3ο ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"

Δρυς-Πεντζάκης Οδυσσεύς el19192
Μαρκαντωνάτος Γεράσιμος el19149

3.1

Τα προγράμματά όταν πατηθεί είτε το PC5 είτε το PD3 στην αρχή ελέγχουν αν το led είναι ανοιχτό και στην περίπτωση που δεν είναι φορτώνουν στον καταχωρητή TCNT1 την κατάλληλη τιμή (3035) και ανάβουν το led, ώστε μετά από 4 δευτερόλεπτα να υπερχειλίσει και και να γίνει διακοπή στον μικροελεγκτή και στην εξυπηρέτηση της διακοπής να σβήσει. Αν είναι ήδη ανοιχτοι τότε φορτώνουν στον καταχωρητή TCNT1 την τιμή 57722 και ανάβουν όλα τα led ώστε μετά από 0.5 δευτερόλεπτα να σβήσουν στην ρουτίνα εξυπηρέτησης της υπερχείλισης και μετά φορτώνουν στον καταχωρητή TCNT1 την τιμή 10847 για να μείνει ανοιχτό το led για 3.5 δευτερόλεπτα.

Assembly

```
.include "m328PBdef.inc"

.equ FOSC_MHZ=16

.equ DEL_mS=500

.equ DEL_NU= FOSC_MHZ*DEL_mS ;

.org 0x0
rjmp reset
.org 0x4
rjmp ISR1
.org 0x1A
rjmp ISR_TIMER1_OVF

reset:
   ldi r23,(1<< ISC11)|(1<<ISC10)
   sts EICRA, r23
```

```
    ldi r23, (1<<INT1)
    out EIMSK, r23
    sei

    ldi r24, (1<<TOIE1) ; ενεργοποίηση διακοπής υπερχείλισης του μετρητή TCNT1
    sts TIMSK1, r24 ; για τον timer1
    ser r26
    out DDRB, r26
    clr r26
    out DDRC, r26
    ldi r24, LOW(RAMEND)
    out SPL, r24
    ldi r24, HIGH (RAMEND)
    out SPH, r24
    clr r16
    out PORTB,r16
    ldi r24 ,(1<<CS12) | (0<<CS11) | (1<<CS10) ; CK/1024
    sts TCCR1B, r24

check:
    in r17, PINC
    com r17
    andi r17, 0x20
    cpi r17, 0x00
    breq check
delay:
    ldi r24, low(5*16)
    ldi r25, high(5*16)
    rcall delay_mS
    in r17, PINC
    com r17
    andi r17, 0x20
    cpi r17, 0x00
    brne delay

    in r18,PORTB
    cpi r18, 0x01
    brne first

    ldi r24, HIGH(57722)
    sts TCNT1H, r24
    ldi r24, LOW(57722)
    sts TCNT1L, r24
    ser r20
    out PORTB, r20
wait1:
    in r17,PORTB
    cpi r17, 0x00
```

```
    brne wait1

    ldi r24, HIGH(10847)
    sts TCNT1H, r24
    ldi r24, LOW(10847)
    sts TCNT1L, r24
    ldi r20, 0x01
    out PORTB, r20
    rjmp check

first:
    ldi r24, HIGH(3035) ;
    sts TCNT1H, r24 ;
    ldi r24, LOW(3035)
    sts TCNT1L, r24
    ldi r20, 0x01
    out PORTB, r20
    rjmp check

ISR1:
    push r23
    push r24
    in r24, SREG
    push r24

start:
    ldi r16, (1 << INTF1)
    out EIFR, r16
    ldi r24, low(5*16)
    ldi r25, high(5*16)
    rcall delay_mS
    in r16, EIFR
    cpi r16,0
    brne start
    sei
    in r17,PORTB
    cpi r17, 0x01
    breq all_lights

lights_on:
    ldi r24, HIGH(3035)
    sts TCNT1H, r24
    ldi r24, LOW(3035)
    sts TCNT1L, r24
    ldi r20, 0x01
    out PORTB, r20
    pop r23
    pop r24
```

```
    out SREG, r24
    pop r24
    reti

all_lights:
    ldi r24, HIGH(57722)
    sts TCNT1H, r24
    ldi r24, LOW(57722)
    sts TCNT1L, r24
    ser r20
    out PORTB, r20
wait2:
    in r17,PORTB
    cpi r17, 0x00
    brne wait2

    ldi r24, HIGH(10847)
    sts TCNT1H, r24
    ldi r24, LOW(10847)
    sts TCNT1L, r24
    ldi r20, 0x01
    out PORTB, r20
    pop r23
    pop r24
    out SREG, r24
    pop r24
    reti

ISR_TIMER1_OVF:
    clr r20
    out PORTB, r20
    reti


delay_mS:

    ldi r23, 249
loop_inn:
    dec r23
    nop
    brne loop_inn

    sbiw r24, 1
    brne delay_mS

    ret
```

