Μικροεπεξεργαστές και Περιφερειακά

Αναφορά άσκησης εργαστηρίου

Group A – άσκηση με σηματοδότη

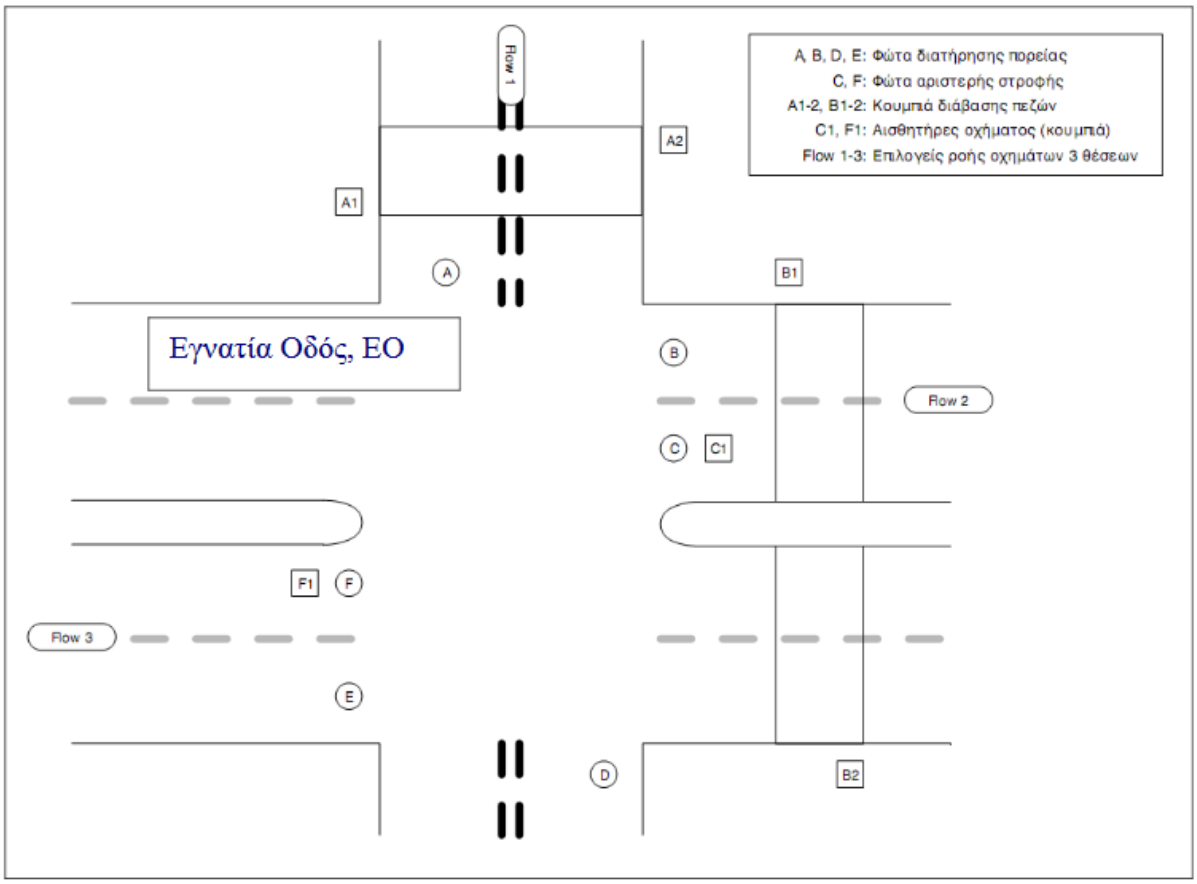
Ομάδα 5:
Θωμάς Πλιάκης tpliakis@ece.auth.gr **9018**

Χρήστος Παπακωνσταντίνου papachri@ece.auth.gr **8531**

Contents

1 Λογική Καταστάσεων

Στην άσκηση αυτή μας ζητήθηκε να υλοποιήσουμε το σύστημα που θα ελέγχει τον σηματοδότη για την κυκλοφορεία της εθνκής οδού και του καθέτου δρόμου της.Το σχηματικό διάγραμμα φαίνεται παρακάτω.



