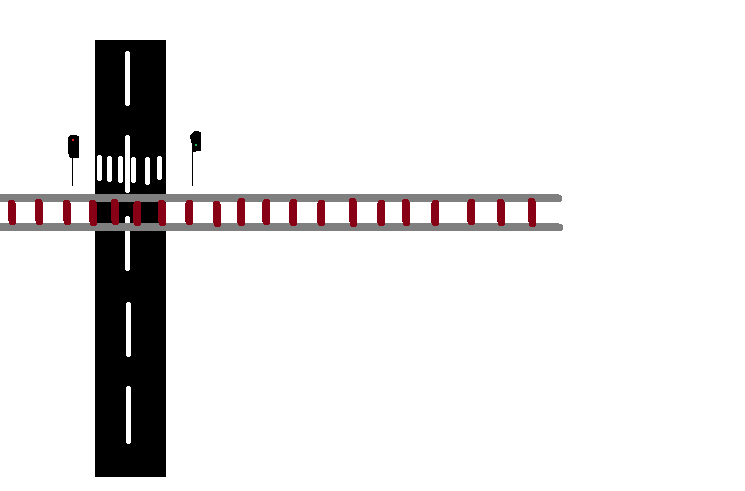
**Εργαστήριο Προηγμένων Μικροεπεξεργαστών**

ΟΜΑΔΑ Β4 - ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 1

Μιχαηλίδης Νεκτάριος – 1072594 – 4ο έτος

Περικλής Κύρκος – 1067524 – 5ο έτος



Ερωτήματα :

Με βάση την εκφώνηση το φανάρι των αυτοκινήτων παραμένει Πράσινο πάντα , εκτός από το αν πατηθεί το κουμπί ή περάσει το τρένο. Το τρένο περνάει από την διάβαση κάθε t1 δευτερόλεπτα.

Κάναμε τον timer t2 μικρότερο από τον timer t1 για να προλαβαίνει το Φανάρι των αυτοκινήτων να ανάψει πράσινο πριν έρθει το επόμενο Τραμ.

Κάναμε τον timer t3 μεγαλύτερο από τον timer t2 αλλιώς θα ηταν το ιδιο με το να μην υπάρχει καθόλου ο t3.

Άρα συνεπώς έχουμε t1 > t3 > t2 για το δικο μας πρόβλημα.

1. Εικόνα που περιέχει διάγραμμα

   Περιγραφή που δημιουργήθηκε αυτόματαΔιάγραμμα ροής για το ερώτημα 1 :

ΚΩΔΙΚΑΣ ΕΡΩΤΗΜΑΤΟΣ 1 :

/\*

 \* GccApplication1.c

 \*

 \* Author : nektar24

 \*/

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#define T2 100;

#define T3 200;

void LED\_on();

void LED\_off();

void start\_timer\_t2\_and\_t3();

bool button\_pressed = false; // is true when t3 is running

int LED = 0; // is true when t2 is running (and light is on)

int main() {

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

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

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

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

    // Listen for interrupt on bit 5

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

    // Enable Interrupt

    sei();

    while (1)

    {

        while (!button\_pressed)

        {

            LED\_off();

        }

    }

    // Disable Interrupt

    cli();

}

// Button Press INTERRUPT

ISR(PORTF\_PORT\_vect){

    button\_pressed = true;

    LED\_on();

    start\_timer\_t2\_and\_t3();

    // Reset Interrupt Flag

    int intflags = PORTF.INTFLAGS;

    PORTF.INTFLAGS = intflags;

}

// TIMER T2 INTERRUPT

ISR(TCA0\_CMP0\_vect){

    LED\_off();

    // Reset Interrupt Flag

    int intflags2 = TCA0.SINGLE.INTFLAGS;

    TCA0.SINGLE.INTFLAGS = intflags2;

}

// TIMER T3 INTERRUPT

ISR(TCA0\_CMP1\_vect){

    button\_pressed = true;

    // stop timer to save resources

    TCA0.SINGLE.CTRLA = 0 ;

    // Reset Interrupt Flag

    int intflags2 = TCA0.SINGLE.INTFLAGS;

    TCA0.SINGLE.INTFLAGS = intflags2;

}

void LED\_on(){

    PORTD.OUTCLR = PIN0\_bm;

    PORTD.OUTCLR = PIN2\_bp;

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

    LED = 1;

