Εργαστήριο Προηγμένοι Μικροεπεξεργαστές

**Εργαστηριακή Άσκηση 4**

**Προσομοίωση συστήματος συναγερμού**

**Στοιχεία Ομάδας :**

**Ταμβάκης Θωμάς-Χρυσοβαλάντης, ΑΜ:1072631**

**Διαμαντόπουλος Ιωάννης , ΑΜ:1064281**

**Ομάδα : Β3**

**Ερώτημα 1.α - Σύστημα αναμονής εισαγωγής σωστού συνδυασμού καθώς και ενεργοποίηση και απενεργοποίηση του χρονιστή**

#include <avr/io.h>

#include <avr/interrupt.h>

#define time 10 //define time for timer

int x=0;

int y=0;

int cnt=0;

int test=0;

int main() {

while(1){

//pullup enable and Interrupt enabled with sense on both edges

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

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

sei(); //enable interrupts

while(cnt<4){ //until correct password

while (x==0) { //while for switch 5 and 6

;

}//end of while x==0

}//end of while cnt<4

cli(); //disable interrupts (for 5656 code)

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

//Normal Mode

TCA0.SINGLE.CTRLB = 0;

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

TCA0.SINGLE.CMP0 = time;

TCA0.SINGLE.CTRLA = 0x7<<1; //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 (y==0) { //while for timer

;

}

cli(); //disable interrupts (for timer)

}

}

ISR(PORTF\_PORT\_vect){ //ISR for switch 5 and 6

//clear all intflags except bit5 and bit6

int all;

all = PORTF.INTFLAGS;

all &= ~(0b01100000);

PORTF.INTFLAGS=all;

if ( ((PORTF.INTFLAGS & 0b00100000)==32)&&((cnt==0)||(cnt==2)) ) { //check if 5 is pressed at first or third position

if( !((PORTF.INTFLAGS & 0b01000000)==64) ){ //check if 6 has not pressed at the same time with 5

int y = PORTF.INTFLAGS; //clear flags

PORTF.INTFLAGS=y;

cnt++;

x=1;

}else{

int y = PORTF.INTFLAGS; //clear flags

PORTF.INTFLAGS=y;

cnt=0;

x=1;

}

}else if( ((PORTF.INTFLAGS & 0b01000000)==64)&&((cnt==1)||(cnt==3)) ) //check if 6 is pressed at second or fourth position

{

int y = PORTF.INTFLAGS; //clear flags

PORTF.INTFLAGS=y;

cnt++;

x=1;

}else{

int y = PORTF.INTFLAGS; //clear flags

PORTF.INTFLAGS=y;

cnt=0;

x=1;

}

}

ISR(TCA0\_CMP0\_vect){ //ISR for timer

TCA0.SINGLE.CTRLA = 0; //Disable

//clear flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

y=1;

}

**Ερώτημα 1.β - Προσθήκη λειτουργίας του ADC, του LED0, του χρονιστή (timer) και έλεγχος εισαγωγής σωστού συνδυασμού σε τρεις προσπάθειες μέσα στο χρονικό περιθώριο που ορίζει ο χρονιστής**

#include <avr/io.h>

#include <avr/interrupt.h>

#define time 10 //define time for timer

int x=0; //logic flag

int y=0;

int cnt=0;

int fail=0;

int adc = 0;

int time\_has\_passed = 0;

int main() {

PORTD.DIR |= 0b00000001; //PIN is output

while(1){

PORTD.OUT |= 0b00000001; //LED is off

//pullup enable and Interrupt enabled with sense on both edges

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

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

sei(); //enable interrupts

while(cnt<4){ //until correct password

while (x==0) { //while for switch 5 and 6

;

}//end of while x==0

}//end of while cnt<4

cli(); //disable interrupts (for 5656 code)

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

//Normal Mode

TCA0.SINGLE.CTRLB = 0;

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

TCA0.SINGLE.CMP0 = time;

TCA0.SINGLE.CTRLA = 0x7<<1; //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 (y==0) { //while for timer

;

}

cli(); //disable interrupts (for timer)

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(adc==0){ //while for ADC

;

}

cli();

cnt=0;

fail=0;

time\_has\_passed=0;

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

//Normal Mode

TCA0.SINGLE.CTRLB = 0;

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

TCA0.SINGLE.CMP0 = time;

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

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

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

sei();

while(fail<3 && time\_has\_passed==0){ //while for alarm disable

}

cli();

}

}

ISR(PORTF\_PORT\_vect){ //ISR for switch 5 and 6

//clear all intflags except bit5 and bit6

int all;

all = PORTF.INTFLAGS;

all &= ~(0b01100000);

PORTF.INTFLAGS=all;

if ( ((PORTF.INTFLAGS & 0b00100000)==32)&&((cnt==0)||(cnt==2)) ) { //check if 5 is pressed at first or third position

if( !((PORTF.INTFLAGS & 0b01000000)==64) ){ //check if 6 has not pressed at the same time with 5

int y = PORTF.INTFLAGS; //clear flags

PORTF.INTFLAGS=y;

cnt++;

x=1;