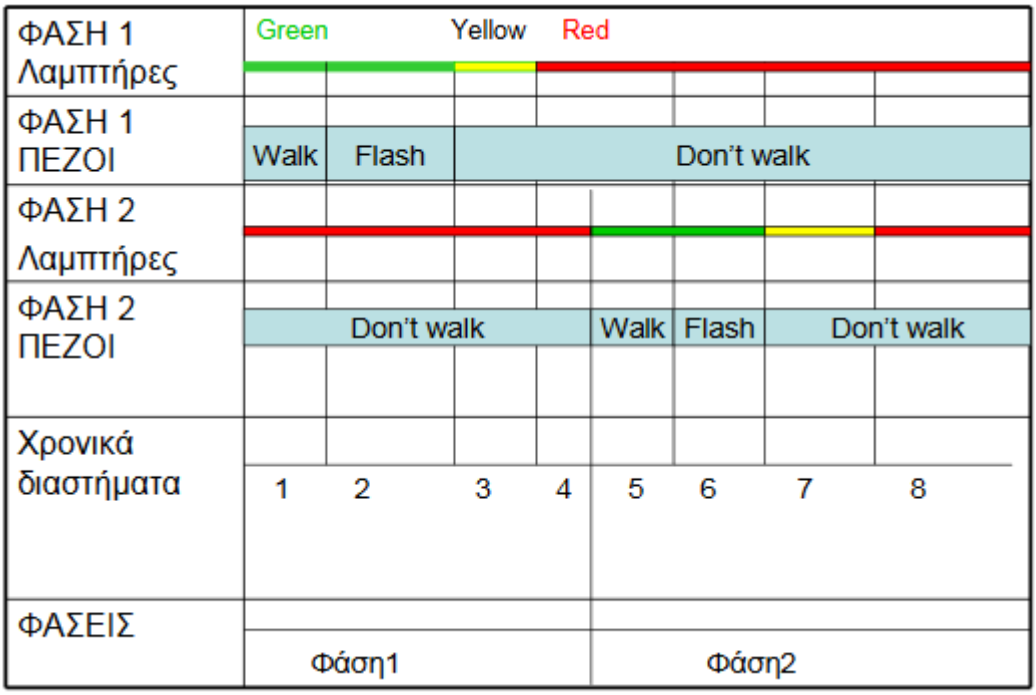
1.1 Οι καταστάσεις

Αρχικά η διαδικασία κωδικοποιήθηκε σαν μία μηχανή πεπερασμένων καταστάσεων, σύμφωνα με το μοντέλο του Moore. Σε αυτό οι έξοδος είναι συνάρτηση μόνο της παρούσας κατάστασης. Σαν εισόδους θεωρήσαμε :

- Το σήμα από την ρουτίνα του χρονιστή
- Το πάτημα κάποιου πλήκτρου

Η έξοδος είναι το χρώμα των σηματοδοτών.

Μπορούμε εύκολα να εξάγουμε τις καταστάσεις του συστήματος με τη βοήθεια του παρακάτω πίνακα.



Πινακας1

Οι καταστάσεις που θεωρήσαμε είναι οι εξής(Τα ονόματά τους είναι έτσι όπως αναφέρονται και στον κώδικα) :

- walk1: Αντιστοιχεί στην κατάσταση όπου έχουμε κυκλοφορεία στην ΕΟ και πράσινο για πεζούς στον κάθετο δρόμο.

LED0	LED1	E	B	A	D	C	F
ON	OFF	Green	Green	Red	Red	Red	Red

- ph1: Αντιστοιχεί στην κατάσταση όπου έχουμε κυκλοφορεία στην ΕΟ και Blink για πεζούς στον κάθετο δρόμο.

LED0	LED1	E	B	A	D	C	F
Blink	OFF	Green	Green	Red	Red	Red	Red

- ph2: Αντιστοιχεί στην κατάσταση όπου έχουμε κίτρινο στην ΕΟ και κόκκινο για πεζούς στον κάθετο δρόμο.

LED0	LED1	E	B	A	D	C	F
OFF	OFF	Yellow	Yellow	Red	Red	Red	Red

- ph3: Παντού κόκκινο.

LED0	LED1	E	B	A	D	C	F
OFF	OFF	Red	Red	Red	Red	Red	Red

- walk2: Αντιστοιχεί στην κατάσταση όπου έχουμε κυκλοφορεία στην κάθετη οδό και πράσινο για πεζούς στην ΕΟ.

LED0	LED1	E	B	A	D	C	F
OFF	ON	Red	Red	Green	Green	Red	Red

- ph4: Αντιστοιχεί στην κατάσταση όπου έχουμε κυκλοφορεία στην κάθετη οδό και blink για πεζούς στην ΕΟ.

