CPE301 – SPRING 2019

Design Assignment DA2C

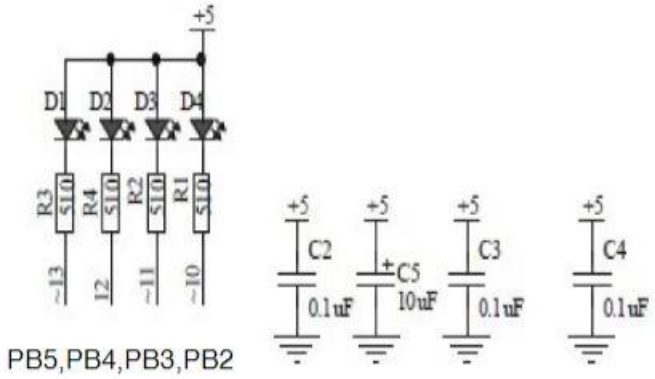
Student Name: Jett Guerrero
Student #: 5002377713
Student Email: guerrj1@unlv.nevada.edu
Primary Github address: https://github.com/guerrj1/Submission_DA.git
Directory: DA2C - https://github.com/guerrj1/Submission_DA/tree/master/DA2C

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

-ATMega328p
-Arduino Shield
-Micro-USB Cable



PB2 and PC2 PINS were used on the Atmega328P and Arduino Shield

2. DEVELOPED CODE OF TASK 1 C CODE

Task 1 Part 1

```
//DA2CT1_1.c

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

int main()
{
	uint8_t OVFCount = 0;				//overflow flag counter
	DDRB |= (1<<2);					//set pb2 as output
	PORTB |= (1<<2);				//led output
	TCNT0 = 0x1D;					//starting counter
	TCCR0B = (1<<CS02) | (1 << CS00);		//sets prescaler to 1024

	while(1)
	{
		while((TIFR0 & 0x01) == 0);	//while flag is 0

			TCNT0 = 0x1D;			//starting counter for tcnt0
			TIFR0 = 0x01;			//resets the overflow flag
			//OVFCount++;

			if(OVFCount == 19 )		//overflow counter counts up to this
			{
				PORTB ^= (1<<2);	//turns led off
			}
			else if(OVFCount == 49)	//overflow counter counts up to this
			{
				PORTB ^= (1<<2);	//keeps led on
				OVFCount = 0;		//resets the overflow count
			}
			OVFCount++;			//increment overflow count
	}
	return 0;
}
```

Task 1 Part 2

```
// DA2CT1_2.c

#define F_CPU 16000000UL                //sets CPU clock speed
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
        DDRB |= (1<<2);                     //sets pb2 as input
        PORTB |= (1<<2);                    //sets pb2 to high for LED
        DDRC &= (0<<2);                     //sets PC2 to input
        PORTC |= (0<<2);                    //sets PC2 to low 0

        TCCR0A = 0;
        TCCR0B = (1<<CS02) | (1<<CS00);     //sets prescaler to 1024
        int OVFCount = 0;

        while (1)
        {
                if (!(PINC & (1<<PINC2)))    //checks if the pushbutton is pressed
                {
                        OVFCount = 0;
                        TCNT0 = 0;
                }

                while ((TIFR0 & 0x01) == 0);

                        TCNT0 = 0x05;                //starts at this value to count from
                        TIFR0 = 0x01;                //rests the overflow flag
                        OVFCount++;                     //overflow flag counter increment

                if (OVFCount <= 78)             //when overflow counter is less than or equal
to 78
                {
                        PORTB = (0<<2);          //then portb2 led is on
                }
                else
                {
                PORTB = (1<<2);          //then portb2 led is off
                }
        }
        return 0;
}
```

