**Προηγμένοι Μικροεπεξεργαστές**

**Εργαστηριακή Άσκηση 4**

**Μέλη Ομάδας**

**1ο μέλος: 2ο μέλος:**

**ΟΝΟΜΑ: ΛΟΥΚΑΣ ΟΝΟΜΑ: ΕΥΑΓΓΕΛΟΣ**

**ΕΠΩΝΥΜΟ: ΤΡΙΑΝΤΑΦΥΛΛΟΠΟΥΛΟΣ ΕΠΩΝΥΜΟ: ΜΠΕΝΕΤΟΣ**

**ΕΞΑΜΗΝΟ: 8 ΕΞΑΜΗΝΟ: 10**

**ΑΜ: 1084495 ΑΜ: 1072628**

1) Στην συγκεκριμένη άσκηση μας ζητείται να κατασκευάσουμε ένα έξυπνο θερμοκήπιο, μέσω του προσομοιωτή, και σε αυτό πρέπει να εκτελείται ένα σύνολο από διακριτές λειτουργίες. Για το πρώτο ερώτημα, έχουμε ως ζητούμενο το θερμοκήπιό μας να μπορεί να ανιχνεύσει εάν επικρατεί χαμηλή ή υψηλή υγρασία εντός αυτού. Για να το επιτύχουμε αυτό, κάναμε χρήση του μετατροπέα αναλογικού σε ψηφιακό σήμα (ADC), δύο κατώφλια (ένα κάτω κι ένα άνω) και δύο λαμπάκια LED για ειδοποίηση ως προς τον χρήστη για το ποια λειτουργία πρέπει να εφαρμοστεί αφού ανάψει το ανάλογο λαμπάκι.

Στο κώδικα που παραθέτουμε για το πρώτο ερώτημα ορίζουμε δύο μεταβλητές, τις plant\_hydrate και airconditioning, τύπου ακεραίου, όπου αντιστοιχούν στα δύο κατώφλια. Η plant\_hydrate αντιστοιχεί στο κατώτερο κατώφλι κι αν δοθεί μικρότερη τιμή, μέσω του καταχωρητή RES του ADC, τότε θα πρέπει να ανάψει το LED0 που θα ειδοποιεί πως πρέπει να γίνει η ενέργεια του ποτίσματος. Ομοιώς για την airconditioning, αν δοθεί τιμή μεγαλύτερη αυτης, θα πρέπει να ανάψει το LED1 για να προβούμε στην ενέργεια ενεργοποίησης του συστήματος αερισμού. Επιπλέον θέτουμε δύο βοηθητικές μεταβλητές, τύπου bool, τις οποίες θα τις αξιοποιήσουμε για τις δομές επανάληψης while εντός της main. Τέλος, κάνουμε χρήσης της συνάρτησης ISR(ADC0\_WCOMP\_vect), στην οποία θα ανάβει το αντίστοιχο LED, ανάλογα με τι τιμή εισήχθη μέσω του καταχωρητή RES.

Με βάση όλες τις λεπτομέρειες που αναφέρθηκαν παραπάνω, στην συνέχεια παραθέτουμε τον κώδικα που εκτελεί το πρώτο ζητούμενο για την άσκηση:

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

int plant\_hydrate = 10; //Low threshold for watering

int airconditioning = 80; //High threshold for airconditioning

bool aux\_flag = true; //auxiliary flag for the loop

bool aux\_flag2 = false; //similar to a nop function

int main()

{

PORTD.DIR |= PIN0\_bm; //Set PIN0 as output for low humidity

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

PORTD.DIR |= PIN1\_bm;// Set PIN1 as output for high humidity

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

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //PIN5 is for the starting the watering the plant

PORTF.PIN6CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //PIN6 is for the starting the airconditioning

//initialize the ADC for Free-Running mode

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

ADC0.CTRLA |= ADC\_FREERUN\_bm; //ADC Freerun mode bit mask

//Window Comparator Mode

ADC0.WINHT |= airconditioning; //Set threshold for high humidity

ADC0.WINLT |= plant\_hydrate; //Set threshold for low humidity

while(aux\_flag == true)

{

sei();

ADC0.CTRLA |= ADC\_ENABLE\_bm; //ADC Enable bit mask

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc; //Analog Channel Selection Bits select

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Debug run bit mask

ADC0.INTCTRL |= ADC\_WCMP\_bm; //Enable Interrupts for WCM

ADC0.CTRLE = 0x4; //Interrupt when RESULT < WINLT or RESULT > WINHT

ADC0.COMMAND |= ADC\_STCONV\_bm; //Start Conversion

while(aux\_flag2 == false)

