**Εργαστήριο Προηγμένοι Μικροεπεξεργαστές**

**Εργαστηριακή Άσκηση 4 - Σαριδάκης Γεώργιος 1072478**

**Κώδικας για την πρώτη συνάρτηση:**

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define ped 10

#define SW5\_PIN PIN5\_bm

#define SW6\_PIN PIN6\_bm

#define MAX\_TRIES 3

int x=0;

int numTries = 0;

int main() {

while (1) {

PORTF.DIRCLR = SW5\_PIN | SW6\_PIN;

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

PORTF.PIN6CTRL = PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc;

sei();

// PORTD.DIR |= 0b00000010; //PIN is output

//PORTD.OUTCLR= PIN1\_bm; //LED1 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

if (numTries >= MAX\_TRIES)

{

//πχ PORTD.OUT |= PIN1\_bm; //LED1 is off

}

}

return 0;

}

ISR(TCA0\_CMP0\_vect){

TCA0.SINGLE.CTRLA = 0; //Disable

//clear flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

PORTD.OUT |= PIN1\_bm; //LED1 is off

}

ISR(PORTF\_PORT\_vect){

int intflags = PORTF.INTFLAGS;

if (((PORTF.INTFLAGS == 0b00100000) == 1) && (x==0))

{

x=1;

}

else if (((PORTF.INTFLAGS == 0b01000000) == 1) && (x==1))

{

x=2;

}

else if (((PORTF.INTFLAGS == 0b00100000) == 1) && (x==2))

{

x=3;

}

else if (((PORTF.INTFLAGS == 0b01000000) == 1) && (x==3))

{

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

}

else{ numTries++;

x=0;

}

int z = PORTF.INTFLAGS;

PORTF.INTFLAGS=z;

}

***Επεξήγηση:***

Αρχικά αρχικοποιώ τις βιβλιοθήκες και τις μεταβλητές που θα χρησιμοποιήσω. Στην main με την while(1) αρχικοποιώ τα κουμπιά SW5,SW6 του PORTF και ενεργοποιώ τις διακοπές. Ακόμη αρχικοποιώ τον timer TCA0 χωρίς όμως να είναι ακόμα enabled. Μόλις πατηθεί ένα από τα δύο PINS η ροή προγράμματος μεταφέρεται στην ISR του PORTF στην οποία με την if (((PORTF.INTFLAGS == 0b00100000) == 1) && (x==0))

{

x=1;

}

συγκρίνω αν πατήθηκε το PIN5 και αν αρχικά x=0,τότε θέτω την μεταβλητή x ίση με 1.Στην συνέχεια μόλις ξαναπατηθεί κάποιο PIN,θα ελεγθεί με την τιμή 0b01000000 και αν είναι ίση δηλαδή πατήθηκε το PIN6 και η μεταβλητή x είναι ίση με 1,δηλαδή πατήθηκε το PIN5 πριν θα τεθεί x=2. Παρόμοια μόλις πατηθεί το PIN6 με την προηγούμενη σωστή ακολουθία PIN5 PIN6 PIN5 θέτω enable τον timer. Αν οποιαδήποτε στιγμή δεν είναι σωστή η ακολουθία των PINS τότε αυξάνεται η μεταβλητή numTries ,η οποία στην Main όταν έχει τιμή >=3 (δηλαδή 3 λάθος προσπάθειες εισαγωγής κωδικού) με την if συνεχίζονται οι υπόλοιπες λειτουργίες της άσκησης .Επίσης ,αν είναι λάθος η ακολουθία θέτεται x=0 έτσι ώστε η ακολουθία να ξεκινάει από την αρχή. Τέλος, στο τέλος της ISR κάθε φορά μηδενίζεται η τιμή του PORTF.

***Κώδικας για την δεύτερη συνάρτηση:***

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define ped 10

#define SW5\_PIN PIN5\_bm

#define SW6\_PIN PIN6\_bm

#define MAX\_TRIES 3

int x=0;

int numTries = 0;

int z=0;

int y=0;

int t=0;

int main() {

while (1) {

PORTF.DIRCLR = SW5\_PIN | SW6\_PIN;

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

PORTF.PIN6CTRL = PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc;

sei();

//PORTD.OUTCLR= PIN1\_bm; //LED1 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

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

//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

if ( ((numTries >= MAX\_TRIES) && (z==1) ) || (y==1) )

