

## Θεόδωρος Μπαρκας 2016030050 LAB1

### Πρώτο σκέλος ( Υλοποίηση όπως σε Arduino, counter με loop )

Το πρόβλημα που έχουμε είναι ότι δεν φτάνει ένας 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  $\rightarrow$  LOOP1\*1CK , dec  $\rightarrow$  LOOP1\*1CK , and BNEQ  $\rightarrow$  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:  $TOTAL\_EXEC\_COUNT = EXEC\_COUNT\_LOOP1 + 1ck(\text{for LDI})$

$TOTAL\_EXEC\_COUNT = 3*loop1*loop2 - 3*loop1$

```
start:
    LDI r16,0x14

loop1:
```

We now just find two values LOOP1, LOOP2 that are relatively close to 10000 (because  $f_{ck}=10\text{MHz}$ )

LOOP1=20, LOOP2=166

TOTAL\_EXEC\_COUNT =  $3 \cdot \text{loop1} \cdot \text{loop2} - 3 \cdot \text{loop1} = 10020$

The image shows two side-by-side screenshots of an AVR assembly editor. Both screenshots display the same assembly code, which includes a loop structure with labels 'start:', 'loop1:', 'loop2:', and 'successloop:'. The code calculates a total execution count based on LOOP1=20 and LOOP2=166. Below the code, there are two panels: 'Registers' and 'Processor Status'. In the left screenshot, the 'Registers' panel shows R19 = 0xFF, and the 'Processor Status' panel shows the 'Cycle Counter' as 10020. In the right screenshot, the 'Registers' panel shows R19 = 0x00, and the 'Processor Status' panel shows the 'Cycle Counter' as 10021.

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

## Δεύτερο σκέλος ( Υλοποιηση με Interrupt )

Ο Prescaler αλλάζει , τη συχνότητα με την οποία ο counter μετράει άρα κάνει  $f_{counter} = f_{ck} / N$  επομένως :

$f_{interrupt} = f_{ck} / (N \cdot (\text{cycles counter does until it gives interrupt})) \rightarrow$

Έστω κατά αντιστοιχία :  $f_{int} = f_{ck} / N \cdot \text{counter\_ck} (1)$

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

00). The waveform frequency is defin

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

Example: f<sub>clk</sub> = 10 MHz, N = 16, OCRn = 100

Θέλουμε το  $f_{int} = f_{ck}/10000$  έτσι ώστε το interrupt να γίνεται μετά από 1ms αφού  $f_{ck}=10\text{MHz}$

Επομένως ο τύπος **(1)** γίνεται :

$$f_{ck}/10000 = f_{ck}/N * counter\_ck \rightarrow 10000 = N * counter\_ck$$

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

$$\Theta\epsilon\Lambda\Omega \quad 10000 \simeq N * counter\_ck$$

$$N=1 \rightarrow 10000 >> 1 * 256$$

$$N=8 \rightarrow 10000 >> 8 * 256$$

$$N=64 \rightarrow 10000 = 64 * counter\_ck \rightarrow counter\_ck = 156.25 \simeq 156$$

### **Prescaler and TCCR0**

$$N=64 \rightarrow CS02..CS00 = 011 = 0x3$$

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 αρχικοποιώ 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 ίσο με 0x00 .

main.asm

```

; Replace with your application code

.org $0000 ;reset
rjmp reset

.org OVFOaddr ;
rjmp timerOVF

.org 0x100
reset:
ldi r16, HIGH(RAMEND) ; Upper byte
out SPH,r16 ; to stack pointer

ldi r16, LOW(RAMEND) ; Lower byte
out SPL,r16 ; to stack pointer

sei
ldi r16,0x64 ; start from 100 = 256 - (156)
;ldi r16,0xB2 ;start from 178 = 256 - (78) Wrong
out TCNT0,r16 ;setting timer

ldi r17,(1<<TOIE0) ;mask for register 1
out TIMSK,r17 ;setting timer

;ldi r18,0x01
ldi r18,0x03 ;cs2cs1cs0 =binary 011= 0x03
out TCCR0,r18 ;setting prescaler clk/64 AND starting clock for timer

waitloop:
rjmp waitloop

timerOVF: ;Interrupt Handler
ldi r19,0xFF ;EXIT
ldi r18,0x00 ;cs2cs1cs0 =binary 000= 0x00
out TCCR0,r18 ;setting prescaler to not count at all
reti

```

Filter:

Name	Value
AD_CONVERTER	
AD_CONVERTER	
BOOTLOADER (BOOT_LOAD)	
CPU REGISTERS (CPU)	
EEPROM (EEPROM)	
EXTERNAL INTERRUPTS (EXINT)	
I/O PORT (PORTA)	
I/O PORT (PORTB)	
I/O PORT (PORTC)	

Name	Address	Value	Bits
SFIO	0x50	0x00	
TCNT0	0x52	0x00	
TCCR0	0x53	0x03	
TIFR	0x58	0x00	
O...		0x00	
T...		0x00	
TIMSK	0x59	0x01	
OCR0	0x5C	0x00	

Processor Status

Name	Value
Program Counter	0x00000012
Stack Pointer	0x045D
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	11111111
Cycle Counter	9988
Frequency	10,000 MHz
Stop Watch	998,80 μs
Registers	

Παρατηρούμε ότι επειδή στρογγυλοποιήσα προς τα κάτω  $156.25 \approx 156$  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.