Elizabeth Heider

CPE301 – SPRING 2018

Design Assignment X

**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

The student understands that all required components should be submitted in complete for grading of this assignment.

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SUBMISSION ITEM** | **COMPLETED (Y/N)** | **MARKS**  **(/MAX)** |
| 1 | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |
| 2. | INITIAL CODE OF TASK 1/A |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E |  |  |
| 4. | SCHEMATICS |  |  |
| 5. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 5. | SCREENSHOT OF EACH DEMO |  |  |
| 6. | VIDEO LINKS OF EACH DEMO |  |  |
| 7. | GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

**1) Components List and Connection Block Diagram with PIN**

* ATMEGA 328P Microcontroller Chip
* Green LED
* Pushbutton
* 10K resistors

LED

328P

CHIP

SWITCH

POWER   
SUPPLY

**TASK1 – Assembly Code**

.org 0

SBI DDRB, 2; Set Portb.2 as an output

LDI R17, 0x00;

LDI R18, 0x05; // prescalar value to 1024 [101]

STS TCCR1B, R18; // presc value to register

STS PORTB, R18; //

toggle:

RCALL delay;

EOR R17, R18; toggle

OUT PORTB, R18;

RJMP toggle

delay:

LDS R29, TCNT1H;

LDS R28, TCNT1L;

CPI R28, 0xF3; // 0xF3 = 243

BRSH body

RJMP delay

body:

CPI R29, 0x00;

BRSH done

RJMP delay

done:

LDI R20, 0x00;

STS TCNT1H, R20;

STS TCNT1L, R20;

RET

**TASK1 – C Code**

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

// 1 MHZ clock , 64 prescalar

// set LED out port

// set timer prescaler

DDRB = 0xFF;

PORTB = 0x10; // PortB.2

TCCR1B = 0b00000011; // prescaler set to 64

while (1)

{

if (TCNT1 >= 3905)

{

PORTB = PORTB ^ 0xFF; // toggle LED

TCNT1 = 0; // Reset Counter

}

}

}

**TASK2 – Assembly Code**

.org 0

main:

LDI R16, 0x20;

SBI DDRB, 0x05; // PortB.5 Set as output

LDI R17, 0x00;

LDI R18, 0x00;

OUT DDRD, R18; // input to DDRD

LDI R20, 13;

STS TCCR1B, R20; // prescalar 1024

IN R20, PIND;

ANDI R20, 0x02;

CPI R20, 0x02;

BRNE main

begin:

RCALL delay

EOR R17, R16

OUT PORTB, R17

RJMP begin

delay:

LDS R29, TCNT1H ;

LDS R28, TCNT1L ;

CPI R28, 0xF3;

BRSH body ;

RJMP delay ;

body:

CPI R29, 0x00;

BRSH done

RJMP delay

done:

LDI R20, 0x00;

STS TCNT1H, R20;

LDI R20, 0x00;

STS TCNT1L, R20;

RET

**TASK2 – C Code**

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRB = 0xFF; // Set PortB as an OUTPUT

DDRD = 0x00; // Set Port D as an INPUT

while (1)

{

if((PIND&0b00000100)==0b00000100)

{

PORTB = 0xFF; //LED on when pushbutton is pressed

*\_delay\_ms*(1000); //1 second delay, turn off LED after

}

else

PORTB = 0x00; //Turns off LED when button is not pressed.

}

return 0;

}

**TASK3- Assembly Code**

// DA task 3 ASM

// Program to generate a waveform on PORTB.2 with 50% DC and 0.5 sec period

// (Toggle and LED at every 0.25 seconds for a total period of 0.5 seconds)

start:

; Toggle PORTB.5 every ~1 second

SBI DDRB,2 ;PB.2 as an output

LDI R18,0 ;PB.2 = 0

OUT PORTB,R18

LDI R16,0x04 ;R16 = 0x20: bit 5 = 1

LDI R21, 15 ;initialize loop count

Begin:

LDI R19, 0x0 ;load Timer0 = 0

OUT TCNT0,R19

OUT TCCR0A,R18 ;Timer0: normal mode, internal clock

LDI R17,(1<<CS00) | (1<<CS01) ;Timer0: enabled, prescalar = 64

OUT TCCR0B, R17

Again:

IN R20,TIFR0 ;read Timer0 flags register

SBRS R20,0 ;if overflow (TOV0) is set skip next instruction

RJMP Again

LDI R20,0x0 ;stop/disable Timer0

OUT TCCR0B,R20

LDI R20,(1<<TOV0) ;clear Timer0 overflow flag (TOV0)

OUT TIFR0,R20

DEC R21 ; R21--

BRNE Begin ; Repeat if Timer0 hasn't overflowed 30 times

Toggle:

EOR R18,R16 ;toggle bit 2 of R18

OUT PORTB,R18 ;toggle PB.2

LDI R21, 15 ;reinitialize loop count

RJMP BEGIN

**TASK3 C Code**  
// DA task 3 ASM

// Program to generate a waveform on PORTB.2 with 50% DC and 0.5 sec period

// (Toggle and LED at every 0.25 seconds for a total period of 0.5 seconds)

#include <avr/io.h>

int main(void) {

unsigned char count = 0; // count to keep track of timer0 overflows

DDRB |= (1 << 2); // connect LED to pin PB.2

PORTB = 0; // PB.2 LED is off

while (1) {

// set up Timer0 with prescaler = 64 and normal mode

TCCR0A = 0;

TCCR0B |= (1 << CS01)|(1 << CS00);

TCNT0 = 0; // initialize counter

while( (TIFR0 & 0x1) == 0 ) ; // wait until overflow flag is set

TCCR0B = 0; // stop/disable Timer 0

TIFR0 |= 1; // clear overflow flag

if (count == 15)

{

PORTB ^= 0x04; // toggle PB.2

count = 0; // reset counter

}

else

count++; // increment counter

}

}

**TASK4 Assembly Code**

;DA2 TASK4

; Task 1 Using TIMER0\_OVF\_vect interrupt mechanism

.org 0

jmp main

.org 0x20

jmp T0\_OVF ; Timer0 overflow interrupt vector

main:

SBI DDRB,2 ;PB.2 as an output

LDI R18,0 ;PB.2 = 0

OUT PORTB,R18

LDI R16,0x04 ;

LDI R21, 15 ;initialize loop count to 30

Begin:

LDI R19, 0x0 ;load Timer0 = 0

OUT TCNT0,R19

OUT TCCR0A,R18 ;Timer0: normal mode, internal clock

LDI R17,(1<<CS00) | (1<<CS01) ;Timer0: enabled, prescalar = 64

OUT TCCR0B, R17

;enable interrupts

LDI R20, 0x01 ;can also use (1<<TOIE0)

STS TIMSK0, R20 ;interrupt overflow enabled

SEI ;global interrupts enabled

Loop:

RJMP LOOP ;LOOP INFINITELY!!!!

T0\_OVF:

LDI R20,0x0 ;stop/disable Timer0

OUT TCCR0B,R20

LDI R20,(1<<TOV0) ;R20 = 0x01

OUT TIFR0,R20 ;clear TOV0 flag

DEC R21 ;R21--

BRNE finish ;repeat if Timer0 hasn't overflowed 30 times

LDI R21, 15 ;reinitialize loop count to 30

EOR R18,R16 ;toggle bit 5 of R18

OUT PORTB,R18 ;toggle PB.5

finish:

LDI R19, 0 ;load Timer0 = 0

OUT TCNT0,R19

LDI R17,(1<<CS00) | (1<<CS01) ;Timer0: enabled, prescalar = 1024

OUT TCCR0B, R17

RETI ;Interrupts Enabled

**TASK4 C Code**

#include <avr/io.h>

#include <avr/interrupt.h>

// 1Mhz clock & 64 prescaler

// global variable for keeping track of # of times Timer0 overflows

volatile int count;

// this interrupt service routine (ISR) runs whenever an overflow on Timer0 occurs

ISR (TIMER0\_OVF\_vect)

{

if (count == 15) {

PORTB ^= (1 << 2); // Toggle PB.5

count = 0; // reinitialize cnt

}

else

count++;

}

int main(void) {

count = 0; // initialize count to keep track of number of

DDRB |= (1 << 2); // connect LED to pin PB.2

// set up Timer0 with prescaler = 64 and normal mode

TCCR0A = 0;

TCCR0B |= (1 << CS00)|(1 << CS01);

TCNT0 = 0; // initialize counter

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

sei(); // enable global interrupts

while(1) ; // loop forever

}

**TASK5 Assembly Code**

.org 0

RJMP begin

.org 0x06

RJMP int0\_isr

begin:

; initiate stack pointer

LDI R20, low(RAMEND)

OUT SPL, R20

LDI R20, high(RAMEND)

OUT SPH, R20;

SBI DDRB, 0x05; // PortB.5 set as output

LDI R17, 0x00; //

LDI R20, 0x01; //

OUT EIMSK, R20 ; INT0 set to 1 in EIMSK Register, ext interrupt activated

SEI

loop:

JMP loop

int0\_isr:

LDI R20, 0x01;

LDI R16, 0x20;

EOR R17, R16; XOR to toggle bits

OUT PORTB, R17; Output toggle onto LED

LDI R18, 0xF3; TCNT = 243

check:

SUBI R18, 0x01;

CPI R18, 0x00;

BRNE check

LDI R20, 0x00 ; reset

STS TCNT0, R20 ;

RETI

**TASK5 C Code**

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

ISR (INT0\_vect) // Interrupt Service Routine when switch is pressed

{

PORTB = 0xFF;// Set PORTB on High (ON)

*\_delay\_ms*(250); // delay for 0.25 s

PORTB = 0x00; // Set PORTB on LOW (OFF)

*\_delay\_ms*(250); // delay for 0.24 s

}

int main(void)

{

DDRB = 0xFF; // portb as output

EIMSK = 0x01; // external interrupt enabled

EIFR = 0x01; // external interrupt flag cleared

EICRA = 0x03; // rising edge interrupt request

sei();

