Εργαστήριο 1

Ονομα: Μαυρογιώργης Δημήτρης
ΑΜ: 2016030016
ΗΡΥ 411 - Ενσωματωμένα Συστήματα Μικροεπεξεργατών
Πολυτεχνειο Κρητης

April 19, 2021

Σκοπός

Σκοπός της πρώτης εργαστηριακής άσκησης είναι εξοικείωση με το περιβάλλον του Atmel Studio 7, στο οποίο γίνεται η ανάπτυξη μικροελεγκτών AVR. Πιο συγκεκριμένα, για την εξοικείωσή μας με το περιβάλλον ζητήθηκε να υλοποιήσουμε έναν χρονιστή διάρκειας 1 ms με δύο διαφορετικούς τρόπους: 1) με βρόγχο κατάλληλου αριθμού επαναλήψεων και 2) με τη χρήση timer/counter και interrupt του χρονιστή.

Υλοποίηση χρονιστή με βρόγχο επανάληψης

Για τη συγκεκριμένη υλοποίηση του χρονιστή έπρεπε να υπολογίσουμε τον αριθμό των κύκλων που πρέπει να περιμένει ο χρονιστής έτσι, ώστε να έχουμε την επιθυμητή διάρκεια του 1 ms. Ειδικότερα, για συχνότητα ρολογιού 10 MHz, ο αριθμός των κύκλων που πρέπει να περιμένει υπολογίζεται ως εξώς:

$$cycles = \frac{f_{clk}}{f_{1ms}} = \frac{10^7}{10^3} = 10^4 \quad κυκλοι$$

Επειδή, όμως οι καταχωρητές του μικροελεγκτή είναι των 8-bit, με το καταχωρητή-μετρητή μπορούμε να μετρήσουμε μέχρι το $2^8 - 1 = 255$. Γι' αυτό το λόγο θα χρειαστεί να χρησιμοποιήσουμε ένα διπλό loop, όπου στο εσωτερικό loop μετράμε απο 250 μέχρι το 0 και επαναλαμβάνουμε τις εντολές NOP, DEC και BRNE (συνολικά 4 κύκλοι), ενώ στο εξωτερικό από 10 μέχρι το 0. Οπότε με αυτό τον τρόπο επιτυγχάνουμε να καθυστερούμε για περίπου $250 * 4 * 10 = 10^4$ κύκλους, για την αρκίβεια λίγο παραπάνω από 10000 κύκλους (1 + (250 * 4 - 1 + 4) * 10 - 1 + 4 = 10034 κύκλοι). Για τον υπολογισμό του ακριβούς αριθμού των κύκλων αφαιρούμε 11 κύκλους, επειδή το BRNE στην τελευταία επανάληψη χρειάζεται 1 κύκλο. Τέλος, κατα τη διάρκεια αυτών των κύκλων εκτελέστηκαν οι εντολές LDI, NOP, BRNE, DEC και RET, οι οποίες συνολικά στο πλήθος είναι (1 + (250 * 3) + 3) * 10 + 1 = 7532 εντολές .

Επεξήγηση Κώδικα
Αρχικά, πρέπει να γίνει μία αρχικοποίηση του stack pointer στην αρχή της στοίβας, επειδή γίνεται κλήση συνάρτησης. Έπειτα, κάνουμε set το bit 0 του καταχωρητή DDRB, για να δηλώσουμε ότι το bit 0 του PORTB είναι έξοδος. Στη συνέχεια, κάνουμε clear το bit 0 του PORTB, ώστε να έχουμε αρχική κατάσταση του ακροδέκτη "0", και καλούμε τη ρουτίνα delay. Τέλος, μόλις επιστρέψει η ρουτίνα delay κάνουμε το bit 0 του PORTB "1" και καλούμε πάλι

στην delay, για να έχουμε την επιθυμητή καθυστέρηση του 1 ms.

Τα συγκεκριμένα βήματα τα εκτελούμε αενάως έτσι, ώστε να έχουμε εναλλαγή του ακροδέκτη από 0 σε 1 και το αντίστροφο.

Τέλος, όσον αφορά την ρουτίνα delay, η ιδέα της υλοποίησης του διπλού βρόγχου επανάληψης, καθώς και οι εντολές που εκτελούνται κατα τη διάρκεια του 1 ms, αναλύθηκαν παραπάνω.

```
.equ F_CPU=10000000    //Set clock freq to 10 MHz

ldi r16, high(ramend)   // Set stack pointer to the top of the ram
out sph, r16
ldi r16, low(ramend)
out spl, r16

sbi DDRB, 0 // Set PB0 as an output

start:
    cbi PORTB, 0        // Start with 0 output in pin PB0
    rcall delay         //rcall takes 3 cycles
    sbi PORTB, 0        //Set with 1 output in pin PB0
    rcall delay
    rjmp start
```

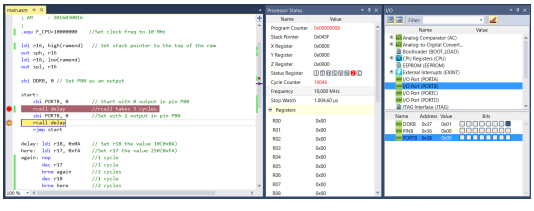
(a) Initialization of stack and registers and main program

```
delay: ldi r18, 0x0A    // Set r18 the value 10(0x0A)
here:  ldi r17, 0xFA    //Set r17 the value 250(0xFA)
again: nop              //1 cycle
       dec r17          //1 cycle
       brne again       //2 cycles
       dec r18          //1 cycle
       brne here        //2 cycles
       ret              //4 cycles
```

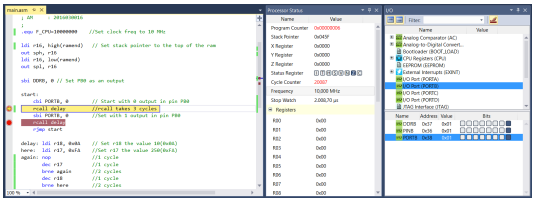
(b) Delay Subroutine for 1ms

Προσομοίωση Αποτελεσμάτων

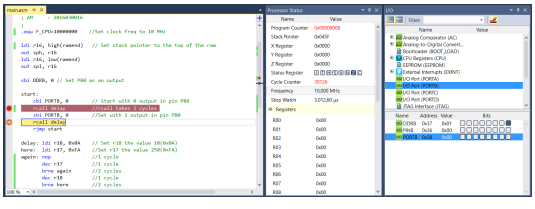
Στις δύο πρώτες εικόνες ((a) και (b)) παρατηρούμε ότι κατα τη διάρκεια του debugging ο καταχωρητης DDRB παίρνει τη σωστή τιμή 0x01, ενώ το bit 0 του PORTB που δηλώθηκε ως έξοδος αρχικοποιήθηκε με την τιμή "0" και παραμένει "0" για διάρκεια 1 ms. Μετά την κλίση της delay βλέπουμε οτι το bit 0 του PORTB αλλάζει από "0" σε "1" και κρατάει αυτή την τιμή για ακόμα 1 ms.
Στις, επόμενες δύο εικόνες ((c) και (d)) παρατηρούμε ότι συμβαίνει το ακριβώς αντίθετο, δηλαδή το PORTB0 αλλάζει από από "1" σε "0" και και ξανά από "0" σε "1". Τέλος, από την επιλογή Stop Watch στο παράθυρο Processor Status βλέπουμε ότι η καθυστέρηση σε κάθε περίπτωση που περιγράφτηκε είναι η επιθυμητή του 1 ms περίπου.



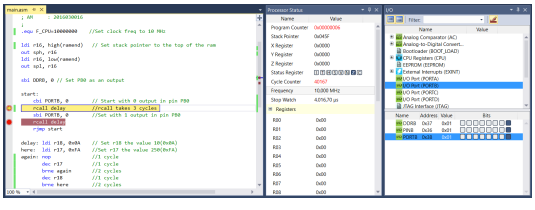
(a) Results from Atmel Studio 7 - PORTB0 is "0" for 1 ms



(b) Results from Atmel Studio 7 - PORTB0 changes from "0" to "1" and stays "1" for 1 ms



(c) Results from Atmel Studio 7 - PORTB0 changes from "1" to "0" and stays "0" for 1 ms



(d) Results from Atmel Studio 7 - PORTB0 changes from "0" to "1" and stays "1" for 1 ms

