Nathan Hanuscin

CPE301 – SPRING 2018

Design Assignment 2

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 w/ PINS

One 220Ω resistor
Two 10KΩ resistors
One pushbutton
One red LED
(See schematics for diagrams)

2. INITIAL/DEVELOPED CODE OF TASK 1/A

No initial code for any tasks.

3. MODIFIED CODE OF TASK 1/A

Task 1 assembly code:

```
.org 0x00
    LDI R16, HIGH(RAMEND)
    OUT SPH, R16
    LDI R16, LOW(RAMEND)
    OUT SPL, R16

    SBI DDRB, 2             ;set PORT2 as output
    LDI R17, 0              ;used to intialize PB2 OFF
    OUT PORTB, R17          ;turn PB2 off
    LDI R16, 4              ;used to toggle LED

LOOP:
    rcall myDelay           ;call delay subroutine
    EOR R17, R16            ;toggle bits of R17
    OUT PORTB, R17          ;toggle LED
    RJMP LOOP               ;continue looping

myDelay:
    LDI R18, 250            ;counter for delay
L1:
    LDI R19, 250            ;second nested counter for delay
L2:
    NOP                     ;take 1 clock cycle 250*250 times
    DEC R19                 ;take 1 clock cycle 250*250 times and decrement 2nd count
    BRNE L2                 ;keeping delaying if not zero
    DEC R18                 ;decrement primary counter
    BRNE L1                 ;go into nested loop if not zero
    RET                     ;finished .25 second delay
```

Task 1 C code:

```
#include <avr/io.h>
#define F_CPU 1000000UL
#include <util/delay.h>

int main()
{
    DDRB |= (1<<2);             //set PB2 as output
    PORTB &= ~(1<<2);           //set PB2 OFF

    while (1)
    {
        PORTB |= (1<<2);        //Turn on LED
        _delay_ms(250);         //wait 250ms
        PORTB &= ~(1<<2);       //Turn off LED
        _delay_ms(250);         //wait 250ms
    }
}
```

4. MODIFIED CODE OF TASK 2/B

Task 2/B assembly code:

```
.org 0x00
    LDI R16, HIGH(RAMEND)
    OUT SPH, R16
    LDI R16, LOW(RAMEND)
    OUT SPL, R16

    SBI DDRB, 2             ;set PORT2 as output
    LDI R17, 0              ;used to intialize PB2 OFF
    OUT PORTB, R17          ;set PB2 OFF
    CBI DDRD, 2             ;set PD2 as input
    LDI R17 ,0x04           ;set PD2 as input
    OUT PORTD, R17          ;set PD2 as input
    LDI R25, 0

LOOP:
    IN R16, PIND            ;R16 gets PIND values
    CPI R16, 0x00           ;check if button was pressed
    BRNE LOOP               ;if not 0 keep polling
    OUT PORTB, R17          ;Turn on LED
    rcall myDelay250ms      ;call 250ms delay 4 times for
    rcall myDelay250ms      ;overall 1 second delay
    rcall myDelay250ms
    rcall myDelay250ms
    OUT PORTB, R25          ;turn off LED
    jmp LOOP

myDelay250ms:
    LDI R18, 250            ;counter for delay
L1:
    LDI R19, 250            ;second nested counter for delay
L2:
    NOP                     ;take 1 clock cycle 250*250 times
    DEC R19                 ;take 1 clock cycle 250*250 times and decrement 2nd count
    BRNE L2                 ;keeping delaying if not zero
    DEC R18                 ;decrement primary counter
    BRNE L1                 ;go into nested loop if not zero
    RET                     ;finished .25 second delay
```

Task 2/B C code:

```
#include <avr/io.h>
#define F_CPU 1000000UL
#include <util/delay.h>

int main(void)
{
    DDRB |= 0xFF;           //set PORTB as output
    PORTB |= 0x00;          //initialize LED OFF
    PORTD |= 0x04;          //turn on pull-up

    while (1)               //while button is not pressed
    {
        if(PIND & 0x04)         //if the button isn't pressed
        {
            //do nothing
        }
        else
        {
            PORTB |= (1<<2);    //Turn on LED
            _delay_ms(1000);    //wait 1 second
            PORTB &= ~(1<<2);   //Turn off LED
        }

    }


}
```

5. MODIFIED CODE OF TASK 3/C

Task 3/C assembly code:

```
.org 0
    LDI R16, 4          ;used to toggle LED
    LDI R18, 0          ;used to initialize TCCR0A
    SBI DDRB, 2         ;PB2 as output
    LDI R17,0           ;needed to toogle led
    OUT PORTB, R17      ;turn LED off
begin:
    LDI R20, 12         ;250ms delay with 1024 prescaler
    OUT TCNT0, R20      ;load value into timer
    LDI R20, 5          ;to set prescaler
    OUT TCCR0B, R20     ;Prescaler: 1024
    OUT TCCR0A, R18     ;Timer0, normal mode, initalize clock
loop:
    IN R20, TIFR0       ;read in TIFR0
    SBRS R20, 0         ;if TOV0 is set skip next instruction
    RJMP LOOP           ;keep polling

    LDI R20, 0
    OUT TCCR0B, R20     ;stop the timer
    LDI R20, 1
    OUT TIFR0, R20      ;reset TOV0 flag
    EOR R17, R16        ;XOR to toogle led
    OUT PORTB, R17      ;toggle LED
    RJMP begin          ;reset
```

Task 3/C C code:

```
#include <avr/io.h>


int main(void)
{
    DDRB |= (1<<2);             //set PB2 as output
    PORTB &= ~(1<<2);           //turn PB2 LED off
    TCCR0A = 0;                 //Timer0, normal mode, initialize clock
    TCCR0B = 5;                 //prescaler of 1024
    TCNT0 = 12;                 //250ms delay value


    while (1)
    {
        if(TIFR0 & (1 << TOV0))         //If overflow bit is high
        {
            PORTB ^= (1<<2);            //toggle LED
            TIFR0 |= (1<<TOV0);         //reset overflow bit
            TCNT0 = 12;             //reset counter
        }
    }
}
```

6. MODIFIED CODE OF TASK 4/D

Task 4/D assembly code:

```
.org 0x0
    jmp MAIN
.org 0x1A                           ;addr for Timer1 overflow
    jmp T1_OV_ISR
.org 0x100
MAIN:
    ldi R17, HIGH(RAMEND)           ;initialize the stack
    out SPH, R17
    ldi R17, LOW(RAMEND)
    out SPL, R17
    sbi DDRB, 2          ;set PB2 as output
    ldi R17, 0
    out PORTB, R17              ;turn off LED initially
    ldi R17, 0xF0               ;upper bits of 61630
    sts TCNT1H, R17             ;set high bits of counter
    ldi R17, 0xBE               ;lower bits of 61630
    sts TCNT1L, R17             ;set low bits
    ldi R17, 0
    sts TCCR1A, R17             ;normal mode
    ldi R17, 3
    sts TCCR1B, R17             ;set prescaler to 64
    ldi R17, (1<<TOIE1)
    sts TIMSK1, R17             ;set flag bit
    sei                ;enable the interrupt
again:
    jmp again                       ;loop until interrupt occurs

T1_OV_ISR:
    LDI R20, 1<<TOV1            ;clear the flag bit
    sts TIFR1, R20              ;flag bit cleared
    IN R16, PORTB               ;read in PB2
    LDI R17, 0x04
    EOR R16, R17                ;toggle PB2
    OUT PORTB, R16              ;toggle LED
    ldi R17, 0xF0               ;reload upper bits
    sts TCNT1H, R17             ;reset the couter
    ldi R17, 0xBE               ;reload lower bits
    sts TCNT1L, R17             ;reset the counter
    RETI                    ;return
```

