**Εργαστήριο Προηγμένοι Μικροεπεξεργαστές**

Εργαστηριακή Άσκηση 03:

Έξυπνη Οικιακή Συσκευή, με Κίνηση στο Χώρο

2024-25

**Περιεχόμενα**

[Σχεδιασμός / Διάγραμμα Ροής 3](#__RefHeading___Toc1188_1975538019)

[Περιγραφή Κώδικα – Σχεδιαστική ιδέα 4](#__RefHeading___Toc30_1975538019_Copy_1)

[Global: 5](#__RefHeading___Toc666_1975538019)

[main: 6](#__RefHeading___Toc666_1975538019_Copy_1)

[ISR(ADC0\_RESRDY\_vect): 8](#__RefHeading___Toc2469_2567516344)

[ISR(ADC0\_WCOMP\_vect): 8](#__RefHeading___Toc2471_2567516344)

[ISR(PORTF\_PORT\_vect): 9](#__RefHeading___Toc2473_2567516344)

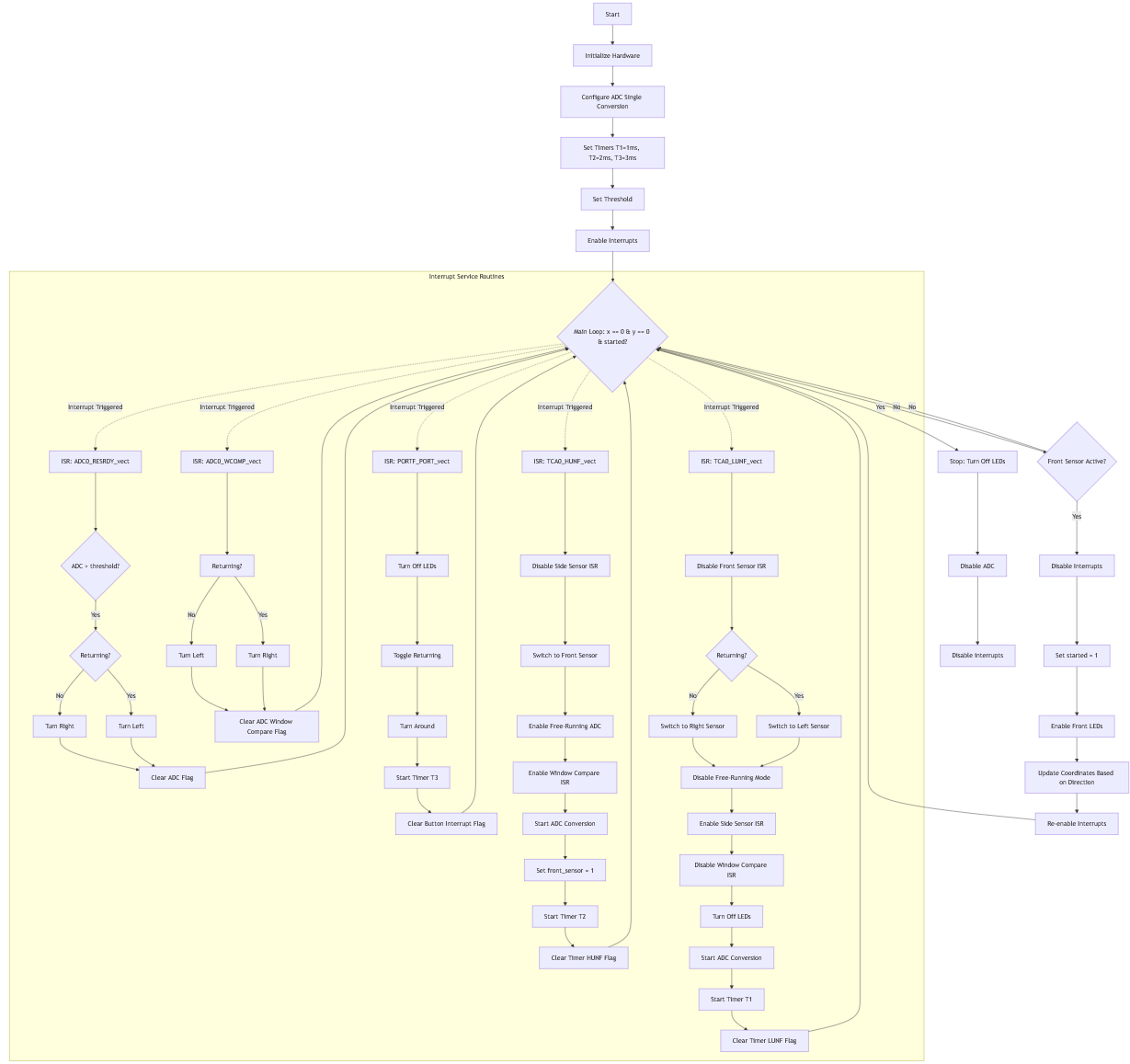
[ISR(TCA0\_LUNF\_vect): 9](#__RefHeading___Toc2475_2567516344)