{

;//similar to a nop function

}

aux\_flag2 = false;

}

}

ISR(ADC0\_WCOMP\_vect)

{

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

ADC0.CTRLA = 0;

if(ADC0.RES < plant\_hydrate)

{

PORTD.OUTCLR = PIN0\_bm; //LED0 for watering is on

}

if(ADC0.RES > airconditioning)

{

PORTD.OUTCLR = PIN1\_bm; //LED1 for airconditioning is on

}

aux\_flag2 = true;

}

2) Για το επόμενο ερώτημα θα προσθέσουμε δύο χρονιστές τύπου TCA0, έναν για το πότισμα λόγω της χαμηλής υγρασίας κι έναν για τον αερισμό λόγω της υψηλής υγρασίας. Επιπροσθέτως θα εισάγουμε και ένα τρίτο λαμπάκι, το LED2, το οποίο θα αναβοσβήνει μέσω του PWM παλμού. Επίσης θα προστεθούν οι δύο διακόπτες, οι οποίοι θα εφαρμόζουν τις δύο ζητούμενες λειτουργίες. Στον κώδικα που θα δείξουμε στην συνέχεια, ορίζουμε κάποιες επιπλέον μεταβλητές που θα βοηθήσουν στην υλοποίηση των λειτουργιών. Η πρώτη είναι η hydrate\_time, τύπου ακεραίου, και θα ισούται με την διαφορά του κατώτερου κατωφλίου με την τρέχουσα τιμή του καταχωρητή RES. Η μεταβλητή αυτή θα χρησιμοποιηθεί ως χρονική διάρκεια από τον χρονιστή TCA0 για το πότισμα. Η δεύτερη μεταβλητή που προστίθεται, είναι η button\_value και σε αυτήν αποθηκεύεται η τιμή του καταχωρητή INTFLAGS από το PORTF πριν γίνει reset από την ανάλογη ISR συνάρτηση, καθώς θα την χρειαστούμε για τις συνθήκες if για την εκάστοτε περίπτωση. Η τρίτη μεταβλητή, η rising\_edges τύπου ακεραίου, θα χρησιμοποιηθεί ως μετρητής και θα αυξάνεται κατά ένα κάθε φορά που θα εισέρχεται στην ISR(TCA0\_OVF\_vect). Κάθε φορά που θα αυξάνεται αυτός ο μετρητής, θα αναβοσβήνει το LED2 μέχρις ότου να φτάσει τον συνολικό αριθμό των τεσσάρων ανερχόμενων ακμών.

Για την main δομή, παίρνουμε την αρχική μορφή από το πρώτο ερώτημα και προσθέτουμε την διαφορά hydrate\_time που θα αξιοποιηθεί ως χρονική περίοδος για το πότισμα. Στην συνέχεια θέτουμε την συνθήκη if(button\_value == 0x20), δηλαδή αν ο διακόπτης που έχει ενεργοποιηθεί είναι το switch5, τότε πρέπει να λειτουργήσει ο χρονιστής TCA0 με χρονική περίοδο hydrate\_time. Η δεύτερη συνθήκη είναι η if(button\_value == 0x40),δηλαδή αν ο διακόπτης που έχει ενεργοποιηθεί είναι το switch6 τότε πρέπει, ανάλογα με το τι τιμή έχει ο μετρητής rising\_edges, να αναβοσβήνει το LED2. Στο τέλος του κώδικα προστίθενται οι δύο ISR που αντιστοιχούν στους δύο χρονιστές και η ISR για τους διακόπτες.

Λαμβάνοντας υπόψιν τις αλλαγές και τις προσθήκες που έγιναν στον αρχικό κώδικα, παρακάτω δίνουμε τον ανανεωμένο κώδικα με τις επιπλέον λειτουργίες που ζητήθηκαν από το δεύτερο ερώτημα:

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

int plant\_hydrate = 10; //Low threshold for watering

int airconditioning = 80; //High threshold for airconditioning

bool aux\_flag = true; //auxiliary flag for the loop

bool aux\_flag2 = false; //similar to a nop function

bool aux\_flag3 = false; //similar to a nop function

bool aux\_flag4 = false; //similar to a nop function

int hydrate\_time; // this variable ==> plant\_hydrate - ADCO.RES (current value of RESULT register which is lesser than WINLT)

int button\_value; // keeping the current value of button to see which button is pressed

int rising\_edges; // we want the counter of rising edges after switch6 is pressed

int main()

