

# Design Assignment 2

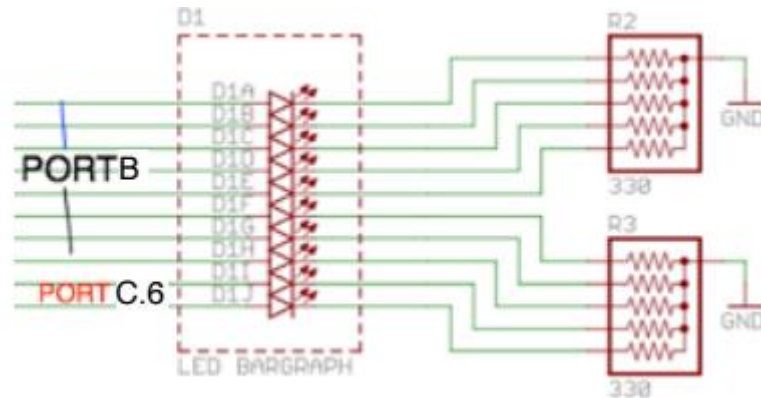
---

**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)
0.	COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS		
1.	INITIAL CODE OF TASK 1 C Code		
2.	INITIAL CODE OF TASK 1 ASM Code		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 2 C Code		
4.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 2 ASM Code		
5.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 3 C Code		
6.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 4 C Code		
7.	SCHEMATICS		
8.	SCREENSHOTS OF EACH TASK OUTPUT		
9.	SCREENSHOT OF EACH DEMO		
10.	VIDEO LINKS OF EACH DEMO		
11.	GOOGLECODE LINK OF THE DA		

0.	COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS		
----	--	--	--



\*Diagram provided via DA2 handout\*

1.	INITIAL CODE OF TASK 1/A C Code		
----	---------------------------------	--	--

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

void sub_delay()
//delay subroutine that creates a 50% duty cycle clock w/ period of 5 seconds
{
    TCNT1=63583;           //sets counter to 63583, which takes 0.25 s to overflow
    TCCR1A=0x00;           //normal mode operation
    TCCR1B=0x05;           //prescaler of 1024
    while((TIFR1&0x01)==0); //loops until TOV1 is set
    TCCR1B=0x00;           //stops the timer
    TIFR1|=(1<<TOV1);       //clear TOV1 flag
    PORTC=PORTC ^ (0x01);   //xor PORTC, PC.0 specifically to toggle a clock
}

int main()
{
    DDRC=0x01;             //PC.4 and 5 were used to output every 5 and 10 rising
                           //pulses due to availability on the board
    PORTC=0x00;             //Clears the PORTC to output LOW signal

    while(1)
    //while loop that will execute forever
        sub_delay();        //Subroutine to cause a delay of 0.25s which will cause
                           //a 5s period for the clock

    return 1;
}
```

2.	INITIAL CODE OF TASK 1/A ASM Code		
----	-----------------------------------	--	--

```
;=====TASK1=====
.dseg
    .def XOR1=R24;

.cseg
.MACRO INITSTACK
    LDI @0,HIGH(@1)
    OUT SPH,@0
    LDI @0,LOW(@1)
```

```

        OUT SPL,@0
.ENDMACRO

        INITSTACK R16, RAMEND           ;use Macro here

        LDI R16,0x01
        SBI DDRC, 0                     ;PC.0 as an output
        LDI R17,0
        OUT PORTC,R17                   ;PORTC = 0

BEGIN:
        RCALL DELAY
        EOR R17,R16                     ;toggle B0 of R17;
        OUT PORTC,R17                  ;toggle PC.0
        RJMP BEGIN

;TIMER1 DELAY
DELAY:
        LDI R20,0xF8
        STS TCNT1H,R20                 ;TCNT1H = 0xF8 timer1 high
        LDI R20,0x5F
        STS TCNT1L,R20                 ;TCNT1L = 0x5F timer1 low
        LDI R20,0x00
        STS TCCR1A,R20                 ;set normal mode
        LDI R20,0x05
        STS TCCR1B,R20                 ;Normal mode, prescaler = 1024