Υλοποίηση χρονιστή με τη χρήση timer/counter και interrupt

Για την υλοποίηση του χρονιστή με τη χρήση του timer/counter0 και interrupt, χρειάστηκε να υπολογίσουμε την τιμή αρχικοποίησης των καταχωρητών TCNT0, TCCR0 και TIMSK. Η τιμή αρχικοποίησης του καταχωρητή TCNT0 υπολογίστηκε ως εξής:

$$TCNT0\_VAL = MAX\_VAL - \frac{DELAY * FREQ}{PRESCALE} = 255 - \frac{1ms * 10MHz}{64} \approx 99(0x063)$$

Μετά από δοκιμές, η ελάχιστη τιμή του prescale που χρειαζόμαστε για να επιτύχουμε τη ζητούμενη καθυστέρηση είναι 64. Με δεδομένο αυτό, θα πρέπει να θέσουμε τα 3 LSB (CS02, CS01 και CS00)του καταχωρητή TCCR0 με την τιμή "011". Επιπλέον, επειδή θελουμε ο μετρητής να μετρά από το 99 ως το 255, θα πρέπει να αρχικοποιήσουμε τα bit WGM00 και WGM01 με την τιμή "0", η οποία υποδηλώνει ότι ο μετρητής τρέχει σε normal mode, δηλαδή προς τα πάνω. Τέλος, όσον αφορά τον καταχωρητή TIMSK, θα πρέπει να θέσουμε το bit TOIE0 με την τιμή "1" έτσι, ώστε να ενεργοποιήσουμε τα interrupt του timer/counter0. Συνεπώς, με βάση την παραπάνω ανάλυση η τιμή αρχικοποίησης του TCNT0 είναι η "0x063", του TCCR0 είναι η "0x03" και του TIMSK η "0x01".

Κατα τη διάρκεια εκτέλεσης του προγράμματος και μέχρι να έρθει το Interrupt από τον timer/counter0, θα έχουν περάσει 157 * 64 = 10048 κύκλοι, καθώς ο μετρητής μετράει από το 99 μέχρι το 255 (157 φορές) και κάθε φορά που αυξάνεται κατά 1 περνάνε 64 κύκλοι του ρολογιού. Επιπλέον, μέσα σε αυτύς τους κύκλους εκτελούνται 1 εντολή NOP (1 cycle) και ένα RJMP (2 cycles). Συνεπώς, το συνολικό πλήθος των εντολών που εκτελούνται είναι 3350 NOP και 3349 RJMP, δηλαδή 6699 εντολές.

Επεξήγηση Κώδικα

Αρχικά, ορίζουμε με την εντολή .org τη διεύθυνση που ξεκινάει το πρόγραμμα και τη διεύθυνση του interrupt handler. Έπειτα, αφού κάνουμε τις αρχικοποιήσεις των καταχωρητών, όπως αναλύθηκε παραπάνω, θέτουμε το bit 0 του DDRB "1", για να δηλώσουμε ότι το PORTB0 είναι έξοδος, ένω το αρχικοποιούμε με "0". Τέλος, για ενεργοποιήσουμε τα global interrupt καλούμε την εντολή SEI, η οποία θέτει το I-bit του Status Register με "1".

Όσον αφορά τον κύριο κορμό του προγράμματος, δηλαδή τη main, εκτελούμε εντολές NOP, μέχρι να έρθει το interrupt από τον timer/counter0 και να εκτελεστεί ο handler.

Τέλος, στον handler το μόνο που χρειάζεται να κάνουμε είναι να αλλάξουμε την τιμή της εξόδου PORTB0 και να αρχικοποιήσουμε πάλι τον καταχωρητή TCNT0 με την τιμή "0x63", ώστε να έχουμε σε κάθε κύκλο τη σωστή καθυστέρηση του 1 ms.

```
.equ F_CPU=10000000

.org 0x00           //Address of the beginning of the program
    rjmp start
.org 0x12           // Address ofthe interrupt handler
    rjmp handler

start:
    ldi r16, high(ramend)
    out sph, r16
    ldi r16, low(ramend)
    out spl, r16

    ldi r16, 0x63   // Set TCNT0 the value of 99(0x63) = 255 - (1 ms)*(1 MHz)/64(prescale)
    out TCNT0, r16

    ldi r16, 0x03   //Set prescale 64 - these bits of TCCR0 are set - CS02 = 0  CS01 = 1  CS00 = 1
    out TCCR0, r16

    ldi r16, 0x01   // TOIE0 is set to 1 in order to enable the timer overflow interrupt
    out TIMSK, r16

    sbi DDRB, 0     // Set PB0 as an output
    cbi PORTB, 0    //Initiate the output as 0

    sei     // Enable global interrupt

    rjmp main
```