3. DEVELOPED CODE FOR TASK 2 C CODE

Task 2 Part 1

```
//DA2CT2_1.c

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

int main(void)
{
	DDRB |= (1<<2);                    //sets pb2 as output
	PORTB |= (1<<2);                   //sets pb2 led
	//TCNT0 = 0x1D;
	TCNT0 = 0;
	TCCR0A = 0;
	TCCR0B = (1<<CS02) | (1 << CS00);  //sets prescaler to 1024

	TIMSK0 |= (1<< TOIE0);             //enables interrupt
	sei();                             //enables interrupt

	while(1)
	{

	}
}

ISR(TIMER0_OVF_vect)
{
	uint8_t OVFCount = 0;

	while(1)
	{
		while((TIFR0 & 0x01) == 0);      //while flag is 0

		TCNT0 = 0x1D;                //starting counter for tcnt0
		TIFR0 = 0x01;                //resets the overflow flag
		//OVFCount++;

		if(OVFCount == 19 )          //overflow counter counts up to this
		{
			PORTB ^= (1<<2);         //turns led off
		}
		else if(OVFCount == 49)      //overflow counter counts up to this
		{
			PORTB ^= (1<<2);       //keeps led on
			OVFCount = 0;          //resets the overflow count
		}
		OVFCount++;                      //increment overflow count
	}

}
```

Task 2 Part 2

```
// DA2CT2_2.c

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

int main(void)
{
	DDRB |= (1<<2);                    //sets pb2 as output
	PORTB |= (1<<2);                   //sets pb2 led
	//TCNT0 = 0x1D;
	TCNT0 = 0;
	TCCR0A = 0;
	TCCR0B = (1<<CS02) | (1 << CS00);  //sets prescaler to 1024

	TIMSK0 |= (1<< TOIE0);             //enables interrupt
	sei();                             //enables interrupt

	while(1)
	{

	}
}

ISR(TIMER0_OVF_vect)
{
	int OVFCount = 0;
	while (1)
	{
		if (!(PINC & (1<<PINC2)))    //checks if the pushbutton is pressed
		{
			OVFCount = 0;
			TCNT0 = 0;
		}

		while ((TIFR0 & 0x01) == 0);

			TCNT0 = 0x05;                //starts at this value to count from
			TIFR0 = 0x01;                //rests the overflow flag
			OVFCount++;                     //overflow flag counter increment

		if (OVFCount <= 78)             //when overflow counter is less than or equal
to 78
		{
			PORTB = (0<<2);          //then portb2 led is on
		}
		else
		{
		PORTB = (1<<2);          //then portb2 led is off
		}
	}
}
```