C

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

ISR (INT1_vect){
    EIFR= (1 << INTF1);
    _delay_ms(5);
    while (EIFR==1) {
        EIFR= (1 << INTF1);
    _delay_ms(5);
    }

    sei();
     if (PORTB==0x01){
        TCNT1=57722;
        PORTB=0xFF;
        while(PORTB==0xFF);
        TCNT1=10847;
        PORTB=0x01;
        return;
    }
    else{
        TCNT1=3035;
        PORTB=0x01;
        return;
        }

}

ISR (TIMER1_OVF_vect){
    PORTB=0x00;

}


int main(void) {
    DDRB=0xFF;
    EICRA=(1<< ISC11)|(1<< ISC10);
    EIMSK=(1<<INT1);
    TIMSK1=(1<<TOIE1);
```

```
    TCCR1B=(1<<CS12) | (0<<CS11) | (1<<CS10);
    sei();
    PORTB=0x00;

   while(1){

       if(PINC==0x5F){
        while(PINC!=0x7F)_delay_ms(5);
        if (PORTB==0x01){
            TCNT1=57722;
            PORTB=0xFF;
            while (PORTB==0xFF);
            TCNT1=10847;
            PORTB=0x01; }
        else {
            TCNT1=3035;
            PORTB=0x01;
        }
        }


   }

}
```

3.2

Βάζουμε τις 13 τιμές που μπορεί να πάρει ο καταχωρητής OCR1A για τις διάφορες τιμές του Duty Cycle σε πίνακα και ανάλογα αν αυξάνουμε ή μειώνουμε το Duty Cycle φορτώνουμε την επόμενη ή την προηγούμενη τιμή του πίνακα στον OCR1A, ελέγχοντας βέβαια αν έχουμε φτάσει στις ακριανές τιμές του πίνακα, στην οποία περίπτωση δεν αλλάζει η τιμή του καταχωρητή OCR1A και το Duty Cycle μένει σταθερό.

Assembly

```
.include "m328PBdef.inc"

ldi r24 ,(1<<WGM10) | (1<<COM1A1)
sts TCCR1A, r24
```

```
ldi r24 ,(1<<WGM12) | (1<<CS11)
sts TCCR1B, r24
ser r24
out DDRB,r24
ldi ZH,HIGH(Array *2)
ldi ZL,LOW(Array *2)
adiw zl, 6
lpm
mov r18,r0
ldi r21,0x00
sts OCR1AH,r21
sts OCR1AL,r18
loop:
   in r17, PIND
   com r17
   cpi r17,0x00
   breq loop
delay:
   ldi r24, low(5*16)
   ldi r25, high(5*16)
   rcall delay_mS
   in r20, PIND
   com r20
   andi r20,0x06
   cpi r20, 0x00
   brne delay

   mov r16,r17
   andi r17, 0x02
   cpi r17, 0x02
   breq raise
   andi r16, 0x04
   cpi r16, 0x04
   breq lower
   rjmp loop

raise:
   cpi r18,0xFB
   breq loop
   sbiw zl,1
   lpm
   mov r18,r0
   ldi r21,0x00
   sts OCR1AH,r21
   sts OCR1AL,r18
   rjmp loop

lower:
```

```
    cpi r18,0x1A
    breq loop
    adiw zl,1
    lpm
    mov r18,r0
    ldi r21,0x00
    sts OCR1AH,r21
    sts OCR1AL,r18
    rjmp loop



Array:
.DW 0xE6FB, 0xBDD2, 0x94A9, 0x6C80
.DW 0x4357, 0x332E, 0x001A

delay_mS:

    ldi r23, 249
loop_inn:
    dec r23
    nop
    brne loop_inn

    sbiw r24, 1
    brne delay_mS

    ret
```

C

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

int main(void) {
    TCCR1A = (1<<WGM10) | (1<<COM1A1) ;
    TCCR1B = (1<<WGM12) | (1<<CS11) ;
    DDRB=0b00111111;
    int data[] = { 0xFB,0xE6,0xD2, 0xBD,0xA9,0x94,0x80, 0x6C,0x57,0x43,0x33 ,0x2E,0x1A
};
    int i=6;
    OCR1A = data[i];
    while(1) {
```

```
        if(PIND==0b11111101){
            while(PIND==0b11111101)_delay_ms(5);
            if (i==0);
            else {
                i--;
                OCR1A = data[i];
            }
        }

        else if (PIND==0b11111011){
            while(PIND==0b11111011)_delay_ms(5);
            if (i==12);
            else {
                i++;
                OCR1A = data[i];
            }
        }

}
}
```