[ISR(TCA0\_LUNF\_vect): 10](#__RefHeading___Toc2477_2567516344)

[Κώδικας: 11](#__RefHeading___Toc2477_2567516344_Copy_1)

# Σχεδιασμός / Διάγραμμα Ροής

Το διάγραμμα δημιουργήθηκε μεσω mermaid.js



# Περιγραφή Κώδικα – Σχεδιαστική ιδέα

Ολόκληρος ο κώδικας ειναι διαθέσιμος στο τέλος της αναφοράς.

Ο επεξεργαστής τρέχει στα 20MHz (Το Default του Microchip ειναι 1MHz).

Καλούμαστε να υλοποιήσουμε την λειτουργία μιας οικιακής συσκευής *(στην αναφορά μας αποκαλείται ρομπότ για ευαναγνωσιμότητα)* η οποία ξεκινάει απο μία γωνία ενός δωματίου και σχεδιάζει το περίγραμμά του. **Ζητείται να λειτουργεί με ένα απλό τετράγωνο δωμάτιο και με ενα τυχαίο δωμάτιο που περιέχει δύο αμβλείες γωνίες.**

Στην υλοποίηση χρησιμοποιούμε 2 ISR για τον ADC (ADC0\_RESRDY\_vect / ADC0\_WCOMP\_vect). Οταν λειτουργεί ο πλαϊνός σένσορας απενεργοποιείται το interrupt του WCOMP και ενεργοποιείται του RESRDY. Αντίστροφα, οταν λειτουργεί ο μπροστά σένσορας, απενεργοποιείται το ιnterrupt του RESRDY και ενεργοποιείται το WCOMP.

Με αυτό τον τρόπο ο κώδικας γίνεται πιο ευέλικτος, καθώς η κάθε ISR αναλαμβάνει μόνο ένα είδος σένσορα, αλλά και πιο ευανάγνωστος (εφόσον η διαχείρηση των 2 αισθητήρων είναι σε δύο ξεχωριστά κομμάτια του εκτελέσιμου).

Το ρομπότ λειτουργεί βάση συντεταγμένων που ανανεώνονται σε κάθε κύκλο που μετακινείται το ρομποτ.

Με αυτό τον τρόπο διαβεβαιώνεται η σωστή λειτουργία του για **όλα**  τα είδη δωματίων, όχι μόνο για τετράγωνα ή με 2 αμβλείες, και σε **όλες** τις περιπτώσεις.

**Καθώς οι συντεταγμένες ανανεώνονται σε κάθε κύκλο ρολογιού, υπάρχει προϋπόθεση το δωμάτιο να μην είναι απίθανα μικρό ( Tcomplete >> T1 + T2 + T3).**

**Υποθέτουμε οτι αυτό ισχύει.**

*Γνωρίζουμε πως προτείνεται η χρήση 2 threshold και ενός ISR για τον έλεγχο. Θεωρούμε την υλοποίηση μας ισοδύναμη, καθώς δεν μπορούμε να σκεφτούμε παραπάνω αδυναμίες συγκριτικά με την προτεινούμενη.*

## Global:

Θέτουμε τιμές για τους χρόνους Τ1 Τ2 Τ3 και το threshold που θα χρησιμοποιούν οι αισθητήρες (οι πλαϊνοι και ο μπροστά χρησιμοποιούν το ίδιο threshold, καθώς θέλουμε να είναι ομοιόμορφη η απόσταση του ρομπότ απο τις άκρες του τοίχου.

