**Εργαστήριο Προηγμένοι Μικροεπεξεργαστές**

Εργαστηριακή Άσκηση 3

Σκάγκος Ιωάννης 1072611

Στεργίου Γεώργιος 1072503

**Ερώτημα 1**

Διάγραμμα Ροής:

Εικόνα που περιέχει κείμενο, διάγραμμα, σκίτσο/σχέδιο, Σχέδιο

Περιγραφή που δημιουργήθηκε αυτόματα

Κώδικας ερωτήματος 1:

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

void TCA0\_init(void);

void TCA0\_init(void)

{

TCA0.SPLIT.CTRLA = TCA\_SINGLE\_CLKSEL\_DIV16\_gc ;

TCA0.SPLIT.LPER = 0x20; //select the resolution

TCA0.SPLIT.HPER = 0x40;

TCA0.SPLIT.HCNT = 2;

TCA0.SPLIT.HCMP0 = 0x20; //select the duty cycle 50%

TCA0.SPLIT.LCMP0 = 0x14; //select the duty cycle 40%

//TCA0.SPLIT.HCMP1 = 0x08;

//TCA0.SPLIT.LCMP1 = 0x00;

//select Single\_Slope\_PWM

TCA0.SPLIT.CTRLB |= TCA\_SINGLE\_WGMODE\_SINGLESLOPE\_gc;

TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_HUNF\_bm | TCA\_SPLIT\_LUNF\_bm;

sei();

}

int main(void)

{

PORTD.DIR |= (PIN1\_bm|PIN2\_bm|PIN0\_bm); //PIN is output

//ola svhsta

PORTD.OUT |= PIN0\_bm;

PORTD.OUT |= PIN1\_bm;

PORTD.OUT |= PIN2\_bm;

//set interrupt switches

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc;

sei();

TCA0\_init();

while(1){

;

}

}

//otan patithei to koumpi mpainei edw

ISR(PORTF\_PORT\_vect){

//clear the interrupt flag

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm;

}

ISR(TCA0\_HUNF\_vect){

//clear the interrupt flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

PORTD.OUTTGL = PIN0\_bm; //PIN toggle anemistiras

}

ISR(TCA0\_LUNF\_vect){

//clear the interrupt flag

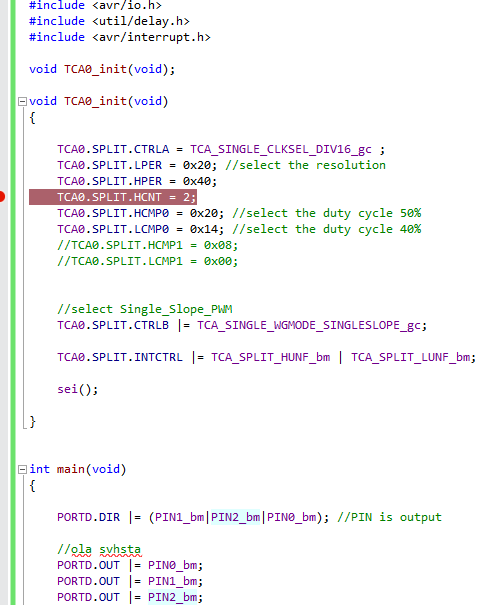
int intflags = TCA0.SINGLE.INTFLAGS;

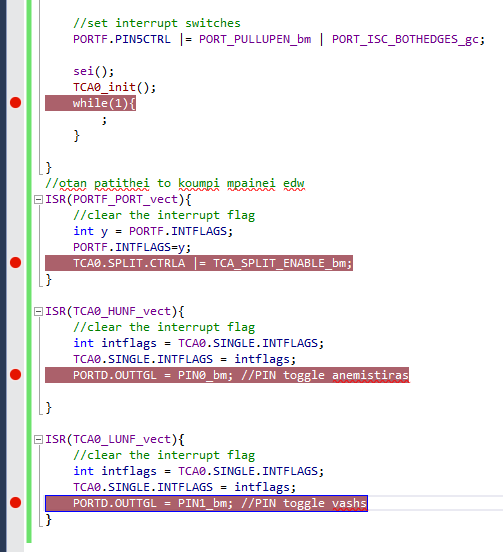
TCA0.SINGLE.INTFLAGS = intflags;

PORTD.OUTTGL = PIN1\_bm; //PIN toggle vashs

}

Screenshots με breakout points:





Σχόλια – Επεξήγηση κώδικα:

Αρχικά θέτουμε τον TIMER TCA και τον κάνουμε split ώστε να τον χρησημοποιήσουμε και για τους 2 παλμούς.

