PB2 as an Output

DDRC &= ~SWITCH; // Set PC2 as an Input

TCCR0A = (0<<WGM01)|(0<<WGM00); // Set WGM to Normal

TCCR0B = (0<<WGM02)|(1<<CS02)|(0<<CS01)|(1<<CS00); // Set WGM to Normal

// (Cont.),Prescaler '1024'

while (1) {

PORTB &= ~LED; // Set Output LED PB2 to 'Low'

PORTC |= SWITCH; // Activate Pull-up on PC2 (resistor connected to VCC)

if ((~PINC & SWITCH) == SWITCH) { // If SWITCH 'High'->LED 'ON' for 1250ms

PORTB |= LED; // Set Output LED PB2 to 'High'

i = 0; // Initialize Counter 'i' to zero

while (i < d\_end) { // Loop Counter 'i' until Delay is met

while((TIFR0 & 0x01) == 0); // Check if Timer0 Overflow Set

TCNT0 = 0x00; // If Overflow set, Restart Timer0

TIFR0 = 0x01; // And Reset Overflow Flag

i++; // Increment Counter 'i'

}

while (TCNT0 < d\_leftover); // If Counter 'i' at end of delay,

// then finish remainder

TCNT0 = 0x00; // Restart Period and Timer0

}

}

}

#define *F\_CPU* 16000000UL // Used to slow delay from 16MHz to 1MHz (delay library)

#include <avr/io.h> // Standard AVR Library

#include <avr/interrupt.h> // AVR library containing interrupt functions

#include <util/delay.h> // AVR library containing \_delay\_ms() function

#define LED 0b00000100 // Modify LED Bit (Currently: PB2)

#define SWITCH 0b00000100 // Modify SWITCH bit here (PC2 in program)

#define delay 19530 // 'delay = (16MHz\*1.250s/1024) - 1' for 1250ms delay

volatile unsigned char i = 0; // 8-bit Positive Counter 'i'

int main(void) {

unsigned char d\_end = delay/256; // Quotient of ‘Delay/Counter Size’

unsigned char d\_leftover = delay%256; // Remainder of ‘Delay/Counter Size’

DDRB = LED; // Set PB2 as an Output

DDRC &= ~SWITCH; // Set PC2 as an Input

TIMSK0 |= (1 << TOIE0); // Enable Timer0 Overflow Interrupt

sei(); // Enable Global Interrupts

TCCR0A = (0<<WGM01)|(0<<WGM00); // Set WGM to Normal

TCCR0B = (0<<WGM02)|(1<<CS02)|(0<<CS01)|(1<<CS00); // Set WGM to Normal

// (Cont.),Prescaler '1024'

while (1) {

PORTB &= ~LED; // Set Output LED PB2 to 'Low'

PORTC |= SWITCH; // Activate Pull-up on PC2 (resistor connected to VCC)

if ((~PINC & SWITCH) == SWITCH) { // If SWITCH 'High'->LED 'ON' for 1250ms

PORTB |= LED; // Set Output LED PB2 to 'High'

i = 0; // Initialize Counter 'i' to zero

while (i < d\_end); // Loop Counter 'i' until Delay is met

while (TCNT0 < d\_leftover); // If Counter 'i' at end of delay, then

// finish remainder

TCNT0 = 0x00; // Restart Period and Timer0

}

}

}

ISR(TIMER0\_OVF\_vect) {

i++; // Increment Counter 'i'

return; // Resume code from where interrupt left off

}

#define *F\_CPU* 16000000UL // Used to slow delay from 16MHz to 1MHz (delay library)

#include <avr/io.h> // Standard AVR Library

#include <avr/interrupt.h> // AVR library containing interrupt functions

#include <util/delay.h> // AVR library containing \_delay\_ms() function

#define LED 0b00000100 // Modify LED Bit (Currently: PB2)

#define SWITCH 0b00000100 // Modify SWITCH bit here (PC2 in program)

#define delay 19530 // 'delay = (16MHz\*1.250s/1024) - 1' for 1250ms delay

volatile unsigned char i = 0; // 8-bit Positive Counter 'i'

int main(void) {

unsigned char d\_end = delay/256; // Quotient of ‘Delay/Counter Size’

unsigned char d\_leftover = delay%256; // Remainder of ‘Delay/Counter Size’