LED0	LED1	E	B	A	D	C	F
OFF	Blink	Red	Red	Green	Green	Red	Red

- ph5: Αντιστοιχεί στην κατάσταση όπου έχουμε κίτρινο στην κάθετη και κόκκινο για πεζούς στην ΕΟ.

LED0	LED1	E	B	A	D	C	F
OFF	OFF	Red	Red	Yellow	Yellow	Red	Red

- ph6: Παντού κόκκινο.

LED0	LED1	E	B	A	D	C	F
OFF	OFF	Red	Red	Red	Red	Red	Red

- F1G: Πράσινο για την μερία όπου γίνεται η στροφή στην ΕΟ και όλα τα άλλα κόκκινο.

LED0	LED1	E	B	A	D	C	F
OFF	OFF	Green	Red	Red	Red	Red	Green

- F1Y: Ανάβει κίτρινο για το B ωστέ να περάσουμε μετά στην F1G.

LED0	LED1	E	B	A	D	C	F
OFF	OFF	Green	Yellow	Red	Red	Red	Red

- FY: Τα φανάρια E,F γίνονται κίτρινα και επιστρέφουμε στην ph3.

LED0	LED1	E	B	A	D	C	F
OFF	OFF	Yellow	Red	Red	Red	Red	Yellow

- C1G: Πράσινο για την μερία όπου γίνεται η στροφή στην ΕΟ και όλα τα άλλα κόκκινο.

LED0	LED1	E	B	A	D	C	F
OFF	OFF	Red	Green	Red	Red	Green	Red

- C1Y: Ανάβει κίτρινο για το B ωστέ να περάσουμε μετά στην C1G.

LED0	LED1	E	B	A	D	C	F
OFF	OFF	Yellow	Green	Red	Red	Red	Red

- CY: Τα φανάρια B,C γίνονται κίτρινα και επιστρέφουμε στην ph3.

LED0	LED1	E	B	A	D	C	F
OFF	OFF	Red	Yellow	Red	Red	Yellow	Red

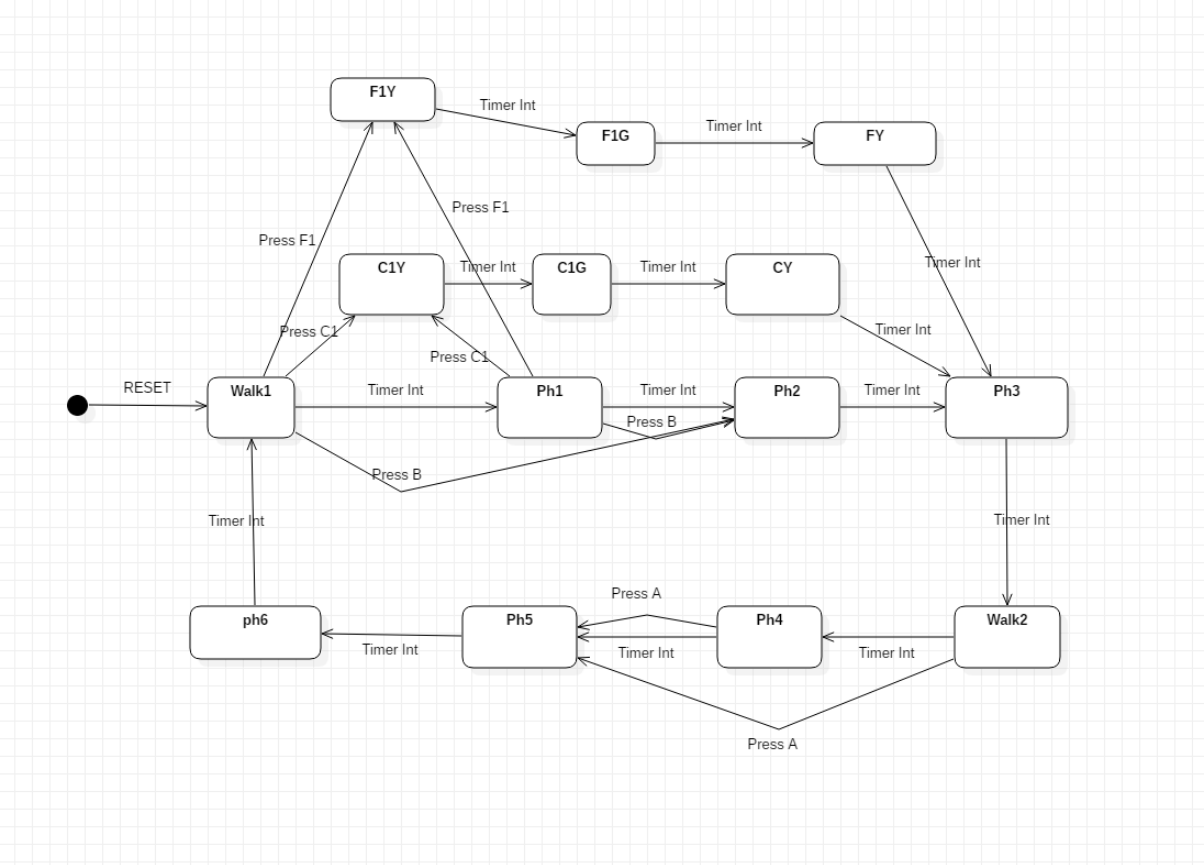
Η αλλαγή από κατάσταση σε κατάσταση γίνεται όταν τελειώσει ο χρόνος για τον οποίο το φανάρι πρέπει να είναι αναμένο(Μεταβένοντας στην επόνη σύμφωνα με τον τρόπο λειτοθργίας των φαραριών) ή όταν πατηθεί κάποιο πλήκτρο. Συγκεκριμένα αν :

