CPE301 – SPRING 2019

Design Assignment 2C

Student Name: Tanner Tindall

Student #: 8000733733

Student Email: tindat1@unlv.nevada.edu

Primary Github address: <https://github.com/TannerTindall51>

Directory: <https://github.com/TannerTindall51/tindalltannerm_submission/tree/master/Design_Assignments/DA2C>

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.

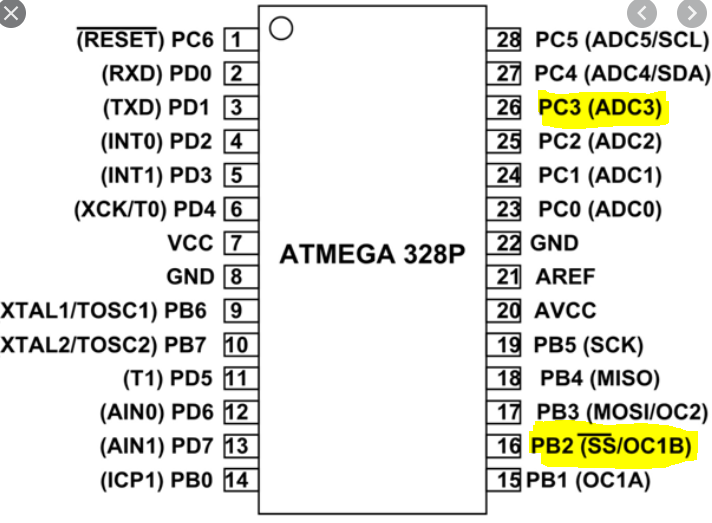
2. Implement Design Assignment 2A using TIMER0\_OVF\_vect interrupt mechanism in normal mode.

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

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

* Atmel Studio 7.0 (Assembler, Simulator, & Debugger)
* Atmega328PB-Xmini
* PC Multi-Function Shield
* Logic Analyzer
* Switches
* LEDs

PB2 = LED

PC3 = Interrupt

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

|  |
| --- |
|  |

Task 1:

//

//Design Assignment 2C - Task 1

//Tanner Tindall

//

#define *F\_CPU* 16000000UL

#include <avr/io.h>

int main(void)

{

DDRB = 0xFF; //sets DDRB to OUTPUT

DDRC = 0x00; //sets DDRC to INPUT

PORTB = 0xFF; //sets all of PORTB logic level HIGH

TCNT0 = 0x00; //initializes the 8 bit timer0 register to begin at 0

TCCR0A = 0x00; //sets the timer to "Normal" Mode by WGM00

TCCR0B |= (1 << CS02) | (1 << CS00); //sets prescaler value to 1024, refer to TCCROB truthTable

int overFlow = 0; //initializes variable to be used as counter for overflow occurrences

while (1) //primary (infinite) loop for the program

{

TCNT0 = 0x00; //resetting the timer to 0 when the code loops back up

overFlow = 0; //resetting the counter to 0 when the code loops back up

while(overFlow < 25) //55% DutyCycle Loop (.75s = 45 therefore 55% \* 45 = 25)

{

if(TCNT0 ==0xFF) //once the timer reaches the max value...

{

TCNT0 = 0x00; //...reset it and...

overFlow++; //...add to the overflow counter and begin again

}

else if(!(PINC & (1 << 3))) //checks to see if SW3 is pressed

{

PORTB |= (1<<3); //turns off LED at PB3

PORTB &= ~(1<<2); //turns on LED PB2

overFlow = 0; //reset overflow counter

TCNT0 = 0x00; //reset timer

while(overFlow < 120) //2s Delay Loop (1 sec = 60 therefore 2 \* 60 = 120)

{

if(TCNT0 ==0xFF) //once the timer reaches the max value...

{

TCNT0 = 0x00; //...reset it and...

overFlow++; //...add to the overflow counter and begin again

}

}

PORTB |= (1<<2); //turns off LED at PB2

}

}

while (overFlow < 20) //55% DutyCycle Loop (.75s = 45 therefore 45% \* 45 = 20)

{

if(TCNT0 ==0xFF) //once the timer reaches the max value...

{

TCNT0 = 0x00; //...reset it and...

overFlow++; //...add to the overflow counter and begin again

}

else if(!(PINC & (1 << 3)))

{

PORTB |= (1 << 3); //turn off LED at PB3

PORTB &= ~(1 << 2); //turn on LED at PB2

overFlow = 0; //reset overflow counter

TCNT0 = 0x00; //reset timer

while(overFlow < 120) //2s Delay Loop (1 sec = 60 therefore 2 \* 60 = 120)

{

if(TCNT0 == 0xFF) //once the timer reaches the max value...

