ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΕΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

Αναφορα Εργαστηριου

8ο εξαμηνο

ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΕΣ ΚΑΙ ΠΕΡΙΦΕΡΕΙΑΚΑ

Κωνσταντίνος Χατζηαντωνίου	8941
Ιωάννης Μπουντουρίδης	8872

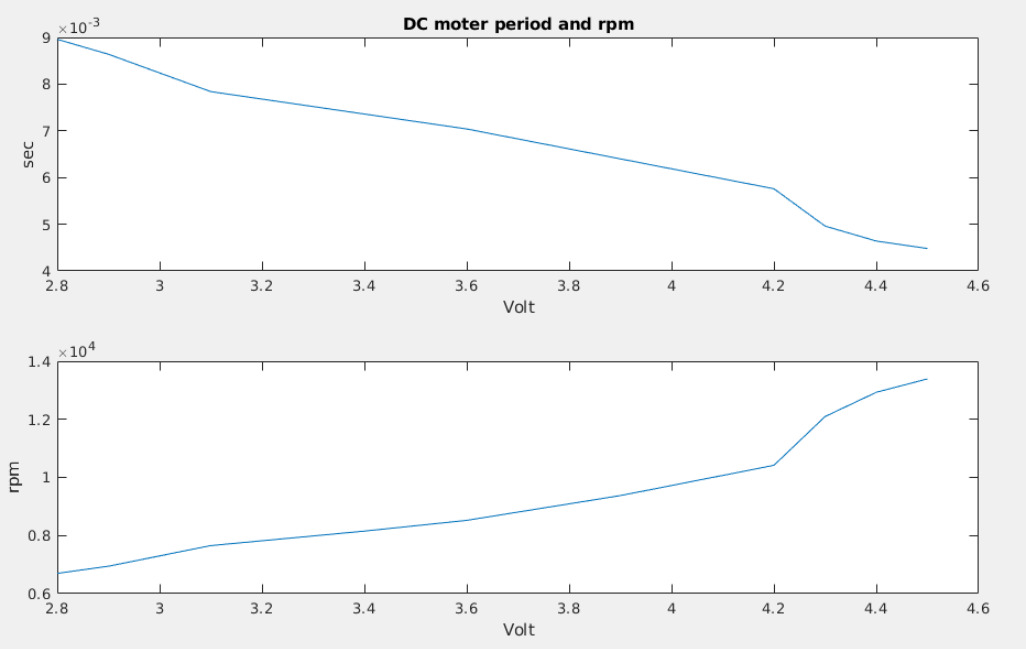
5 Ιουνίου, 2019

Τμήμα Α1-Α2

Συνδέοντας την έξοδο του Οπτικού Αποκωδικοποιητή στον παλμογράφο παρατηρούμε ένα ημιτονοειδές σήμα. Συνδέοντας την έξοδο του 7414 και ρυθμίζοντας κατάλληλα τον ροοστάτη R4 παρατηρούμε τετραγωνικούς παλμούς.

Μετρήσεις με τον παλμογράφο

Με τον παλμογράφο αρχικά μετράμε την περίοδο του σήματος από τον οπτικό κωδικοποιητή. Στη συνέχεια βρίσκουμε την συχνότητα του κινητήρα με την σχέση $fm = \frac{1}{32T_{OK}}$, και μετά υπολογίζουμε τα rpm: $rpm = f \cdot 60$.



Τμήμα Α3

Στο τμήμα αυτό, έχουμε να μετρήσουμε την συχνότητα του κινητήρα με τον AVR με 2 τρόπους:

- Μέτρηση παλμών σε χρονικό παράθυρο 1sec
- Χρονική διάρκεια 4 παλμών

Παράλληλα έχουμε να μετρήσουμε και την τάση που δίνουμε στον κινητήρα με τον ADC του AVR.

Μέτρηση παλμών σε χρονικό παράθυρο

Περιληπτικά

Σε αυτήν τη μέθοδο, θα χρησιμοποιήσουμε τον Timer1 σαν counter και τον Timer0 σαν χρονιστή. Θα χρησιμοποιήσουμε prescaler 1024, ώστε το overflow να συμβαίνει σε 0.065 sec. Σε συνδυασμό με έναν counter που θα μετράει μέχρι το 16, πετυχαίνουμε παράθυρο 1 sec.Με το που ενεργοποούμε τα interrupt για τον Timer0 ενεργοποιούμε και την λειτουργία counter στον Timer1. Μόλις έρθει το interrupt από τον timer0, τα σταματάμε και αποθηκεύουμε τις τιμές από TCNT1H, TCNT1L, που περιέχουν πόσοι παλμοί έγιναν στο χρονικό διάστημα 0.065sec. Οι 2 αυτές τιμές αποθηκεύτηκαν στην Data Memory, για να εμφανιστούν στα LEDs μετά το τέλος των μετρήσεων.