while(1) // loop

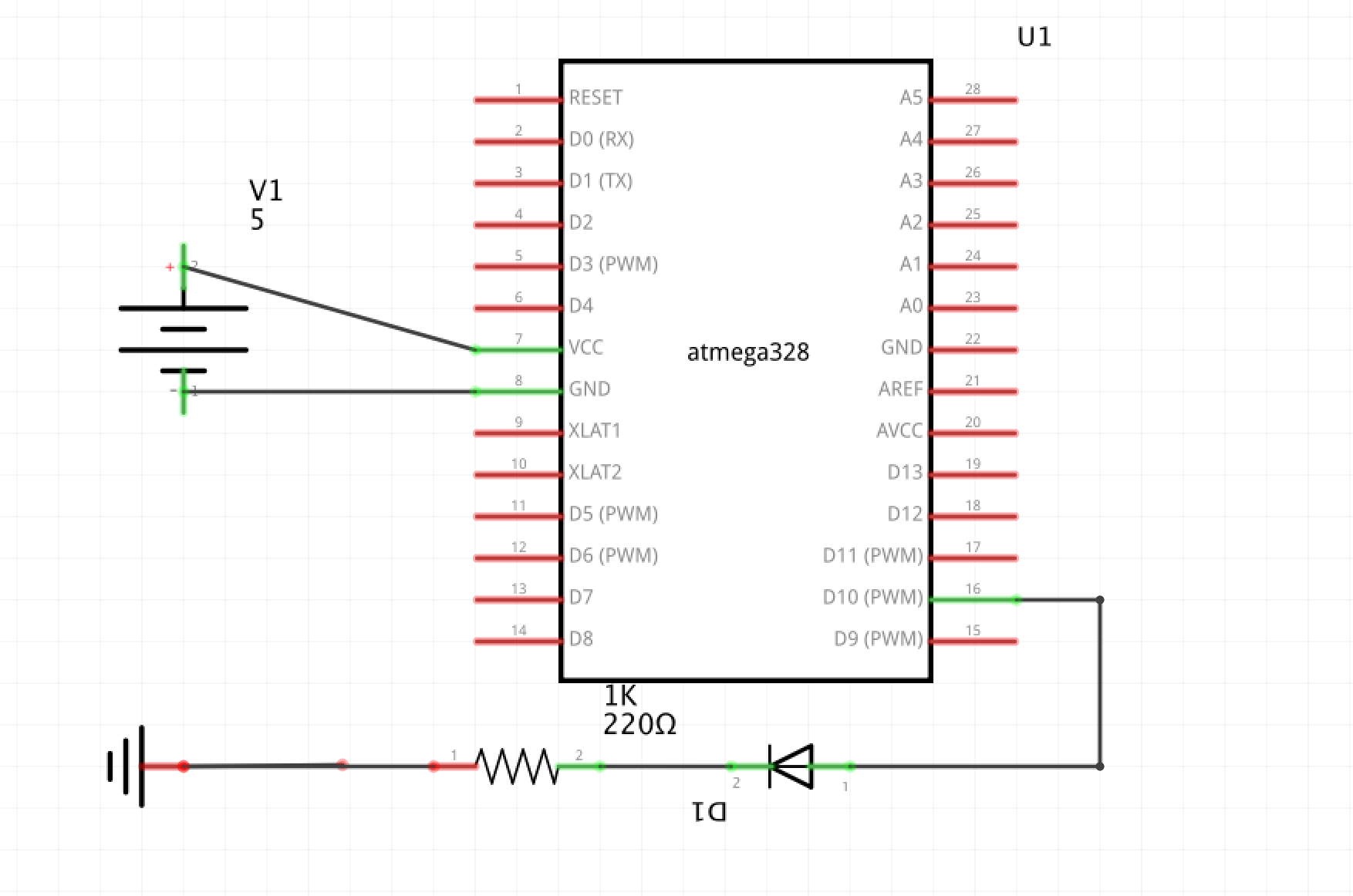
{

}

}

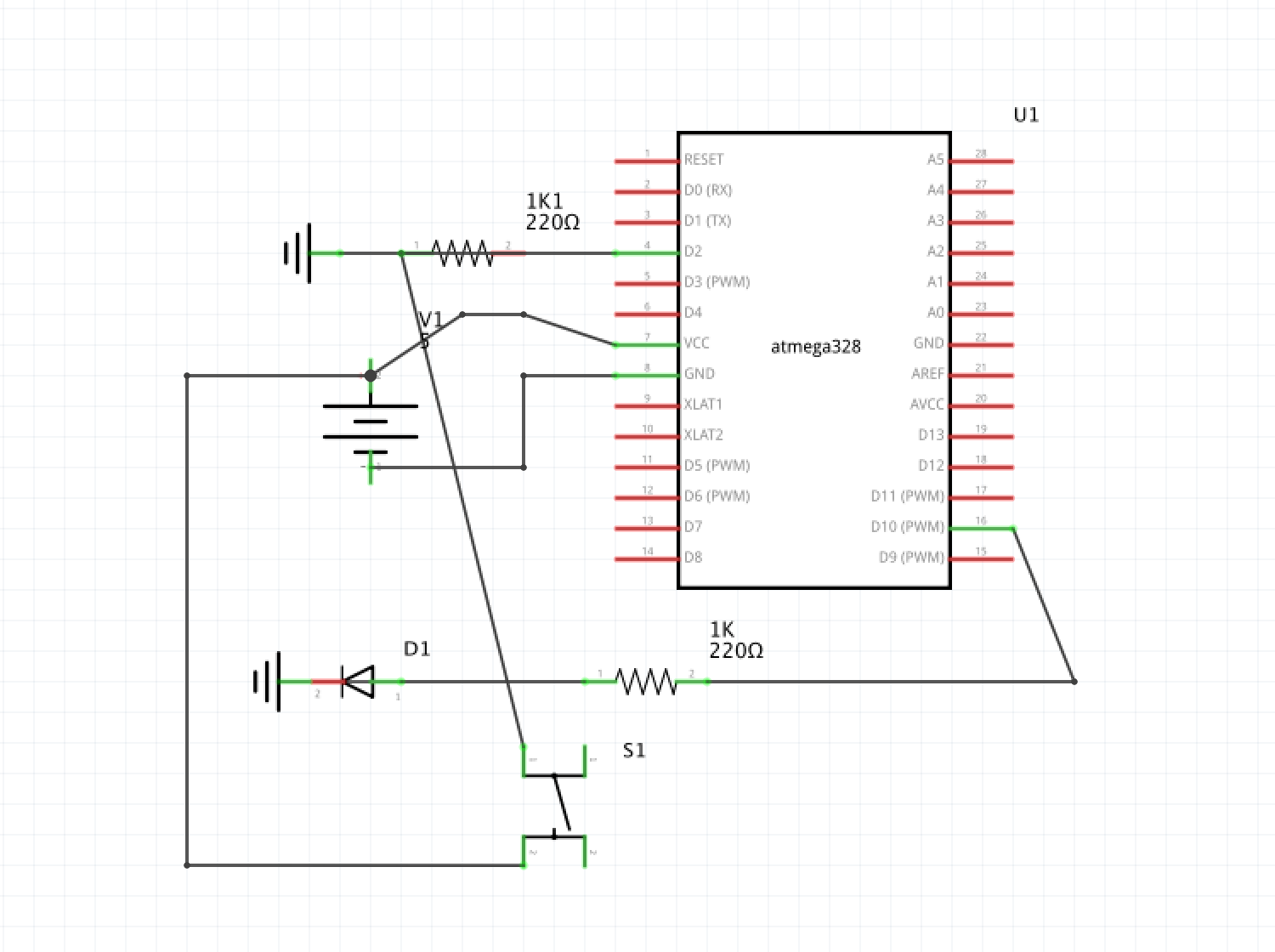
**Schematics**

