Εικόνα που περιέχει έμβλημα, μπρούντζος, κέρμα, κύκλος

Περιγραφή που δημιουργήθηκε αυτόματα

Εργαστήριο Προηγμένοι Μικροεπεξεργαστές

4η Εργαστηριακή Άσκηση:

Λειτουργία Έξυπνου Θερμοκηπίου

***Μέλη Ομάδας:***

***Μουσελέ Χριστιάνα ΑΜ:1090068***

***Μπαλάσης Ιωάννης ΑΜ:1084631***

Για την υλοποίηση της εργασίας έχει ληφθεί υπόψιν ότι η λειτουργία του SW6 θα προσομοιωθεί με έναν παλμό PWM με περίοδο T1 = 1 ms και Κύκλο Λειτουργίας (Duty Cycle ) 50%. Επομένως, για τον υπολογισμό της περιόδου του παλμού έχουμε:

**Κώδικας 1ου Ερωτήματος**

#include <avr/io.h>

#include <avr/interrupt.h>

int z = 0; // ADC Interrupt Flag

int water = 0; // var if watering

int air = 0; //var if ventilation

int main(){

PORTD.DIR |= PIN0\_bm; //PIN is output

PORTD.DIR |= PIN1\_bm; //PIN is output

while(1){

//initialize the ADC for Free-Running mode

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

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode

//Window Comparator Mode

ADC0.WINLT |= 4; //Set threshold //OTAN DIABASEI MIKROTERES APO AFTH PREPEI NA POTISTEI

ADC0.WINHT |= 8; //PREPEI NA AERISTEI AN TO PERASEI

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

ADC0.CTRLE |= 0x4; //Interrupt when RESULT < WINLT or RESULT >WINHT

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

sei(); //enable interrupts

while (z==0)

{

;

}

}

}

ISR(ADC0\_WCOMP\_vect){

int res = ADC0\_RES; //var for register RES in ADC

if(res < ADC0.WINLT)

{

PORTD.OUTCLR |= PIN0\_bm;

PORTD.OUT |= PIN1\_bm;

water=1;

sei();

}

if (res > ADC0.WINHT)

{

PORTD.OUTCLR |= PIN1\_bm;

PORTD.OUT |= PIN0\_bm;

air=1;

sei();

}

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

ADC0.CTRLA =0 ; //clear ADC

z=1;

}

**Εξήγηση Κώδικα 1ου Ερωτήματος**

Στην αρχή του κώδικα, ορίζονται οι μεταβλητές water , air και η σημαία z. Η λειτουργία που έχουν οι μεταβλητές water και air σε αυτό το ερώτημα είναι για να μας εμφανίζει ποια ενέργεια απαιτείται στο θερμοκήπιο. H μεταβλητή z είναι ένα Flag για το ADC. Στη συνέχεια ορίζονται οι έξοδοι PIN0 και PIN1 , οι οποίοι θα αντιστοιχούν σε κάθε ενέργεια ( PIN0 - πότισμα και ΡΙΝ1 – σύστημα αερισμού). Έπειτα, έχουμε την κύρια λούπα του προγράμματος στην οποία αρχικοποιούμε τον ADC , θέτουμε τα δύο Threshold WINLT και WINHT 4 και 8 αντίστοιχα , ενεργοποιούμε τις διακοπές και θέτουμε στον καταχωρητή CTRLE την τιμή 0x4 η οποία επιτρέπει τις διακοπές όταν ισχύουν οι εξής σχέσεις: RES < WINLT και RES > WINHT. Όταν ικανοποιούνται οι προηγούμενες σχέσεις πραγματοποιείται η διακοπή (ADC\_WC0MP\_vect) στην οποία αρχικά ορίζουμε την μεταβλητή res για τις τιμές που θα έχει ο καταχωρητής RES του ΑDC. Έπειτα πραγματοποιείται έλεγχος έτσι ώστε να καταλάβουμε ποια από τις δύο σχέσεις έχει προκαλέσει την διακοπή. Ελέγχουμε δηλαδή αν η τιμή που έλαβε ο ADC είναι μεγαλύτερη από το μεγαλύτερο κατώφλι ή μικρότερη από το μικρότερο κατώφλι. Αν ισχύει η σχέση RES < WINLT δηλαδή ικανοποιείται η συνθήκη res < ADC0.WINLT τότε ανάβει το LED του PIN0 και η μεταβλητή water θα πάρει την τιμή 1. Αντίστοιχα, αν ισχύει η σχέση RES > WINHT δηλαδή ικανοποιείται η συνθήκη res > ADC0.WINHT τότε ανάβει το LED του PIN1 και η μεταβλητή air θα πάρει την τιμή 1. Τέλος, καθαρίζουμε τα intflags, μηδενίζουμε τον ADC και θέτουμε το z ίσο με 1 για να βγούμε από τη λούπα.

**Διάγραμμα Ροής 1ου Ερωτήματος**

**Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, σχεδίαση

Περιγραφή που δημιουργήθηκε αυτόματα**

**Κώδικας 2ου και 3ου Ερωτήματος**

#include <avr/io.h>

#include <avr/interrupt.h>

int z = 0; // ADC Interrupt Flag

int water = 0; // var if watering

int air = 0; //var if ventilation

int akmh=0; //var for count rising edges

int main(){

PORTD.DIR |= PIN0\_bm; //PIN is output

PORTD.DIR |= PIN1\_bm; //PIN is output

PORTD.DIR |= PIN2\_bm; //PIN is output

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //Set PIN5 PORTF for Switch

PORTF.PIN6CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //Set PIN6 PORTF for Switch

while(1){

//initialize the ADC for Free-Running mode

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

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Enable Debug Mode

//Window Comparator Mode

ADC0.WINLT |= 4; //Set threshold //OTAN DIABASEI MIKROTERES APO AFTH PREPEI NA POTISTEI

ADC0.WINHT |= 8; //PREPEI NA AERISTEI AN TO PERASEI

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

ADC0.CTRLE |= 0x4; //Interrupt when RESULT < WINLT or RESULT >WINHT

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

sei(); //enable interrupts

while (z==0)

{

;

}

if (water==2) {

TCA0.SINGLE.CNT = 0;

//Clear Counter

TCA0.SINGLE.CTRLB = 0; //Normal Mode

TCA0.SINGLE.CMP0 = ADC0.WINLT - ADC0\_RES ;//When CMP0 reaches this value -> interrupt

TCA0.SINGLE.CTRLA = TCA\_SINGLE\_CLKSEL\_DIV1024\_gc;

TCA0.SINGLE.CTRLA |= 1;

//Enable

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_CMP0\_bm; //Interrupt Enable (=0x10)

sei();

//Begin Accepting Interrupt Signals

while (water==2)

{

;

}

}

if (air==2) {

TCA0.SINGLE.CTRLA=TCA\_SINGLE\_CLKSEL\_DIV1024\_gc;

TCA0.SINGLE.PER = 20; //Select the resolution

TCA0.SINGLE.CMP0 = 10; //Select the duty cycle

//Select Single\_Slope\_PWM

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

//Enable interrupt Overflow

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

//Enable interrupt COMP0

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

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

sei();

while (akmh<4) {

;

}

}

sei();

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

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

PORTD.OUT |= PIN2\_bm; //LED is off

}

}

ISR(PORTF\_PORT\_vect){

cli();

if ( PORTF.INTFLAGS & 0x20 && water==1)

{

water++;

}

if (PORTF.INTFLAGS & 0x40 && air == 1)

{

air++;

}

else {

PORTD.OUTCLR |= PIN0\_bm;

PORTD.OUTCLR |= PIN1\_bm;

PORTD.OUTCLR |= PIN2\_bm;

}

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

sei();

}

ISR(TCA0\_CMP0\_vect){

cli();

if (water==2) {

TCA0.SINGLE.CTRLA = 0;

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

water=0; // epistrefei sthn arxikh thesh

TCA0.SINGLE.CNT = 0; // timer clear

PORTD.OUT |= PIN0\_bm; //led0 off

}

else { //when it's going down

//clear the interrupt flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

PORTD.OUT |= PIN2\_bm; //LED is off

}

sei();

}

//when it's going up

ISR(TCA0\_OVF\_vect){

cli();

//clear the interrupt flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

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

akmh++;

sei();

}

ISR(ADC0\_WCOMP\_vect){

int res = ADC0\_RES;

if(res < ADC0.WINLT)

{

PORTD.OUTCLR |= PIN0\_bm;

PORTD.OUT |= PIN1\_bm;

water=1;

sei();

}

if (res > ADC0.WINHT)

{

PORTD.OUTCLR |= PIN1\_bm;

PORTD.OUT |= PIN0\_bm;

air=1;

sei();

