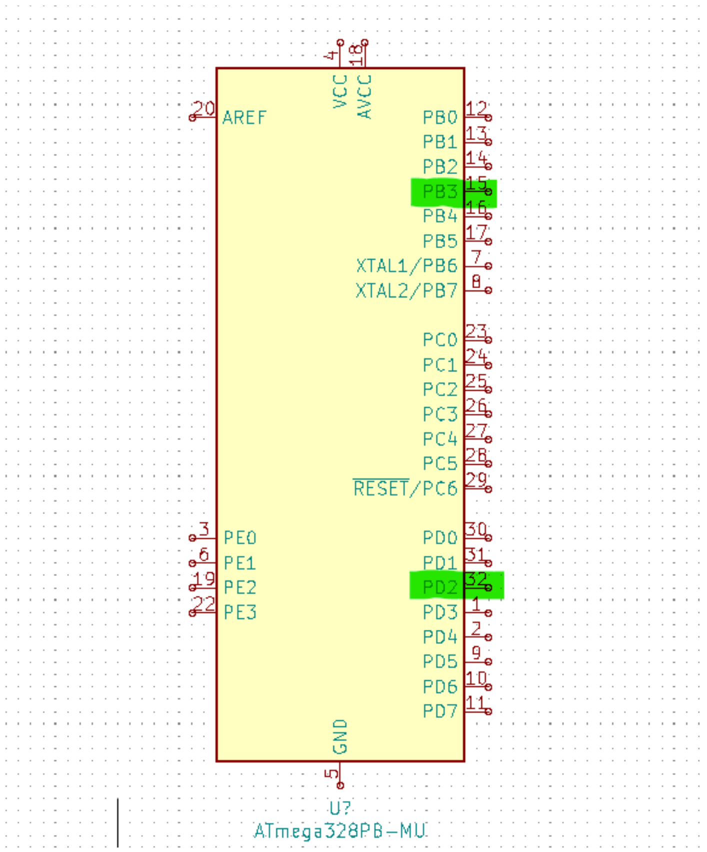
CPE301 – SPRING 2021

Design Assignment 2B

Student Name: Brian Wolak
Student #: 2000509437
Student Email: wolak@unlv.nevada.edu
Primary Github address: https://github.com/brianwolak/submission_da.git
Directory: [submission_da/DA_2B at main · brianwolak/submission_da (github.com)](https://github.com/brianwolak/submission_da)

Task 1:

This design assignment will use a similar format to the previous 2 assignment 2A where we create a 1 second period LED flashing LED using a 75% duty cycle with reverse logic. In this version we will be using an interrupt on pin PD2 to sense a logic change and initiate a 2 second delay. Once the delay is completed the device will return to the normal 1 second period with 75% duty cycle. This program will be written in C and AVR assembly using an ATMEGA328pb microcontroller.



