Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχ. Και Μηχ. Υπολογιστών
Εργαστήριο Μικροϋπολογιστών , 7ο εξάμηνο - Ροή Υ
Ακαδημαϊκή Περίοδος : 2011-2012

6η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ

Ομάδα: C16
Γερακάρης Βασίλης Α.Μ.: 03108092
Διβόλης Αλέξανδρος Α.Μ.: 03107238
Ίεντσεκ Στέργιος Α.Μ.: 03108690

Σε όλες τις ασκήσεις έχουμε γράψει ένα κομμάτι κώδικα ώστε να αποφύγουμε το φαινόμενο του σπινθηρισμού κάθε φορά που καλείται η ρουτίνα εξυπηρέτησης μιας εκ των διακοπών INT0 και INT1

Άσκηση 2.1

Σε αυτή την άσκηση υλοποιήσαμε ένα πρόγραμμα, σε assembly του AVR, το οποίο υλοποιεί το μετρητή που μας δόθηκε, με την προσθήκη μιας ρουτίνας εξυπηρέτησης για τη διακοπή INT0. Το πλήθος των διακοπών έπρεπε να εμφανίζεται στα LEDS της PORTB. Στην περίπτωση που το dip switch PD0 βρίσκεται σε λογικό 0 και δινόταν εξωτερική διακοπή INT0 στον PD2, έπρεπε να αυξανόταν ο μετρητής των διακοπών, αλλιώς να έμενε σταθερός. Ο κώδικας της άσκησης που υλοποιήσαμε παρατίθεται παρακάτω :

```
.include "m16def.inc"

.def temp=r16
.def leds=r17
.def counter=r18
.def icounter=r19
	jmp reset 			;Reset Handler
	jmp interr0 			;IRQ0 Handler

reset:
	ldi temp,high(RAMEND)
	out SPH,temp
	ldi temp,low(RAMEND)
	out SPL,temp 			;Setting Stack
	clr temp
	out DDRD,temp
	ser temp
	out PORTD,temp 			;Setting PORTD as input
	out DDRA,temp 			;Setting PORTA as output
	out DDRB,temp 			;Setting PORTB as output
	clr temp
	out PORTB,temp
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
	ldi temp,0x03 			;0b0000 0011->ISC01-ISC00 = 11, Sleep enable - off
	out MCUCR,temp 			;and Rising edge trigger
	ldi temp,0x40
	out GIMSK,temp 			;Setting Interrupt Mask
	sei
	clr counter
	clr icounter
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
loop:
	out PORTA,counter 		;Print loop counter (Leds A)
	ldi r24,low(100) 		;load r25:r24 with 100
	ldi r25,high(100) 		;delay 100 ms
	rcall wait_msec
	inc counter 			;Increase counter
	rjmp loop 			;Loop!
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
interr0:
```

```
        rcall spitha0
        in temp,PIND                ;Input port D
        sbrc temp,0                 ;If (PD0==1) return
        rjmp int0_ext
        inc icounter                ;else increase icounter
        out PORTB,icounter          ;Print it!
int0_ext:
        sei
        reti                        ;return

wait_usec:
        sbiw r24 ,1                 ; 2 cycles (0.250 micro sec)
        nop                         ; 1 cycles (0.125 micro sec)
        nop                         ; 1 cycles (0.125 micro sec)
        nop                         ; 1 cycles (0.125 micro sec)
        nop                         ; 1 cycles (0.125 micro sec)
        brne wait_usec              ; 1 or 2 cycles (0.125 or 0.250 micro sec)
        ret                         ; 4 cycles (0.500 micro sec)

wait_msec:
        push r24                    ; 2 cycles (0.250 micro sec)
        push r25                    ; 2 cycles
        ldi r24 , 0xe6              ; load register r25:r24 with 998 (1 cycles- 0.125 µsec)
        ldi r25 , 0x03              ; 1 cycles (0.125 micro sec)
        rcall wait_usec             ; 3 cycles (0.375 micro sec), total delay 998.375 µsec
        pop r25                     ; 2 cycles (0.250 micro sec)
        pop r24                     ; 2 cycles
        sbiw r24 , 1                ; 2 cycles
        brne wait_msec              ; 1 or 2 cycles (0.125 or 0.250 micro sec)
        ret                         ; 4 cycles (0.500 micro sec)

spitha0:
        ldi temp,0x40               ;0b0100 0000
        out GIFR,temp               ;Setting zero INTF0
        ldi r24,0x05
        ldi r25,0x00
        rcall wait_msec             ;wait 5 msec
        in temp,GIFR                ;Check if INTF0==1
        sbrc temp,6
        rjmp spitha0                ;If INTF0==1 loop
        ret                         ;If INTF0==0 return
```