Παρουσίαση και εξήγηση κώδικα

Πρώτα, παρουσιάζονται τα σεταρίσματα των timers και μεταβλητών.

```
; ---------- INSTALL TIMER 1 AS COUNTER (PB1) ----------
    ldi temp, 0b00000111        ; set external pulse (T1) == (PB1)
    out TCCR1B, temp            ; apply settings
    clr temp
    out TCCR1A, temp
    out TCNT1H, temp            ; initial value
    out TCNT1L, temp

    clr secCom ; count 16 times => 1/ 16 sec

    ldi flag, 0x00              ; singals the main the time window is over
```

```
; ---------- INSTALL TIMER0 AS CLOCK ----------
    ;** CPU 4MHZ
    ;** PRESCALER 1024
    ;** we will count 1/32 sec for 32 times

    ldi temp, 0b00000101        ; setting the prescaler 1024
    out TCCR0, temp
    ldi temp, 0b00001100        ; initial value of clock at 12
    out TCNT0, temp
```

```

; ---------- ENABLE TIMER1 OVERFLOW INTERRUPT ----------
    ldi temp, 1<<TOIE0
    out TIMSK, temp

    ; ---------- ENABLE GLOBAL INTERUPTS ----------
    sei
```

Εδώ, παρουσιάζεται η ρουτίνα εξυπηρέτησης του timer0 overflow interrupt. Αρχικά αυξάνουμε τον counter και ελέγχουμε αν πέρασε 1 δευτερόλεπτο. Αν ο counter είναι 16 (δηλαδή πέρασε 1sec), διαβάζουμε τις τιμές του timer1 και τις αποθηκεύουμε. Έπειτα διαβάζουμε την τάση που δώσαμε στον κινητήρα και την αποθηκεύουμε. Τέλος ξανασετάρουμε τα περιεχόμενα τον μετρητών για να μετρήσουμε από την αρχή.

```
INTERRUPT:
    push temp
    in SREG, temp
    push temp
    inc secCom        ; 1/16 sec  -- counter
    cpi secCom,16     ; check if 1 sec completed
    brne WAIT_ONE_SEC
; ------------------- DISABLE TIMER0 -------------------
    clr temp
    out TCCR0, temp
    ldi flag, 0xFF    ; change flag to signal main
    in holderLow, TCNT1L
    in holderHigh, TCNT1H    ; Read the counter values

; ------------------- SAVE RESULT -------------------
    st X+, holderLow
    st X+, holderHigh

; ------------------- START CONVERSION ADC -------------------

    ldi temp,0b00100000 ; bits 7:6 used for ref voltage, bit 5 used to left
    justify, bit 4:0 enable adc0
    out ADMUX, tmp ; enable ADC0
    ldi tmp, 0b11000000
    out ADCSRA,tmp ; start conversion

; ------------------- CONVERSION IN PROGRESS -------------------
wait_convertsion:
    in tmp, ADCSRA
    andi tmp,0b00010000
    cpi tmp,0b00010000
    breq conversion_finished
    rjmp wait_convertsion
conversion_finished:

; ------- STORE RESULT OF ADC -------------------
    in tmp,ADCH
    st X+,tmp
; ------------------- CLEAR TIMER1  and TIMER0 ---------
    clr temp
    out TCNT1H, temp
    out TCNT1L, temp
    out TCNT0, temp

```

```
44
45 WAIT_ONE_SEC:
46     pop temp                    ; reload sreg and temp
47     out sreg, temp
48     pop temp
49     reti
```

Παρακάτω παρουσιάζεται ο βρόγχος αναμονής. Όταν περάσει το 1 δευτερόλεπτο και ολοκληρωθούν οι μετρήσεις, η ρουτίνα εξυπηρέτησης του interrupt, κάνει το flag 0xFF και βγαίνει από τον βρόγχο MAIN. Στον βρόγχο NEXT_FREQ πατώντας ενά από τα κουμπια sw0, sw1 μπορούμε είτε να πάμε σε νέα μέτρηση με διαφορετική τάση είτε στο τέλος όπου θα εμφανιστούν οι τιμές στα LED.

