Elizabeth Heider

CPE301 – SPRING 2018

Design Assignment X

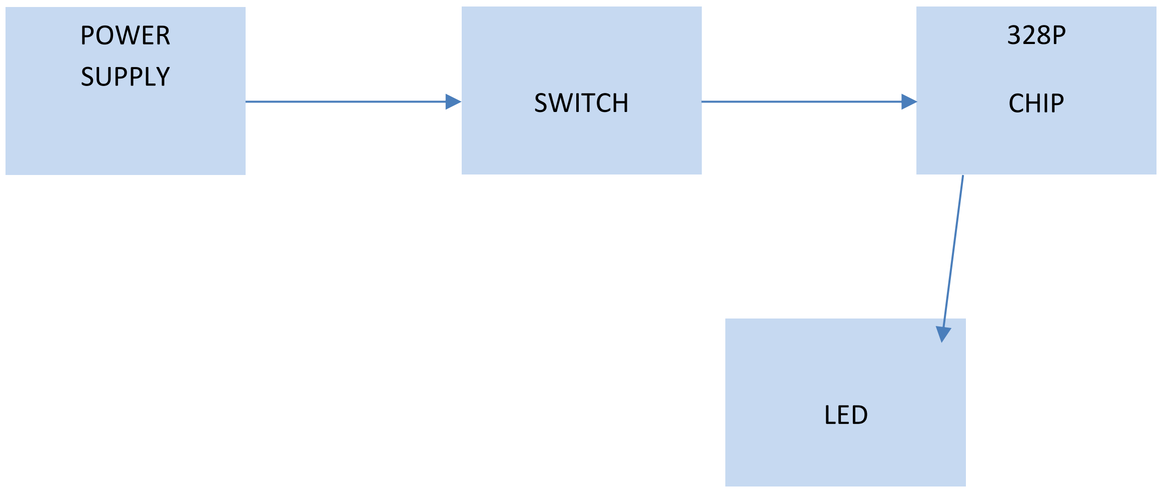
<u>DO NOT REMOVE THIS PAGE DURING SUBMISSION:</u>

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
-



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

```c
#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
}
```

