CPE301 – SPRING 2019

Design Assignment 2C

Student Name: Geovanni Portillo
Student #: 8000603824
Student Email: portig1@unlv.nevada.edu
Primary Github address: https://github.com/portig1/submissions_E
Directory: submissions_E/DA/LAB2C/

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

Atmel Studio 7
ATmega328PB Xplained mini
Multi-function Shield

Figure 1-1. ATmega328P Xplained Mini Headers and Connectors

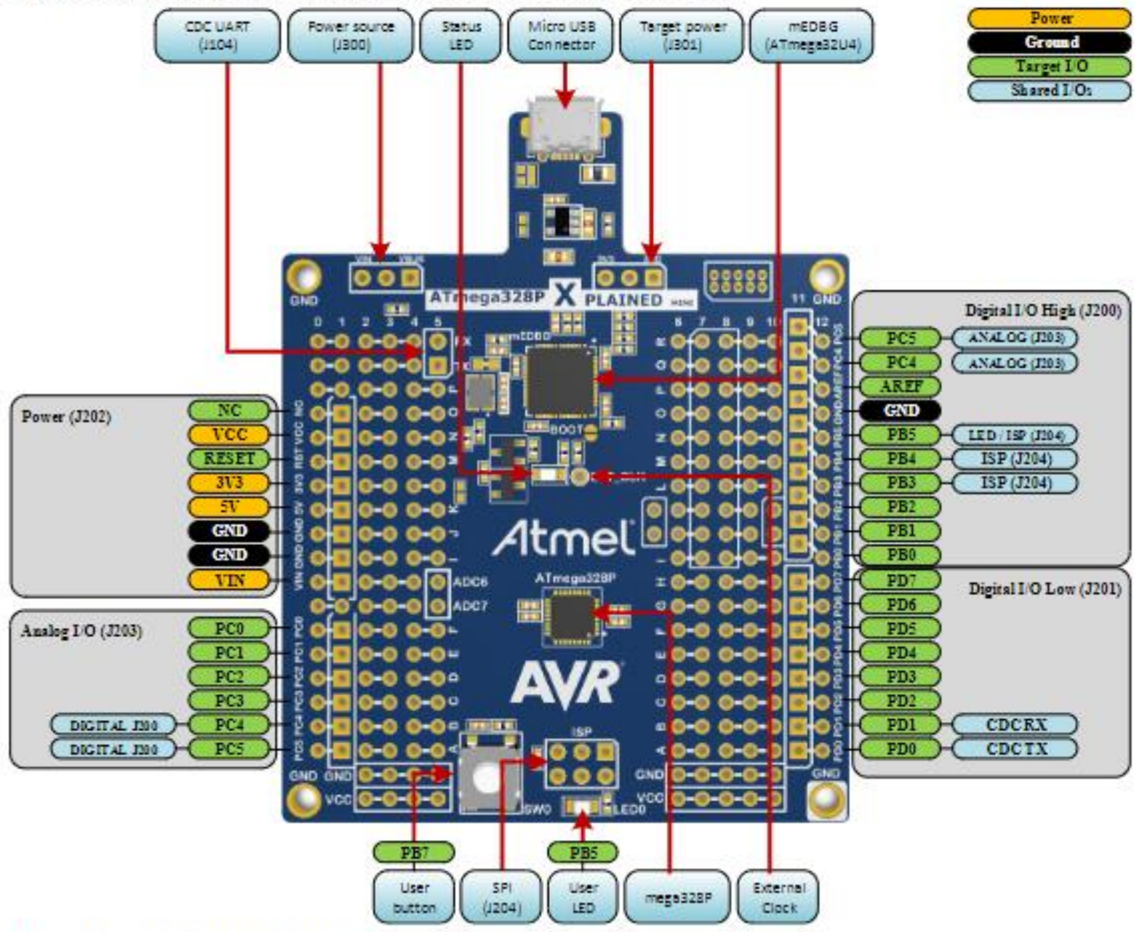
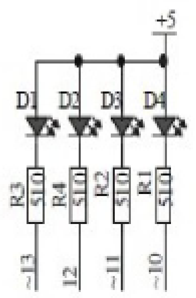
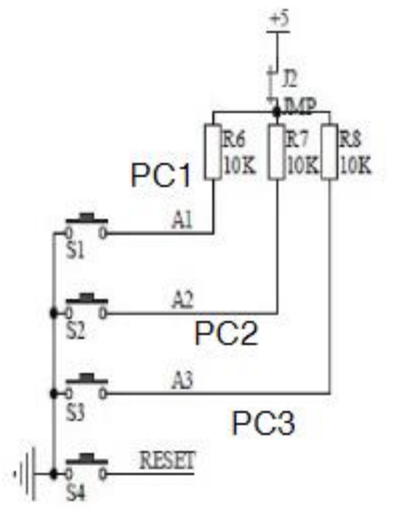