DDRB = LED; // Set PB2 as an Output

DDRC &= ~SWITCH; // Set PC2 as an Input

OCR0A = 0xFF; // Load Compare Register Value

TIMSK0 |= (1 << OCIE0A); // Set Interrupt on Compare Match

sei(); // Enable Global Interrupts

TCCR0A = (0<<COM0A1)|(0<<COM0A0); // Set Compare Output Mode

TCCR0A = (0<<WGM01)|(0<<WGM00); // Set WGM to Normal

TCCR0B = (0<<WGM02)|(1<<CS02)|(0<<CS01)|(1<<CS00); // Set WGM to Normal

// (Cont.),Prescaler '1024'

while (1) {

PORTB &= ~LED; // Set Output LED PB2 to 'Low'

PORTC |= SWITCH; // Activate Pull-up on PC2 (resistor connected to VCC)

if ((~PINC & SWITCH) == SWITCH) { // If SWITCH 'High'->LED 'ON' for 1250ms

PORTB |= LED; // Set Output LED PB2 to 'High'

i = 0; // Initialize Counter 'i' to zero

while (i < d\_end); // Loop Counter 'i' until Delay is met

while (TCNT0 < d\_leftover); // If Counter 'i' at end of delay, then

// finish remainder

TCNT0 = 0x00; // Restart Period and Timer0

}

}

}

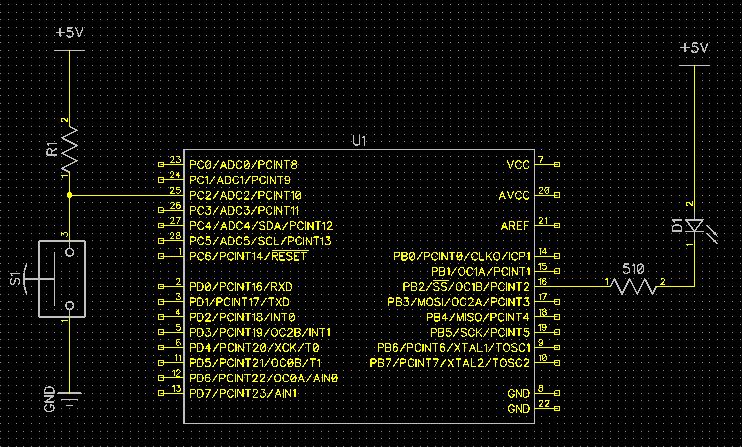
ISR(TIMER0\_COMPA\_vect) {

i++; // Increment Counter 'i'