#define T2 39 // T = 39 / 19.5 = 2ms

#define T1 19.5 // T = 19.5 / 19.5 = 1ms

#define T3 58.5 // T = 58.5 / 19.5 = 3ms

#define threshold 25

Αρχικοποιούμε 2 μεταβλητές τύπου long int και μία int, οι οποίες θα κρατούν τις συντεταγμένες του ρομπότ και την κατεύθυνση του αντίστοιχα.

volatile long int x = 0; // Current X coordinate

volatile long int y = 0; // Current Y coordinate

volatile int current\_dir = 0; // 0=north, 1=west, 2=south, 3=east

Ορίζουμε μεταβλητές που λειτουργούν σαν σημαίες για έλεγχο:

1. Εκκίνησης του ρομπότ

2. Επιστροφής του ρομπότ (έχει πατηθεί το κουμπί)

volatile int started = 0; // Flag to check if movement has begun

volatile int returning = 0;

Ορίζουμε συναρτήσεις για τις περιστροφές της σκούπας, στις οποίες ανάβουμε τα κατάλληλα LED και εκτελούμε μια mod 4 πράξη για να πάρουμε την νέα κατεύθυνση.

void turn\_left() {

PORTD.OUT = 0b00000011;

current\_dir = (current\_dir + 1) % 4; // Cycle: n ? w ? s ? e ? n

}

void turn\_right() {

PORTD.OUT = 0b00000110;

current\_dir = (current\_dir - 1 + 4) % 4; // Cycle: n ? e ? s ? w ? n

}

## main:

Αρχικοποιούμε τα LED, το κουμπί και τον prescaler για τους χρονιστές.

PORTD.DIR |= 0b00000111; // PIN 0-2 is output

PORTD.OUT |= 0b00000111; // init LED (all off)

// pullup enable and Interrupt enabled with sense on rising edge (button release)

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

// split timer with prescaler 1024

TCA0.SPLIT.CTRLA = TCA\_SPLIT\_CLKSEL\_DIV1024\_gc | TCA\_SPLIT\_ENABLE\_bm;

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

Αρχικοποιούμε τον ADC σε Single Conversion Mode για την πρώτη δειγματοληψία, ενεργοποιούμε τα interrupts όταν είναι έτοιμο το αποτέλεσμα αποθηκεύουμε το threshold στο WINLT για μελλοντική χρήση και ξεκινάμε το conversion.

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

ADC0.CTRLA &= ~ADC\_FREERUN\_bm; // Disable Free Running (Enable Single Mode)

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

ADC0.MUXPOS = ADC\_MUXPOS\_AIN7\_gc; // Switch to Input 7

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; // Enable Debugging

ADC0.INTCTRL |= ADC\_RESRDY\_bm; // Enable Single Conversion ISR

ADC0.WINLT = threshold; // Set threshold (for future use)

ADC0.COMMAND = ADC\_STCONV\_bm; // Start the conversion

Αρχίζουμε τον χρονιστή με την τιμη Τ1 (1ms) για το ISR του HUNF και ενεργοποιούμε τα interrupts.

// Start T1 timer

TCA0.SPLIT.HCNT = T1;

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

sei();

Αρχίζουμε τον βρόγχο επανάληψης, ο οποίος σε κάθε κύκλο ελέγχει εαν η σκούπα έχει γυρίσει στο αρχικό της σημείο, βγαίνοντας απο τον βρόγχο εαν έχει.

while (1) {

if (y == 0 && x == 0 && started) // Might need a small threshold for real world scenario

break;

Στην συνέχεια, αν το ρομπότ προχωράει μπροστα (χρησιμοποιείται ο μπροστά αισθητήρας) εκτελεί τα παρακάτω βήματα:

1. Απενεργοποιεί τα interrupts και ενεργοποιεί τα led που υποδεικνύουν μπροστά κίνηση.

2. Θέτει το start = 1, εάν ειναι 0 (στην αρχή του προγράμματος)

3. Χρησιμοποιώντας την παρούσα κατεύθυνση του ρομπότ, μεταβάλλει κατα 1 την κατάλληλη μεταβλητή.

4. Ενεργοποιεί τα interrupts.