}

void LED\_off(){

    PORTD.OUTCLR = PIN0\_bp;

    PORTD.OUTCLR = PIN2\_bm;

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

    LED = 0;

}

void start\_timer\_t2\_and\_t3(){

    //-----initialize timer

    TCA0.SINGLE.CNT = 0;

    TCA0.SINGLE.CTRLB = 0;

    // TRIGGER INTERRUPT AT T2

    TCA0.SINGLE.CMP0 = T2;

    // TRIGGER INTERRUPT AT T3

    TCA0.SINGLE.CMP1 = T3;

    TCA0.SINGLE.CTRLA = 0x7<<1;

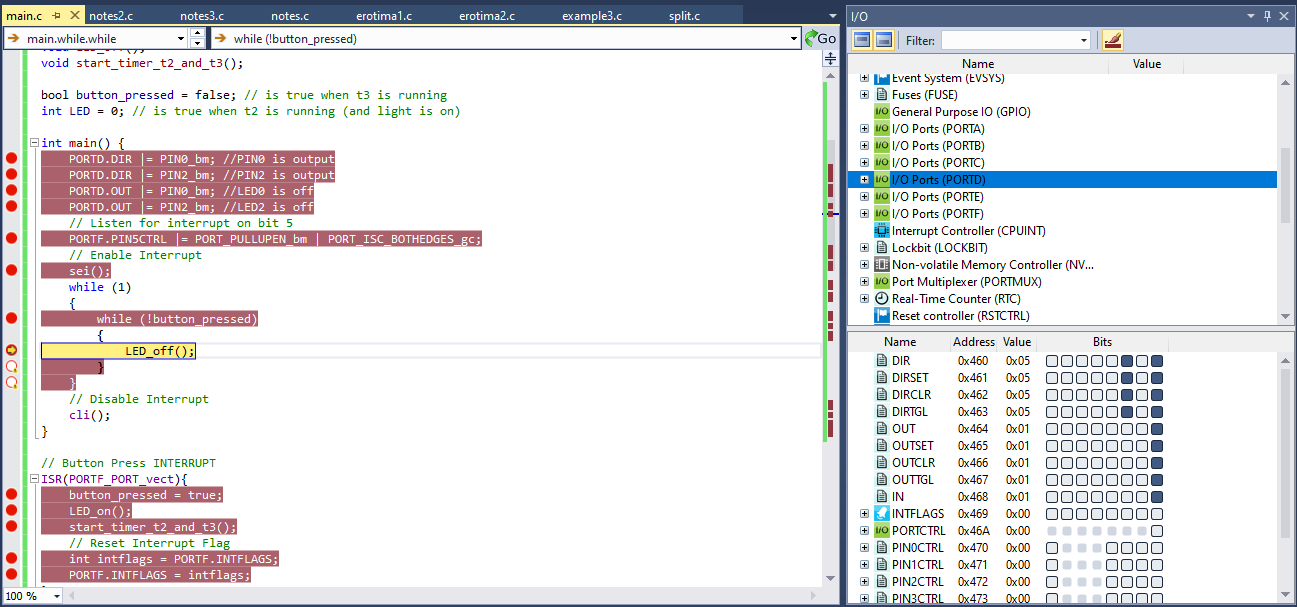
    TCA0.SINGLE.CTRLA |= 1;

