



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΤΕΧΝΟΛΟΓΙΑΣ & ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

3^η ΑΣΚΗΣΗ

07/05/2021

Αλέξανδρος Νατσολλάρη AM: 1057769

Παναγιώτης Μπαρμπούνης AM: 1054382

Παραδοχές :

- Στο PIN1 θεωρούμε ότι ανάβει το LED1 για να πηγαίνει ευθεία η οικιακή συσκευή.
Στο PIN2 θεωρούμε ότι ανάβει το LED2 για να στρίβει αριστερά η οικιακή συσκευή.
Στο PIN3 θεωρούμε ότι ανάβει το LED3 για να στρίβει δεξιά η οικιακή συσκευή.
- Βάλαμε για timer τη συνάρτηση `_delay_ms()`. Ορίσαμε τα 5 ms για το χρόνο που χρειάζεται η συσκευή για δεξιά ή αριστερή στροφή(90 μοιρών) και 10 ms για την αναστροφή(180 μοιρές) αφού θα κάνει την διπλάσια κίνηση από μια απλή στροφή.
- Το threshold το βάλαμε να είναι 10 οπότε οποιαδήποτε τιμή κάτω από αυτήν στο RES του ADC ενεργοποιεί το interrupt για να στρίψει αριστερά η συσκευή στην κανονική πορεία και δεξιά στην ανάποδη πορεία.
- Η τιμή του ADC (που προσομοιώνει το ποτενσιόμετρο την αλλάζουμε manually από το IO) , την βάζουμε σε χαμηλή τιμή(κάτω από 10) για να προσομοιώσουμε ότι υπάρχει τοίχος μπροστά σε κοντινή απόσταση και μεγαλύτερη από 10 για να βγούμε από το interrupt με την λογική ότι αφού έστριψε η συσκευή δεν υπάρχει τοίχος μπροστά σε κοντινή απόσταση. Εδώ να σημειώσουμε ότι αν η τιμή του RES στο ADC δεν αλλάξει πριν η ροή του κώδικα φτάσει στα if και else του interrupt `ISR(ADC0_WCOMP_vect)` τότε θα τρέξει για παραπάνω από μία φορές. Για να το αποφύγουμε αυτό βάλαμε breakpoint στην γραμμή 95 όπου μόλις φτάσουμε εκεί αλλάζουμε την τιμή του RES σε κάτι μεγαλύτερο από το 10.
- Ξεκινάμε με ανοιχτό το LED1 που σημαίνει ότι ξεκινάμε με ευθεία πορεία.
- Η μεταβλητή counter μετράει τις γωνίες που συναντάει η συσκευή.
- Για το interrupt για να ξεκινήσει η ανάποδη πορεία χρησιμοποιούμε το PIN5 του PORTC

Πείραμα

Ο κώδικας της άσκησης είναι ο ακόλουθος:

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>

int counter=0;
int interr=0;
int main(){

    //LED1 gia eutheia
    PORTD.DIR |= PIN1_bm; //PIN is output
    PORTD.OUT |= PIN1_bm;

    //LED2 gia aristera
    PORTD.DIR |= PIN2_bm; //PIN is output

    //LED3 gia dexia
```

```

PORTD.DIR |= PIN3_bm; //PIN is output

while(counter<8 && counter!=-1){
    //pullup enable and Interrupt enabled with sense on both edges
    PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
    sei(); //enable interrupts

    printf("%d",counter);

    //initialize the ADC for Free-Running mode
    ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; //10-bit resolution
    ADC0.CTRLA |= ADC_FREERUN_bm; //Free-Running mode enabled
    ADC0.CTRLA |= ADC_ENABLE_bm; //Enable ADC
    ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc; //The bit
    //Enable Debug Mode
    ADC0.DBGCTRL |= ADC_DBGRUN_bm;
    //Window Comparator Mode
    ADC0.WINLT |= 10; //Set threshold
    ADC0.INTCTRL |= ADC_WCMP_bm; //Enable Interrupts for WCM
    ADC0.CTRLE |= ADC_WINCM0_bm; //Interrupt when RESULT < WINLT
    sei();
    ADC0.COMMAND |= ADC_STCONV_bm; //Start Conversion