return; // Resume code from where interrupt left off

}

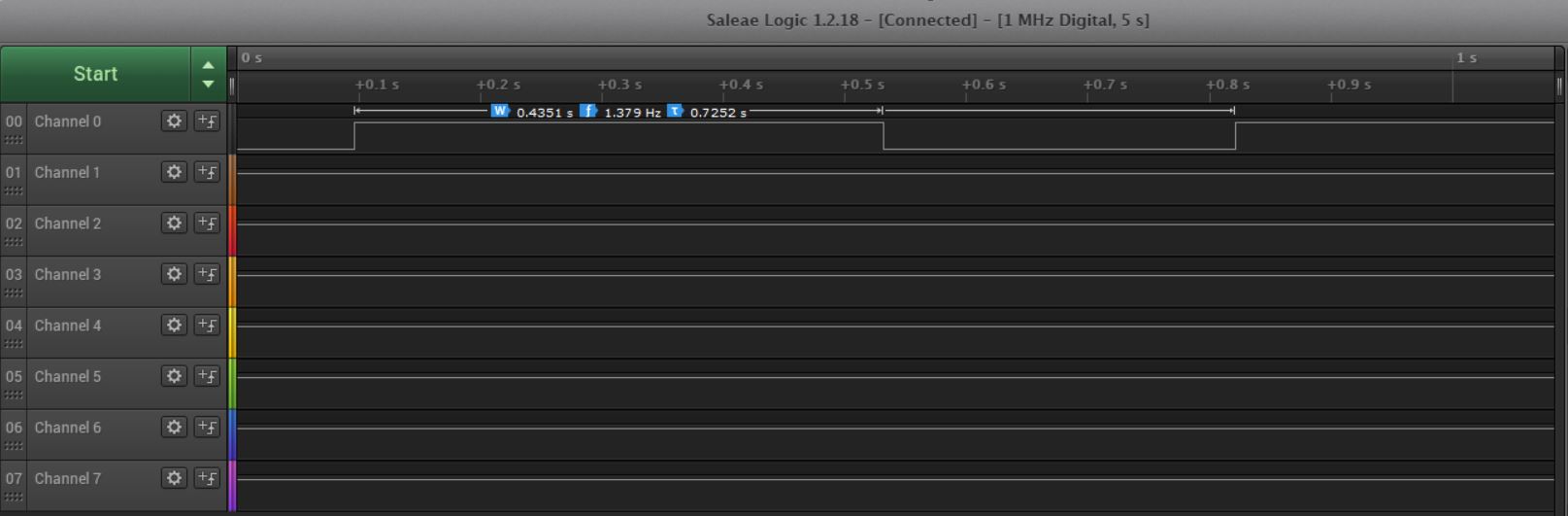
1. **SCHEMATICS**



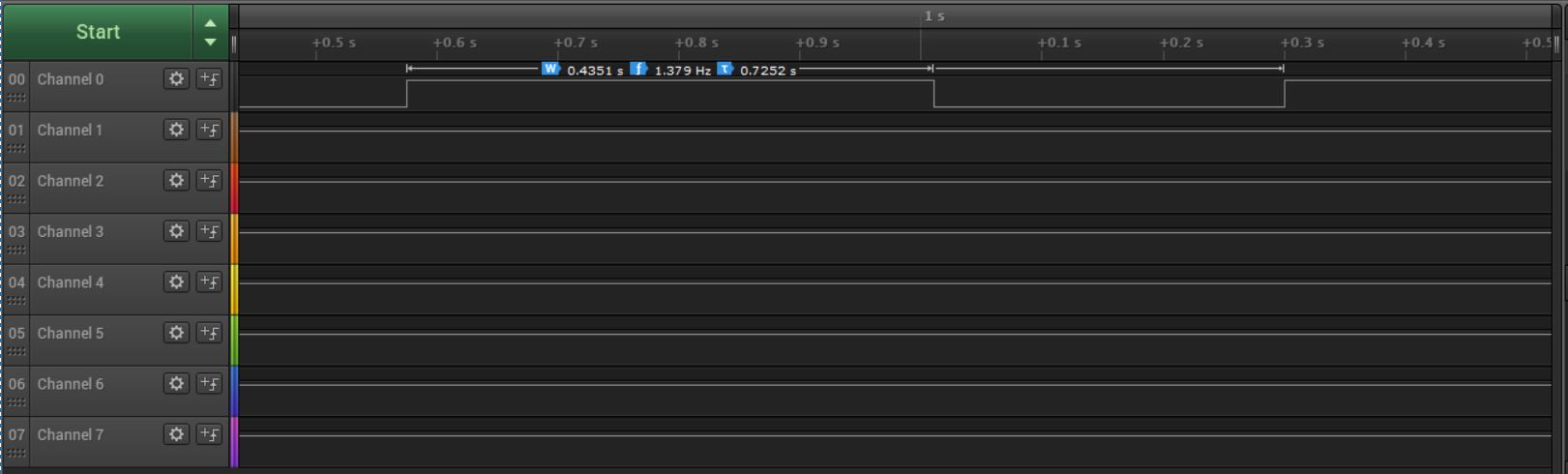
*Figure 2 – Schematic Connections for DA2C Task 1 and Task 2*