Άσκηση 2.2

Σε αυτή την άσκηση προσθέσαμε μια ρουτίνα εξυπηρέτησης της διακοπής INT1 στο προηγούμενο πρόγραμμα, ώστε να μετρά τα dip switches της PORTB τα οποία είναι ON και να απεικονίζει το πλήθος τους στα 4 LSBits της PORTC. Στο πρόγραμμα αυτό μετατρέπουμε συχνά την PORTB από είσοδο σε έξοδο και το αντίστροφο. Πιο συγκεκριμένα, όταν συμβαίνει INT1 πρέπει να λειτουργεί προσωρινά ως είσοδος και στο τέλος της ρουτίνας να λειτουργεί πάλι ως έξοδος.
Κατά τη μετατροπή αυτή παρατηρήσαμε ότι η αλλαγή αυτή δεν πραγματοποιούνταν όπως θα έπρεπε και έτσι δεν μπορούσαμε να κάνουμε σωστά την ανάγνωση. Παρατηρήσαμε ότι η PORTB δεν μπορεί να χρησιμοποιηθεί άμεσα ως είσοδος αλλά πρέπει να περάσει κάποιο χρονικό διάστημα. Στο AVR Studio αυτό γινόταν με την εισαγωγή 2 εντολών NOP. Στην πλακέτα του ATMega16 (ως πραγματικό, μη ιδανικό στοιχείο) αυτό δεν αρκούσε. Το πρόβλημα αντιμετωπίστηκε βάζοντας το κομμάτι ελέγχου για σπινθηρισμό πρίν από τον κώδικα για ανάγνωση από την PORTB. Η καθυστέρηση των 5ms που προκαλεί ο έλεγχος για σπινθιρισμό, είναι αρκετή για να μπορούμε από εκεί και πέρα να χρησιμοποιήσουμε την PORTB ως είσοδο. Ο κώδικας παρατίθεται παρακάτω:

```
.include "m16def.inc"

.def temp=r16
.def in_thng=r17
.def counter=r18
.def icounter=r19
.def loop_cnt=r20
.def sw_counter=r21
       jmp reset                  ;Reset Handler
       jmp interr0                ;IRQ0  Handler
       jmp interr1                ;IRQ1  Handler

reset:
       ldi temp,high(RAMEND)
       out SPH,temp
       ldi temp,low(RAMEND)
       out SPL,temp
       clr temp
       out DDRD,temp
       ser temp
       out PORTD,temp             ;Setting PORTD as input
       out DDRA,temp              ;Setting PORTA as output
       out DDRB,temp              ;Setting PORTB as output
       out DDRC,temp              ;Setting PORTC as output
       clr temp
       out PORTB,temp
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
       ldi temp,0x0f              ;0b0000 1111->ISC01-ISC00 = 11, ISC11-ISC10 = 11,
       out MCUCR,temp             ;Sleep enable-off and Rising edge triggered interrupts
       ldi temp,0xc0
       out GIMSK,temp             ;Setting Interrupt Mask
       sei
       clr counter                ;Set counters to zero
```

```
        clr icounter
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
loop:
        out PORTA,counter           ;Print loop counter (Leds A)
        ldi r24,low(100)            ;load r25:r24 with 100
        ldi r25,high(100)           ;delay 100 ms
        rcall wait_msec
        inc counter                 ;Increase counter
        rjmp loop                   ;Loop!
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
interr0:
        rcall spitha0
        in temp,PIND                ;Input port D
        sbrc temp,0                 ;If (PD0==1) return
        rjmp int0_ext
        inc icounter                ;else increase icounter
