ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εργαστήριο Μικροϋπολογιστών

2η Εργαστηριακή Άσκηση

Ονοματεπώνυμα: **Μπέτζελος Χρήστος, Γιαννιός Γεώργιος-Ταξιάρχης**

Α.Μ. : **031 16 067, 031 16 156**

Ομάδα : **Α 16**

Εξάμηνο: **7ο**

1η Άσκηση

Υποθέτουμε ότι «τρέχει» το προηγούμενο πρόγραμμα του μετρητή (Πίνακας 2.1) με θύρα όμως εξόδου το PORTB. Δημιουργήσαμε ρουτίνα εξυπηρέτησης της εξωτερικής διακοπής INT1 (PD3) που όταν ενεργοποιείται να απαριθμεί το πλήθος των διακοπών με την προϋπόθεση ότι το dip switch PD7 είναι στο λογικό ‘1’, αλλιώς όχι. Ο παραπάνω μετρητής (Πίνακας 2.1, με θύρα όμως εξόδου το PORTB) που αποτελεί το κύριο πρόγραμμα, απεικονίζει στα leds PB7-PB0 την μέτρησή του και τρέχει με ταχύτητα μιας μέτρησης ανά ~ 0.2 του δευτερολέπτου. Η μέτρηση του πλήθους των εξωτερικών διακοπών INT1 δίνεται σε δυαδική μορφή στα leds PA7-PA0. Το πρόγραμμα δόθηκε σε assembly. Δεδομένου ότι υπήρχε η πιθανότητα στο πάτημα οποιουδήποτε διακόπτη να εμφανιστεί το φαινόμενο του σπινθηρισμού δηλ. να εμφανίζεται σαν να είχε πατηθεί περισσότερες φορές, δημιουργήσαμε κατάλληλη διαδικασία (βλ spin).

```
#include "m16def.inc"


.org 0x0
rjmp reset

;Αν ο διακόπτης PD7 είναι 1
;κάνε διακοπή
;αλλιώς συνέχισε τη μέτρηση
.org 0x4
sbic PIND, 7
rjmp ISR1
rjmp loop

;Αρχικοποίηση δείκτη στοίβας
;για ρουτίνες χρονοκαθυστέρησης
reset:
```

```
ldi r24 , low(RAMEND)
out SPL , r24
ldi r24 , high(RAMEND)
out SPH , r24

ser r26
out DDRB, r26;PORTB έξοδος
out DDRA, r26;PORTA έξοδος
clr r26
out DDRD, r26;PORTD είσοδος
ldi r23, 0x00;r23 : μετρητής διακοπών


;Μετρητής στην PORTB
loop:
;Ενεργοποίηση διακοπής INT1,στην ανερχόμενη ακμή
ldi r24 ,( 1 << ISC11) | ( 1 << ISC10)
out MCUCR , r24
ldi r24 ,( 1 << INT1)
out GICR , r24
sei

out PORTB , r26
ldi r24 , low(100)
ldi r25 , high(100)
rcall wait_msec
inc r26                          ;Αύξηση μετρητή
rjmp loop


;Εξυπηρέτηση διακοπής INT1
ISR1:
sts 0x3D2, r26                   ;Αποθήκευση τιμής μετρητή

;Αντιμετώπιση σπινθιρισμού
spin:
ldi r24 ,(1 << INTF1)            ;Γράφω λογικό ένα στο bit 7 του GIFR
out GIFR ,r24

ldi r24 , low(5)
ldi r25 , high(5)
rcall wait_msec

in r24, GIFR
sbrc r24, 7
rjmp spin

inc r23
out PORTA , r23     ;Εμφάνιση πλήθους διακοπών σε PORTA
lds r26, 0x3D2             ;Ανάκτηση τιμής μετρητή
reti

;Ρουτίνες χρονοκαθυστέρησης
wait_msec:
 push r24
```

```
 push r25
 ldi r24, low(998)
 ldi r25, high(998)
 rcall wait_usec
 pop r25
 pop r24
 sbiw r24 , 1
 brne wait_msec
 ret

wait_usec:
 sbiw r24 ,1
 nop
 nop
 nop
 nop
 brne wait_usec
 ret
```