}

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

ADC0.CTRLA =0 ;

z=1;

}

**Εξήγηση Κώδικα 2ου και 3ου Ερωτήματος**

Για την πραγματοποίηση των ερωτημάτων 2 και 3 ενεργοποιήσαμε αρχικά τους δύο διακόπτες PIN5 και PIN6 του PORTF. Όταν καλεστεί η αντίστοιχη διακοπή (ISR(PORTF\_PORT\_vect)) , θα πραγματοποιηθεί έλεγχος με bit masking για να ελέγξουμε ποιος διακόπτης πατήθηκε . Ανάλογα με το ποιος διακόπτης έχει πατηθεί, έχουμε έναν έλεγχο που ελέγχει αν έχει πατηθεί την σωστή στιγμή if ( PORTF.INTFLAGS & 0x20 && water==1). Στην συγκεκριμένη περίπτωση , αν η τιμή είναι water==1 δηλαδή αν θέλει πότισμα τότε αυξάνουμε κατά 1 την τιμή της μεταβλητής water. Αλλιώς αν η τιμή water δεν είναι 1 σημαίνει ότι έχουμε πατήσει το λάθος κουμπί και ενεργοποιείται η λειτουργία που ειδοποιεί τον χρήστη για λάθος εισαγωγή κουμπιού ( ανάβουν όλα τα led) . Αντίστοιχα έχουμε τον ίδιο έλεγχο για το αν έχει πατηθεί το PIN6 if (PORTF.INTFLAGS & 0x40 && air == 1), όπου ελέγχει και εδώ αν είναι σωστή η χρήση αυτού του διακόπτη εξετάζοντας την τιμή της μεταβλητής air. Αν είναι σωστή τότε αυξάνουμε την τιμή κατά 1 . Αν όχι , τότε έχουμε την αντίστοιχη λειτουργία λάθους. Όταν βγούμε από την διακοπή , έχουμε προσθέσει δύο ελέγχους την κύρια λούπα του προγράμματός μας if (air==2) και if (water==2) για την διαδικασία του αερισμού και ποτίσματος αντίστοιχα. Στην διαδικασία του ποτίσματος έχουμε αρχικοποιήσει έναν Timer ο οποίος θα μετράει για ADC0.WINLT - ADC0\_RES χρόνο και θα τρέχει καθώς water==2. Όταν παρέλθει αυτός ο χρόνος τότε θα προκαλέσει μια διακοπή ISR(TCA0\_CMP0\_vect) στην οποία αφού ελέγξουμε ότι είμαστε στην διαδικασία ποτίσματος τότε θα καθαρίσουμε τις τιμές από τα flags και τον timer καθώς και την τιμή της μεταβλητής water την οποία θα θέσουμε με 0 έτσι ώστε να είναι έτοιμο το σύστημα μας να δεχτεί την επόμενη τιμή από τον ADC. Ο επόμενος έλεγχος που υλοποιήθηκε είναι αυτός για την διαδικασία του αερισμού. Αρχικά αρχικοποιούμε έναν PWM παλμό, ο οποίος θα οδηγεί το LED2 του PORTD με περίοδο Τ1 = 1 ms και duty cycle 50% . Όταν είμαστε στην ανερχόμενη ακμή θα πραγματοποιείται η διακοπή ISR(TCA0\_OVF\_vect) στην οποία αυξάνουμε την μεταβλητή akmh κατά 1 και θα ανάβει το PIN2. Στην κατερχόμενη ακμή θα ενεργοποιείται η διακοπή του ISR(TCA0\_CMP0\_vect) , αφού γίνει έλεγχος ότι δεν είμαστε στο κομμάτι του ποτίσματος θα ισχύουν οι εντολές του else , άρα θα καθαριστούν τα flags και θα σβήσει το PIN2 . Αυτό θα γίνεται μέχρι να παρέλθουν τέσσερις ανερχόμενες ακμές (rising edges) του παλμού ( while (akmh<4) {;} ) , και μετά θα σβήσουν όλα τα led και το σύστημα θα είναι έτοιμο να δεχτεί την επόμενη τιμή από τον ADC.

**Διάγραμμα Ροής 2ου και 3ου Ερωτήματος**

Εικόνα που περιέχει στιγμιότυπο οθόνης, μαύρο, ασπρόμαυρο, σχεδίαση

Περιγραφή που δημιουργήθηκε αυτόματα