ATMEGA328pb Ports Used in Design Assignment 2B

Video Link:
https://youtu.be/mUgpAFkUDNM

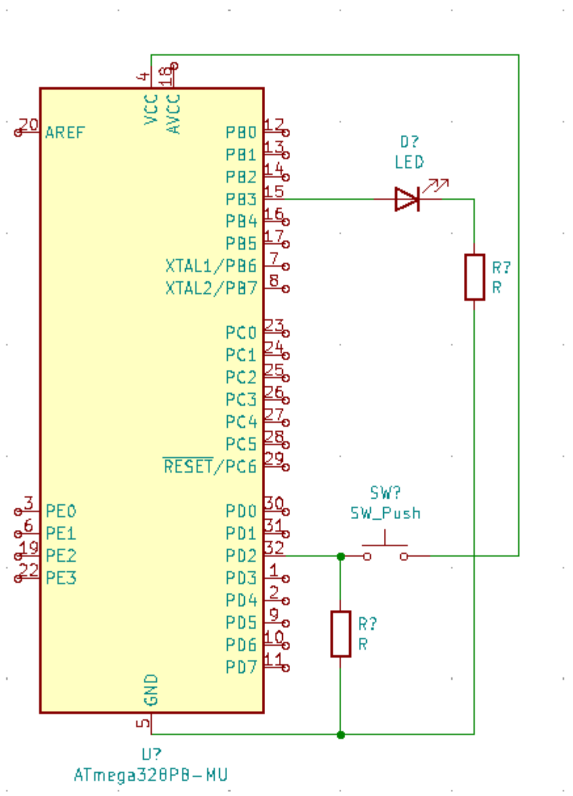
Assembly Code:

```
.org 0
rjmp INITIALIZE             ;jump to initialize

.org 0x02                   ;setting up the interrupt
rjmp EX0_ISR                ;go to the interrupt routine

INITIALIZE:
sei                         ;enable interrupts
ldi r16, 0x00               ;load r16 with 0 value
ldi r17, 0x08               ;load r17 with 0x08 value
ldi r18, 0xff               ;load r18 with 0xff value
ldi r19, 1                  ;load r19 with 1 value
ldi r20, 5                  ;load r20 with 5 for prescale value
ldi r21, 0xC6               ;load r21 for 75% T1 low value
ldi r26, 0x2D               ;load r26 for 75% T1 high value
ldi r27, 0x3D               ;load r27 for 1 sec T1 high value
sts TCCR1A, r16             ;timer 1 setup from r16
sts TCCR1B, r20             ;prescale setup of 1024
out DDRB, r18               ;setting DDRB at output
ldi r28, 0x04               ;load r28 with 0x04 value
out PORTD, r28              ;set pin 3 as output on PORTD
ldi r28, 0x02               ;load r28 with 0x02 value
sts 0x69, r28               ;save r28 register to EICRA
ldi r28, 0x01               ;load re28 with 0x01
sts 0x3D, r28               ;save r28 register to SPL

BEGIN:
out PORTB, r16              ;set DDRB output to 0
sts TCNT1H, r16             ;set timer1 high bits to zero
sts TCNT1L, r16             ;set timer1 low bits to zero

DELAY:
lds r22, TCNT1L             ;T1L 75%
lds r23, TCNT1L             ;T1L 100%
lds r24, TCNT1H             ;T1H 75%
lds r25, TCNT1H             ;T1H 100%
cp r22, r21                 ;compare r22 and r16
breq SUB1                   ;go to SUB if equal
cp r23, r17                 ;compare r23 and r17
breq SUB2                   ;go to SUB2 if equal
rjmp DELAY                  ;restart delay

SUB1:
cp r24, r26                 ;compare 75% high timer values
breq TOGGLE                 ;to TOGGLE if same or higher
rjmp DELAY                  ;back to delay

SUB2:
cp r25, r27                 ;compare T1 full second time values
breq BEGIN                  ;if less than jump to DELAY
rjmp DELAY                  ;back to BEGIN
```

```
TOGGLE:
out PORTB, r17              ;output 1 to PORTB
rjmp DELAY                  ;back to delay

EX0_ISR:
;this will be the interrupt routine that will delay the state for 2 seconds
sts TCNT1H, r16             ;set timer1 high bits to zero
sts TCNT1L, r16             ;set timer1 low bits to zero

START2:
lds r22, TCNT1L             ;load low counter value to r22
lds r23, TCNT1H             ;load high counter value to r23
cpi r22, 0x11               ;compare low count with 0x11
breq SUB3                   ;branch to SUB3 if equal
rjmp START2                 ;otherwise jump back to START2

SUB3:
cpi r23, 0x7A               ;compare high count with 0x7A
breq RETURN                 ;branch to RETURN if equal
rjmp START2                 ;jump to START2

RETURN:
sts 0x3C, r28               ;clear flag using r28 value
RETI                        ;exit interput routine and back to main program
```

