ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ (MICROLAB)

3η Εργαστηριακη Αναφορα στο μαθημα "**ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ**" του 7ου Εξαμηνου

των φοιτητών της **ομάδας 17**,

Εμμανουήλ Αναστάσιου Σερλή, Α.Μ. 03118125
Ευάγγελου Περσόπουλου, Α.Μ.: 03121701

1η Άσκηση: -> micro_lab03_ex01.asm και micro_lab03_ex01.c

Το ζητούμενο πρόγραμμα ελέγχει το άναμμα και το σβήσιμο ενός φωτιστικού, με τρόπο παρόμοιο με αυτόν της άσκησης 2.3 του προηγούμενου εργαστηρίου. Η διαφορά έγκειται στην χρήση χρονιστή, ο οποίος αρχικοποιείται κάθε φορά που απαιτείται σβήσιμο των leds μετά από 4sec στην αντίστοιχη ρουτίνα εξυπηρέτησης.

Assembly Program:

```
.include "m328PBdef.inc" ;checked
.equ FOSC_MHZ=16
.def counter=r19
.org 0x0
rjmp reset
.org 0x4
rjmp ISR1
.org 0x1A
rjmp ISR_TIMER1_OVF


reset:
    ;init stack pointer
    ldi r24, LOW(RAMEND)
    out SPL, r24
    ldi r24, HIGH(RAMEND)
    out SPH, r24

    ;interrupt on the rising edge of INT1 pin
    ldi r24, (1<<ISC11) | (1 << ISC10)
    sts EICRA, r24

    ;Enable INT1
    ldi r24, (1<<INT1)
    out EIMSK, r24

    ldi r24, (1<<CS12) | (0<<CS11) | (1<<CS10)    ;prescale=1024
    sts TCCR1B, r24


    sei


IO_set:
    ;PORTB as output
    ser r24
    out DDRB, r24

    ;PORTC as input
    clr r24
```

```
    out DDRC, r24

    clr counter
main:
    in r24, PINC
    cpi r24, 0x5F
    brne main
    ldi r24, LOW(16*200)
    ldi r25, HIGH(16*200)
    rcall wait_x_msec ;sparkle effect
    cpi counter, 0x00
    breq pc5_routine
    ser r24
    out PORTB, r24  ;leds on
    ldi r24, LOW(16*500)
    ldi r25, HIGH(16*500)
    rcall wait_x_msec


pc5_routine:

    ldi r24, 0x01
    out PORTB, r24

    ldi r24, HIGH(3035)        ;4 sec timer
    sts TCNT1H, r24
    ldi r24, LOW(3035)
    sts TCNT1L, r24
    ;enable TCNT1 of TIME/COUNTER1
    ldi r24, (1<<TOIE1)
    sts TIMSK1, r24
    inc counter
    rjmp main


ISR_TIMER1_OVF:

    clr r21
    out PORTB, r21
    clr counter
    ldi r18, (0<<TOIE1)
    sts TCNT1L, r18
    reti

ISR1:
    cli
    push r25
    push r24
    in r24, SREG
    push r24
check:
    ldi r24, (1 << INTF1) ;includes sparkle effect
```

```
    out EIFR, r24         ;at the start of int. routine
    ldi r24, low(16*5)
    ldi r25, high(16*5)
    rcall wait_x_msec ;5ms delay before interrupt routine
    in r24, EIFR
    cpi r24, 2
    breq check

    ;rcall wait_x_msec     ;sparkle effect
    ldi r24, HIGH(3035)        ;4 sec timer
    sts TCNT1H, r24
    ldi r24, LOW(3035)
    sts TCNT1L, r24
    ;enable TCNT1 of TIME/COUNTER1
    ldi r24, (1<<TOIE1)
    sts TIMSK1, r24
    ldi r24, 0x01
    out PORTB, r24
    ;inc counter
    cpi counter, 0x00
    breq telos_int1

    ldi r24, 0xFF
    out PORTB, r24
    ldi r24, LOW(16*500)
    ldi r25, HIGH(16*500)
    rcall wait_x_msec
    clr r24
    sts TCNT1H, r24
    sts TCNT1L, r24

    ldi r24, HIGH(3035)        ;4 sec timer
    sts TCNT1H, r24
    ldi r24, LOW(3035)
    sts TCNT1L, r24
    ldi r24, 0x01
    out PORTB, r24

telos_int1:
    inc counter
    pop r24
    out SREG, r24
    pop r24
    pop r25
    sei
    reti


;produces x sec delay
wait_x_msec:
    sbiw r24 , 1
    breq telos      ;check if r24 = 1
```

```
loop1:
    rcall wait_1msec
    rcall wait4
    nop
    sbiw r24 , 1
    brne loop1 ;1 or 2 cycles
    nop ;1 cycle
    nop
telos:
    rcall wait_1msec
    nop
    ret ;3 cycles

wait_1msec:               ;produce 988usec delay
    ldi r26 , 97
loop2:
    rcall wait4     ;4+3=7 cycles
    dec r26     ;1 cycle
    brne loop2
    rcall wait4
    nop
    nop
    nop
    nop
    ret    ;4 cycles

 wait4:    ret
```

