CPE301 – SPRING 2019

Design Assignment 2C

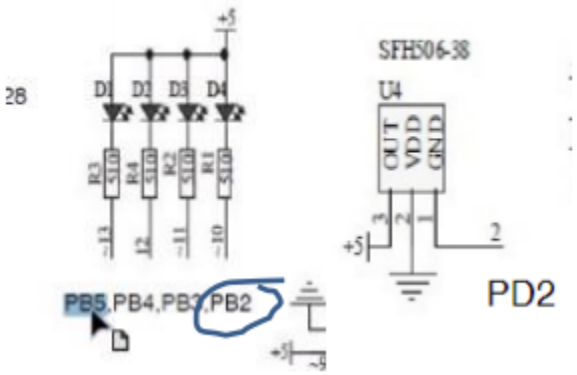
Student Name: Itzel Becerril
Student #: 2000478001
Student Email: becerri2@unlv.nevada.edu
Primary Github address: HadidBuilds
Directory:DA2C

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

List of Components Used:

- Atmel Studio 7
- Atmega328P
- Xplained mini
- MicroUSB Cord
- Multifunctional Shield
- Switch on LED Board

Block diagram with pins used in the Atmega328P



2. DEVELOPED MODIFIED CODE OF TASK 1/C from DA2

Modified TASK 1 from 2A using Normal Mode Timer 0:

```
#include <avr/io.h>

int main(void)
{
 uint8_t OVFCount = 0;//set to 0
 TCCR0B = 5;//prescaler 1024
 DDRB |= (1<<2);//set as output
 PORTB |= (1<<2);//set low
 DDRB |= (1<<5);//set as output
 PORTB |= (1<<5);//set low
 while (1)
  {
          OVFCount = 0;//reset to 0
             TCNT0 = 0;//reset to 0
             while(OVFCount <= 25)
             {
                    TCNT0 = 0;//reset to 0
```

```
                        while ((TIFR0 & 0x01) == 0)
                        {
                                PORTB ^= (1<<2);//turn LED on
                        }
                        TCNT0 = 0;//reset to 0
                        TIFR0=0x01;//reset flag
                        OVFCount++;//increment counter
                }
                OVFCount = 0;//reset to 0
                TCNT0 = 0;//reset to 0
                while(OVFCount <= 17)
                {
                        while ((TIFR0 & 0x01) == 0)
                        {
                                PORTB |= (1<<2);//turn off LED
                        }
                        OVFCount++;//increment counter
                        TIFR0 = 0x01;//reset flag
                        TCNT0 = 0;//reset to 0
                }
    }
}
```

Modified TASK 2 from 2A using Normal Mode Timer 0:

```
#define F_CPU 16000000UL //Setting to 16Mhz

#include <avr/io.h>

int main(void)
{
        uint8_t OVFCount = 0; //set OVFCOUNT to zero
        DDRB |= (1<<2); //set PB.2 as an output
        DDRB |= (1<<5); //set PB.5 as an output
        PORTB |= (1<<2); //set low
        PORTB |= (1<<5); //set low

        DDRC &= (0<<2); //set PC.2 as an input
        PORTC |= (0<<2); //set

        TCCR0A = 0; // Normal Operation
        TCNT0=0; // start the timer at 0
        TCCR0B |= 5; //set prescaler 1024

        while (1)
        {
                if(!(PINC & (1<<PINC2))) //if high enter statement
                {
                        TCNT0 = 0; //reset TCNT0

                        while(OVFCount < 77) //while less then 0
                        {
                                while ((TIFR0 & 0x01) == 0) //while not 255
                                {
                                        PORTB &= ~(1<<2); //turn PB.2 on
                                }
                                OVFCount++; //increment counter
                                TIFR0=0x01; //reset flag
                                TCNT0 = 0; //reset TCNT0
                        }
```

```
                TCNT0 = 0;//reset TCNT0
                OVFCount = 0; //set OVFcount to 0
        }
        else
        {
                PORTB |= (1<<2); //turn PB.2 off
        }
    }
    return 0;
    }
```