}else{

int y = PORTF.INTFLAGS; //clear flags

PORTF.INTFLAGS=y;

cnt=0;

x=1;

fail++;

}

}else if( ((PORTF.INTFLAGS & 0b01000000)==64)&&((cnt==1)||(cnt==3)) ) //check if 6 is pressed at second or fourth position

{

int y = PORTF.INTFLAGS; //clear flags

PORTF.INTFLAGS=y;

cnt++;

x=1;

}else{

int y = PORTF.INTFLAGS; //clear flags

PORTF.INTFLAGS=y;

cnt=0;

x=1;

fail++;

}

}

ISR(TCA0\_CMP0\_vect){ //ISR for timer

TCA0.SINGLE.CTRLA = 0;

//clear flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

y=1;

time\_has\_passed = 1;

PORTD.OUT |= PIN0\_bm; //PIN0 OFF

}

ISR(ADC0\_WCOMP\_vect){ //ISR for ADC

int intflags = ADC0.INTFLAGS; //clear flags

ADC0.INTFLAGS = intflags;

adc=1;

}

**Ερώτημα 1.γ - Υλοποίηση σειρήνας, ορισμός ενός PWM παλμού για την ενεργοποίηση και απενεργοποίηση του LED**

#include <avr/io.h>

#include <avr/interrupt.h>

#define time 10 //define time for timer

int x=0;

int y=0;

int cnt=0;

int test=0;

int timer\_for\_alarm\_off=0;

int fail=0;

int adc = 0;

int time\_has\_passed = 0;

int check\_alarm = 0;

int main() {

PORTD.DIR |= 0b00000001; //PIN is output

while(1){

PORTD.OUT |= 0b00000001; //LED is off

//pullup enable and Interrupt enabled with sense on both edges

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

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

sei(); //enable interrupts

while(cnt<4){ //until correct password

while (x==0) { //while for switch 5 and 6

;

}//end of while x==0

}//end of while cnt<4

cli(); //disable interrupts (for 5656 code)

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

//Normal Mode

TCA0.SINGLE.CTRLB = 0;

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

TCA0.SINGLE.CMP0 = time;

TCA0.SINGLE.CTRLA = 0x7<<1; //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 (y==0) { //while for timer

;

}

cli(); //disable interrupts (for timer)

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(adc==0){ //while for ADC

;

}

cli();

cnt=0;

fail=0;

time\_has\_passed=0;

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

//Normal Mode

TCA0.SINGLE.CTRLB = 0;

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

TCA0.SINGLE.CMP0 = time;

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

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

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

sei();

while(fail<3 && time\_has\_passed==0){ //while for alarm disable

}

if(fail==3 || time\_has\_passed==1){ //check if attempts for disable failed

check\_alarm = 1;

cnt=0;

TCA0.SPLIT.CTRLA = 0b00000000; //disable timer

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

TCA0.SINGLE.PER = 64; //select the resolution

TCA0.SINGLE.CMP0 = 31; //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 CMP0

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

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

while (cnt<4){ //until correct password

}

check\_alarm=0;

PORTD\_OUT |= PIN0\_bm; //LED0 off

}

}

}

ISR(PORTF\_PORT\_vect){ //ISR for switch 5 and 6

//clear all intflags except bit5 and bit6

int all;

all = PORTF.INTFLAGS;

all &= ~(0b01100000);

PORTF.INTFLAGS=all;

if ( ((PORTF.INTFLAGS & 0b00100000)==32)&&((cnt==0)||(cnt==2)) ) { //check if 5 is pressed at first or third position

if( !((PORTF.INTFLAGS & 0b01000000)==64) ){ //check if 6 has not pressed at the same time with 5

int y = PORTF.INTFLAGS; //clear flags

PORTF.INTFLAGS=y;

cnt++;

x=1;

}else{

int y = PORTF.INTFLAGS; //clear flags

PORTF.INTFLAGS=y;

cnt=0;

x=1;

fail++;

}

}else if( ((PORTF.INTFLAGS & 0b01000000)==64)&&((cnt==1)||(cnt==3)) ) //check if 6 is pressed at second or fourth position

{

int y = PORTF.INTFLAGS; //clear flags

PORTF.INTFLAGS=y;

cnt++;

x=1;

}else{

int y = PORTF.INTFLAGS; //clear flags

PORTF.INTFLAGS=y;

cnt=0;

x=1;

fail++;

}

}

ISR(TCA0\_CMP0\_vect){ //ISR for timer

if(check\_alarm ==0){TCA0.SINGLE.CTRLA = 0;} //Disable if alarm in not enable

//clear flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS=intflags;

y=1;

time\_has\_passed = 1;

