Εργαστήριο Προηγμένοι Μικροεπεξεργαστές

Εργαστηριακή Άσκηση 2-Σαριδάκης Γεώργιος 1072478

*1.Κώδικας για το 1ο ερώτημα:*

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define ped 10

int x=0;

int y=0;

int main(){

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

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

//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 //Enable Debug Mode

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

ADC0.WINLT |= 10; //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(1){;}

if (y=4){break;}

if(x == 1){

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

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

//(σελ 219, 224, 205) 16-bit counter high and low

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

//Normal Mode (TCA\_SINGLE\_WGMODE\_NORMAL\_gc σελ 207)

TCA0.SINGLE.CTRLB = 0;

//When CMP0 reaches this value -> interrupt //CLOCK FREQUENCY/1024

TCA0.SINGLE.CMP0 = ped;

TCA0.SINGLE.CTRLA = 0x7<<1; //TCA\_SINGLE\_CLKSEL\_DIV1024\_gc σελ 224

TCA0.SINGLE.CTRLA |=1;//Enable

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

sei(); //begin accepting interrupt signals

cli();}

}

ISR(ADC0\_WCOMP\_vect){

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

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

x=1;

}

ISR(TCA0\_CMP0\_vect){

TCA0.SINGLE.CTRLA = 0; //Disable

//clear flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

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

x=0;

y=y+1;

}

*Διάγραμμα ροής για το 1ο ερώτημα:*

*Εικόνα που περιέχει διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα*

*Κώδικας για το 2ο ερώτημα:*

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define ped 10

int x,y=0;

int s=0;

int main(){

while(s<8){

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

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

//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 //Enable Debug Mode

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

ADC0.WINLT |= 10; //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

*\_delay\_ms*(3);

ADC0.CTRLA |= 0;

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

*\_delay\_ms*(2);

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

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

*\_delay\_ms*(3);

if(x == 1){

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

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

//(σελ 219, 224, 205) 16-bit counter high and low

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

//Normal Mode (TCA\_SINGLE\_WGMODE\_NORMAL\_gc σελ 207)

TCA0.SINGLE.CTRLB = 0;

//When CMP0 reaches this value -> interrupt //CLOCK FREQUENCY/1024

TCA0.SINGLE.CMP0 = ped;

TCA0.SINGLE.CTRLA = 0x7<<1; //TCA\_SINGLE\_CLKSEL\_DIV1024\_gc σελ 224

TCA0.SINGLE.CTRLA |=1;//Enable

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

sei(); //begin accepting interrupt signals

while (y==2) {

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

s = s+1;

cli();

}

}

if(x == 3){

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

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

//(σελ 219, 224, 205) 16-bit counter high and low

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

//Normal Mode (TCA\_SINGLE\_WGMODE\_NORMAL\_gc σελ 207)

TCA0.SINGLE.CTRLB = 0;

//When CMP0 reaches this value -> interrupt //CLOCK FREQUENCY/1024

TCA0.SINGLE.CMP1 = ped;

TCA0.SINGLE.CTRLA = 0x7<<1; //TCA\_SINGLE\_CLKSEL\_DIV1024\_gc σελ 224

TCA0.SINGLE.CTRLA |=1;//Enable

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_CMP1\_bm; //Interrupt Enable (=0x10)

sei(); //begin accepting interrupt signals

while (y==4) {

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

s = s+1;

cli();

}

}

if (s=8){break;}

}

}

ISR(ADC0\_WCOMP\_vect){

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

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

if(){

x = 3;

}else x=1;

}

ISR(TCA0\_CMP0\_vect){

TCA0.SINGLE.CTRLA = 0; //Disable

//clear flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

y=2;

}

ISR(TCA0\_CMP1\_vect){

TCA0.SINGLE.CTRLA = 0; //Disable

//clear flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

y=4;

}

*Διάγραμμα ροής για το 2ο ερώτημα:*

*Εικόνα που περιέχει διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα*

*2.*

*-*Για το 1o ερώτημα:

Στο πρόγραμμα αρχικά θέτουμε την μεταβλητή ped η οποία καθορίζει τον χρόνο που θα μετράει ο timer,δηλαδή τον χρόνο που θα είναι αναμμένο το LED όταν η οικιακή συσκευή κάνει στροφή αριστερά. Ακόμη αρχικοποιούνται οι μεταβλητές x,y που θα χρησιμοποιηθούν ως flags στην συνέχεια του προγράμματος. Στην main αρχικά ανάβουμε το LED1 της μπροστινής πορείας και ενεργοποιούμε τον ADC σε Free-Running mode με threshold 10. Όταν οι μετρήσεις του ADC είναι κάτω από 10,που σημαίνει αριστερή στροφή ενεργοποιείται η ISR ρουτίνα του ADC. Η ροή προγράμματος μεταβαίνει εκεί όπου και σβήνει το LED1 της μπροστινής πορείας και θέτουμε x=1 όπου και η ροή προγράμματος μεταβαίνει στην main και συγκεκριμένα στην if(x == 1) μέσα στην οποία ανάβει το LED2 για την αριστερή πορεία της συσκευής και αρχικοποιείται ο timer όπου όταν μετρήσει μέχρι την τιμή 0 ενεργοποιείται η ISR ρουτίνα του και η ροή προγράμματος μεταβαίνει εκεί. Στην ρουτίνα αυτή το LED2 σβήνει όπου και σημαίνει την ολοκλήρωση της στροφής και η μεταβλητή y που είναι ο αριθμός των στροφών της συσκευής αυξάνεται κατά 1 και η μεταβλητή x παίρνει την τιμή 0 έτσι ώστε να σβήσει ο timer και η ροή προγράμματος να επιστρέψει στην main και συγκεκριμένα να συνεχίσει να δέχεται μετρήσεις του ADC αφού βρίσκεται σε Free-Running mode.Όταν μεταβλητή y αποκτήσει την τιμή 4 σημαίνει ότι η συσκευή έχει πραγματοποιήσει 4 στροφές που σημαίνει ότι έφτασε στην αρχική της θέση και πρέπει να σταματήσει το πρόγραμμα , που συμβαίνει με την εντολή if (y=4){break;}.

