ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 1 ΠΡΟΗΓΜΕΝΟΙ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΕΣ

Ομάδα

Ζήκος Σπύρος 1084581

Κυριακουλόπουλος Καλλίνικος 1084583

Αναλυτική Περιγραφή Κώδικα

Ορίζουμε τους χρόνους Τ1, Τ2, Τ3 που ζητούνται στην εκφώνηση. Θέσαμε αυτές τις τιμές ώστε το τραμ να περνάει λιγότερο συχνά από το χρόνο που χρειάζεται για να διασχίσει την διασταύρωση. Επίσης, ο χρόνος μεταξύ δύο διαδοχικών πατημάτων του κουμπιού είναι μεγαλύτερος από την διάρκεια του πράσινου φαναριού στους πεζούς.

Υπολογισμός χρόνου:

ftimer = 20MHz/16 = 1.25 MHz

Τ1: T = 26 / (1.25 MHz) = 20.8 μs

T2: T = 8 / (1.25 MHz) = 6.4 μs

T3: T = 16 / (1.25 MHz) = 12.8 μs

Βάλαμε μικρούς χρόνους ώστε το πρόγραμμα να εκτελείται αρκετά γρήγορα για να μην περιμένουμε πολύ για την διαδοχική εκτέλεση μεταξύ δύο breakpoints της main().

#define T1 0b00011010

#define T2 0b00001000

#define T3 0b00010000

Ορίζουμε μεταβλητές (flags) οι οποίες υποδεικνύουν αν περνάει το τραμ, αν το κουμπί των πεζών είναι ενεργό, αν το T2 μόλις πέρασε, αν το πρόγραμμα τρέχει πρώτη φορά και μία βοηθητική μεταβλητή για τα while loops.

int tram\_passing = 0;

int button\_en = 1;

int t2\_just\_passed = 1;

int first\_run = 1;

int x = 0;

int main(void)

Στην συνάρτηση αυτή κάνουμε setup τα PINs του PORTD και PORTF και του timer TCAsplit. Μετά, τρέχουμε μία φορά το low κομμάτι του timer για να τον αρχικοποιήσουμε. Ύστερα, κάνουμε πράσινο το φανάρι του αυτοκινητόδρομου και κόκκινο των πεζών. Μετά, αρχίζουμε τον high timer ο οποίος μετράει Τ1 χρόνο και μετά κάνει interrupt. Το while(x==1) χρησιμεύει στο τρέξιμο της όλης διαδικασίας και ο κώδικας δεν βγαίνει ποτέ από αυτό.

void car\_pass();

Η συνάρτηση θέτει το φανάρι των πεζών και του τραμ κόκκινο και το φανάρι των αυτοκινήτων πράσινο.

void pedestrian\_pass();

Η συνάρτηση θέτει το φανάρι των πεζών πράσινο και το φανάρι αυτοκινήτων κόκκινο.

void tram\_pass();

Η συνάρτηση θέτει το φανάρι των πεζών και του τραμ πράσινο και το φανάρι των αυτοκινήτων κόκκινο.

Πράσινο είναι το φανάρι όταν το αντίστοιχο PIN είναι 0, ενώ κόκκινο 1.

Τα PINs αντιστοιχίζονται ως εξής: PIN0->πεζοί PIN1->τραμ PIN2->αυτοκίνητα

void setup();

Mε τη συνάρτηση αυτή αρχικοποιούμε την κατεύθυνση των PINs του PORTD, τις ρυθμίσεις του TCA0 και το PIN5 του PORTF.

Επιλέγουμε για έξοδο τα PINs0-2 του PORTD για να εμφανίζεται η έξοδος μέσω των LEDs.

O TCA0 ορίζεται σε spilt mode ώστε να μπορεί να γίνει παράλληλη μέτρηση 2 διαφορετικών τιμών.

Για τη μέτρηση στα low bits επιλέγουμε την ενεργοποίηση του LCMP0 για δημιουργία interrupt όταν ο counter φτάσει στην τιμή του LCMP0 που έχει οριστεί 0. Επίσης, ο counter αρχικοποιείται στην τιμή FF(hex). Για τα high bits επιλέγουμε να προκαλείται interrupt όταν γίνει underflow. Επομένως ο TCA0 δέχεται interrupt από τον LCMP0 και HUNF.

