Πρώτη Αναφορά Εργαστηρίου Μικροϋπολογιστών

ΕΜΜΑΝΟΥΗΛ ΞΕΝΟΣ (AM):03120850 ΠΑΠΑΔΟΠΟΥΛΟΣ ΣΠΥΡΙΔΩΝ (AM): 03120033

Ζήτημα Πρώτο

Στο πρώτο ζήτημα μας ζητήθηκε να δημιουργήσουμε μία συνάρτηση wait_x_msec η οποία θα υλοποιεί ένα ακριβές χρονόμετρο σε msec (από 1 έως 4095 msec). Για την υλοποίηση της συνάρτησης αυτής θεωρήσαμε, ότι ο χρήστης φορτώνει τα ζητούμενα msec της καθυστέρησης στους καταχωρητές r24(Low byte) r25(High byte). Επιπλέον θεωρήσαμε πως οι τιμές των r24 και των r25 μπορεί να αλλάξουν κατά την εκτέλεση της συνάρτησης(συνεπώς είναι caller-save). Στην συνέχεια παραθέτουμε τον ζητούμενο κώδικα με σχόλια, σε όποια σημεία κρίνεται απαραίτητο.

```
.include "m328PBdef.inc"
.equ FOSC MHZ=16
                   ;MHz
.equ DEL mS=255
set1:
   ldi r24, low(DEL_mS)
   ldi r25, high(DEL mS)
   rcall wait_x_msec
    rjmp set1
delay_inner:
          r23, 242 ; (1 cycle)
loop3:
   dec r23
                         ; 1 cycle
                      ; 1 cycle
                        ; 1 or 2 cycles
   brne loop3
                      ; 1 cycle
    ret
                      ; 4 cycles
;this routine is used to produce a delay of (980*Delay+14) cycles
wait x msec:
   push r24
                      ; (2 cycles)
   push r25
                      ; (2 cycles) Save r24:r25
loop4:
    rcall delay_inner
                          ; (3+973)=976 cycles
    sbiw r24 ,1
                         ; 2 cycles
   brne loop4
                        ; 1 or 2 cycles
```

```
extra:
    ;we need 20(delay-1) more cycles
    pop r25
                       ; (2 cycles)
    pop r24
                       ; (2 cycles) Restore r24:r25
    sbiw r24 ,1
                           ; 2 cycles
loop5:
                     ; 20 cycles
    sbiw r24 ,1
                           ; 2 cycles
    brne loop5
                         ; 1 or 2 cycles
    ret
                       ;4 cycles
```

