

Πανεπιστήμιο Πατρών

Τμήμα Μηχανικών Η/Υ και Πληροφορικής

**Εργαστηριακή Άσκηση 3**

Συνεργάτες:

Θεόφραστος Παξιμάδης 1093460

Κωνσταντίνος Αναστασόπουλος 1093320

Τμήμα: Β4

Ομάδα: 1

Μέρος 1: Κώδικας

/\*

\* lab3.c

\*

\* Created: 20/03/2025 16:23:54

\* Author : Theo & Dean

\*/

#include <avr/io.h>

#include <avr/interrupt.h>

//counter thresholds

#define value1 20

#define value2 40

#define value3 60

//distance sensor threshold

#define DISTANCE\_1 8

#define DISTANCE\_2 9

int x; //loop control flag

int z; //ADC ISR control flag

int inverted = 0; //clockwise or counter clockwise indicator

int left = 0; //left turn counter

int right = 0; //right turn counter

int main() {

//led setup

PORTD.DIR |= PIN0\_bm | PIN1\_bm | PIN2\_bm; //first three pins outputs

PORTD.OUT |= PIN0\_bm | PIN2\_bm; //PIN0 and PIN2 leds initially turned off

PORTD.OUT &= ~PIN1\_bm; //PIN1 led initially turned on

//ADC setup

ADC0.CTRLA |= ADC\_RESSEL\_10BIT\_gc | ADC\_ENABLE\_bm; //10-bit resolution and enable ADC

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc; //bit 7 analog input

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //enable Debug Mode

ADC0.CTRLA |= ADC\_FREERUN\_bm; //free-running mode enabled

ADC0.WINLT |= DISTANCE\_1; //set threshold for forward sensor

ADC0.WINHT |= DISTANCE\_2; //set threshold for side sensor

ADC0.INTCTRL |= ADC\_WCMP\_bm; //enable interrupts for WCM

ADC0.CTRLE = ADC\_WINCM\_ABOVE\_gc; //interrupt when RESULT > WINLT

//counter setup

TCA0.SINGLE.CNT = 0; //clear counter

TCA0.SINGLE.CTRLB = TCA\_SINGLE\_WGMODE\_NORMAL\_gc; //normal mode

TCA0.SINGLE.CTRLA = TCA\_SINGLE\_CLKSEL\_DIV1024\_gc | TCA\_SINGLE\_ENABLE\_bm; //prescaler value 1024 and counter enable

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_CMP0\_bm; //enable compare channel 0 interrupt

ADC0.COMMAND |= ADC\_STCONV\_bm; //start conversion

//button setup

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

sei(); // start accepting interrupts

x = 1; //side sensor works first (insert breakpoint)

while(1) { // main loop

if (x == 1) { //side sensor

ADC0.CTRLA &= ~ADC\_FREERUN\_bm; // Disable free run, enable single

TCA0.SINGLE.CNT = 0; //clear counter

TCA0.SINGLE.CMP0 = value1; //assign value1 to the counter compare

z = 1; //side sensor part of the ISR should run

x = 0; //don't use the ADC until interrupt

ADC0.CTRLE = ADC\_WINCM\_ABOVE\_gc; //interrupt when RESULT > WINLT (insert breakpoint)

}

if (x == 2) { //forward sensor

ADC0.CTRLA |= ADC\_FREERUN\_bm; //Free-Running mode enabled

ADC0.COMMAND |= ADC\_STCONV\_bm; //start conversion

TCA0.SINGLE.CNT = 0; //clear counter

TCA0.SINGLE.CMP0 = value2; //assign value1 to the counter compare

z = 2; //forward sensor part of the ISR should run

x = 0; //don't use the ADC until interrupt

ADC0.CTRLE = ADC\_WINCM\_BELOW\_gc; //interrupt when RESULT < WINLT (insert breakpoint)

}

if ( (left == (right + 4)) || (right == (left + 4))) //termination condition if not inverted

break; //terminate

if ((inverted == 1) && ((right == -1) || (left == -1))) //termination condition if inverted

break; //terminate

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

}

ADC0.CTRLA = 0; // disable ADC0

}

ISR(ADC0\_WCOMP\_vect) { //ADC interrupt service routine

if (z == 1 && inverted == 0) { //turn right

PORTD.OUT |= PIN1\_bm; //turn off led 1

PORTD.OUT &= ~PIN0\_bm; //turn on led 0

right = right + 1; //CHANGE VALUE (insert breakpoint)