{

PORTD.DIR |= PIN0\_bm; //Set PIN0 as output for low humidity

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

PORTD.DIR |= PIN1\_bm;// Set PIN1 as output for high humidity

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

PORTD.DIR |= PIN2\_bm;// Set PIN2 as output for airconditioning

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

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //PIN5 is for the starting the watering the plant

PORTF.PIN6CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //PIN6 is for the starting the airconditioning

//initialize the ADC for Free-Running mode

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

ADC0.CTRLA |= ADC\_FREERUN\_bm; //ADC Freerun mode bit mask

//Window Comparator Mode

ADC0.WINHT |= airconditioning; //Set threshold for high humidity

ADC0.WINLT |= plant\_hydrate; //Set threshold for low humidity

while(aux\_flag == true)

{

sei();

ADC0.CTRLA |= ADC\_ENABLE\_bm; //ADC Enable bit mask

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc; //Analog Channel Selection Bits select

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Debug run bit mask

ADC0.INTCTRL |= ADC\_WCMP\_bm; //Enable Interrupts for WCM

ADC0.CTRLE = 0x4; //Interrupt when RESULT < WINLT or RESULT > WINHT

ADC0.COMMAND |= ADC\_STCONV\_bm; //Start Conversion

while(aux\_flag2 == false)

{

;//similar to a nop function

}

aux\_flag2 = false;

hydrate\_time = plant\_hydrate - ADC0.RES; //The difference of Low threshold and

// the RESULT register is saved at hydrate\_time variable to be used as time value

// for the TCA0 timer if switch5 is pressed

if(button\_value == 0x20)

{

TCA0.SINGLE.CNT = 0; //Clear Counter

TCA0.SINGLE.CTRLB = TCA\_SINGLE\_WGMODE\_NORMAL\_gc; //Waveform generation mode selection ==> Normal Mode

TCA0.SINGLE.CMP0 = hydrate\_time; // watering duration

TCA0.SINGLE.CTRLA = TCA\_SINGLE\_CLKSEL\_DIV1024\_gc; //Clock Selection mode ==> System Clock/1024

TCA0.SINGLE.CTRLA |= 1; //Enable

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_CMP0\_bm; //Compare 0 Interrupt bit mask

while(aux\_flag3 == false)

{

;//similar to a nop function

}

aux\_flag3 = false;

TCA0.SINGLE.CTRLA = 0; //Disable

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

}

if(button\_value == 0x40)

{

TCA0.SINGLE.CTRLA=TCA\_SINGLE\_CLKSEL\_DIV1024\_gc;

TCA0.SINGLE.PER = 20; //1ms duration

TCA0.SINGLE.CMP0 = 10; //select the duty cycle

TCA0.SINGLE.CTRLB=TCA\_SINGLE\_WGMODE\_SINGLESLOPE\_gc; //select Single\_Slope\_PWM

while(rising\_edges < 4)

{

if(rising\_edges == 0)

{

PORTD.OUTCLR = PIN2\_bm; //LED2 for airconditioning is on

// for the first rising edge

}

if(rising\_edges == 1)

{

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

// for the second rising edge

}

if(rising\_edges == 2)

{

PORTD.OUTCLR = PIN2\_bm; //LED2 for airconditioning is on

// for the third rising edge

}

if(rising\_edges == 3)

{

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

// for the fourth rising edge

}

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_OVF\_bm; //enable interrupt Overflow

TCA0.SINGLE.CTRLA |= TCA\_SINGLE\_ENABLE\_bm; //Enable TCA

while(aux\_flag4 == false)

{

;//similar to a nop function

}

aux\_flag4 = false;

}

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

PORTD.OUT |= PIN2\_bm; //LED1 for airconditioning is off

TCA0.SPLIT.CTRLA=0; // We turn off the timer

}

rising\_edges = 0; // We reset the counter

cli();

}

}

ISR(ADC0\_WCOMP\_vect)

{

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

ADC0.CTRLA = 0;

if(ADC0.RES < plant\_hydrate)

{

PORTD.OUTCLR = PIN0\_bm; //LED0 for watering is on

}

if(ADC0.RES > airconditioning)

{

PORTD.OUTCLR = PIN1\_bm; //LED1 for airconditioning is on

}

aux\_flag2 = true;

}

ISR(TCA0\_CMP0\_vect)

{

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

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

aux\_flag3 = true; //change flag to get out of the loop

}

ISR(TCA0\_OVF\_vect)