**Η απενεργοποίηση / ενεργοποίηση των interrupts ειναι απαραίτητη, καθώς χωρίς αυτή ένα interrupt μπορεί να μεταβάλλει την κατεύθυνση του ρομπότ, αλλά να βρίσκεται ήδη μέσα στο switch, και έτσι να μεταβάλλει την θέση λανθασμένα).**

if( ADC0.MUXPOS == ADC\_MUXPOS\_AIN6\_gc) {// Device moves while in front sensor mode

cli(); // Disable interrupts to avoid race conditions (e.g. turning left while inside switch statement)

if(started == 0) started = 1;

PORTD.OUT = 0b00000101; // Enable Front LED's

switch (current\_dir) { // Increment coordinates

case 0: y++; break;

case 1: x--; break;

case 2: y--; break;

case 3: x++; break;

}

sei(); // Re enable Interrupts after incrementing coordinates

}

}

Τέλος, όταν βγεί απο τον βρόγχο, απενεργοποιούνται όλα τα led, ο ADC και τα interrupts πριν τερματίσει το πρόγραμμα.

PORTD.OUT = 0b00000111; // Turn off the lights

ADC0.CTRLA &= ~ADC\_ENABLE\_bm; // Disable ADC

cli();

## ISR(ADC0\_RESRDY\_vect):

Αυτό το ISR ενεργοποιείται όταν επιστρέφεται αποτέλεσμα από Single mode.

Ελέγχεται εάν το αποτέλεσμα ειναι μεγαλύτερο του επιτρεπτού (άρα δέν υπάρχει τοίχος στον πλαϊνό σένσορα), όπου στρίβει δεξία ή αριστερά ανάλογα το εάν έχει πατηθεί το κουμπί επιστροφής.

// Right turn (left when returning)

ISR(ADC0\_RESRDY\_vect) {

if (ADC0.RES > threshold) { // if there isn't a wall on the left/right (returning/not), turn

if (returning)

turn\_left();

else

turn\_right();

}

Τέλος, καθαρίζονται τα flags.

ADC0.INTFLAGS = ADC\_RESRDY\_bm;

}

## ISR(ADC0\_WCOMP\_vect):

Αυτό το ISR ενεργοποιείται όταν το αποτέλεσμα στο Free Running Mode ειναι μικρότερο του threshold.

Στρίβει δεξία ή αριστερά ανάλογα το εάν έχει πατηθεί το κουμπί επιστροφής.

// Left turn (right when returning)

ISR(ADC0\_WCOMP\_vect) { // if this triggers it means there's a wall in front, so turn

if (returning)

turn\_right();

else

turn\_left();

ADC0.INTFLAGS = ADC\_WCMP\_bm;

}

## ISR(PORTF\_PORT\_vect):

Αυτό το ISR ενεργοποιείται όταν πατιέται το κουμπί στο ρομπότ.

Ανάβουν όλα τα φώτα, θέτεται 1 η μεταβλητή επιστροφής και η κατεύθυνση αντιστρέφεται.

// Button pressed

ISR(PORTF\_PORT\_vect) {

PORTD.OUTCLR = 0b00000111; // turn on all the LEDs

returning = 1; // Toggle returning

current\_dir = (current\_dir + 2) % 4; // Turn Around

Αρχίζει να μετράει το Τ3 (ο χρόνος που θα ειναι αναμμένα τα LED) και, τέλος, καθαρίζονται τα flags του κουμπιού.

TCA0.SPLIT.LCNT = T3; // Start timer for LED lights

TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_LUNF\_bm;

TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_LUNF\_bm;

PORTF.INTFLAGS |= PORT\_INT5\_bm;

}

## ISR(TCA0\_LUNF\_vect):

Αυτό το ISR ενεργοποιείται όταν χτυπάει το Τ2 (1ms μετά την δειγματοληψία του πλαϊνού αισθητήρα).

Απενεργοποιείται το interrupt RESRDY (καθώς το χρησιμοποιούμε για την δειγματοληψία τον πλαινών αισθητήρων).

ISR(TCA0\_HUNF\_vect) {

ADC0.INTCTRL &= ~ADC\_RESRDY\_bm; // Disable RESRDY ISR

Επιλέγεται η κατάλληλη είσοδος για τον ADC, ενεργοποιείται το Free Running Mode καθώς και το WCMP interrupt (που κάνει handle τον μπροστά αισθητήρα), και ενεργοποιούμε το Window Comparator Mode.

