**Εργαστήριο Μικροϋπολογιστών**

**5η Άσκηση**

Ομάδα: Δ12

Βακαλόπουλος Θεόδωρος, ΑΜ: 03114013

Μαυρομμάτης Ιάσων, ΑΜ: 03114771

Νικητοπούλου Δήμητρα, ΑΜ: 03114954

**Άσκηση 1**

.INCLUDE "m16def.inc"

.DEF counter = r20

.DEF reg = r16

.DEF interr = r18

.org 0x00

rjmp reset

.org 0x04

rjmp ISR1

reset:

ldi r24, low(RAMEND)

out SPL, r24

ldi r24, high(RAMEND)

out SPH, r24

ser reg

out DDRB, reg ;PortB as counter output

out DDRA, reg ;PortA as interr output

clr reg

out PORTA,reg

out DDRD, reg ;PortD as input

clr counter

clr interr ; interrupt number is zero initially

ldi reg, ( 1 << ISC11) | ( 1 << ISC10) ; posedge for int1

out MCUCR,reg

ldi reg, (1 << INT1)

out GICR,reg

sei

count:

out PORTB, counter

ldi r24,low(200)

ldi r25,high(200) ; delay 200ms=0.2sec

rcall wait\_msec

inc counter

rjmp count

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

ISR1:

push r24 ; save delay registers

push r25

in reg,SREG ; save status register

push reg

check:

ldi reg, (1 << INTF1)

out GIFR,reg

ldi r24,low(5)

ldi r25,high(5)

rcall wait\_msec

in reg,GIFR ; INTF1 must be zero to start interrupt

andi reg,0x80

cpi reg,0x00

brne check ; if carry is set wait again 5msec and check

in reg, PIND

andi reg, 0x80 ;Isolate PD7

cpi reg,0x00

breq leave ; if PD7=0, skip interruption (return)

inc interr

out PORTA,interr ; show number of interrupts

ldi r24, low(998) ; wait 1sec for this interrupt

ldi r25, high(998)

rcall wait\_msec

leave:

pop reg

out SREG,reg

pop r25

pop r24

reti

**Σχόλια:**

* Οι τιμές στους καταχωρητές GICR και MCUCR δίνονται έτσι ώστε το πρόγραμμα να δέχεται και να εξυπηρετεί ΙΝΤ1 διακοπές στο πάτημα του διακόπτη PD3.
* Το τμήμα του μετρητή που αποτελεί το κύριό μας πρόγραμμα καθώς και το 1ο τμήμα της ρουτίνας εξυπηρέτησης που αφορά στην αντιμετώπιση του φαινομένου της αναπήδησης έχουν παρθεί από την εκφώνηση της άσκησης.
* Ο έλεγχος για το διακόπτη PD7 γίνεται μέσα στη ρουτίνα εξυπηρέτησης της διακοπής και λαμβάνεται εκεί η απόφαση για εκτέλεση η μη της αύξησης του μετρητή διακοπών και της εμφάνισης του στην αντίστοιχη θύρα.

**Άσκηση 2**

.INCLUDE "m16def.inc"

.DEF counter = r20

.DEF reg = r16

.DEF mask = r18

.DEF temp = r17

.org 0x00

rjmp reset

.org 0x02

rjmp ISR0

reset:

ldi r24, low(RAMEND)

out SPL, r24

ldi r24, high(RAMEND)

out SPH, r24

ser reg

out DDRB, reg ;PortB as counter output

out DDRC, reg ;PortC as interr output

clr reg

;out DDRD, reg ;PortD as input

out DDRA,reg ;PortA as input

clr counter

ldi reg,(1<<ISC01)|(1<<ISC00)

out MCUCR,reg

ldi reg,1<<INT0

out GICR,reg

sei

out PORTC, counter

count:

out PORTB, counter

ldi r24,low(200)

ldi r25,high(200)

rcall wait\_msec

inc counter

rjmp count

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

ISR0:

push r24 ; save delay registers

push r25

in reg,SREG ; save status register

push reg

check:

ldi reg,(1<<INTF0)