{

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

aux\_flag4 = true;

rising\_edges++;

}

ISR(PORTF\_PORT\_vect)

{

button\_value = PORTF.INTFLAGS; //Save the value of the current pressed button

int y = PORTF.INTFLAGS; //Procedure to

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

}

3) Για το τρίτο ερώτημα θέλουμε, όταν ο χρήστης επιλέξει να εφαρμόσει λάθος λειτουργία από την επιτρεπτή, να λαμβάνει μια ένδειξη που να τον προειδοποιεί. Για να το επιτύχουμε αυτό, πήραμε τον κώδικα του προηγούμενου ερωτήματος και αλλάξαμε τις συνθήκες if. Συγκεκριμένα θέσαμε όταν ο RES είναι μικρότερος του κατώτερου κατωφλίου και η τιμή του button\_value είναι 0x20 (switch5) τότε θα ξεκινήσει το πότισμα. Αν το button\_value δεν είναι 0x20, δηλαδή (ADC0.RES < plant\_hydrate) && (button\_value != 0x20), τότε σε ένα βήμα του debugger θα ανάψουν ταυτόχρονα και τα τρία LED και το επόμενο βήμα θα σβήσουν και τα τρία συγχρόνως, σηματοδοτώντας ότι έγινε λάθος επιλογή από αυτή που πρέπει και το πρόγραμμα θα ξεκινήσει από την αρχή. Με τον ίδιο ακριβώς τρόπο διαχειριζόμαστε και το αερισμό. Αν (ADC0.RES > airconditioning) && (button\_value == 0x40), δηλαδή αν η τιμή του καταχωρητή RES μεγαλύτερη του μεγαλύτερου κατωφλίου και ταυτόχρονα έχει ενεργοποιηθεί ο διακόπτης 6, τότε θα πρόγραμμα σε διαδικασία αερισμού. Αν ο διακόπτης 6 δεν έχει πατηθεί, δηλαδή αν (ADC0.RES > airconditioning) && (button\_value != 0x40), τότε απλά θα ανάψουν τρία LEDs σε ένα βήμα και στο επόμενο θα σβήσουν και τα τρία και το πρόγραμμα θα ξεκινήσει από την αρχή.

Επομένως, η τελική μορφή του κώδικα που υλοποιεί όλες τις λειτουργίες που ζητήθηκαν από την άσκηση, φαίνεται παρακάτω:

. #include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

int plant\_hydrate = 10; //Low threshold for watering

int airconditioning = 80; //High threshold for airconditioning

bool aux\_flag = true; //auxiliary flag for the loop

bool aux\_flag2 = false; //similar to a nop function

bool aux\_flag3 = false; //similar to a nop function

bool aux\_flag4 = false; //similar to a nop function

int hydrate\_time; // this variable ==> plant\_hydrate - ADCO.RES (current value of RESULT register which is lesser than WINLT)

int button\_value; // keeping the current value of button to see which button is pressed

int rising\_edges; // we want the counter of rising edges after switch6 is pressed

int main()

{

PORTD.DIR |= PIN0\_bm; //Set PIN0 as output for low humidity

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

PORTD.DIR |= PIN1\_bm;// Set PIN1 as output for high humidity

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

PORTD.DIR |= PIN2\_bm;// Set PIN2 as output for airconditioning

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

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //PIN5 is for the starting the watering the plant

PORTF.PIN6CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //PIN6 is for the starting the airconditioning

//initialize the ADC for Free-Running mode

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

ADC0.CTRLA |= ADC\_FREERUN\_bm; //ADC Freerun mode bit mask

//Window Comparator Mode

ADC0.WINHT |= airconditioning; //Set threshold for high humidity

ADC0.WINLT |= plant\_hydrate; //Set threshold for low humidity

while(aux\_flag == true)

{

sei();

ADC0.CTRLA |= ADC\_ENABLE\_bm; //ADC Enable bit mask

ADC0.MUXPOS |= ADC\_MUXPOS\_AIN7\_gc; //Analog Channel Selection Bits select

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; //Debug run bit mask

ADC0.INTCTRL |= ADC\_WCMP\_bm; //Enable Interrupts for WCM

ADC0.CTRLE = 0x4; //Interrupt when RESULT < WINLT or RESULT > WINHT

ADC0.COMMAND |= ADC\_STCONV\_bm; //Start Conversion

while(aux\_flag2 == false)