- Στην ph1 , walk1 πατηθεί :
  - F1 , τότε μεταβαίνω στην F1Y - > F1G ->FY - > ph3
  - C1 , τότε μεταβαίνω στην C1Y - > C1G ->CY - > ph3
  - B τότε μεταβαίνω στην ph2.

  Και έπειτα συνεχίζει κανονικά η ροή.

- Στην ph4,walk2 πατηθεί το A τότε πάω στην ph5 και έπειτα συνεχίζει κανονικά η ροή.

Παρακάτω φαίνεται αναλυτικά το διάγραμμα καταστάσεων σε UML:



1.2 Οι χρονοι

Οι χρόνοι σύμφωνα με τις προδιαγραφές της εκφώνσης είναι :

T_Green = 51 sec

T_Red = 90 sec

Επίσης τα χρονικά διαστήματα κατά τα οποία αλλάζω κατάσταση προκύπτουν από τον πίνακα1 και είναι :

Καταστάσεις	Χρονικό διάστημα μέχρι να κάνω αλλαγή	Επεξήγηση
walk1, walk2	10sec	Από εκφώηση
ph1, ph4	41sec	T = T_Green – 10
ph2, ph5	3 sec	Από εκφώνση
ph4, ph6	36sec	T = T_Red – 10 – 3 – T_Green

2 Ο Κώδικας

2.1 RESET

Αρχικά στον κώδικα καλούμε τη διαδικασία RESET όπου πρέπει να κάνουμε τα εξής απραίτητα βήματα για την ορθή λειτουργία του προγράμματος :

- Αρχικοποίηση του Stack Pointer.
- Να θέσουμε τα κατάλληλα Pins για είσοδο ή έξοδο , όπως προκύτπουν από το συνδεσμολογία που δόθηκε.
- Να κάνουμε set το Timer1 OverFlow Interrupt για να χρονομετρήσουμε.

Το κομμάτι του RESET φαίνεται στην παρακάτω εικόνα.

```
RESET :

    ldi temp,HIGH(RAMEND);init stack pointer
    out sph,temp
    ldi temp, LOW(RAMEND)
    out spl,temp

    ;Set PORTB pins 0-6 as INPUTS

    ldi temp, 0b11000000
    out DDRB , temp
    com temp
    out PORTB , temp ; Set pull ups


    ;Ser potrd as output
    ldi temp,0xFF
    out DDRD,temp
    out PORTD,temp

    ;Set PORTA pins 0-3 as INPUTS and 4 - 7 as OUTPUTS

    ldi temp, 0b11110000
    out DDRA , temp
    ldi temp, 0xFF
    out PORTA , temp ; Set pull ups and close leds

    ;Set PORTC as  OUTPUT

    out DDRC,temp
    out PORTC,temp

    ;Set Timer1 Interupt


    ldi temp , 0b00000100 ; Activate overfow interupt
    out TIMSK,temp

    ldi temp,0x00

    out TCNT1H,temp ; Set starting value

    ldi temp,0x00
    out TCNT1L,temp

    ldi temp,0b00000101 ; Prescaller at 1024

    out TCCR1B,temp ; Set prescaler

    ldi button_pressed,0x00;Control variables
    ldi car_pressed,0x00
    sei;Enable interrupts
```