C Program:

```
#define F_CPU 16000000UL //checked
#include "avr/io.h"
#include<avr/interrupt.h>
#include<util/delay.h>

unsigned int counter=0;


ISR(INT1_vect)
{
   //_delay_ms(50);
   TCNT1= 3035;
   PORTB = 0x01;
   counter++;
   if (counter > 1)
   {
       PORTB = 0xFF;
       _delay_ms(500);
       TCNT1 = 3035;
       PORTB = 0x01;
```

```
   }
}

ISR(TIMER1_OVF_vect)
{
    PORTB = 0;
    counter = 0;
}

void pc5_routine()
{
    PORTB = 0x01;
    _delay_ms(200); //for sparkle effect
    TCNT1 = 3035;
    counter++;
}

void pc5_again()
{
    PORTB = 0xFF;
    _delay_ms(500);
    PORTB = 0x01;
    TCNT1 = 3035;
}

int main()
{
    EICRA = (1<<ISC11) | (1<<ISC10); //INT1-PD3
    EIMSK = (1 << INT1);
    TCCR1B = (1<<CS12) | (0<<CS11) | (1<<CS10);  //prescale 1024
    TIMSK1 = (1<<TOIE1);
    sei(); // enable all interrupts
    DDRC = 0x00;
    DDRB = 0xFF;
    unsigned int pc5=0;
    counter = 0;
    while(1)
    {
        pc5 = PINC;
        //asm("NOP");
        if (pc5 == 0x5F)
        //if(pc5 == 0x20) //only for simulator
        {
            pc5_routine();
            if (counter > 1)
            {
              pc5_again();
            }


        }
```

```
    }
    return 0;
}
```

2η Άσκηση: -> micro_lab03_ex02.asm και micro_lab03_ex02.c

Το ζητούμενο πρόγραμμα καλείται να ρυθμίσει την φωτεινότητα του PB1 μέσω του duty cycle της αντίστοιχης PWM κυματομορφής. Συγκεκριμένα, η μέγιστη τιμή που μπορεί να λάβει ο καταχωρητής OCR1A είναι 255 (με prescaler 8), η οποία αντιστοιχεί στην μέγιστη φωτεινότητα του LED. Έτσι, υπολογίστηκαν και αποθηκεύτηκαν σε πίνακα όλες οι υπόλοιπες τιμές (από 5 ως 250), οι οποίες απαιτούνται στις περιπτώσεις αυξομείωσής της φωτεινότητας.