ADC0.CTRLA |= ADC\_FREERUN\_bm; // Enable Free Running Mode

ADC0.INTCTRL |= ADC\_WCMP\_bm; // Enable WCMP ISR

ADC0.CTRLE |= ADC\_WINCM0\_bm; // Tells the ADC to use WCMP Mode

Ξεκινάει η δειγματοληψία και αρχίζει το Τ2.

TCA0.SPLIT.LCNT = T2; // Start T2

TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_LUNF\_bm;

TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_LUNF\_bm;

1. Τέλος, καθαρίζουμε τα flags του HUNF.
2. TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_HUNF\_bm;
3. }

## ISR(TCA0\_LUNF\_vect):

Αυτό το ISR ενεργοποιείται όταν χτυπάει το Τ1 (μετά απο 2ms συνεχούς δειγματοληψείας του μπροστινού αισθητήρα).

Επιλέγεται η κατάλληλη είσοδος για τον ADC.

if(!returning)

ADC0.MUXPOS = ADC\_MUXPOS\_AIN7\_gc; // IN 7 = right sensor

else

ADC0.MUXPOS = ADC\_MUXPOS\_AIN5\_gc; // IN 5 = left sensor

Ενεργοποιείται το Single Conversion Mode, ενεργοποιείται το interrupt RESRDY και απενεργοποιείται το WCMP. Επίσης απενεργοποιείται το Window Comparator Mode, καθώς δεν μας χρειάζεται στην προκειμένη περίπτωση.

ADC0.CTRLA &= ~ADC\_FREERUN\_bm; // Disable Free Running Mode (Enable Single Conversion)

ADC0.INTCTRL |= ADC\_RESRDY\_bm; // Enable RESRDY ISR (side sensor ISR)

ADC0.INTCTRL &= ~ADC\_WCMP\_bm; // Disable WCMP Interrupt

ADC0.CTRLE &= ~ADC\_WINCM0\_bm; // Disable Window Comparator Mode

Απενεργοποιούνται όλα τα LED και ξεκινάει το conversion.

Τέλος, ξεκινάει ο χρονιστής για το Τ1 και καθαρίζουμε τα flags του LUNF.

TCA0.SPLIT.HCNT = T1; // Start T1

TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_HUNF\_bm;

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

TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_LUNF\_bm; // Clear Flags

}

## Κώδικας:

#include <avr/io.h>

#include <avr/interrupt.h>

// f\_clk = 20MHz

// prescaler = 1024

// ftimer = 20MHz / 1024 = 19.531KHz

// T = value / ftimer

#define T2 39 // T = 39 / 19.5 = 2ms

#define T1 19.5 // T = 19.5 / 19.5 = 1ms

#define T3 58.5 // T = 58.5 / 19.5 = 3ms

#define threshold 25

volatile long int x = 0; // Current X coordinate

volatile long int y = 0; // Current Y coordinate

volatile int current\_dir = 0; // 0=north, 1=west, 2=south, 3=east

volatile int started = 0; // Flag to check if movement has begun

volatile int returning = 0;

void turn\_left() {

PORTD.OUT = 0b00000011;

current\_dir = (current\_dir + 1) % 4; // Cycle: n ? w ? s ? e ? n

}

void turn\_right() {

PORTD.OUT = 0b00000110;

current\_dir = (current\_dir - 1 + 4) % 4; // Cycle: n ? e ? s ? w ? n

}

int main() {

PORTD.DIR |= 0b00000111; // PIN 0-2 is output

PORTD.OUT |= 0b00000111; // init LED (all off)

// pullup enable and Interrupt enabled with sense on rising edge (button release)

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

// split timer with prescaler 1024

TCA0.SPLIT.CTRLA = TCA\_SPLIT\_CLKSEL\_DIV1024\_gc | TCA\_SPLIT\_ENABLE\_bm;

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

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

ADC0.CTRLA &= ~ADC\_FREERUN\_bm; // Disable Free Running (Enable Single Mode)