    //pullup enable and Interrupt enabled with sense on both edges
    //PORTC PIN5 SWITCH gia interrupt gia na ksekinisei anapodh poreia
    PORTC.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;

    if(interr==1){
        while(counter>(-1)){

            printf("%d",counter);
            //pullup enable and Interrupt enabled with sense on both edges
            PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
            sei(); //enable interrupts

            //initialize the ADC for Free-Running mode
            ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; //10-bit resolution
            ADC0.CTRLA |= ADC_FREERUN_bm; //Free-Running mode enabled
            ADC0.CTRLA |= ADC_ENABLE_bm; //Enable ADC
            ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc; //The bit
            //Enable Debug Mode
            ADC0.DBGCTRL |= ADC_DBGRUN_bm;
            //Window Comparator Mode
            ADC0.WINLT |= 10; //Set threshold
            ADC0.INTCTRL |= ADC_WCMP_bm; //Enable Interrupts for WCM
            ADC0.CTRLE |= ADC_WINCM0_bm; //Interrupt when RESULT < WINLT
            sei();
            ADC0.COMMAND |= ADC_STCONV_bm; //Start Conversion

        }
    }
}

```

```

}

//entolh gia na kleisei to LED1 afou ektelestei
//kai na termatisei h diadikasia
PORTD.OUTCLR= PIN1_bm;
printf("%d",counter);
}

//Se ayto to interrupt ginontai allages twn LED
//sumfwna me ton aisththora(ADC se emas) poy deixnei apostash apo ton toixo
//kai analoga sthn poreia ths syskeuhs(kanonikh h anapodh dhladh)
ISR(ADC0_WCOMP_vect){
    int intflags = ADC0.INTFLAGS;
    ADC0.INTFLAGS = intflags; //edw thelei breakpoint kai
    //molis ftasei se auto to breakpoint to run thelei Allagh h timh
    //tou ADC se kati megalutero tou 10 gia na treksei mia fora
    //an den allaksei twra tote trexei perissoteres apo 1 fores
    if(interr==1){
        PORTD.OUTCLR= PIN1_bm; //kleinei to LED1 molis ektelestei
        PORTD.OUT |= PIN3_bm;//anoigei to LED3 molis ektelestei
        _delay_ms(5);
        PORTD.OUTCLR= PIN3_bm; //kleinei to LED3 molis ektelestei
        PORTD.OUT |= PIN1_bm;//anoigei to LED1 molis ektelestei
        counter--;
    }
    else if (interr==0){
        PORTD.OUTCLR= PIN1_bm; //kleinei to LED1 molis ektelestei
        PORTD.OUT |= PIN2_bm;//anoigei to LED2 molis ektelestei
        _delay_ms(5);
        PORTD.OUTCLR= PIN2_bm; //kleinei to LED2 molis ektelestei
        PORTD.OUT |= PIN1_bm;//anoigei to LED1 molis ektelestei
        counter++;
    }
}

//Se auto to interrupt strivei deksia an erthei interrupt apo to
//PORTF PIN5 sthn kanonikh poreia h aristera an eimaste sthn anapodh
//poreia
ISR(PORTF_PORT_vect){
    //clear the interrupt flag
    int intflags = PORTF.INTFLAGS;
    PORTF.INTFLAGS=intflags;
    if(interr==1){
        PORTD.OUTCLR= PIN1_bm; //kleinei to LED1 molis ektelestei
        PORTD.OUT |= PIN2_bm;//anoigei to LED2 molis ektelestei
        _delay_ms(5);
        PORTD.OUTCLR= PIN2_bm; //kleinei to LED2 molis ektelestei
        PORTD.OUT |= PIN1_bm;//anoigei to LED1 molis ektelestei
        counter--;
    }
    else if (interr==0){
        PORTD.OUTCLR= PIN1_bm; //kleinei to LED1 molis ektelestei
        PORTD.OUT |= PIN3_bm;//anoigei to LED3 molis ektelestei
        _delay_ms(5);
        PORTD.OUTCLR= PIN3_bm; //kleinei to LED3 molis ektelestei
        PORTD.OUT |= PIN1_bm;//anoigei to LED1 molis ektelestei
        counter++;
    }
}
}

```