3. DEVELOPED MODIFIED CODE OF TASK 2/C from DA2

Modified TASK 1 from 2A Duty Cycle with Interrupt:

```
#define F_CPU 16000000UL //Setting to 16Mhz
#include <avr/interrupt.h>
#include <avr/io.h>

int main(void)
{
    DDRB |= (1<<2); //set PB.2 as an output
    DDRB |= (1<<5); //set PB.5 as an output
    PORTB |= (1<<2); //set low
    PORTB |= (1<<5); //set low

    DDRC &= (0<<2); //set PC.2 as an input
    PORTC |= (0<<2); //set

    TIMSK0 |= (1<<TOIE0); //enabling timer overflow interrupt

    TCNT0 = 0; // start the timer at 0

    sei();//enable interrupts
    TCCR0B = 5;   //set prescaler

    while (1)
    {
        //main loop
    }
}

ISR(TIMER0_OVF_vect)//timer 0 overflow interrupt
{
    uint8_t OVFCount = 0; //set to 0

    OVFCount = 0;//set to 0
    TCNT0 = 0; //set to 0
    while(OVFCount <= 25)
    {
        TCNT0 = 0;//set to 0
        while ((TIFR0 & 0x01) == 0)
        {
            PORTB ^= (1<<2); //light up LED
        }
        TCNT0 = 0;//set to 0
        TIFR0=0x01;//reset flag
        OVFCount++;//increment counter
    }
```

```
	OVFCount = 0;//reset to 0
	TCNT0 = 0;//reset to 0
	while(OVFCount <= 17)
	{
		while ((TIFR0 & 0x01) == 0)
		{
			PORTB |= (1<<2); //keep LED off
		}
		OVFCount++;//increment counter
		TIFR0 = 0x01;//reset flag
		TCNT0 = 0;//reset to 0
	}
	}
```

Modified TASK 2 from 2A Switch with Interrupt :

```
#define F_CPU 16000000UL //Setting to 16Mhz
#include <avr/interrupt.h>
#include <avr/io.h>

int main(void)
{
	DDRB |= (1<<2); //set PB.2 as an output
	DDRB |= (1<<5); //set PB.5 as an output
	PORTB |= (1<<2); //set low
	PORTB |= (1<<5); //set low

	DDRC &= (0<<2); //set PC.2 as an input
	PORTC |= (0<<2); //set

	TIMSK0 |= (1<<TOIE0); //enabling timer overflow interrupt

	TCNT0 = 0; // start the timer at 0

	sei();//enable interrupts
	TCCR0B = 5;   //set prescaler

	while (1)
	{
	 //main loop
	}
}

ISR(TIMER0_OVF_vect)//timer 0 overflow interrupt
{
		uint8_t OVFCount = 0; //set OVFCOUNT to zero
		if(!(PINC & (1<<PINC2))) //if high enter statement
		{
			TCNT0 = 0; //reset TCNT0

			while(OVFCount < 77) //while less then 0
			{
				while ((TIFR0 & 0x01) == 0) //while not 255
				{
					PORTB &= ~(1<<2); //turn PB.2 on
				}
				OVFCount++; //increment counter
				TIFR0=0x01; //reset flag
				TCNT0 = 0; //reset TCNT0
			}
```

```
                        TCNT0 = 0;//reset TCNT0
                        OVFCount = 0; //set OVFcount to 0
                }
                else
                {
                        PORTB |= (1<<2); //turn PB.2 off
                }
}
```

4. DEVELOPED MODIFIED CODE OF TASK 3/C from DA2

Modified TASK 1 from 2A CTC Mode Duty Cycle:

```
#define F_CPU 16000000UL //Setting to 16Mhz
#include <avr/interrupt.h>
#include <avr/io.h>

int main(void)
{
        DDRB |= (1<<2); //set PB.2 as an output
        DDRB |= (1<<5); //set PB.5 as an output
        PORTB |= (1<<2); //set low
        PORTB |= (1<<5); //set low

        OCR0A = 0xFF; //set output compare register
        TCCR0A = (1<<WGM01); //set mode to CTC
        TCCR0B = (1<<CS02) | (1<<CS00); //set prescaler to 1024
        TIMSK0 |= (1<<OCIE0A); //enable overflow interrupt
        TCNT0 = 0;//set to 0

        sei();//enable interrupts

        while (1)
        {
                //main loop
        }
}

ISR(TIMER0_COMPA_vect)//timer 0 overflow interrupt
{
        uint8_t OVFCount = 0; //set to 0

        OVFCount = 0;//set to 0
        TCNT0 = 0; //set to 0
        while(OVFCount <= 25)
        {
                TCNT0 = 0;//set to 0
                while ((TIFR0 & (1<<OCF0A)) == 0)
                {
                        PORTB ^= (1<<2); //light up LED
                }
                TCNT0 = 0;//set to 0
                TIFR0 |= (1<<OCF0A);//reset flag
                OVFCount++;//increment counter
        }
        OVFCount = 0;//reset to 0
        TCNT0 = 0;//reset to 0
        while(OVFCount <= 17)
        {
```