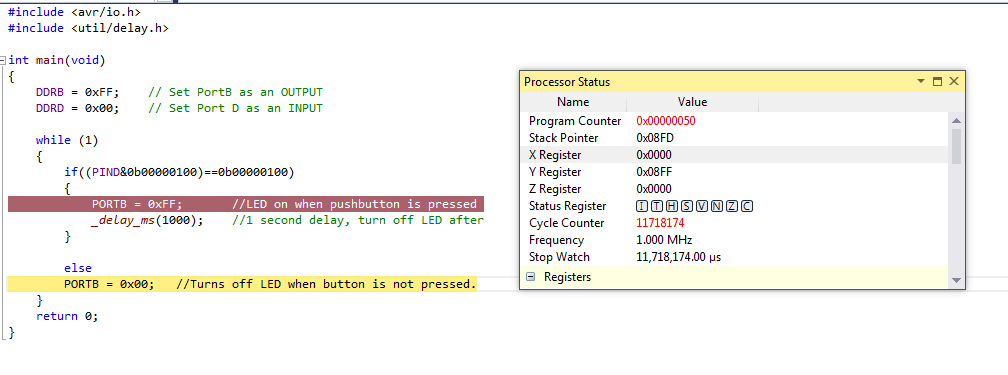
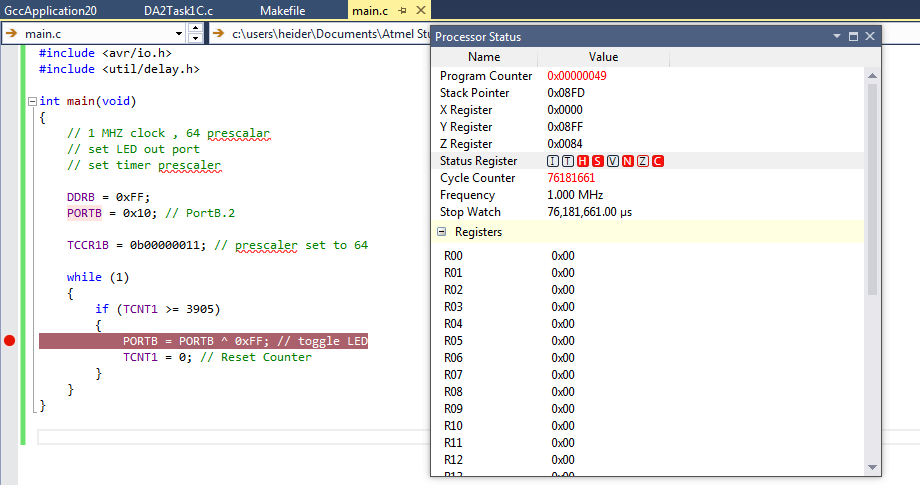
Schematic for Task 1,3 and 4

****

Schematic for Tasks 2 and 5.