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

}

ISR(ADC0\_WCOMP\_vect){ //ISR for ADC

int intflags = ADC0.INTFLAGS; //clear flag

ADC0.INTFLAGS = intflags;

adc=1;

}

ISR(TCA0\_OVF\_vect){ //ISR for PWM

//clear the interrupt flag

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

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

}

**Ερώτημα 2.α – Επεξήγηση κώδικα και διάγραμμα ροής**

Σε ότι έχει να κάνει με τον κώδικα του πρώτου ερωτήματος, αρχικά, ορίζουμε τρεις μεταβλητές, την *x* την *y* και τη *cnt*. Την πρώτη την αξιοποιούμε για το πάτημα των κουμπιών 5 και 6, δηλαδή μέχρι να πατηθεί κάποιο από τα κουμπιά 5 και 6, το πρόγραμμα παραμένει στη *while (x==0)*. Τη δεύτερη, την χρησιμοποιούμε για τον Timer, όταν ενεργοποιηθεί ο συναγερμός, όταν δηλαδή παρέλθει ο χρόνος που έχουμε ορίσει, το πρόγραμμα εξέρχεται της *while (y==0)*. Την τρίτη μεταβλητή, την αξιοποιούμε για να μετράμε τα σωστά πατήματα των κουμπιών 5 και 6. Για να ενεργοποιηθεί ο συναγερμός, απαιτείται ο συνδυασμός 5656. Αν πατηθεί κάποιο κουμπί λάθος, για παράδειγμα πρώτα το 6 ή πρώτα το 5 και μετά πάλι το 5 (δηλαδή η ακολουθία είναι η 55), η μεταβλητή *cnt* μηδενίζεται, με σκοπό να γραφτεί η σωστή ακολουθία (5656).

Στην main(), ορίζουμε τη *while(1)*, με σκοπό το πρόγραμμά μας να μην σταματήσει να τρέχει. Στη συνέχεια με τις εντολές *PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc;* και *PORTF.PIN6CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc;* ορίζουμε ότι το πρόγραμμα, θα πάει στη *ISR(PORTF\_PORT\_vect)* όταν ενεργοποιηθούν οι διακοπές (sei()), μονάχα στο πάτημα των κουμπιών 5 και 6. Στη συνέχεια ενεργοποιούμε τις διακοπές με την εντολή *sei()* και το πρόγραμμα εισέρχεται στη *while(cnt<4)* και στην *while (x==0)*, και περιμένει από το χρήστη να πληκτρολογήσει τον κωδικό 5656. Ανάλογα με το πάτημα κάποιου κουμπιού, το πρόγραμμα μεταφέρεται στην *ISR(PORTF\_PORT\_vect)*, όπου σε κάθε περίπτωση εκτελούνται οι εντολές

int all;

all = PORTF.INTFLAGS;

all &= ~(0b01100000);

PORTF.INTFLAGS=all;

Με τη χρήση μάσκας, μας ενδιαφέρουν μονάχα τα κουμπιά 5 και 6. Υπάρχουν 2 περιπτώσεις ανάλογα με την ακολουθία των αριθμών :

* Εισαγωγή σωστού κωδικού: Έστω ότι ο χρήστης πληκτρολογεί πρώτα το κουμπί 5, το πρόγραμμα εισέρχεται στην *if ( ((PORTF.INTFLAGS & 0b00100000)==32)&&((cnt==0)||(cnt==2)) )* καθώς έχει πατηθεί μονάχα το κουμπί 5 και το *cnt* ισούται με 0 (δηλαδή είναι το πρώτο ψηφίο του κωδικού). Στη συνέχεια εισέρχεται στην if( !((PORTF.INTFLAGS & 0b01000000)==64) καθώς δεν έχουν πατηθεί ταυτόχρονα και το 5ο και το 6ο κουμπί, αλλά μόνο το 5. Εκεί, μηδενίζουμε όλα τα flags με τις εντολές *int y = PORTF.INTFLAGS;* και *PORTF.INTFLAGS=y;* και αυξάνουμε κατά 1 τη μεταβλητή *cnt*, με την εντολή *cnt++*. Στη συνέχεια, έστω ότι ο χρήστης πληκτρολογεί το κουμπί 6. το πρόγραμμα εισέρχεται *else if( ((PORTF.INTFLAGS & 0b01000000)==64)&&((cnt==1)||(cnt==3)) )* καθώς έχει πατηθεί μονάχα το κουμπί 6 και το *cnt* ισούται με 1 (δηλαδή είναι το δεύτερο ψηφίο του κωδικού). Εκεί μηδενίζουμε όλα τα flags με τις εντολές *int y = PORTF.INTFLAGS;* και *PORTF.INTFLAGS=y,* αυξάνουμε κατά 1 τη μεταβλητή *cnt*, με την εντολή *cnt++* και θέτουμε τη μεταβλητή *x* ίση με 1 ώστε να εξέλθει της *while(x==0)*. Αντίστοιχη θα είναι η ακολουθία των εντολών εάν πατήσουμε ξανά διαδοχικά τα κουμπιά 5 και 6. Εν τέλει θα έχουμε την σωστή ακολουθία κωδικού (5656) και η μεταβλητή *cnt* ισούται με 4. Το πρόγραμμα, αφού εξέλθει της ISR, θα εξέλθει και της *while(cnt<4)* και θα απενεργοποιηθούν οι διακοπές με την εντολή *cli().*
* Εισαγωγή λάθος κωδικού:

1. Έστω ότι ο χρήστης πληκτρολογεί πρώτα το κουμπί 5, το πρόγραμμα εισέρχεται στην *if ( ((PORTF.INTFLAGS & 0b00100000)==32)&&((cnt==0)||(cnt==2)) )* καθώς έχει πατηθεί μονάχα το κουμπί 5 και το *cnt* ισούται με 0 (δηλαδή είναι το πρώτο ψηφίο του κωδικού). Στη συνέχεια εισέρχεται στην if( !((PORTF.INTFLAGS & 0b01000000)==64) καθώς δεν έχουν πατηθεί ταυτόχρονα και το 5ο και το 6ο κουμπί, αλλά μόνο το 5. Εκεί, μηδενίζουμε όλα τα flags με τις εντολές *int y = PORTF.INTFLAGS;* και *PORTF.INTFLAGS=y;* και αυξάνουμε κατά 1 τη μεταβλητή *cnt*, με την εντολή *cnt++*. Στη συνέχεια, έστω ότι ο χρήστης πληκτρολογεί ξανά το κουμπί 5. το πρόγραμμα δεν εισέρχεται ούτε στη *if( !((PORTF.INTFLAGS & 0b01000000)==64) )* ούτε στην *else if( ((PORTF.INTFLAGS & 0b01000000)==64)&&((cnt==1)||(cnt==3)) )* για να είναι σωστή η εισαγωγή του κωδικού θα έπρεπε να πατηθεί το κουμπί 6 και η μεταβλητή *cnt* να ισούται με 1. Σε αυτή την περίπτωση τo *cnt* ισούται με 1, όμως πατήθηκε το κουμπί 5. Έτσι, το πρόγραμμα εισέρχεται στην else της μεγάλης if, μηδενίζονται τα flags με τις εντολές *int y = PORTF.INTFLAGS;* και *PORTF.INTFLAGS=y,* μηδενίζουμε τη μεταβλητή *cnt*, με την εντολή *cnt=0* και θέτουμε τη μεταβλητή *x* ίση με 1 ώστε να εξέλθει της *while(x==0)*.
2. Έστω ότι ο χρήστης πληκτρολογεί ταυτόχρονα το κουμπί 5 και το κουμπί 6, το πρόγραμμα εισέρχεται στην *if ( ((PORTF.INTFLAGS & 0b00100000)==32)&&((cnt==0)||(cnt==2)) )* καθώς έχει πατηθεί το κουμπί 5 και το *cnt* ισούται με 0 (δηλαδή είναι το πρώτο ψηφίο του κωδικού). Στη συνέχεια δεν εισέρχεται στην if( !((PORTF.INTFLAGS & 0b01000000)==64) καθώς έχουν πατηθεί ταυτόχρονα και το 5ο και το 6ο κουμπί. Εκεί, μηδενίζουμε όλα τα flags με τις εντολές *int y = PORTF.INTFLAGS;* και *PORTF.INTFLAGS=y;* και μηδενίζουμε τη μεταβλητή *cnt*, με την εντολή *cnt=0* και θέτουμε τη μεταβλητή *x* ίση με 1 ώστε να εξέλθει της *while(x==0)*.

Στην πρώτη περίπτωση (σωστή εισαγωγή κωδικού), αφού το πρόγραμμα εξέλθει της *while(cnt<4)* και απενεργοποιηθούν οι διακοπές ορίζουμε timer. Αρχικά καθαρίζουμε τον counter με την εντολή *TCA0.SINGLE.CNT = 0* και ορίζουμε την κανονική λειτουργία με την εντολή *TCA0.SINGLE.CTRLB = 0*. Επίσης, ορίζουμε τη διάρκεια του με την εντολή *TCA0.SINGLE.CMP0 = time* με time, τιμή ορισμένη από εμάς. Ακόμη ορίζουμε τη συχνότητα και ενεργοποιούμε τον μετρητή με τις εντολές *TCA0.SINGLE.CTRLA = 0x7<<1* και *TCA0.SINGLE.CTRLA |=1* αντίστοιχα. Τέλος, αφού ενεργοποιηθούν οι διακοπές και το πρόγραμμα εισαχθεί στην *while(y==0)*, o timer αρχίζει να μετράει. Όταν γίνει η περάτωση του χρόνου, το πρόγραμμα εισέρχεται στην *ISR(TCA0\_CMP0\_vect)*, όπου απενεργοποιείται ο timer με την εντολή *TCA0.SINGLE.CTRLA = 0* και μηδενίζονται τα flags με τις εντολές *int intflags = TCA0.SINGLE.INTFLAGS* και *ΤCA0.SINGLE.INTFLAGS=intflags*. Κλείνοντας, η μεταβλητή y γίνεται 1, ώστε όταν το πρόγραμμα εξέλθει της *ISR(TCA0\_CMP0\_vect)*, να εξέλθει και της *while(y==0)* και να απενεργοποιηθούν οι διακοπές.