1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

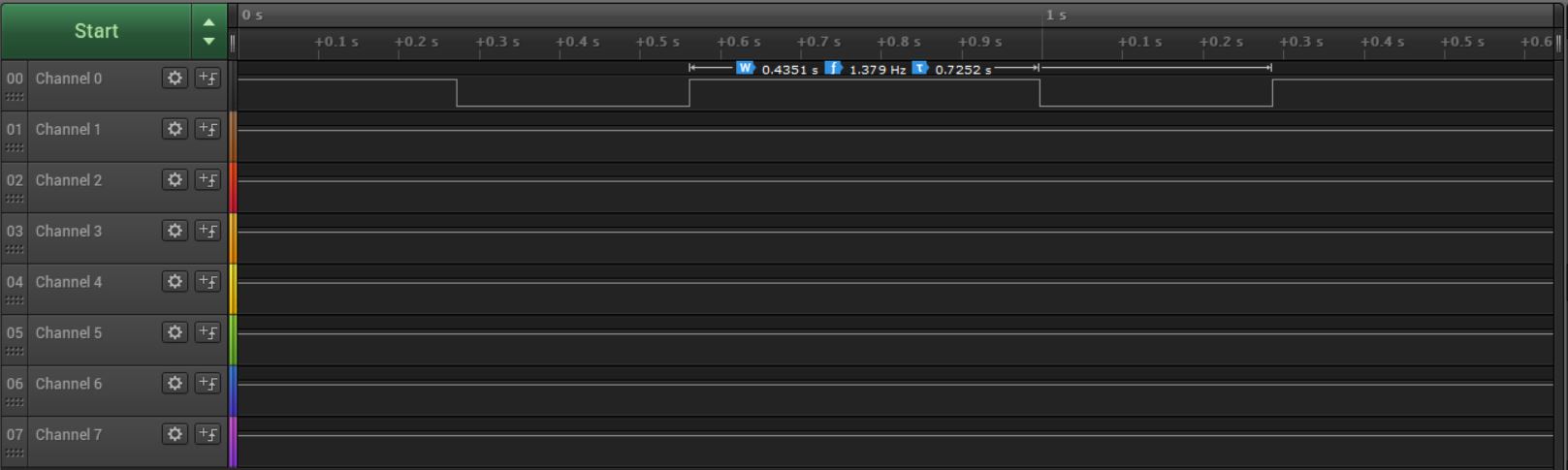
Using a Logic Analyzer for Task 1 and Task 2, we can see the same response as previously demonstrated for DA2A except with the use of Timers and Timer interrupts. The same goal is accomplished which the results are portrayed below in *Figure 3(a-f)*:



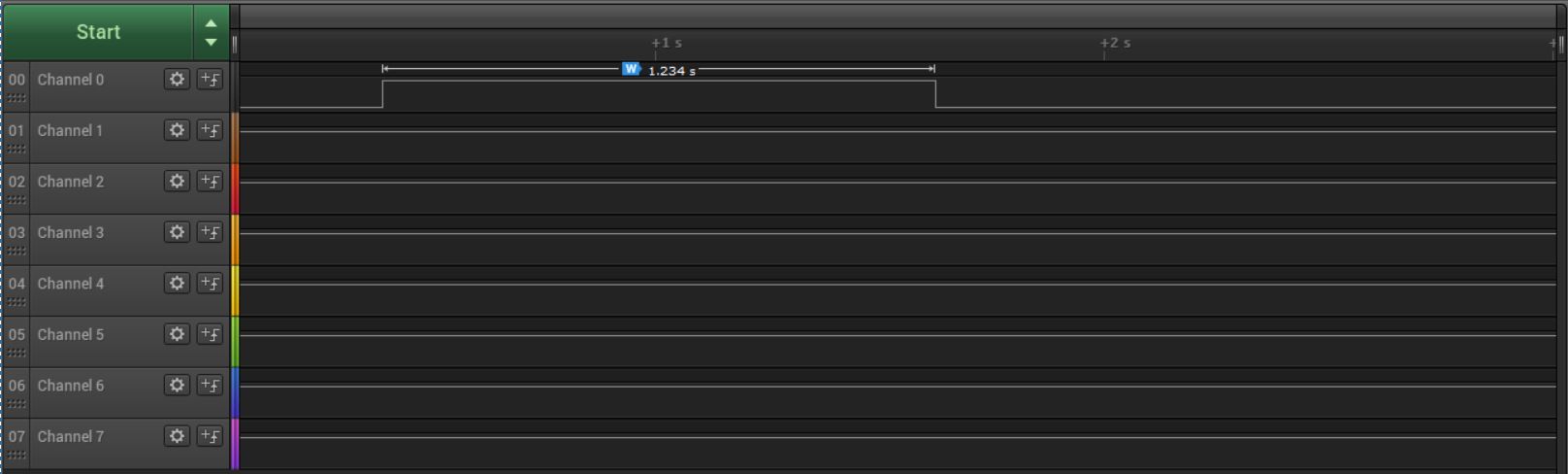
*Figure 3a – Output Waveform of Task 1/DA2C Normal (C Coding)*