Για την δημιουργία της συνάρτησης υπολογίσαμε επακριβώς του κύκλους που παίρνει κάθε εντολή προκειμένου να προκύψει το ζητούμενο χρονόμετρο. Στο delay inner εκτελείται r23 φορές ένα loop που παίρνει 4 κύκλους για κάθε εκτέλεση του(το τελευταίο branch παίρνει 1 κύκλο, αλλά αναπληρώνεται από την nop που έχουμε προσθέσει αμέσως μετά). Το r23 το θέσαμε στην τιμή 242 προκειμένου η καθυστέρηση που εκτελεί αυτό το Loop να είναι μικρότερη της ζητούμενης . Με βάση αυτό , μέχρι το σημείο που υπάρχει η ετικέτα extra έχουν εκτελεστεί 980*Delay+14 κύκλοι(συμπεριλαμβάνεται το ret και οι τέσσερις nop στο τέλος). Εμείς για συχνότητα 1Mhz θέλουμε να εκτελούνται 1000*Delay κύκλοι. Συνεπώς το κομμάτι extra θα πρέπει να εκτελεί τους υπολειπόμενος κύκλους που είναι 20*Delay -14. Στο extra προσθέτουμε αρχικά 2 pop και μία sbiw, προκειμένου να απαιτούνται πλέον 20(Delay-1) κύκλοι και να επαναφέρουμε το delay στους καταχωρητές r24,r25. Έτσι για να εκτελέσουμε τους εναπομείναντες κύκλους εκτελούμε ένα loop, το οποίο εκτελεί 20 κύκλους. Προφανώς αυτό το loop πρέπει να εκτελεστεί Delay-1 φορές. Εφόσον εκτελεστεί αυτό το loop πλέον έχουμε τη ζητούμενη καθυστέρηση(συμπεριλαμβάνοντας τις 4 nop και το ret στο τέλος)και συνεπώς επιστρέφουμε στο κύριο πρόγραμμα. Παραθέτουμε και δύο εικόνες από την εκτέλεση στην προσομοίωση για δύο διαφορετικά delay.

```
:.include "m328PBdef.inc"
 2
         .equ FOSC MHZ=16
         .equ DEL mS=255
 3
                                ;mS
 4
 5
 6
 7
         set1:
 8
              ldi r24, low(DEL mS)
 9
              ldi r25, high (DEL mS)
 rcall wait_x_msec
 rjmp setl
12
13
14
15
16
17
18
         delay_inner:
19
              ldi
                    r23, 242
                                      ; (1 cycle)
20
         loop3:
21
              dec r23
                                         ; 1 cycle
22
              nop
                                     ; 1 cycle
23
              brne loop3
                                         ; 1 or 2 cycles
24
              nop
                                     ; 1 cycle
Stopwatch × Output
                      Call Stack
                                   Breakpoints
     Target halted. Stopwatch cycle count = 2 (2 µs)
    Target halted. Stopwatch cycle count = 255000 (255 ms)
    Target halted. Stopwatch cycle count = 4(4 \mu s)
    Target halted. Stopwatch cycle count = 255000 (255 ms)
III
(1)
```

```
:.include "m328PBdef.inc"
 1
 2
         .equ FOSC MHZ=16
                                  :MHz
 3
         .equ DEL mS=71
                            ;mS
 4
 5
 6
 7
         set1:
              ldi r24, low(DEL mS)
 8
 9
              ldi r25, high (DEL mS)
rcall wait x msec
rjmp setl
12
13
14
15
16
17
18
         delay inner:
19
              ldi
                      r23, 242 ; (1 cycle)
20
         loop3:
21
              dec r23
                                           ; 1 cycle
22
              nop
                                     ; 1 cycle
23
                                         ; 1 or 2 cycles
              brne loop3
24
              nop
                                      ; 1 cycle
Stopwatch × Output
                       Call Stack
                                     Breakpoints
    Target halted. Stopwatch cycle count = 4 (4 µs)
    Target halted. Stopwatch cycle count = 71000 (71 ms)
    Target halted. Stopwatch cycle count = 4(4 \mu s)
   Target halted. Stopwatch cycle count = 71000 (71 ms)
    Target halted. Stopwatch cycle count = 4 (4 µs)
    Target halted. Stopwatch cycle count = 71000 (71 ms)
```

Ζήτημα Δεύτερο

Στο δεύτερο ζήτημα υλοποιήθηκε κώδικας assembly AVR για τον υπολογισμό των παρακάτω λογικών συναρτήσεων:

```
F0 = (A' \cdot B' \cdot C' + D)'
F1= (A'+C) \cdot (B'+D')
```

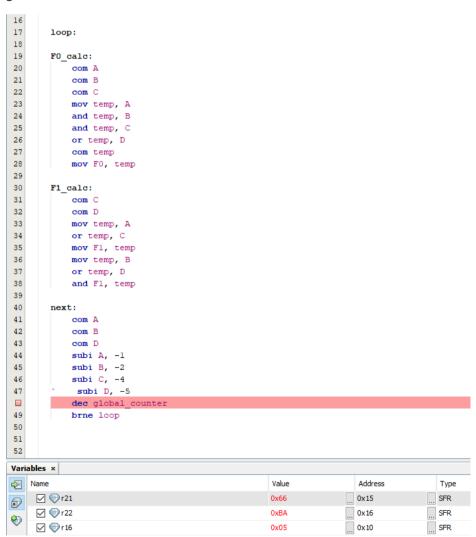
πέντε φορές με βάση κάποιες αρχικές τιμές των μεταβλητών A, B, C, D οι οποίες έχουν μία σταθερή αύξηση σε κάθε επανάληψη. Ακολουθεί ο κώδικας καθώς και η επεξήγηση αυτού.