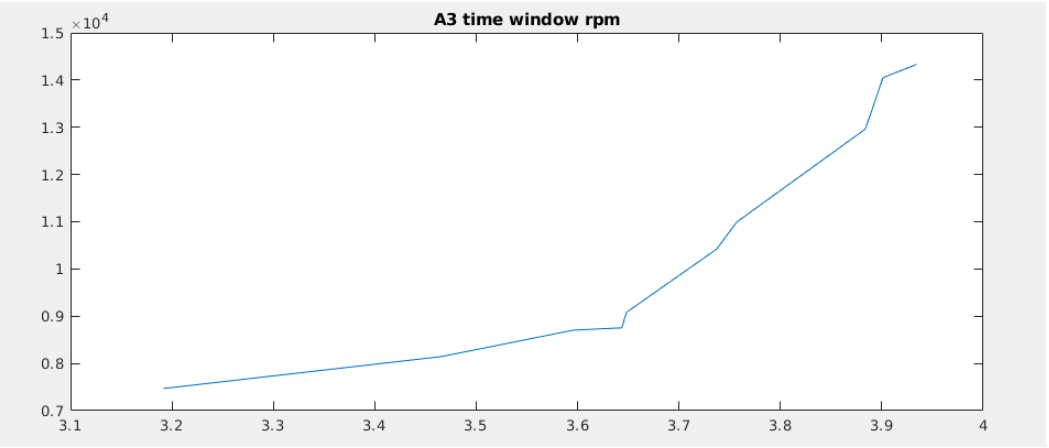
```
1  MEASURE_AGAIN:
2      clr secCom ; count 16 times => 1/ 16 sec
3      ldi flag, 0x00
4
5  MAIN:
6  ; ------------------- WAIT INTERRUPT (EVERY 1/16 SEC) -------------------
7      cpi flag, 0xFF
8      breq NEXT_FREQ
9      rjmp MAIN
10 NEXT_FREQ:
11     sbis PINB, 2
12     rjmp RELEASE_NEXT_FREQ    ; press btn 2 to measure the next frequency
13     sbis PINB, 3
14     rjmp LOOP_RESULT          ; press btn 3 to go to end and show the values
15     rjmp NEXT_FREQ
16
17 RELEASE_NEXT_FREQ:
18     sbis PINB, 2              ; wait btn 2 to release
19     rjmp RELEASE_NEXT_FREQ
20     rjmp MEASURE_AGAIN
```

Τέλος, είναι ο κώδικας που παρουσιάζει της τιμές στα LED, Τα LED είναι συνδεδεμένα στο PORTD. Οι τιμές εμφανίζονται με την αντίθετη σειρά που αποθηκεύτηκαν (Voltage, CountHigh, CountLow).

```
1  LOOP_RESULT:
2      cpi XL,0 ; check if all results appeared
3      breq end_game ; end of program
4    ld ADCHIGH,-X ;pre decrement and load ADCH
5    com ADCHIGH ; inverse
6      out PORTD,ADCHIGH ; show ADCH
7      rcall sw0_click ; wait to click sw3 for high result
8      ld resultH,-X ; pre decrement and load high result
9      com resultH ; inverse
10     out PORTD, resultH ; show high byte result
11     rcall sw0_click ; wait to click sw3 for low result
12     ld resultL,-X ; pre decrement and load low result
13     com resultL ; inverse logic
```

```
14      out PORTD, resultL  ; show low byte result
15      rcall sw0_click ; wait to click sw3 for high result
16      rjmp LOOP_RESULT
17
18
19  end_game:
20      rjmp end_game
21
22  ; ---------- FUNCTION IMPLEMENTS SW0 CLICK ----------
23  sw0_click:
24    sbic PINB,3
25      rjmp sw0_click
26  sw0_release:
27      sbis PINB,3
28      rjmp sw0_release
29    ret
```