PORTD.OUT |= PIN0\_bm; //turn off led 0

PORTD.OUT &= ~PIN1\_bm; //turn on led 1

}

if (z == 2 && inverted == 0) { //turn left

PORTD.OUT |= PIN1\_bm; //turn off led 1

PORTD.OUT &= ~PIN2\_bm; //turn on led 2

left = left + 1; //CHANGE VALUE (insert breakpoint)

PORTD.OUT |= PIN2\_bm; //turn off led 2

PORTD.OUT &= ~PIN1\_bm; //turn on led 1

}

if ((inverted == 1) && (z == 1)) { //inverted turn left

PORTD.OUT |= PIN1\_bm; //turn off led 1

PORTD.OUT &= ~PIN2\_bm; //turn on led 2

right = right - 1; // CHANGE VALUE (insert breakpoint)

PORTD.OUT |= PIN2\_bm; //turn off led 2

PORTD.OUT &= ~PIN1\_bm; //turn on led 1

}

if ((inverted == 1) && (z == 2)) { //inverted turn right

PORTD.OUT |= PIN1\_bm; //turn off led 1

PORTD.OUT &= ~PIN0\_bm; //turn on led 0

left = left - 1; // CHANGE VALUE (insert breakpoint)

PORTD.OUT |= ~PIN0\_bm; //turn off led 0

PORTD.OUT &= PIN1\_bm; //turn on led 1

}

int intflags = ADC0.INTFLAGS; //reset the interrupt flags of the ADC

ADC0.INTFLAGS = intflags;

}

ISR(TCA0\_CMP0\_vect) { //timer interrupt service routine

if (z == 1) { //side sensor was used

x = 2; // forward sensor is about to be used

ADC0.CTRLE = 0b00000000; // disable ADC comparisons (insert breakpoint)

}

if (z == 2) { //forward sensor was used

x = 1; // side sensor is about to be used

ADC0.CTRLE = 0b00000000; // disable ADC comparisons (insert breakpoint)

}

if (inverted == 2) { // the device is about to be inverted

inverted = 1; //the device is inverted

PORTD.OUT |= PIN0\_bm | PIN2\_bm; //led 0 and 2 are turned off

x = 1; //side sensor is about to be used

}

int intflags = TCA0.SINGLE.INTFLAGS; //reset the interrupt flags of the counter

TCA0.SINGLE.INTFLAGS=intflags;

}

ISR(PORTF\_PORT\_vect) {

int y = PORTF.INTFLAGS; //reset the interrupt flags of the buttons

PORTF.INTFLAGS=y;

inverted = 2; //the device is about to be inverted (insert breakpoint)

x = 0; //don't use the ADC until interrupt

z = 0; //don't use the ADC until interrupt

PORTD.OUT &= ~PIN0\_bm & ~PIN1\_bm & ~PIN2\_bm; //the first three leds are turned on

TCA0.SINGLE.CNT = 0; //clear counter

TCA0.SINGLE.CMP0 = value3; //assign value3 to the counter compare

ADC0.CTRLE = 0b00000000; // disable ADC comparisons (insert breakpoint)

}

Μέρος 2: Επεξήγηση Κώδικα & Διάγραμμα Ροής

Το πρώτο μέρος του κώδικα είναι η αρχικοποίηση των led, ADC, timer και setup. Πιο συγκεκριμένα, τα led0, led1, led2 ορίζονται σαν έξοδο και αρχικά μόνο το led1 είναι ανοιχτό που υποδηλώνει την ευθεία πορεία. Έπειτα, αρχικοποιείται ο ADC0 με τις εντολές της εκφώνησης και ορίζονται τα ADC0.WINLT και ADC0.WINHT, που θα είναι τα δύο thresholds του αισθητήρα για μπροστά και πλάγια αντίστοιχα. Τέλος, ο timer TCA ορίζεται για single mode και επιτρέπονται οι διακοπές από το pin5 του portf.