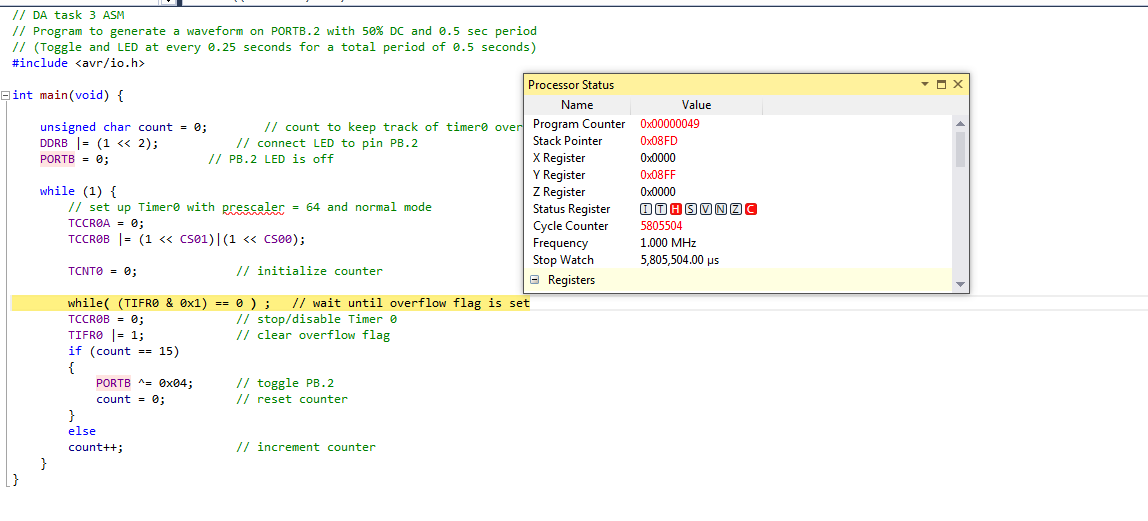
PB.2



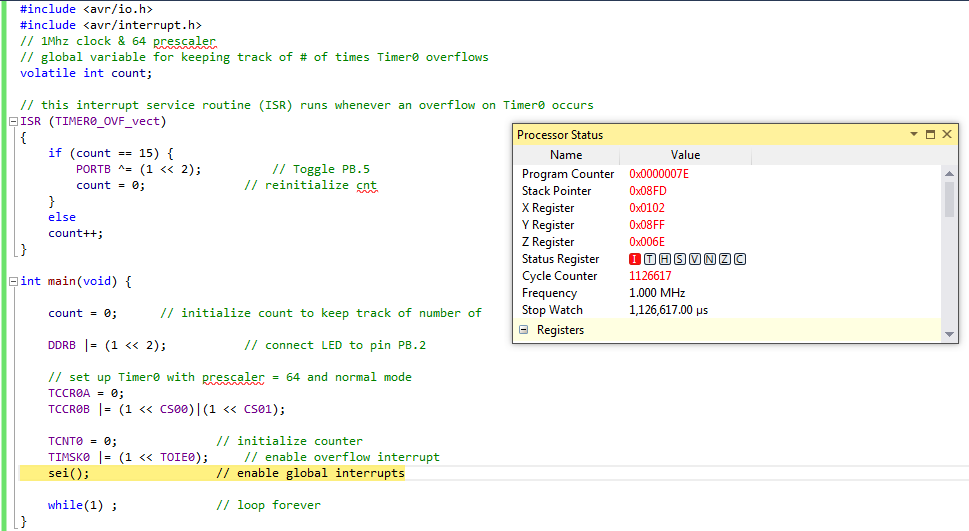
**Screenshots of Output**

Output Simulation Task 1 C Code

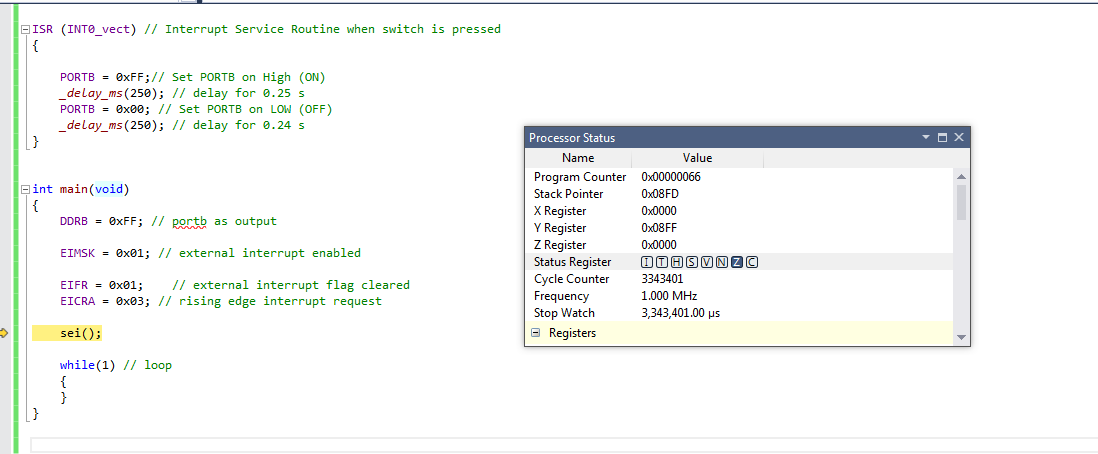
Output Simulation Task 2 C code

****

Output Task 3 C Code

