

Εργαστήριο Προηγμένοι Μικροεπεξεργαστές

**2η Εργαστηριακή Άσκηση : Έξυπνη οικιακή συσκευή με κίνηση στον χώρο**

***Μέλη Ομάδας :***

***Μουσελέ Χριστιάνα ΑΜ: 1090068***

***Μπαλάσης Ιωάννης ΑΜ: 1084631***

**Κώδικας 1ου Ερωτήματος:**

#include <avr/io.h>

#include <avr/interrupt.h>

#define left\_led 4

int x=0;

int y=0;

int z=0; //Var for Corner Counter

int main(){

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

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

while (z<=4){

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on // Forward

PORTD.OUT |= PIN2\_bm; //PIN2 is off // NOT Left

//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

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

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

sei();

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

while (y==0) {

}

PORTD.OUT |= PIN1\_bm; //PIN1 is off // NOT Forward

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

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0= left\_led;

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

TCA0.SINGLE.CTRLA |=1;

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

while (x==0) {

}

x=0;

cli();

}

}

ISR(ADC0\_WCOMP\_vect){

cli();

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

ADC0.CTRLA = 0; //Disable ADC

y=1; //MUST TURN LEFT(EXIT THE FIRST LOOP)

sei();

}

ISR(TCA0\_CMP0\_vect){

cli();

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

z++;

x=1; //EXIT THE SECOND LOOP

y=0;

sei();

}

**Εξήγηση Κώδικα 1ου Ερωτήματος**

Θα ορίσουμε δύο μεταβλητές x,y τις οποίες θα τις θέσουμε ίσες με μηδέν και θα τις χρησιμοποιήσουμε ως Flags για τον χρόνο του Timer και για τη μικρότερη τιμή του Threshold στον ADC. Επίσης ορίζουμε και μία μεταβλητή z η οποία αφορά τις γωνίες ενός σπιτιού και την θέτουμε και αυτή ίση με το 0. Ακόμα , ορίζουμε το PIN1 και το PIN2 ως εξόδους. Για όσο διάστημα η μεταβλητή z – δηλαδή οι γωνίες του σπιτιού- είναι μικρότερο από το 4 (λόγω ότι έχω τετράγωνο σπίτι) θα έχουμε την εξής λειτουργία:

* Θέτουμε το PIN1 PORTD ίσο με 0 έτσι ώστε να ανάψει το LED που έχουμε ορίσει ότι η κίνηση θα είναι ευθεία (PIN1).
* Θέτουμε το PIN2 PORTD ίσο με 1 έτσι ώστε να κλείσει (σβήσει) το LED που έχουμε ορίσει ότι θα πραγματοποιείται αριστερή στροφή (PIN2).
* Ο ADC πρέπει να είναι σε Free-Running Mode διότι δέχεται συνεχές σήμα μόνο από τον μπροστινό αισθητήρα.
* Ορίζουμε το Threshold και ενεργοποιούμε τα Interruption όταν ισχύει ότι RES<WINLT.
* Όταν γίνει η διακοπή, θα κληθεί η ISR -στην οποία έχω Clear Flags- και θα έχουμε αλλαγή της τιμής του y από 0 σε 1 με σκοπό να βγούμε από το While Loop.
* Θέτουμε το PIN1 PORTD ίσο με 1 έτσι ώστε να κλείσει (σβήσει) το LED που έχουμε ορίσει ότι η κίνηση θα είναι ευθεία(PIN1).
* Θέτουμε το PIN2 PORTD ίσο με 0 έτσι ώστε να ανάψει το LED που έχουμε ορίσει ότι θα πραγματοποιηθεί αριστερή στροφή.
* Έπειτα γίνεται Clear TCA Counter και τον θέτουμε σε Normal Mode ενώ επίσης θέτουμε και την τιμή που θέλουμε να γίνει το Interrupt (ο χρόνος που θέλουμε για την αριστερή στροφή) ενώ τέλος ορίζουμε τιμή του Prescaler, κάνουμε Enable τον Timer και τα Interrupts.
* Όσο ο Timer δεν έχει φθάσει την τιμή που εμείς έχουμε ορίσει(left\_led 4) , το πρόγραμμα θα μένει μέσα σε ένα While Loop.
* Όταν ο Timer φθάσει την τιμή που ορίσαμε τότε θα γίνει διακοπή, θα κληθεί η ISR (καθαρισμός Flags) θα αυξηθούν οι αριστερές στροφές (aristera++) και θα αλλάξει το x από 0 σε 1 για να βγούμε από το While Loop.

