

Πανεπιστήμιο Πατρών

Τμήμα Μηχανικών Η/Υ και Πληροφορικής

**Εργαστηριακή Άσκηση 2**

Συνεργάτες:

Θεόφραστος Παξιμάδης 1093460

Κωνσταντίνος Αναστασόπουλος 1093320

Τμήμα: Β4

Ομάδα: 1

Μέρος 1: Κώδικας

/\*

\* lab2.c

\*

\* Created: 15/03/2025 15:44:32

\* Author : Theo & Dean

\*/

#include <avr/io.h>

#include <avr/interrupt.h>

#define value1 200 // Tram counter counts down from this value

#define value2 30 // After the press of the button, the counter counts down to value2 and changes the state of the traffic light

#define value3 55 // The pedestrian cannot press the button until the counter counts down to this value

int allow\_button = 1; // Pedestrians are allowed to press the button

int int\_type; // Indicates which type of interrupt is allowed

int x = 0; // Loop control logic flag

int main() {

// Initialize the traffic light

PORTD.DIR |= 0b00000111; // Pins 0, 1 and 2 of port D are outputs

PORTD.OUT |= 0b00000011; // Red light for pedestrians and tram does not pass

PORTD.OUT &= 0b11111011; // Green light for cars

// Initialize the counter

TCA0.SPLIT.CTRLD = 1; // Enable split mode

TCA0.SPLIT.HCNT = value1; // Start from value1 until you reach zero

TCA0.SPLIT.CTRLA = 0x7<<1 | 1; // Prescaler value = 1024 and the peripheral is enabled

TCA0.SPLIT.INTCTRL |= 0b00010010; // Compare interrupt enable (low) and underflow interrupt enable (high)

// Initialize the button

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; // Pullup enable and interrupt enable with sense on both edges

sei(); // Start accepting interrupts

while (1) { // Main loop

if (x == 1) { // This block runs after the press of the button

x = 0; // Break out of the conditional block after this iteration

TCA0.SPLIT.LCNT = value3; // Initialize counter to value3

TCA0.SPLIT.LCMP0 = 0; // Pedestrian can use the button after the counter counts down from value3 (insert breakpoint)

int\_type = 2; // Button time interrupt

}

if (x == 2) { // This block runs after tram passes

x = 0; // Break out of the conditional block after this iteration

TCA0.SPLIT.HCNT = value1; // Reset tram counter

TCA0.SPLIT.LCNT = value2; // Clear traffic light counter

TCA0.SPLIT.LCMP0 = 0; // (insert breakpoint)

int\_type = 3; // Reset phase after tram passes

}

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

}

cli(); // Stop accepting interrupts

}

ISR(PORTF\_PORT\_vect) { // Is triggered after the press of the button

if (allow\_button == 1) { // If pedestrian is allowed to press the button

allow\_button = 0; // Pedestrian cannot press the button anymore

PORTD.OUT |= 0b00000100; // Red light for cars

PORTD.OUT &= 0b11111110; // Green light for pedestrians

int\_type = 1; // traffic light interrupt

TCA0.SPLIT.LCNT = value2; // Clear counter

TCA0.SPLIT.LCMP0 = 0; // (insert breakpoint)

}

PORTF.INTFLAGS = PIN5\_bm; // Clear the interrupt flag

}

ISR(TCA0\_LCMP0\_vect) { // Runs after the low counter counts to zero

if (int\_type == 1) { // traffic light changes state

PORTD.OUT |= 0b00000001; // Red light for pedestrians

PORTD.OUT &= 0b11111011; // Green light for cars

x = 1; // Change flag to get out of the loop (insert breakpoint)

}

if (int\_type == 2) { // After time passes after the press of the button

allow\_button = 1; // Allow the press of the button (insert breakpoint)

int\_type = 0; // None of the low counter interrupts should happen

}

if (int\_type == 3) { // Reset phase after tram passes

int\_type = 0; // None of the counter interrupts should happen

PORTD.OUT &= 0b11111011; // Green light for cars

PORTD.OUT |= 0b00000011; // Red light for pedestrians and tram does not pass

x = 1; // (insert breakpoint)

}

TCA0.SPLIT.INTFLAGS = TCA\_SPLIT\_LCMP0\_bm; // Clear the interrupt flag

}

ISR(TCA0\_HUNF\_vect) { // Tram interrupt

PORTD.OUT |= 0b00000100; // Red light for cars

PORTD.OUT &= 0b11111100; // Green light for pedestrians and tram passes

allow\_button = 0; // Do not allow the press of the button

x = 2; // (insert breakpoint)