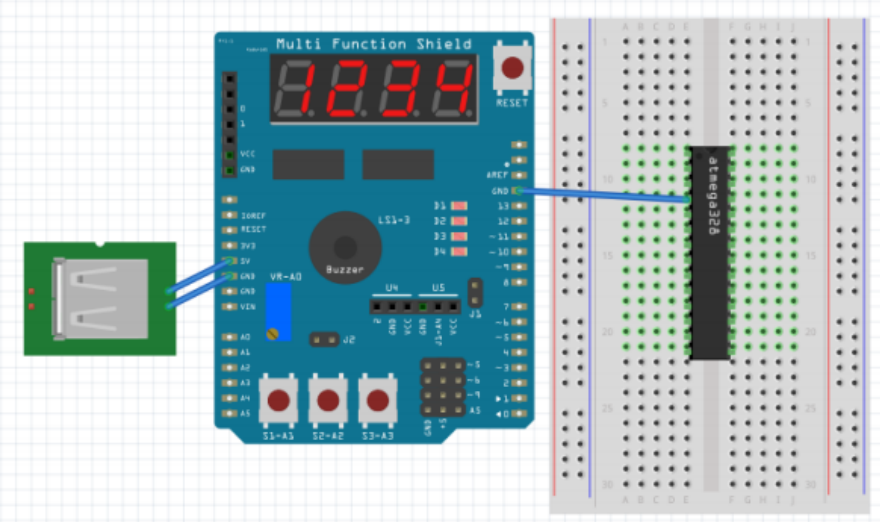
```
		while ((TIFR0 & (1<<OCF0A)) == 0)
		{
			PORTB |= (1<<2); //keep LED off
		}
		OVFCount++;//increment counter
		TIFR0 |= (1<<OCF0A);//reset flag
		TCNT0 = 0;//reset to 0
	}
	}
```

Modified TASK 2 from 2A CTC Switch:

```
#define F_CPU 16000000UL //Setting to 16Mhz
#include <avr/interrupt.h>
#include <avr/io.h>

int main(void)
{
	DDRB |= (1<<2); //set PB.2 as an output
	DDRB |= (1<<5); //set PB.5 as an output
	PORTB |= (1<<2); //set low
	PORTB |= (1<<5); //set low

	OCR0A = 0xFF; //set output compare register
	TCCR0A = (1<<WGM01); //set mode to CTC
	TCCR0B = (1<<CS02) | (1<<CS00); //set prescaler to 1024
	TIMSK0 |= (1<<OCIE0A); //enable overflow interrupt
	TCNT0 = 0;//set to 0

	sei();//enable interrupts

	while (1)
	{
		//main loop
	}
}

ISR(TIMER0_COMPA_vect)//timer 0 overflow interrupt
{
	uint8_t OVFCount = 0; //set OVFCOUNT to zero
	if(!(PINC & (1<<PINC2))) //if high enter statement
	{
		TCNT0 = 0; //reset TCNT0

		while(OVFCount < 77) //while less then 0
		{
			while ((TIFR0 & (1<<OCF0A)) == 0) //while not 255
			{
				PORTB &= ~(1<<2); //turn PB.2 on
			}
			OVFCount++; //increment counter
			TIFR0 |= (1<<OCF0A); //reset flag
			TCNT0 = 0; //reset TCNT0
		}

		TCNT0 = 0;//reset TCNT0
		OVFCount = 0; //set OVFcount to 0
	}
	else
	{
```

```
            PORTB |= (1<<2); //turn PB.2 off
        }
        }
```

5. SCHEMATICS



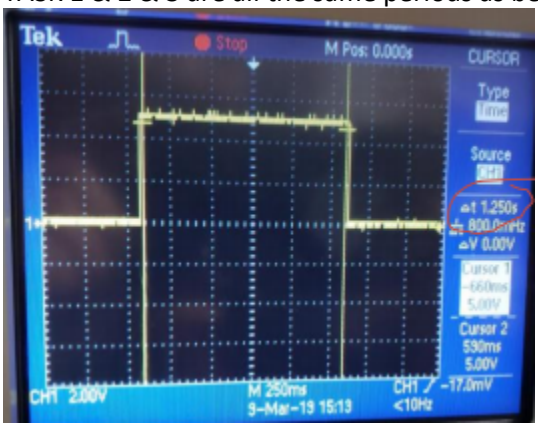
6. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

TASK 1 & 2 & 3 are all the same times as below for the Duty Cycle:

Cycle Counter	6960148
Frequency	16.000 MHz
Stop Watch	435,009.25 µs
Registers	

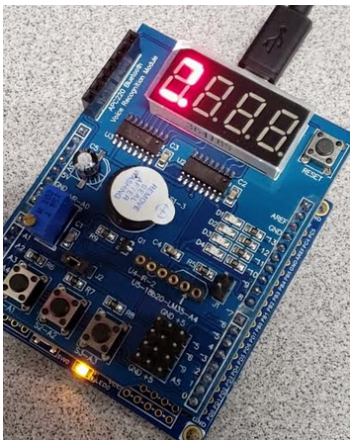
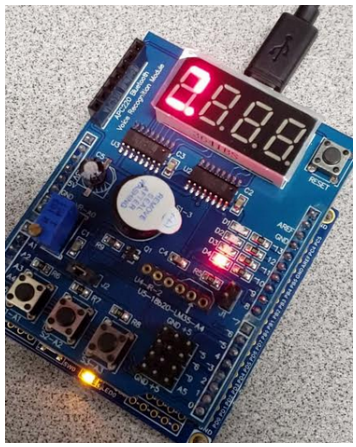
Frequency	16.000 MHz
Stop Watch	728,129.13 µs
Registers	

TASK 1 & 2 & 3 are all the same periods as below for the Switch:

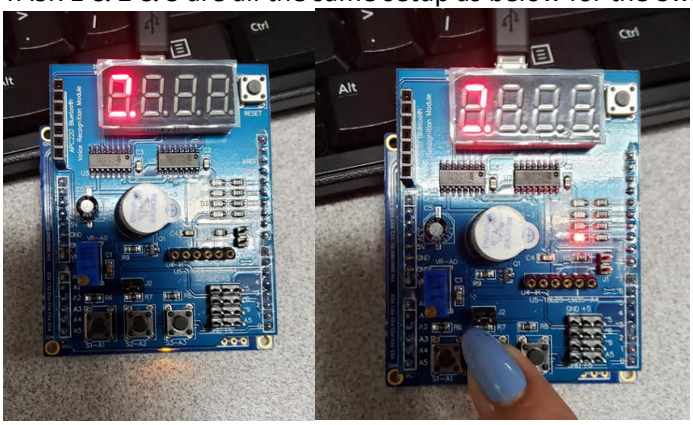


7. SCREENSHOT OF EACH DEMO (BOARD SETUP)

TASK 1 & 2 & 3 are all the same setup as below for the Duty Cycle:

TASK 1 & 2 & 3 are all the same setup as below for the Switch:



8. VIDEO LINKS OF EACH DEMO

TASK 1 Normal Mode with OVF:

Duty cycle:
https://www.youtube.com/watch?v=4d23EmAQTgU
Switch:
https://www.youtube.com/watch?v=C3juVMiIcCg

TASK 2 Normal Mode with Interrupt:

Duty cycle:
https://www.youtube.com/watch?v=B9reVX-JrlM
Switch:
https://www.youtube.com/watch?v=H4ghHeDbMBs

TASK 3 CTC Mode with Interrupt:

Duty cycle:
https://www.youtube.com/watch?v=5BmUMVxoN9s
Switch:
https://www.youtube.com/watch?v=trQ_UEqd3_c

9. GITHUB LINK OF THIS DA

https://github.com/HadidBuilds/hw_sub_da1

Student Academic Misconduct Policy

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

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

Itzel Becerril