Θεοδωρος Μπαρκας 2016030050 LAB1

Πρώτο σκέλος (Υλοποιιση οπως σε Arduino, counter με loop)

Το πρόβλημα που έχουμε είναι ότι δεν φτανει ενας Register των 8 Bits για να μετρήσουμε 10000 κύκλους που αντιστοιχούν σε 1ms.Αρα πρέπει να το σπάσουμε σε 2 loop που σίγουρα επαρκούν.
Στα comments τα LOOP1,LOOP2 αντιστοιχούν στο πλήθος που θα γίνουν τα LOOP δηλαδή τελικά στις τιμές των καταχωρητών όταν αυτοί αρχικοποιούνται (LDI).
Για Branch χρησιμοποιείτε το BRNE που όταν το flag Z παει στο 1 (δηλαδή η προηγούμενη εντολή DEC κάνει τον καταχωρητή ίσο με 0) βγαίνει από το LOOP .

EXEC_COUNT_LOOP1 counts CK cycles inside loop1 .

BRNE takes 2 cycles in general but when we dont have branch it takes only one.
ldi→LOOP1*1CK , dec→ LOOP1*1CK , and BNEQ→ LOOP1*2CK but for the last time it takes 1
so `EXEC_COUNT_LOOP1= LOOP1*4-1 + EXEC_COUNT_LOOP2`

EXEC_COUNT_LOOP2 counts CK cycles inside loop2

BRNE takes 2 cycles in general but when we dont have branch 1CK(loop1 times)
so `EXEC_COUNT_LOOP2= 3* LOOP1 * LOOP2 -LOOP1`

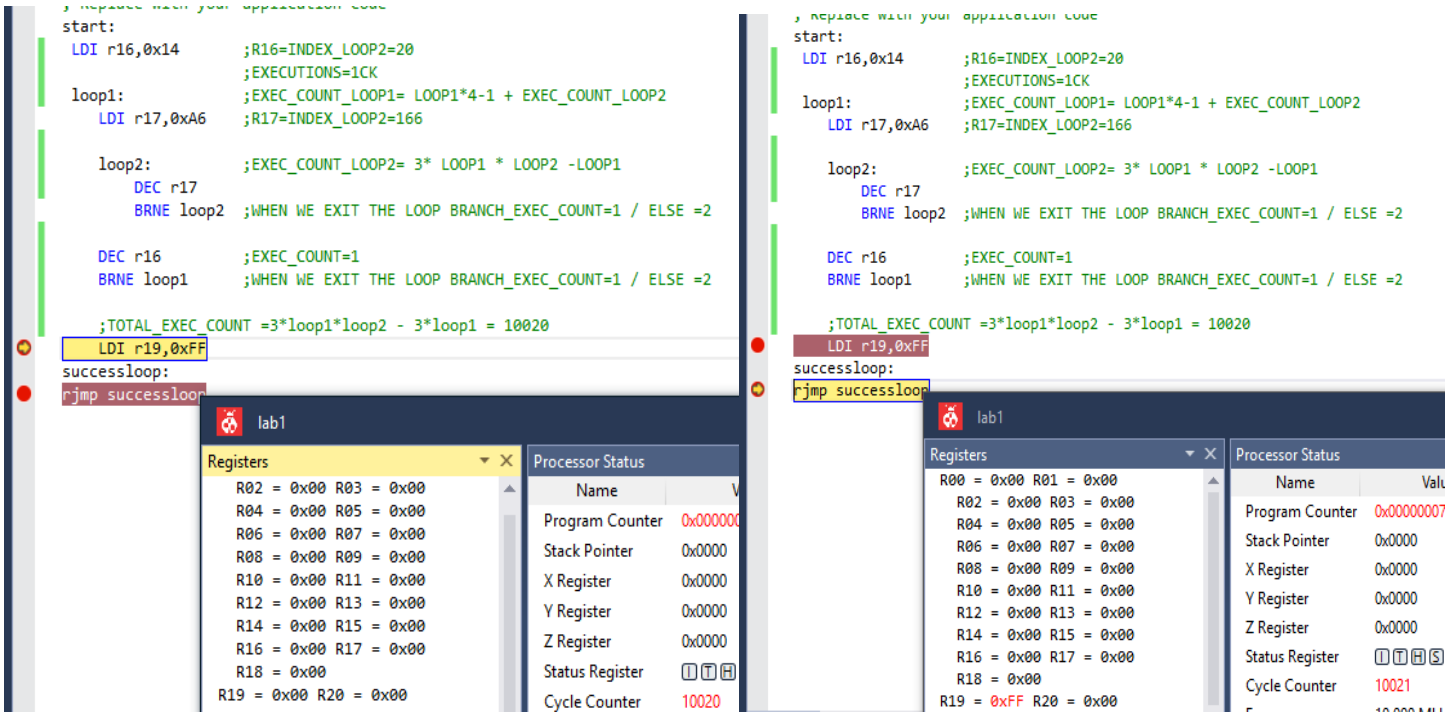
`Therefore EXEC_COUNT_LOOP1= LOOP1*3-1 + EXEC_COUNT_LOOP2=3*loop1*loop2-3*loop1 -1`