**TCA**

Χρησημοποιούμε τον καταχωρητή high για την προσομοίωση του παλμού των λεπιδών του ανεμιστήρα και τον βάζουμε να πραγματοποιεί παλμό με duty cicle 50% με τις ενολές (TCA0.SPLIT.HPER = 0x40;, TCA0.SPLIT.HCMP0 = 0x30;) και τον low για την βάση του ανεμιστήρα με duty cicle 40% αντίστοιχα.Επιπλέον καθώς υπάρχει κύνδινος ανα δύο παλμούς οι παλμοί να φτάνουν ταυτόχρονα σε ανερχόμενη παρυφή και να πρέπει να εκτελούνται ταυτόχρονα οι 2 ρουτινες διακοπής τους, θέτουμε αρχική φάση στον ένα από τους δύο καταχωρητές (με την εντολή TCA0.SPLIT.HCNT = 2;) ώστε οι ρουτίνες να μην εκτελεστούν ταυτόχρονα. Τέλος θέτουμε να καλόυνται οι ρουτίνες διακοπών HUNF και LUNF (με την εντολή TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_HUNF\_bm | TCA\_SPLIT\_LUNF\_bm;).

**MAIN**

Στην main αρχικά θέτουμε εξόδους στα PIN1,PIN0,PIN2 και τα σβήνουμε. Θέτουμε τον διακόπτη (switch) που όταν πατηθεί θα ενεργοποιήσει την δικία του ρουτίνα διακοπής στην οποία κάνουμε enable τον split timer ώστε να αρχίσει η λειτουγεία του ανεμηστήρα δηλαδή η κυκλική κίνηση των λεπιδών και της βάσης.

**ISR LUNF/HUNF**

Οι καταχωρητές HCMP0 LCMP0 αρχίζουν να μετράνε από τις τιμές που τους εχουμε θέσει 0x20 και 0x40 αντιστοιχα και όταν φτάσουν στις τιμές σύγκρισης (0x20 και 0x14) και κάνουν underflow τότε ενεργοποιούνται οι ρουτίνες διακοπής τους. Εκεί κάνουμε clear τα interrupt flags και με την εντολή PORTD.OUTTGL αναβοσβήνουμε τα PIN0 και PIN1 αντίστοιχα σύμφωνα με τους παλμούς τους.

**Ερώτημα 2**

Διάγραμμα Ροής:

Εικόνα που περιέχει κείμενο, διάγραμμα, γραμματοσειρά, Σχέδιο

Περιγραφή που δημιουργήθηκε αυτόματα

Κώδικας ερωτήματος 2:

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

void TCA0\_init(void);

void ADC0\_init(void);

void TCA0\_init(void)

{

TCA0.SPLIT.CTRLA = TCA\_SINGLE\_CLKSEL\_DIV16\_gc ;

TCA0.SPLIT.LPER = 0x20; //select the resolution

TCA0.SPLIT.HPER = 0x40;

TCA0.SPLIT.HCNT = 2;

TCA0.SPLIT.HCMP0 = 0x20; //select the duty cycle 50%

TCA0.SPLIT.LCMP0 = 0x14; //select the duty cycle 40%

//TCA0.SPLIT.HCMP1 = 0x08;

//TCA0.SPLIT.LCMP1 = 0x00;

//select Single\_Slope\_PWM

TCA0.SPLIT.CTRLB |= TCA\_SINGLE\_WGMODE\_SINGLESLOPE\_gc;

TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_HUNF\_bm | TCA\_SPLIT\_LUNF\_bm;

sei();

}

void ADC0\_init(void)

{

ADC0.CTRLA |= ADC\_RESSEL\_10BIT\_gc; //10-bit resolution

ADC0.CTRLA |= ADC\_FREERUN\_bm; //Free-Running mode enabled

ADC0.CTRLA |= ADC\_ENABLE\_bm; //Enable ADC

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc; //The bit //Enable DebugMode