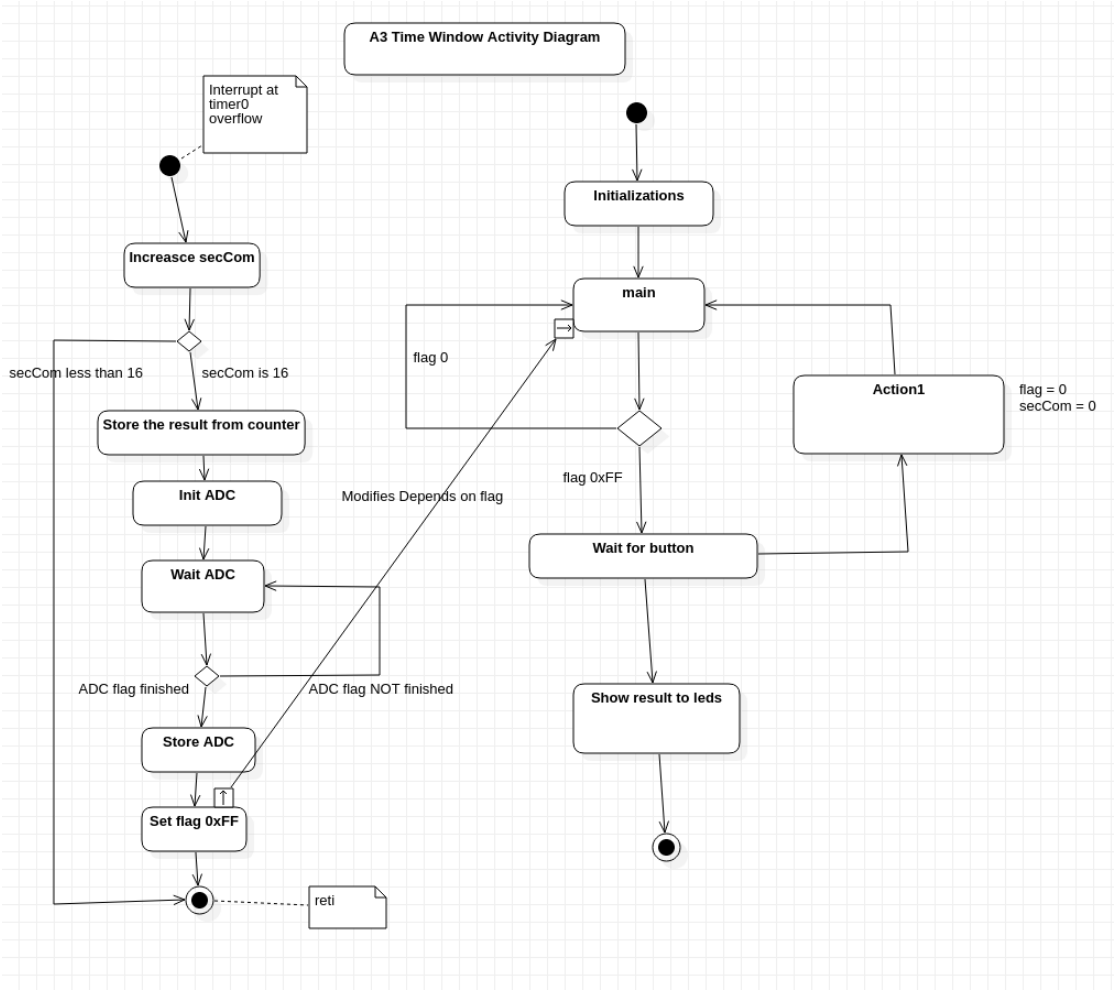
Μετρήσεις



Σχόλια

Επιλέξαμε να χρησιμοποιήσουμε τον timer1 σαν μετρητή, γιατί σε 1 δευτερόλεπτο θα γίνουν παραπάνω από 255 παλμοί. Παρ'όλο που ο timer0 δεν αρκεί από μόνος του για μέτρηση 1 sec,η χρήση του μετρητή έλυσε εύκολα το πρόβλημα

Διάγραμμα Ροής



Μέτρηση διάρκειας 4 παλμών

Περιληπτικά

Σε αυτή τη μέθοδο, θα χρησιμοποιήσουμε τον timer1 σαν χρονιστή σε συνδιασμό με την λειτουργία Input Capture. Στο πρώτο και τέταρτο input capture θα αποθηκεύουμε τα timestamps, θα τα αφαιρούμε και θα τα διαιρούμε δια 4, για να υπολογίσουμε την περίοδο του OK.

