**ΕΡΓΑΣΤΗΡΙΑΚΗ ΆΣΚΗΣΗ 1**

**ΜΈΛΟΙ ΟMAΔΑΣ:**

**Σίδερης Στυλιανός:ΑΜ:1075217.έτος: 4ο**

**Ψυρρή Διονυσία: AM:1080424, έτος: 4ο**

Στην αρχή θα θέλαμε να σας επισημάνουμε κάποια βασικά σημεία της εργασίας. Για την παραγωγή των PWM χρησιμοποιήσαμε τους μετρητές TCAO και TCB0.Δεν κάναμε split διότι σε καμιά άσκηση μέχρι τώρα δεν είχαμε κάνει γνωριμία με τον μετρητή TCB0.

Για την έξοδο των PWM συνδέσαμε τις κατάλληλες εξόδους στους κατάλληλους πολυπλέκτες που συνδέονται απευθείας με τους τους καταχωρητές CMP.Αυτό έγινε με τις εντολές:

PORTMUX.TCAROUTEA |=PORTMUX\_TCA0\_PORTC\_gc;

PORTMUX.TCBROUTEA |=0x0;

Κατεβάσαμε το ρολόι του συστήματος στα 32kHz και δώσαμε στον TCB0 ρολόι από το ΜΑΙΝ CLOCK του συστήματος με τις εντολές.

CPU\_CCP = CCP\_IOREG\_gc; -> για να μπορέσουμε να αλλάξουμε τιμές σε καταχωρητές σαν τον παρακάτω για θέματα ασφαλείας.

CLKCTRL.MCLKCTRLA=CLKCTRL\_CLKSEL\_OSCULP32K\_gc;-> ρολόι στα 32ΚHz

Υπολογίσαμε τις τιμές στους καταχωρητές PER που δηλώνουν περίοδο με τον παρακάτω τύπο.

Letter

Description automatically generated

* Περίοδος για TCA0 = 32000/500 -1 = 63
* Περίοδος για TCB0 = 32000/1000 -1= 31

Οι καταχωρητές CMP που δηλώνουν το duty cycle πήραν τιμή:

* Για κίνηση βάσης :0,4\*63
* Για κίνηση λεπίδες : 0,5\*31

Για να κάνουμε έλεγχο βάλαμε μεγαλύτερες τιμές και βάλαμε και DIV 64 στο main clock:

CPU\_CCP = CCP\_IOREG\_gc;-> για να μπορέσουμε να αλλάξουμε τιμές σε καταχωρητές σαν τον παρακάτω για θέματα ασφαλείας.

CLKCTRL.MCLKCTRLB = CLKCTRL\_PDIV\_64X\_gc | CLKCTRL\_PEN\_bm;

**Μέρος πρώτο(ενεργοποίηση ανεμηστήρα με switch)**

Στην αρχή περιμένουμε σε μια while να σταθεροποιηθεί το νέο ρολόι.

Μετά κάνουμε αρχικοποίηση στους δύο timer για την παραγωγή των PWM(δεν ξεκινάνε όμως ακόμα). Για αυτή την ενέργεια έχουμε φτιάξει συναρτήσεις. Αρχικοποιούμε και το switch για λειτουργεία on off.

Ξεκινάει το πρόγραμμα και είμαστε στην while μέσα στην main.Εφόσον δεν έχουμε ενεργοποιήσει το switch on off δεν θα φύγουμε ποτέ από εκεί και δεν θα έχουμε έξοδο PWM και ο ανεμιστήρας θα είναι σε κατάσταση off.

Όταν πατηθεί το switch θα έχουμε interrupt.Μέσα στην ISR θα ενεργοποιηθούν και οι δύο μετρητές θα γίνουν δηλαδή enable με κατάλληλο **CMP** έτσι ώστε να έχουμε διαφορά φάσης στους δύο μετρητές και να μην έχουμε ταυτόχρονα interrupt (overflow)και από τους δύο ταυτόχρονα .Τώρα λοιπόν έχουμε παραγωγή PWM και ο ανεμιστήρας λειτουργεί. Στην έξοδο **PORTC** στο **PIN 0** παίρνουμε το σήμα **PWM** για το μοτέρ της βάσης και στο **PORTA PIN 2** για το μοτέρ των λεπίδων.