AGAIN:
        IN R20,TIFR1                   ;read TIFR1
        SBRS R20,TOV1                  ;if TOV1 is set skip next instruction
        RJMP AGAIN
        LDI R20,0x00
        STS TCCR1B,R20                 ;logic 00 if fed to the register to stop the timer
        LDI R20,0x01
        STS TCCR1B,R20                 ;stop Timer1
        LDI R20,0x01
        OUT TIFR1,R20                  ;a logic 1 in the TOV1 bit causes a reset/clear
        OUT TIFR1,R20                  ;clear TOV1 flag

        RET

;=====END_TASK1=====

```

3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 2 C Code		
----	--	--	--

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

```

void sub_delay()
//delay subroutine that creates a 50% duty cycle clock w/ period of 5 seconds
{
    TCNT1=63583;           //sets counter to 63583, which takes 0.25 s to overflow
    TCCR1A=0x00;           //normal more operation
    TCCR1B=0x05;           //prescaler of 1024
    while((TIFR1&0x01)==0); //loops until TOV1 is set
    TCCR1B=0x00;           //stops the timer
    TIFR1|=(1<<TOV1);      //clear TOV1 flag
    PORTC=PORTC ^ (0x01);   //xor PORTC, PC.0 specifically to toggle a clock
}

int main(void)
{

```

```

DDRB=0xFF;           //PORTB will be used to output the counter
PORTB=0x00;          //Initialize the out to output LOW signal
DDRC=0x31;           //PC.4 and 5 were used to output every 5 and 10 rising
                      //pulses due to availability on the board
PORTC=0x00;          //Clears the PORTC to output LOW signal
DDRD=0x00;           //PD.4 is set to be used at T0 to take input from the
                      //clock generated

TCNT0=0x00;          //counter 0 that will output to PORTB
TCCR0A=0x00;          //Setting counter to normal mode operation
TCCR0B=0x07;          //sets counter to accept external clock (PD4)

while(1)
//while loop that will execute forever
{
    sub_delay();       //Subroutine to cause a delay of 0.25s which will cause
                      //a 5s period for the clock
    PORTB=TCNT0;       //output counter/timer0 to PORTB (8 bit leds)
}
return 1;
}

```

4.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 2 ASM Code		
----	--	--	--

```

.dseg
.def XOR1=R24;

.cseg
.MACRO INITSTACK
LDI @0,HIGH(@1)
OUT SPH,@0
LDI @0,LOW(@1)
OUT SPL,@0
.ENDMACRO

INITSTACK R16, RAMEND ;use Macro here

LDI XOR1,0x01
SBI DDRC, 0           ;PC.0 as an output
LDI R17,0
OUT PORTC,R17         ;PORTC = 0
LDI R17, 0xFF
OUT DDRB, R17         ;loads R17(0xFF) to DDRB, to set as output
LDI R17, 0x00
OUT PORTB, R17        ;initialize PORTB to 0
OUT DDRD, R17         ;sets DDRD as output for use with PD.4

OUT TCNT0, R17        ;initialize TCNT0 to 0
OUT TCCR0A, R17       ;normal mode operation counter
LDI R17, 0x07
OUT TCCR0B, R17       ;sets counter to accept external clock (PD4)

BEGIN:
RCALL DELAY           ;calls delay subroutine
EOR R17, XOR1         ;toggle B0 of R17;
OUT PORTC,R17         ;toggle PC.0
IN R16, TCNT0         ;loads TCNT0 to R16
OUT PORTB, R16        ;outputs TCNT0 to PORTB

```

```

        RJMP BEGIN

;TIMER1 DELAY
DELAY:
        LDI R20,0xF8
        STS TCNT1H,R20                ;TCNT1H = 0xF8 timer1 high
        LDI R20,0x5F
        STS TCNT1L,R20                ;TCNT1L = 0x5F timer1 low
        LDI R20,0x00
        STS TCCR1A,R20                ;set normal mode
        LDI R20,0x05
        STS TCCR1B,R20                ;Normal mode, prescaler = 1024

AGAIN:
        IN R20,TIFR1                  ;read TIFR1
        SBRS R20,TOV1                 ;if TOV1 is set skip next instruction
        RJMP AGAIN
        LDI R20,0x00                  ;logic 00 if fed to the register to stop the timer
        STS TCCR1B,R20                ;stop Timer1
        LDI R20,0x01                  ;a logic 1 in the TOV1 bit causes a reset/clear
        OUT TIFR1,R20                 ;clear TOV1 flag