Παρουσίαση και εξήγηση κώδικα

```
1  ; ———————— CLEAR TIMER1 ————————
2    clr  tmp
3    out  TCNT1L, tmp
```

```
    out TCNT1H,tmp

    ; ---------------------- INSTALL TIMER1 (PD6) ----------------------
    clr tmp
    out TCCR1A,tmp
    ldi tmp,1<<ICF1 ; clear input capture flag
    out TIFR,tmp ; clear input capture flag
    ldi tmp,0b01000001 ; capture positive pulse
    out TCCR1B,tmp ; prescaler CK / 1

    ; ---------------------- CLEAR INPUT CAPTURE COUNTER ----------------------
    clr tmp
    out ICR1H, tmp
    out ICR1L, tmp

    ; ---------------------- ENABLE INPUT CAPTURE INTERUPTS ----------------------
    ldi tmp,1<<TICIE1
    out TIMSK,tmp
    ldi flag,0x05 ; set flag
    sei ; enable interupts
```

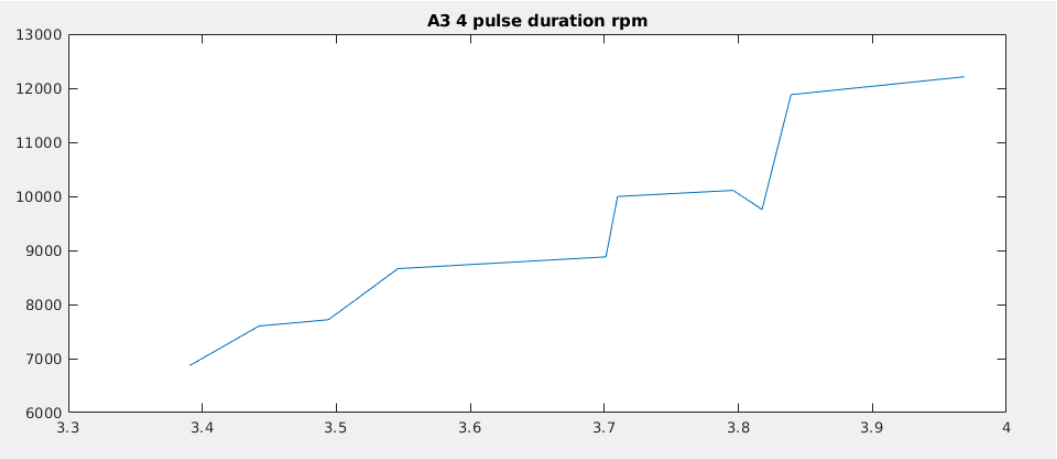
Αυτή είναι η ρουτίνα εξυπηρέτησης Input Capture interrupt. Αν βρισκόμαστε στην πρώτη μέτρηση (flag = 5) ή στην 4η (flag = 1) αποθηκεύουμε τα αποτελέσματα σε καταχωρητές. Τέλος μειώνεται κατά ένα το flag.

```
INTERUPT:
    ; ---------------------- INTERUPT APPEARED ----------------------
    push temp
    in sreg, temp
    cli ; disable interupts
    cpi flag,5 ; if 1st interupt comes
    breq SAVE_FIRST_CAPTURE ; save result of input capture to registers
    cpi flag,1 ; if 4th interupt comes
    breq SAVE_FOURTH_CAPTURE ; save result of input capture to registers
    rjmp FINISH ; other interupts will not be saved
SAVE_FIRST_CAPTURE:
    in capture1L,ICR1L ; saving 1st capture
    in capture1H, ICR1H ; saving 1st capture
    rjmp FINISH
SAVE_FOURTH_CAPTURE:
    in capture2L,ICR1L ; saving 4th capture
    in capture2H,ICR1H ; saving 4th capture
FINISH:
    dec flag ; decrease capture flag
    sei ; enable interupts
    out sreg, temp
    pop temp
    reti ; return interupt
```

Όπως και στην προηγούμενη μέθοδο, έχουμε έναν βρόγχο αναμονής. Όταν το flag γίνει 0, μετά από τις μειώσιες στην ρουτίνα εξυπηρλετησεις του interrupt, πηγαίνουμε στην ρουτίνα CALCULATE_RESULT. Δεν παρουσιάζονται(όπως και η ένδειξη σε LED) γιατί είναι ίδιες με το προηγούμενο.

```
1  WAIT:
2      ; ———————————— WAIT INTERUPTS ————————————
3      cpi flag,0x00 ; if 5 interupts arrived
4      breq CALCULATE_RESULT ; calculate period
5    rjmp WAIT ; else wait for interupts
```