TCA0.SPLIT.INTFLAGS = TCA\_SPLIT\_HUNF\_bm; // Clear the interrupt flag

}

Μέρος 2: Επεξήγηση Κώδικα & Διάγραμμα Ροής

Για την αρχικοποίηση των leds, θέτουμε τα τρία δεξιότερα bits του PORTD ως output (PORTD.DIR |= 0b00000111;) και τα αρχικοποιούμε στις τιμές 0, 1 και 1 αντίστοιχα. Άρα, σύμφωνα με την εκφώνηση, στην αρχή το μόνο φανάρι που είναι πράσινο (ανοιχτό led) είναι αυτό των αμαξιών. Στην συνέχεια, γίνεται η αρχικοποίηση των μετρητών. Πρώτα ενεργοποιείται το split mode (TCA0.SPLIT.CTRLD = 1;), θέτουμε το value του prescaler 1024 (TCA0.SPLIT.CTRLA = 0x7<<1 | 1;), και στον counter high τοποθετούμε την τιμή value1 ώστε να μετράει από αυτήν τιμή προς το 0. Επίσης, στον Interrupt control register ενεργοποιούνται τα bits 1 και 4 (TCA0.SPLIT.INTCTRL |= 0b00010010;), ώστε να ενεργοποιηθούν οι διακοπές από το underflow του high counter και το compare του low αντίστοιχα. Επιπλέον, ενεργοποιούνται οι διακοπές στο pin 5 του portf (PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc;). Τέλος, πρέπει να σημειωθεί ότι εξαιτίας του split mode οι μετρητές μετράνε αντίστροφα.

Στο while loop που ακολουθεί, μπορούν να κληθούν δύο διακοπές. Η πρώτη είναι το κουμπί που πατάνε οι πεζοί (portf pin5) ή όταν περάσει το χρονικό διάστημα Τ1 για το τραμ. Πρώτα θα εξεταστεί η περίπτωση του κουμπιού.

Αν πατηθεί το pin5 του portf, τότε το πρόγραμμα οδηγείται στην ISR(PORTF\_PORT\_vect). Η ISR της διακοπής θέτει την global μεταβλητή allow\_button = 0, η οποία είχε αρχικοποιηθεί με 1, ώστε εάν πατηθεί ξανά το κουμπί πριν έχει περάσει το κατάλληλο χρονικό διάστημα (Τ3), να μην γίνουν αλλαγές στα led. Στην συνέχεια, σβήνει το led για τα αμάξια, ανάβει το led των πεζών και θέτει την μεταβλητή int\_type = 1 (περισσότερα για αυτήν την μεταβλητή στην συνέχεια). Παράλληλα, ο low counter γίνεται ίσος με value2 και μετράει μέχρι το 0, ώστε μετά από χρονικό διάστημα Τ2, να κλείσει το led των πεζών και να ανοίξει των αμαξιών (Τ2 = 35/ftimer = 1,843 ms). Όταν περάσει αυτό το χρονικό διάστημα, τότε καλείται η ISR(TCA0\_LCMP0\_vect), το είδος της διακοπής είναι το πρώτο (int\_type == 1), σβήνεται το led των πεζών, ενεργοποιείται το led των αμαξιών και η μεταβλητή x γίνεται 1. Μετά, όταν επιστρέφουμε στην main, επειδή η μεταβλητή x έγινε 1, τότε πάλι ξεκινάει να μετράει ο low counter από την τιμή value3 μέχρι το 0 (Τ3 = 76/ftimer = 3.891 ms) και το int\_type γίνεται 2. Το Τ3 πρόκειται για το χρονικό διάστημα μετά την αλλαγή των φαναριών όπου δεν επιτρέπεται στους πεζούς να ξαναπατήσουν το φανάρι. Εάν πατηθεί το φανάρι ξανά πριν περάσει το χρονικό διάστημα Τ3, τότε επειδή η μεταβλητή allow\_button είναι 0, δεν θα γίνει καμία αλλαγή στα φανάρια. Όταν περάσει το Τ3, τότε ξανακαλείται η ISR(TCA0\_LCMP0\_vect) και επειδή το int\_type είναι 2, ενεργοποιείται ξανά η μεταβλητή allow\_button, που επιτρέπει το πάτημα του κουμπιού από τους πεζούς.

Η δεύτερη περίπτωση είναι αυτή του τραμ. Το τραμ περνάει κάθε φορά που ο high counter παθαίνει underflow, δηλαδή ανά Τ1 = 200/ftimer = 10,022 ms. Η ISR(TCA0\_LCMP0\_vect) κλείνει το led των αμαξιών και ανάβει του τραμ και των πεζών. Επίσης, η μεταβλητή allow\_button γίνεται 0 ενώ η μεταβλητή x γίνεται 2. Επιστρέφοντας στο while loop, επειδή η μεταβλητή x έγινε 2, τότε ξεκινάει να μετράει ο low counter μέχρι το value2, προκειμένου η κατάσταση των φαναριών να διατηρηθεί για χρονικό διάστημα Τ2, και η μεταβλητή int\_type γίνεται 3. Επίσης στον high counter τοποθετούμε την τιμή value1 ώστε να μετράει πάλι από αυτήν την τιμή προς το 0. Όταν περάσει το χρονικό διάστημα Τ2, τότε καλείται η ISR(TCA0\_LCMP0\_vect), και επειδή η int\_type είναι 3, τότε σβήνουν τα led των πεζών και τραμ και ανάβει των αμαξιών. Τέλος, όπως και στην περίπτωση των πεζών, η μεταβλητή x γίνεται 1, προκειμένου να μην είναι επιτρεπτό οι πεζοί να πατήσουν το κουμπί για να ανοίξει το φανάρι τους εάν δεν έχει περάσει χρονικό διάστημα Τ3.

A diagram of a flowchart

AI-generated content may be incorrect.

Διάγραμμα ροής λειτουργίας φαναριών.