-Για το 2ο ερώτημα:

Στο πρόγραμμα αρχικά θέτουμε την σταθερά ped ίση με 10 και τις μεταβλητές x,y,s ίσες με 0. Η μεταβλητή s είναι ο αριθμός των στροφών που έχει πραγματοποιήσει η συσκευή η λειτουργία της οποίας δεν μπορεί να ξεπεράσει τις 8 καθώς έχουμε 2 αμβλείες γωνίες σε ένα τετράγωνο όπου μαζί με την στροφή της αρχικής θέσης . Το πρόγραμμα θα τερματίσει με την if (s=8){break;} και θα λειτουργούν οι λειτουργίες του ADC στην main για όσο s<8. Το s θα αυξάνεται κατά ένα κάθε φορά που σβήνει ένα LED του timer ,δηλαδή πραγματοποιείται στροφή. Το πρόγραμμα ξεκινάει ενεργοποιώντας το LED1 (ευθέια πορεία). Μετά αρχικοποιώντας και ενεργοποιώντας τον ADC σε Free-Running mode όπου και μετά από 3s αλλάζει σε Single-Conversion Mode με την εντολή ADC0.CTRLA |= 0; και ενεργοποιώντας την μέτρηση τιμής του δηλαδή την μετατροπή. Με την εντολή delay\_ms() καθορίζουμε την περίοδο που θα λειτουργούν οι δύο αισθητήρες. Αν ο ADC δεχθεί τιμή RESULT μικρότερη του 10 ενώ βρίσκεται σε Free mode(αριστερή στροφή) ή μεγαλύτερη μεγαλύτερη του 10(δεξιά στροφή) εάν βρίσκεται σε Single mode τότε η ροή προγράμματος μεταβαίνει στην ISR ρουτίνα .Εκεί αφού σβήσει το LED1 της ευθείας πορείας ενεργοποιείται ανάλογα με την τιμή του RESULT η αντίστοιχη τιμή του x με την If. Αν η συσκευή πρέπει να στρίψει αριστερά θα είναι x=3 και έτσι η ροή προγράμματος επιστρέφει στην main και συγκεκριμένα στην if (x ==3) όπου και ανάβει το LED2 , αρχικοποιείται ο timer TCA0\_CMP1 και αφού μετρήσει μέχρι την σταθερά 10 η ροή προγράμματος μεταφέρεται στην ρουτίνα ISR(TCA0\_CMP1\_vect) όπου και θε τεθεί y=4 και μετά στην main με την while(y==4) θα σβήσει το LED2 που σημαίνει ότι θα έχει ολοκληρωθεί και αριστερά στροφή. Με τον ίδιο τρόπο αν η συσκευή λάβει μέτρηση μεγαλύτερη του 10 σε Single mode θα πρέπει να στρίψει δεξιά όπου και η ISR ρουτίνα θα θέσει x=1,η if στην main θα ανάψει το LED0 και θα αρχικοποιήσει τον TCA\_CMP0 ο οποίος όταν μετρήσει 10 θα συνεχίσει την ροή στην ρουτίνα ISR(TCA0\_CMP0\_vect) , οπου με y=4 θα επιστραφεί στην main και με την while θα σβήσει το LED0 δηλαδή θα έχει πραγματοποιηθεί η δεξιά στροφή.

\*Θεώρησα ότι γίνεται να χρησιμοποιήσουμε δύο timers και ότι αυτός είναι ο τρόπος υλοποίησης τους και επίσης δε βρήκα πως θα συγκρίνουμε το RESULT με το WINLT στην if στην ISR του ADC ωστόσο μου φάνηκε πως έτσι θα λυθεί.\*

-Για το 3ο ερώτημα:

Ο κώδικας και θα είναι παρόμοιος με του ερωτήματος 2 μόνο που θα έχει κάποιες αλλαγές. Στην main θα προστεθεί PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc;

sei(); //enable interrupts

while (x==0) { PORTD.OUTCLR= 0b00000010; //on }

PORTD.OUT |= 0b00000010; //off

cli(); //disenable interrupts

έτσι ώστε να μπορεί να πατηθέι το switch. Όταν αυτό πατηθεί θα ενεργοποιείται η ρουτίνα του: ISR(PORTF\_PORT\_vect){ //clear the interrupt flag

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

PORTD.OUTCLR=PIN0\_bm;

PORTD.OUTCLR=PIN1\_bm;

PORTD.OUTCLR=PIN2\_bm;

}

Έτσι σε οποιαδήποτε στιγμή πατηθεί το switch να ανάψουν όλα τα LEDS. Ταυτόχρονα πρέπει με μια σημαία να ενεργοποιείται ένας timer με μετάβαση στην main ο οποίος μόλις τελειώσει μετρήσει κάπιον χρόνο στην ρουτίνα ISR του να σβήνει τα LED και να αντιστρέφει την πορεία της συσκευής.