(a) Initialization of stack and registers

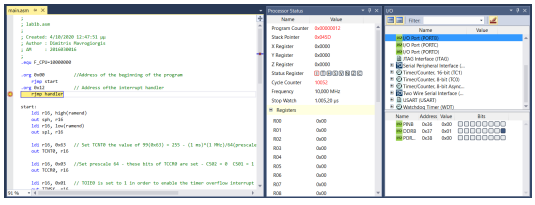
```
main:
    nop         //We are doing no op while we wait for the interrupt
    rjmp main

handler:
        in r16, PORTB
        com r16             //Set the opposite value in r16 - eg from 0 -> 1 or 1 -> 0
        andi r16, 0x01      //Set only bit 0 of r16 the opposite value and dont change the other
        out PORTB, r16

        ldi r16, 0x63       //Set once again the value of TCNT0 in order to roll again after the handling
        out TCNT0, r16
        reti
```

(b) Main program and interrupt handler

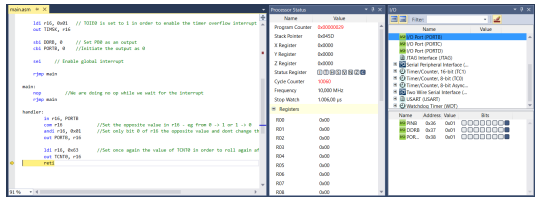
Προσομοίωση Αποτελεσμάτων

Με βάση τα παρακάτω αποτελέσματα προσομοίωσης, παρατηρούμε ότι ο καταχωρητής DDRB έχει πάρει τη σωστή τιμή "0x01" και ότι το PORTB0 έχει αρχικοποιηθεί με "0". Παράλληλα, από τις εικόνες φαίνεται ότι και το I-bit του Status Register έχει πάρει την κατάλληλη τιμή "1", που σημαίνει ότι έχουν ενεργοποιηθεί τα global interrupts.

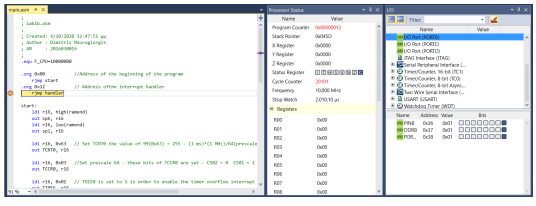
Επιπλέον, βλέπουμε ότι το PORTB0 έχει τιμή "0" για περίπου 1 ms, μέχρι που έρχεται το interrupt και εκτελείται ο handler, ο οποίος αλλάζει την τιμή του σε "1" (εικόνες (a) και (b)). Στις εικόνες (c) και (d), βλέπουμε ότι το PORTB0 παραμένει "1" για περίπου 1 ms, ενώ όταν έρχεται το interrupt η τιμή του γίνεται "0". Τέλος, όπως φαίνεται και στην εικόνα (e), οι καταχωρητές TCNT0, TCCR0 και TIMSK έχουν αρχικοποιηθεί και αυτοί με τις σωστές τιμές που αναφέρθηκαν κατα την ανάλυση του προγράμματος.



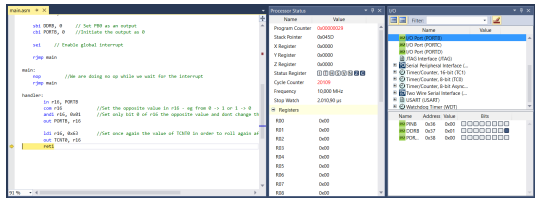
(a) Results from Atmel Studio 7 - PORTB0 is "0" for 1 ms



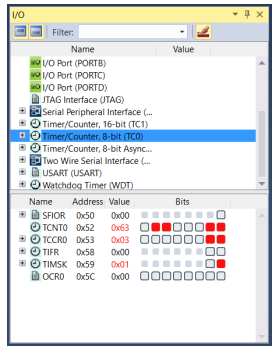
(b) Results from Atmel Studio 7 - Interrupt Handler Execution - PORTB0 changes from "0" to "1"



(c) Results from Atmel Studio 7 - PORTB0 is "1" for 1 ms



(d) Results from Atmel Studio 7 - Interrupt Handler Execution - PORTB0 changes from "1" to "0"



(e) Results from Atmel Studio 7 - Timer/Counter0 Register Initialization