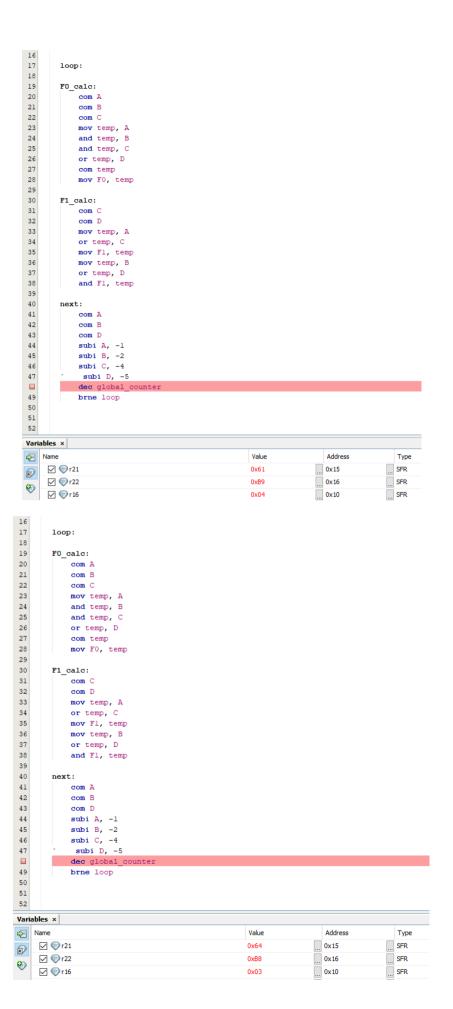
```
.include "m328PBdef.inc"
.def global_counter=r16
.def A=r17
.def B=r18
.def C=r19
```

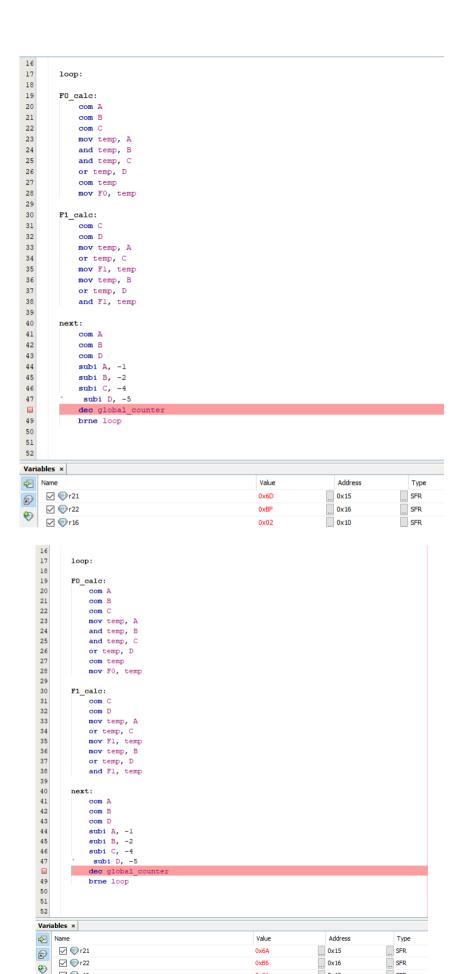
```
.def D=r20
.def F0=r21
.def F1=r22
.def temp=r23
ldi global_counter, 5
ldi A, 0x45
ldi B, 0x23
ldi C, 0x21
ldi D, 0x01
loop:
F0_calc:
    com A
    com B
    com C
   mov temp, A
   and temp, B
   and temp, C
   or temp, D
   com temp
   mov F0, temp
F1_calc:
   com C
    com D
   mov temp, A
   or temp, C
   mov F1, temp
   mov temp, B
   or temp, D
   and F1, temp
next:
   com A
    com B
   com D
   subi A, -1
   subi B, -2
   subi C, -4
    subi D, -5
    dec global_counter
   brne loop
```

Αρχικά για ευκολία ορίζουμε κάποιους καταχωρητές με τα ονόματα των μεταβλητών και του αποτελέσματος κάθε λογικής συνάρτησης για ευκολία. Επίσης ονομάζουμε ένα καταχωρητή temp που θα τον χρησιμοποιούμε για την αποτίμηση του αποτελέσματος. Έπειτα φορτώνουμε στις μεταβλητές τις αρχικές τιμές τους που ορίζονται στην εκφώνηση της άσκησης. Στην συνέχεια υπολογίζουμε αρχικά το αποτέλεσμα της F0 και έπειτα το αποτέλεσμα της F1 κάνοντας τις απαραίτητες λογικές πράξεις (αντιστροφές, and, or). Ο υπολογισμός γίνεται αρχικά χρησιμοποιώντας τον καταχωρητή temp και στην συνέχεια μεταφέρεται στον αντίστοιχο καταχωρητή. Στο τέλος κάθε επανάληψης επαναφέρουμε τις τιμές των μεταβλητών που παρέμειναν αντεστραμμένες στην κανονική τους μορφή. Για να τις μεταβλητές όπως ζητείται στην άσκηση αφαιρούμε τον αντίστοιχο αρνητικό αριθμό (έτσι έχουμε πχ για την μεταβλητή Β Β=Β-(-2)). Για να κάνουμε ακριβώς πέντε επαναλήψεις κρατάμε έναν global counter, αρχικά με την τιμή 5, και στο τέλος κάθε επανάληψης τον μειώνουμε κατά 1. Αν στο τέλος μίας επανάληψης ο global_counter δεν είναι 0 τότε δεν θα είναι ένα το zero flag οπότε με την εντολή brne loop το PC θα μεταβεί στην ετικέτα loop. Όταν o global_counter είναι Ο τότε δεν θα γίνει το προαναφερθέν branch και το πρόγραμμα θα τερματιστεί.