Table 1-1. Default Configurations



PB5,PB4,PB3,PB2

Multifunction Shield LED schematic



Multifunction Shield Switch Schematic

2. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/ PART1

```
#include<avr/io.h>
int main(void)
{

	TCCR0A = 0; // Normal Operation
	DDRB |= (1 << DDB2); //PB2 as output
	TCCR0B |= (1 << CS02) | (1 << CS00);
	// set prescaler to 1024 and start the timer
	int onCount, offCount;

	//For a period of 0.725s with a duty cycle of 60%, LED is on for 0.435s and off
for 0.29s
	while (1)
	{
		onCount = 0;
		offCount = 0;
		TCNT0=0x00; // start the timer

		//Using formula TCNT = (clock/prescaler*desired_time_in_seconds) - 1,
calculated that a delay of 0.435s and a prescaler of 1024 requires 6795.875 for TCNT.
TCNT0 can only count to 255 so 26 iterations are needed along with an additional count to
140 to achieve approximately 0.435ms
		PORTB = (0 << 2); //Turn on LED
		while(onCount < 26)
		{
			while ((TIFR0 & 0x01) == 0);
			TCNT0=0x00;
			TIFR0=0x01; // reset the overflow flag
			onCount++;
		};
		while(TCNT0 < 140);

		TCNT0=0x00; // restart the timer
		//identical calculations for 0.29s, we end up with 17 iterations and a
final count up to 179
		PORTB = (1 << 2);
		while(offCount < 17)
		{
			while ((TIFR0 & 0x01) == 0);
			TCNT0=0x00;
			TIFR0=0x01; // reset the overflow flag
			offCount++;
		};
		while(TCNT0 < 179);

	}
}
```

3. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/ PART2

```
#include<avr/io.h>
int main (void)
{
	/* set PORTB.2 for output*/
	DDRB |= (1 << 2);
	PORTB |= (1 << 2);

	/* set PORTC.2 for input*/
	DDRC &= (0 << 2);
	PORTC |= (1 << 2);   //enable pull-up
	/* A switch is connected to PORTC.2 and when pressed PINC.2 is set low. */

	TCCR0A = 0; // Normal Operation
	TCCR0B |= (1 << CS02) | (1 << CS00); // set prescaler to 1024 and start the timer

	int overflowCount = 0;

	//From the same calculations of Task1_Part1, for a 1.25s delay 76 overflows need
to occur and a count of 75 cycles.
	while (1) {
		if(!(PINC & (1 << PINC2)))
		{
			PORTB &= ~(1 << 2);
			TCNT0=0x00;   // reset counter
			TIFR0=0x01; // reset the overflow flag
			while(overflowCount < 76)
			{
				while ((TIFR0 & 0x01) == 0);
				TCNT0=0x00;
				TIFR0=0x01; // reset the overflow flag
				overflowCount++;
			};
			while(TCNT0 < 75);
			overflowCount = 0;
		}
		else
			PORTB |= (1 << 2);
	}
	return 0;
}
```

4. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 2/ PART1

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

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

//global variables
int overflowCounter = 0;

int main(void)
{
	DDRB |= (1 << DDB2); //PB2 as output
	TIMSK0 |= (1 << TOIE0);
	TCNT0 = 0; // initial value
	sei(); //enable interrupts
	int cycleStatus = 1; //cycle status will be for if duty cycle should be in on 
portion(1) or off portion(0)
	TCCR0B |= (1 << CS02) | (1 << CS00); // set prescaler to 1024 and start the timer