{

;//similar to a nop function

}

aux\_flag2 = false;

hydrate\_time = plant\_hydrate - ADC0.RES; //The difference of Low threshold and

// the RESULT register is saved at hydrate\_time variable to be used as time value

// for the TCA0 timer if switch5 is pressed

if((ADC0.RES < plant\_hydrate) && (button\_value == 0x20))

{

TCA0.SINGLE.CNT = 0; //Clear Counter

TCA0.SINGLE.CTRLB = TCA\_SINGLE\_WGMODE\_NORMAL\_gc; //Waveform generation mode selection ==> Normal Mode

TCA0.SINGLE.CMP0 = hydrate\_time; // watering duration

TCA0.SINGLE.CTRLA = TCA\_SINGLE\_CLKSEL\_DIV1024\_gc; //Clock Selection mode ==> System Clock/1024

TCA0.SINGLE.CTRLA |= 1; //Enable

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_CMP0\_bm; //Compare 0 Interrupt bit mask

while(aux\_flag3 == false)

{

;//similar to a nop function

}

aux\_flag3 = false;

TCA0.SINGLE.CTRLA = 0; //Disable

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

button\_value = 0;

}

else if((ADC0.RES < plant\_hydrate) && (button\_value != 0x20))

{

PORTD.OUTCLR = PIN0\_bm | PIN1\_bm | PIN2\_bm;// all LEDs are on as an indication

// that the wrong switch has been pressed

PORTD.OUT |= PIN0\_bm | PIN1\_bm | PIN2\_bm; //all LEDs are off as an indication

// that the smart garden

button\_value = 0;

}

if((ADC0.RES > airconditioning) && (button\_value == 0x40))

{

TCA0.SINGLE.CTRLA=TCA\_SINGLE\_CLKSEL\_DIV1024\_gc;

TCA0.SINGLE.PER = 20; //1ms duration

TCA0.SINGLE.CMP0 = 10; //select the duty cycle

TCA0.SINGLE.CTRLB=TCA\_SINGLE\_WGMODE\_SINGLESLOPE\_gc; //select Single\_Slope\_PWM

while(rising\_edges < 4)

{

if(rising\_edges == 0)

{

PORTD.OUTCLR = PIN2\_bm; //LED2 for airconditioning is on

// for the first rising edge

}

if(rising\_edges == 1)

{

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

// for the second rising edge

}

if(rising\_edges == 2)

{

PORTD.OUTCLR = PIN2\_bm; //LED2 for airconditioning is on

// for the third rising edge

}

if(rising\_edges == 3)

{

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

// for the fourth rising edge

}

TCA0.SINGLE.INTCTRL = TCA\_SINGLE\_OVF\_bm; //enable interrupt Overflow

TCA0.SINGLE.CTRLA |= TCA\_SINGLE\_ENABLE\_bm; //Enable TCA

while(aux\_flag4 == false)

{

;//similar to a nop function

}

aux\_flag4 = false;

button\_value = 0;

}

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

PORTD.OUT |= PIN2\_bm; //LED1 for airconditioning is off

TCA0.SPLIT.CTRLA=0; // We turn off the timer

}

else if((ADC0.RES > airconditioning) && (button\_value != 0x40))

{

PORTD.OUTCLR = PIN0\_bm | PIN1\_bm | PIN2\_bm;// all LEDs are on as an indication

// that the wrong switch has been pressed

PORTD.OUT |= PIN0\_bm | PIN1\_bm | PIN2\_bm; //all LEDs are off as an indication

// that the smart garden

button\_value = 0;

}

rising\_edges = 0; // We reset the counter

cli();

}

}

ISR(ADC0\_WCOMP\_vect)

{

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

ADC0.CTRLA = 0;

if(ADC0.RES < plant\_hydrate)

{

PORTD.OUTCLR = PIN0\_bm; //LED0 for watering is on

}

if(ADC0.RES > airconditioning)

{

PORTD.OUTCLR = PIN1\_bm; //LED1 for airconditioning is on

}

aux\_flag2 = true;

}

ISR(TCA0\_CMP0\_vect)

{

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

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

aux\_flag3 = true; //change flag to get out of the loop

}

ISR(TCA0\_OVF\_vect)

{

int intflags = TCA0.SINGLE.INTFLAGS;

TCA0.SINGLE.INTFLAGS = intflags;

aux\_flag4 = true;

rising\_edges++;

}

ISR(PORTF\_PORT\_vect)

{

button\_value = PORTF.INTFLAGS; //Save the value of the current pressed button

int y = PORTF.INTFLAGS; //Procedure to

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

}