Task 4/D C code:

```
#include <avr/io.h>
#include <avr/interrupt.h>

int main()
{
    DDRB |= 0x04;               //set PB2 as output
    PORTB = 0;                  //initialize LED off
    TCCR1A = 0;                 //normal mode
    TCCR1B = 3;                 //set pre-scaler to 64
    TCNT1 = 61630;              //set timer value
    TIMSK1 = (1<<TOIE1);        //enable overflow interrupt
    sei ();                     //enable interrupts

    while(1)
    {
        //wait for interrupt
    }
}

ISR (TIMER1_OVF_vect)
{
    TIFR1 |= (1<<TOV1);         //reset flag bit
    PORTB ^= 0x04;              //toggle LED
    TCNT1 = 61630;              //reset timer
}
```

7. MODIFIED CODE OF TASK 5/E

Task 5/E assembly code:

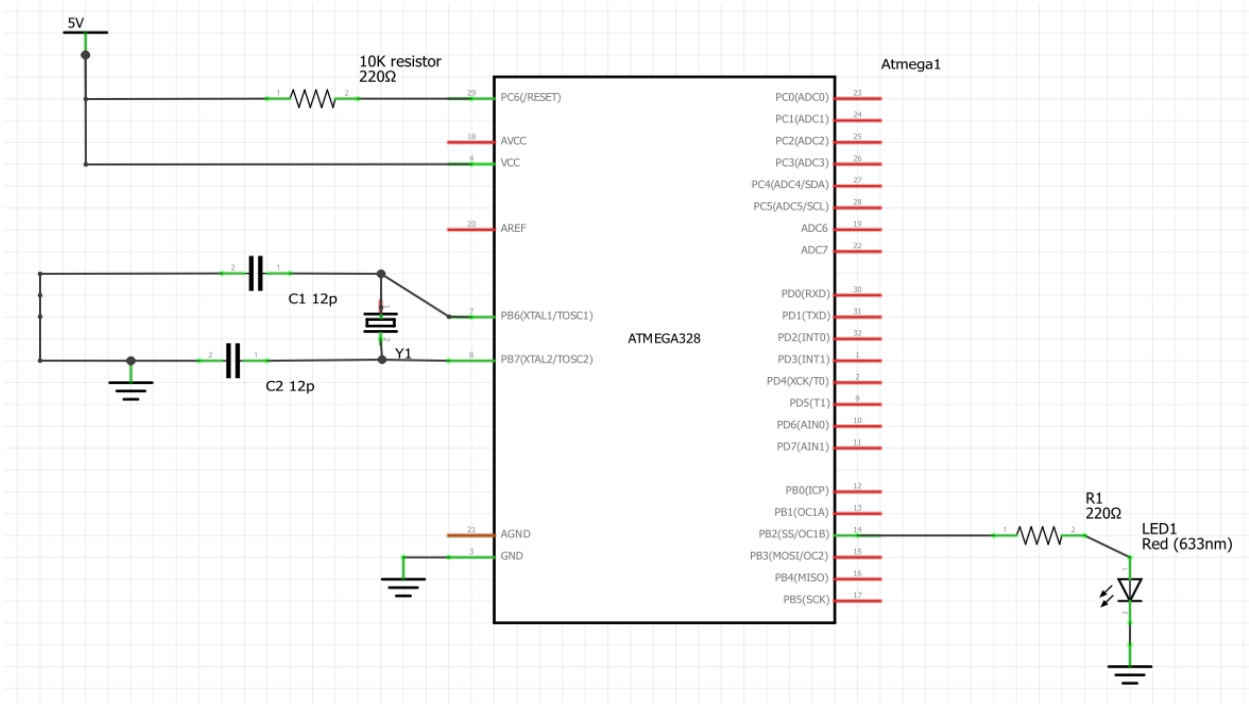
```
.ORG 0                              ;location for reset
     JMP MAIN
.ORG 0x02                           ;location for EXT_INT0
     JMP EX0_ISR
.ORG 0x1A                           ;location for TIM1_OVF
     JMP T1_OV_ISR

MAIN:
     LDI R20,HIGH(RAMEND)           ;initialize the stack
     OUT SPH,R20
     LDI R20,LOW(RAMEND)
     OUT SPL,R20
     SBI DDRB,2                     ;PC.3 = output
     SBI PORTD,2                    ;pull-up activated
     LDI R20,1<<INT0                ;Enable INT0
     OUT EIMSK,R20                  ;Enable INT0
     LDI R20,(1<<ISC01)             ;Configure to falling edge triggered
     sts EICRA,R20
     LDI R20,(1<<TOIE1)             ;Enable timer1 overflow interrupt
     STS TIMSK1, R20                ;Enable TOIE1
     SEI                            ;Set I (Enable Interrupts)

HERE:
     JMP HERE                       ;wait for interrupt

EX0_ISR:
     LDI R20, 1<<INTF0              ;load in flag position
     OUT EIFR, R20                  ;clear flag
     LDI R22,0x04                   ;send 1 to PB2
     OUT PORTB,R22                  ;turn on LED
     LDI R17, 0xF0                  ;get upper bits of 61630
     STS TCNT1H, R17                ;set high bits of counter
     LDI R17, 0xBE                  ;get lower bits of 61630
     STS TCNT1L, R17                ;set lower bits
     LDI R17, 0
     STS TCCR1A, R17                ;normal mode
     LDI R17, 4
     STS TCCR1B, R17                ;set prescaler to 256
     RETI                           ;finish interrupt

T1_OV_ISR:
    LDI R20, 1<<TOV1                ;clear flag bit
    STS TIFR1, R20
    LDI R17, 0
    OUT PORTB, R17                  ;turn off LED
    STS TCCR1B, R17                 ;turn off timer
    RETI                            ;finish interrupt
```

Task 5/E C code:

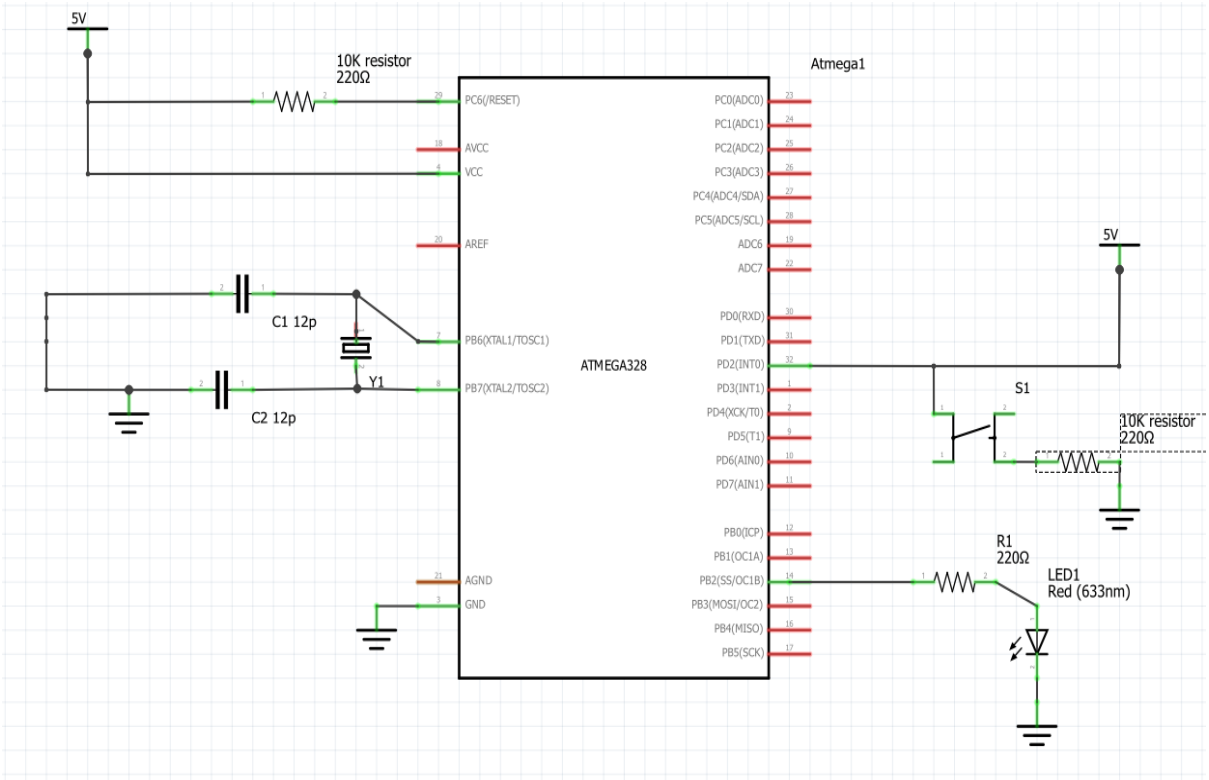
```
#include <avr/io.h>
#include <avr/interrupt.h>

int main()
{
    DDRB = 1<<2;                //set PB2 as output
    PORTD = 1<<2;               //set up pull up resistor
    EIMSK = (1<<INT0);          //enable external interrupt 0
    EICRA = (1<<ISC01);         //falling edge trigger
    TIMSK1 = (1<<TOIE1);        //overflow interrupt timer1 enabled
    sei();

    while (1)
    {
        //wait for interrupts
    }
}
ISR (INT0_vect)
{
    EIFR |= (1<<INTF0);         //reset flag
    PORTB |= (1<<2);            //turn on LED
    TCNT1 = 61630;              //set counter value
    TCCR1A = 0;                 //normal mode
    TCCR1B = (1<<CS12);         //set prescaler to 256
}

ISR (TIMER1_OVF_vect)
{
    TIFR1 |= (1<<TOV1);         //reset flag
    PORTB &= ~(1<<2);           //turn LED off
    TCCR1B = 0;                 //turn off timer
}
```

8. SCHEMATICS

Schematic for tasks 1,3, and 4



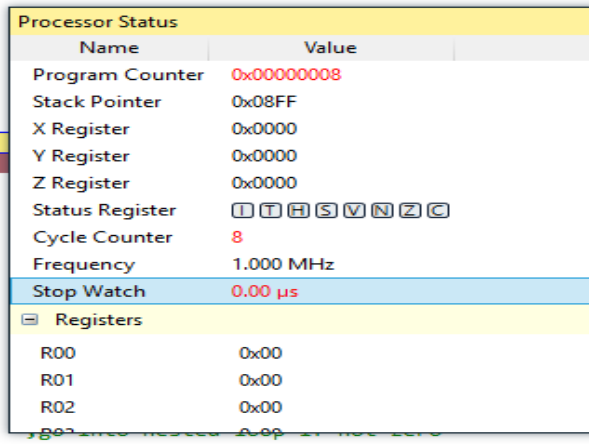
Schematic for tasks 2 and 5



9. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

Task 1/A assembly before delay:

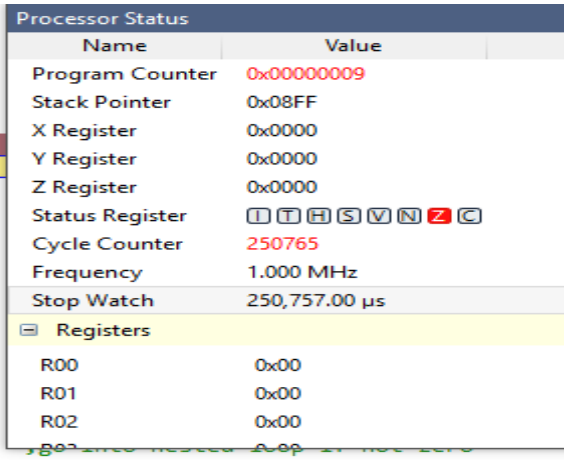
```
    SBI DDRB, 2
    LDI R17, 0
    OUT PORTB, R17
    LDI R16, 4

LOOP:
    rcall myDelay
    EOR R17, R16
    OUT PORTB, R17
    RJMP LOOP

myDelay:
    LDI R18, 250
L1:
    LDI R19, 250
L2:
    NOP
    DEC R19
    BRNE L2
    DEC R18
    BRNE L1
    RET
```




```
;finished .25 second delay
```

After delay:

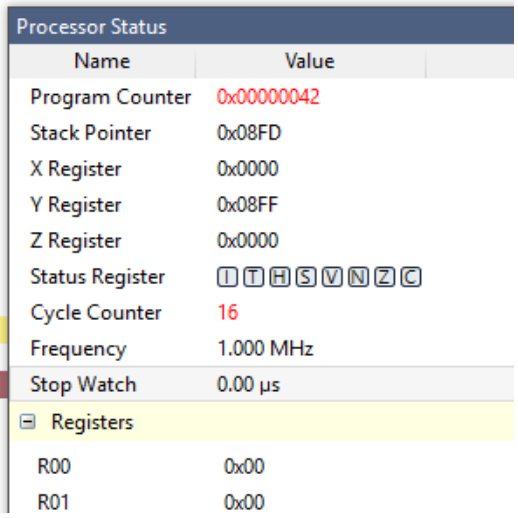
```
    SBI DDRB, 2
    LDI R17, 0
    OUT PORTB, R17
    LDI R16, 4

LOOP:
    rcall myDelay
    EOR R17, R16
    OUT PORTB, R17
    RJMP LOOP

myDelay:
    LDI R18, 250
L1:
    LDI R19, 250
L2:
    NOP
    DEC R19
    BRNE L2
    DEC R18
    BRNE L1
    RET
```




```
;finished .25 second delay
```

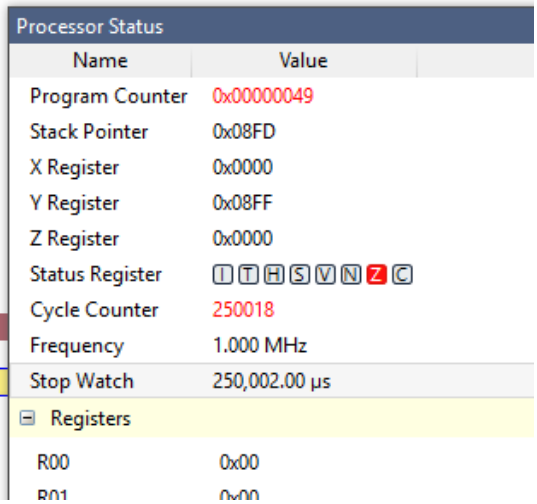
Task 1/A C code before delay:

```
#include <avr/io.h>
#define F_CPU 1000000UL
#include <util/delay.h>

int main()
{
    DDRB |= (1<<2);
    PORTB &= ~(1<<2);

    while (1)
    {
        PORTB |= (1<<2);
        _delay_ms(250);
        PORTB &= ~(1<<2);
        _delay_ms(250);
    }
}
```