2η Άσκηση

Στο προηγούμενο ζήτημα για το κύριο πρόγραμμα του μετρητή, στη θέση της διακοπής INT1 και της ρουτίνας εξυπηρέτησης της, δόθηκε μια άλλη ρουτίνα εξυπηρέτησης συνδεδεμένης με την εξωτερική διακοπή INT0 (PD2) που όταν ενεργοποιείται άναβει τόσα LEDs της θύρας PORTC (PC0-7) αρχίζοντας από τα LSB όσο το πλήθος των διακοπτών (dip switches) της θύρας PORTA (PA7-PA0) που είναι ON. Το πρόγραμμα δόθηκε σε assembly.

```
#include "m16def.inc"

.org 0x0
rjmp reset

.org 0x2
rjmp ISR0

;Αρχικοποίηση δείκτη στοίβας
reset:
ldi r24 , low(RAMEND)
out SPL , r24
ldi r24 , high(RAMEND)
out SPH , r24

ser r26
out DDRB, r26 ;PORTB έξοδος
out DDRC, r26 ;PORTC έξοδος
clr r26
out DDRA, r26 ;PINA είσοδος
ldi r22, 0x00

;Μετρητής
loop:
```

```
;Ενεργοποίηση διακοπής INT0 στην
;ανερχόμενη ακμή
ldi r24 ,( 1 << ISC01) | ( 1 << ISC00)
out MCUCR , r24
ldi r24 ,( 1 << INT0)
out GICR , r24
sei

out PORTB , r26           ;Εμφάνιση μέτρησης
ldi r24, low(200)
ldi r25, high(200)
rcall wait_msec
inc r26                   ;Αύξηση μετρητή
rjmp loop

;Εξυπηρέτηση διακοπής INT0
ISR0:
sts 0x3D2, r26
ldi r22, 0x00
;Αντιμετώπιση φαινομένου σπινθιρισμού
spin:
ldi r24 ,(1 << INTF1)
out GIFR ,r24
ldi r24 , low(5)
ldi r25 , high(5)
rcall wait_msec
in r24, GIFR
sbrc r24, 7
rjmp spin

in r23, PINA              ;Μέτρηση ανοικτών διακοπτών PINA
add r23,r22               ;Ενημέρωση Z καταχωρητή

count:
breq done                 ;Αν Z =1 τότε τελείωσε η μέτρηση
lsr r23                   ;Αλίως αριστερή περιστροφή και αποθήκευση σε C
brcs increase             ;Αν C = 1 ,αύξησε το μετρητή των άσσων
rjmp count

increase:
inc r22
rjmp count

done:                     ;Τέλος μέτρησς
out PORTC , r22           ;Εμφάνιση αποτελέσματος
lds r26, 0x3D2
reti

;Ρουτίνες χρονοκαθυστέρησης
wait_msec:
 push r24
 push r25
 ldi r24 , low(998)
 ldi r25 , high(998)
 rcall wait_usec
```

```
 pop r25
 pop r24
 sbiw r24 , 1
 brne wait_msec
 ret

wait_usec:
 sbiw r24 ,1
 nop
 nop
 nop
 nop
 brne wait_usec
 ret
```

3η Άσκηση

Υλοποιήθηκε αυτοματισμός που ελέγχει το άναμμα και το σβήσιμο ενός φωτιστικού σώματος. Όταν πατάμε το push button PD3 (δηλαδή με την ενεργοποίηση της INT1) ή το PA7 (που υποθέτουμε ότι αντιστοιχεί σε ένα αισθητήρα κίνησης) ανάβει το led PB0 της θύρας PORTB (που υποθέτουμε ότι αντιπροσωπεύει το φωτιστικό σώμα). Το led θα σβήνει μετά από 4 sec, εκτός και αν ενδιάμεσα υπάρξει νέο πάτημα του PD3 ή PA7, οπότε και ο χρόνος των 4 sec θα ανανεώνεται. Κάθε φορά που γίνεται ανανέωση να ανάβουν όλα τα led της θύρας PORTB (PB7-PB0) για 0.5 sec, μετά να σβήνουν εκτός από το led PB0 που παραμένει συνολικά για 4 sec εκτός και αν ανανεωθεί. Έγινε χρήση του Χρονιστή Timer1. Το πρόγραμμα δόθηκε σε assembly και σε C.