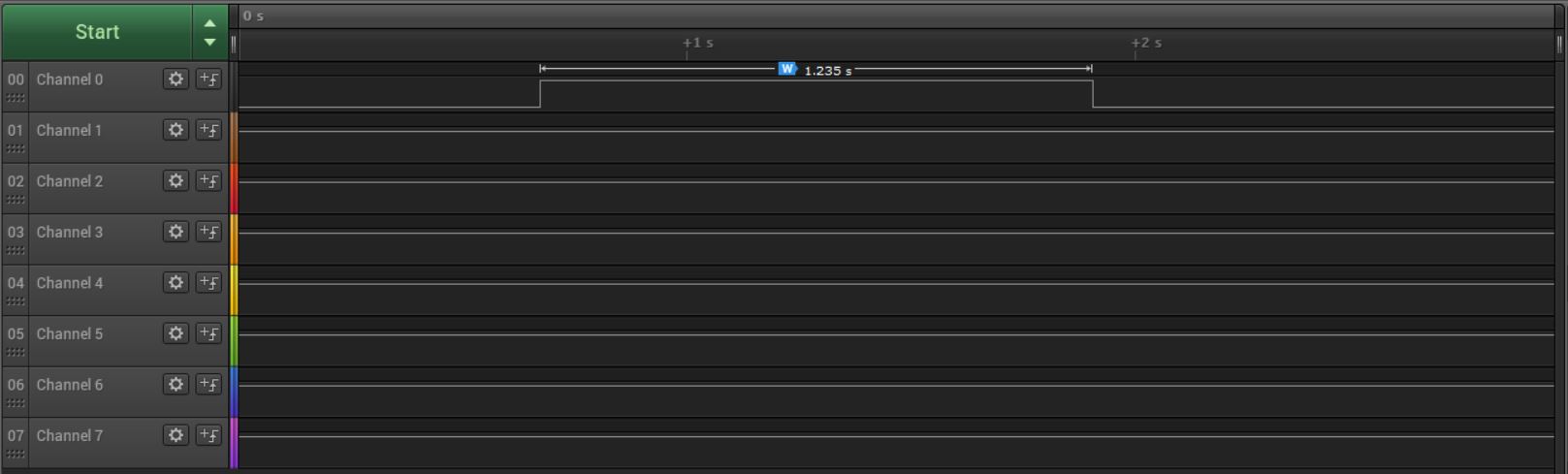
*Figure 3b – Output Waveform of Task 1/DA2C Normal with Interrupt (C Coding)*



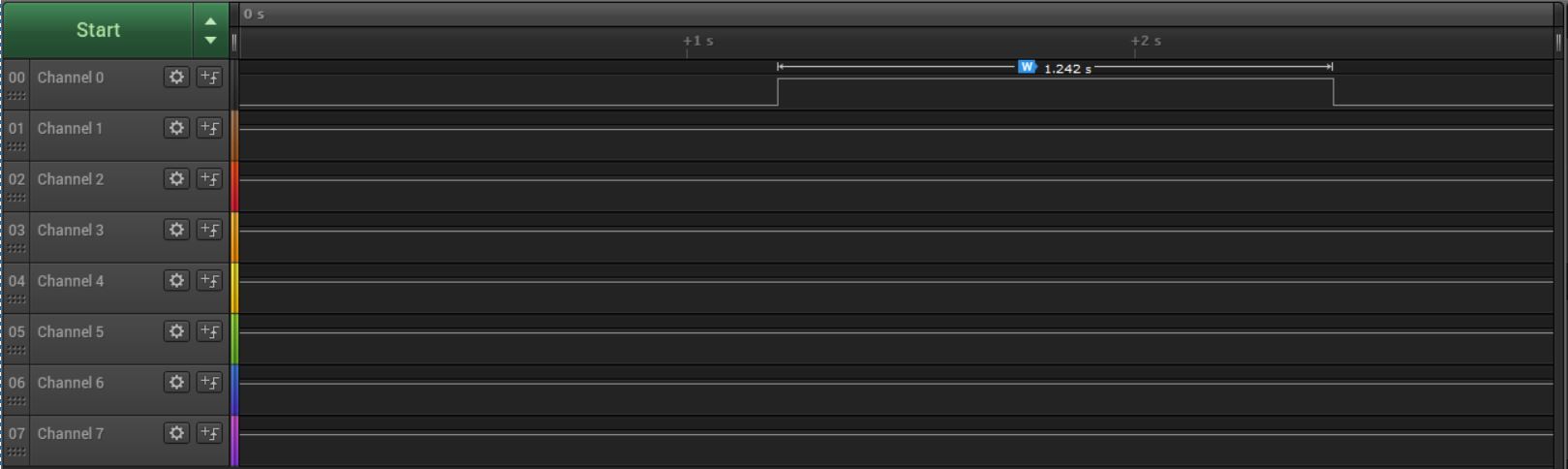
*Figure 3c – Output Waveform of Task 1/DA2C CTC with Interrupt (C Coding)*