Στον βρόγχο while που ακολουθεί υπάρχει το κυρίως μέρος του κώδικα. Ειδικότερα, αρχικά η μεταβλητή x είναι 1 και εξετάζουμε τον πλαϊνό sensor για Τ1 = value1/ftimer = 20/ftimer = 1,024 ms. Επειδή θέλουμε single conversion και όχι free run, απενεργοποιούμε το bit 1 του control A (ADC0.CTRLA &= ~ADC\_FREERUN\_bm;). Η global μεταβλητή z γίνεται 1 και επίσης ορίζουμε το ADC0.CTRLE = ADC\_WINCM\_ABOVE\_gc;, γιατί θέλουμε να γίνει διακοπή αν η τιμή του καταχωρητή Result ξεπεράσει την τιμή του threshold που έχουμε ορίσει. Εάν αυτό γίνει, τότε οδηγούμαστε στην ISR(ADC0\_WCOMP\_vect) και επειδή πρόκειται για τον πλαϊνό αισθητήρα και η συσκευή εκτελεί κανονική και όχι αντίθετη πορεία (z == 1 & inverted == 0), τότε ανοίγει στιγμιαία το led0 και αυξάνεται ο global μετρητής right κατά 1 (οι global μετρητές right και left χρησιμοποιούνται για να καταγράφονται οι περιστροφές της συσκευής). Όταν τελειώσει το Τ1, καλείται η ISR(TCA0\_CMP0\_vect), ορίζεται το x = 2 ώστε στην συνέχεια να εξεταστεί ο μπροστινός αισθητήρας. Επιπλέον, απενεργοποιούνται οι συγκρίσεις με τον καταχωρητή result (ADC0.CTRLE = 0b00000000;) ώστε να μην υπάρξει κάποια ανεπιθύμητη επιπλέον διακοπή κατά την αλλαγή των αισθητήρων.

Στην συνέχεια που το x = 2, εξετάζεται ο μπροστινός αισθητήρας. Ομοίως με πριν, η μεταβλητή z γίνεται 2 και για Τ2 = value2/ftimer = 2,048 ms, εάν ο καταχωρητής result έχει μικρότερη τιμή από το πρώτο threshold (ADC0.CTRLE = ADC\_WINCM\_BELOW\_gc;, ώστε διακοπή όταν RESULT < WINLT), τότε καλείται πάλι η ISR(ADC0\_WCOMP\_vect). Επίσης, ξανα-ενεργοποιείται το conversion (ADC0.COMMAND |= ADC\_STCONV\_bm;), επειδή απενεργοποιήθηκε στο τέλος του πλαϊνού αισθητήρα εξαιτίας του single conversion. Στην ISR(ADC0\_WCOMP\_vect), λόγο των μεταβλητών z και inverted, ανάβει στιγμιαία το led2 που υποδηλώνει την αριστερή στροφή και αυξάνεται ο μετρητής left. Τέλος, καλείται η ISR(TCA0\_CMP0\_vect) και η μεταβλητή x γίνεται 1, ώστε να εξεταστεί ο πλαϊνός αισθητήρας. Αυτή είναι η επαναληπτική διαδικασία που εκτελείται συνέχεια.

Η συσκευή θα έχει φτάσει στην αρχική της θέση όταν έχει εκτελέσει περιστροφή 360ο περισσότερες στην μία κατεύθυνση από ότι στην άλλη. Για παράδειγμα μπορεί είτε να έχει κάνει 4 αριστερές στροφές ή 1 δεξιά και 5 αριστερές. Για αυτόν τον λόγο, στην while ελέγχονται οι κατάλληλες συνθήκες ( if ( (left == (right + 4)) || (right == (left + 4))) break; ), ώστε το πρόγραμμα να απενεργοποιεί τον ADC0 και να τερματίζει εάν κάποια από αυτές ισχύει.

Για την αντίστροφη πορεία, πρέπει να ενεργοποιηθεί το pin5 του portf και να κληθεί η ISR(PORTF\_PORT\_vect). Σύμφωνα με την εκφώνηση, ανάβουν τα τρια led για Τ3 = value3/ftimer και όταν περάσει αυτό το χρονικό διάστημα, καλείται η ISR(TCA0\_CMP0\_vect) η οποία επειδή η μεταβλητή inverted έγινε 2 στην ISR(PORTF\_PORT\_vect), τώρα γίνεται 1 και μπορεί να εκτελεστεί η αντίστροφη πορεία. Η αντίστροφη πορεία εκτελεί τις ίδιες ενέργειες με την αρχική, αλλά τώρα κάθε φορά που η συσκευή κάνει μια δεξιά, για παράδειγμα, περιστροφή, τότε στην ISR(ADC0\_WCOMP\_vect) μειώνεται ο μετρητής left κατά ένα (ομοίως για τις αριστερές). Αυτό σημαίνει ότι όταν οι μεταβλητές left και right γίνουν 0, τότε η συσκευή έχει αναιρέσει τις περιστροφές που έκανε στην κανονική της πορεία και όταν φτάσει στην αρχική της θέση, θα κάνει μία απόπειρα για μία ακόμα περιστροφή, ο μετρητής left ή right θα γίνει -1, και τότε το πρόγραμμα θα τερματίσει (if ((inverted == 1) && ((right == -1) || (left == -1))) break;).

A diagram of a flowchart

AI-generated content may be incorrect.

Διάγραμμα ροής λειτουργίας συσκευής.