{

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

//prescaler=1024

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

TCA0.SINGLE.PER = 254; //select the resolution

TCA0.SINGLE.CMP0 = 127; //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()

z=0;

y=0;

t=1;

}

return 0;

}

ISR(TCA0\_CMP0\_vect){

TCA0.SINGLE.CTRLA = 0; //Disable

//clear flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

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

PORTD.OUTCLR |= PIN0\_bm; //LED0 is ON

if(z==1){

y=1;

}

}

ISR(PORTF\_PORT\_vect){

int intflags = PORTF.INTFLAGS;

if (((PORTF.INTFLAGS == 0b00100000) == 1) && (x==0))

{

x=1;

}

else if (((PORTF.INTFLAGS == 0b01000000) == 1) && (x==1))

{

x=2;

}

else if (((PORTF.INTFLAGS == 0b00100000) == 1) && (x==2))

{

x=3;

}

else if (((PORTF.INTFLAGS == 0b01000000) == 1) && (x==3))

{

x=0;

numTries=0;

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

if( z==1 ){

ADC0.CTRLA &= ~(ADC\_ENABLE\_bm);//disable ADC

PORTD.OUT |= PIN0\_bm; //LED0 is off

TCA0.SINGLE.CTRLA &= ~(TCA\_SINGLE\_ENABLE\_bm); //disable timer

x=0;

y=0;

z=0;

}

else if(t==1){

ADC0.CTRLA &= ~(ADC\_ENABLE\_bm);//disable ADC

PORTD.OUT |= PIN0\_bm; //LED0 is off

TCA0.SINGLE.CTRLA &= ~(TCA\_SINGLE\_ENABLE\_bm); //disable timer

x=0;

y=0;

z=0;

}

}

else{ numTries++;

x=0;

}

int z = PORTF.INTFLAGS;

PORTF.INTFLAGS=z;

}

ISR(ADC0\_WCOMP\_vect){

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

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

z=1;

}

ISR(TCA0\_OVF\_vect){

//clear the interrupt flag

int intflags =

TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

PORTD.OUTCLR= PIN0\_bm; //LED0 is on

}

ISR(TCA0\_CMP0\_vect){

//clear the interrupt flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

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

}

***Επεξήγηση:***

Στην δεύτερη συνάρτηση τώρα αφότου εισαχθεί ο σωστός κωδικός ενεργοποιείται ο timer και μηδενίζεται η μεταβλητή x έτσι ώστε να γίνει ξανά η σύγκριση στις if της ISR του PORTF όταν χρειαστεί. Μόλις ο timer τερματίσει ενεργοποιείται ο ADC που έχει αρχικοποιηθεί στην main και ανάβει το LED0 .Όταν ο ADC συναντήσει τιμή μικρότερη του κατωφλίου ενεργοποιείται Η ISR στη οποία ενεργοποιεί ξανά τον timer θέτει z=1 .Αναμένοντας εισαγωγή κωδικού αν η ακολουθία είναι σωστή επειδή τώρα είναι z=1 μέσω των If της ΙSR του PORTF θα απενεργοποιηθούν οι ADC,timer και LED0 και επειδή μηδενίζονται όλα τα flags το πρόγραμμα θα μπορεί να ξαναξεκινήσει .Αν η ακολουθία εισαχθεί λάθος 3 φορές δηλαδή τίθεται numtries=3 και z=1 δηλαδή έχει ενεργοποιηθεί ο ADC,timer ή έχει λήξει ο timer y==1(καθώς όταν λήξει στην ISR του επειδή z==1 θα τεθεί y=1 ) ενεργοποιείται η σειρήνα PWM στο LED0 και τίθεται z,y=0 και t=1 για να μπορούν να γίνουν οι συγκρίσεις παρόμοια τόσο στην εισαγωγή σωστής ακολουθίας όπου τώρα αν εισαχθεί σωστά μαζί την την if(t==1) θα απενεργοποιηθεί η σειρήνα όσο και στον έλεγχο λάθους ακολουθίας αναμένεται η σωστή εισαγωγή της. Τέλος, στις ISR του PWM high και Low rise ανάβει η κλείνει το LED0. Ουσιαστικά οι λειτουργίες επιτυγχάνονται με το άνοιγμα και κλείσιμο των flags για κάθε περίπτωση σε συνδυασμό με τις if και οι λάθος προσπάθειες εισαγωγής με την σύγκριση αυτών με την numtries μεταβλητή ή η απνεργεοποίηης όλων των σημάτων αφού εισαχθεί σωστά.