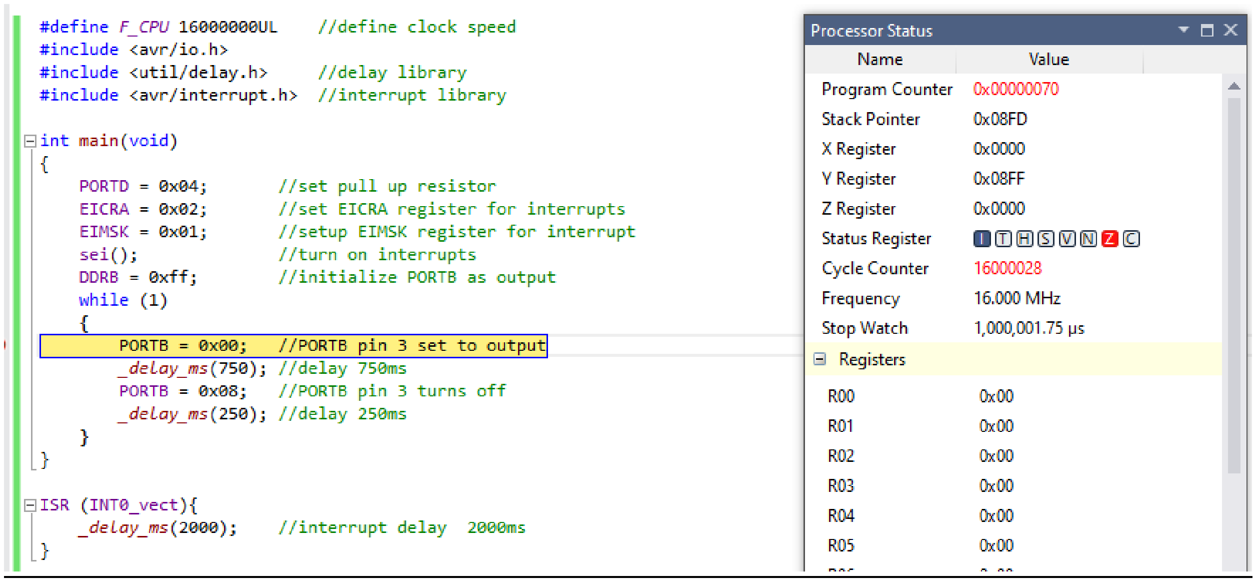
C Code:

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

int main(void)
{
	PORTD = 0x04;           //set pull up resistor
	EICRA = 0x02;           //set EICRA register for interrupts
	EIMSK = 0x01;           //setup EIMSK register for interrupt
	sei();                  //turn on interrupts
	DDRB = 0xff;            //initialize PORTB as output
	while (1)
	{
		PORTB = 0x00; //PORTB pin 3 set to output
		_delay_ms(750); //delay 750ms
		PORTB = 0x08; //PORTB pin 3 turns off
		_delay_ms(250); //delay 250ms
	}
}

ISR (INT0_vect){
	_delay_ms(2000);        //interrupt delay  2000ms
}
```



Design Assignment 2B Circuit

ATMEL Duty Cycle Output:



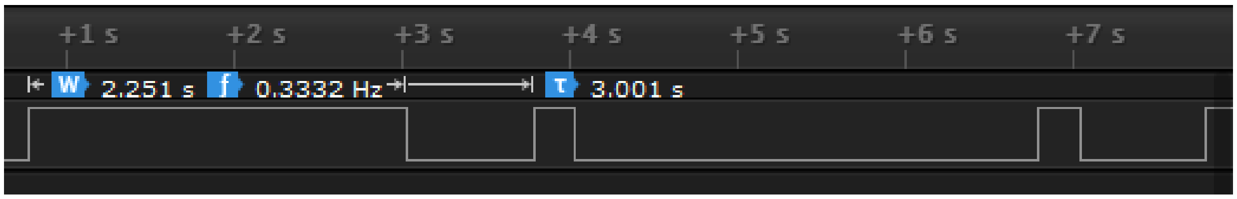
```
#define F_CPU 16000000UL    //define clock speed
#include <avr/io.h>
#include <util/delay.h>     //delay library
#include <avr/interrupt.h>  //interrupt library

int main(void)
{
    PORTD = 0x04;        //set pull up resistor
    EICRA = 0x02;        //set EICRA register for interrupts
    EIMSK = 0x01;        //setup EIMSK register for interrupt
    sei();               //turn on interrupts
    DDRB = 0xff;         //initialize PORTB as output
    while (1)
    {
        PORTB = 0x00;    //PORTB pin 3 set to output
        _delay_ms(750); //delay 750ms
        PORTB = 0x08;    //PORTB pin 3 turns off
        _delay_ms(250); //delay 250ms
    }
}

ISR (INT0_vect){
    _delay_ms(2000);     //interrupt delay  2000ms
}
```

Processor Status

Name	Value
Program Counter	0x00000070
Stack Pointer	0x08FD
X Register	0x0000
Y Register	0x08FF
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	16000028
Frequency	16.000 MHz
Stop Watch	1,000,001.75 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00

Above we can see the confirmation of the 1 second continuous period with a 75% duty cycle using simulation

Waveform Confirmation:



Above we see the button press when duty cycle is high and we get a delay of 2.25 as the .25 standard cycle was already initiated. Afterwards we see a .75 low value and .25 high value as our normal operating period. The button is then pressed while in a low state and we see a delay of 2.75 as the .75 normal duty + full 2 second delay

1. GITHUB LINK OF THIS DA

submission_da/DA_2B at main · brianwolak/submission_da (github.com)