```

//Se auto to interrupt ginetai anastrofh(strofh 180 moires afou anoigoun
//kai ta 3 LED)
//kai energopoiei thn anapodh poreia
ISR(PORTC_PORT_vect){
    //clear the interrupt flag
    int intflags = PORTC.INTFLAGS;
    PORTC.INTFLAGS=intflags;
    //To LED1 den xreiazetai na to anoikoume einai hdh anoixto
    PORTD.OUT |= PIN2_bm;//anoigei to LED2 molis ektelestei
    PORTD.OUT |= PIN3_bm;//anoigei to LED3 molis ektelestei
    _delay_ms(10);
    PORTD.OUTCLR= PIN2_bm; //kleinei to LED2 molis ektelestei
    PORTD.OUTCLR= PIN3_bm; //kleinei to LED3 molis ektelestei
    interr=1;
    counter--;
}

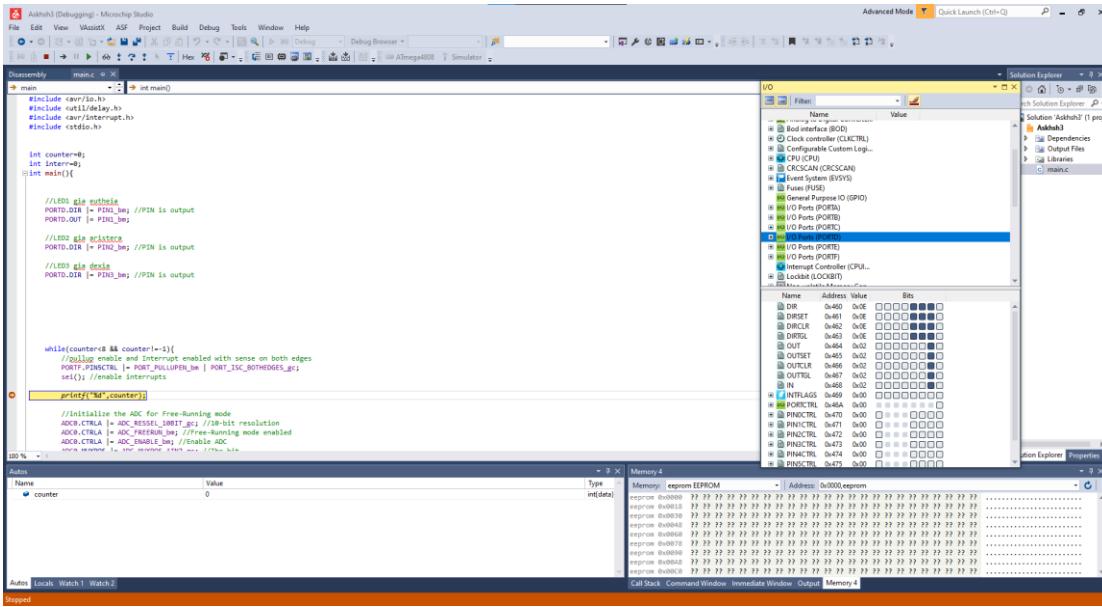
```

Αποτελέσματα Εκτέλεσης