<u>TASK5 Assembly Code</u>

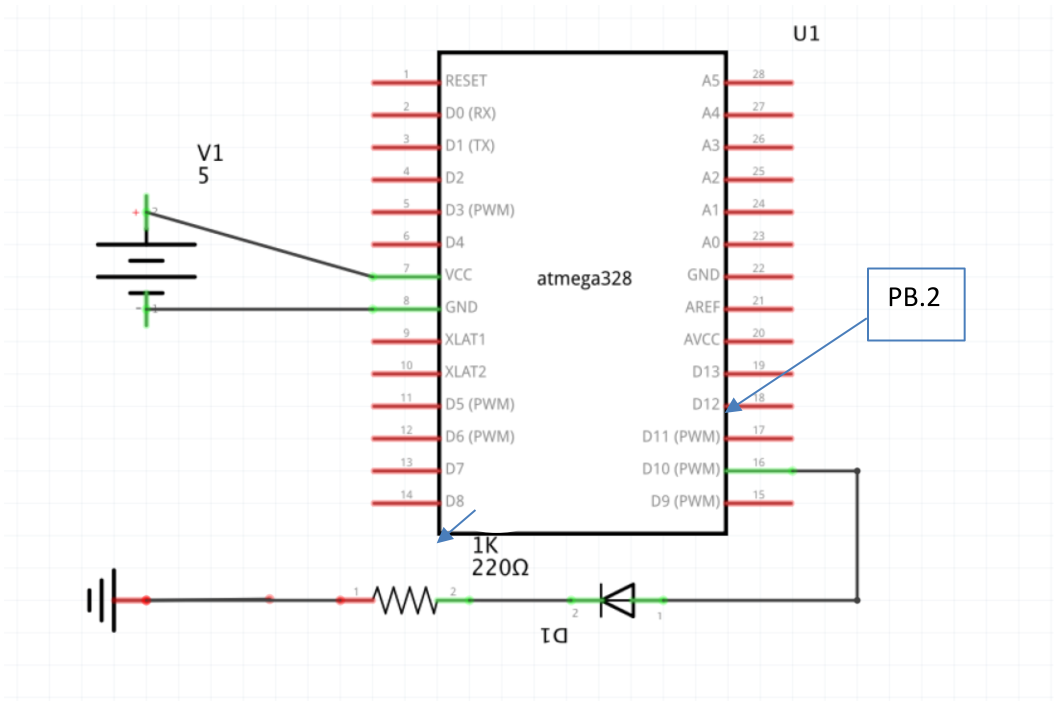
```
.org 0
      RJMP begin

.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

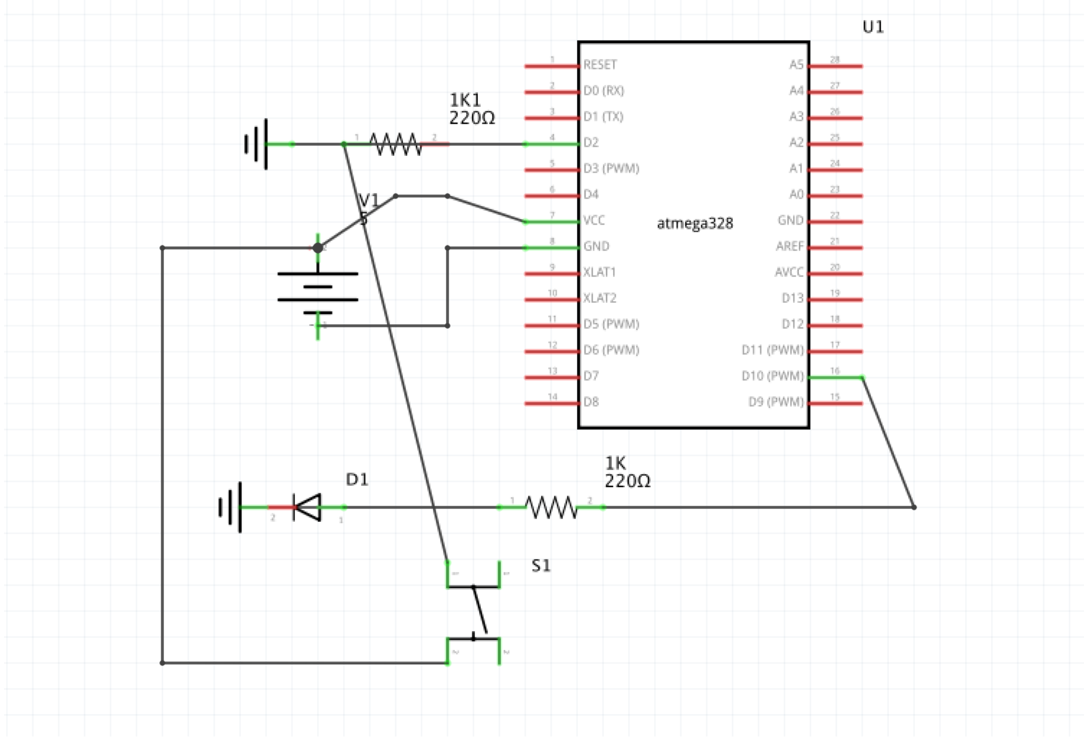
```c
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