2.2 Main Loop

Στην κύρια λούπα του προγράμματος καλούμε διαδοχικά μια μια τις συναρτήσεις που υλοποιούν την κάθε κατάσταση και όταν μία επισρέψει καλείται η επόμενη σε λογική σειρά ,εκτός αν πατήθηκε κάποιο από τα πλήκτρα στο ενδιάμεσο.

```
Loop:
    rcall walk1
    rcall ph1
    rcall ph2
    rcall ph3
    rcall walk2
    rcall ph4
    rcall ph5
    rcall ph6
rjmp Loop
```

2.3 Ρουτίνες καταστάσεων και Timer

Αρχικά θα σημειώσουμε ότι επειδή η συχνότητα ρολογιού του ATMEGA16 είναι 4MHz και η μέγιστη τιμή του prescaler είναι 1024 προκύεπτει πως αν αρχίσει να μετρά από το 0 το χρονικό διάστημα που μπορούμε να μετρήσουμε με ένα Interrupt είναι 16,777 sec.

Συνεπώς σε κάποιες καταστάσεις πρέπει να περιμένουμε να γίνουν πολλαπλά Interrupt από τον Timer για να μετρήσουμε τον χρόνο που θέλουμε.

Αρχικά σε κάθε κατάσταση ελέγχουμε αν θα εκτελεστεί η ρουτίνα της ή όχι ανάλογα με το αν έχει πατηθεί πιο πριν κάποιο πλήκτρο.

Π.χ. Αν είμαι στο walk1 και πατηθεί το F1 τότε καλώ την ρουτίνα F1Y - > F1G->FY και μετά θα επιστρέψω πάλι στο walk1 οπότε πρέπει να έχω μια μεταβλητή ελέγχου που να με αναγκάσει να επιστρέψω κατευθείαν από την ρουτίνα αυτή .Η επόμενη σε σειρά ρουτίνα είναι η ph1 η οποία δεν πρέπει να εκτελεστεί ,διοτί αφού πατήθηκε το F1 και πέρασαν οι αντίστοιχες φάσεις και έτσι πρέπει να επιστρέψει αμέσως και να εκτελεστεί η ph3.Το ίδιο ισχύει και για τη ph2 .

Αυτό επιτυγχάνεται με τη μεβλητή ελέγχου car_pressed.

Τα αντίστοιχα ισχύουν και για το πάτημα κουμπιού από πεζό.Η μεταβλητή ελέγχου για τα κουμπια είναι η button_pressed. Επίσης φροντίζουμε στην ph2 , ph5 να μηδενίζουμε τον timer για να μετρήσει από την αρχή εφόσον αν πατηθεί το πλήτρκο απλά επιστρέφω από προηγούμενη ρουτίνα χωρίς να μεσολαβήσει διακοπή.

Έπειτα αν η ρουτίνα πρέπει να εκτελεστεί αποθηκεύουμε σε έναν μετρητή,τον timL, το πόσες φορές πρέπει να κάνουμε το interrupt για να μετρήσουμε το χρόνο που θέλουμε και σε έναν καταχωρητή ελέγχου να πούμε ποιον χρόνο μετράμε :

Ο καταχωρητής αυτός στο πρόγραμμα ονομάστηκε control.

Στη συνέχεια καλώ τις ρουτίνες για να ανάψουν τα κατάλληλα φανάρια και τέλος περιμένω σε μια λούπα μέχρι να μηδενιστή ο μετρητής των Interrupt και επιστρέφω. Παράλληλα στη λούπα ελέγχω για τυχόν πλήκτρα που μπορεί να πατήθκαν ή μπορεί να κλαώ ρουτίνες μικρής καθυστέρησης για να αναβοσβήσει κάποιο LED.