Ακολουθούν στιγμιότυπα εκτέλεσης του προγράμματος με τις τιμές των F0 και F1 και του global counter.







0xB6

0x01

... 0x10

SFR

€

Έπειτα από την τελευταία εκτέλεση δεν έχουμε καμία αλλαγή στην τιμή του global counter και των F0, F1.

Ζήτημα Τρίτο

Στο τρίτο ζήτημα καλούμαστε να υλοποιήσουμε αυτοματισμό βαγονέτου που κινείται συνεχώς δεξιά και αριστερά ξεκινώντας από δεξιά. Μεταξύ κάθε εναλλαγής θέσης του βαγονέτου μεσολαβεί περίπου 1.5 sec και κάθε φορά που αλλάζει κατεύθυνση (δηλαδή στις ακραίες θέσης) προστίθεται καθυστέρηση 2 sec φτάνοντας την συνολική καθυστέρηση στα 3.5 περίπου sec. Η κατεύθυνση της κίνησης φαίνεται στο T flag SREG. Όταν το T flag είναι 1 τότε το βαγόνι κινείται προς τα αριστερά, ενώ όταν είναι 0 κινείται προς τα δεξιά (κατά σύμβαση). Παρατίθεται ο κώδικας που υλοποιεί τα παραπάνω:

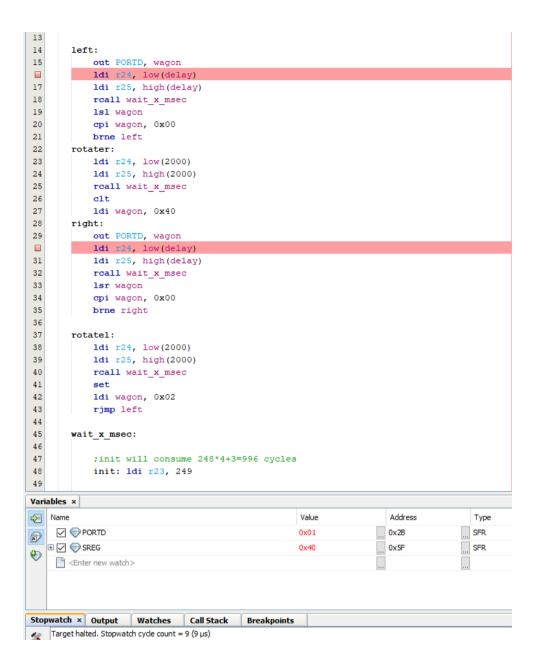
```
.include "m328PBdef.inc"
.def wagon=r16
ldi wagon, LOW(RAMEND)
out SPL, wagon
ldi wagon,HIGH(RAMEND)
out SPH, wagon
ser wagon
out DDRD, wagon
ldi wagon, 0x01
set
left:
    out PORTD, wagon
    ldi r24, low(1500)
    ldi r25, high(1500)
    rcall wait_x_msec
    1sl wagon
    cpi wagon, 0x00
    brne left
rotater:
    ldi r24, low(2000)
    ldi r25, high(2000)
    rcall wait_x_msec
    clt
    ldi wagon, 0x40
right:
    out PORTD, wagon
    ldi r24, low(1500)
    ldi r25, high(1500)
    rcall wait_x_msec
    1sr wagon
    cpi wagon, 0x00
```