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Window Comparator Mode

ADC0.WINLT |= 5; //Set threshold

ADC0.INTCTRL |= ADC\_WCMP\_bm; //Enable Interrupts for WCM

ADC0.CTRLE |= ADC\_WINCM0\_bm; //Interrupt when RESULT < WINLT

ADC0.COMMAND |= ADC\_STCONV\_bm; //Start Conversion

}

int main(void)

{

PORTD.DIR |= (PIN1\_bm|PIN2\_bm|PIN0\_bm); //PIN is output

//ola svhsta

PORTD.OUT |= PIN0\_bm;

PORTD.OUT |= PIN1\_bm;

PORTD.OUT |= PIN2\_bm;

//set interrupt switches

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc;

sei();

TCA0\_init();

while(1){

;

}

}

//otan patithei to koumpi mpainei edw

ISR(PORTF\_PORT\_vect){

//clear the interrupt flag

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm;

PORTD.OUT |= PIN2\_bm;

}

ISR(TCA0\_HUNF\_vect){

//clear the interrupt flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

PORTD.OUTTGL = PIN0\_bm; //PIN toggle anemistiras

}

ISR(TCA0\_LUNF\_vect){

//clear the interrupt flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

PORTD.OUTTGL = PIN1\_bm; //PIN toggle vashs

}

ISR(ADC0\_WCOMP\_vect){

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

TCA0.SPLIT.CTRLA &= ~TCA\_SPLIT\_ENABLE\_bm; // Disable TCA0

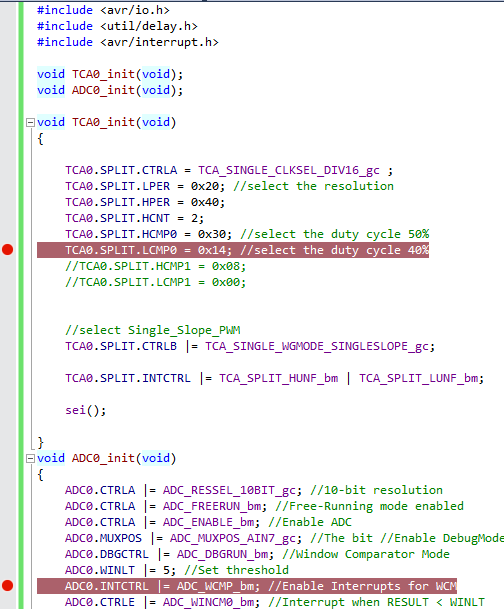
PORTD.OUTCLR= PIN2\_bm; //LED is on

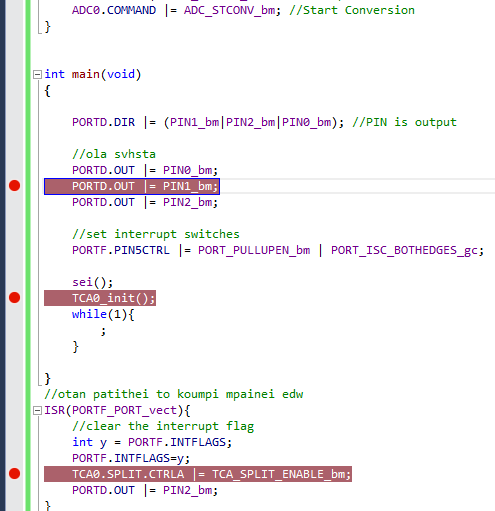
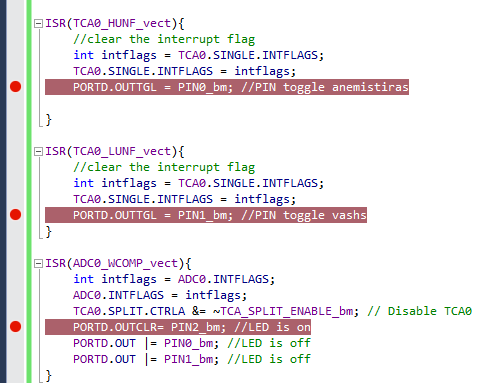
PORTD.OUT |= PIN0\_bm; //LED is off

PORTD.OUT |= PIN1\_bm; //LED is off

}

Screenshots με breakout points:



Σχόλια – Επεξήγηση κώδικα:

Στο ερώτημα 1 προσθέσαμε τα παρακάτω:

**ADC**

Προσθέτουμε έναν ADC ο οποίος αρχικοποιείται σε Free-running mode με threshold 5 και ελέγχει αν υπάρχει εμπόδιο. Αν εντοπίσει εμπόδιο είσερχεται στην ρουτίνα διακοπής του ADC όπου εκεί απενεργοποιείται ο timer TCA (δηλαδή σταματάει η κίνηση του ανεμηστήρα και της βάσης έως ότου ξαναπατηθεί το κουμπί ενεργοποίσης) ανάβει το PIN2 και σβήνουν τα PIN1 και PIN2.