Ακολουθεί το διάγραμμα ροής:



**Ερώτημα 2.β – Επεξήγηση κώδικα και διάγραμμα ροής**

Στο ερώτημα αυτό διατηρήθηκαν οι λειτουργίες του πρώτου ερωτήματος και προστέθηκε ο ADC, ο timer, το LED0 και ο έλεγχος εισαγωγής σωστού συνδυασμού σε τρεις προσπάθειες μέσα στο χρονικό περιθώριο που ορίζει ο timer. Ειδικότερα, προστέθηκαν οι μεταβλητές *fail*, adc, time\_has\_passed. Η πρώτη μετράει των αριθμό των λανθασμένων κωδικών, η δεύτερη αξιοποιείται για τον ADC, και γίνεται 1 στην *ISR(ADC0\_WCOMP\_vect)* με σκοπό το πρόγραμμα να εξέλθει της *while(adc==0)*, και η τρίτη γίνεται 1 όταν παρέλθει ο χρόνος που διαθέτει ο χρήστης για να βάλει τον σωστό κωδικό.

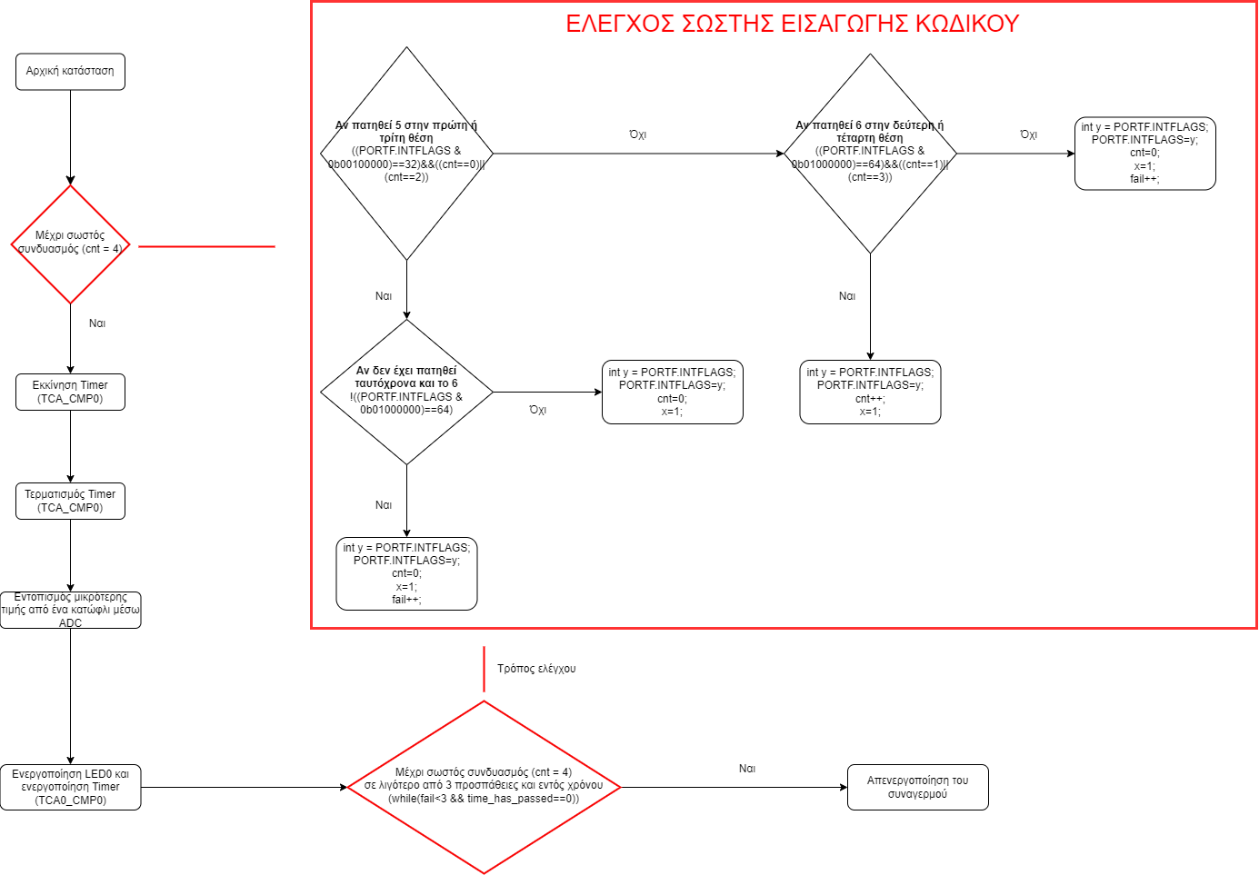
Επιπρόσθετα, στην main(), με την εντολή *PORTD.DIR |= 0b00000001*, ορίζουμε ότι θα χρησιμοποιήσουμε το PIN0 και στην αρχή της *while(1)*, με την εντολή *PORTD.OUT |= 0b00000001*, το απενεργοποιούμε. Στη συνέχεια γίνεται ότι ακριβώς αναφέρθηκε και αναλύθηκε στο πρώτο ερώτημα, είτε με τη σωστή, είτε με λανθασμένη εισαγωγή του κωδικού. Στην περίπτωση που εισαχθεί σωστά ο κωδικός, και τρέξει ο timer ο συναγερμός έχει ενεργοποιηθεί. Ακόμη, ορίζουμε τον ADC με τις κάτωθι εντολές :