	//For a period of 0.725s, LED is on for 0.435s and then off for 0.29s. 
Calculations are reused from Task1_Part1
	while (1)
	{
		if((overflowCounter >= 26) & (cycleStatus == 1)) {
			while (TCNT0 < 140);
			PORTB = (1 << 2);
			cycleStatus = 0;//set so that the LED stays off until the the cycle 
goes through its "off" portion
		}
		else if((overflowCounter >= 44) & (cycleStatus == 0)) {
			while(TCNT0 < 64);
			PORTB = (0 << 2);
			overflowCounter = 0; //reset cycle counter
			cycleStatus = 1;     //set so that the LED stays on until the the 
cycle goes through its "on" portion
			TCNT0 = 0;
		}
	}
}

ISR (TIMER0_OVF_vect) // timer0 overflow interrupt
{
	TCNT0 = 0;
	overflowCounter++;
}
```

5. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 2/ PART2

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

int switchStatus = 0; //1 = pressed, 0 = not pressed
int overflowCounter = 0; //counts overflows
int main (void)
{
	/* set PORTB.2 for output*/
	DDRB |= (1 << 2);
	PORTB |= (1 << 2);

	/* set PORTC.2 for input*/
	DDRC &= (0 << 2);
	PORTC |= (1 << 2);   //enable pull-up
	/* A switch is connected to PORTC.2 and when pressed PINC.2 is set low. */

	TIMSK0 |= (1 << TOIE0);
	TCNT0 = 0; // initial value
	sei(); //enable interrupts

	TCCR0A = 0; // Normal Operation
	TCCR0B |= (1 << CS02) | (1 << CS00); // set prescaler to 1024 and start the timer


	//From the same calculations of Task1_Part2, for a 1.25s delay 76 overflows need
to occur and a count of 75 cycles.
	while (1) {
		if(!(PINC & (1 << PINC2)))
		{
			PORTB &= ~(1 << 2);
			switchStatus = 1;
		}
	}
	return 0;
}

ISR (TIMER0_OVF_vect) // timer0 overflow interrupt
{
	TCNT0 = 0;
	if(switchStatus == 1)
	{
		if(overflowCounter >= 76)
		{
			for(int i = 0; i < 75; i++);
			PORTB |= (1 << 2);
			switchStatus = 0;
			overflowCounter = 0;
		}
		overflowCounter++;
	}
}
```

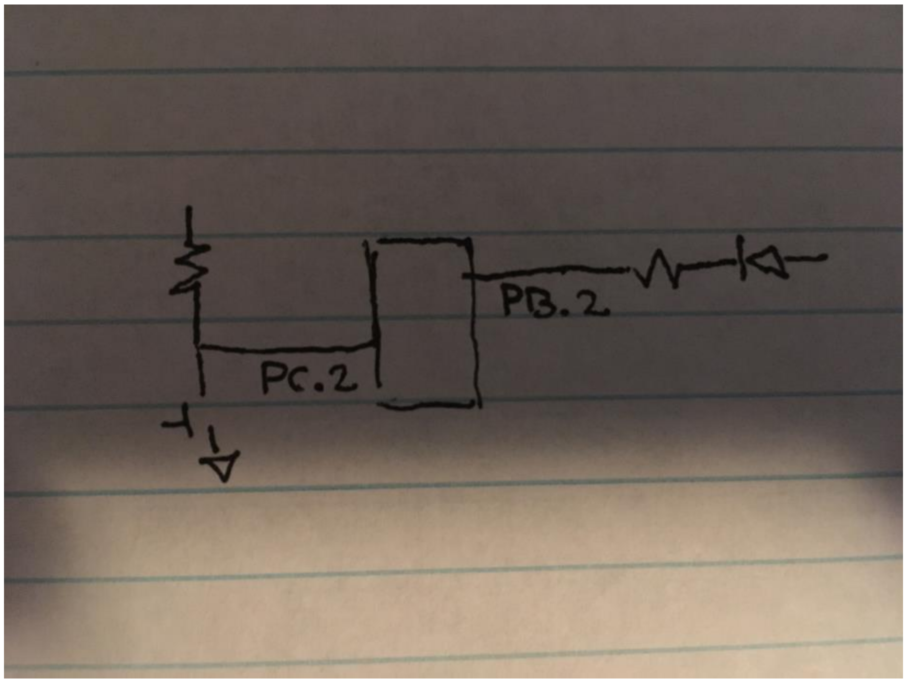
6. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 3/ PART1