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

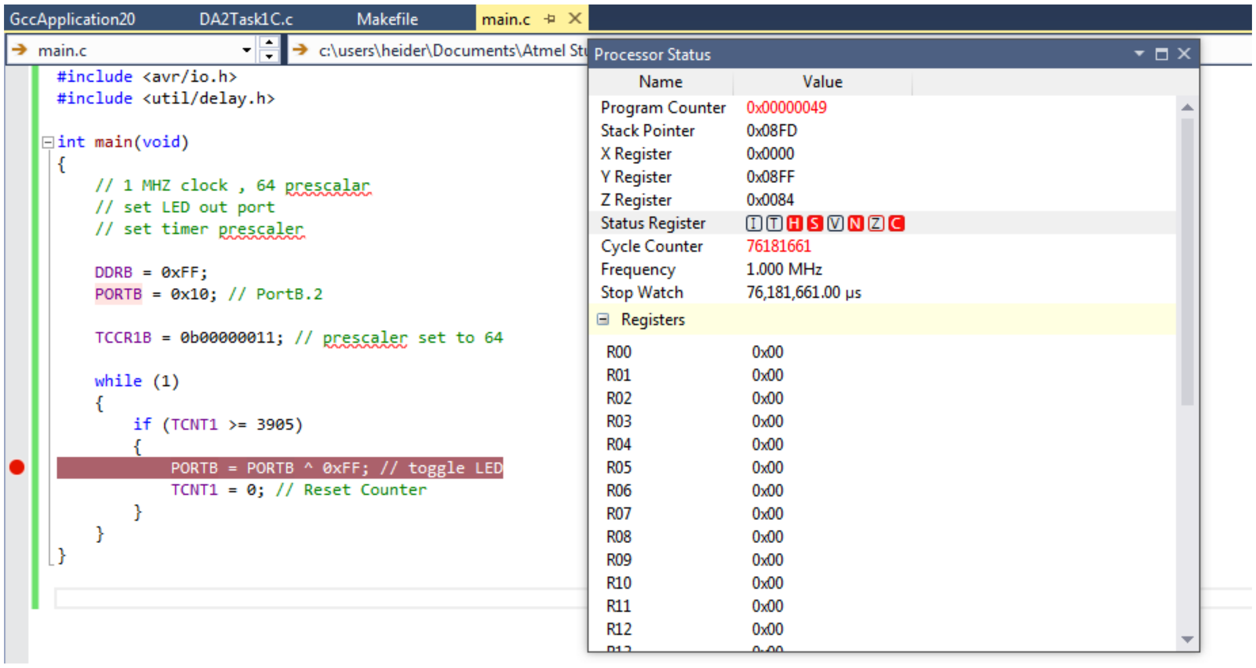
Schematic for Task 1.3 and 4



Schematic for Tasks 2 and 5.



Screenshots of Output

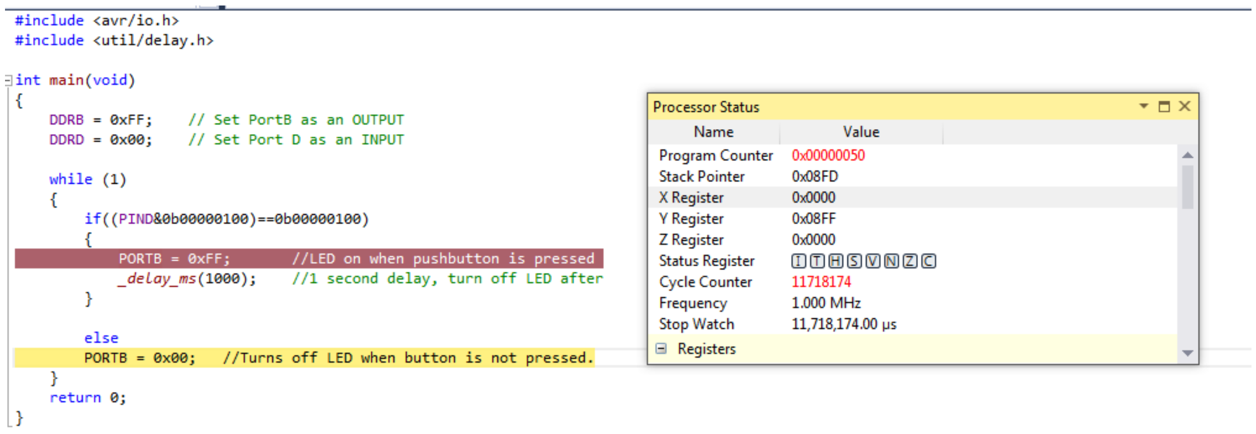


```c
#include <avr/io.h>
#include <util/delay.h>

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
        }
    }
}
```

Processor Status

Name	Value
Program Counter	0x00000049
Stack Pointer	0x08FD
X Register	0x0000
Y Register	0x08FF
Z Register	0x0084
Status Register	I T H S V N Z C
Cycle Counter	76181661
Frequency	1.000 MHz
Stop Watch	76,181,661.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00

Output Simulation Task 1 C Code

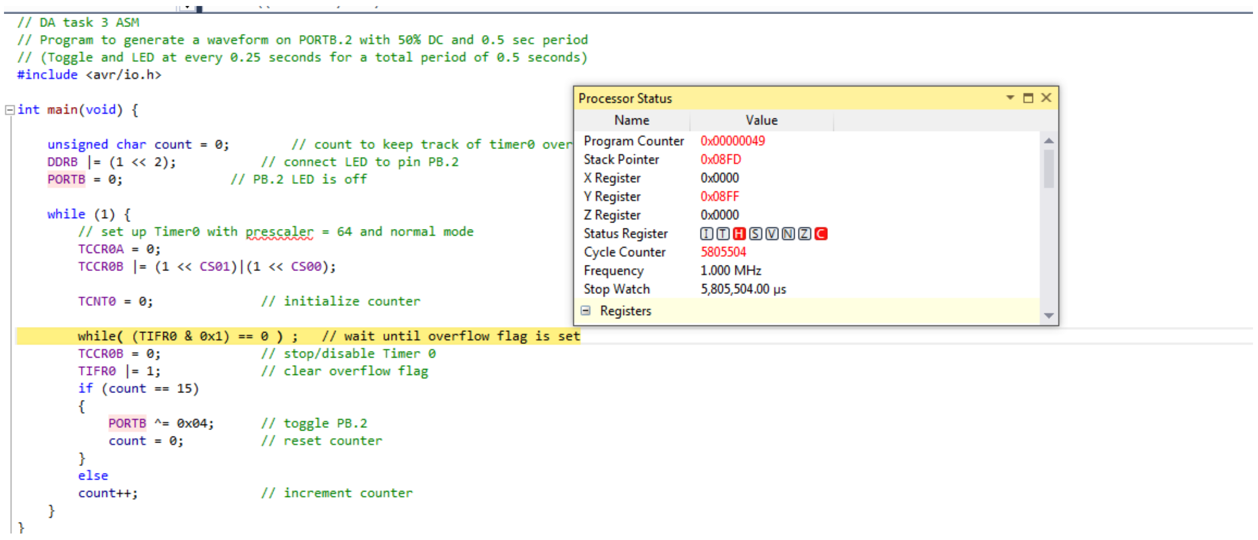


```c
#include <avr/io.h>
#include <util/delay.h>

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;
}
```

Processor Status

Name	Value
Program Counter	0x00000050
Stack Pointer	0x08FD
X Register	0x0000
Y Register	0x08FF
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	11718174
Frequency	1.000 MHz
Stop Watch	11,718,174.00 µs
Registers	

Output Simulation Task 2 C code

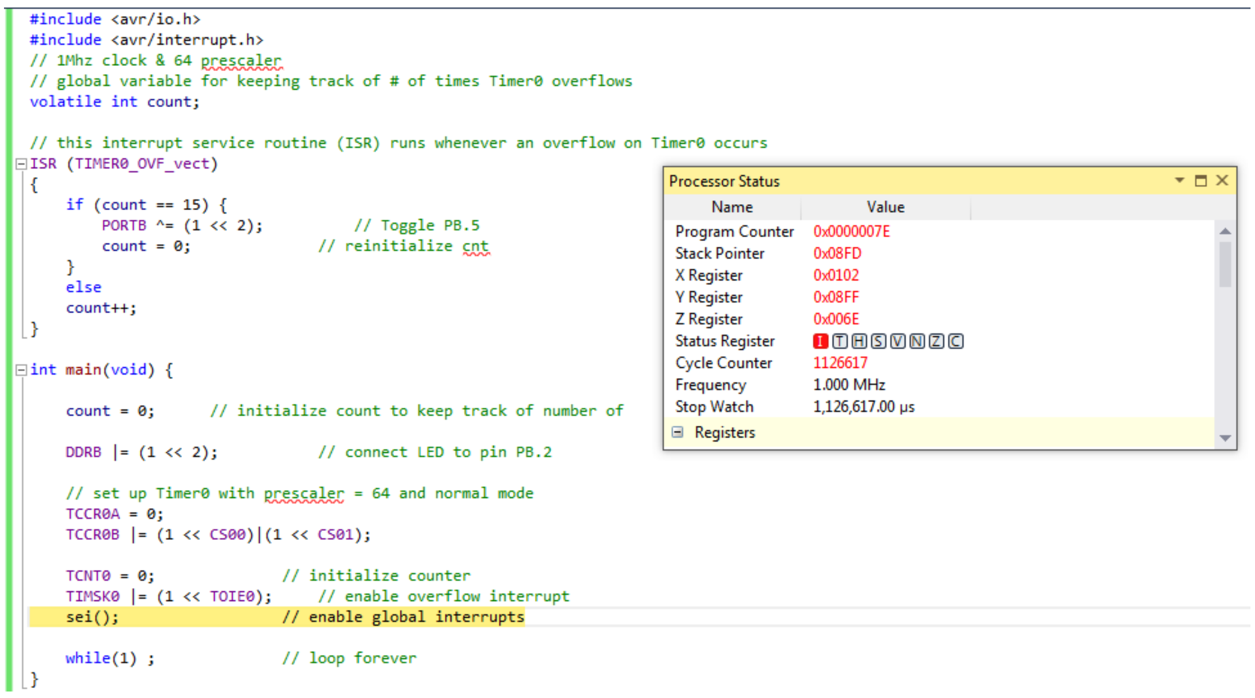


```
// DA task 3 ASM
// Program to generate a waveform on PORTB.2 with 50% DC and 0.5 sec period
// (Toggle and LED at every 0.25 seconds for a total period of 0.5 seconds)
#include <avr/io.h>

int main(void) {

    unsigned char count = 0;        // count to keep track of timer0 over
    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
    }
}
```