**Διάγραμμα Ροής 1ου Ερωτήματος**

**Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, κύκλος

Περιγραφή που δημιουργήθηκε αυτόματα**

**Κώδικας 2ου Ερωτήματος**

#include <avr/io.h>

#include <avr/interrupt.h>

#define Sensor1 11 //Forward Sensor

#define Sensor2 10 //Side Sensor

#define left\_led\_on 4

#define right\_led\_on 4

int x=0; //for TCA0\_CMP0\_vect

int y=0; //for ADC0\_WC0MP\_vect

int aristera=0;

int main(){

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

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

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

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

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

//Window Comparator Mode

ADC0.WINLT |= 4; //Set threshold

ADC0.WINHT |= 4; //Set threshold

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

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

while (aristera<6){

PORTD.OUT |= PIN0\_bm; //PIN0 is off //NOT Right

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on //Forward

PORTD.OUT |= PIN2\_bm; //PIN2 is off //N0T Left

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=Sensor2;

TCA0.SINGLE.CTRLA |=1;

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

sei();

while(x==0){

//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.CTRLE = 0b00000010; //Interrupt when RESULT > WINHT //distance from the wall(right)

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

while ((y==0)&&(x==0)) {

}

if(x==0){

PORTD.OUT |= PIN1\_bm; //PIN1 is off//NOT Forward

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on//Right

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=right\_led\_on;

TCA0.SINGLE.CTRLA |=1;

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

while (x==0) {

}

}

}

cli();

x=0;

if(y==0){

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=Sensor1;

TCA0.SINGLE.CTRLA |=1;

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

ADC0.CTRLA = 0;

sei();

while(x==0){

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

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

ADC0.CTRLE = 0b00000001; //Interrupt when RESULT < WINLT

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

while ((y==0)&&(x==0)) {

}

if(x==0){

aristera++;

PORTD.OUT |= PIN1\_bm; //PIN1 is off//NOT Forward

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

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=left\_led\_on;

TCA0.SINGLE.CTRLA |=1;

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

while (x==0) {

}

}

}

cli();

x=0;

}

y=0;

}

}

ISR(ADC0\_WCOMP\_vect){

cli();

int intflags = ADC0.INTFLAGS;

ADC0.CTRLA = 0;

ADC0.INTFLAGS = intflags;

y=1;

sei();

}

ISR(TCA0\_CMP0\_vect){

cli();

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

x=1;

sei();

}

**Εξήγηση Κώδικα 2ου Ερωτήματος**

**Διάγραμμα Ροής 2ου Ερωτήματος**

**Εικόνα που περιέχει στιγμιότυπο οθόνης, μαύρο, ασπρόμαυρο, σχεδίαση

Περιγραφή που δημιουργήθηκε αυτόματα**

**Κώδικας 3ου Ερωτήματος**

#include <avr/io.h>

#include <avr/interrupt.h>

#define left\_led\_on 4

#define right\_led\_on 4

#define Sensor1 11 //Forward Sensor

#define Sensor2 10 //Side Sensor

#define all\_leds 3 //Time for all leds on

int x=0; //for TCA0\_CMP0\_vect

int y=0; //for ADC0\_WC0MP\_vect

int z=0; //for PORTF\_PORT\_vect

int aristera=0; //Counter for Left Turns

int deskia=0; //Counter for Right Turns

int turns=0; //Counter For All Turns (Right+Left)

int main(){

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

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

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

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

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc; //The Bit for ADC

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

//Window Comparator Mode

ADC0.WINLT |= 4; //Set threshold

ADC0.WINHT |= 4; //Set threshold

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

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

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

sei();

while ((aristera<6)&&(z==0)){

if(z==0){

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on //Forward

PORTD.OUT |= PIN2\_bm; //PIN2 is off //NOT Right

PORTD.OUT |= PIN0\_bm; //PIN0 is off //NOT Left

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=Sensor2;

TCA0.SINGLE.CTRLA |=1;

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

}

while((x==0)&&(z==0)){

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.CTRLE = 0b00000010; //Interrupt when RESULT > WINHT

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

while ((y==0)&&(x==0)&&(z==0)) {

}

if((x==0)&&(y==1)){

deskia++;

turns++;

PORTD.OUT |= PIN1\_bm; //PIN1 is off//NOT Forward

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on //Right

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=right\_led\_on;

TCA0.SINGLE.CTRLA |=1;

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

while (x==0) {

}

}

}

x=0;

if((y==0)&&(z==0)){

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=Sensor1;

TCA0.SINGLE.CTRLA |=1;

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

ADC0.CTRLA = 0;

while((x==0)&&(z==0)){

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

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

ADC0.CTRLE = 0b00000001; //Interrupt when RESULT < WINLT

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

while ((y==0)&&(x==0)&&(z==0)) {

}

if((x==0)&&(y==1)){

aristera++;

turns++;

PORTD.OUT |= PIN1\_bm; //PIN1 is off //NOT Forward

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

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=left\_led\_on;

TCA0.SINGLE.CTRLA |=1;

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

while (x==0) {

}

}

}

x=0;

}

y=0;

}

//Reverse Mode

if(z==1){

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on //Right

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on //Forward

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

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=all\_leds; //time for leds on

TCA0.SINGLE.CTRLA |=1;

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

while(x==0){

}

x=0;

while (turns>0){

PORTD.OUTCLR = PIN1\_bm; //PIN1 is on //Forward

PORTD.OUT |= PIN2\_bm; //PIN2 is off //NOT Left

PORTD.OUT |= PIN0\_bm; //PIN0 is off //NOT Right

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=Sensor2;

TCA0.SINGLE.CTRLA |=1;

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

while(x==0){

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.CTRLE = 0b00000010; //Interrupt when RESULT > WINHT

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

while ((y==0)&&(x==0)) {

}

if(x==0){

turns--;

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

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

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=left\_led\_on;

TCA0.SINGLE.CTRLA |=1;

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

while (x==0) {

}

}

}

x=0;

if(y==0){

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=Sensor1;

TCA0.SINGLE.CTRLA |=1;

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

ADC0.CTRLA = 0;

while(x==0){

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

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

ADC0.CTRLE = 0b00000001; //Interrupt when RESULT < WINLT

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

while ((y==0)&&(x==0)) {

}

if(x==0){

turns--; //turn-1

PORTD.OUT |= PIN1\_bm; //PIN1 is off //NOT Forward

PORTD.OUTCLR = PIN0\_bm; //PIN0 is on //Right

TCA0.SINGLE.CNT=0;

TCA0.SINGLE.CTRLB=0;

TCA0.SINGLE.CMP0=right\_led\_on;

TCA0.SINGLE.CTRLA |=1;

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

while (x==0) {

}

}

}

x=0;

}

y=0;

}

z=0;

}

cli();

}

ISR(ADC0\_WCOMP\_vect){

cli();

int intflags = ADC0.INTFLAGS;

ADC0.CTRLA = 0;

ADC0.INTFLAGS = intflags;

y=1;

sei();

}

ISR(TCA0\_CMP0\_vect){

cli();

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

x=1;

sei();

}

ISR(PORTF\_PORT\_vect){

cli();

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

z=1;

sei();

}

**Εξήγηση Κώδικα 3ου Ερωτήματος**

**Διάγραμμα Ροής 3ου Ερωτήματος**

**Εικόνα που περιέχει στιγμιότυπο οθόνης, μαύρο, ασπρόμαυρο, τέχνη

Περιγραφή που δημιουργήθηκε αυτόματα**