Τα interrupt για overflow για τους μετρητές τα θέλουμε μόνο για να ανάβουμε και να σβήνουμε τα led που μας ζητάτε γιατί όπως αναφέραμε και πιο πάνω τα **PWM** τα παίρνουμε απευθείας από τις καθορισμένες εξόδους.

Αν πατήσουμε το **switch on off** και ο ανεμιστήρας είναι σε λειτουργεία **on** τότε ο ανεμιστήρας θα κλείσει.

Για το αν ανοίξει ή αν κλείσει το καταλαβαίνουμε με μία βοηθητική μεταβλητή **fun\_on\_off** μέσα στην ISR.Το άναμμα και το σβήσιμο των κατάλληλων led έχει γίνει σύμφωνα με τις οδηγίες.

**Διάγραμμα ροής:**

**Κώδικας:**

#include <avr/io.h>

#include <avr/interrupt.h>

#define FREQ\_BASE 300

#define DUTY\_BASE FREQ\_BASE\*0.40

#define FREQ\_LEPIDES 255

#define DUTY\_LEPIDES FREQ\_LEPIDES/2

int fun\_on\_off=0;

//-------------------sinarthseis----------------------------------

void init\_timer\_base(void){

TCA0.SINGLE.CTRLA=TCA\_SINGLE\_CLKSEL\_DIV1\_gc;//xoris prescaler

TCA0.SINGLE.PER=FREQ\_BASE;

TCA0.SINGLE.CMP0=DUTY\_BASE;

//RITHMISI MODE

TCA0.SINGLE.CTRLB=TCA\_SINGLE\_CMP0EN\_bm|TCA\_SINGLE\_WGMODE\_SINGLESLOPE\_gc;

//ENABLE INTERRUPR

TCA0.SINGLE.INTCTRL=TCA\_SINGLE\_OVF\_bm;

TCA0.SINGLE.INTCTRL|=TCA\_SINGLE\_CMP0\_bm;

TCA0.SINGLE.CNT=0;

//H EXODOS PWM THA BGENEI STO PORTC

PORTMUX.TCAROUTEA |=PORTMUX\_TCA0\_PORTC\_gc;

}

void init\_timer\_lepides(void){

//ENERGOPIHSI GIA PWM

TCB0.CTRLA = (TCB0.CTRLA & ~TCB\_CLKSEL\_gm) | TCB\_CLKSEL\_CLKDIV1\_gc;

TCB0.CTRLB |= TCB\_CCMPEN\_bm;

TCB0.CTRLB |= TCB\_CNTMODE\_PWM8\_gc;

TCB0\_CCMPL= FREQ\_LEPIDES;

TCB0\_CCMPH= DUTY\_LEPIDES;

TCB0.CNT=15;

//H EXODOS NA BGENEI STO PORTA

PORTMUX.TCBROUTEA |=0x0;

TCB0.INTCTRL|=TCB\_CAPT\_bm;

}

int main(void)

{

//epitrepsi prosvasis se eidikoys kataxorites

CPU\_CCP = CCP\_IOREG\_gc;

//syxnothtia sta 32 KHh

CLKCTRL.MCLKCTRLA=CLKCTRL\_CLKSEL\_OSCULP32K\_gc;

//CLKCTRL.MCLKCTRLB = CLKCTRL\_PDIV\_64X\_gc | CLKCTRL\_PEN\_bm;

//ELENXOS AN TO ROLOI EINAI STATHERO

while (CLKCTRL.MCLKSTATUS & CLKCTRL\_SOSC\_bm)

{

;

}

//Led gia sixnothta

PORTD.DIR |= PIN0\_bm|PIN1\_bm;

//Exodoi gia pwm palmo

PORTC.DIR |= PIN0\_bm;

PORTA.DIR |= PIN2\_bm;

init\_timer\_base();

init\_timer\_lepides();

//on off switch mono se kathodoki akmi

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

sei();

while (1)

{

PORTE.OUT|= 0b00000000;

}

cli();

}

//----------------------ISR------------------------------

ISR(PORTF\_PORT\_vect){

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

if(!fun\_on\_off){ //an eimastan off

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLA |=TCA\_SINGLE\_ENABLE\_bm;

TCB0\_CNT=15;