Processor Status

Name	Value
Program Counter	0x00000049
Stack Pointer	0x08FD
X Register	0x0000
Y Register	0x08FF
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	5805504
Frequency	1.000 MHz
Stop Watch	5,805,504.00 μs
Registers	

Output Task 3 C Code

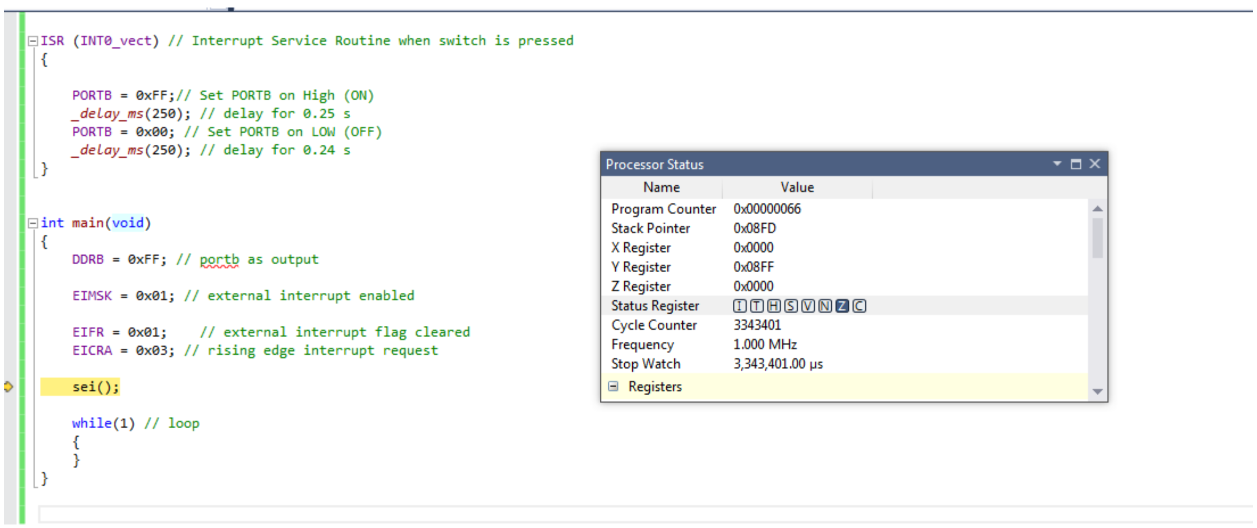


```
#include <avr/io.h>
#include <avr/interrupt.h>
// 1Mhz clock & 64 prescaler
// global variable for keeping track of # of times Timer0 overflows
volatile int count;

// 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
}
```

Processor Status

Name	Value
Program Counter	0x0000007E
Stack Pointer	0x08FD
X Register	0x0102
Y Register	0x08FF
Z Register	0x006E
Status Register	I T H S V N Z C
Cycle Counter	1126617
Frequency	1.000 MHz
Stop Watch	1,126,617.00 μs
Registers	