        out PORTB,icounter          ;Print it!
int0_ext:
        sei
        reti                        ;return
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
interr1:
        clr temp
        out DDRB,temp
        ser temp
        out PORTB,temp              ;Making input PORTB
        rcall spitha1
        clr sw_counter              ;Clear switch counter
        in in_thng,PINB             ;Read PORTB
        ldi loop_cnt,8
on_loop:
        sbrc in_thng,0              ;Check if 0 bit is 1, if not skip increasing
        inc sw_counter
        ror in_thng
        dec loop_cnt
        brne on_loop
        out PORTC,sw_counter        ;Print result

        ser temp
        out DDRB,temp               ;Making output PORTB
        out PORTB,icounter          ;Refresh leds of interrupt counting
        sei                         ;Reenabling interrupts
        reti                        ;Return
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
wait_usec:
        sbiw r24 ,1                 ; 2 cycles (0.250 micro sec)
        nop                         ; 1 cycles (0.125 micro sec)
        nop                         ; 1 cycles (0.125 micro sec)
        nop                         ; 1 cycles (0.125 micro sec)
        nop                         ; 1 cycles (0.125 micro sec)
        brne wait_usec              ; 1 or 2 cycles (0.125 or 0.250 micro sec)
        ret                         ; 4 cycles (0.500 micro sec)

wait_msec:
        push r24                    ; 2 cycles (0.250 micro sec)
        push r25                    ; 2 cycles
        ldi r24 , 0xe6              ; load register r25:r24 with 998 (1 cycles-0.125 μsec)
        ldi r25 , 0x03              ; 1 cycles (0.125 micro sec)
```

```
        rcall wait_usec             ; 3 cycles (0.375 micro sec), total delay 998.375 µsec
        pop r25                     ; 2 cycles (0.250 micro sec)
        pop r24                     ; 2 cycles
        sbiw r24 , 1                ; 2 cycles
        brne wait_msec              ; 1 or 2 cycles (0.125 or 0.250 micro sec)
        ret                         ; 4 cycles (0.500 micro sec)

spitha0:
        ldi temp,0x40               ;0b0100 0000
        out GIFR,temp               ;Setting zero INTF0
        ldi r24,0x05
        ldi r25,0x00
        rcall wait_msec             ;wait 5 msec
        in temp,GIFR                ;Check if INTF0==1
        sbrc temp,6
        rjmp spitha0                ;If INTF0==1 loop
        ret                         ;If INTF0==0 return

spitha1:
        ldi temp,0x80               ;0b1000 0000
        out GIFR,temp               ;Setting zero INTF1
        ldi r24,0x05
        ldi r25,0x00
        rcall wait_msec             ;wait 5 msec
        in temp,GIFR                ;Check if INTF1==1
        sbrc temp,6
        rjmp spitha1                ;If INTF1==1 loop
        ret                         ;If INTF1==0 return
```