        RET

```

5.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 3		
----	---	--	--

```

#include <avr/io.h>

void sub_delay()
//delay subroutine that creates a 50% duty cycle clock w/ period of 5 seconds
{
    TCNT1=63583;                      //sets counter to 63583, which takes 0.25 s to overflow
    TCCR1A=0x00;                      //normal more operation
    TCCR1B=0x05;                      //prescaler of 1024
    while((TIFR1&0x01)==0);           //loops until TOV1 is set
    TCCR1B=0x00;                      //stops the timer
    TIFR1|=(1<<TOV1);                 //clear TOV1 flag
    PORTC=PORTC ^ (0x01);             //xor PORTC, PC.0 specifically to toggle a clock
}

int main(void)
{
    unsigned char toggle1=0;          //toggle counter for 5 rising clock edges
    unsigned char toggle2=0;          //toggle counter for 10 rising clock edges
    DDRB=0xFF;                        //PORTB will be used to output the counter
    PORTB=0x00;                       //Initialize the out to output LOW signal
    DDRC=0x31;                        //PC.4 and 5 were used to output every 5 and 10 rising
    //pulses due to availability on the board
    PORTC=0x00;                       //Clears the PORTC to output LOW signal
    DDRD=0x0C;                        //PD.4 is set to be used at T0 to take input from the
    //clock generated

    TCNT0=0x00;                      //counter 0 that will output to PORTB
    TCCR0A=0x00;                      //Setting counter to normal mode operation
    TCCR0B=0x07;                      //sets counter to accept external clock (PD4)

    while(1)
    //while loop that will execute forever

```

```

{
    sub_delay();           //Subroutine to cause a delay of 0.25s which will cause
                           //a 5s period for the clock

    PORTB=TCNT0;
    if ((PORTC&(0x01))==1)
    //checks if clock rising edge to increment toggle counters
    {
        ++toggle1;
        ++toggle2;
    }
    if (toggle1==5)
    //if toggle1=5, toggle PC4
    {
        PORTC^=(1<<PORTC4);    //xor PORTC with 0x10
        toggle1=0;             //reset toggle1 counter
    }
    if (toggle2==10)
    //if toggle2=10, toggle PC5
    {
        PORTC^=(1<<PORTC5);    //xor PORTC with 0x20
        toggle2=0;             //clear toggle2
    }
}
return 1;
}

```

6.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 4		
----	---	--	--

```

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

void sub_delay()
//delay subroutine that creates a 50% duty cycle clock w/ period of 5 seconds
{
    TCNT1=63583;           //sets counter to 63583, which takes 0.25 s to overflow
    TCCR1A=0x00;           //normal mode operation
    TCCR1B=0x05;           //prescaler of 1024
    while((TIFR1&0x01)==0); //loops until TOV1 is set
    TCCR1B=0x00;           //stops the timer
    TIFR1|=(1<<TOV1);       //clear TOV1 flag
    PORTC=PORTC ^ (0x01);   //xor PORTC, PC.0 specifically to toggle a clock
}

ISR (INT0_vect)
//interrupt routine that toggles PC.4 every 5 rising clock edges
{
    PORTC^=(1<<PORTC4);
}

ISR (INT1_vect)
//interrupt routine that toggles PC.5 every 10 rising clock edges
{
    PORTC^=(1<<PORTC5);
}

int main(void)

```

```

{
    unsigned char toggle1=0;    //toggle counter for 5 rising clock edges
    unsigned char toggle2=0;    //toggle counter for 10 rising clock edges
    DDRB=0xFF;                  //PORTB will be used to output the counter
    PORTB=0x00;                  //Initialize the out to output LOW signal
    DDRC=0x31;                  //PC.4 and 5 were used to output every 5 and 10 rising
                                //pulses due to availability on the board
    PORTC=0x00;                  //Clears the PORTC to output LOW signal
    DDRD=0x0C;                  //PD.4 is set to be used at T0 to take input from the
                                //clock generated