* Με την εντολή *ADC0.CTRLA |= ADC\_RESSEL\_10BIT\_gc* ορίζουμε ότι το Resolution θα είναι 10 bit για τη μετατροπή της αναλογικής τιμής σε ψηφιακή.
* Με την εντολή *ADC0.CTRLA |= ADC\_FREERUN\_bm* ο ADC θα λειτουργήσει σε Free-Running Mode (στον ADC.CTRLA τίθεται ‘1’ στο Free-Running bit).
* Με την εντολή *ADC0.CTRLA |= ADC\_ENABLE\_bm* ενεργοποιείται ο ADC.
* Με την εντολή *ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc* ορίζεται το bit με το οποίο συνδέεται ο ADC.
* Με την εντολή *ADC0.WINLT |= 10* θέτουμε το κατώφλι στον καταχωρητή *ADC0.WINLT.*
* Με την εντολή *ADC0.CTRLE |= ADC\_WINCM0\_bm* ορίζουμε ότι θέλουμε να γίνεται interrupt όταν RESULT < WINLT.

Εφόσον ενεργοποιήσαμε τον ADC, ενεργοποιούμε τις διακοπές με την εντολή *sei()* και στη συνέχεια με την εντολή *ADC0.COMMAND |= ADC\_STCONV\_bm* ξεκινάει το conversion. To πρόγραμμά μας εισέρχεται στη *while (adc==0)*, και παραμένει εκεί έως ότου ο αισθητήρας του συναγερμού αντιληφθεί κάποια κίνηση στο χώρο, δηλαδή έως ότου η μεταβλητή *adc* γίνει 1. Όταν ο αισθητήρας αντιληφθεί κάποια κίνηση στο χώρο, δηλαδή όταν η τιμή του RESULT γίνει μικρότερη του WINLT, το πρόγραμμα εισέρχεται στην *ISR(ADC0\_WCOMP\_vect)*, στην οποία καθαρίζουμε τα INTFLAGS με τις εντολές *intflags = ADC0.INTFLAGS* και *ADC0.INTFLAGS = intflags* και τίθεται η μεταβλητή adc, στην τιμή 1. Όταν η τιμή του RESULT γίνει μεγαλύτερη της WINLT, το πρόγραμμα εξέρχεται της ISR και αφού έχει τεθεί η μεταβλητή *adc* στην τιμή 1, εξέρχεται και της *while(adc==0)*. Συμπληρωματικά, απενεργοποιούνται οι διακοπές με την εντολή *cli()*, αρχικοποιούμε ξανά και ενεργοποιούμε τον timer. Σε αυτό το χρονικό διάστημα, θα πρέπει ο χρήστης να εισάγει σωστά τον κωδικό. Το πρόγραμμα του δίνει την δυνατότητα να κάνει τρεις φορές λάθος τον κωδικό. Όταν ενεργοποιηθούν οι διακοπές με την εντολή *sei()* ξεκινάει ο timer και δίνεται η δυνατότητα στο χρήστη να εισάγει σωστά στον κωδικό. Έτσι το πρόγραμμα εισέρχεται στην *while(fail<3 && time\_has\_passed==0)* και παραμένει, έως ότου είτε ο χρήστης κάνει τρεις φορές λάθος τον κωδικό είτε μέχρι να παρέλθει ο χρόνος που έχει οριστεί. Όταν ο χρήστης πατάει τα κουμπιά για να εισάγει τον κωδικό εισέρχεται στην *ISR(PORTF\_PORT\_vect)* όπως και στο πρώτο ερώτημα, με τη μόνη διαφορά, ότι σε αυτό το ερώτημα σε κάθε λάθος συνδυασμό (πχ 55, 6, 5655) η μεταβλητή *fail* αυξάνεται κατά ένα. Έτσι έχουμε τις ακόλουθες τρεις περιπτώσεις:

* Είτε θα εισαχθεί σωστά ο κωδικός μέσα στο χρονικό διάστημα που έχουμε θέσει
* Είτε θα παρέλθει ο χρόνος πριν εισάγει κωδικό ο χρήστης (στο επόμενο ερώτημα θα ενεργοποιηθεί ο συναγερμός και θα αναβοσβήνει το LED0).
* Είτε πριν παρέλθει ο χρόνος, ο χρήστης κάνει τρεις φορές λάθος τον κωδικό (στο επόμενο ερώτημα θα ενεργοποιηθεί ο συναγερμός και θα αναβοσβήνει το LED0).