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

//Global variables
int overflowCounter = 0;
int cycleStatus = 1; //cycle status will be for if duty cycle should be in on portion(1)
or off portion(0)

int main(void)
{
	DDRB |= (1 << DDB2); //PB2 as output

	OCR0A = 255; //Load Compare Reg value
	TCCR0A |= (1 << WGM01); // Set to CTC Mode
	TIMSK0 |= (1 << OCIE0A); //Set interrupt on compare match
	TCCR0B |= (1 << CS02) | (1 << CS00); // set prescaler to 1024 and starts Timer
	sei(); // enable interrupts
	while (1)
	{ // Main loop
	}
}
ISR (TIMER0_COMPA_vect)
{
	//For a period of 0.725s, LED is on for 0.435s and then off for 0.29s. 
Calculations are reused from Task1_Part1
	overflowCounter++;
	if((overflowCounter >= 26) & (cycleStatus == 1))
	{
		OCR0A = 140;
		if(overflowCounter >= 27)
		{
			OCR0A = 255;
			cycleStatus = 0; //set so that the LED stays off until the the cycle 
goes through its "off" portion
			PORTB = (1 << 2);
		}
	}
	else if((overflowCounter >= 44) & (cycleStatus == 0))
	{
		OCR0A = 64;
		if(overflowCounter >= 45)
		{
			OCR0A = 255;
			cycleStatus = 1; //set so that the LED stays on until the the cycle 
goes through its "on" portion
			PORTB = (0 << 2);
			overflowCounter = 0; //reset counter to cycle the waveform
		}
	}
}
```

7. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 3/ PART2

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

int switchStatus = 0; //1 = pressed, 0 = not pressed
int overflowCounter = 0; //counts overflows
int main (void)
{
	/* set PORTB.2 for output*/
	DDRB |= (1 << 2);
	PORTB |= (1 << 2);

	/* set PORTC.2 for input*/
	DDRC &= (0 << 2);
	PORTC |= (1 << 2);   //enable pull-up
	/* A switch is connected to PORTC.2 and when pressed PINC.2 is set low. */

	OCR0A = 255; //Load Compare Reg value

	TIMSK0 |= (1 << OCIE0A);
	sei(); //enable interrupts

	TCCR0A |= (1 << WGM01); // Set to CTC Mode
	TCCR0B |= (1 << CS02) | (1 << CS00); // set prescaler to 1024 and start the timer
	//From the same calculations of Task1_Part2, for a 1.25s delay 76 overflows need
to occur and a count of 75 cycles.
	while (1) {
		if(!(PINC & (1 << PINC2)))
		{
			PORTB &= ~(1 << 2);
			switchStatus = 1;
		}
	}
	return 0;
}

ISR (TIMER0_COMPA_vect) // timer0 compare interrupt
{
	if(switchStatus == 1)
	{
		if(overflowCounter >= 76)
		{
			OCR0A = 75;
			if(overflowCounter >= 77)
			{
				PORTB |= (1 << 2);
				switchStatus = 0;
				overflowCounter = 0;
				OCR0A = 255;
			}
		}
		overflowCounter++;
	}
}
```