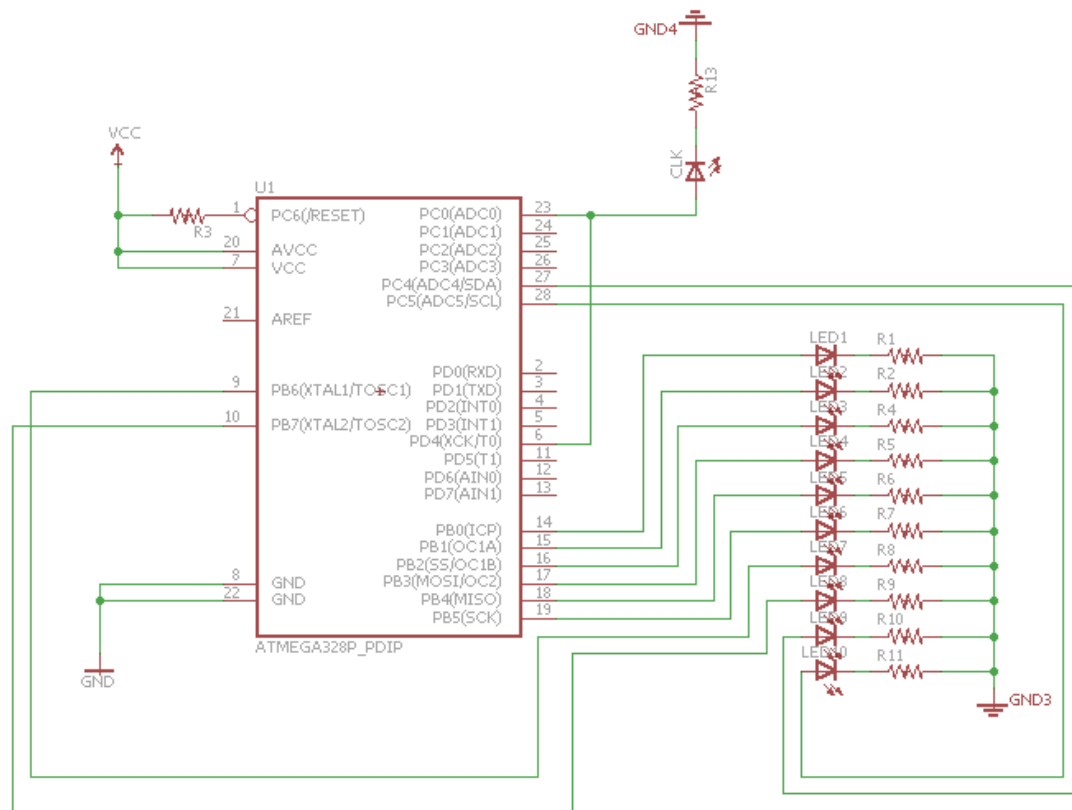
    TCNT0=0x00;                  //counter 0 that will output to PORTB
    TCCR0A=0x00;                  //Setting counter to normal mode operation
    TCCR0B=0x07;                  //sets counter to accept external clock (PD4)

    //This set of instructions sets the interrupt registers
    EICRA=0x0F;                  //sets interrupts INT0 and INT1 to trigger on rising
                                //edge
    EIMSK=0x03;                  //enables INT0 and INT1 to be used as outputs
    sei();                        //enables all global interrupts

    while(1)
    //while loop that will execute forever
    {
        sub_delay();              //Subroutine to cause a delay of 0.25s which will cause
                                //a 5s period for the clock

        PORTB=TCNT0;
        if ((PORTC&(0x01))==1)
        //checks if clock rising edge to increment toggle counters
        {
            ++toggle1;
            ++toggle2;
        }
        if (toggle1==5)
        //if toggle1=5, output 1 to PD.2 to trigger INT0
        {
            PORTD^=(1<<PORTD2); //set PD.2
            toggle1=0;           //reset toggle1
            PORTD^=(1<<PORTD2); //clear PD.2
        }
        if (toggle2==10)
        //if toggle2=10, output 1 to PD.3 to trigger INT0
        {
            PORTD^=(1<<PORTD3); //set PD.3
            toggle2=0;           //reset toggle2
            PORTD^=(1<<PORTD3); //clear PD.3
        }
    }
    return 1;
}

```





8.	SCREENSHOTS OF EACH TASK OUTPUT		
----	---------------------------------	--	--

TASK 1/A:

Verify duty cycle and period: 50% duty cycle, period = 0.5 second

C code

The screenshot shows the AVR Studio IDE. On the left, the 'C\_Code' window displays the source code for 'main.c'. The function 'sub\_delay' is defined, which sets TCNT1 to 63583, configures TCCR1A and TCCR1B, and enters a while loop that toggles PORTC until TIFR1 is set. The line 'PORTC=PORTC ^ (0x01);' is highlighted. On the right, the 'I/O View' window shows the state of various hardware registers. The 'Stop Watch' is highlighted in yellow, showing a value of 249.99 ms. Other visible values include Program Counter (0x00000000), Stack Pointer (0x08FB), X Register (0x0000), Y Register (0x08FF), Z Register (0x0000), Status Register (0x00000000), Cycle Counter (1999908), and Frequency (8.000 MHz).

