**Εργαστήριο Προηγμένων Μικροεπεξεργαστών**

ΟΜΑΔΑ Β4 - ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 3

Μιχαηλίδης Νεκτάριος – 1072594 – 4ο έτος

Περικλής Κύρκος – 1067524 – 5ο έτος

Παραδοχές :

Γράψαμε τον κώδικα χρησιμοποιόντας δυο TCB timers (τον TCB0 και τον TCB1).

Ο τρόπος που αναγνωρίζει το πρόγραμμα ποιος παλμός εκανε trigger το Interrupt είναι κοιτόντας τα flags TCB1 μέσα στο interrupt του TCB0. Ο TCB0 που είναι και ο πιο γρηγορος θα πετάει interrupt και το led0 θα κάνει toggle και μετα θα ελέγχουμε τα INTFLAGS του TCB1 μεσα στο Interrupt του TCB0 και αν είναι 1 τότε ξέρουμε ότι έκανε interrupt και ο TCB1 οπότε θα κάνουμε toggle και το LED1.

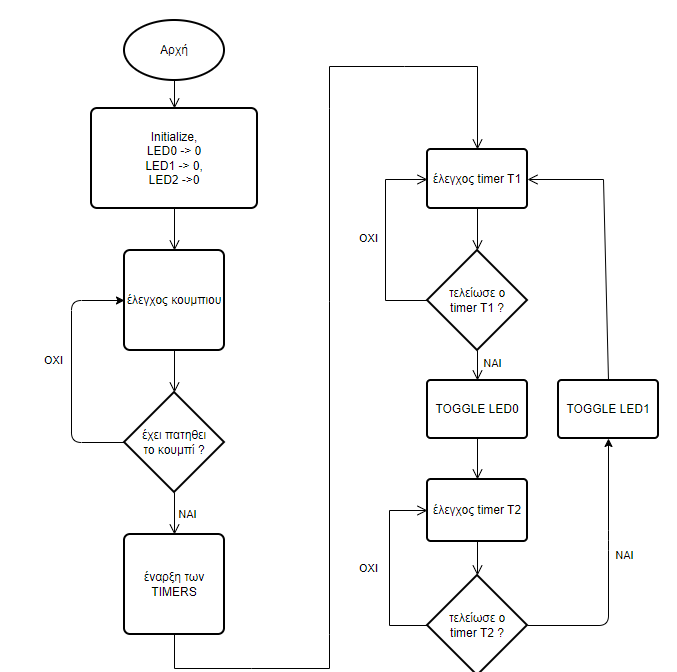
Χρησιμοποιούμε την μεταβλητή FINGER για να διαφοροποιήσουμε την κατάσταση επανέναρξης της συσκευης με το πάτημα του κουμπιού από την δεύτερη λειτουργία της συσκευης με το πάτημα του κουμπιού.

Χρησιμοποιούμε την εντολή button\_state για να ξέρει το πρόγραμμα πόσες φορες έχει πατηθει το κουμπι. 1 αντιστοιχεί στην πρώτη κανονική κατάσταση, 2 αντιστοιχεί στην πιο γρηγορη κατάσταση και 3 αντιστοιχει στην σβηστη κατάσταση. Μέσα στον main κώδικα ελέγχουμε αν η κατάσταση είναι 3 ή μεγαλύτερη και τερματίζουμε το πρόγραμμα. (το προγραμμα τρεχει while (button\_state < 3)).

Ο τρόπος που κάναμε initialize to PWD timer είναι βάλαμε στο ccmpl την τιμη των milisecond και στην ccmph την μιση από την τιμη των milisecond και έτσι πετυχένουμε περίοδο όσα milisecond βάζουμε και κύκλο λειτουργίας 50%. Οι τιμες είναι hardcoded στην αρχη του προγράμματος με #define.

Ο τρόπος που επιτύχαμε την διπλασίαση της ταχύτητας του ανεμιστήρα στο δευτερο ερώτημα είναι με μια συνάρτηση όπου ξανακανει initialize και τους δυο timers αλλα με την διπλάσια τιμή στον TCB1 που είναι ξανα hardcoded με #define (δεν χρειαζοταν να ξανακανουμε τα initialization αλλα πιστευω είναι πιο ξεκαθαρος ο κωδικας ετσι).

1. Διάγραμμα ροής πρώτου ερωτήματος :



Κώδικας Πρώτου ερωτήματος :

/\*

 \* GccApplication1.c

 \*

 \* Author : nektar24

 \*/

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#define Tb 20;

#define Tl 10;

#define Db 10;

#define Dl 5;

void start\_timer\_t1\_and\_t2();

void LED0\_on();

void LED0\_off();

void LED1\_on();

void LED1\_off();

int LED0 = 0;

int LED1 = 0;

int main() {

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

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

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

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

    // Listen for interrupt on bit 5

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

    // Enable Interrupt

    sei();

    while (1)

    {

        ;

    }

    // Disable Interrupt

    cli();

}

// Button Press INTERRUPT

ISR(PORTF\_PORT\_vect){

    start\_timer\_t1\_and\_t2();

    // Reset Interrupt Flag

    int intflags = PORTF.INTFLAGS;

    PORTF.INTFLAGS = intflags;

}

ISR(TCB1\_INT\_vect){

    if (TCB0.INTFLAGS == 1 ){

        // Timer Βάσης

        // TOGGLE LED 1

        switch (LED1){

            case 0:

                LED1 = 1;

                LED1\_on();

            break;

            case 1:

                LED1 = 0;

                LED1\_off();

            break;

        }

    }

    // Timer λεπίδων

    // TOGGLE LED 0

    switch (LED0){

        case 0:

            LED0 = 1;

            LED0\_on();

        break;

        case 1:

            LED0 = 0;

            LED0\_off();

        break;

    }

    // Reset Interrupt Flag

    int intflags = TCB1.INTFLAGS;

    TCB1.INTFLAGS = intflags;

}

void start\_timer\_t1\_and\_t2(){

    // Set TCB0 to compare/capture mode with waveform generator

    TCB0.CCMPL = Tb;

    TCB0.CTRLA |= TCB\_ENABLE\_bm;

    TCB0.CTRLB |= TCB\_CCMPEN\_bm;

    TCB0.CTRLB |= TCB\_CNTMODE\_PWM8\_gc;

    // Set TCB1 to compare/capture mode with waveform generator

    TCB1.CCMPL = Tl;

    TCB1.CTRLA |= TCB\_ENABLE\_bm;

    TCB1.CTRLB |= TCB\_CCMPEN\_bm;

    TCB1.CTRLB |= TCB\_CNTMODE\_PWM8\_gc;

    // Set TCB0 and TCB1 period

    TCB1.CCMPH = Dl;

    TCB0.CCMPH = Db;

}

// TURN ON LED 0 - κίνηση λεπίδων

void LED0\_on(){

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

}

// TURN ON LED 0 - κίνηση λεπίδων

void LED0\_off(){

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

}

// TURN ON LED 1 - κίνηση βάσης

void LED1\_on(){

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

}

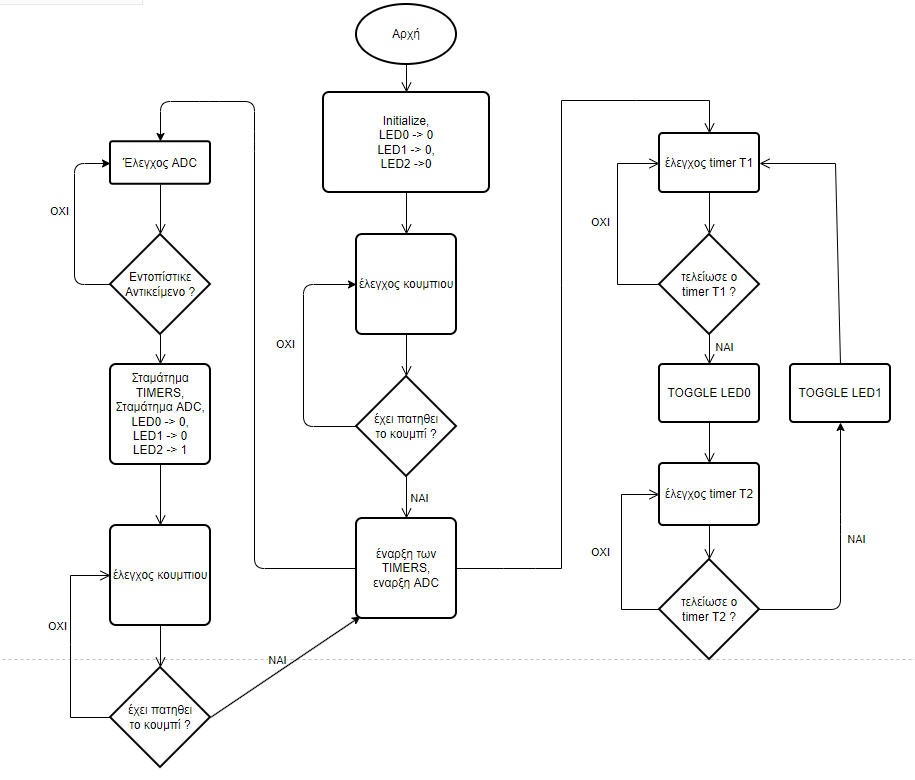
// TURN ON LED 1 - κίνηση βάσης

void LED1\_off(){

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

}

1. Διάγραμμα ροής δεύτερου ερωτήματος :



Κώδικας Δεύτερου Ερωτηματος :

/\*

 \* GccApplication1.c

 \*

 \* Author : nektar24

 \*/

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#define Tb 20;

#define Tl 10;

#define Db 10;

#define Dl 5;

void start\_timer\_t1\_and\_t2();

void turn\_off\_TIMERS();

void LED0\_on();

void LED0\_off();

void LED1\_on();

void LED1\_off();

void LED2\_on();

void LED2\_off();

bool FINGER = false;

int button\_state = 0;

int LED0 = 0;

int LED1 = 0;

int main() {

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

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

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

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

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

    PORTD.OUT |= PIN2\_bm; //LED2 is off

    //iniitalize the ADC for value LT = 10

    ADC0.CTRLA |= ADC\_RESSEL\_10BIT\_gc;

    ADC0.CTRLA |= ADC\_FREERUN\_bm;

    ADC0.CTRLA |= ADC\_ENABLE\_bm;

    ADC0.MYXPOS |= ADC\_MYXPOS\_AIN7\_gc;

    ADC0.DBGCTRL |= ADC\_DBGRUN\_bg;

    ADC0.WINLT |= 10;

    ADC0.INTCTRL |= ADC\_WCWP\_bm;

    ADC0.CTRLE |= ADC\_WINCM0\_bm;

    // ENABLING ADC

    ADC0.COMMAND |= ADC\_STCONV\_bm;

    // Listen for interrupt on bit 5

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

    // Enable Interrupt

    sei();

    while (1)

    {

        ;

    }

    // Disable Interrupt

    cli();

}

// Δάχτυλο στον Ανεμυστήρα

ISR(ADC0\_WCOMP\_vect){

    FINGER = true;

    turn\_off\_TIMERS();

    LED0\_off();

    LED1\_off();

    LED2\_on();

    // Reset Interrupt Flag

    int intflags = ADC0.INTFLAGS;

    ADC0.INTFLAGS = intflags;

}

// Button Press INTERRUPT

ISR(PORTF\_PORT\_vect){

    switch (button\_state){

        case 0:

            // Πάτημα κουμπιού για πρώτη φορά

            start\_timer\_t1\_and\_t2();

            button\_state = 1;

            break;

        case 1:

            if (FINGER){

                // Πάτημα κουμπιού για επανέναρξη μετα απο emergency shutdown

                start\_timer\_t1\_and\_t2();

                FINGER = false;

                LED2\_off();

            }

            break;

    }

    // Reset Interrupt Flag

    int intflags = PORTF.INTFLAGS;

    PORTF.INTFLAGS = intflags;

}

ISR(TCB1\_INT\_vect){

    if (TCB0.INTFLAGS == 1 ){

        // Timer Βάσης

        // TOGGLE LED 1

        switch (LED1){

            case 0:

                LED1 = 1;

                LED1\_on();

            break;

            case 1:

                LED1 = 0;

                LED1\_off();

            break;

        }

    }

    // Timer λεπίδων

    // TOGGLE LED 0

    switch (LED0){

        case 0:

            LED0 = 1;

            LED0\_on();

        break;

        case 1:

            LED0 = 0;

            LED0\_off();

        break;

    }

    // Reset Interrupt Flag

    int intflags = TCB1.INTFLAGS;

    TCB1.INTFLAGS = intflags;

}

void start\_timer\_t1\_and\_t2(){

    // Set TCB0 to compare/capture mode with waveform generator

    TCB0.CCMPL = Tb;

    TCB0.CTRLA |= TCB\_ENABLE\_bm;

    TCB0.CTRLB |= TCB\_CCMPEN\_bm;

    TCB0.CTRLB |= TCB\_CNTMODE\_PWM8\_gc;

    // Set TCB1 to compare/capture mode with waveform generator

    TCB1.CCMPL = Tl;

    TCB1.CTRLA |= TCB\_ENABLE\_bm;

    TCB1.CTRLB |= TCB\_CCMPEN\_bm;

    TCB1.CTRLB |= TCB\_CNTMODE\_PWM8\_gc;

    // Set TCB0 and TCB1 period

    TCB1.CCMPH = Dl;

    TCB0.CCMPH = Db;

}

void turn\_off\_TIMERS(){

    TCB0.CTRLA &= ~TCB\_ENABLE\_bm;

    TCB1.CTRLA &= ~TCB\_ENABLE\_bm;

}

// TURN ON LED 0 - κίνηση λεπίδων

void LED0\_on(){

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

}

// TURN ON LED 0 - κίνηση λεπίδων

void LED0\_off(){

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

}

// TURN ON LED 1 - κίνηση βάσης

void LED1\_on(){

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

}

// TURN ON LED 1 - κίνηση βάσης

void LED1\_off(){

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

}

// TURN ON LED 2 - emergency LED

void LED2\_on(){

    PORTD.OUTCLR = PIN2\_bm; //LED2 is on

}

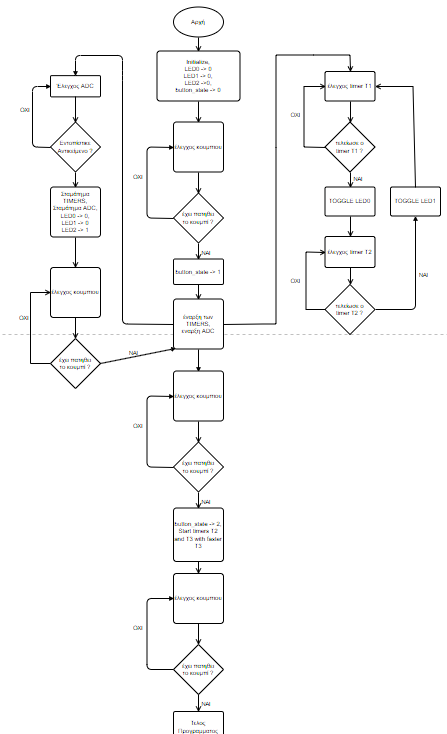
// TURN ON LED 2 - emergency LED

void LED2\_off(){

    PORTD.OUT |= PIN2\_bm; //LED2 is off

}

1. Διάγραμμα ροής τρίτου ερωτήματος :



Κώδικας Τρίτου Ερωτήματος :

/\*

 \* GccApplication1.c

 \*

 \* Author : nektar24

 \*/

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#define Tb 20;

#define Tl 10;

#define Db 10;

#define Dl 5;

#define Tl2 20;

#define Dl2 10;

void start\_timer\_t1\_and\_t2();

void start\_timer\_t1\_and\_t2\_with\_faster\_t2();

void turn\_off\_TIMERS();

void LED0\_on();

void LED0\_off();

void LED1\_on();

void LED1\_off();

void LED2\_on();

void LED2\_off();

bool FINGER = false;

int button\_state = 1;

int LED0 = 0;

int LED1 = 0;

int main() {

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

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

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

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

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

    PORTD.OUT |= PIN2\_bm; //LED2 is off

    //iniitalize the ADC for value LT = 10

    ADC0.CTRLA |= ADC\_RESSEL\_10BIT\_gc;

    ADC0.CTRLA |= ADC\_FREERUN\_bm;

    ADC0.CTRLA |= ADC\_ENABLE\_bm;

    ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc;

    ADC0.DBGCTRL |= ADC\_DBGRUN\_bp;

    ADC0.WINLT |= 10;

    ADC0.INTCTRL |= ADC\_WCMP\_bm;

    ADC0.CTRLE |= ADC\_WINCM0\_bm;

    // ENABLING ADC

    ADC0.COMMAND |= ADC\_STCONV\_bm;

    // Listen for interrupt on bit 5

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

    // Enable Interrupt

    sei();

    while (button\_state < 3)

    {

        ;

    }

    // Disable Interrupt

    cli();

}

// Δάχτυλο στον Ανεμυστήρα

ISR(ADC0\_WCOMP\_vect){

    FINGER = true;

    turn\_off\_TIMERS();

    LED0\_off();

    LED1\_off();

    LED2\_on();

    // Reset Interrupt Flag

    int intflags = ADC0.INTFLAGS;

    ADC0.INTFLAGS = intflags;

}

// Button Press INTERRUPT

ISR(PORTF\_PORT\_vect){

    switch (button\_state){

        case 1:

            // Πάτημα κουμπιού για πρώτη φορά

            start\_timer\_t1\_and\_t2();

            button\_state = 2;

            break;

        case 2:

            if (!FINGER){

                // Πάτημα κουμπιού δευτερη φορά

                // η περίοδος της κυκλικής κίνησης διπλασιάζεται

                start\_timer\_t1\_and\_t2\_with\_faster\_t2();

                button\_state = 3;

            }

            break;

        case 3:

            // LEDs to 0

            LED0\_off();

            LED1\_off();

            // Disable ADC

            ADC0.CTRLA &= ~ADC\_ENABLE\_bm;

            // Disable TCB0 and TCB1

            TCB0.CTRLA &= ~TCB\_ENABLE\_bm;

            TCB1.CTRLA &= ~TCB\_ENABLE\_bm;

            break;

    }

    if (FINGER){

        // Πάτημα κουμπιού για επανέναρξη μετα απο emergency shutdown

        start\_timer\_t1\_and\_t2();

        FINGER = false;

        LED2\_off();

    }

    // Reset Interrupt Flag

    int intflags = PORTF.INTFLAGS;

    PORTF.INTFLAGS = intflags;

}

ISR(TCB1\_INT\_vect){

    if (TCB0.INTFLAGS == 1 ){

        // Timer Βάσης

        // TOGGLE LED 1

        switch (LED1){

            case 0:

                LED1 = 1;

                LED1\_on();

            break;

            case 1:

                LED1 = 0;

                LED1\_off();

            break;

        }

    }

    // Timer λεπίδων

    // TOGGLE LED 0

    switch (LED0){

        case 0:

            LED0 = 1;

            LED0\_on();

        break;

        case 1:

            LED0 = 0;

            LED0\_off();

        break;

    }

    // Reset Interrupt Flag

    int intflags = TCB1.INTFLAGS;

    TCB1.INTFLAGS = intflags;

}

void start\_timer\_t1\_and\_t2(){

    // Set TCB0 to compare/capture mode with waveform generator

    TCB0.CCMPL = Tb;

    TCB0.CTRLA |= TCB\_ENABLE\_bm;

    TCB0.CTRLB |= TCB\_CCMPEN\_bm | TCB\_CNTMODE\_PWM8\_gc;

    // Set TCB1 to compare/capture mode with waveform generator

    TCB1.CCMPL = Tl;

    TCB1.CTRLA |= TCB\_ENABLE\_bm;

    TCB1.CTRLB |= TCB\_CCMPEN\_bm | TCB\_CNTMODE\_PWM8\_gc;

    // Set TCB0 and TCB1 period

    TCB1.CCMPH = Dl;

    TCB0.CCMPH = Db;

}

void start\_timer\_t1\_and\_t2\_with\_faster\_t2(){

    // Set TCB0 to compare/capture mode with waveform generator

    TCB0.CCMPL = Tb;

    TCB0.CTRLA |= TCB\_ENABLE\_bm;

    TCB0.CTRLB |= TCB\_CCMPEN\_bm | TCB\_CNTMODE\_PWM8\_gc;

    // Set TCB1 to compare/capture mode with waveform generator

    TCB1.CCMPL = Tl2;

    TCB1.CTRLA |= TCB\_ENABLE\_bm;

    TCB1.CTRLB |= TCB\_CCMPEN\_bm | TCB\_CNTMODE\_PWM8\_gc;

    // Set TCB0 and TCB1 period with faster TCB1

    TCB1.CCMPH = Dl2;

    TCB0.CCMPH = Db;

}

void turn\_off\_TIMERS(){

    TCB0.CTRLA &= ~TCB\_ENABLE\_bm;

    TCB1.CTRLA &= ~TCB\_ENABLE\_bm;

}

// TURN ON LED 0 - κίνηση λεπίδων

void LED0\_on(){

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

}

// TURN ON LED 0 - κίνηση λεπίδων

void LED0\_off(){

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

}

// TURN ON LED 1 - κίνηση βάσης

void LED1\_on(){

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

}

// TURN ON LED 1 - κίνηση βάσης

void LED1\_off(){

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

}

// TURN ON LED 2 - emergency LED

void LED2\_on(){

    PORTD.OUTCLR = PIN2\_bm; //LED2 is on

}

// TURN ON LED 2 - emergency LED

void LED2\_off(){

    PORTD.OUT |= PIN2\_bm; //LED2 is off

}