**Εργαστήριο Προηγμένοι Μικροεπεξεργαστές**

Εργαστηριακή Άσκηση 02:

Φανάρια Κυκλοφορίας

2024-25

**Περιεχόμενα**

[Διάγραμμα Ροής 3](#__RefHeading___Toc1188_1975538019)

[Περιγραφή Κώδικα 5](#__RefHeading___Toc30_1975538019_Copy_1)

[Global: 5](#__RefHeading___Toc666_1975538019_Copy_1)

[Main: 5](#__RefHeading___Toc666_1975538019)

[ISR(PORTF\_PORT\_vect): 7](#__RefHeading___Toc1520_1909938359)

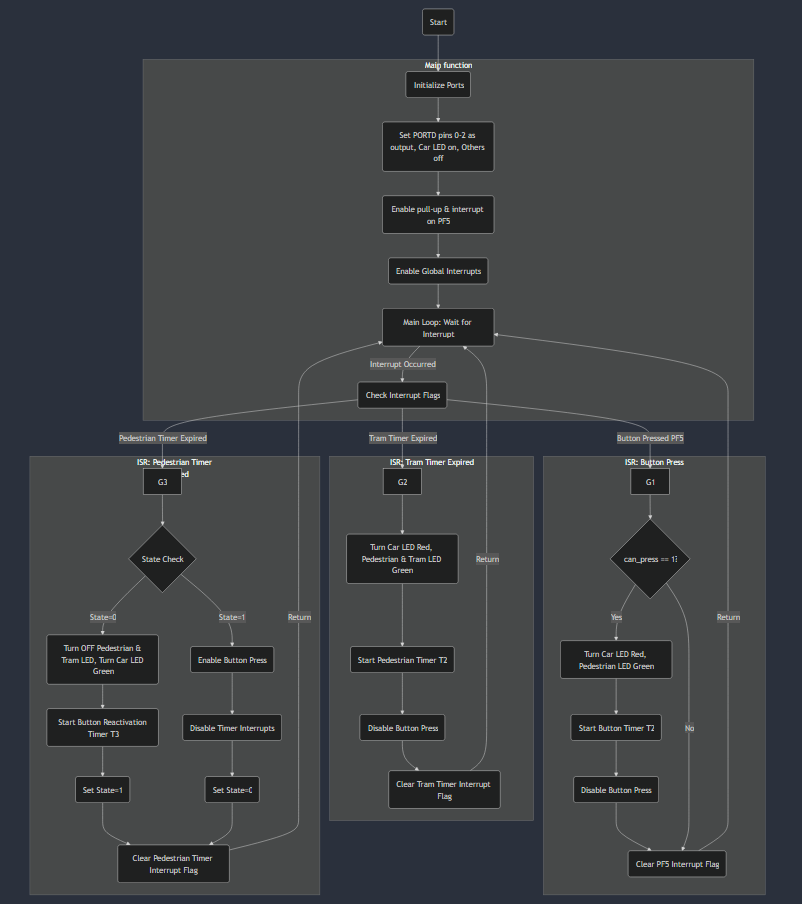
[ISR(TCA0\_HUNF\_vect): 7](#__RefHeading___Toc1522_1909938359)

[ISR(TCA0\_LUNF\_vect): 8](#__RefHeading___Toc1524_1909938359)

[Κώδικας: 9](#__RefHeading___Toc666_1975538019_Copy_1_)

# Διάγραμμα Ροής

*Το διάγραμμα δημιουργήθηκε μεσω mermaid.js*



# Περιγραφή Κώδικα

Ολόκληρος ο κώδικας ειναι διαθέσιμος στο τέλος της αναφοράς.

Ο επεξεργαστής τρέχει στα 20MHz (Το Default του Microchip ειναι 1MHz)

## Global:

Υπολογίζουμε τους χρόνους T1, T2, T3 για την προσομοίωση όπως αναγράφονται στα σχόλια, με στόχο το T1 να ειναι μεγαλύτερο του Τ2 + Τ3, βεβαιώνοντας οτι θα υπάρχει χρόνος για τα αυτοκίνητα ανάμεσα στα περάσματα του τραμ.

Εαν επιθυμούσαμε να έχουμε μεγαλύτερους, θα έπρεπε εέτε:

α) Να χρησιμοποιήσουμε μεγαλύτερο counter

β) Όταν χτυπάει το interrupt να αυξάνεται μια μεταβλητή και οι δράσεις να γίνονται όταν αυτή η μεταβλητή φτάνει μια προκαθορισμένη τιμή (π.χ. εαν θέλαμε το τραμ να περνάει ανα 130ms, θα έπρεπε να έχουμε μια global μεταβλητή και ο κώδικας του ISR θα έπρεπε να κανει increment τον counter 10 φορές πρωτού εκτελέσει τις λειτουργίες.