**Ερώτημα 3**

Κώδικας ερωτήματος 3:

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

int per=0x40;

int duty=0x20;

int koumpi=0;

void TCA0\_init(void);

void ADC0\_init(void)

void TCA0\_init(void)

{

TCA0.SPLIT.CTRLA = TCA\_SINGLE\_CLKSEL\_DIV16\_gc ;

TCA0.SPLIT.LPER = 0x20; //select the resolution

TCA0.SPLIT.HPER = per;

TCA0.SPLIT.HCNT = 2;

TCA0.SPLIT.HCMP0 = duty; //select the duty cycle 50%

TCA0.SPLIT.LCMP0 = 0x14; //select the duty cycle 40%

//TCA0.SPLIT.HCMP1 = 0x08;

//TCA0.SPLIT.LCMP1 = 0x00;

//select Single\_Slope\_PWM

TCA0.SPLIT.CTRLB |= TCA\_SINGLE\_WGMODE\_SINGLESLOPE\_gc;

TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_HUNF\_bm | TCA\_SPLIT\_LUNF\_bm;

sei();

}

void ADC0\_init(void)

{

ADC0.CTRLA |= ADC\_RESSEL\_10BIT\_gc; //10-bit resolution

ADC0.CTRLA |= ADC\_FREERUN\_bm; //Free-Running mode enabled

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc; //The bit //Enable DebugMode

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Window Comparator Mode

ADC0.WINLT |= 5; //Set threshold

ADC0.INTCTRL |= ADC\_WCMP\_bm; //Enable Interrupts for WCM

ADC0.CTRLE |= ADC\_WINCM0\_bm; //Interrupt when RESULT < WINLT

ADC0.COMMAND |= ADC\_STCONV\_bm; //Start Conversion

}

int main(void)

{

PORTD.DIR |= (PIN1\_bm|PIN2\_bm|PIN0\_bm); //PIN is output

//ola svhsta

PORTD.OUT |= PIN0\_bm;

PORTD.OUT |= PIN1\_bm;

PORTD.OUT |= PIN2\_bm;

//set interrupt switches

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc;

sei();

TCA0\_init();

while(1){

;

}

}

//otan patithei to koumpi mpainei edw

ISR(PORTF\_PORT\_vect){

//clear the interrupt flag

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

koumpi=++;

PORTD.OUT |= PIN2\_bm;

if (koumpi%%3==1){

TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm;

ADC0.CTRLA |= ADC\_ENABLE\_bm; //Enable ADC

}

else if (koumpi%%3==2){

per=0x80;

duty=0x40;

TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm;

}

else if (koumpi%%3==0) {

TCA0.SPLIT.CTRLA &= ~TCA\_SPLIT\_ENABLE\_bm; // Disable TCA0

ADC0.CTRLA |= ~ADC\_ENABLE\_bm; //Enable ADC

PORTD.OUT |= PIN0\_bm; //LED is off

PORTD.OUT |= PIN1\_bm; //LED is off

per=0x40;

duty=0x20;