TCB0.CTRLA |=TCB\_ENABLE\_bm;

fun\_on\_off=1;

}

else{ //an eimastan on

TCA0.SINGLE.CTRLA &=0x0;

TCB0.CTRLA &=0x0;

//sbino ta led

PORTD.OUTCLR |=0x0F;

fun\_on\_off=0;

}

}

ISR(TCA0\_OVF\_vect){//gia otan exoyme ena pliri kyklo

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

//led on off

PORTD.OUTTGL=PIN1\_bm;

}

ISR(TCA0\_CMP0\_vect){//otan teleivsei to duty cycle

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

}

ISR(TCB0\_INT\_vect)//lepides otan exoyme ena pliri kyklo

{

int intflags = TCB0.INTFLAGS;

TCB0.INTFLAGS = intflags;

//led on off

PORTD.OUTTGL =PIN0\_bm;

}

**Μέρος δεύτερο(διακοπή λειτουργείας με ADC)**

Σε αυτό το μέρος προσθέσαμε και τον ADC σε λειτουργεία free running mode.Για την αρχικοποίηση έχουμε φτιάξει μια συνάρτηση. Καθώς ανάβουμε τον ανεμιστήρα το πρώτο πράγμα που ενεργοποιείτε είναι ο ADC για λόγους ασφαλείας. Μετά ενεργοποιούνται οι μετρητές για να παράγουν έξοδο PWM.Αν λοιπόν καθώς ο ανεμιστήρας είναι σε λειτουργία και κάποιος βάλει κοντά το χέρι του τότε ο υποτιθέμενος αισθητήρας θα δώσει μια τάση στον ADC και αυτός θα την μετατρέψει σε μια διακριτή τιμή μικρότερη από το κατώφλι που του έχουμε ορίσει. Θα γίνει interrupt και μέσα στην ISR απενεργοποιούνται οι μετρητές κατά συνέπεια και τα PWM και σταματάνε οι λεπίδες και η κίνηση της βάσης του ανεμιστήρα.

Αν κάποιος προσπαθήσει να πατήσει το switch να ανοίξει ο ανεμιστήρας αλλά ο ADC δείχνει ότι υπάρχει κάτι ακόμα μπροστά από τον αισθητήρα τότε ποτέ δεν θα ενεργοποιηθούν οι λεπίδες για λόγους ασφαλείας. Έχουμε και ένα **led pin2** για να βλέπουμε πως έχουμε εμπόδιο στον αισθητήρα .Κατά τα υπόλοιπα είναι όλα όπως και στο πρώτο μέρος.

**Διάγραμμα ροής:**

**Κώδικας:**

#include <avr/io.h>

#include <avr/interrupt.h>

#define FREQ\_BASE 255

#define DUTY\_BASE FREQ\_BASE\*0.40

#define FREQ\_LEPIDES 127

#define DUTY\_LEPIDES FREQ\_LEPIDES/2

int fun\_on\_off=0;

//-------------------sinarthseis----------------------------------

void init\_timer\_base(void){

TCA0.SINGLE.CTRLA=TCA\_SINGLE\_CLKSEL\_DIV1\_gc;//xoris prescaler

TCA0.SINGLE.PER=FREQ\_BASE;

TCA0.SINGLE.CMP0=DUTY\_BASE;

//RITHMISI MODE

TCA0.SINGLE.CTRLB=TCA\_SINGLE\_CMP0EN\_bm|TCA\_SINGLE\_WGMODE\_SINGLESLOPE\_gc;

//ENABLE INTERRUPR

TCA0.SINGLE.INTCTRL=TCA\_SINGLE\_OVF\_bm;

//TCA0.SINGLE.INTCTRL|=TCA\_SINGLE\_CMP0\_bm;

TCA0.SINGLE.CNT=0;

//H EXODOS PWM THA BGENEI STO PORTC

PORTMUX.TCAROUTEA |=PORTMUX\_TCA0\_PORTC\_gc;

//Disable kai ta dio PWM

TCA0.SINGLE.CTRLA &=0x0;

TCB0.CTRLA &=0x0;

}

void init\_timer\_lepides(void){

//ENERGOPIHSI GIA PWM

TCB0.CTRLA = (TCB0.CTRLA & ~TCB\_CLKSEL\_gm) | TCB\_CLKSEL\_CLKDIV1\_gc;