```

C_Code main.c
sub_delay
    TCNT1=63583; //set
    TCCR1A=0x00; //no
    TCCR1B=0x05; //pre
    while((TIFR1&0x01)==0); //lo
    TCCR1B=0x00; //sto
    TIFR1|=(1<<TOV1); //cle
    PORTC=PORTC ^ (0x01); //xor
}
  
```

Program Counter	0x00000000
Stack Pointer	0x08FB
X Register	0x0000
Y Register	0x08FF
Z Register	0x0000
Status Register	0x00000000
Cycle Counter	1999908
Frequency	8.000 MHz
Stop Watch	249.99 ms
Registers	
R00	0x00

ASM code

The screenshot shows the AVR Studio IDE. On the left, the 'ASM Code' window displays the assembly code for the delay function. The code includes instructions to load R16 with 0x01, set DDRC to 0, load R17 with 0, and output R17 to PORTC. A loop labeled 'BEGIN' calls the 'DELAY' macro, toggles R17 with 'EOR R17,R16', and jumps back to 'BEGIN'. The line 'EOR R17,R16' is highlighted. On the right, the 'I/O View' window shows the state of various hardware registers. The 'Stop Watch' is highlighted in yellow, showing a value of 249.99 ms. Other visible values include X Register (0x0000), Y Register (0x0000), Z Register (0x0000), Status Register (0x00000000), Cycle Counter (1999907), and Frequency (8.000 MHz).

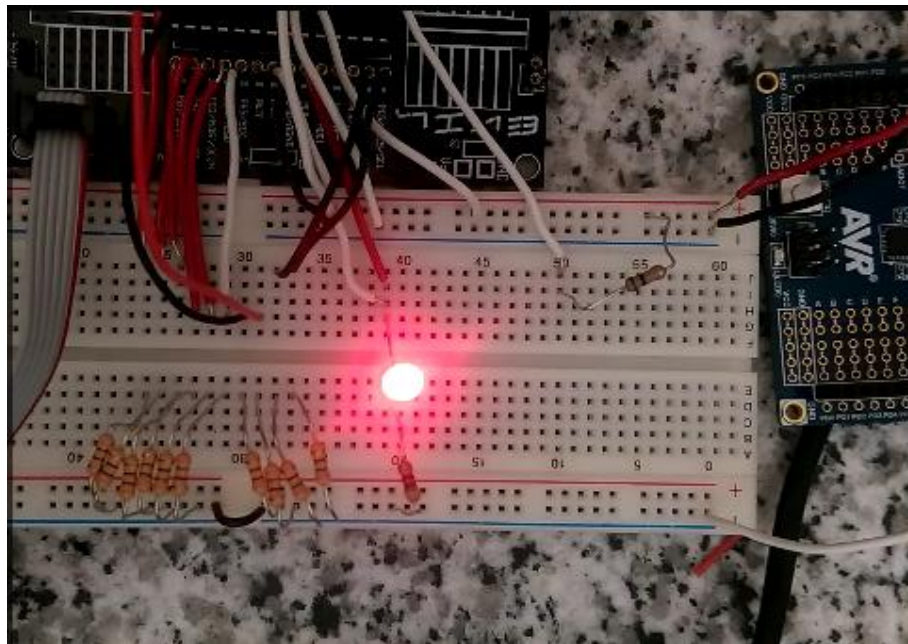
```