Tέλος ορίζουμε το PIN5 από το PORTF να προκαλεί interrupt και στις 2 ακμές.

void timer\_high()

Η συνάρτηση αυτή καλείται για να θέσει το χρόνο τον οποίο θα πρέπει να μετρήσει ο timer για τα high bits. Ο χρόνος αυτός αντιπροσωπεύει το διάστημα που κάνει το τραμ από τη στιγμή που πέρασε μέχρι να ξαναπεράσει. Κατά τη διάρκεια εκτέλεσης της συνάρτησης οι διακοπές είναι απενεργοποιημένες.

void timer\_low(int choose\_counter)

Η συνάρτηση καλείται για να θέσει το χρόνο του timer για τα low bits. Αναλόγως με το όρισμα που κληθεί (2 ή 3) δίνει την κατάλληλη τιμή για μέτρηση. Ο χρόνος Τ2 αντιστοιχεί στο χρόνο που διαρκεί το πράσινο φανάρι των πεζών και στο χρόνο που κάνει να περάσει το τραμ, ενώ ο Τ3 είναι ο χρόνος που θα πρέπει να περάσει ώστε να μπορέσει το κουμπί του φαναριού να ξαναπατηθεί από τους πεζούς. Κατά τη διάρκεια εκτέλεσης της συνάρτησης οι διακοπές είναι απενεργοποιημένες.

Στον ISR(TCA0\_HUNF\_vect) καθαρίζουμε τα intflags και απενεργοποιούμε το κουμπί των πεζών. Επίσης, ενεργοποιούμε την σημαία tram\_passing και αλλάζουμε τα PINs του PORTD με τις εντολές tram\_pass(), pedestrian\_pass() ώστε να υπάρχει 1 μόνο στο PIN2 του αυτοκινητόδρομου (δηλαδή ο αυτοκινητόδρομος έχει κόκκινο φανάρι). Τέλος, καλεί την timer\_low(2) η οποία μετράει Τ2 με τον low timer.

Στον ISR(TCA0\_LCMP0\_vect) καθαρίζουμε τα intflags. Μετά, ελέγχουμε αν η συνάρτηση καλείται για πρώτη φορά και αν ναι τότε κάνουμε το first\_run μηδέν και το x ένα ώστε να μην ξαναμπεί σε αυτό το if και να ξεκολλήσει από το πρώτο while της main. Ύστερα, ελέγχουμε αν περνάει το τραμ και

-αν ναι τότε αυτό σημαίνει ότι το interrupt προκλήθηκε επειδή τελείωσε το πέρασμα του τραίνου. Γι’ αυτό, αλλάζουμε κατάλληλα τα PINs με το car\_pass(). Τέλος, θέτουμε πάλι τον timer high σε T1 ώστε να μετρήσει πότε θα ξαναπεράσει το τρένο.

-Αν το tram\_passing δεν είναι 1 τότε το interrupt προκλήθηκε από το κουμπί και

-αν το t2\_just\_passed είναι 1 τότε αυτό σημαίνει ότι προκλήθηκε interrupt λόγω του τέλους της μέτρησης του Τ2 και άρα πρέπει να μετρήσουμε Τ3. Έτσι, κάνουμε το t2\_just\_passed μηδέν γιατί πρέπει να μετρήσουμε Τ3 και όταν ξαναμπούμε στην ISR(TCA0\_LCMP0\_vect) θα έχει τελειώσει η μέτρηση του Τ3 και θα ενεργοποιήσουμε το κουμπί.

-αν το t2\_just\_passed είναι 0 τότε το interrupt προήλθε από το τέλος της μέτρησης του Τ3 και ενεργοποιούμε το κουμπί.

Ο ISR(PORTF\_PORT\_vect) καλείται όταν προκληθεί interrupt από το PIN5 του PORTF, δηλαδή το PIN5 γίνει 1. Στην αρχή καθαρίζονται τα interrupt flags και στη συνέχεια αν έχει ενεργοποιηθεί το κουμπί του φαναριού από τους πεζούς θα καθαρίσει το flag που δηλώνει πως πατήθηκε το κουμπί, θα καλέσει τη συνάρτηση pedestrian\_pass() για να τεθούν τα φανάρια στις κατάλληλες καταστάσεις και τέλος θα κληθεί η συνάρτηση timer\_low() για να ξεκινήσει η μέτρηση για το Τ2.

ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ

A diagram of a computer

Description automatically generated

Using: https://app.diagrams.net

Κώδικας

// Zikos Spiridon 1084581

// Kyriakoulopoulos Kallinikos 1084583

// Group A4

// Time spent: 18 hours (No kidding)

#include <avr/io.h>

#include <avr/interrupt.h>

#define T1 0b00011010

#define T2 0b00001000

#define T3 0b00010000

void car\_pass();

void pedestrian\_pass();

void setup();

void timer\_high();

void timer\_low(int choose\_counter);

int tram\_passing = 0;

int button\_en = 1;

int t2\_just\_passed = 1;

int first\_run = 1;

int x = 0;

int main(void){

setup(); // setup PORTD pins, TCA split, PORTF pin5 interrupt

sei(); // enable interrupts

timer\_low(2); // run timer for the first time to fix issues related to first run of the timer

while(x==0){} // wait for T2 to pass

car\_pass();

// initializes the tram

// remove that line and the tram is disabled

timer\_high();

while (x==1) {} // needs a breakpoint to see stuff

}

void car\_pass() {

PORTD.OUT |= 0b00000001; // pedestrian not pass

PORTD.OUT |= 0b00000010; // tram not pass

PORTD.OUTCLR = 0b00000100; // car pass

}

void pedestrian\_pass() {

PORTD.OUT |= 0b00000100; // car not pass

PORTD.OUTCLR = 0b00000001; // pedestrian pass

}

void tram\_pass() {

PORTD.OUTCLR = 0b00000010; // tram pass

}

void setup(){

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Output LEDs \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//pin0->pedestrian pin1->tram pin2->car

PORTD.DIR |= 0b00000111; //PINS 0&1&2 is output

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Timers Setup \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

TCA0.SPLIT.CTRLA = TCA\_SPLIT\_CLKSEL\_DIV16\_gc | TCA\_SPLIT\_ENABLE\_bm;

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

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

TCA0.SPLIT.LCMP0 = 0;

TCA0.SPLIT.LCNT = 0xFF; // Initialize counter

TCA0.SPLIT.INTCTRL = TCA\_SPLIT\_HUNF\_bm | TCA\_SPLIT\_LCMP0\_bm;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Button \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

}

void timer\_high() {

cli();

TCA0.SPLIT.HCNT = T1;

sei();

}

void timer\_low(int choose\_counter) { // flag: 2 or 3

cli();

TCA0.SPLIT.LCNT = (choose\_counter==2)?T2:T3;

sei();

}

/\* T1 timer interrupt \*/

ISR(TCA0\_HUNF\_vect){

cli();

int intflags = TCA0.SPLIT.INTFLAGS; //Procedure to

TCA0.SPLIT.INTFLAGS=intflags; //clear interrupt flag

button\_en = 0;

tram\_passing = 1;

tram\_pass();

pedestrian\_pass();

timer\_low(2);

sei();

}

/\* T2/T3 timer interrupt \*/

ISR(TCA0\_LCMP0\_vect){

cli();

int intflags = TCA0.SPLIT.INTFLAGS; //Procedure to

TCA0.SPLIT.INTFLAGS=intflags; //clear interrupt flag

if (first\_run) {

first\_run = 0;

x = 1;

return;

}

if (tram\_passing) {

tram\_passing = 0;

button\_en = 1;

car\_pass();

timer\_high();

} else {

if (t2\_just\_passed) {

t2\_just\_passed = 0;

car\_pass();

timer\_low(3);

} else {

t2\_just\_passed = 1;

button\_en = 1;

}

}

sei();

}

/\* Button \*/

ISR(PORTF\_PORT\_vect){

int y = PORTF.INTFLAGS; //Procedure to

PORTF.INTFLAGS=y; //clear the interrupt flag

if (button\_en) {

button\_en = 0;

pedestrian\_pass();

timer\_low(2);

}

}