{

TCNT0 = 0x00; //...reset it and...

overFlow++; //...add to the overflow counter and begin again

}

}

PORTB |= (1<<2); //turn off LED at PB2

}

}

PORTB ^= (1<<3); //inverts LED for each execution of the loop to produce strobe

}

return 0;

}

Task 2:

//

//Design Assignment 2C - Task 2

//Tanner Tindall

//

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

volatile int overFlow\_Count = 0; //initializes variable to be used as counter for overflow occurrences

volatile int overFlow\_Max = 21; //45% Duty Cycle @337.5ms (((16MHz/1024) \* .3375)/256) = 21

int main(void)

{

DDRB = 0xFF; //sets DDRB to OUTPUT

DDRC = 0x00; //sets DDRC to INPUT

PORTB = 0xFF; //sets all of PORTB logic level HIGH

TCNT0 = 0; //initializes the 8 bit timer0 register to begin at 0

TCCR0A = 0x00; //sets the timer to "Normal" Mode by WGM00

TCCR0B |= (1 << CS02) | (1 << CS00);//sets prescaler value to 1024, refer to TCCROB truthTable

TIMSK0 = (1<<TOIE0); //enable timer0 overflow interrupt

sei(); //turn on interrupts

while (1) //infinite loop which is interrupted by the function below

{

}

}

ISR (TIMER0\_OVF\_vect)

{

overFlow\_Count++; //increments the counter upon interrupt

if (overFlow\_Count == overFlow\_Max) //comparison between counter and counterMax values

{

PORTB ^= (1<<3); //toggle LED at PB3

PORTB |=(1<<2); //turn off LED at PB2

if(overFlow\_Max == 21) //if 45% DutyCycle, then begin this statement

{

overFlow\_Max = 25; //change max counter to 55% Duty Cycle @412.5ms (((16MHz/1024) \* .4125)/256) = 25

if(!(PINC & (1<<3))) //checks to see if SW3 is pressed

{

PORTB &= ~(1<<2); //turns on LED at PB2

overFlow\_Max = 120; //2s Delay Loop (1 sec = 60 therefore 2 \* 60 = 120)

PORTB |= (1<<3); //turns off LED at PB3

}

}

else //if 55% DutyCycle, then begin this statement

{

overFlow\_Max = 21; //reset back to 45% DutyCycle for next iteration

if(!(PINC & (1<<3))) //checks to see if SW3 is pressed

{

PORTB &= ~(1<<2); //turns on LED at PB2

overFlow\_Max = 120; //2s Delay Loop (1 sec = 60 therefore 2 \* 60 = 120)

PORTB |= (1<<3); //turns off LED at PB3

}

}

overFlow\_Count = 0; //resetting counter

}

TCNT0 = 0; //resetting timer

}

Task 3:

//

//Design Assignment 2C - Task 3

//Tanner Tindall

//

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

volatile int overFlow\_Count = 0; //initializes variable to be used as counter for overflow occurrences

volatile int overFlow\_Max = 21; //45% Duty Cycle @337.5ms (((16MHz/1024) \* .3375)/256) = 21

int main(void)

{

DDRB = 0xFF; //sets DDRB to OUTPUT

DDRC = 0x00; //sets DDRC to INPUT

PORTB = 0xFF; //sets all of PORTB logic level HIGH

TCNT0 = 0; //initializes the 8 bit timer0 register to begin at 0

TCCR0A = 0x02; //sets the timer to "CTC" Mode by setting WGM01 high

TCCR0B = (1 << CS02) | (1 << CS00);//sets prescaler value to 1024, refer to TCCROB truthTable

TIMSK0 = (1<<OCIE1A); //enable timer0 overflow interrupt

OCR0A = 255;

sei(); //turn on interrupts

while (1) //infinite loop which is interrupted by the function below

{

}

}

ISR (TIMER0\_COMPA\_vect)

{

overFlow\_Count++; //increments the counter upon interrupt

if (overFlow\_Count == overFlow\_Max) //comparison between counter and counterMax values

{

PORTB ^= (1<<3); //toggle LED at PB3

PORTB |=(1<<2); //turn off LED at PB2

if(overFlow\_Max == 21) //if 45% DutyCycle, then begin this statement

{

overFlow\_Max = 25; //change max counter to 55% Duty Cycle @412.5ms (((16MHz/1024) \* .4125)/256) = 25

if(!(PINC & (1<<3))) //checks to see if SW3 is pressed

{

PORTB &= ~(1<<2); //turns on LED at PB2

overFlow\_Max = 120; //2s Delay Loop (1 sec = 60 therefore 2 \* 60 = 120)

PORTB |= (1<<3); //turns off LED at PB3

}

}

else //if 55% DutyCycle, then begin this statement

{

overFlow\_Max = 21; //reset back to 45% DutyCycle for next iteration

if(!(PINC & (1<<3))) //checks to see if SW3 is pressed

{

PORTB &= ~(1<<2); //turns on LED at PB2

overFlow\_Max = 120; //2s Delay Loop (1 sec = 60 therefore 2 \* 60 = 120)