TCB0.CTRLB |= TCB\_CCMPEN\_bm;

TCB0.CTRLB |= TCB\_CNTMODE\_PWM8\_gc;

TCB0\_CCMPL= FREQ\_LEPIDES;

TCB0\_CCMPH= DUTY\_LEPIDES;

TCB0.CNT=50;

//H EXODOS NA BGENEI STO PORTA

PORTMUX.TCBROUTEA |=0x0;

TCB0.INTCTRL|=TCB\_CAPT\_bm;

}

void init\_ADC(void){

//DEXOMASTE ANALOGIKH EISIDO APO PIN7

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc;

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm;

//OTAN XEPRASO TO KATOFLI NA KANO INTERRUPT

ADC0.INTCTRL |= ADC\_WCMP\_bm;

// IFREE RUNNING MODE

ADC0.CTRLA |= ADC\_FREERUN\_bm;

//ORIO KATOFLI

ADC0.WINLT |= 10;

//OTAN RES < KATOFLI

ADC0.CTRLE |= ADC\_WINCM0\_bm;

}

int main(void)

{

//epitrepsi prosvasis se eidikoys kataxorites

CPU\_CCP = CCP\_IOREG\_gc;

//syxnothtia sta 32 KHh

CLKCTRL.MCLKCTRLA=CLKCTRL\_CLKSEL\_OSCULP32K\_gc;

//pali theloyme prosvasi

CPU\_CCP = CCP\_IOREG\_gc;

//autpo xreiazete mono gia test,an to ektelesoyme

//se ATmega den theloyme prescaler

CLKCTRL.MCLKCTRLB = CLKCTRL\_PDIV\_64X\_gc | CLKCTRL\_PEN\_bm;

//ELENXOS AN TO ROLOI EINAI STATHERO

while (!(CLKCTRL.MCLKSTATUS & 0x20))

{

;

}

//Led gia sixnothta kai ADC

PORTD.DIR |= PIN0\_bm|PIN1\_bm|PIN2\_bm;

//svista ola

PORTD.OUT |= PIN0\_bm|PIN1\_bm|PIN2\_bm;

//Exodoi gia pwm palmo

PORTC.DIR |= PIN0\_bm;

PORTA.DIR |= PIN2\_bm;

//arxiopihsi se ola

init\_timer\_base();

init\_timer\_lepides();

init\_ADC();

//on off switch mono se kathodoki akmi

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

sei();

while (1)

{

PORTE.OUT|= 0b00000000;

}

cli();

}

//----------------------ISR------------------------------

ISR(PORTF\_PORT\_vect){

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

if(!fun\_on\_off){

//prota afoy anavoyme energopoioyme ton ADC

ADC0.CTRLA |= ADC\_ENABLE\_bm;

ADC0.COMMAND |= ADC\_STCONV\_bm;

//meta energopioyme ta PWM

TCA0.SINGLE.CNT=10;

TCA0.SINGLE.CTRLA |=TCA\_SINGLE\_ENABLE\_bm;

TCB0\_CNT=50;

TCB0.CTRLA |=TCB\_ENABLE\_bm;

fun\_on\_off=1;

//svino pithano anameno to led ADC

PORTD.OUT |=PIN2\_bm;

}

else{

TCA0.SINGLE.CTRLA &=0x0;

TCB0.CTRLA &=0x0;

//sbino ta led

PORTD.OUT |=0x03;

//disable ADC

ADC0.CTRLA &=0x02;

fun\_on\_off=0;

}

}

ISR(TCA0\_OVF\_vect){//gia otan exoyme ena pliri kyklo

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

//led on off

PORTD.OUTTGL=PIN1\_bm;

}

ISR(TCA0\_CMP0\_vect){//otan teleivsei to duty cycle

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

}

ISR(TCB0\_INT\_vect)//lepides otan exoyme ena pliri kyklo

{

int intflags = TCB0.INTFLAGS;

TCB0.INTFLAGS = intflags;

//led on off

PORTD.OUTTGL =PIN0\_bm;

}

ISR(ADC0\_WCOMP\_vect) { //interrupt ??? ??? ADC

//katharizo tis simaies gia tous metrites giati

//thelo na apofoigo na kanei interrupt meta apo edo

int intflags\_1 = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags\_1;

int intflags\_2 = TCB0.INTFLAGS;