out GIFR,reg

ldi r24,low(5)

ldi r25,high(5)

rcall wait\_msec

in reg,GIFR

andi reg,0x80 ;isolate 1st bit

cpi reg,0x00

brne check ; if carry is set wait again 5msec and check

ldi mask,0xff ;11111111 at first

ldi temp,0x08 ;8 switches, 8 bits to check through carry

in reg, PINA

loop1:

ror reg

brcc not\_on

lsl mask ;11111110 add zeros at the end every time you find 1 at a switches

not\_on:

dec temp

brne loop1

com mask ;take complement of mask to light the leds

out PORTC,mask

ldi r24, low(998) ; wait

ldi r25, high(998)

rcall wait\_msec

leave:

pop reg

out SREG,reg

pop r25

pop r24

reti

**Σχόλια:**

* Οι τιμές στους καταχωρητές GICR και MCUCR δίνονται έτσι ώστε το πρόγραμμα να δέχεται και να εξυπηρετεί ΙΝΤ0 διακοπές στο πάτημα του διακόπτη PD2.
* Το τμήμα του μετρητή που αποτελεί το κύριό μας πρόγραμμα καθώς και το 1ο τμήμα της ρουτίνας εξυπηρέτησης που αφορά στην αντιμετώπιση του φαινομένου της αναπήδησης έχουν παρθεί από την εκφώνηση της άσκησης.
* Η λογική της υλοποίησης της ρουτίνας εξυπηρέτησης που ακολουθήθηκε είναι ότι θέτουμε στο ff έναν καταχωρητή και για κάθε οn διακόπτη που διαβάζουμε στο input εκχωρούμε ένα μηδενικό στο τέλος και άρα τελικά η ζητούμενη έξοδος βρίσκεται στο συμπλήρωμα αυτού του καταχωρητή.

**Άσκηση 3**

.include "m16def.inc"

.def reg = r20

.def temp = r21

.def flag = r22

.org 0x0

rjmp reset

.org 0x4

rjmp interrupt1

.org 0x10

rjmp ISR\_TIMER1\_OVF

reset:

ldi reg, low(RAMEND)

out SPL, reg

ldi reg, high(RAMEND)

out SPH, reg

ser flag ;Initialise flag = ffh

ser reg

out DDRB, reg ;Port B as output

clr reg

out DDRA, reg ;Port A as input

out DDRD, reg ;Port D as input

ldi reg,(1<<ISC11)|(1<<ISC10)

out MCUCR, reg

ldi reg, (1<<INT1)

out GICR, reg

ldi reg, (1<<TOIE1)

out TIMSK, reg

ldi reg, (1<<CS12)|(0<<CS11)|(1<<CS10)

out TCCR1B, reg

sei

start:

in reg, PINA

andi reg, 0x80

cpi reg, 0x00

breq start

ldi reg, 0x85 ;Initialise start point of timer1 at 34286(10)

out TCNT1H, reg ;so as to cause an interrupt after 4s

ldi reg, 0xEE

out TCNT1L, reg

cpi flag, 0xff

breq next ;At first don't switch all leds on for 0.5s

clr reg

out GICR, reg ;Disable interrupts

ldi r24, low(500)

ldi r25, high(500)

ldi reg, 0xff

out PORTB, reg

rcall wait\_msec ;Call routine for delay 0.5s

ldi temp, (1<<INT1)

out GICR, temp ;Enable interrupts

next:

clr flag

ldi reg, 0x01

out PORTB, reg

loop1:

in reg, PINA

andi reg, 0x80

cpi reg, 0x80

breq loop1

rjmp start

ISR\_TIMER1\_OVF:

clr temp

out PORTB, temp ;Switch off led after 4s

ser flag ;Reinitialise flag

reti

interrupt1:

in temp,SREG ;save status register

push temp

check:

ldi temp,(1<<INTF1)

out GIFR,temp

ldi r24,low(5)

ldi r25,high(5)

rcall wait\_msec

in temp, GIFR

andi temp,0x80

cpi temp,0x00

brne check

ldi temp, 0x85 ;Initialise start point of timer1 at 34286(10)