****

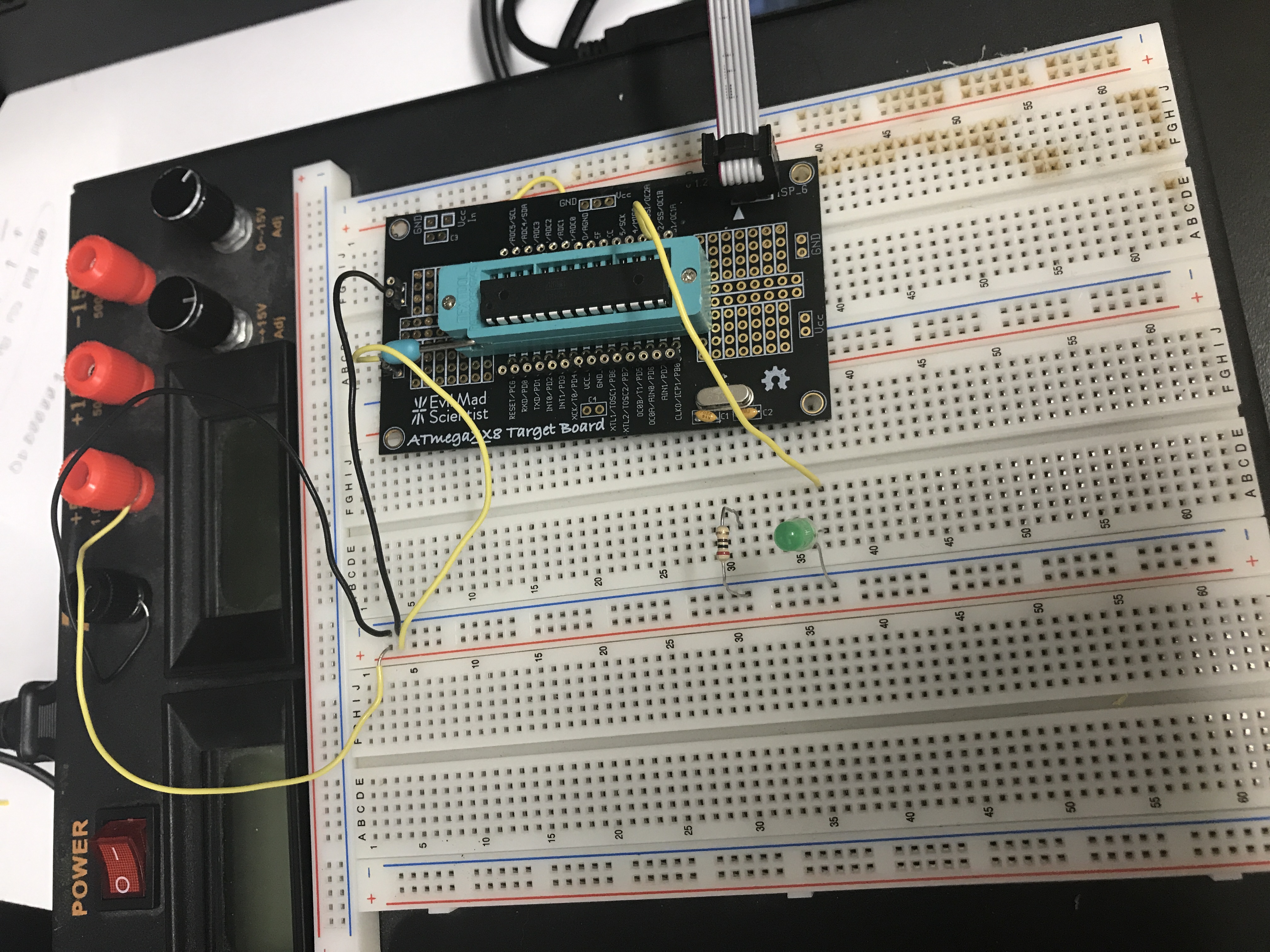
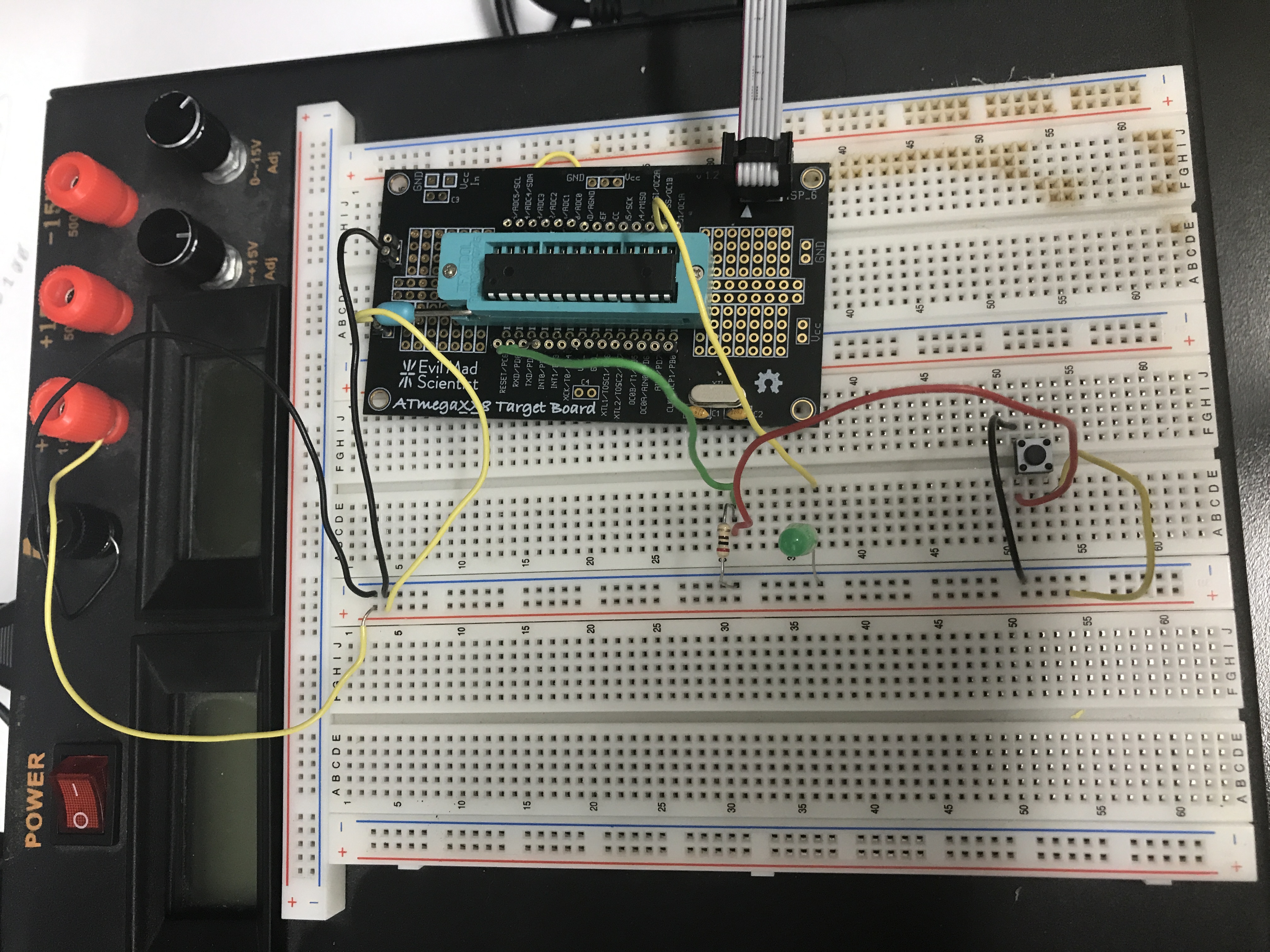
Output Task 4 C Code



Output Task 5 C Code

**Screenshots (Photos) of Each Output**

Breadboard set up for DA2 Tasks 1, 3, 4.



Breadboard set up for DA2 Tasks 2 and 5.

**Screenshots (Photos) of Each Output**

I was having a lot of issues uploading the videos. Will be attached to another document in DA Folder (sorry, thank you)!

**GITHUB LINK OF THIS DA**

[**https://github.com/lizheider/CPE301/tree/master**](https://github.com/lizheider/CPE301/tree/master)