Assembly Program:

```
.include "m328PBdef.inc" ;checked
.def counter=r19
.def duty=r20
.def temp=r18
.equ FREQ=16 ;operating freq

reset:
    ;init stack pointer
    ldi r24, LOW(RAMEND)
    out SPL, r24
    ldi r24, HIGH(RAMEND)
    out SPH, r24

    ;fast PWM mode and prescaler=8
    ldi r25 ,(1<<WGM10) | (1<<COM1A1)
    sts TCCR1A, r25

    ldi r25 ,(1<<WGM12) | (1<<CS11)
    sts TCCR1B, r25

    sei ;enable all interrupts

    ;set PB1 as output
    ldi r27, 0x02
    out DDRB, r27

    nop

    ldi counter, 4 ;initialize variables and d.c.
    rcall load_dc

    nop
```

```
main:
    in temp, PIND
    cpi temp, 0xFD ;check for PD1 ;possibly needs reverse logic value
    breq incr
    cpi temp, 0xFB ;check for PD2 ;possibly needs reverse logic value
    breq decr

    rjmp main

incr:
    cpi counter, 12 ;check if index exceeds 12
    breq incr_sp
    inc counter
    rcall load_dc ;load dc value
    rjmp main

incr_sp:
    ldi counter,12
    rcall load_dc ;load dc value
    rjmp main

decr:
    cpi counter, 0 ;check if index exceeds 0
    breq decr_sp
    dec counter
    rcall load_dc ;load dc value
    rjmp main

decr_sp:
    ldi counter,0
    rcall load_dc ;load dc value
    rjmp main

load_dc:
    ldi Zh, HIGH(Table*2) ;multiply by 2 for byte access
    ldi Zl, LOW(Table*2)
    clr r18
    add zl, counter ;access table value by index
    adc zh, r18
    lpm ; rz to r0
    mov r22,r0
    sts OCR1AL,r22
    sts OCR1AH,r18
    ldi r24, low(FREQ*200) ;then wait for 50ms
    ldi r25, high(FREQ*200)
    rcall delay_mS ;delay 50ms
    ret

delay_mS: ;create ms delay
    ldi r23,249
loop_inn:
```

```
    dec r23
    nop
    brne loop_inn

    sbiw r24,1
    brne delay_mS

    ret

;c values -> {5,25,46,66,86,107,127,148,168,189,209,229,250}
Table:
.DW 0x1905,0x422E,0x6B56,0x947F,0xBDA8,0xE5D1,0xFAFA
```

C Program:

```
#define F_CPU 16000000UL //checked
#include "avr/io.h"
#include <util/delay.h>
#include<avr/interrupt.h>

const int duty_arr[]={5,25,46,66,86,107,127,148,168,189,209,229,250};

int main() {
    unsigned char duty;
    unsigned int temp;
    unsigned int counter;

    //fast PWM mode and prescaler at 8
    TCCR1A = (1<<WGM10) | (1<<COM1A1);
    TCCR1B = (1<<WGM12) | (1<<CS11);

    sei(); // enable all interrupts

    DDRB |=0b00000010; //PB1 as output
    DDRD |=0b00000000;
    //PWM_init();
    duty=duty_arr[6];
    counter=6; //initialize counter
    OCR1A=duty;
    while(1){
        temp=PIND;
        if(temp==0xFB){
            counter++;
            if(counter<13){
                duty=duty_arr[counter];
                OCR1A=duty;
                _delay_ms(200);
            }
            else{
```

```
                counter=12;
                duty=duty_arr[counter];
                OCR1A=duty;
                _delay_ms(200);
            }
        }
        if(temp==0xFD){
            if (counter == 0){
                counter=0;
                duty=duty_arr[counter];
                OCR1A=duty;
                _delay_ms(200);
            }
            if(counter>0){
                counter--;
                duty=duty_arr[counter];
                OCR1A=duty;
                _delay_ms(200);
            }
        }
    }
}
```

<u>3η Άσκηση:</u> -> micro_lab03_ex03.asm και micro_lab03_ex03.c

Το ζητούμενο πρόγραμμα καλείται να παράξει μία κυματομορφή PWM με έξοδο το PB1 και duty cycle 50%. Στην συνέχεια, ανάλογα με το πατημένο πλήκτρο του PORTD, αλλάζει η συχνότητα της κυματομορφής σύμφωνα με τον δοθέντα πίνακα. Οι τιμές του καταχωρητή ICR1(=TOP) υπολογίστηκαν μέσω της σχέσης:

$$TOP = \frac{f_{clk}}{N * f_{pwm}} - 1$$

, όπου N=8 η τιμή του prescaler

$f_{clk} = 16MHZ$ η ονομαστική συχνότητα λειτουργίας του μικροελεγκτή

f_{pwm} η ζητούμενη τιμή συχνότητας της κυματομορφής

Έτσι, λάβαμε τον κάτωθι πίνακα:

ΠΛΗΚΤΡΟ	f_{pwm}(kHz)	TOP
PD.0	125	15999
PD.1	250	7999
PD.2	500	3999
PD.3	1000	1999

Assembly Program:

```
.include "m328PBdef.inc" ;checked
.def freql=r19
.def freqh=r20
.def duty=r17
.def temp=r18
.equ FREQS=16 ;operating freq
.equ HALF_DUTY=128

reset:
    ;init stack pointer
    ldi r24, LOW(RAMEND)
    out SPL, r24
    ldi r24, HIGH(RAMEND)
    out SPH, r24

    ;fast PWM mode and prescaler=8
    ldi r25 ,(1<<WGM11)| (0<<WGM10) | (1<<COM1A1)
    sts TCCR1A, r25

    ldi r26 ,(1<<WGM12) | (1<<CS11) | (1<<WGM13)
    sts TCCR1B, r26

    sei ;enable all interrupts

    ;set PB1 as output
    ldi r27, 0x02
    out DDRB, r27


    clr duty ;initialize d.c. at 50%
    clr r18
    rcall load_dc

    nop

main:
    in temp, PIND
    cpi temp, 0xFF ;check pushed PDi and change freq accordingly
    breq freq0
```

```
    cpi temp, 0xFE ;possibly needs reverse logic value
    breq freq1
    cpi temp, 0xFD
    breq freq2
    cpi temp, 0xFB
    breq freq3
    cpi temp, 0xF7
    breq freq4
    rjmp main

freq0:
    clr duty ;initialize d.c. at 50%
    clr r18
    rcall load_dc
    rjmp main
freq1:
    ldi duty, HALF_DUTY ;initialize d.c. at 50%
    clr r18
    rcall load_dc
    ldi freql,0x7F
    ldi freqh,0x3E
    rcall load_freq
    rjmp main
freq2:
    ldi duty, HALF_DUTY ;initialize d.c. at 50%
    clr r18
    rcall load_dc
    ldi freql,0x3F
    ldi freqh,0x1F
    rcall load_freq
    rjmp main
freq3:
    ldi duty, HALF_DUTY ;initialize d.c. at 50%
    clr r18
    rcall load_dc
    ldi freql,0x9F
    ldi freqh,0xF
    rcall load_freq
    rjmp main
freq4:
    ldi duty, HALF_DUTY ;initialize d.c. at 50%
    clr r18
    rcall load_dc
    ldi freql,0xCF
    ldi freqh,0x7
    rcall load_freq
    rjmp main


load_dc:
    sts OCR1AH, r18
    sts OCR1AL,duty
```

```
    ret
load_freq:
    sts ICR1H,freqh
    sts ICR1L,freql
    ret

delay_mS: ;create ms delay
    ldi r23,249
loop_inn:
    dec r23
    nop
    brne loop_inn

    sbiw r24,1
    brne delay_mS

    ret
```

C Program:

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

unsigned char duty;
unsigned int top;
unsigned int temp;

const int icr1_arr[]={15999,7999,3999,1999}; //output value on top



int main(){

    //fast PWM mode and prescaler at 8
    TCCR1A = (1<<WGM11)| (0<<WGM10) | (1<<COM1A1);
    TCCR1B = (1<<WGM12) | (1<<CS11) | (1<<WGM13);

    sei(); // enable all interrupts

    duty=128; //50% duty cycle
    DDRB |=0b00000010; //PB1 as output
    temp=0;
    OCR1AL=duty;
    asm("NOP");
    while(1){
        asm("NOP");
        temp=PIND;
        if(temp==0xFE){ //cases
            OCR1A=duty;
```

```
            //_delay_ms(10);
            ICR1=icr1_arr[0];
        }
        else if(temp==0xFD){
            OCR1A=duty;
            //_delay_ms(10);
            ICR1=icr1_arr[1];
        }
        else if(temp==0xFB){
            OCR1A=duty;
            //_delay_ms(10);
            ICR1=icr1_arr[2];
        }
        else if(temp==0xF7){
            OCR1A=duty;
            //_delay_ms(10);
            ICR1=icr1_arr[3];
        }
        else if (temp==0xFF){
            OCR1A=0;
            //_delay_ms(10);
        }
}
}
```