LDI R16,0x01
SBI DDRC, 0 ;PC.0 as
LDI R17,0
OUT PORTC,R17 ;PORTC =
BEGIN:
    RCALL DELAY
    EOR R17,R16 ;toggle B
    OUT PORTC,R17 ;toggle P
    RJMP BEGIN
  
```

X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	0x00000000
Cycle Counter	1999907
Frequency	8.000 MHz
Stop Watch	249.99 ms
Registers	
R00	0x00

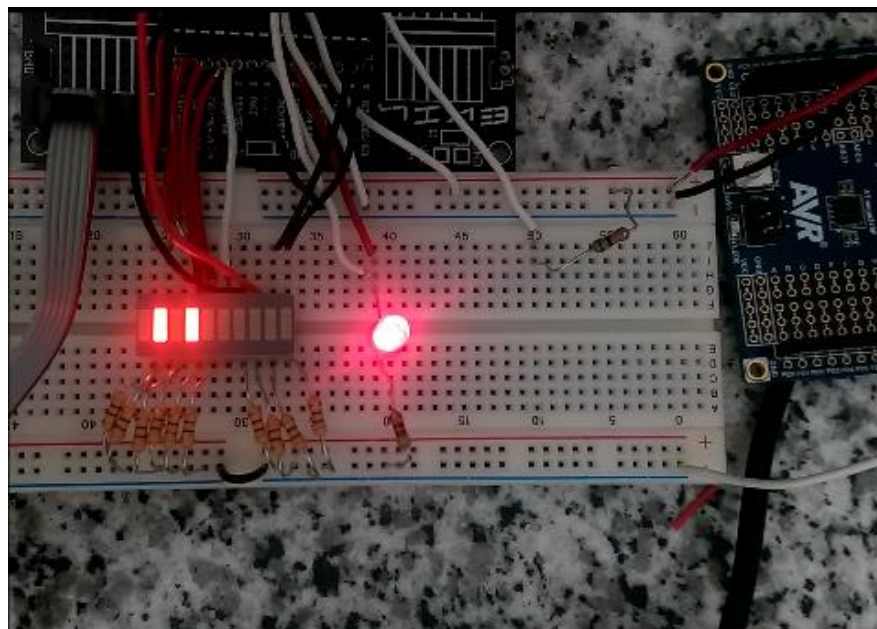
9.	SCREENSHOT OF EACH DEMO		
----	-------------------------	--	--

TASK 1/A: LED blinks every 5 seconds (duty cycle of 50%)



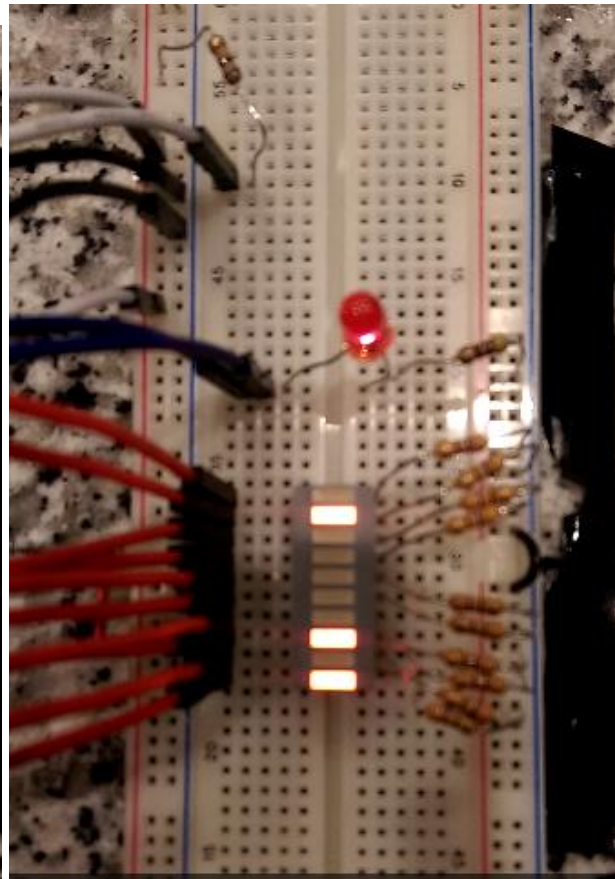
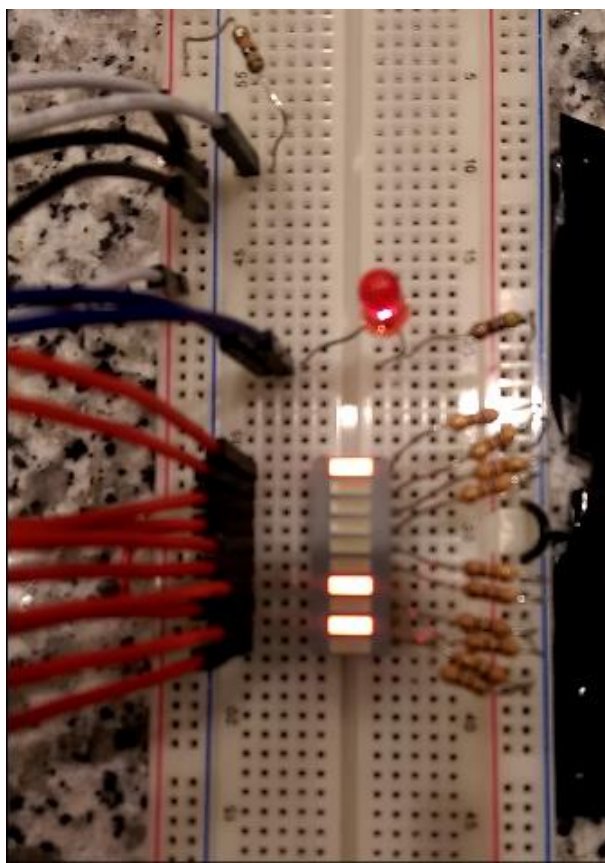
The LED blinks and shows the clock pulses (period 5 seconds)

TASK 2/B: LED blinks every 5 seconds (duty cycle of 50%) and counter counts on rising edge of clock

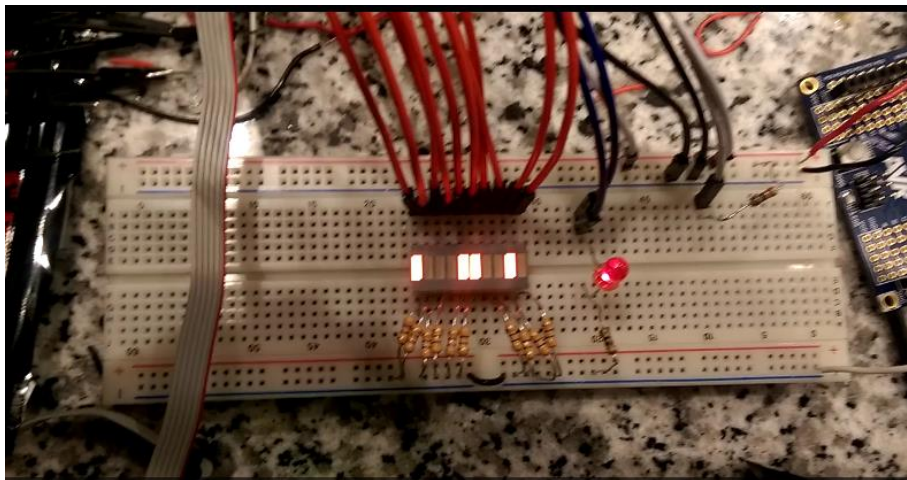


The clock output was also wired to T0 to clock the counter to counter on the rising edge of the clock generated.

TASK 3/C: LED blinks every 5 seconds (duty cycle of 50%) and counter counts on rising edge of clock with toggling of the 9<sup>th</sup> and 10<sup>th</sup> bit on every 5 and 10 rising clock pulses, respectively.



TASK 4/D: LED blinks every 5 seconds (duty cycle of 50%) and counter counts on rising edge of clock with toggling of the 9<sup>th</sup> and 10<sup>th</sup> bit on every 5 and 10 rising clock pulses, respectively.



10.	VIDEO LINKS OF EACH DEMO		
<a href="https://www.youtube.com/channel/UCpl-ILLVfaKxzVCt4e2VNoQ">https://www.youtube.com/channel/UCpl-ILLVfaKxzVCt4e2VNoQ</a>			
11.	GOOGLECODE LINK OF THE DA		
<a href="https://github.com/nhand2/CPE301S16">https://github.com/nhand2/CPE301S16</a>			

**Student Academic Misconduct Policy**

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

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

Derek Nhan