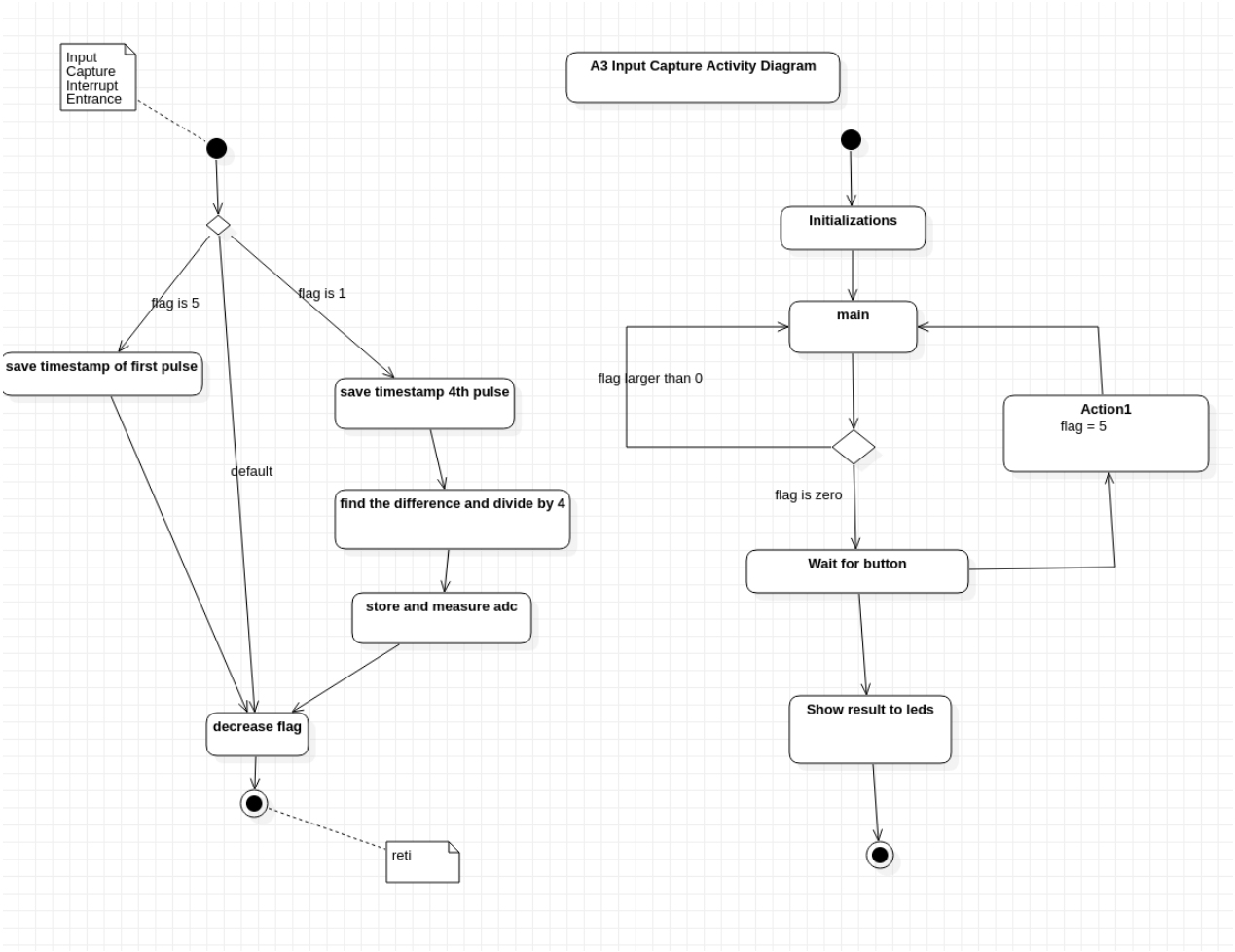
Μετρήσεις



Σχόλια

Επιλέξαμε να χρησιμοποιήσουμε τον timer1 σαν Input Capture και χρονιστή, για να έχουμε μεγαλύτερη ακρίβεια.

Διάγραμμα ροής



Τμήμα Β1

Σε αυτό το τμήμα έχουμε να ελέγξουμε τον κινητήρα με χρήση PWM. Με ένα κουμπί το duty cycle θα αυξάνεται, όπως και τα rpm του κινητήρα, ενώ με ένα άλλο θα μειώνεται. Κάθε φορα που πατάμε το κουμπί, πριν μεταβάλλουμε το duty cycle, θα μετράμε τη συχνότητα του κινητήρα όπως στο τμήμα Α3 με τη μέθοδο Input Capture.