TOTAL_EXEC_COUNT counts CK cycles from Start to the last Comment.
Therefore:**`TOTAL_EXEC_COUNT`**`=EXEC_COUNT_LOOP1+1ck(for LDI)`
`TOTAL_EXEC_COUNT =3*loop1*loop2-3*loop1`

```
start:
 LDI r16,0x14

 loop1:
```

We now just find two values LOOP1, LOOP2 that are raletively close to 10000(because f_ck=10MHz)
LOOP1=20,LOOP2=166
TOTAL_EXEC_COUNT =3*loop1*loop2-3*loop1 = 10020



Ύστερα ο Καταχωρητής R19 που στη δικιά μου υλοποίηση πρέπει να ανοίγει αφού φτάνω στο 1ms γίνεται 0xFF .

<u>Δεύτερο σκέλος (Υλοποιιση με Interrupt)</u>

Ο Prescaler αλλάζει , τη συχνότητα με την οποία ο counter μετράει άρα κάνει f_counter= f_ck/ N επομένως :
f_interrupt = f_ck/(N(cycles counter does until it gives interrupt))→*
Έστω κατά αντιστοιχία : *f_int= f_ck/N*counter_ck* ***(1)***

(Στην αρχή το έκανα με τον τύπο της σελ 77 του Manual για το CTC mode:
άλλα βρήκα λάθος τιμή. Συγκεκριμένα τους μίσους Κύκλους.)

00). The waveform frequency is defin

$$f_{OCn} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

Θέλουμε το *f_int= f_ck/10000* έτσι ώστε το interrupt να γίνεται μετά από 1ms αφού f_ck=10MHz
Επομένως ο τύπος ***(1)*** γίνεται *:*
*f_ck/10000=f_ck/N*counter_ck* → **10000= N*counter_ck**

Προφανώς για να πετύχω τη μεγίστη ακρίβεια πρέπει να έχω όσο το δυνατόν μικρότερη δύναμη του 2 στο ρολόι που μπαίνει στον Counter .
Ο counter μεταβάλλεται με τιμή που διαμορφώνεται από τον prescaler του. Αρά θέλουμε όσο το δυνατόν μικρότερη τιμή του Prescaler **N→ MINIMUM**.
Η μεγίστη δυνατή τιμή του **counter_ck** (δηλαδή των πόσων κύκλων θα γίνουν μέχρι να δοθεί Interrupt) **counter_ck(max)=256** και αυτό επειδή ο Καταχωρητής του Counter είναι των 8 bit.
Επομένως για

ΘΕΛΩ 10000≃N*counter_ck
N=1 → 10000>>1*256
N=8 → 10000>>8*256
N=64 → 10000=64*counter_ck → counter_ck=156.25 ≃156

Prescaler_and TCCR0

N=64→ CS02..CS00 = 011=0x3

Τρέχω σε Normal Mode τον counter/timer και περιμένω Interrupt overflow
Επομένως ο **TCCR0** παίρνει την τιμή **0x03**