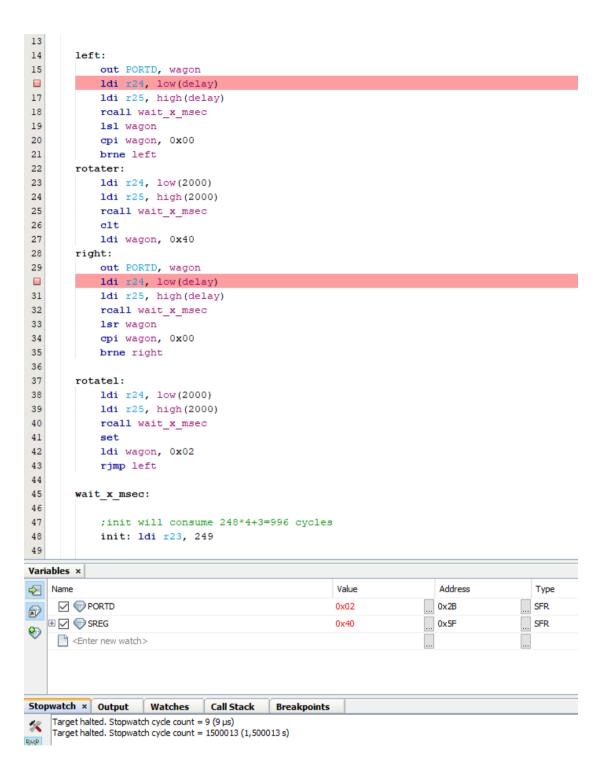
```
brne right
rotatel:
   ldi r24, low(2000)
   ldi r25, high(2000)
   rcall wait_x_msec
   set
   ldi wagon, 0x02
    rjmp left
delay_inner:
         r23, 242 ; (1 cycle)
    loop3:
   nop
brne loop3
    ret
                      ; 4 cycles
    ;this routine is used to produce a delay of (980*Delay+14) cycles
   wait_x_msec:
    push r24
                      ; (2 cycles)
    push r25 ; (2 cycles) Save r24:r25
    loop4:
   extra:
    ;we need 20(delay-1) more cycles
   pop r25 ; (2 cycles)
pop r24 ; (2 cycles)
sbiw r24 ,1 ; 2 cycles
sbiw r24 ,1 ; 2 cycles
sbiw r24 ,1 ; 2 cycles
```

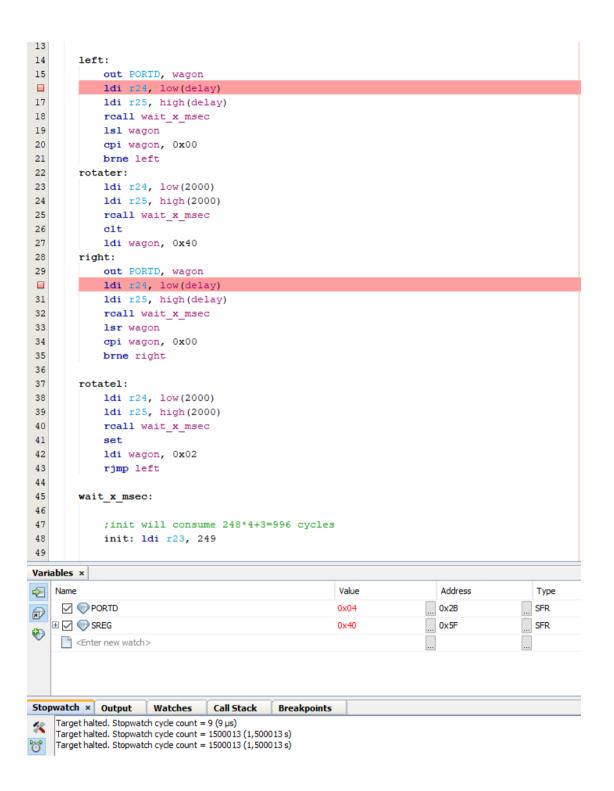
```
nop
nop
nop
nop
brne loop5 ; 1 or 2 cycles
nop
nop
nop
nop
nop
ret ;4 cycles
```