Παρουσίαση και εξήγησ κώδικα

Πρώτα σετάρουμε το PWM. Θα χρησιμοποιήσουμε τον timer0 για αυτή τη δουλειά.

```
1  ;————————— init PWM —————————
```

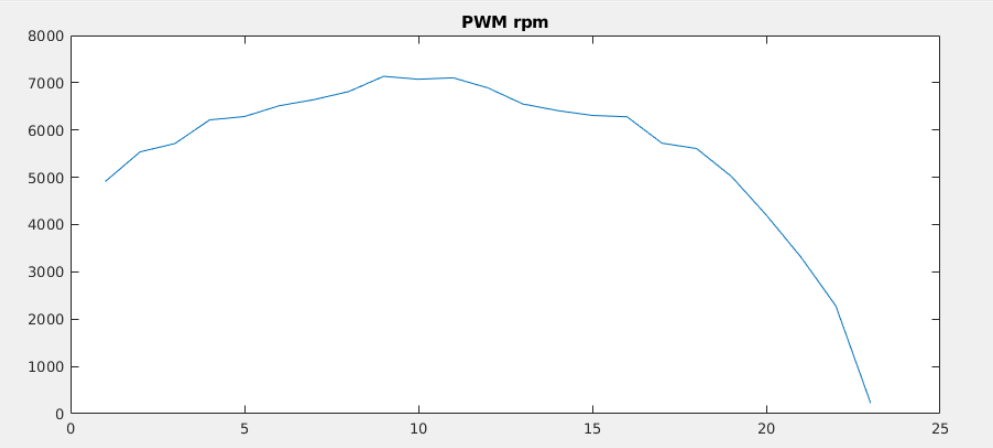
```
    ldi temp, 0b00001000 ;set pb3 as output
    out DDRB, temp
    ldi temp, 0b01100011 ; wgm00, com01, cs0-1/0
    out TCCR0, temp

    ; load 20% pwm as init val
    ldi curr_pwm, 55
    out OCR0, curr_pwm

    ;---------- Set buttons to pd0,1 ---------
    ldi temp, 0b00000000 ; the whole ddrd is input
    out DDRD, temp
```

Στη συνέχεια βρισκόμαστε σε έναν βρόγχο επανάληψης, μέχρι να πατήσουμε κάποιο από τα κουμπιά. Όταν πατήσουμε το κουμπί για αύξηση, αυξάνουμε κατα 12 (5% του 255) τον OCR0. Ομοίως αν πατήσουμε το κουμπί μείωσης θα αφαιρέσουμε 12. Πριν γίνει οποιαδήποτε αλλαγή στο PWM, βλέπουμε ότι καλείται η συνάρτηση 'measuring_loop'. Αυτή πραγματοποιεί την ίδια διαδικάσια με το τμήμα A3 β, δηλαδή μετράει την χρονική απόσταση 4 παλμων, διαρεί δια 4 και την αποθηκεύει. Η εμφάνιση στα led γίνεται όπως και στο A3.

```
speed_handler:
    ; skip if bit is set
    sbis PIND, 0 ; sw0 speed up
    rjmp inc_speed
    sbis PIND, 1 ; sw1 slow down
    rjmp dec_speed
    sbis PIND, 2 ; show results
  rjmp SHOW_RESULT
    rjmp speed_handler

inc_speed:
    ; wait user to release
    sbis PIND, 0
    rjmp inc_speed
    rcall measuring_loop
    ldi temp, pwm_step
    add curr_pwm, temp
    out OCR0,curr_pwm
    rjmp speed_handler
    ;

dec_speed:
    sbis PIND, 1
    rjmp dec_speed
    rcall measuring_loop
    cpi curr_pwm, pwm_step
    brlo zeroing_speed
    subi curr_pwm, pwm_step
    out OCR0,curr_pwm
    rjmp speed_handler

```

```
32 zeroing_speed:
33     ldi curr_pwm, 0x00
34     out OCR0, curr_pwm
35     rjmp speed_handler
```

Μετρήσεις



Σχόλια

Επιλέξαμε τον timer0 ως PWM γιατί ο timer1 χρησιμοποιείται για την μέτρηση συχνότητας. Χάνουμε όμως ακρίβεια στο duty cycle, αφού 12/255 = 4.7% ενώ 13/255 = 5.1%. Σε συγκεκριμένες εφαρμογές, τέτοια απόκλιση μπορεί να μην είναι ανεκτή και να αναγκαστούμε να χρησιμοποιήσουμε τον 10bit pwm του timer1.