PORTB |= (1<<3); //turns off LED at PB3

}

}

overFlow\_Count = 0; //resetting counter

}

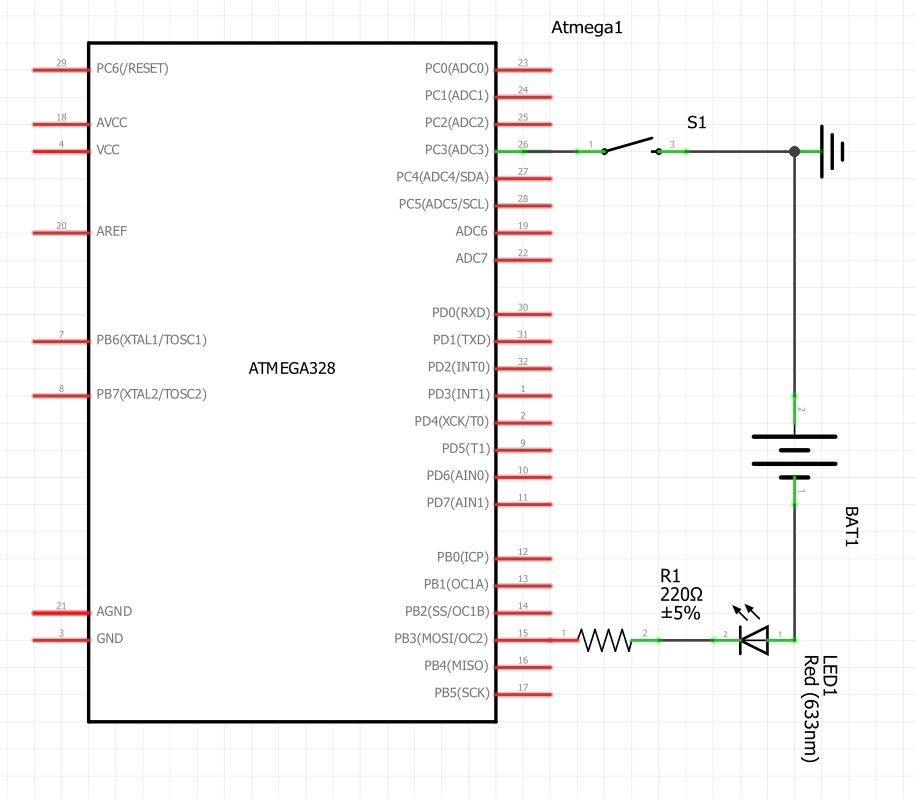
TCNT0 = 0; //resetting timer

}

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

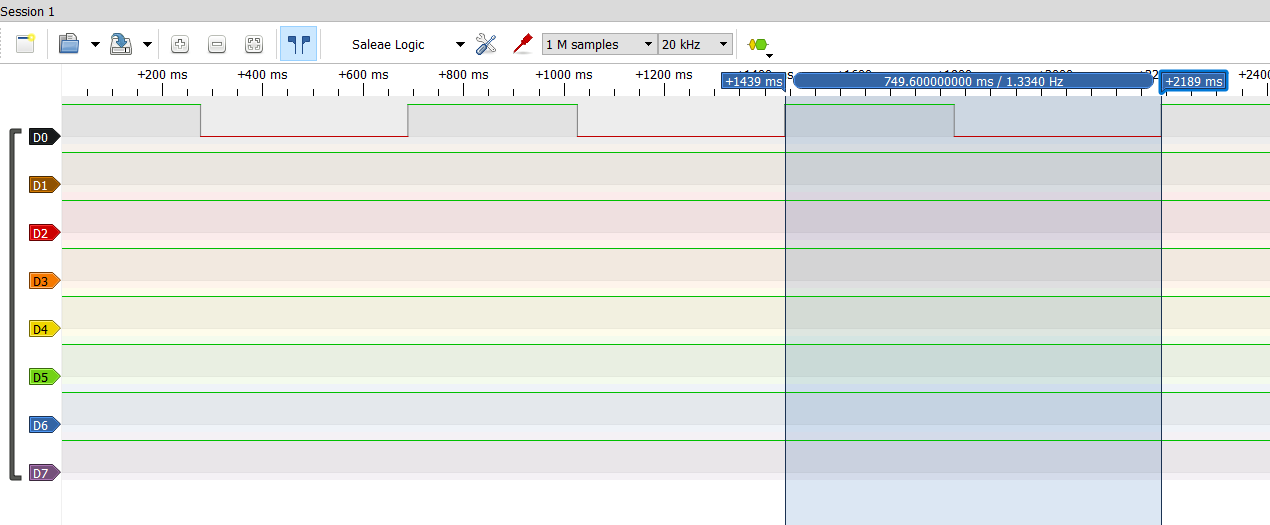
N/A

1. **SCHEMATICS**



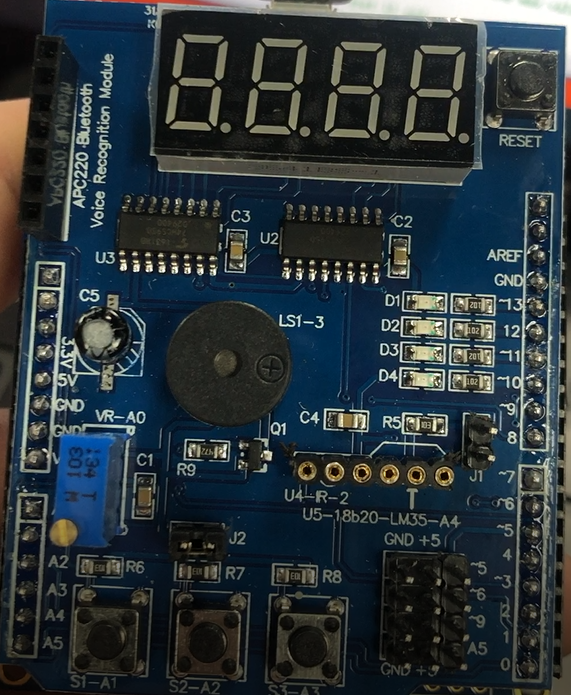
1. **SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)**

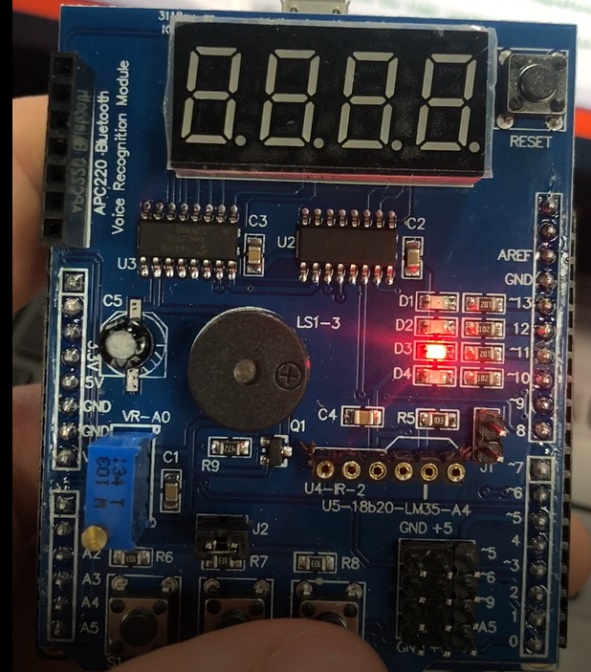
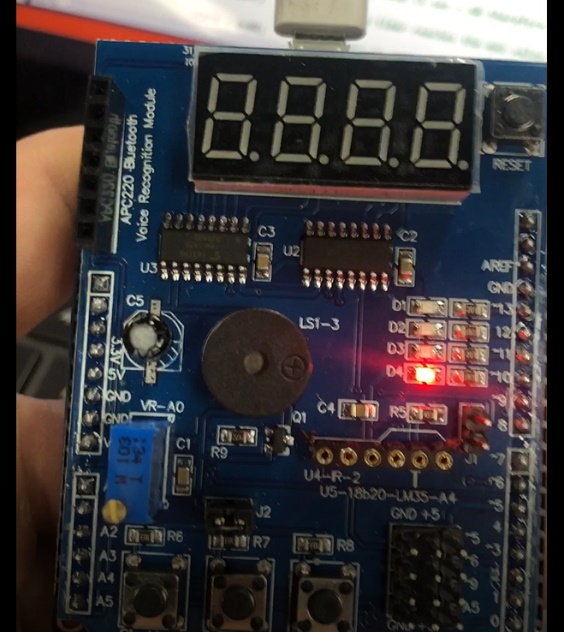
The image below displays the output waveform of the logic analyzer displaying the length (in time) of one period (750ms).



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

The images below show the proper LED in both the “ON” and “OFF” phase of the duty cycle as well as the interrupt LED activated respectively from left to right.





1. **VIDEO LINKS OF EACH DEMO**

Task 1: <https://youtu.be/FfGhHwiPopQ>

Task 2: <https://youtu.be/YfQ9ZGTPBqw>

Task 3: <https://youtu.be/axcj6RoTxxM>

1. **GITHUB LINK OF THIS DA**

<https://github.com/TannerTindall51/tindalltannerm_submission/tree/master/Design_Assignments/DA2C>

**Student Academic Misconduct Policy**

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

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

Tanner Tindall