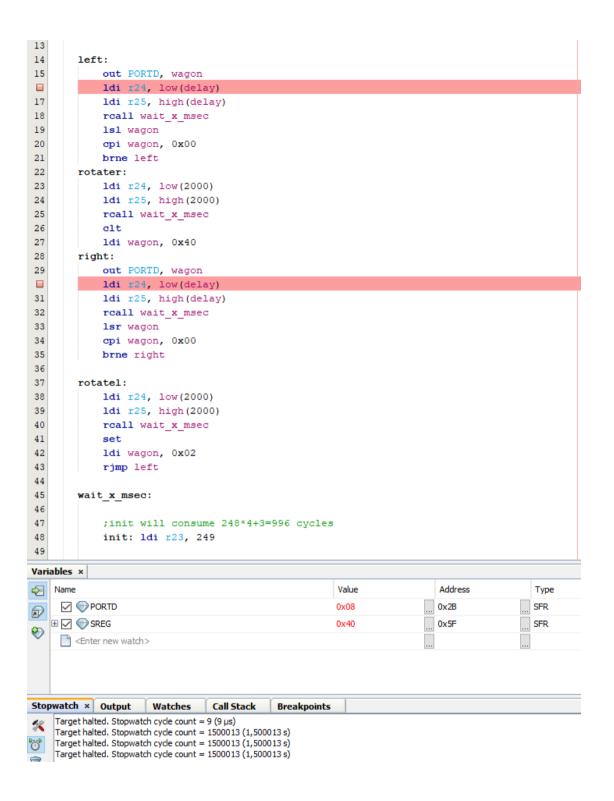
Αρχικά ορίζουμε τον stack pointer, καθώς θα τον χρειαστούμε για να χρησιμοποιήσουμε την καθυστέρηση ως ρουτίνα, και θέτουμε την θύρα PORTD ως έξοδο. Επίσης ορίζουμε την αρχική θέση του βαγονιού ως την δεξιότερη μέσω του ldi wagon, 0x01 και θέτουμε το T flag ίσο με 1 (σύμφωνα με την σύμβαση που αναφέραμε προηγουμένως) μέσω της εντολής set. Έπειτα εξάγουμε την θέση του βαγονιού στην PORTD και βάζουμε στο ζεύγος καταχωρητών r24, r25 το 1500 που συμβολίζει τα msec που θέλουμε να υποβάλουμε καθυστέρηση (ίδια διαδικασία με το ζήτημα 1) και καλούμε την wait x msec ώστε να γίνει η επιθυμητή καθυστέρηση. Στην συνέχεια κάνουμε αριστερό shift την θέση του wagon και ελέγχουμε αν η τιμή του καταχωρητή r16 (που συμβολίζει την θέση του βαγονιού) είναι 0. Σε περίπτωση που δεν είναι 0 σημαίνει ότι δεν έχουμε φτάσει στην αριστερότερη θέση και επιστρέφουμε στην ετικέτα left. Αν είναι 0 τότε σημαίνει ότι είμαστε στην αριστερότερη θέση και πρέπει να αλλάξουμε κατεύθυνση επιτελώντας τις πρόσθετες κινήσεις που αναφέρονται στην άσκηση. Αρχικά επιβάλλουμε την πρόσθετη καθυστέρηση ίση με 2 sec, θέτουμε το T flag ίσο με 0 (μέσω της εντολής clt) και τοποθετούμε το βαγόνι στην δεύτερη αριστερότερη θέση μέσω της εντολής ldi wagon, 0x40. Η συνέχεια για την κίνηση προς τα δεξιά και την αλλαγή κατεύθυνσης προς τα αριστερά είναι εντελώς ανάλογη με την διαδικασία που περιγράφηκε προηγουμένως. Μόλις επιτευχθεί η αλλαγή κατεύθυνσης προς τα αριστερά το πρόγραμμα μεταβαίνει στην ετικέτα left και ξεκινάει να κινείται προς τα αριστερά. Έτσι το πρόγραμμα «τρέχει»