Τμήμα B2

Για το τμήμα B2, δεν προλάβαμε να το τελειώσουμε, παρουσιάσουμε και να πάρουμε μετρήσεις (ήταν χαλασμένο το AVR και αργήσαμε να ξεκινήσουμε στο τελευταίο εργαστήριο). Παρ'όλα αυτά έχουμε σχεδιάσει την ρουτινά εξυπηρέτησης Overflow Interrupt για να μεταβάλλεται αυτόματα το pwm.

Κάθε φόρα που συμβαίνει interrupt,αν έχει περάσει 1 sec, ανακατευθυνόμαστε στο σωστό mode. Στο πρώτο mode, αυξάνουμε κατα βήματα το PWM. Όταν γίνουν 10 βήματα αύξησης, το mode αλλάξε στο 3. Στο 3, αφού περάσουν 10 βήματα (10 sec), το mode αλλάξει στο 3, όπου γίνονται 10 βήματα μείωσης, και επανερχόμοαστε στο mode 1.

```
1  timer2INT:
2     inc t2Counter
3     cpi t2Counter, 16
4     brlo RET
5     ldi t2Counter, 0
```

```
6       cpi t2Mode, 1
7       breq MODE_1             ;increase
8       cpi t2Mode, 2
9       breq MODE_2             ;decrease
10      cpi t2Mode, 3
11      breq MODE_3             ;stay
12
13      rjmp timer2INT
14
15  MODE_1:
16      inc t2StepCounter
17      cpi t2StepCounter, INCREASE_STEP        ; if stepCounter == INC_STEP =>
        increase PWM
18      breq MODE_1_CONT
19      reti
20
21  MODE_1_CONT:
22  ; wait user to release
23      ldi t2StepCounter, 0
24      rcall measuring_loop
25      ldi temp, pwm_step
26      add curr_pwm, temp
27      out OCR0, curr_pwm
28      ;rjmp MODE_CHECK
29      ldi t2Mode, 3
30      reti
31
32  MODE_2:
33      inc t2StepCounter
34      cpi t2StepCounter, INCREASE_STEP        ; if stepCounter == INC_STEP =>
        increase PWM
35      breq MODE_2_CONT
36      reti
37
38  MODE_2_CONT:
39      ldi t2StepCounter, 0
40      rcall measuring_loop
41      cpi curr_pwm, pwm_step
42      brlo zeroing_speed_2
43      subi curr_pwm, pwm_step
44      out OCR0, curr_pwm
45      ;rjmp MODE_CHECK
46      ldi t2Mode, 1
47      reti
48  zeroing_speed_2:
49      ldi curr_pwm, 0x00
50      out OCR0, curr_pwm
51      ;rjmp MODE_CHECK
52      reti
53
54  MODE_3:
55      inc t2StepCounter
56      cpi t2StepCounter, TEN_SEC_CHECK
```

```
57      brge MODE_3_CONT
58      reti
59  MODE_3_CONT:
60      ldi t2StepCounter, 0
61      ldi t2Mode, 2
62      reti
63  RET:
64      reti
```

Προβλήματα

- Μερικές φορές, οι μετρήσεις που παίρναμε δεν ήταν λογικές. Αλλά αν παίρναμε ξανά μέτρηση, χωρίς να πειράξουμε την τάση, 2-3 φορές, έβγαινε σωστή.
- Ήταν δύσκολο το debugging λόγω των interrupt και I/O παλμών.

Οργάνωση αρχείων

Οι κώδικες έχουν χωριστεί σε 4 διαφορετικά αρχεία.

- a3a.asm .Περιέχει την υλοποίηση για τον υπολογισμό συχνότητας με χρονικό παράθυρο 1sec.
- a3b.asm Περιέχει την υλοποίηση για τον υπολογισμό συχνότητας με Input Capture και τη διαφορά χρόνου 4 παλμών.
- b1.asm Περιέχει την υλοπίηση για τον έλεγχο κινητήρα με pwm, πατώντας κουμπιά.
- b2.asn Περιέχει την μη επαληθευμένη υλοποίηση για τον αυτόματο έλεγχο κινητληρα με pwm.