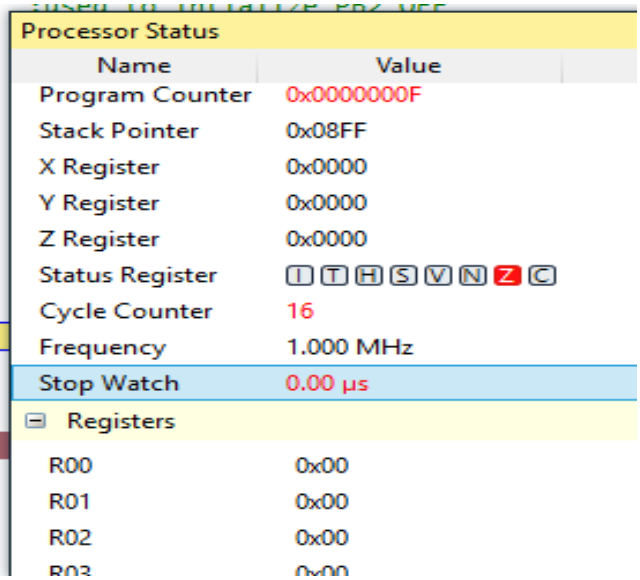
After delay:

```
#include <avr/io.h>
#define F_CPU 1000000UL
#include <util/delay.h>

int main()
{
    DDRB |= (1<<2);
    PORTB &= ~(1<<2);

    while (1)
    {
        PORTB |= (1<<2);
        _delay_ms(250);
        PORTB &= ~(1<<2);
        _delay_ms(250);
    }
}
```



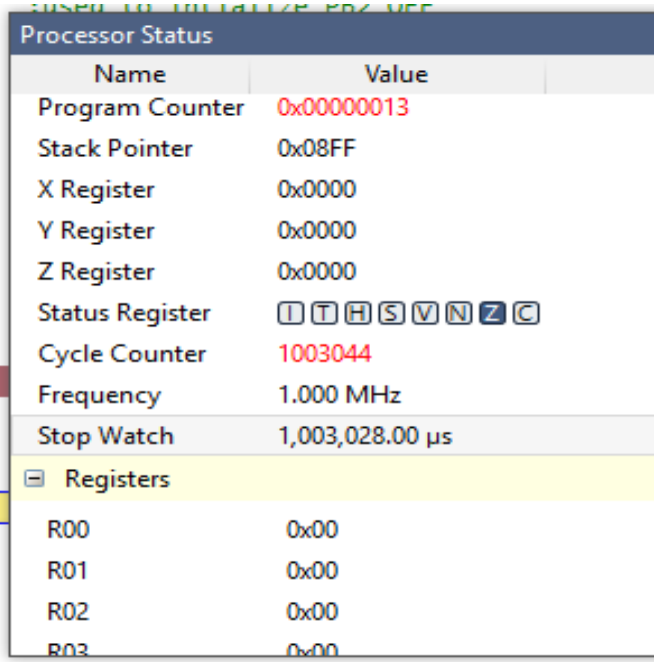
Task 2/B assembly before button pushed:

```
LDI R17, 0
OUT PORTB, R17
CBI DDRD, 2
LDI R17 ,0x04
OUT PORTD, R17
LDI R25, 0

LOOP:
IN R16, PIND
CPI R16, 0x00
BRNE LOOP
OUT PORTB, R17
rcall myDelay250ms
rcall myDelay250ms
rcall myDelay250ms
rcall myDelay250ms
OUT PORTB, R25
jmp LOOP

myDelay250ms:
LDI R18, 250
```

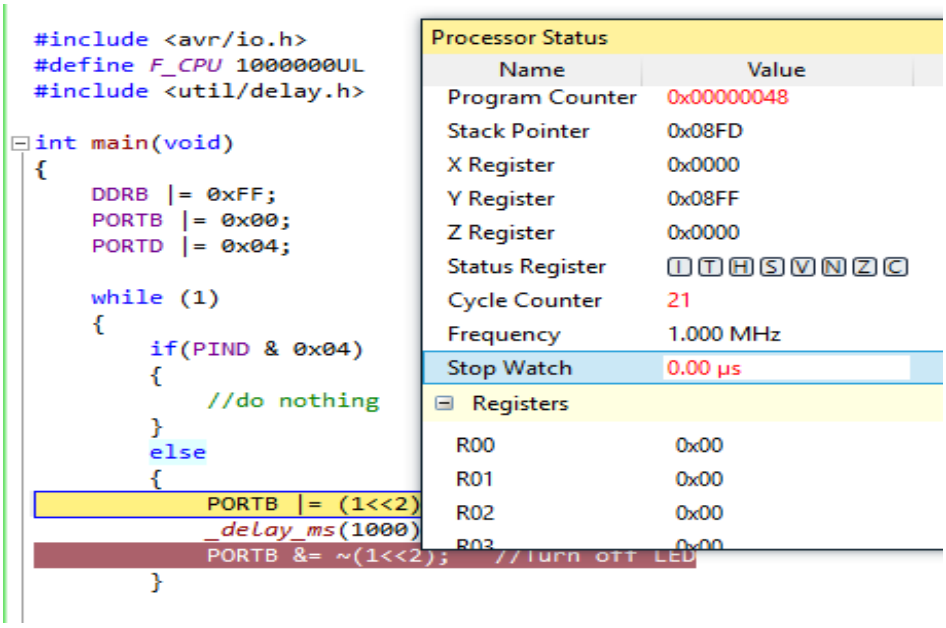


After button pushed:

```
LDI R17, 0
OUT PORTB, R17
CBI DDRD, 2
LDI R17 ,0x04
OUT PORTD, R17
LDI R25, 0

LOOP:
IN R16, PIND
CPI R16, 0x00
BRNE LOOP
OUT PORTB, R17
rcall myDelay250ms
rcall myDelay250ms
rcall myDelay250ms
rcall myDelay250ms
OUT PORTB, R25
jmp LOOP

myDelay250ms:
LDI R18, 250
```



Task 2/B C code before button pushed:

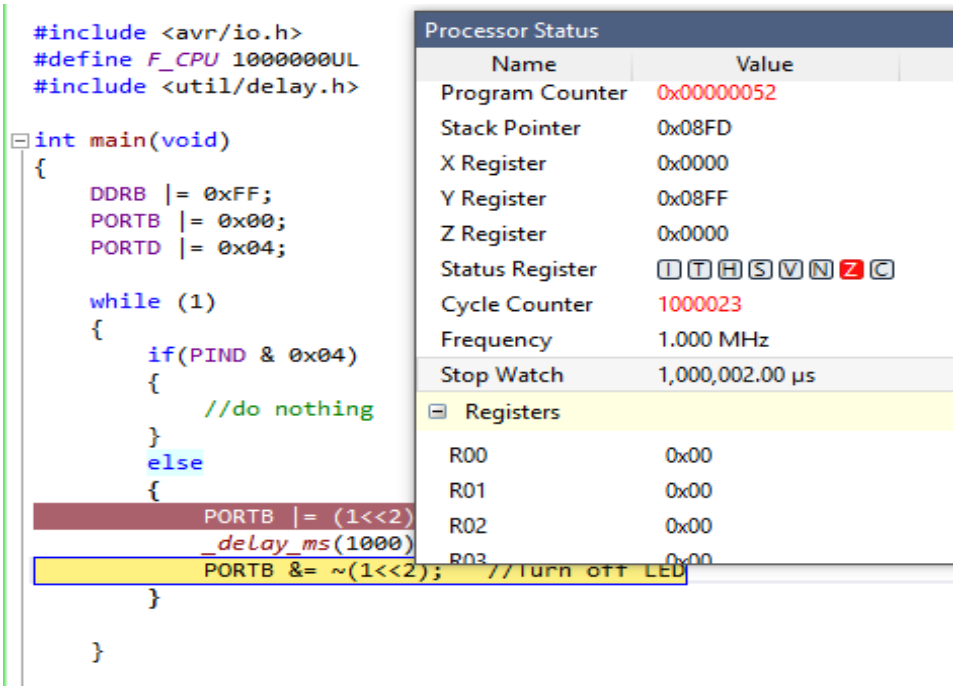


```
#include <avr/io.h>
#define F_CPU 1000000UL
#include <util/delay.h>

int main(void)
{
    DDRB |= 0xFF;
    PORTB |= 0x00;
    PORTD |= 0x04;

    while (1)
    {
        if(PIND & 0x04)
        {
            //do nothing
        }
        else
        {
            PORTB |= (1<<2);
            _delay_ms(1000);
            PORTB &= ~(1<<2);   //Turn off LED
        }
```

Processor Status

Name	Value
Program Counter	0x00000048
Stack Pointer	0x08FD
X Register	0x0000
Y Register	0x08FF
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	21
Frequency	1.000 MHz
Stop Watch	0.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00

After button pushed:

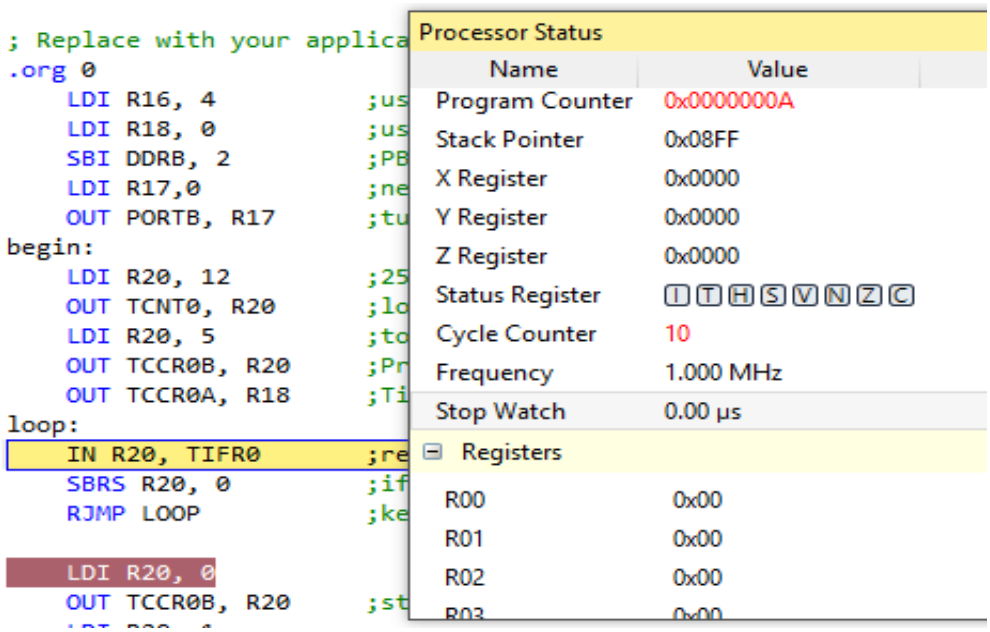


```
#include <avr/io.h>
#define F_CPU 1000000UL
#include <util/delay.h>

int main(void)
{
    DDRB |= 0xFF;
    PORTB |= 0x00;
    PORTD |= 0x04;

    while (1)
    {
        if(PIND & 0x04)
        {
            //do nothing
        }
        else
        {
            PORTB |= (1<<2);
            _delay_ms(1000);
            PORTB &= ~(1<<2);   //Turn off LED
        }

    }
```

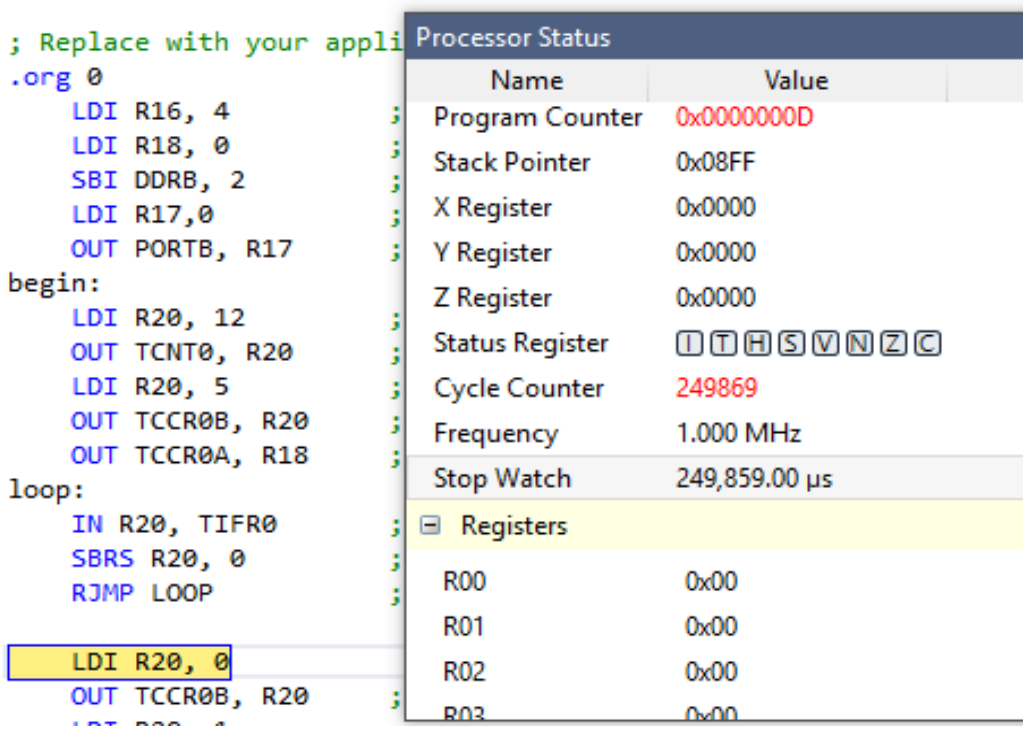
Processor Status

Name	Value
Program Counter	0x00000052
Stack Pointer	0x08FD
X Register	0x0000
Y Register	0x08FF
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	1000023
Frequency	1.000 MHz
Stop Watch	1,000,002.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00