out TCNT1H, temp ;so as to cause an interrupt after 4s

ldi temp, 0xEE

out TCNT1L, temp

cpi flag, 0xff

breq continue

ser temp

out PORTB, temp ;Switch on all leds of Port B for 0.5s

clr temp

out GICR, temp ;Disable interrupts

ldi r24, low(500)

ldi r25, high(500)

rcall wait\_msec ;Call routine for delay 0.5s

ldi temp, (1<<INT1)

out GICR, temp ;Enable interrupts

jmp continue2

continue:

clr flag

continue2:

ldi temp, 0x01

out PORTB, temp

pop temp

out SREG, temp

reti

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

**Σχόλια:**

* Σε αυτή την άσκηση χρησιμοποιούμε χρονιστή προκειμένου να μετράμε 4s στα οποία μένει αναμμένο το φωτιστικό (PINB0). Συγκεκριμένα χρησιμοποιούμε τον 16-ψήφιο χρονιστή TCNT1 στον οποίο ενεργοποιούμε τη διακοπή υπερχείλισης, δηλαδή όταν ο χρονιστής φτάσει στην τιμή 65536 προκαλεί μία διακοπή, την οποία διαχειριζόμαστε σβήνοντας το φωτιστικό. Επιπλέον, επιλέγουμε συχνότητα αύξησης του χρονιστή ίση με το 1/1024 της συχνότητας του μικροελεγκτή, δηλαδή 8MHz/1024 = 7812.5 Hz. Επομένως για να μετρήσουμε 4s πρέπει να μετρήσουμε 4Χ7812.5 = 31250 κύκλους οπότε κάθε φορά στην ανανέωση του χρονιστή πρέπει να τον αρχικοποιούμε στη τιμή 65536-31250=.
* Εφόσον το φωτιστικό μπορεί να τεθεί σε λειτουργία με δύο διακόπτες (PD7 και PD3-διακοπή INT1) πρέπει να διαχειριζόμαστε το πάτημα αυτών των δύο κουμπιών στον κώδικα. Συγκεκριμένα, ο έλεγχος για το πάτημα του PD7 γίνεται στο κύριο σώμα του προγράμματος, το οποίο αρχικά αναμένει το πάτημα του διακόπτη, στη συνέχεια ανάβει το φωτιστικό ή ανανεώνει το χρόνο των 4s αν είναι ήδη ενεργοποιημένο, αναμένει την επαναφορά του διακόπτη και επαναλαμβάνει αενάως την ίδια διαδικασία. Η διαχείριση της ενεργοποίησης της διακοπής INT1 γίνεται στη ρουτίνα εξυπηρέτησής της interrupt1 όπου ανάβει το φωτιστικό αν ήταν σβηστό ή ανανεώνεται ο χρόνος των 4s αν ήταν αναμμένο.
* Για να μπορούμε να ξεχωρίσουμε αν πρόκειται για άναμμα του φωτιστικού ή για ανανέωση του χρόνου λειτουργίας του (κατά την οποία ανάβουν όλα τα leds της θύρας PORTB για 0,5s) χρησιμοποιούμε έναν καταχωρητή ως σημαία. Ειδικότερα, τον αρχικοποιούμε στην τιμή FFh και μόλις ενεργοποιηθεί το φωτιστικό με κάποιον από τους δύο τρόπους τον μηδενίζουμε. Στη συνέχεια, αν ξαναπατηθεί κάποιος διακόπτης εντός 4 δευτερολέπτων θα γίνει ανανέωση του χρόνου με ταυτόχρονο άναμμα όλων των leds για 0.5s με χρήση της ρουτίνας wait\_msec. Σε αντίθετη περίπτωση με τη λήξη του χρόνου αρχικοποιούμε ξανά τον καταχωρητή-σημαία στην τιμή FFh μέσα στη ρουτίνα εξυπηρέτησης της διακοπής του χρονιστή, ώστε στην επόμενη ενεργοποίηση να μην ενεργοποιηθούν όλα τα leds αλλά μόνο αυτό που αντιστοιχεί στο φωτιστικό.