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

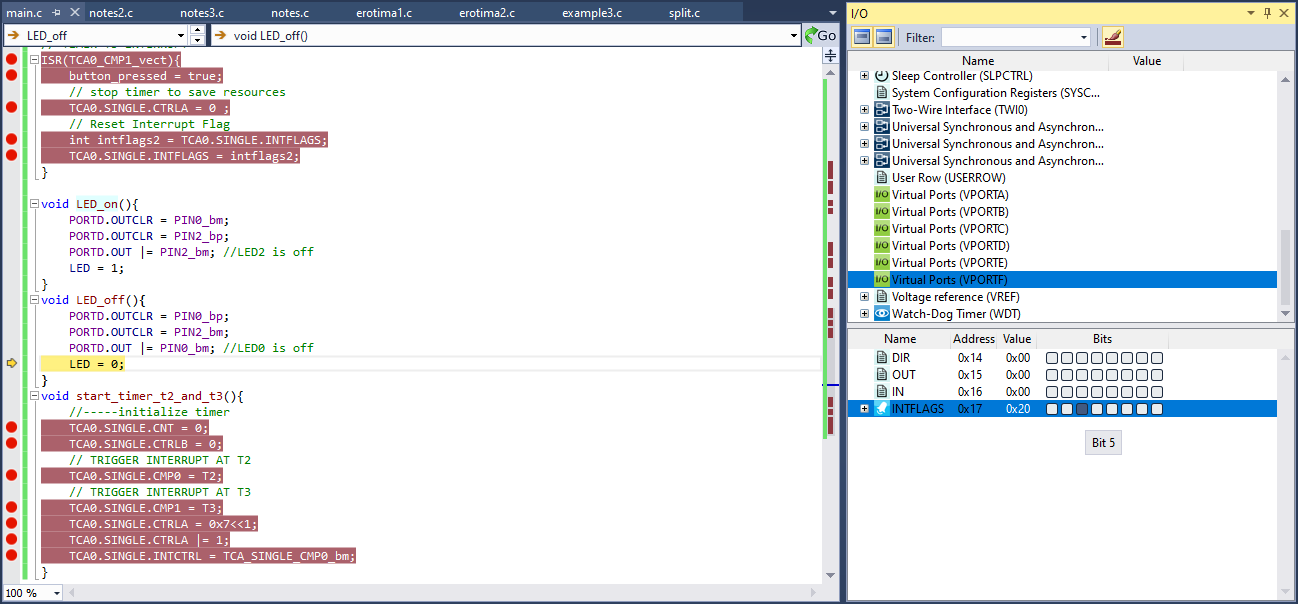
}

Χρησιμοποιούμε την μεταβλητή button\_pressed για να δούμε αν είναι δυνατή η χρήση του κουμπιού. Αν η μεταβλητή button\_pressed είναι false μπορούμε να πατήσουμε το κουμπί. Αν είναι true σημαίνει ότι έχει πατηθεί στο παρελθόν και δεν έχει κάνει ακόμα reset. Κάθε φορά που ολοκληρώνεται ο timer\_t3 η μεταβλητή button\_pressed γίνεται false.

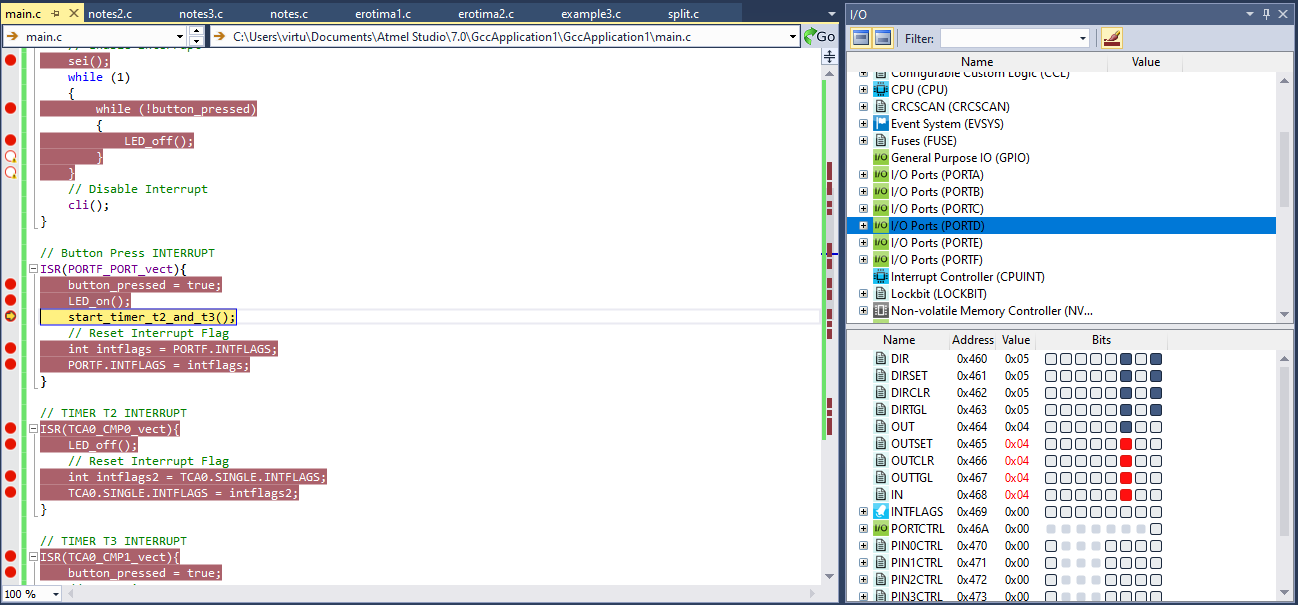
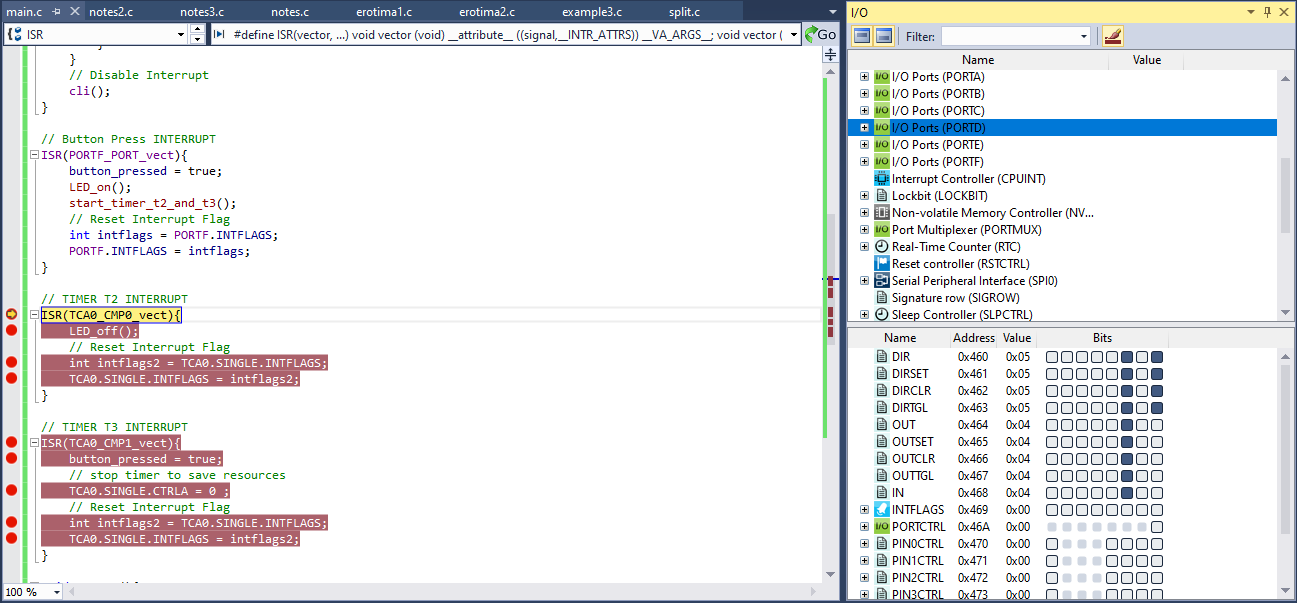
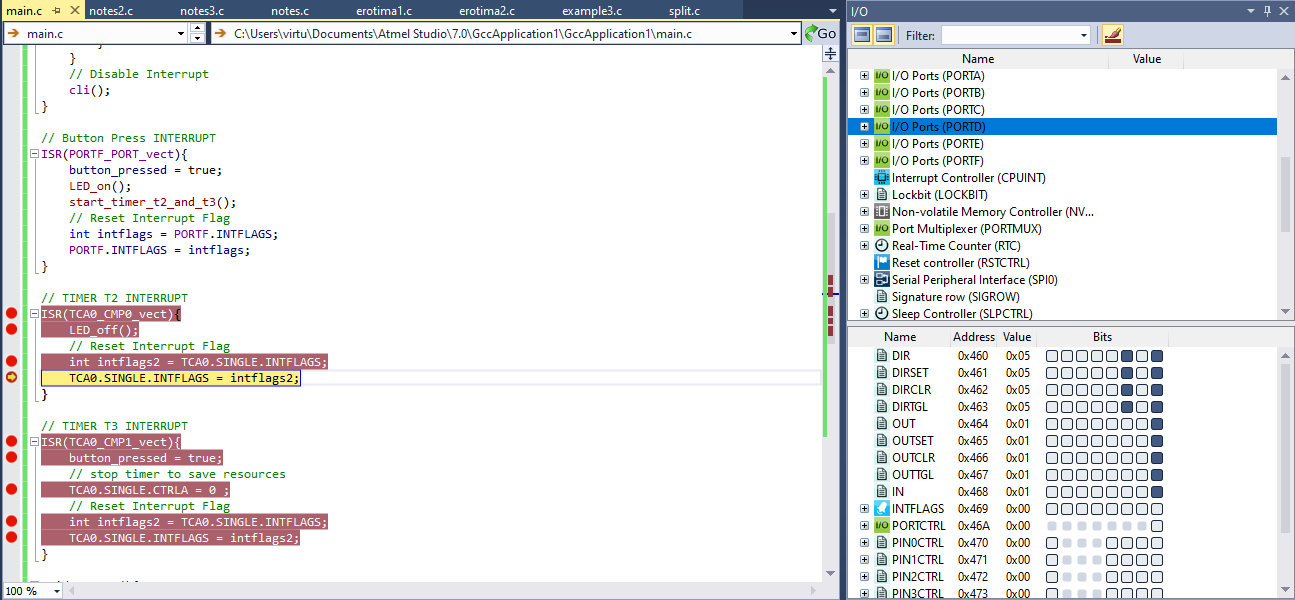
Χρησιμοποιούμε την μεταβλητή LED για να δούμε αν είναι πράσινο το φανάρι των πεζών. LED = 1 σημαίνει πράσινο φανάρι για τους πεζούς.

Στην αρχή του κώδικα εκτελείται η συνάρτηση LED\_off() και το πρόγραμμα περιμένει το interrupt από το VPORTF bit 5.

Όπως βλέπουμε στο screenshot. To LED των αυτοκινήτων PD2 είναι 0 στο output του, δηλαδή πάντα πράσινο, και το LED των πεζών είναι 1.

Όταν κάνουμε το bit 5 του VPORTF 1 ενεργοποιείτε το Interrupt του κουμπιού.

Το οποίο προκαλεί την συνάρτηση LED\_on() να τρέξει , η οποία ανάβει το φανάρι των πεζών και κλείνει το φανάρι των αυτοκινήτων όπως φαίνεται στην εικόνα

στη συνέχεια ξεκινάνε οι timer 2 και 3 και το πρόγραμμα επιστρέφει στο αρχικό loop όπου δεν μπορεί να αλλάξει την θέση των φαναριών καθώς η μεταβλητή button\_pressed είναι true.

Όταν ολοκληρωθεί ο timer 2 ο οποίος είναι και ο μικρότερος πάντα σύμφωνα με την υπόθεση μας το πρόγραμμα μπαίνει στο interrupt ISR του t2 και αντιστρέφει τα φανάρια. (όπως φαίνεται στο ακριβώς από πάνω screenshot)

Όταν ολοκληρωθεί ο t3 το πρόγραμμα μπαίνει στο interrupt ISR του t3 o οποίος απενεργοποιεί τον timer και απενεργοποιεί την μεταβλητή button\_pressed για να επιστρέψει το πρόγραμμα στην αρχική του λειτουργεία.

1. Εικόνα που περιέχει διάγραμμα, σχηματικό

   Περιγραφή που δημιουργήθηκε αυτόματαΔιάγραμμα ροής δεύτερου ερωτήματος (με το τρένο) :

ΚΩΔΙΚΑΣ ΕΡΩΤΗΜΑΤΟΣ 2 :

/\*

 \* GccApplication1.c

 \*

 \* Author : nektar24

 \*/

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#define T1 800;

#define T2 100;

#define T3 200;

void LED\_on();

void LED\_off();

void start\_timer\_t1();

void start\_timer\_t2();

void start\_timer\_t3();

void init\_timer\_t2\_and\_t3();

bool button\_pressed = false; // is true when t3 is running

int LED = 0; // is true when t2 is running (and light is on)

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

    start\_timer\_t1();

    // Listen for interrupt on bit 5

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

    // Enable Interrupt

    sei();

    while (1)

    {

        LED\_off();

    }

    // Disable Interrupt

    cli();

}

// Button Press INTERRUPT

ISR(PORTF\_PORT\_vect){

    button\_pressed = true;

    start\_timer\_t2();

    start\_timer\_t3();

    LED\_on();

    // Reset Interrupt Flag

    int intflags = PORTF.INTFLAGS;

    PORTF.INTFLAGS = intflags;

}

// TIMER T1 INTERRUPT

ISR(TCB0\_CCMP\_vect){

    start\_timer\_t2();

    LED\_on();

    start\_timer\_t1();

    PORTD.OUTCLR = PIN1\_bp; // TRAM LED IS ON

    // Reset Interrupt Flag

    int intflags = TCB0.INTFLAGS;

    TCB0.INTFLAGS = intflags;

}

// TIMER T2 INTERRUPT

ISR(TCA0\_CMP0\_vect){

    LED\_on();

    // Reset Interrupt Flag

    int intflags2 = TCA0.SINGLE.INTFLAGS;

    TCA0.SINGLE.INTFLAGS = intflags2;

}

// TIMER T3 INTERRUPT

ISR(TCA0\_CMP1\_vect){

    button\_pressed = true;

    TCA0.SINGLE.CTRLA = 0 ;

    // Reset Interrupt Flag

    int intflags3 = TCA0.SINGLE.INTFLAGS;

    TCA0.SINGLE.INTFLAGS = intflags3;

}

void LED\_on(){

    PORTD.OUTCLR = PIN0\_bm;

    PORTD.OUTCLR = PIN2\_bp;

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

    LED = 1;

}

void LED\_off(){

    PORTD.OUTCLR = PIN0\_bp;

    PORTD.OUTCLR = PIN1\_bp;

    PORTD.OUTCLR = PIN2\_bm;

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

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

    LED = 0;

}

void start\_timer\_t1(){

    //-----initialize timer t1

    TCB0.CNT = 0;

    TCB0.CTRLB = 0;

    // TRIGGER INTERRUPT AT T1

    TCB0.CCMP = T1;

    TCB0.CTRLA = 0x7<<1;

    TCB0.CTRLA |= 1;

    TCB0.INTCTRL = TCB\_CCMPINIT\_bm;

}

void start\_timer\_t2(){

    // TRIGGER INTERRUPT AT T2

    TCA0.SINGLE.CMP0 = T2;

}

void start\_timer\_t3(){

    // TRIGGER INTERRUPT AT T3

    TCA0.SINGLE.CMP1 = T3;

}

void init\_timer\_t2\_and\_t3(){

    //-----initialize timer

    TCA0.SINGLE.CNT = 0;

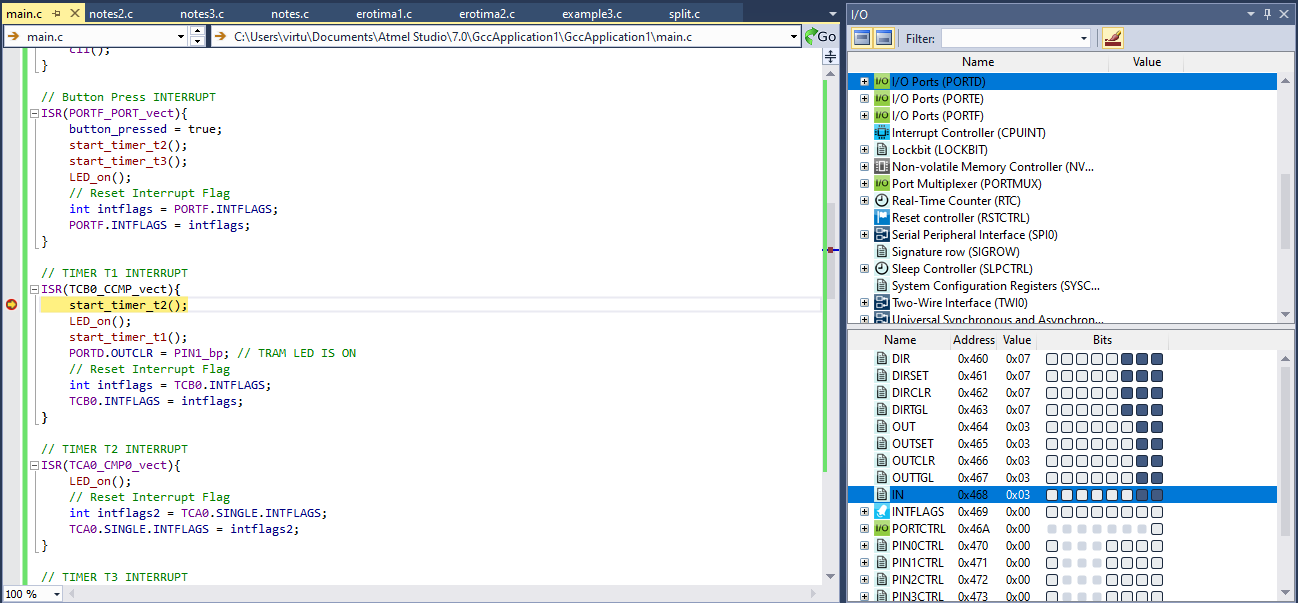
    TCA0.SINGLE.CTRLB = 0;

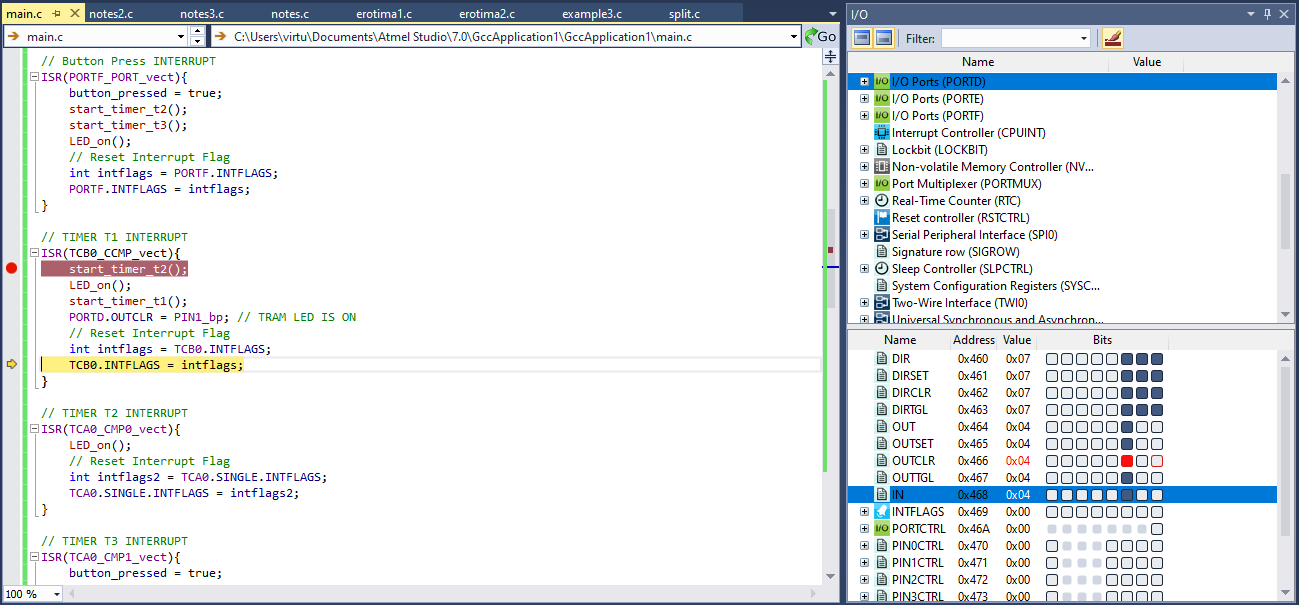
    TCA0.SINGLE.CTRLA = 0x7<<1;

    TCA0.SINGLE.CTRLA |= 1;

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

}

Η μόνη διαφοροποίηση του ερωτήματος 2 είναι ότι προσθέτουμε έναν timer t1 και τον τρέχουμε στο TCB0 (καθώς δεν καταφέραμε να κάνουμε το TCA0.SPLIT να δουλέψει.) ο οποίος ξεκινάει να τρέχει στην αρχή του προγράμματος , μετά τα initializations (στην γραμμή 27). Κάθε φορά που τελειώνει ο timer t1 καλεί το ISR interrupt το οποίο αλλάζει χρώμα στα τρία LED και ξεκινάει τον timer t2 καθώς και τον timer t1 πάλι , για να έχουμε περιοδικότητα της διάβασης του Τραμ.



Παραδοχή :

Στο ερώτημα 2 δεν καταφέραμε να κάνουμε το split και έτσι χρησιμοποιήσαμε τον TCB0 για το τρένο και τον TCA0 για το t2 και t3 τα οποία κάνουν δυο διαφορετικά checks χρησιμοποιόντας τα cmp0 και cmp1.