Ένα παράδειγμα του κώδικα φαίνεται παρακάτω για τη ρουτίνα της φάσης walk1 και της ph1 .

```
walk1:

    ldi control,Walk ; Let the timer now that we are counting time for the Walk phase
    ldi timL,W_loops; Load in the counter number of interrupts needed (1 in this case)

    ;Routines for the lights
    rcall light_E_G
    rcall light_B_G
    rcall light_A_R
    rcall light_D_R
    rcall light_F_R
    rcall light_C_R
    rcall led0_on

    lw1:
        ;If B is pressed button_pressed = 1 else continue
        sbis PINA,1;If bit PINA(1) =1 then skip next command
        ldi button_pressed,0x01
        ;Check if C1/F1 is pressed
        ;If they are pressed the routines for these phase will execute inside checkC1 or checkF1 and
        ;When they execute they will make car_pressed = 1
        ;Then return
        rcall checkF1
        rcall checkC1
        clz;clear zero flag
        ;If C1/F1 was pressed return immidietly
        cpi car_pressed,1
        breq retw1
        clz
        ;If B was pressed  return immidietly
        cpi button_pressed,1
        breq retw1
        clz ; clear zero flag
        cpi timL,0
        breq retw1 ;Check If i did all the timer loops for green time  if i did then return else jumo ti lw1
        rjmp lw1
retw1:
ret

```

```
ph1:
    ;If B or C1 or F1 was pressed if return immidietly else execute routine normally
    clz
    cpi car_pressed,1
    breq ret1
    cpi button_pressed,1
    breq ret1
    ldi control,Green ; tell the control register that we wait for the green time to finish
    ldi timL,G_loops ; Initialize loop counter

    rcall light_E_G
    rcall light_B_G
    rcall light_A_R
    rcall light_D_R
    rcall light_F_R
    rcall light_C_R

    l1:
        rcall led0_on
        rcall delay;Call to blink the leds
        rcall led0_off
        ;Check if B is pressed
        sbis PINA,1;If bit PINA(1) =1 then skip next command
        ldi button_pressed,0x01
        rcall checkF1
        rcall checkC1
        clz
        cpi car_pressed,1
        breq ret1
        clz
        cpi button_pressed,1
        breq ret1
        rcall delay
        clz ; clear zero flag
        cpi timL,0
        breq ret1 ; If i did all the timer loops the green time passed and i return
        rjmp l1
ret1:
ret
```

*Η delay προκαλεί καθυστέρηση 0.25 δευτερόλεπτα για να αναβοσβήσει το led . Σε αυτόν τον χρόνο δεν κάνουμε έλεγχο για τα κουμπιά , όμως θεωρούμε ότι το κάθε κουμπί εφόσον πατιέται από άνθρωπο θα είναι πτημένο για αρκετό χρονικό διάστημα για να ανιχνευτεί από τον μικροεπεξεργαστή , όπως και έγινε με επιτυχεία στο εργαστήριο.

Στην ρουτίνα εξυπηρέτησης διακοπής του timer1 αρχικά ελέγχουμε την τιμή του control,που μας δίνει την πληροφορία για το ποιον χρόνο θέλουμε να μετρήσουμε, και μεταβαίνουμε στην κατάλληλη ετικέτα , όπου ελέγχουμε την τιμή του μετρητή timL, αν η τιμή του είναι 1 βάζουμε στον timer1 την τιμή που πρέπει για να μετρήσουμε τον χρόνου θέλουμε αν είναι 0 ή μεγαλύτερη του 1 μηδενίζουμε τον timer, μειώνουμε κατά 1 την τιμή του μετρητή timL και επιστρέφουμε.