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

ADC0.MUXPOS = ADC\_MUXPOS\_AIN7\_gc; // Switch to Input 7

ADC0.DBGCTRL |= ADC\_DBGRUN\_bm; // Enable Debugging

ADC0.INTCTRL |= ADC\_RESRDY\_bm; // Enable Single Conversion ISR

ADC0.WINLT = threshold; // Set threshold (for future use)

ADC0.COMMAND = ADC\_STCONV\_bm; // Start the conversion

// Start T1 timer

TCA0.SPLIT.HCNT = T1;

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

sei();

while (1) {

if (y == 0 && x == 0 && started) // Might need a small threshold for real world scenario

break;

if( ADC0.MUXPOS == ADC\_MUXPOS\_AIN6\_gc) {// Device moves while in front sensor mode

cli(); // Disable interrupts to avoid race conditions (e.g. turning left while inside switch statement)

if(started == 0) started = 1;

PORTD.OUT = 0b00000101; // Enable Front LED's

switch (current\_dir) { // Increment coordinates

case 0: y++; break;

case 1: x--; break;

case 2: y--; break;

case 3: x++; break;

}

sei(); // Re enable Interrupts after incrementing coordinates

}

}

PORTD.OUT = 0b00000111; // Turn off the lights

ADC0.CTRLA &= ~ADC\_ENABLE\_bm; // Disable ADC

cli();

}

// Right turn (left when returning)

ISR(ADC0\_RESRDY\_vect) {

if (ADC0.RES > threshold) { // if there isn't a wall on the left/right (returning/not), turn

if (returning)

turn\_left();

else

turn\_right();

}

ADC0.INTFLAGS = ADC\_RESRDY\_bm;

}

// Left turn (right when returning)

ISR(ADC0\_WCOMP\_vect) { // if this triggers it means there's a wall in front, so turn

if (returning)

turn\_right();

else

turn\_left();

ADC0.INTFLAGS = ADC\_WCMP\_bm;

}

// Button pressed

ISR(PORTF\_PORT\_vect) {

PORTD.OUTCLR = 0b00000111; // turn on all the LEDs

returning = 1; // Toggle returning

current\_dir = (current\_dir + 2) % 4; // Turn Around

TCA0.SPLIT.LCNT = T3; // Start timer for LED lights

TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_LUNF\_bm;

TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_LUNF\_bm;

PORTF.INTFLAGS |= PORT\_INT5\_bm;

}

ISR(TCA0\_HUNF\_vect) {

ADC0.INTCTRL &= ~ADC\_RESRDY\_bm; // Disable RESRDY ISR

ADC0.MUXPOS = ADC\_MUXPOS\_AIN6\_gc; // Switch to front sensor

ADC0.CTRLA |= ADC\_FREERUN\_bm; // Enable Free Running Mode

ADC0.INTCTRL |= ADC\_WCMP\_bm; // Enable WCMP ISR

ADC0.CTRLE |= ADC\_WINCM0\_bm; // Tells the ADC to use WCMP Mode

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

TCA0.SPLIT.LCNT = T2; // Start T2

TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_LUNF\_bm;

TCA0.SPLIT.INTCTRL |= TCA\_SPLIT\_LUNF\_bm;

TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_HUNF\_bm;

}

ISR(TCA0\_LUNF\_vect) {

//switch to side sensor

if(!returning)

ADC0.MUXPOS = ADC\_MUXPOS\_AIN7\_gc; // IN 7 = right sensor

else

ADC0.MUXPOS = ADC\_MUXPOS\_AIN5\_gc; // IN 5 = left sensor

ADC0.CTRLA &= ~ADC\_FREERUN\_bm; // Disable Free Running Mode (Enable Single Conversion)

ADC0.INTCTRL |= ADC\_RESRDY\_bm; // Enable RESRDY ISR (side sensor ISR)

ADC0.INTCTRL &= ~ADC\_WCMP\_bm; // Disable WCMP Interrupt

ADC0.CTRLE &= ~ADC\_WINCM0\_bm; // Disable Window Comparator Mode

PORTD.OUT |= 0b00000111; // Turn OFF LED's

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

TCA0.SPLIT.HCNT = T1; // Start T1

TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_HUNF\_bm;

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

TCA0.SPLIT.INTFLAGS |= TCA\_SPLIT\_LUNF\_bm; // Clear Flags

}