}

}

ISR(TCA0\_HUNF\_vect){

//clear the interrupt flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

PORTD.OUTTGL = PIN0\_bm; //PIN toggle anemistiras

}

ISR(TCA0\_LUNF\_vect){

//clear the interrupt flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

PORTD.OUTTGL = PIN1\_bm; //PIN toggle vashs

}

ISR(ADC0\_WCOMP\_vect){

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

TCA0.SPLIT.CTRLA &= ~TCA\_SPLIT\_ENABLE\_bm; // Disable TCA0

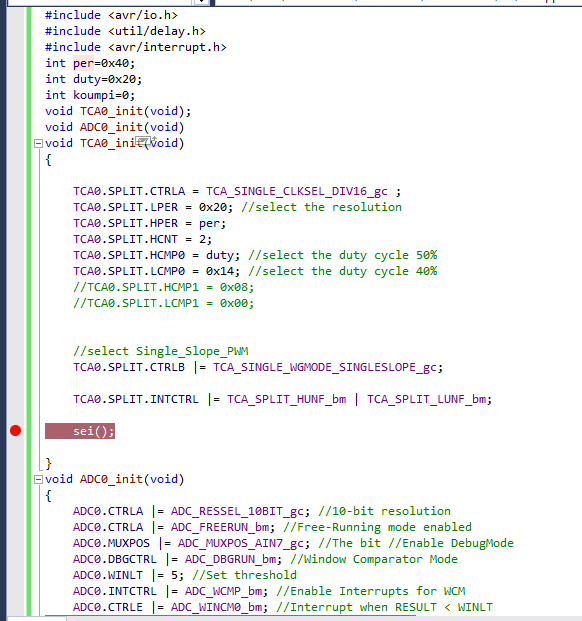
PORTD.OUTCLR= PIN2\_bm; //LED is on

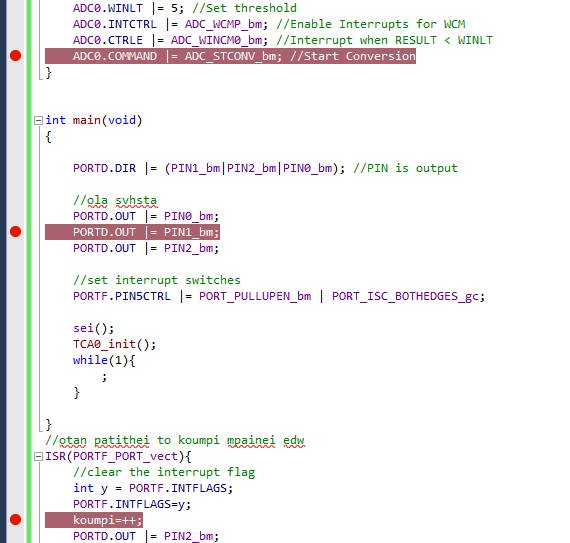
PORTD.OUT |= PIN0\_bm; //LED is off

PORTD.OUT |= PIN1\_bm; //LED is off

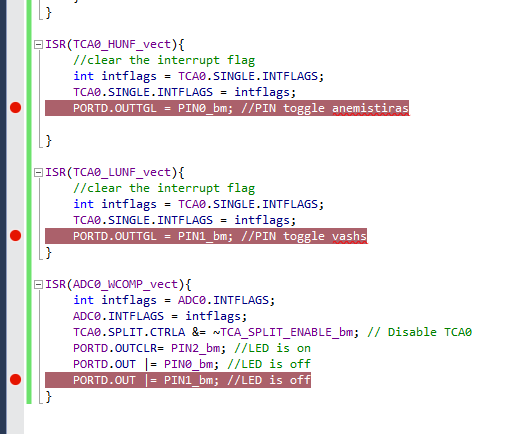
}

Screenshots με breakout points:









Σχόλια – Επεξήγηση κώδικα:

Στα ερώτηματα 1,2 προσθέσαμε τα παρακάτω:

Θέτουμε την μεταβλητή koumpi με την οποία ελέγχουμε πιο πάτημα (1ο,2ο,3ο) του κουμπιού πραγματοποιήθηκε για να γίνουν οι ανάλογες ενέργειες. Πιο συγκεκριμένα κάθε φορά που ενεργοποιήται η ISR του switch δηλαδή σε κάθε πατημα του αυξάνουμε την τιμή της μεταβλητης koumpi κατά +1.Εαν koumpi mod 3 == 1 (δηλαδη στο πρώτο πάτημα) υλοποιήται η λειτουργία των προηγούμενων ερωτημάτων, εάν koumpi mod 3 == 2 (δηλαδη στο δεύτερο πάτημα) διπλασιάζεται η ταχύτητα περιστροφης και εαν koumpi mod 3 == 0 σταματάει ο ανεμηστήρας και αρχικοποιούμε τις απαιτούμενες τιμές για όταν ξαναενεργοποιήθει το switch.