Η ρουτίνα εξυπηρέτησης διακοπής φαίνεται παρακάτω .

```
timer1 :
        sbrc control,0
        rjmp G_int
        sbrc control,1
        rjmp Y_int
        sbrc control,2
        rjmp R_int
        sbrc control,3
        rjmp W_int

        G_int:
            clz
            cpi timL,1
            breq lastG

            ldi temp,0x00

            out TCNT1H,temp ; Set starting value

            ldi temp,0x00
            out TCNT1L,temp
            dec timL
        reti
        R_int :
            clz
            cpi timL,1
            breq lastR

            ldi temp,0x00

            out TCNT1H,temp ; Set starting value

            ldi temp,0x00
            out TCNT1L,temp
            dec timL
        reti
        Y_int :
            clz
            cpi timL,1
            breq lastY

            ldi temp,0x00

            out TCNT1H,temp ; Set starting value

            ldi temp,0x00
            out TCNT1L,temp
            dec timL
        reti
```

```
144         W_int:
145             clz
146             cpi timL,1
147             breq lastW
148 
149             ldi temp,0x00
150 
151             out TCNT1H,temp ; Set starting value
152 
153             ldi temp,0x00
154             out TCNT1L,temp
155             dec timL
156         reti
157 
158 
159 
160 
161         lastG   :
162 
163             ldi temp,high(Gtim0)
164 
165             out TCNT1H,temp ; Set starting value
166 
167             ldi temp,low(Gtim0)
168 
169             out TCNT1L,temp
170 
171             dec timL
172         reti
173 
174 
175         lastR   :
176 
177             ldi temp,high(Rtim0)
178 
179             out TCNT1H,temp ; Set starting value
180 
181             ldi temp,low(Rtim0)
182 
183             out TCNT1L,temp
184 
185             dec timL
186         reti
```

```
187         lastY   :
188
189             ldi temp,high(Ytim0)
190
191             out TCNT1H,temp ; Set starting value
192
193             ldi temp,low(Ytim0)
194
195             out TCNT1L,temp
196
197             dec timL
198         reti
199         lastW :
200             ldi temp,high(Wtim0)
201
202             out TCNT1H,temp ; Set starting value
203
204             ldi temp,low(Wtim0)
205
206             out TCNT1L,temp
207
208             dec timL
209         reti
210 reti
```

2.4 Defs - Equs

Εδώ θα παραθέσω τις συμβάσεις για ονόματα καταχωρητών και σταθερών.

```
.include "8515def.inc"


.def Temp = R16
.def control = r17 ; Control register
.def timL = r18; Loops the timer has to do
.def button_pressed = r19;
.def car_pressed = r20;

.equ Yellow = 2
.equ Green = 1
.equ Red = 4
.equ Walk =8

.equ G_loops = 3;
.equ R_loops = 3;
.equ Y_loops = 1;
.equ W_loops = 1;

.equ Rtim0 = 55981;
.equ Gtim0 = 36452
.equ Ytim0 = 53817
.equ Wtim0 = 26474

.org 0x0000
rjmp RESET
.org 0x0010
rjmp timer1
reti
```

- Temp : Καταχωρητής γενικής χρήσεως
- control : Καταχωρητής που περιέχει την πληροφορία για τον ποιο χρόνο μετράμε. Αρχικοποιείται σε κάθε ρουτίνα υλοποίησης μιας φάσης και ελέγχεται στην ρουτίνα διακοπής.
- timL : Ο μετρητής που μας λέει πόσες φορές πρέπει να εκτελεστεί διακοπή για να μετρήσουμε το διάστημα που θέλουμε.
- button_pressed : Καταχωρητής που ελέγχει αν πατήθηκε κάποιο πλήκτρο από πεζό ώστε να επιστρέψω κατευθείαν από μια ρουτίνα κατάστασης . Στις ροθτίνες ph2 , ph5 μηδενίζεται .
- car_pressed : Καταχωρητής που ελέγχει αν πατήθηκε κάποιο πλήκτρο από αμάξι ώστε να επιστρέψω κατευθείαν από μια ρουτίνα κατάστασης . Στις ροθτίνες ph3 , ph6 μηδενίζεται .
- G_Loops: Σταθερά που μου λέει πόσες διακοπές χρειάζομαι για να μετρήσω τον χρόνο του πράσινου.
- Green: Σταθερά που φορτώνω στον control για να ξέρω ποιον χρόνο μετράω σε μια ρουτίνα κατάστασης.
- Gtim0: Η αρχική τιμή του timer κατά την τελευταία διακοπή για να μετρήσει τα κατάλληλα δευτερόλεπτα .
  *Τα ίδια ισχύουν και για τα άλλα χρώματα.

Για να βρω τις τιμές για τον G_loops , Gtim0 ακολούθησα την εξής διαδικασία (΄μοια και για τα άλλα χρώματα) :
Ξέρω πως σε ένα interrupt μπορώ να μετρήσω ως 16,77 sec.
T_g = 41 sec = 2 * 16,77 + 7,446 sec

Άρα θέλω 3 interrupts για το πράσινο.
Στο τελευταίο interrupt θέλω να μετρήσω 7,446 δευτερόλεπτα , τα οποία αντιχοιχυούν σε x χτύπους ρολογιού και ισχύει :

Συχνότητα ρολογιού : f = 4Mhz/1024

Πρέπει x/f = 7,446 => x = 29086

Ο timer1 είναι 16-bit άρα μετρά μέχρι την τιμή 65535 άρα

Gtim0 = 65536 – x = 36452

.