TCB0.INTFLAGS = intflags\_2;

//apenergopoio PWM

TCA0.SINGLE.CTRLA &=0x0;

TCB0.CTRLA &=0x0;

//disable ADC

ADC0.CTRLA &=0x02;

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

//led ADC ON

PORTD.OUTCLR= 0b00000100;

//led kinisis anemistiron off

PORTD.OUT |= PIN0\_bm|PIN1\_bm;

//gia na xero oti eimai off

fun\_on\_off=0;

}

**Μέρος τρίτο(δεύτερη λειτουργία του παλμού της κυκλικής κίνησης των λεπίδων)**

Για να πετύχουμε το ζητούμενο δεν αλλάχθηκαν πάρα πολλά σε σχέση με το προηγούμενο μέρος. Αρχικά ο διακόπτης μας πλέον έχει 3 θέσεις, μια **off** , **μια on σε λεπίδες στα 500Hz**  και **on με λεπίδες στα 1000Ηz**.Την δουλειά του ελέγχου για το πια είναι η κατάσταση είναι η μεταβλητή **fun\_on\_off.**

H ISR του switch έχει μια παραπάνω if else να αλλάζει την περίοδο και το duty cycle του PWM που παράγει το TCB0 για τις λεπίδες.

**Διάγραμμα ροής:**

**Κώδικας:**

#include <avr/io.h>

#include <avr/interrupt.h>

#define FREQ\_BASE 255

#define DUTY\_BASE FREQ\_BASE\*0.40

#define FREQ\_LEPIDES 127

#define DUTY\_LEPIDES FREQ\_LEPIDES/2

int fun\_on\_off=0;

//-------------------sinarthseis----------------------------------

void init\_timer\_base(void){

TCA0.SINGLE.CTRLA=TCA\_SINGLE\_CLKSEL\_DIV1\_gc;//xoris prescaler

TCA0.SINGLE.PER= FREQ\_BASE;

TCA0.SINGLE.CMP0 = DUTY\_BASE;

//RITHMISI MODE

TCA0.SINGLE.CTRLB=TCA\_SINGLE\_CMP0EN\_bm|TCA\_SINGLE\_WGMODE\_SINGLESLOPE\_gc;

//ENABLE INTERRUPR

TCA0.SINGLE.INTCTRL=TCA\_SINGLE\_OVF\_bm;

//TCA0.SINGLE.INTCTRL|=TCA\_SINGLE\_CMP0\_bm;

//H EXODOS PWM THA BGENEI STO PORTC

PORTMUX.TCAROUTEA |=PORTMUX\_TCA0\_PORTC\_gc;

//Disable kai ta dio PWM

TCA0.SINGLE.CTRLA &=0x0;

TCB0.CTRLA &=0x0;

}

void init\_timer\_lepides(void){

//ENERGOPIHSI GIA PWM

TCB0.CTRLA = (TCB0.CTRLA & ~TCB\_CLKSEL\_gm) | TCB\_CLKSEL\_CLKDIV1\_gc;

TCB0.CTRLB |= TCB\_CCMPEN\_bm;

TCB0.CTRLB |= TCB\_CNTMODE\_PWM8\_gc;

//TCB0.CNT=50;

//H EXODOS NA BGENEI STO PORTA

PORTMUX.TCBROUTEA |=0x0;

TCB0.INTCTRL|=TCB\_CAPT\_bm;

}

void init\_ADC(void){

//DEXOMASTE ANALOGIKH EISIDO APO PIN7

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc;

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm;

//OTAN XEPRASO TO KATOFLI NA KANO INTERRUPT

ADC0.INTCTRL |= ADC\_WCMP\_bm;

// IFREE RUNNING MODE

ADC0.CTRLA |= ADC\_FREERUN\_bm;

//ORIO KATOFLI

ADC0.WINLT |= 10;

//OTAN RES < KATOFLI

ADC0.CTRLE |= ADC\_WINCM0\_bm;

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

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

}

int main(void)

{

//epitrepsi prosvasis se eidikoys kataxorites

CPU\_CCP = CCP\_IOREG\_gc;