#define T1 255                  // T = 255 / 19.5 = 13.07ms

#define T2 128                  //  T = 128 / 19.5 = 6.56ms

#define T3 32                   // T = 32 / 19.5 = 1.64ms

Ορίζουμε 1 global μεταβλητές που ελέγχει εαν τα low bits του TCA0 εκτελούν το T2 ή του T3 (για το ISR).

int state = 0;

## Main:

Αρχικοποιείται το σύστημα θέτωντας τα bits 0,1,2 του PORTD σαν έξοδο (εφόσον αυτά τα LED θα χρησιμοποιήσουμε) και τα θέτουμε κατάλληλα με την υπόθεση οτι στην αρχικοποίηση του συστήματος τα αυτοκίνητα έχουν πράσινο και δεν περνάει τραμ.

    PORTD.DIR |= 0b00000111; //PIN 0-2 is output

    //init LED (assume cars have priority)

    PORTD.OUTCLR = 0b00000001;

    PORTD.OUT |= 0b00000110;

Ορίζουμε ως κουμπί το PIN5 του PORTF και κανουμε split το TCA0 με prescaler 1024.

    // pullup enable and Interrupt enabled with sense on rising edge (button release)

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

    // split timer with prescaler 1024

    TCA0.SPLIT.CTRLA = TCA\_SPLIT\_CLKSEL\_DIV1024\_gc | TCA\_SPLIT\_ENABLE\_bm;

    TCA0.SPLIT.CTRLD = TCA\_SPLIT\_SPLITM\_bm;

Ορίζουμε ως τιμή στα HIGH bits του μετρητή τον Τ2, ενώ στα LOW bits τον Τ1, και ενεργοποιούμε τα interrupts για το Τ1 (μετράει ο χρόνος για το πέρασμα του τραμ).

Έτσι ο μετρητής μετράει απο αυτές τις τιμές μέχρι να κανει underflow (-- ανά ftimer μέχρι το 0), και τότε ενεργοποιούνται τα αντίστοιχα interrupts. Χρησιμοποιούμε αυτή την μεθοδολογία καθώς σε split mode τα high bits δεν περιέχουν HCMP interrupt.

    TCA0.SPLIT.LCNT = T2;

    TCA0.SPLIT.HCNT = T1;

    TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_HUNF\_bm;    //Interrupt Enable for tram

Ενεργοποιούμε τα interrupts και μπαίνουμε σε έναν ατέρμονο βρόγχο while, περιμένοντας interrupt.

  sei();

    while (1) {

        ;

    }

## ISR(PORTF\_PORT\_vect):

Σβήνουμε το led των αυτοκινήτων (κόκκινο) και ανάβουμε το led των πεζών (πράσινο)

PORTD.OUT |= 0b00000001;                    //turn car led OFF (red light)

PORTD.OUTCLR = 0b00000100;                  //turn pedestrian led ON (green light)

Καθαρίζουμε τα flags (για να ξεκινήσει η μέτρηση απο την αρχή) και ενεργοποιούμε τα interrupts για το Τ2

        TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_LUNF\_bm;   //clear flags

        TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_LUNF\_bm ;   //enable timer for lights

        can\_press = 0;

Τέλος, καθαρίζουμε τα interrupt flags για το PORT 5 και απενεργοποιούμε τα interrupts από το κουμπι.

    PORTF.INTFLAGS |= PORT\_INT5\_bm;             //clear the interrupt flag

    PORTF.PIN5CTRL = PORT\_ISC\_INTDISABLE\_gc;    //disable button interrupt

## ISR(TCA0\_HUNF\_vect):

Αυτό το interrupt ενεργοποιείται όταν τελειώνει το Τ1 (περνάει το τραμ).

Αρχικά ανάβει το κόκκινο για τα αυτοκίνητα και το πράσινο για τους πεζούς, καθώς και ενεργοποιείται το led που προσομοιώνει το τραμ.

PORTD.OUT |= 0b00000001;                    //turn car led OFF (red light)

PORTD.OUTCLR = 0b00000110;                  //turn pedestrian/tram led ON (green light)

Καθαρίζουμε τα flags (για να ξεκινήσει η μέτρηση απο την αρχή), ενεργοποιούμε τα interrupts για το Τ2

        TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_LUNF\_bm;   //clear flags

        TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_LUNF\_bm ;   //enable timer for lights

Τέλος, καθαρίζουμε τα interrupt flags για τον HUNF.

    TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_HUNF\_bm;   //clear flags

## ISR(TCA0\_LUNF\_vect):

Αυτό το interrupt ενεργοποιείται όταν τελειώνει το Τ2 ή το Τ3.

Η μεταβλητή state αποθηκεύει εαν ο counter μετράει αντίστροφα απο τον Τ2 (state = 0) ή από τον T3 (state = 1).

Στην πρώτη περίπτωση, κλέινει τα φώτα των πεζών και του τραμ (κόκκινο για τους πεζούς / εφυγε το τραμ), ανάβει το πράσινο για τα αυτοκίνητα, και ορίζει την τιμή του LCNT = T3.

Θέτει state = 1, έτσι ωστε όταν ξαναχτυπήσει αυτό το interrupt να τρέξει ο επόμενος κώδικας (καθώς ο counter λειτουργεί για το T3).

    if(state == 0) {

        PORTD.OUT |= 0b00000110;                    //turn pedestrian / tram led OFF (red light)

        PORTD.OUTCLR = 0b00000001;                  //turn car led ON (green light)

        TCA0.SPLIT.LCNT = T3;                       //start timer for 3

        state = 1;

    }

Στην δέυτερη περίπτωση (πάροδος του Τ3) ενεργοποιεί ξανα το κουμπί, απενεργοποιεί τον μετρητή και επαναφέρει το state = 0.

    else if(state == 1) {

        PORTF.PIN5CTRL = PORT\_PULLUPEN\_bm | PORT\_ISC\_RISING\_gc;     //enable button

        TCA0.SPLIT.INTCTRL &= ~TCA\_SPLIT\_LCMP0\_bm;                  //disable timer

        state = 0;

    }

# Κώδικας:

#include <avr/io.h>

#include <avr/interrupt.h>

//f\_clk = 20Mhz

//prescaler = 1024

//ftimer = 20MHz / 1024 = 19,531KHz

//T = value / ftimer

#define T1 255                  // T = 255 / 19.5 = 13.07ms

#define T2 128                  //  T = 128 / 19.5 = 6.56ms

#define T3 32                   // T = 32 / 19.5 = 1.64ms

int state = 0;

/\*

PF5 -> PEDESTRIAN BUTTON

(1 = RED 0 = GREEN)

P0 -> CAR LED

P1 -> TRAM SIMULATION

P2 -> PEDESTRIAN LED

T1 -> TRAM TIMER

T2 -> BUTTON TIMER

T3 -> BUTTON REACTIVATION TIMER

\*/

int main() {

    PORTD.DIR |= 0b00000111; //PIN 0-2 is output

    //init LED (assume cars have priority)

    PORTD.OUTCLR = 0b00000001;

    PORTD.OUT |= 0b00000110;

    // pullup enable and Interrupt enabled with sense on rising edge (button release)

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

    // split timer with prescaler 1024

    TCA0.SPLIT.CTRLA = TCA\_SPLIT\_CLKSEL\_DIV1024\_gc | TCA\_SPLIT\_ENABLE\_bm;

    TCA0.SPLIT.CTRLD = TCA\_SPLIT\_SPLITM\_bm;

    // HIGH = BUTTON / REACTIVATION TIMER

    // LOW =  TRAM TIMER

    TCA0.SPLIT.LCNT = T2;

    TCA0.SPLIT.HCNT = T1;

    TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_HUNF\_bm;    //Interrupt Enable for tram

    sei();                                      //begin accepting interrupt signals

    while (1) {

        ;                                       //similar to a nop function

    }

}

// button pressed

ISR(PORTF\_PORT\_vect){

        PORTD.OUT |= 0b00000001;                    //turn car led OFF (red light)

        PORTD.OUTCLR = 0b00000100;                  //turn pedestrian led ON (green light)

        TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_LUNF\_bm;   //clear flags

        TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_LUNF\_bm ;   //enable timer for lights

        PORTF.INTFLAGS |= PORT\_INT5\_bm;             //clear the interrupt flag

    PORTF.PIN5CTRL = PORT\_ISC\_INTDISABLE\_gc;    //disable button interrupt

}

ISR(TCA0\_HUNF\_vect){

    PORTD.OUT |= 0b00000001;                    //turn car led OFF (red light)

    PORTD.OUTCLR = 0b00000110;                  //turn pedestrian/tram led ON (green light)

    TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_LUNF\_bm;   //clear flags

    TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_LUNF\_bm ;   //enable timer for lights

    PORTF.PIN5CTRL = PORT\_ISC\_INTDISABLE\_gc;    //disable button interrupt

    TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_HUNF\_bm;   //clear flags

}

//pedestrian walk time ran out OR button timer ran out

ISR(TCA0\_LUNF\_vect){

    if(state == 0) {

        PORTD.OUT |= 0b00000110;                    //turn pedestrian / tram led OFF (red light)

        PORTD.OUTCLR = 0b00000001;                  //turn car led ON (green light)

        TCA0.SPLIT.LCNT = T3;                       //start timer for 3

        state = 1;

    }

    else if(state == 1) {

        PORTF.PIN5CTRL = PORT\_PULLUPEN\_bm | PORT\_ISC\_RISING\_gc;     //enable button

        TCA0.SPLIT.INTCTRL &= ~TCA\_SPLIT\_LCMP0\_bm;                  //disable timer

        state = 0;

    }

    TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_LUNF\_bm;                       //clear flags

}