4. DEVELOPED CODE FOR TASK 3 C CODE

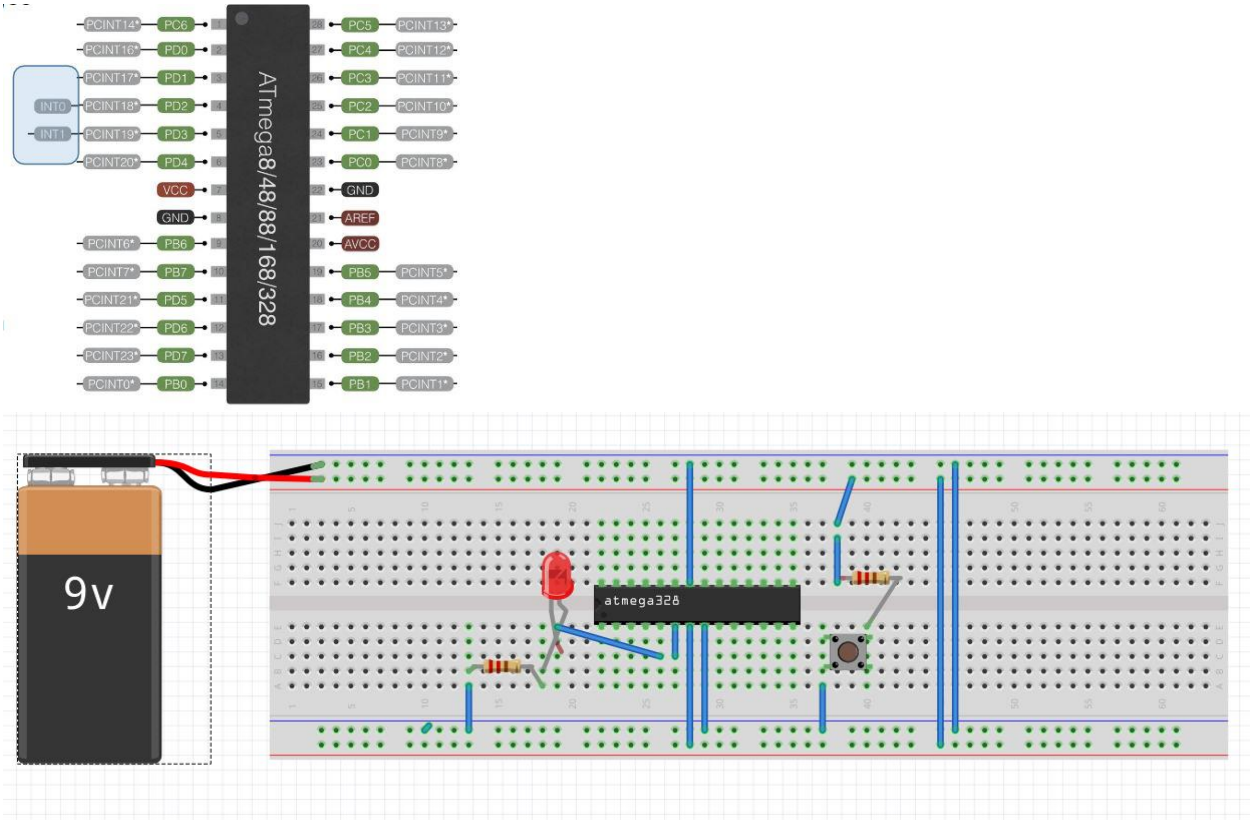
Task 3 Part 1

```
//DA2CT3_1.c

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

int main(void)
{
	DDRB |= (1<<2);                    //sets pb2 as output
	PORTB |= (1<<2);                   //sets for pb2 led
	//TCNT0 = 0x1D;
	TCNT0 = 0;
	TCCR0A = (1<<WGM01);                //sets CTC mode
	TCCR0B = (1<<CS02) | (1 << CS00);   //sets prescaler to 2014

	TIMSK0 |= (1<<OCIE0A);              //sets interrupt compare match
	sei();                              //enables interrupt

	while(1)
	{

	}
}

ISR(TIMER0_COMPA_vect)
{
	uint8_t OVFCount = 0;

	while(1)
	{
		while((TIFR0 & 0x01) == 0);      //while flag is 0

		TCNT0 = 0x1D;                //starting counter for tcnt0
		TIFR0 = 0x01;                //resets the overflow flag
		//OVFCount++;

		if(OVFCount == 19 )          //overflow counter counts up to this
		{
			PORTB ^= (1<<2);         //turns led off
		}
		else if(OVFCount == 49)      //overflow counter counts up to this
		{
			PORTB ^= (1<<2);       //keeps led on
			OVFCount = 0;          //resets the overflow count
		}
		OVFCount++;                      //increment overflow count
	}

}
```

Task 3 Part 2

```
//DA2CT3_2.c

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

int main(void)
{
	DDRB |= (1<<2);                     //sets pb2 as output
	PORTB |= (1<<2);                    //sets pb2 led
	//TCNT0 = 0x1D;
	TCNT0 = 0;
	TCCR0A = (1<<WGM01);                //sets ctc mode
	TCCR0B = (1<<CS02) | (1 << CS00);   //sets prescaler to 1024

	TIMSK0 |= (1<<OCIE0A);              //sets interrupt on compare match
	sei();                              //enables interrupt

	while(1)
	{

	}
}

ISR(TIMER0_COMPA_vect)
{
	int OVFCount = 0;
	while (1)
	{
		if (!(PINC & (1<<PINC2)))    //checks if the pushbutton is pressed
		{
			OVFCount = 0;
			TCNT0 = 0;
		}

		while ((TIFR0 & 0x01) == 0);

		TCNT0 = 0x05;                //starts at this value to count from
		TIFR0 = 0x01;                //rests the overflow flag
		OVFCount++;                     //overflow flag counter increment

		if (OVFCount <= 78)             //when overflow counter is less than or equal
to 78
		{
			PORTB = (0<<2);          //then portb2 led is on
		}
		else
		{
			PORTB = (1<<2);          //then portb2 led is off
		}
	}
}
```

5. SCHEMATICS



Breadboard Schematic using Fritzing

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

Duty Cycle and Period Verification for Task 1-3 Part 1

Processor Status

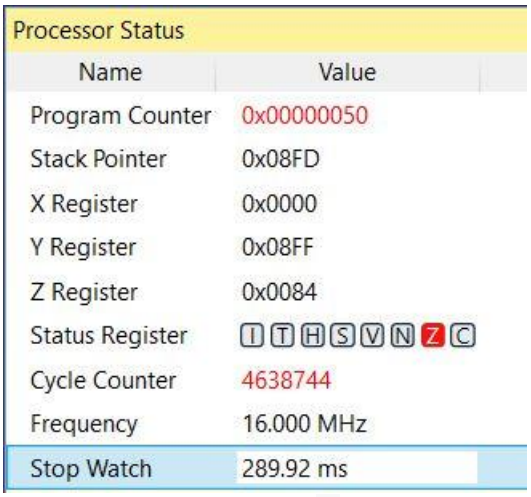
Name	Value
Program Counter	0x0000000D
Stack Pointer	0x08FF
X Register	0x0000
Y Register	0x1A8C
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	6959129
Frequency	16.000 MHz
Stop Watch	434.95 ms

60% Duty Cycle 434.95ms ≈ 0.435s

Processor Status

Name	Value
Program Counter	0x00000013
Stack Pointer	0x08FF
X Register	0x0000
Y Register	0x11B2
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	11597843
Frequency	16.000 MHz
Stop Watch	724.87 ms