//syxnothtia sta 32 KHh

CLKCTRL.MCLKCTRLA=CLKCTRL\_CLKSEL\_OSCULP32K\_gc;

//pali theloyme prosvasi

CPU\_CCP = CCP\_IOREG\_gc;

//autpo xreiazete mono gia test,an to ektelesoyme

//se ATmega den theloyme prescaler

CLKCTRL.MCLKCTRLB = CLKCTRL\_PDIV\_64X\_gc | CLKCTRL\_PEN\_bm;

//ELENXOS AN TO ROLOI EINAI STATHERO

while (!(CLKCTRL.MCLKSTATUS & 0x20))

{

;

}

//Led gia sixnothta kai ADC

PORTD.DIR |= PIN0\_bm|PIN1\_bm|PIN2\_bm;

//svista ola

PORTD.OUT |= PIN0\_bm|PIN1\_bm|PIN2\_bm;

//Exodoi gia pwm palmo

PORTC.DIR |= PIN0\_bm;

PORTA.DIR |= PIN2\_bm;

//arxiopihsi se ola

init\_timer\_base();

init\_timer\_lepides();

init\_ADC();

//on off switch mono se kathodoki akmi

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

sei();

while (1)

{

PORTE.OUT|= 0b00000000;

}

cli();

}

//----------------------ISR------------------------------

ISR(PORTF\_PORT\_vect){

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

if(fun\_on\_off==0){

//prota afoy anavoyme energopoioyme ton ADC

//gia logous asfaleias

ADC0.CTRLA |= ADC\_ENABLE\_bm;

ADC0.COMMAND |= ADC\_STCONV\_bm;

//meta energopioyme ta PWM

TCA0.SINGLE.CNT=10;

TCA0.SINGLE.CTRLA |=TCA\_SINGLE\_ENABLE\_bm;

//arxikopoio sixnothtes kai duty cycle giati

//den einai panta idia gia TCB0

TCB0\_CCMPL= FREQ\_LEPIDES;

TCB0\_CCMPH= DUTY\_LEPIDES;

TCB0\_CNT=50;

TCB0.CTRLA |=TCB\_ENABLE\_bm;

fun\_on\_off=1;

//svino pithano anameno to led ADC

PORTD.OUT |=PIN2\_bm;

}

else if(fun\_on\_off==1){ //allazoyme sixnotita kai duty cycle se lepides

TCB0\_CCMPL= FREQ\_LEPIDES\*2;

TCB0\_CCMPH= (FREQ\_LEPIDES\*2)/2;

TCB0.CNT=15;

fun\_on\_off=2;

}

else{

//disable kai oi dio kai to fun svinei

TCA0.SINGLE.CTRLA &=0x0;

TCB0.CTRLA &=0x0;

//sbino ta led

PORTD.OUT |=0x03;

//disable ADC

ADC0.CTRLA &=0x02;

fun\_on\_off=0;

}

}

ISR(TCA0\_OVF\_vect){//gia otan exoyme ena pliri kyklo

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

//led on off

PORTD.OUTTGL=PIN1\_bm;

}

ISR(TCA0\_CMP0\_vect){//otan teleivsei to duty cycle

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

}

ISR(TCB0\_INT\_vect)//lepides otan exoyme ena pliri kyklo

{

int intflags = TCB0.INTFLAGS;

TCB0.INTFLAGS = intflags;

//led on off

PORTD.OUTTGL =PIN0\_bm;

}

ISR(ADC0\_WCOMP\_vect) { //interrupt ??? ??? ADC

//katharizo tis simaies gia tous metrites giati

//thelo na apofoigo na kanei interrupt meta apo edo

int intflags\_1 = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags\_1;

int intflags\_2 = TCB0.INTFLAGS;

TCB0.INTFLAGS = intflags\_2;

//apenergopoio PWM

TCA0.SINGLE.CTRLA &=0x0;

TCB0.CTRLA &=0x0;

//disable ADC

ADC0.CTRLA &=0x02;

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

//led ADC ON

PORTD.OUTCLR= 0b00000100;

//led kinisis anemistiron off

PORTD.OUT |= PIN0\_bm|PIN1\_bm;

//gia na xero oti eimai off

fun\_on\_off=0;

}

**ΤΕΛΟΣ**