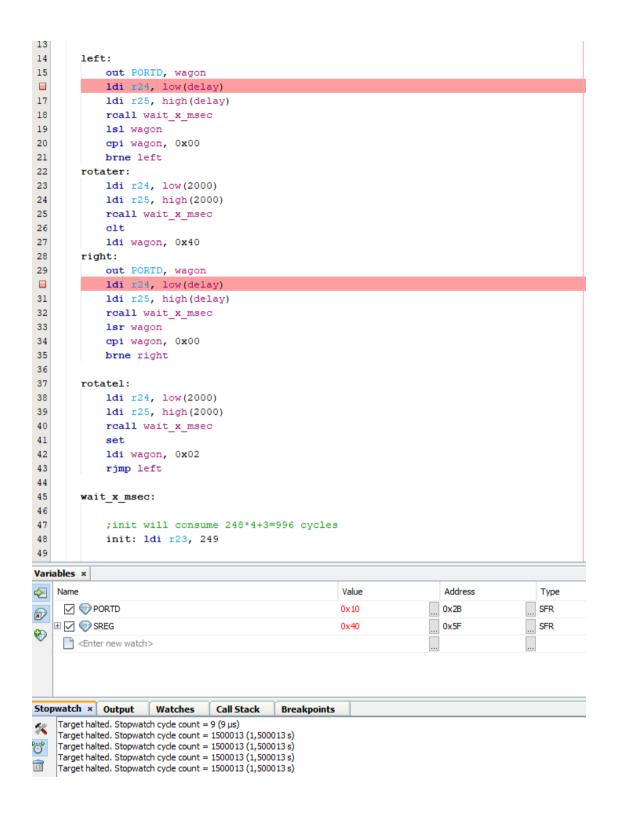
Ακολουθούν στιγμιότυπα της προσομοίωσης την κίνησης του βαγονέτου αριστερά και η περιστροφή που γίνεται στα δεξιά. Σε αυτά φαίνονται η θέση του βαγονέτου, ο καταχωρητής του SREG, του οποίου το έβδομο bit είναι το T flag, καθώς και το χρονόμετρο.











```
13
14
        left:
15
             out PORTD, wagon
 ldi r24, low(delay)
17
             ldi r25, high(delay)
18
             rcall wait_x_msec
19
             lsl wagon
20
             cpi wagon, 0x00
21
             brne left
22
         rotater:
23
             ldi r24, low(2000)
             ldi r25, high(2000)
24
25
             rcall wait_x_msec
26
             clt
27
             ldi wagon, 0x40
        right:
28
             out PORTD, wagon
29
ldi r24, low(delay)
31
             ldi r25, high(delay)
32
             rcall wait_x_msec
33
             lsr wagon
34
             cpi wagon, 0x00
35
             brne right
36
37
        rotatel:
             ldi r24, low(2000)
38
39
             ldi r25, high(2000)
40
             rcall wait_x_msec
41
             set
42
             ldi wagon, 0x02
43
             rjmp left
44
45
        wait_x_msec:
46
47
             ;init will consume 248*4+3=996 cycles
48
             init: ldi r23, 249
49
Variables ×
Name
                                                         Value
                                                                            Address
                                                                                              Туре
     ✓ ⊕ PORTD
                                                                                              SFR
                                                         0x20
                                                                            0x2B
    0x40
                                                                            0x5F
                                                                                              SFR
⊕
     <Enter new watch>
                     Watches
Stopwatch × Output
                                Call Stack
                                             Breakpoints
    Target halted. Stopwatch cycle count = 9 (9 µs)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
***
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
```

```
13
14
         left:
15
             out PORTD, wagon
 ldi r24, low(delay)
17
             ldi r25, high(delay)
18
             rcall wait_x_msec
19
             1s1 wagon
20
             cpi wagon, 0x00
21
             brne left
22
         rotater:
23
             ldi r24, low(2000)
24
             ldi r25, high(2000)
25
             rcall wait_x_msec
             clt
26
27
             ldi wagon, 0x40
28
         right:
             out PORTD, wagon
29
 ldi r24, low(delay)
31
             ldi r25, high(delay)
32
             rcall wait_x_msec
33
             lsr wagon
34
             cpi wagon, 0x00
35
             brne right
36
37
         rotatel:
             ldi r24, low(2000)
38
             ldi r25, high(2000)
39
40
             rcall wait_x_msec
41
             set
             ldi wagon, 0x02
42
43
             rjmp left
44
45
         wait x msec:
46
47
             ;init will consume 248*4+3=996 cycles
48
             init: 1di r23, 249
49
Variables ×
    Name
                                                                              Address
                                                                                                Туре
₽
     ✓ ⊕ PORTD
                                                                                                SFR
                                                          0x40
                                                                              0x2B
⊕ ✓ ⊕ SREG
                                                          0x40
                                                                              0x5F
                                                                                                SFR
0
     <Enter new watch>
                                 Call Stack
Stopwatch × Output
                      Watches
                                              Breakpoints
    Target halted. Stopwatch cycle count = 9 (9 µs)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
⊕
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
Ш
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
```

```
13
14
         left:
15
              out PORTD, wagon
ldi r24, low(delay)
17
              ldi r25, high(delay)
18
             rcall wait_x_msec
19
             lsl wagon
             cpi wagon, 0x00
20
21
             brne left
22
         rotater:
23
             ldi r24, low(2000)
24
             ldi r25, high(2000)
             rcall wait_x_msec
25
26
             clt
27
             ldi wagon, 0x40
28
         right:
             out PORTD, wagon
29
ldi r24, low(delay)
31
             ldi r25, high(delay)
32
              rcall wait_x_msec
33
             lsr wagon
34
             cpi wagon, 0x00
35
             brne right
36
37
         rotatel:
             ldi r24, low(2000)
38
39
             ldi r25, high(2000)
40
             rcall wait x msec
41
             set
             ldi wagon, 0x02
42
43
             rjmp left
44
45
         wait x msec:
46
              ;init will consume 248*4+3=996 cycles
47
             init: ldi r23, 249
48
49
Variables ×