Total Period 724.87ms ≈ 0.725s

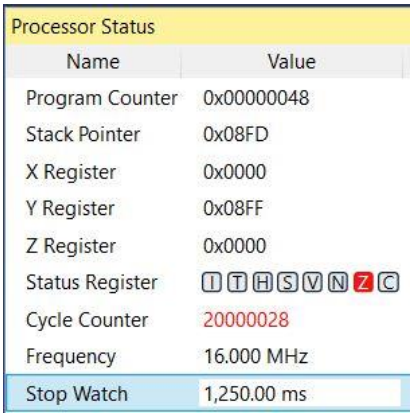


40% Duty Cycle 289.92ms ≈ 0.29s

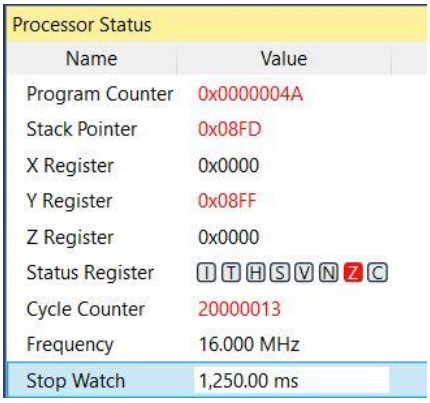
40% Duty Cycle 290.56 ms ≈ 0.29s

Delay Verification for Task 1-3 Part 2

Processor Status Simulation for 1.25 sec Delay

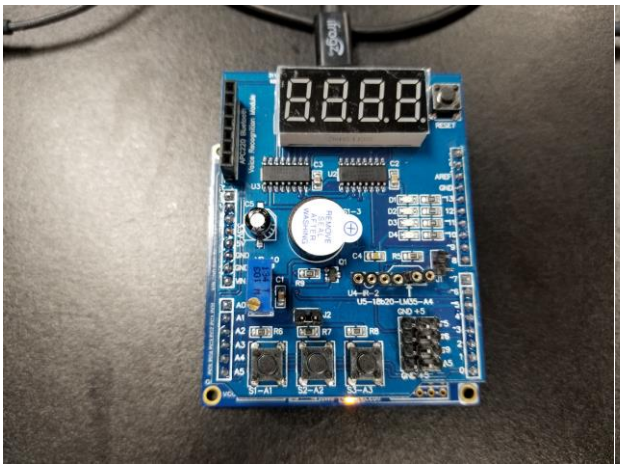


Processor Status Simulation for 1.25 sec Delay

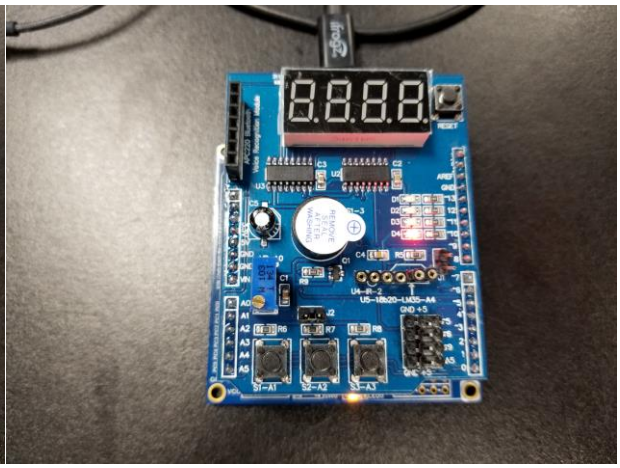


7. SCREENSHOT OF EACH DEMO (BOARD SETUP)

Task 1-3 Part 1

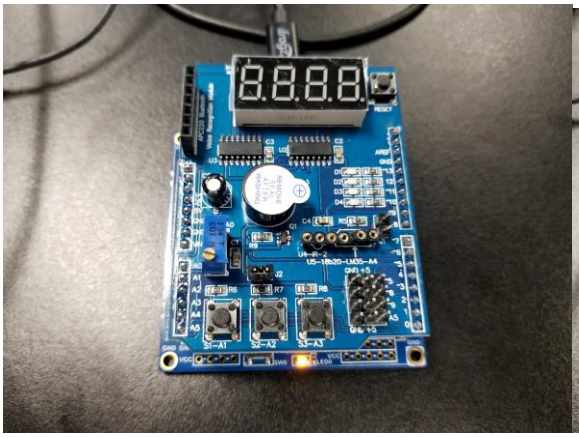


LED OFF



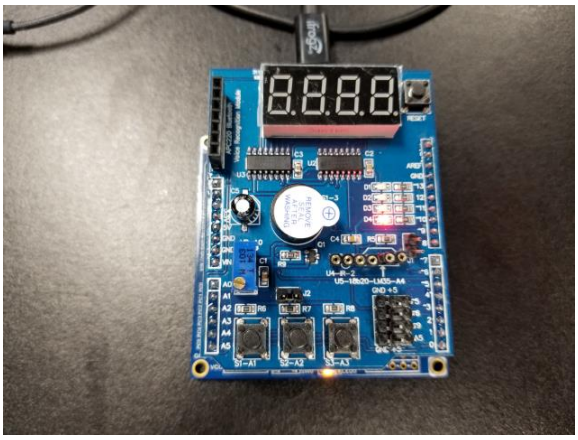
LED ON

Task 1-3 Part 2



LED OFF

LED Turns OFF After Push Button is Pressed



LED Stays ON for the 1.250s

8. VIDEO LINKS OF EACH DEMO

Task 1: Part 1 & Part 2 C Code Demo
https://youtu.be/Jo5-k52JTmM

Task 2: Part 1 & Part 2 C Code Demo
https://youtu.be/h_VvDzXfMnA

Task 3: Part 1 & Part 2 C Code Demo
https://youtu.be/ZZFKP8Zk034

9. GITHUB LINK OF THIS DA

https://github.com/guerrj1/Submission_DA/tree/master/DA2C

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

"*This assignment submission is my own, original work*".
Jett Guerrero