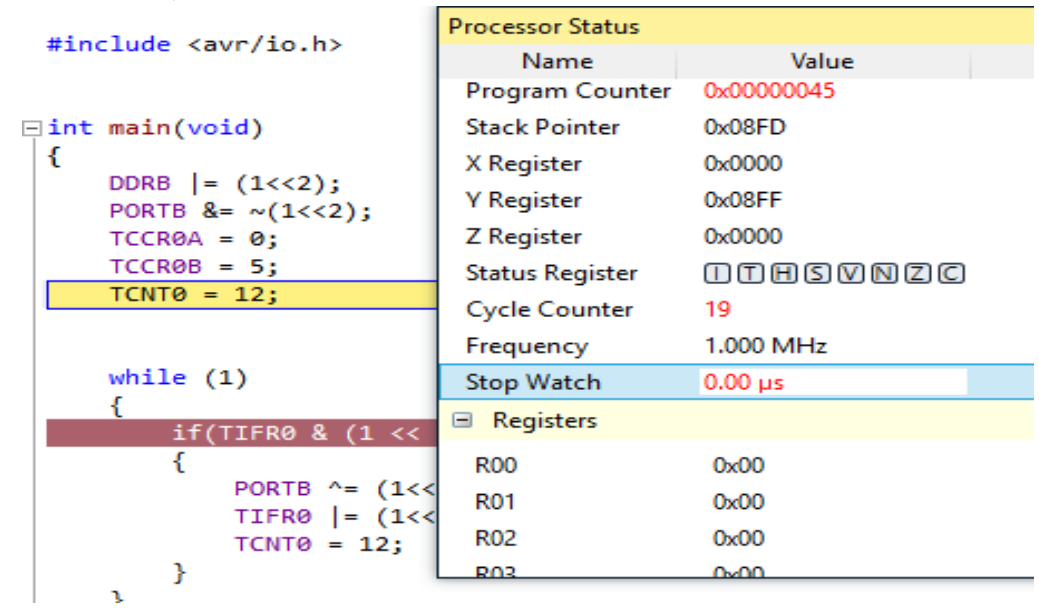
Task 3/C assembly before timer overflow:



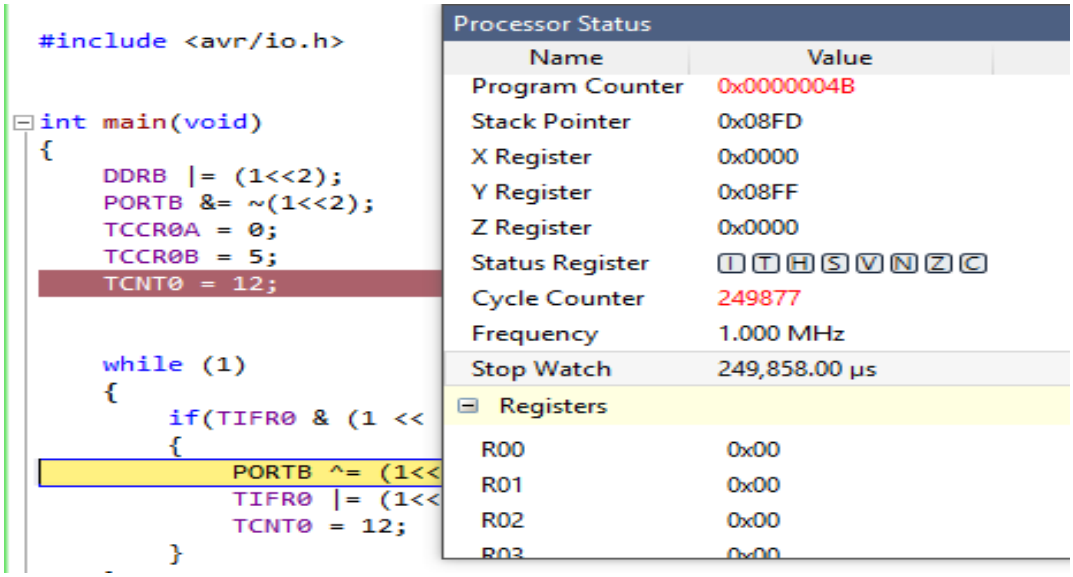
After timer overflow:



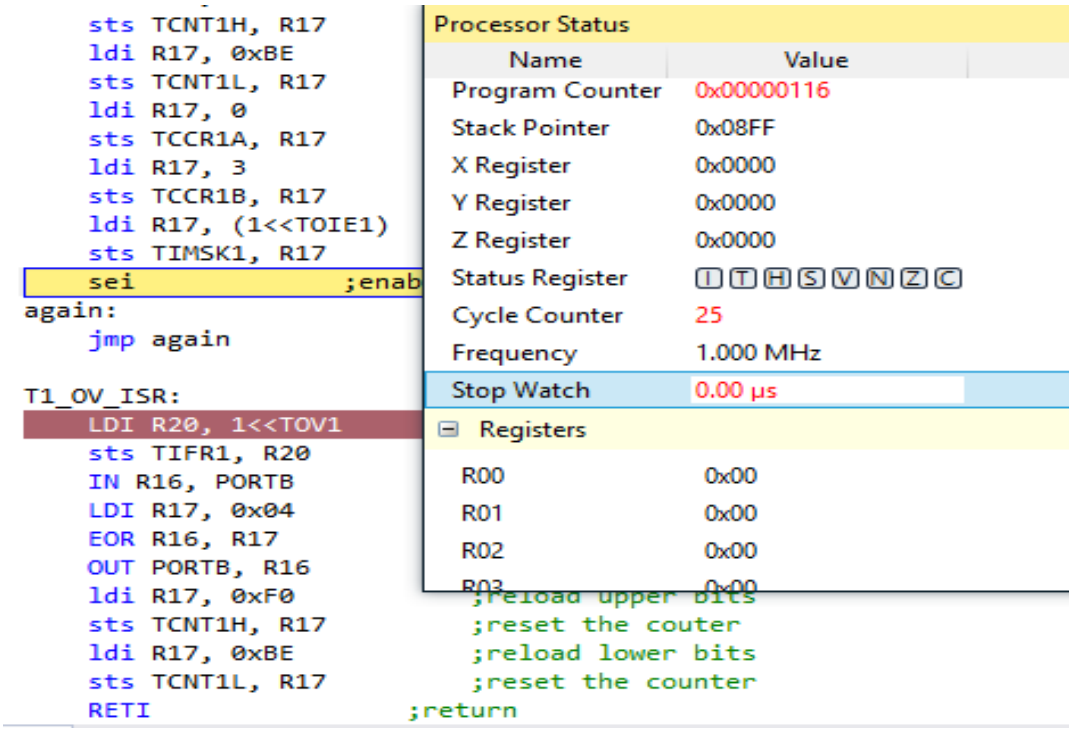
Task 3/C C code before timer overflow:



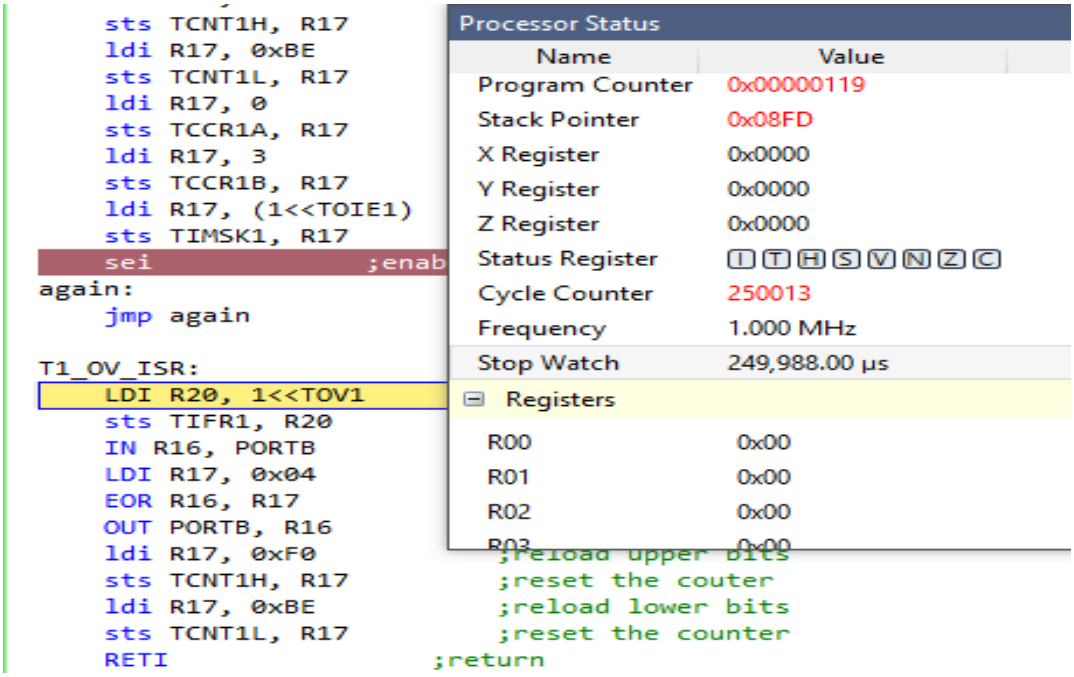
After timer overflow:



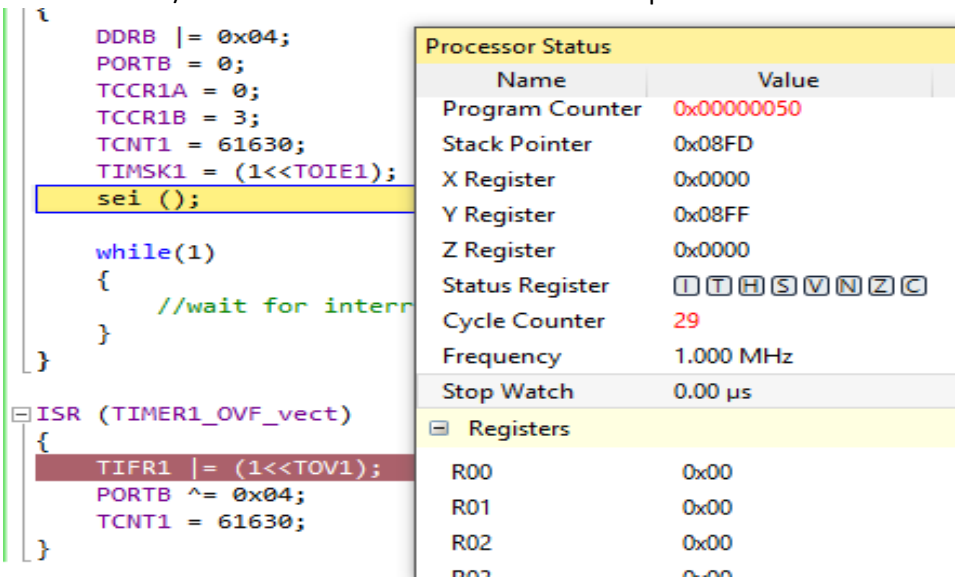
Task 4/D assembly code before timer interrupt overflow:



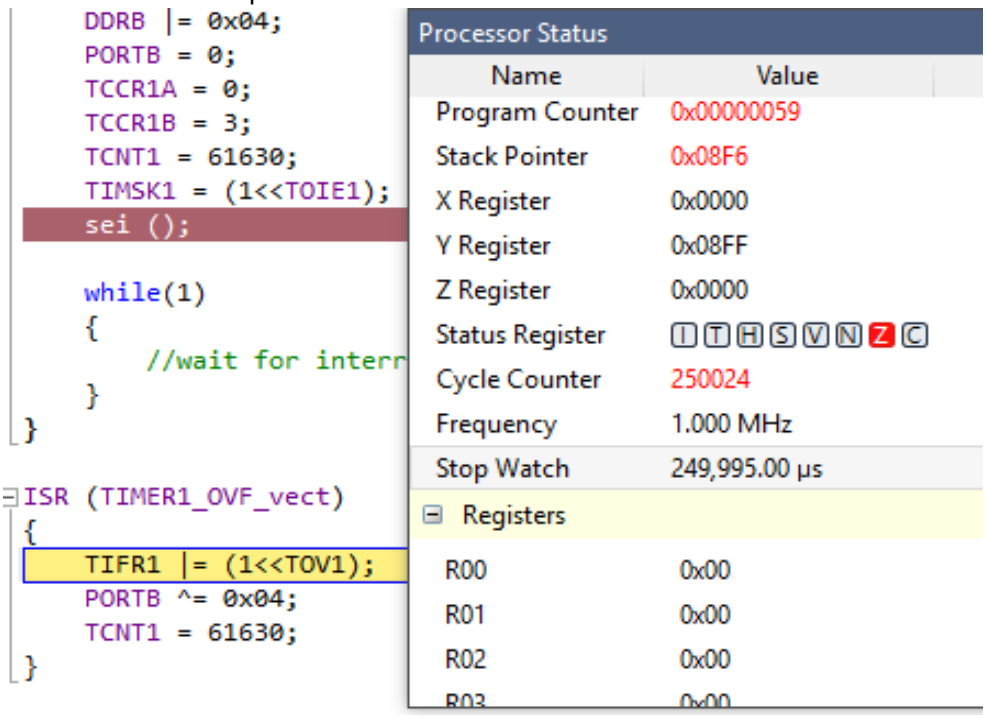
After interrupt occurs:



Task 4/D C code before timer overflow interrupt:



After interrupt occurs:



Task 5/E assembly before timer interrupt:

```
    LDI R17, 0xF0
    STS TCNT1H, R17
    LDI R17, 0xBE
    STS TCNT1L, R17
    LDI R17, 0
    STS TCCR1A, R17
    LDI R17, 4
    STS TCCR1B, R17
    RETI

T1_OV_ISR:
    LDI R20, 1<<TOV1
    STS TIFR1, R20
    LDI R17, 0
    OUT PORTB, R17
    STS TCCR1B, R17
    RETI
```

Processor Status

Name	Value
Program Counter	0x0000003B
Stack Pointer	0x08FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	99
Frequency	1.000 MHz
Stop Watch	0.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00

After interrupt:

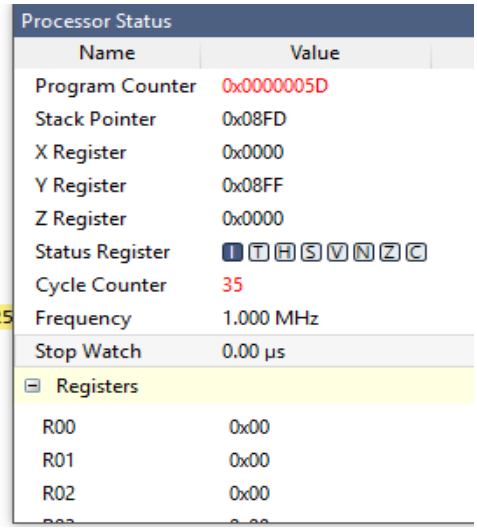
```
    LDI R17, 0xF0
    STS TCNT1H, R17
    LDI R17, 0xBE
    STS TCNT1L, R17
    LDI R17, 0
    STS TCCR1A, R17
    LDI R17, 4
    STS TCCR1B, R17
    RETI

T1_OV_ISR:
    LDI R20, 1<<TOV1
    STS TIFR1, R20
    LDI R17, 0
    OUT PORTB, R17
    STS TCCR1B, R17
    RETI
```