Πρόγραμμα σε assembly

```
#include "m16def.inc"

.org 0x0
rjmp reset

.org 0x4
rjmp ISR1

.org 0x10
rjmp ISR_TIMER1_OVF

;Ενεργοποίηση διακοπής INT1, στην κατερχόμενη ακμή
reset:
ldi r24 ,( 1 << ISC11) | ( 0 << ISC10)
out MCUCR , r24
ldi r24 ,( 1 << INT1)
out GICR , r24
sei

;Αρχικοποίηση δείκτη στοίβας
ldi r24 , low(RAMEND)
```

```
out SPL , r24
ldi r24 , high(RAMEND)
out SPH , r24

ser r26
out DDRB, r26             ;PORTB έξοδος
clr r26
out DDRA, r26             ;PORTA είσοδος

ldi r24 ,(1<<TOIE1)       ;ενεργοποίηση διακοπής υπερχύλισης
out TIMSK ,r24            ;για timer1

ldi r24 ,(1<<CS12) | (0<<CS11) | (1<<CS10) ; CK/1024
out TCCR1B ,r24

main:
sbis PINA, 7              ;έλεγχος πατήματος PA7
rjmp main
main2:
sbic PINA, 7               ;έλεγχος αφήματος PA7
rjmp main2

;8MHz/1024=7812.5Hz
;4sec * 7812.5 Hz = 31250
;65536-31250  = 34286
;34286(DEC) = 0x85EE

ldi r24, 0x85             ;αρχικοποίηση TCNT1
out TCNT1H, r24           ;για 4 sec
ldi r24, 0xEE
out TCNT1L, r24

sbic PINB, 0              ;έλεγχος PB0
rjmp reload1              ;Αν το PB0 =1, είμαστε σε κατάσταση ανανέωσης

ldi r26, 0x01             ;άναμα PB0
out PORTB, r26
rjmp main
;Κατάσταση ανανέωσης
reload1:
ldi r26, 0xFF
out PORTB, r26            ;Άναμμα όλων των LED για 0.5 sec
ldi r24, low(500)
ldi r25, high(500)
rcall wait_msec
ldi r26, 0x01
out PORTB, r26            ;Σβήσιμο όλων εκτός απο PB0
rjmp main

ISR1:
ldi r24,0x85              ;αρχικοποίηση TCNT1
out TCNT1H ,r24           ;για 4 sec
ldi r24, 0xEE
out TCNT1L ,r24
```

```
sbic PINB, 0              ;έλεγχος PB0
rjmp reload2

ldi r26, 0x01             ;άναμα PB0
out PORTB, r26
reti

reload2:
ldi r26, 0xFF
out PORTB, r26
ldi r24, low(500)
ldi r25, high(500)
rcall wait_msec
ldi r26, 0x01
out PORTB, r26
reti


;Ρουτίνα εξυπηρέτησης υπερχείλισης χρονομέτρου

ISR_TIMER1_OVF:
ldi r26, 0x00
out PORTB, r26            ;Σβήνουν όλα τα LED
reti

;Ρουτίνες χρονοκαθυστέρησης
wait_msec:
 push r24
 push r25
 ldi r24 , low(998)
 ldi r25 , high(998)
 rcall wait_usec
 pop r25
 pop r24
 sbiw r24 , 1
 brne wait_msec
 ret

wait_usec:
 sbiw r24 ,1
 nop
 nop
 nop
 nop
 brne wait_usec
 ret
```

Πρόγραμμα σε C

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include<avr/interrupt.h>
#include <util/delay.h>
ISR (INT1_vect)                         // Routine interrupt INT1
{
       if((PINB & 0x01)==1) {           // If PB0=1 -> reload
              TCNT1H=0x85;              // Initiating TCNT1 for 4sec
              TCNT1L=0xEE;
              PORTB=0xFF;               // Open all PB leds
              _delay_ms(500);           // Delay for 0.5sec
              PORTB=0x01;               // Open only PB0
       }
       else {                           // If PB0=0 -> start
              TCNT1H=0x85;
              TCNT1L=0xEE;
              PORTB=0x01;               // Open only PB0
       }
}
ISR (TIMER1_OVF_vect)                   // Routine interrupt overflow Timer1
{
       PORTB=0x00;                      // Switch off all
}
void main(void)
{
       DDRB=0xFF;                       // Initiating PORT B as output
       DDRA=0x00;                       // Initiating PORT B as input
       GICR=1<<INT1;                    // Activation interrupt INT1:On
       MCUCR=(1<<ISC11)|(0<<ISC10);     // INT1 Mode: (1->0)
       sei();                           // Activation interrupts
       TIMSK=1<<TOIE1;                 // Activation interrupt overflow Timer1
       TCCR1B=(1<<CS12)|(0<<CS11)|(1<<CS10);   //CK/1024
       while(1)
       {
              if((PINA & 0x80)==128) {          // Check push PA0
                     while ((PINA & 0x80)==128);// Check release PA0
                     if((PINB & 0x01)==1) {     // If PB0=1 -> reload
                            TCNT1H=0x85;        // Initiating TCNT1 for 4sec
                            TCNT1L=0xEE;
                            PORTB=0xFF;         // Open all PB leds
                            _delay_ms(500);     // Delay for 0.5sec
                            PORTB=0x01;         // Open only PB0
                     }
                     else {                     // If PB0=0 -> start
                            TCNT1H=0x85;        // Initiating TCNT1 for 4sec
                            TCNT1L=0xEE;
                            PORTB=0x01;         // Open only PB0
                     }
              }
       };
}
```