counter_ck and TCNT0

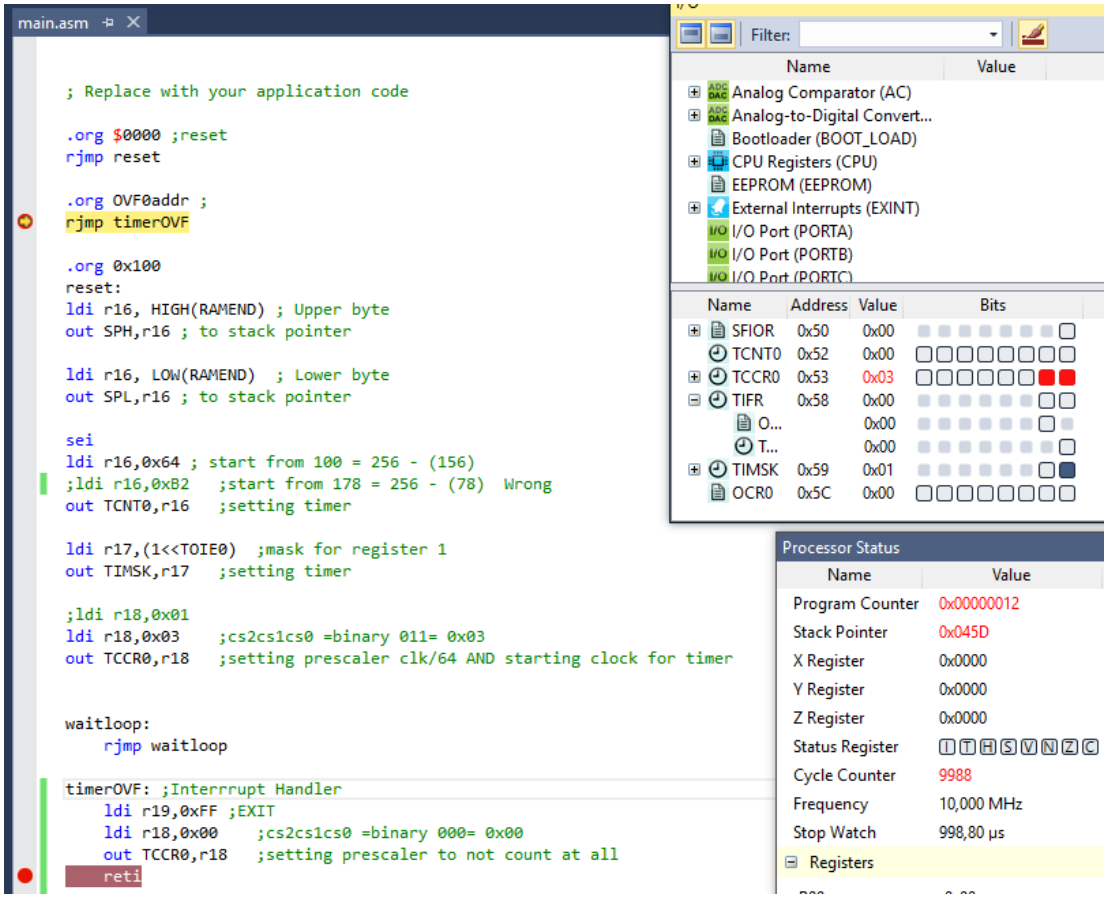
Επιλέγω να χρησιμοποιήσω το Interrupt για OVF επομένως αφού όπως υπολόγισα παραπάνω ο counter/timer μου πρέπει να κάνει 156 βήματα και λόγω normal mode ο counter ανεβαίνει αρχικοποιω τον Counter στη τιμή `100 = 256 – (156)` συγκεκριμένα ο Register TCNT0 παίρνει την τιμή 0x64 .

Interrupt Mask and sei

Για να δεχτώ το Interrupt επιπλέον τρέχω την sei και ενημερώνω τη μάσκα ότι περιμένω OVF του counter 0. Άρα TIMSK=(1<<TOIE0)

Interrupt Handler

Τέλος συμφωνά με την υλοποίηση μου στο Interrupt handler (timerOVF:) πέρα του ότι θέτω πάλι στον r19 την τιμή 0xFF σταματάω τον Counter/timer από το να τρέχει .Επομένως το πρόγραμμα ξαναμπαίνει στο waitloop άλλα αυτή τη φορά σε Infinite Loop.
Για να το κάνω αυτό πρέπει CS02..CS00 = 000 .
Επομένως θέτω των register TCCR0 ισο με 0x00 .



Παρατηρούμε ότι επειδή στρογγυλοποίησα προς τα κάτω $156.25 \simeq 156$ έχουμε πέσει λίγο πιο κάτω από το στόχο μας στα 0,9988 ms ενώ σε αυτό έχουν συμβάλει και οι εντολές για το setup του interrupt.