Output Task 4 C Code



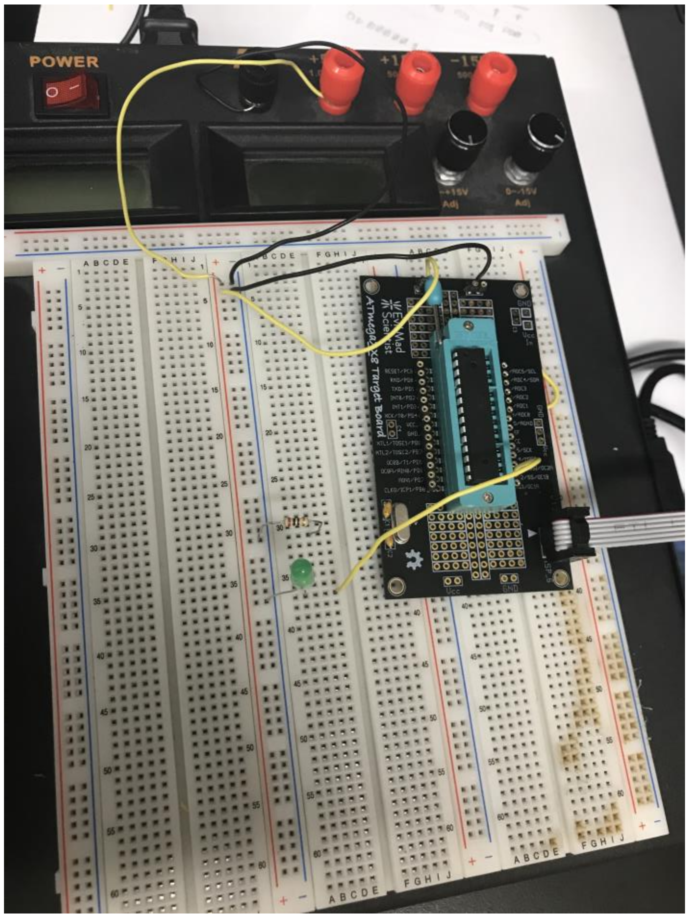
```
ISR (INT0_vect) // Interrupt Service Routine when switch is pressed
{

    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
    {
    }
}
```

Processor Status

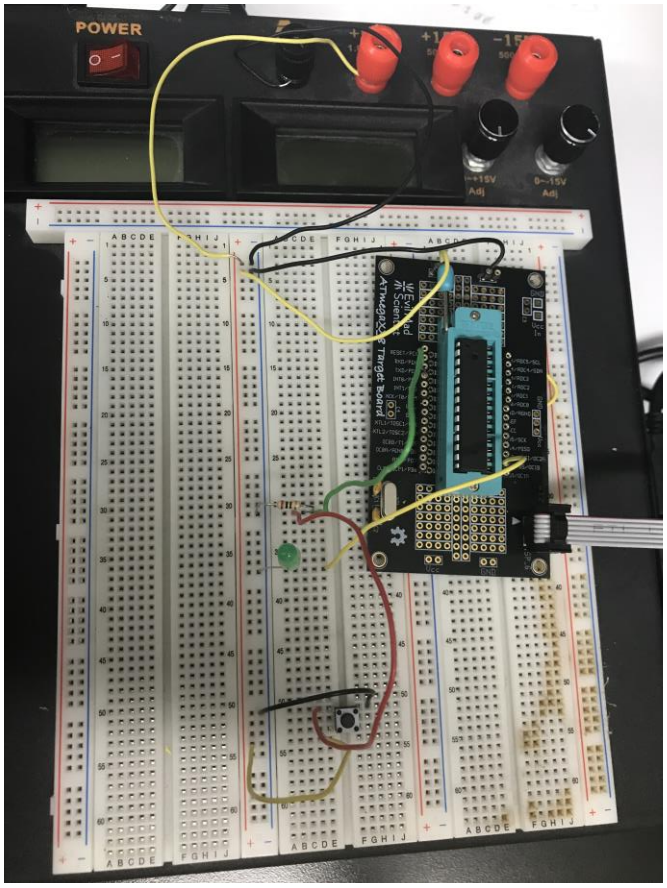
Name	Value
Program Counter	0x00000066
Stack Pointer	0x08FD
X Register	0x0000
Y Register	0x08FF
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	3343401
Frequency	1.000 MHz
Stop Watch	3,343,401.00 μs
Registers	

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