Άσκηση 2.3

Σε αυτή την άσκηση υλοποιήσαμε ένα πρόγραμμα το οποίο ελέγχει την λειτουργία ενός φωτιστικού ως εξής: Το φωτιστικό αυτό ανάβει είτε με το πάτημα του διακόπτη PA0 είτε κατά την ύπαρξη διακοπής INT0. Έπειτα από 3sec το φωτιστικό σβήνει. Αν κατά τη διάρκεια που είναι αναμμένο το φωτιστικό συμβεί ένα από τα δυο προαναφερθέντα γεγονότα, ο χρόνος των τριών δευτερολέπτων ξεκινά από την αρχή. Πιο συγκεκριμένα, όποτε συμβεί ένα από τα δυο γεγονότα, ανάβουμε το φωτιστικό και θέτουμε στο χρονιστή το κατάλληλο νούμερο (0xA472) ώστε ο Timer1 να ξεκινήσει να μετρά από το 42098. Μετρώντας με συχνότητα 8MHz/1024=7812.5, εξασφαλίζουμε ότι μετά από 3 sec θα γίνει υπερχείλιση και θα ενεργοποιηθεί η ρουτίνα εξυπηρέτησης η οποία θα σβήσει τα φώτα. Ο κώδικας παρατίθεται παρακάτω:

```
.include "m16def.inc"
.def temp=r16
.def in_thng=r17
.def hi_clck=r18
.def lo_clck=r19
.def fwta=r20

       jmp reset                  ;Reset Handler
       jmp interr0                ;IRQ0  Handler
.org 0x10
       jmp timer1_rout
reset:
       ldi temp,high(RAMEND)
       out SPH,temp
       ldi temp,low(RAMEND)
       out SPL,temp               ;Setting Stack
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
       ldi temp,0x03              ;0b0000 0011->ISC01-ISC00 = 11, Sleep enable - off
       out MCUCR,temp             ;and Rising edge trigger
       ldi temp,0x40
       out GIMSK,temp             ;Setting Interrupt Mask

       ldi temp,0x05              ;0b0000 0101 -> Clock's Frequency Divisor = 1024
       out TCCR1B,temp
       ldi temp,0x04              ;0b0000 0100 -> Enable Timer1
       out TIMSK,temp
       sei

;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
       clr temp                   ;Setting PORTD as input
       out DDRD,temp
       ser temp
       out PORTD,temp
       ldi temp,0x02              ;0b00000010 : PA0->input PA1(and evrthng else)->output
       out DDRA,temp
       ldi temp,0xfd              ;0b11111101 : Setting PULL-UP Resistors
       out PORTA,temp
```

```
	ldi hi_clck,0xa4		;hi_clck:lo_clck=0xA472 = 0d42098 {=65536-3*7812.5}
	ldi lo_clck,0x72		;8000000/1024=7812.5Hz timer1's frequency
	ldi fwta,0x02			;0b0000 0010 to set lights on!
loop:
	in in_thng,PINA			;Reading input from PORTA
	sbrs in_thng,0			;If PA0==1 exit loop
	rjmp loop			;else loop!
	out PORTA,fwta			;If PA0 has been pressed -> lighs on!
	out TCNT1H,hi_clck		;Setting timer clock!
	out TCNT1L,lo_clck
	rjmp loop			;Loop!
;-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
interr0:
	rcall spitha0			;Catching false triggering from signal bounce
	out PORTA,fwta			;Lights on
	out TCNT1H,hi_clck		;Resetting Clock
	out TCNT1L,lo_clck
	sei				;Reenabling interrupts
	reti

timer1_rout:
	clr fwta
	out PORTA,fwta			;Lights off! (when timer is out)
	ldi fwta,0x02			;0b0000 0010 to set lights on next time!
	sei				;Reenabling interrupts
	reti

wait_usec:
	sbiw r24 ,1			; 2 cycles (0.250 micro sec)
	nop				; 1 cycles (0.125 micro sec)
	nop				; 1 cycles (0.125 micro sec)
	nop				; 1 cycles (0.125 micro sec)
	nop				; 1 cycles (0.125 micro sec)
	brne wait_usec			; 1 or 2 cycles (0.125 or 0.250 micro sec)
	ret				; 4 cycles (0.500 micro sec)

wait_msec:
	push r24			; 2 cycles (0.250 micro sec)
	push r25			; 2 cycles
	ldi r24 , 0xe6			; load register r25:r24 with 998 (1 cycles-0.125 µsec)
	ldi r25 , 0x03			; 1 cycles (0.125 micro sec)
	rcall wait_usec			; 3 cycles (0.375 micro sec), total delay 998.375 µsec
	pop r25				; 2 cycles (0.250 micro sec)
	pop r24				; 2 cycles
	sbiw r24 , 1			; 2 cycles
	brne wait_msec			; 1 or 2 cycles (0.125 or 0.250 micro sec)
	ret				; 4 cycles (0.500 micro sec)

spitha0:
	ldi temp,0x40			;0b0100 0000
	out GIFR,temp			;Setting zero INTF0
	ldi r24,0x05
	ldi r25,0x00
	rcall wait_msec			;wait 5 msec
	in temp,GIFR			;Check if INTF0==1
	sbrc temp,6
	rjmp spitha0			;If INTF0==1 loop
	ret				;If INTF0==0 return
```