Ακολουθεί το διάγραμμα ροής:



**Ερώτημα 2.γ – Επεξήγηση κώδικα και διάγραμμα ροής**

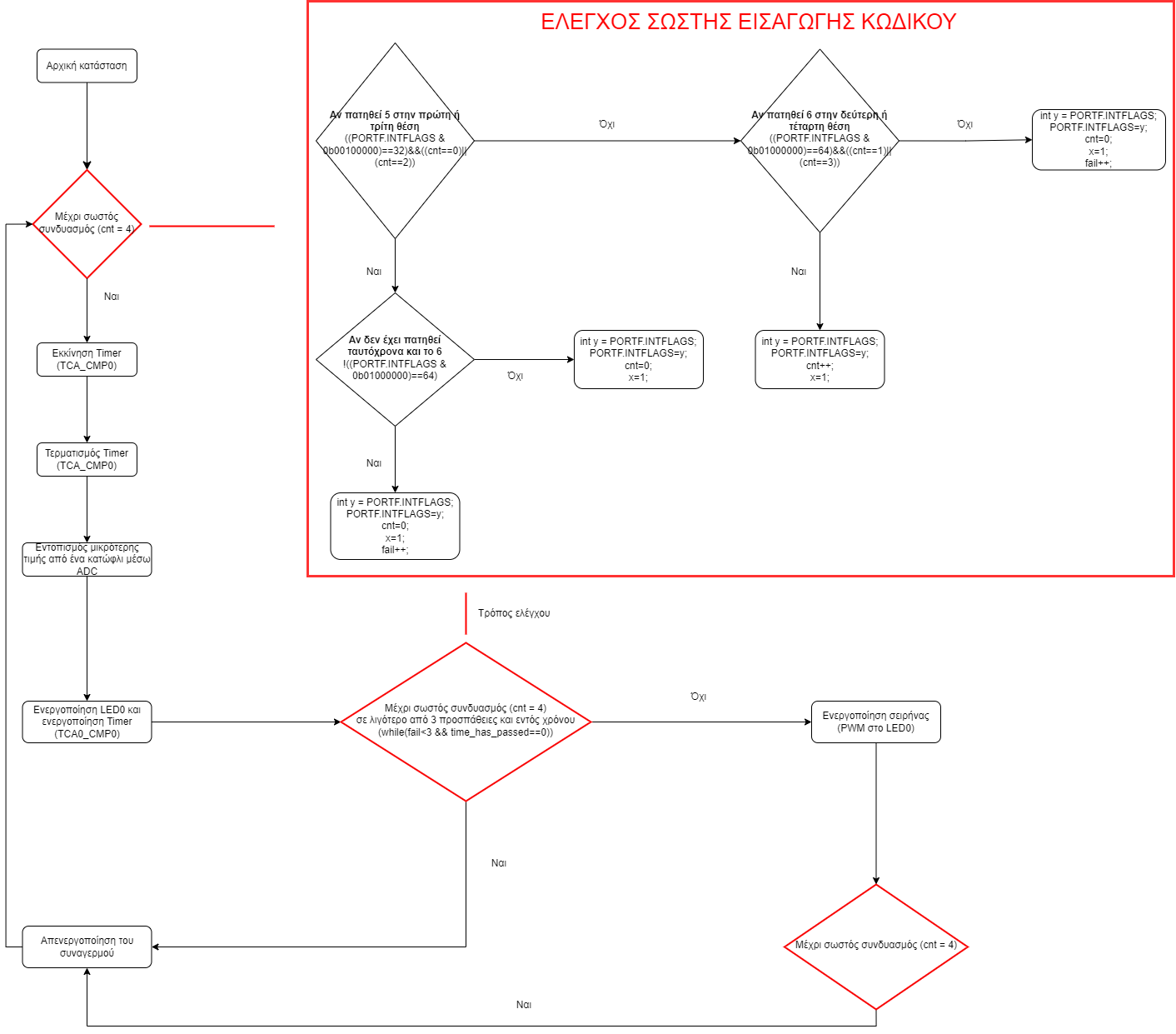
Στο ερώτημα αυτό διατηρήθηκαν οι λειτουργίες του δεύτερου ερωτήματος και προστέθηκε ο PWM παλμός για την ενεργοποίηση και απενεργοποίηση του LED0 (σειρήνα). Ειδικότερα προστέθηκε η μεταβλητή check\_alarm η οποία γίνεται 1, όταν ενεργοποιηθεί ο συναγερμός, δηλαδή είτε όταν παρέλθει ο χρόνος για την εισαγωγή του κωδικού, είτε όταν ο χρήστης κάνει τρεις λάθος προσπάθειες εισαγωγής κωδικού προτού παρέλθει ο χρόνος. Στη συνέχεια γίνεται ότι ακριβώς αναφέρθηκε και αναλύθηκε στο πρώτο και το δεύτερο ερώτημα, είτε με τη σωστή, είτε με λανθασμένη εισαγωγή του κωδικού. Στην περίπτωση που εισαχθεί σωστά ο κωδικός, και τρέξει ο timer ο συναγερμός έχει ενεργοποιηθεί.