8. SCHEMATICS

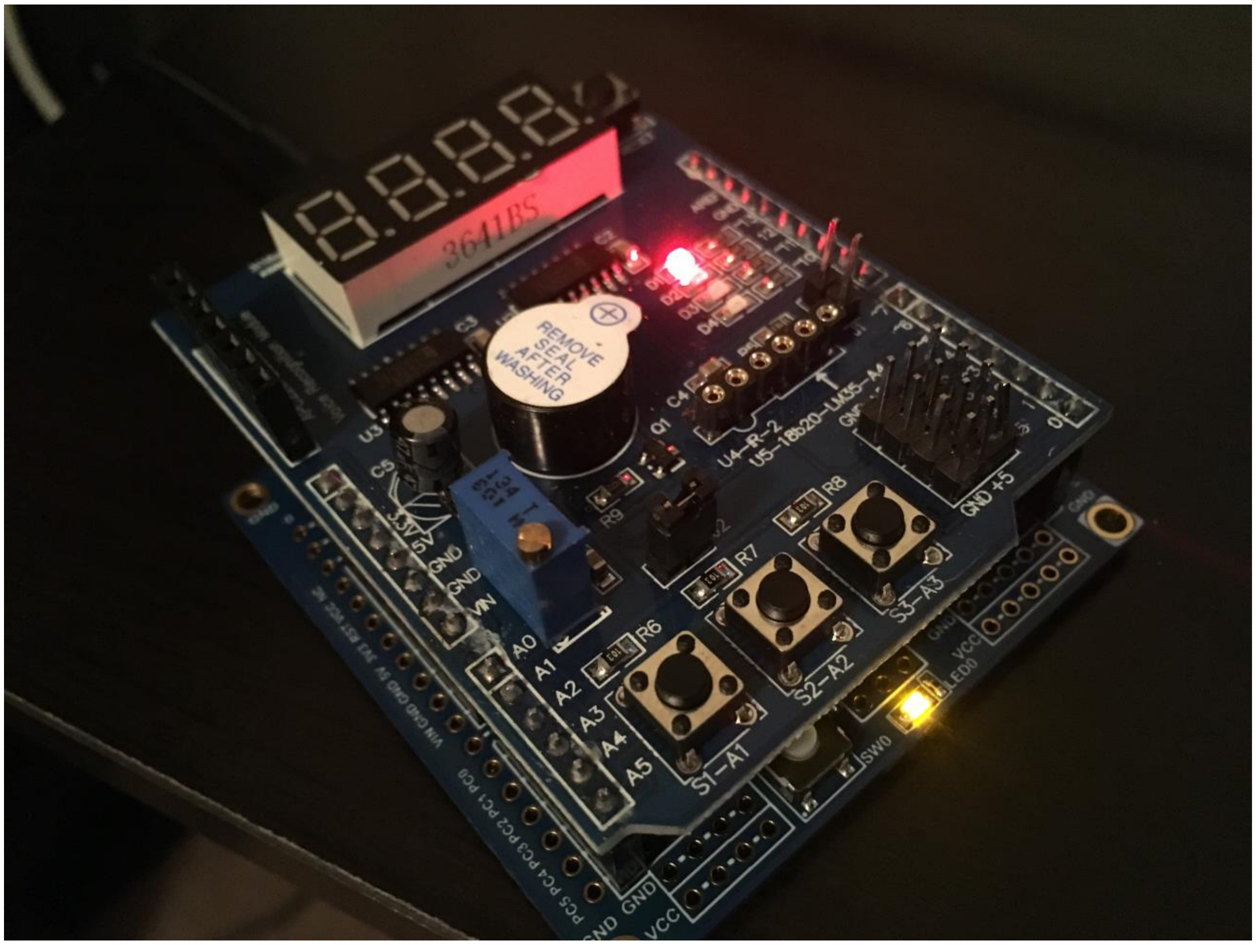


Schematic showing connections for PORTB.2 and PORTC.2

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

Not asked for in Design Assignment 2A instructions

10. SCREENSHOT OF EACH DEMO (BOARD SETUP)


Board setup for Tasks 1 and 2 (Assembly and C)

11. VIDEO LINKS OF EACH DEMO

https://youtu.be/qKaQFMzMypw

12. GITHUB LINK OF THIS DA

https://github.com/portig1/submissions_E/tree/master/DA/LAB2C

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

"This assignment submission is my own, original work".
Geovanni Portillo