₽
    Name
                                                           Value
                                                                               Address
                                                                                                 Туре
     ✓ ⊕ PORTD
                                                           0x80
                                                                               0x2B
                                                                                                 SFR
SFR
                                                           0x54
                                                                               0x5E
(P)
     <Enter new watch>
Stopwatch × Output
                      Watches
                                 Call Stack
                                              Breakpoints
    Target halted. Stopwatch cycle count = 9 (9 µs)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
⊕
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
     Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
```

```
1
         .def wagon=r16
 2
         .equ delay=1*1500
 3
  4
         ldi wagon, LOW (RAMEND)
  5
         out SPL, wagon
  6
         ldi wagon, HIGH (RAMEND)
  7
         out SPH, wagon
  8
 9
         ser wagon
 10
         out DDRD, wagon
11
         ldi wagon, 0x01
12
         set
13
14
         left:
15
              out PORTD, wagon
 ldi r24, low(delay)
17
              ldi r25, high(delay)
              rcall wait x_msec
18
19
              lsl wagon
              cpi wagon, 0x00
20
21
              brne left
22
         rotater:
23
              ldi r24, low(2000)
24
              ldi r25, high(2000)
25
              rcall wait_x_msec
26
              clt
27
              ldi wagon, 0x40
28
         right:
              out PORTD, wagon
29
               ldi r24, low(delay)
 31
              ldi r25, high(delay)
              rcall wait_x_msec
32
33
              lsr wagon
34
              cpi wagon, 0x00
35
              brne right
36
37
         rotatel:
Variables ×
                                                               Value
                                                                                    Address
₽
    Name
                                                                                                       Type
      ✓ ⊕ PORTD
                                                              0x40
                                                                                   0x2B
                                                                                                       SFR
0x02
                                                                                   0x5F
                                                                                                       SFR
(P)
      <Enter new watch>
Stopwatch × Output
                        Watches
                                    Call Stack
                                                Breakpoints
    Target halted. Stopwatch cycle count = 9 (9 µs)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
...
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
Target halted. Stopwatch cycle count = 3500022 (3,500022 s)
```

```
ldi wagon, LOW (RAMEND)
 5
         out SPL, wagon
 6
         ldi wagon, HIGH (RAMEND)
 7
         out SPH, wagon
 8
 9
         ser wagon
10
         out DDRD, wagon
11
         ldi wagon, 0x01
12
         set
13
         left:
14
15
              out PORTD, wagon
ldi r24, low(delay)
17
              ldi r25, high(delay)
              rcall wait_x_msec
18
19
              lsl wagon
20
              cpi wagon, 0x00
21
              brne left
22
         rotater:
              ldi r24, low(2000)
23
24
              ldi r25, high(2000)
              rcall wait_x_msec
25
              clt
26
27
              ldi wagon, 0x40
28
         right:
29
              out PORTD, wagon
              ldi r24, low(delay)
31
              ldi r25, high(delay)
              rcall wait_x_msec
32
33
              lsr wagon
34
              cpi wagon, 0x00
35
              brne right
36
37
         rotatel:
Variables ×
    Name
                                                             Value
                                                                                  Address
                                                                                                     Type
     ✓ ⊕ PORTD
                                                             0x20
                                                                                 0x2B
                                                                                                     SFR
0x00
                                                                                 0x5F
                                                                                                     SFR
•
     <Enter new watch>
Stopwatch × Output
                       Watches
                                   Call Stack
                                               Breakpoints
    Target halted. Stopwatch cycle count = 9 (9 µs)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
•
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
    Target halted. Stopwatch cycle count = 3500022 (3,500022 s)
    Target halted. Stopwatch cycle count = 1500013 (1,500013 s)
```