Εάν εντοπιστεί ένα αντικείμενο στον χώρο ενεργοποιείται ο ADC. Εάν ο χρήστης πληκτρολογήσει τρεις φορές λάθος κωδικό, ή παρέλθει ο χρόνος που του δίνεται για να τον πληκτρολογήσει, ενεργοποιείται η σειρήνα. Επομένως το πρόγραμμα εισέρχεται στην *if(fail==3 ||time\_has\_passed==1)* καθώς έχει γίνει ένα από τα δύο προαναφερθέντα γεγονότα. Στη συνέχεια, θέτουμε την μεταβλητή *check\_alarm* ίση με 1, καθώς έχει ενεργοποιηθεί ο συναγερμός, ενώ παράλληλα θέτουμε τη μεταβλητή *cnt* ίση με 0, ώστε να δοθεί η δυνατότητα στο χρήστη να εισάγει τον κωδικό και να απενεργοποιηθεί η σειρήνα. Ακόμη, ορίζεται ο παλμός PWM που θα χρησιμοποιήσουμε για τη σειρήνα. Ειδικότερα:

* Με την εντολή *TCA0.SINGLE.CTRLA=TCA\_SINGLE\_CLKSEL\_DIV1024\_gc* θέτουμε το ρολόι, με prescaler ίσο με 1024.
* Με τις εντολές *TCA0.SINGLE.PER = 64* και *TCA0.SINGLE.CMP0 = 31* θέτουμε τιμές για τα PER, CMP0. Ειδικότερα, η PER παίρνει τιμή που αφορά το χρόνο του εκάστοτε παλμού (περίοδος), ενώ η CMP0 παίρνει τιμή που αφορά τον κύκλο λειτουργίας (duty cycle). Παρατηρούμε ότι το duty cycle είναι 50%, δηλαδή η τιμή PER ισούται με 64 ενώ η τιμή CMP0 ισούται με 31.
* Με την εντολή *TCA0.SINGLE.CTRLB = TCA\_SINGLE\_WGMODE\_SINGLESLOPE\_gc* ορίζουμε ότι θα έχουμε λειτουργία PWM, δηλαδή όταν η τιμή φτάσει στην CMP0 δεν θα μηδενίσει, θα συνεχίσει μέχρι να πάει στην PER.
* Με την εντολή *TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_OVF\_bm;* ορίζουμε το interrupt overflow.
* Με την εντολή *TCA0.SINGLE.INTCTRL |= TCA\_SINGLE\_CMP0\_bm* ορίζουμε τον interrupt compare
* Με την εντολή *TCA0.SINGLE.INTCTRL |= TCA\_SINGLE\_CMP0\_bm* ενεργοποιώ το άνωθεν interrupt.

Εν συνεχεία το πρόγραμμα εισέρχεται στην *ISR(TCA0\_OVF\_vect)*, που αφορά το overflow. Εκεί μηδενίζονται τα intflags και το LED0 ανάβει (ενεργοποιείται). Στη συνέχεια το πρόγραμμα εισέρχεται στην *ISR(TCA0\_CMP0\_vect)* που αφορά τον timer. Εκεί, επειδή το check\_alarm έχει γίνει 1, δεν απενεργοποιείται ο timer, ενώ το PIN0 σβήνει (απενεργοποιείται). Αυτή η εναλλαγή μεταξύ των δύο προαναφερθέντων ISR και τα εναλλάξ LED0 = ON και LED0 = OFF (σειρήνα) γίνεται έως ότου ο χρήστης πληκτρολογήσει σωστά τον κωδικό. Εάν το πράξει, το πρόγραμμα εξέρχεται της *while (cnt<4)*, η μεταβλητή *check\_alarm* γίνεται 0 και το LED0 σβήνει (απενεργοποιείται).

Τέλος το πρόγραμμα ξεκινάει πάλι από την αρχή, δηλαδή βρίσκεται σε αναμονή του κωδικού για την ενεργοποίηση του συναγερμού.

Ακολουθεί το διάγραμμα ροής: