

Πανεπιστήμιο Πατρών

Τμήμα Μηχανικών Η/Υ και Πληροφορικής

**Εργαστηριακή Άσκηση 4**

Συνεργάτες:

Θεόφραστος Παξιμάδης 1093460

Κωνσταντίνος Αναστασόπουλος 1093320

Τμήμα: Β4

Ομάδα: 1

Μέρος 1: Κώδικας

/\*

\* lab4.c

\*

\* Created: 09/04/2025 18:06:38

\* Author : Theo

\*/

#include <avr/io.h>

#include <avr/interrupt.h>

#define T1 19

#define T2 39

int x = 0; // Loop control variable

int main(){

PORTD.DIR |= PIN0\_bm | PIN1\_bm; // PINS 1 and 2 are outputs

PORTD.OUT |= PIN0\_bm | PIN1\_bm; // Both disabled

TCA0.SPLIT.CTRLD = TCA\_SINGLE\_SPLITM\_bm; // Enable split mode

TCA0.SPLIT.CTRLA = TCA\_SINGLE\_CLKSEL\_DIV1024\_gc; // Prescaler = 1024

TCA0.SPLIT.CTRLB = TCA\_SPLIT\_LCMP0EN\_bm | TCA\_SPLIT\_HCMP0EN\_bm;

// Pullup enable and sense on both edges

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

// High and low underflow interrupt enable

TCA0.SPLIT.INTCTRL = TCA\_SPLIT\_LUNF\_bm | TCA\_SPLIT\_HUNF\_bm;

sei(); // Start accepting interrupts (breakpoint)

while (1) { // Main loop

if (x == 1) { // First button press

x = x + 1; // Change logic flag to get out of this block

TCA0.SPLIT.LPER = T1; // Wings period

TCA0.SPLIT.HPER = T2; // Base period

TCA0.SPLIT.LCMP0 = 0.6 \* TCA0.SPLIT.LPER; // Wings duty cycle

TCA0.SPLIT.HCMP0 = 0.4 \* TCA0.SPLIT.HPER; // Base duty cycle

TCA0.SPLIT.LCNT = 1; // Minor offset so they don't interrupt together

TCA0.SPLIT.HCNT = 0;

TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm; //Enable TCA (breakpoint)

}

if (x == 3) { // Second button press

PORTD.OUT |= PIN0\_bm | PIN1\_bm; // Both PINS are disabled

x = x + 1; // Change logic flag to get out of this block

// The wings period becomes half the original

TCA0.SPLIT.LPER = (T1 / 2);

// The duty cycle becomes 50%

TCA0.SPLIT.LCMP0 = 0.5 \* TCA0.SPLIT.LPER;

TCA0.SPLIT.LCNT = 0; // // Minor offset so they don't interrupt together

TCA0.SPLIT.HCNT = 1; // (breakpoint)

}

if (x == 5) { // Third button press

PORTD.OUT |= PIN0\_bm | PIN1\_bm; // Both PINS are disabled

x = x + 1; // Change logic flag to get out of this block

TCA0.SPLIT.CTRLA &= ~TCA\_SPLIT\_ENABLE\_bm; // Disable TCA (breakpoint)

}

PORTD.OUT |= 0b00000000; // NOP (breakpoint)

}

}

ISR(TCA0\_LUNF\_vect) { // Every 1 ms, wings

cli();

PORTD.OUT ^= PIN0\_bm; // Complement the wings led

int intflags = TCA0.SINGLE.INTFLAGS; // Clear the flags

TCA0.SINGLE.INTFLAGS = intflags; // (breakpoint)

sei();

}

ISR(TCA0\_HUNF\_vect) { // Every 2 ms, base

cli();

PORTD.OUT ^= PIN1\_bm; // Complement the base led

int intflags = TCA0.SINGLE.INTFLAGS; // Clear the flags

TCA0.SINGLE.INTFLAGS = intflags; // (breakpoint)

sei();

}

ISR(PORTF\_PORT\_vect) { // Button press interrupt

cli();

int intflags = PORTF.INTFLAGS; // Clear the flags

PORTF.INTFLAGS = intflags;

x = x + 1; // Change logic flag to get out of loop (breakpoint)

if (x == 7) { // Reset

x = 1; // (breakpoint)

}

sei();

}

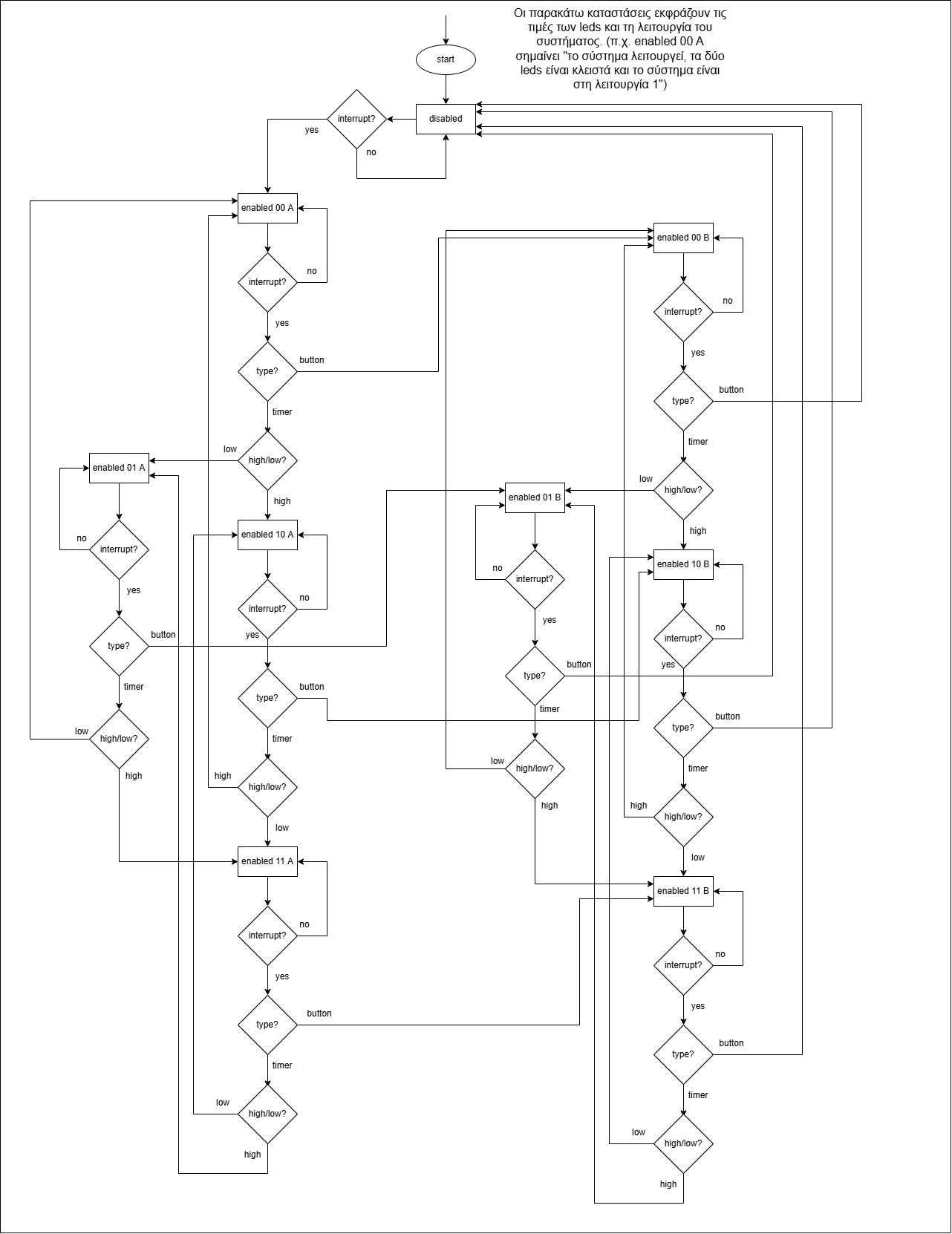
Μέρος 2: Επεξήγηση Κώδικα & Διάγραμμα Ροής

Τα led που χρησιμοποιούνται για την εμφάνιση των κινήσεων του ανεμιστήρα τα PIN0 και PIN1 τα οποία αρχικά είναι και τα δύο κλειστά. Για την δημιουργία των δύο παλμών διαφορετικής διάρκειας, έχει χρησιμοποιηθεί το split mode του TCA0 (TCA0.SPLIT.CTRLD = 1;) με τις αντίστοιχες εντολές. Επιπλέον, έχουν ενεργοποιηθεί οι διακοπές από το pin6 του PORTF. Τα modes του ανεμιστήρα είναι τρία, όταν πατηθεί το κουμπί για πρώτη, δεύτερη και τρίτη φορά.

Την πρώτη φορά που θα πατηθεί, η global μεταβλητή x που αρχικοποιείται με 0, θα γίνει 1 μέσα στην ISR(PORTF\_PORT\_vect) και στην συνέχεια θα οδηγηθεί στην if (x == 1) μέσα στην while. Εκεί, το x θα αυξηθεί πάλι κατά ένα (για να μην τρέξει πάλι η if (x == 1)) και στην συνέχεια θα δημιουργηθούν οι δύο παλμοί. Για τις λεπίδες του ανεμιστήρα, ο παλμός θα είναι διάρκειας Τ1 και το duty cycle είναι το 60% (TCA0.SPLIT.LCMP0 = 0.6 \* TCA0.SPLIT.LPER;). Ομοίως για την βάση που είναι διάρκειας Τ2 με duty cycle 40%. Επίσης, έχει εισαχθεί μία λογική καθυστέρηση, το TCA0.SPLIT.LCNT ξεκινάει από το 1 ενώ το TCA0.SPLIT.HCNT από το 0, για να μην υπάρχει επικάλυψη στις ISR. Επιπλέον, ενεργοποιούνται οι διακοπές από τον TCA0 (TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm;). Οι χρόνοι Τ1 και Τ2 έχουν υπολογιστεί σύμφωνα με τον τύπο της εκφώνησης. Δηλαδή, για τα 2 ms της βάσης υπολογίζεται: . Ομοίως για το Τ1. Στην συνέχεια, σε κάθε θετική ακμή των παλμών αντιστρέφονται τα led0, led1, σύμφωνα με τον κάθε παλμό, που δηλώνουν την κίνηση των λεπίδων και βάσης.

Εάν πατηθεί για δεύτερη φορά το κουμπί, το x αυξάνεται πάλι κατά ένα και γίνεται τώρα 3. Στην while, ισχύει η if (x == 3) και γίνονται οι παρακάτω αλλαγές. Πρώτα το x αυξάνεται πάλι κατά ένα ώστε να μην ισχύει η if (x == 3), αρχικοποιούνται πάλι τα led του ανεμιστήρα ως κλειστά και αλλάζει η διάρκεια του παλμού των λεπίδων. Δηλαδή, η διάρκειά του γίνεται μισή (TCA0.SPLIT.LPER = (T1 / 2);) και το duty cycle 50% (TCA0.SPLIT.LCMP0 = 0.5 \* TCA0.SPLIT.LPER;). Τέλος, εισάγεται η ίδια καθυστέρηση με προηγουμένως.

Εάν ξαναπατηθεί το κουμπί, η μεταβλητή x στην ISR(PORTF\_PORT\_vect), πάλι αυξάνεται κατά 1 και τώρα γίνεται 5. Στην while, ισχύει η if (x == 5) και τώρα απλώς τα led σβήνουν και ο TCA απενεργοποιείται (TCA0.SPLIT.CTRLA &= ~TCA\_SPLIT\_ENABLE\_bm;). Εάν πατηθεί πάλι το κουμπί, τότε η ISR(PORTF\_PORT\_vect) θα κάνει την μεταβλητή x 1 και θα ξεκινήσει πάλι η διαδικασία από την αρχή.



Διάγραμμα Ροής Λειτουργίας Ανεμιστήρα