CPE301 – SPRING 2019

Design Assignment 2c

Student Name: Do Le

Student #: 2001183621

Student Email: led2@unlv.nevada.edu

Primary Github address: <https://github.com/DoVietLe>

Directory: <https://github.com/DoVietLe/assignments>

Design Assignment 2C: The goal of the assignment is use GPIO and delays using Timers and Interrupts:

1. Implement Design Assignment 2A using Timer 0 – normal mode. Count OVF occurrence if needed. Do not use interrupts.

2. Implement Design Assignment 2A using TIMER0\_OVF\_vect interrupt mechanism in normal mode.

3. Implement Design Assignment 2A using TIMER0\_COMPA\_vect interrupt mechanism in CTC mode.

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

Components

• ATmega328PB Xplained Mini Board

• Arduino Multifunction Shield

* LED
* Button

• Atmel Studio 7.0

* Compiler
* Debugger

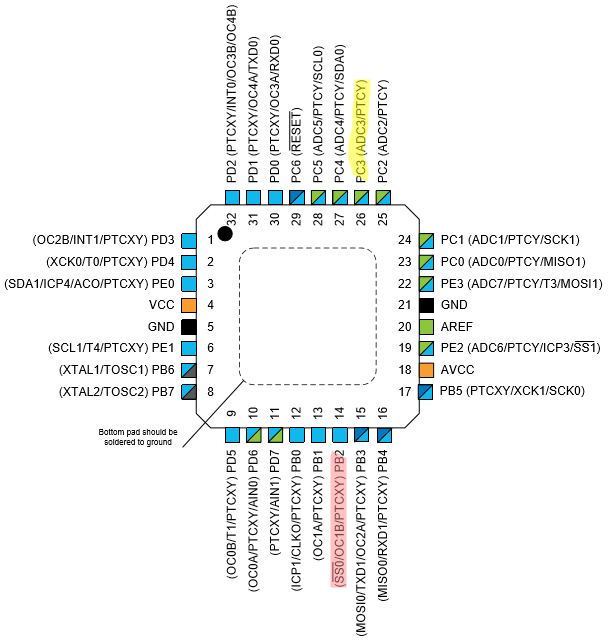
Block Diagram

Figure 1: Pins used

PORTC3: Used as input to read data from a switch.

PORTB2: Used as output to toggle the LED.

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

Task 1 Code

#define cycles\_412500us 6446

#define cycles\_337500us 5274

#define cycles\_2s 31251

#include <avr/io.h>

int main(void)

{

DDRC = 0x00; // Sets PORTC3 as input port.

PORTC = (0x01<<PORTC3); // Activates the pull up resistor on PORTC3.

DDRB = (0x01<<PORTB2 | 0x01<<PORTB3); // Sets PORTB2 as an output port.

PORTB = (0x01<<PORTB2); // Output 1 on PORTB2 (Turns off LED since

// LED is connected to Vcc).

while (1)

{

// Waits for PINC3 to go low (the button is active high).

if ((PINC & (0x01<<PINC3)) == 0x00) {

PORTB &= ~(0x01<<PORTB2); // Turns on the LED by allowing current to sink.

delay(cycles\_2s);

PORTB |= (0x01<<PORTB2); // Turns off the LED.

}

// Generates the waveform.

PORTB |= (0x01<<PORTB3); // Turns PORTB3 high for 55% duty cycle.

delay(cycles\_412500us);

PORTB = ~(0x01<<PORTB3); // Turns PORTB3 low for 45% of the period.

delay(cycles\_337500us);

}

}

// Runs delay given an amount of cycles (unsigned int .

void delay(unsigned int cycles) {

// Sets up timer 0.

TCCR0A = 0x00; // Sets timer0 to normal mode.

TCCR0B = 0x05; // Sets timer0 prescaler to 1024.

// Keeps delaying until the amount of cycles needed is less than 255.

while (cycles >= 255) {

cycles -= 255;

TCNT0 = 0x00;

while (1)

if (TCNT0 == 255)

break;

}

// Delays the last few cycles.

TCNT0 = 0x00;

while (1)

if (TCNT0 == cycles)

break;

}

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

Task 2 Code

#define cycles\_412500us 6445

#define cycles\_337500us 5273

#define cycles\_2s 31250

#include <avr/interrupt.h>

#include <avr/io.h>

volatile unsigned int passes = 0;

int main(void)

{

// Sets PORB3 as an output.

DDRB = (0x01<<PORTB2 | 0x01<<PORTB3);

PORTB |= (0x01<<PORTB2); // Makes sure LED is off.

// Sets timer overflow interrupt.

TIFR0 = 0x01; // Clears the TOV0 flag.

TIMSK0 = 0x01; // Enabled TOIE0 (overflow interrupt).

sei(); // Enables global interrupt.

// Sets timer.

TCCR0A = 0x00; // Sets timer 0 to normal mode.

TCCR0B = 0x05; // Sets pre-scaler to 1024.

TCNT0 = 0x00; // Starts the counter at 0.

// Generates the wave.

while (1) {

// Waits for PINC3 to go low (the button is active high).

if ((PINC & (0x01<<PINC3)) == 0x00) {

PORTB &= ~(0x01<<PORTB2); // Turns on the LED by allowing current to sink.

delay(cycles\_2s);

PORTB |= (0x01<<PORTB2); // Turns off the LED.

}

PORTB |= (0x01<<PORTB3); // Sets PORTB3 to high.

delay(cycles\_412500us); // Waits the 55% duty cycle.

PORTB &= ~(0x01<<PORTB3); // Clears PORTB3.

delay(cycles\_337500us); // Waits the remaining 45% of the period.

}

}

// Creates a delay given a certain amount of cycles.

void delay(unsigned int cycles) {

// Calculates passes and remaining cycles needed.

unsigned int rPasses = cycles/256;

unsigned int rCycles = cycles%256;

// Resets timer.

TCNT0 = 0x00;

passes = 0;

// Do nothing until the passes reaches the required passes.

while (passes < rPasses) {

}

// Sets the timer to overflow after remaining cycles needed passes.

TCNT0 = (255-rCycles);

passes = 0;

// Waits for the final overflow.

while (passes < 1) {

}

}

ISR(TIMER0\_OVF\_vect) {

TIFR0 = 0x01; // Clears the timer0 interrupt flag.

passes++;

}

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

Task 3 Code

#define cycles\_412500us 6445

#define cycles\_337500us 5273

#define cycles\_2s 31250

#include <avr/interrupt.h>

#include <avr/io.h>

volatile unsigned int passes = 0;

int main(void)

{

// Sets PORB3 as an output.

DDRB = (0x01<<PORTB2 | 0x01<<PORTB3);

PORTB |= (0x01<<PORTB2); // Makes sure LED is off.

// Sets timer overflow interrupt.

TIFR0 = 0x02; // Clears the OCF0A flag.

TIMSK0 = 0x02; // Enabled OCIE0A (compare A interrupt).

OCR0A = 128; // Sets compare A to 128.

sei(); // Enables global interrupt.

// Sets timer.

TCCR0A = 0b00000010; // Sets timer 0 to CTC mode.

TCCR0B = 0b00000101; // Sets pre-scaler to 1024.

TCNT0 = 0x00; // Starts the counter at 0.

// Generates the wave.

while (1) {

// Waits for PINC3 to go low (the button is active high).

if ((PINC & (0x01<<PINC3)) == 0x00) {

PORTB &= ~(0x01<<PORTB2); // Turns on the LED by allowing current to sink.

delay(cycles\_2s);

PORTB |= (0x01<<PORTB2); // Turns off the LED.

}

PORTB |= (0x01<<PORTB3); // Sets PORTB3 to high.

delay(cycles\_412500us); // Waits the 55% duty cycle.

PORTB &= ~(0x01<<PORTB3); // Clears PORTB3.

delay(cycles\_337500us); // Waits the remaining 45% of the period.

}

}

// Creates a delay given a certain amount of cycles.

void delay(unsigned int cycles) {

// Calculates passes and remaining cycles needed.

unsigned int rPasses = cycles/129;

unsigned int rCycles = cycles%129;

// Resets timer.

TCNT0 = 0x00;

passes = 0;

// Do nothing until the passes reaches the required passes.

while (passes < rPasses) {

}

// Sets the timer to overflow after remaining cycles needed passes.

TCNT0 = (128-rCycles);

passes = 0;

// Waits for the final overflow.

while (passes < 1) {

}

}

ISR(TIMER0\_COMPA\_vect) {

TIFR0 = 0x02; // Clears the interrupt flag.

passes++;

}

1. **SCHEMATICS**

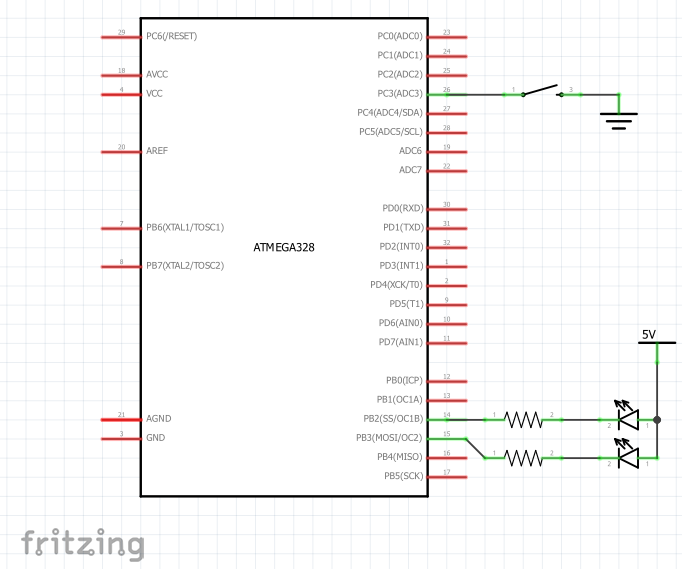


Figure 2: Schematic

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

Task 1

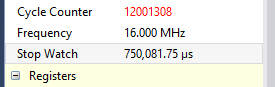


Figure : Period of wave

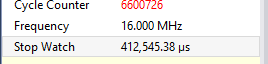


Figure : 55% duty cycle

Task 2

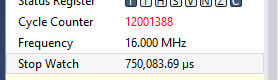


Figure : Period of wave

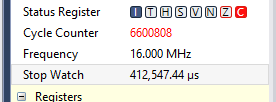


Figure : 55% duty cycle

Task 3

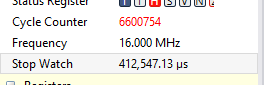
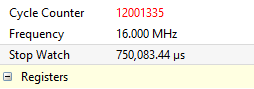


Figure : Period of wave

Figure : 55% duty cycle

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

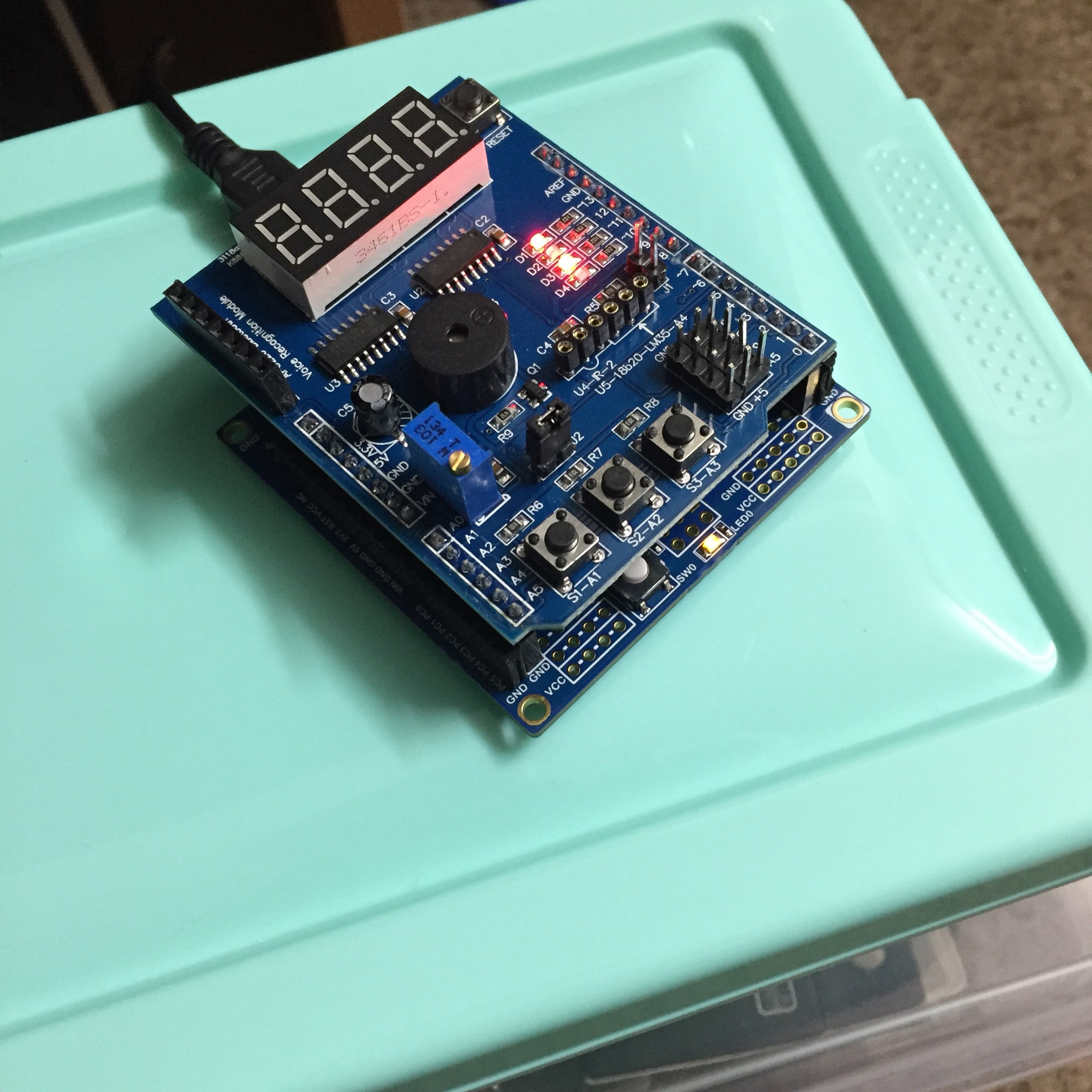


Figure : Board setup (same setup for all 3 tasks)

1. **VIDEO LINKS OF EACH DEMO**

Task 1

https://youtu.be/CVEXZdJTvlk

Task 2

https://youtu.be/\_gfLBhE8ZXk

Task 3

https://youtu.be/HsFavpONkhs

1. **GITHUB LINK OF THIS DA**

https://github.com/DoVietLe/assignments/tree/master/ESD301/LAB02c

**Student Academic Misconduct Policy**

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

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

Do Viet Le