*Figure 3d – Output Waveform of Task 2/DA2C Normal (C Coding)*



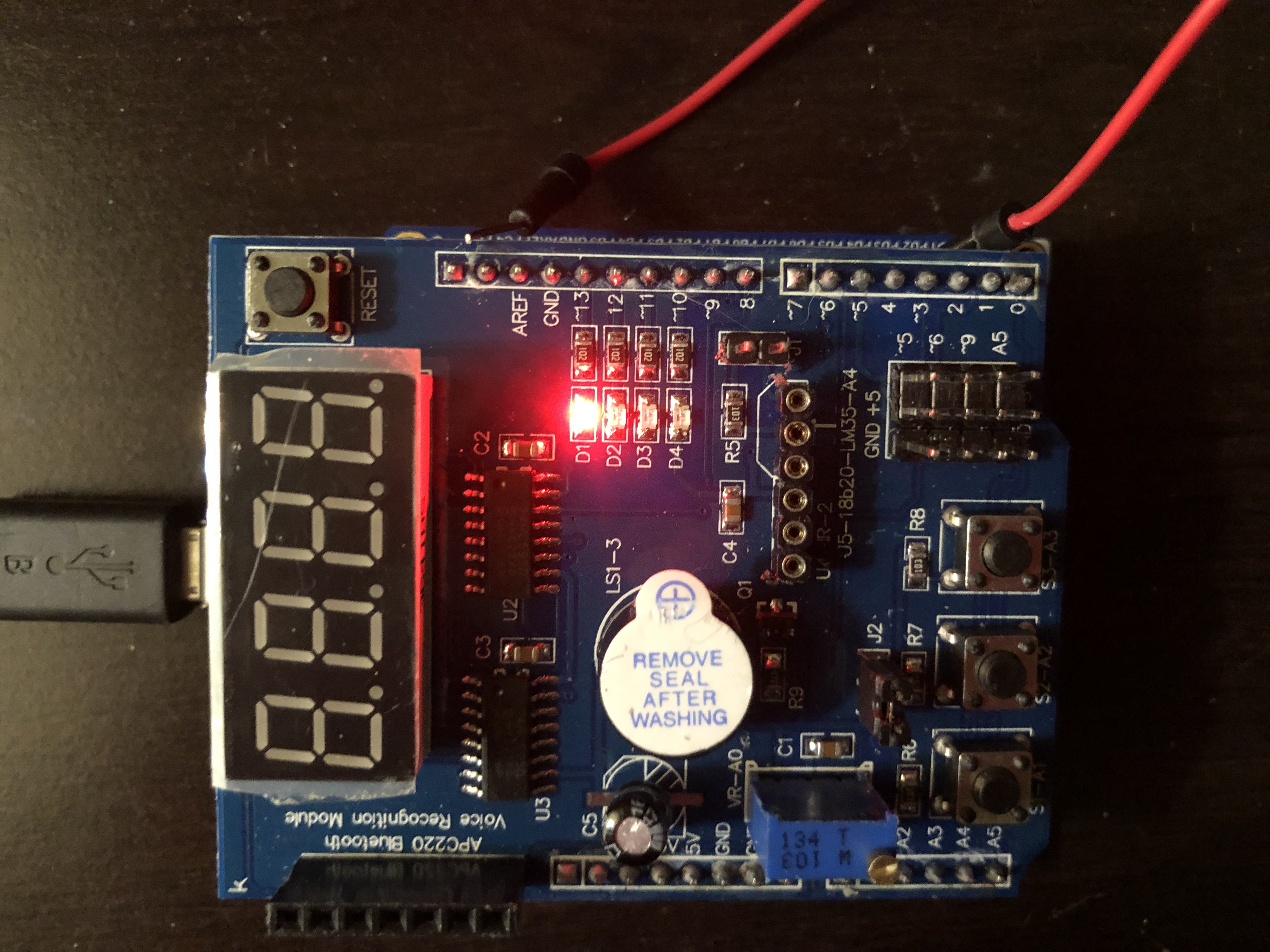
*Figure 3e – Output Waveform of Task 2/DA2C Normal with Interrupt (C Coding)*



*Figure 3f – Output Waveform of Task 2/DA2C CTC with Interrupt (C Coding)*

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

Each Demo utilizes the same set up which includes a Shield Attachment placed on top of the Xplained Mini PB and a red wire connector from PD2 ready to connect to Ground. This set up is shown in *Figure 4*:



*Figure 4 – Wire attached from PD2 (Xplained Mini Board)*

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Alex Cater