3.3

TIM1A ώστε να λειτουργεί σε fast PWM mode και υπολογίζουμε για prescale 8 και συχνότητα ρολογιού 16 MHz τις τιμές που θα πρέπει να έχει το TOP με τον τύπο που μας δίνεται για να έχει τις επιθυμητές συχνότητες η κυματομορφή εξόδου. Κάθε φορά που βάζουμε καινούργια τιμή στον καταχωρητή ICR1 πρέπει να ενημερώνουμε τον καταχωρητή OCR1A ώστε η τιμή του να είναι η μισή του ICR1 για να έχουμε Duty Cycle 50%.

Assembly

```
.include "m328PBdef.inc"
.equ FOSC_MHZ=16

ldi r24 ,(1<<WGM11) | (0<<WGM10) | (1<<COM1A1)
sts TCCR1A, r24
ldi r24 ,(1<<WGM12) | (1<<WGM13) | (1<<CS11) ;prescale 8
sts TCCR1B, r24

ser r26
out DDRB, r26
clr r26
```

```
out DDRD, r26
ldi r16,0x00
sts OCR1AH,r16
ldi r16,0x80
sts OCR1AL,r16  ;duty cycle 50%

main:
ldi r21,0x00
sts ICR1H, r21
ldi r21,0x00
sts ICR1L, r21
in r20, PIND
com r20
cpi r20,0
breq main
cpi r20,0x01
breq d0
cpi r20,0x02
breq d1
cpi r20,0x04
breq d2
cpi r20,0x08
breq d3

d0:
ldi r16,0x1F
sts OCR1AH,r16
ldi r16,0x40
sts OCR1AL,r16
ldi r21,0x3E
sts ICR1H, r21
ldi r21,0x7F
sts ICR1L, r21
d00:
in r20, PIND
com r20
cpi r20,0x01
breq d00
rjmp main

d1:
ldi r16,0x0F
sts OCR1AH,r16
ldi r16,0xA0
sts OCR1AL,r16
ldi r21,0x1F
sts ICR1H, r21
ldi r21,0x3F
```

```
sts ICR1L, r21
d11:
in r20, PIND
com r20
cpi r20,0x02
breq d11
rjmp main

d2:
ldi r16,0x07
sts OCR1AH,r16
ldi r16,0xD0
sts OCR1AL,r16
ldi r21,0x0F
sts ICR1H, r21
ldi r21,0x9F
sts ICR1L, r21
d22:
in r20, PIND
com r20
cpi r20,0x04
breq d22
rjmp main

d3:
ldi r16,0x07
sts OCR1AH,r16
ldi r16,0xBE
sts OCR1AL,r16
ldi r21,0x0F
sts ICR1H, r21
ldi r21,0x7C
sts ICR1L, r21
d33:
in r20, PIND
com r20
cpi r20,0x08
breq d33
rjmp main
```

C

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

int main(void) {
    TCCR1A = (0<<WGM10) | (1<<WGM11) |(1<<COM1A1) ;
    TCCR1B = (1<<WGM12) | (1<<CS11)|(1<<WGM13);
    DDRB=0b00111111;
    OCR1AL = 0x80;
    while(1) {
        ICR1=0x0000;
        if((PIND&0b00000001)==0){
            while((PIND&0b00000001)==0)
                ICR1=0x3E7F;
                OCR1A =ICR1 /2;
            }
        else if ((PIND&0b00000010)==0){
            while((PIND&0b00000010)==0)
                ICR1=0x1F3F;
                OCR1A =ICR1 /2;
            }
        else if ((PIND&0b00000100)==0){
            while((PIND&0b00000100)==0)
                ICR1=0x0F9F;
                OCR1A =ICR1 /2;
            }
        else if ((PIND&0b00001000)==0){
            while((PIND&0b00001000)==0)
                ICR1=0x0F7C;
                OCR1A =ICR1 /2;
            }
        }

}
```