Processor Status

Name	Value
Program Counter	0x0000003E
Stack Pointer	0x08FD
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	1000046
Frequency	1.000 MHz
Stop Watch	999,947.00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00

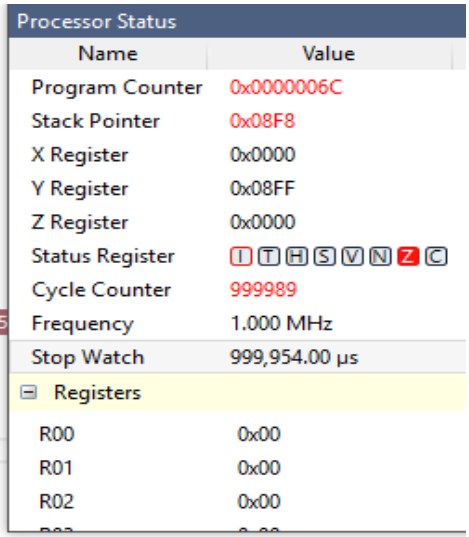
Task 5/E before timer interrupt:

```
    while (1)
    {
        //wait for interrupts
    }
}
ISR (INT0_vect)
{
    EIFR |= (1<<INTF0);      //reset flag
    PORTB |= (1<<2);         //turn on LED
    TCNT1 = 61630;           //set counter value
    TCCR1A = 0;              //normal mode
    TCCR1B = (1<<CS12);      //set prescaler to 25
}

ISR (TIMER1_OVF_vect)
{
    TIFR1 |= (1<<TOV1);      //reset flag
    PORTB &= ~(1<<2);        //turn LED off
    TCCR1B = 0;              //turn off timer
}
```



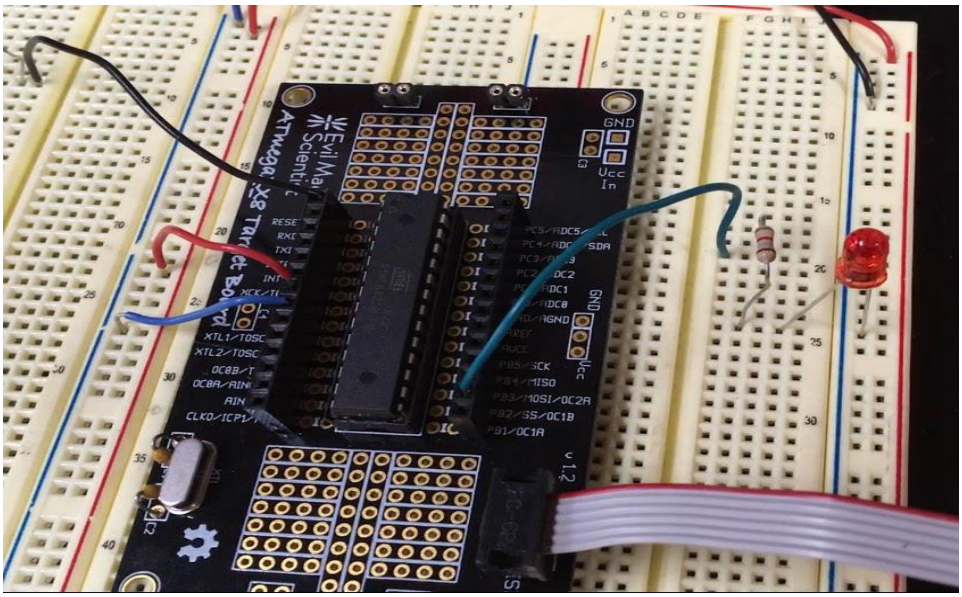
After interrupt:

```
    while (1)
    {
        //wait for interrupts
    }
}
ISR (INT0_vect)
{
    EIFR |= (1<<INTF0);      //reset flag
    PORTB |= (1<<2);         //turn on LED
    TCNT1 = 61630;           //set counter value
    TCCR1A = 0;              //normal mode
    TCCR1B = (1<<CS12);      //set prescaler to 25
}

ISR (TIMER1_OVF_vect)
{
    TIFR1 |= (1<<TOV1);      //reset flag
    PORTB &= ~(1<<2);        //turn LED off
    TCCR1B = 0;              //turn off timer
}
```

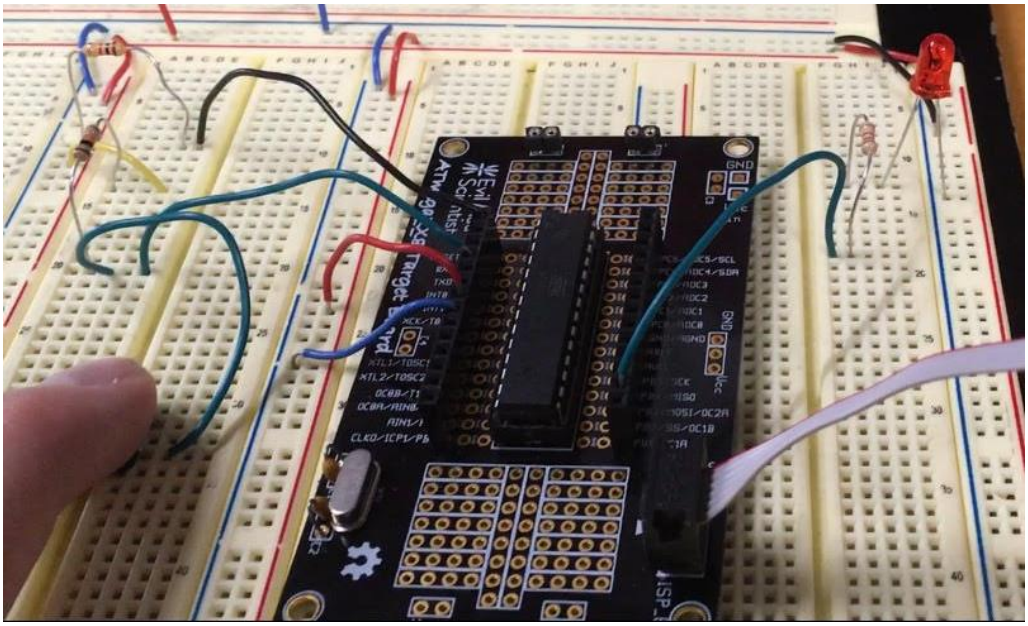


10. SCREENSHOT OF EACH DEMO (BOARD SETUP)

Setup for tasks 1, 3, and 4



Set up for tasks 2 and 5



11. VIDEO LINKS OF EACH DEMO

Tasks 1/3/4 - https://www.youtube.com/watch?v=Q2s7c8BWkkk

Tasks 2/5 - https://www.youtube.com/watch?v=0_fHTzq70SA

12. GITHUB LINK OF THIS DA

https://github.com/nhanuscin/submit/tree/master/DA2

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
Nathan Hanuscin