Ξεκινώντας βάζουμε breakpoint στα εξής σημεία για να ελέγχουμε τα ενδιάμεσα αποτελέσματα(να σημειώσουμε εδώ ότι δεν στέλνουμε το zip με τα breakpoint ήδη μέσα γιατί κατά το testing παρατηρήσαμε ένα bug, καθώς αφού έστειλε ο ένας το zip στον άλλο τα breakpoint δεν δούλευαν κατά την εκτέλεση του κώδικα και υπήρχε πρόβλημα): γραμμή 33, 45, 56, 71, 87, 95, 100, 101, 103, 104, 108, 109, 111, 112, 127, 128, 130, 131, 135, 136, 138, 139, 154, 155, 157, 158.

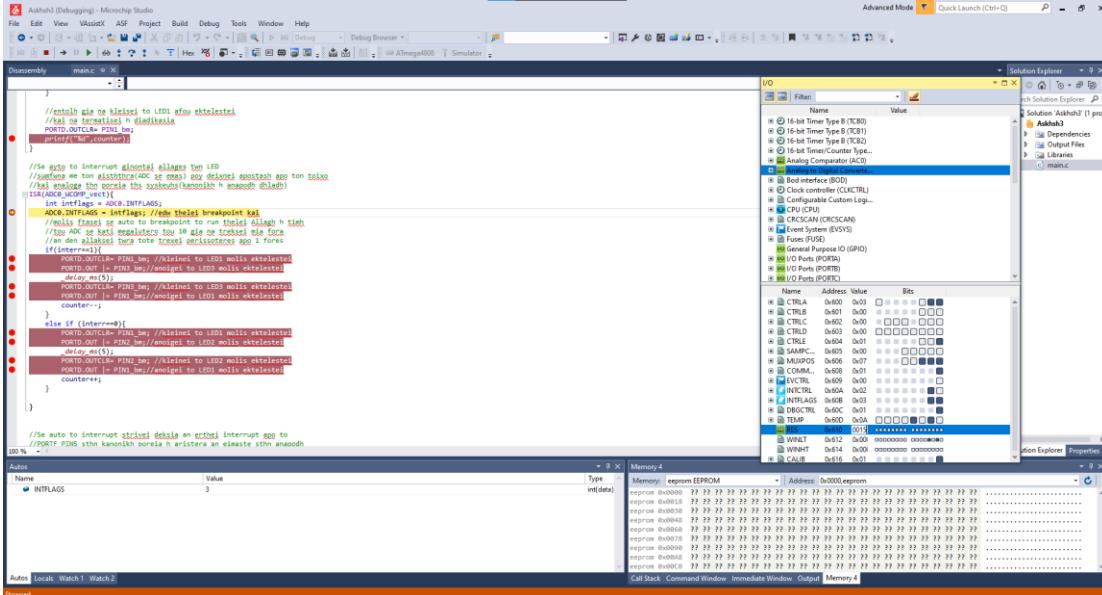
KANONIKH POREIA

Ξεκινώντας ανάβουμε το LED1 για ευθεία πορεία.



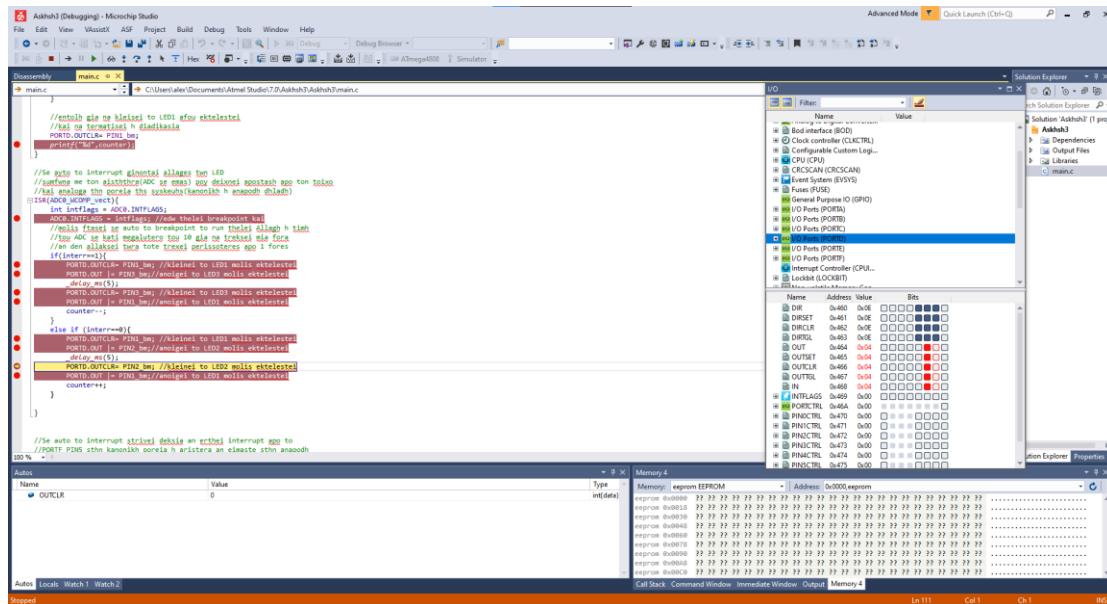
Έστω ότι η συσκευή προχώρησε ευθεία και βρήκε τοίχο(δηλαδή η τιμή του RES στο ADC <10) τότε καλείται το interrupt **ISR(ADC0_WCOMP_vect)**:

Μόλις η ροή φτάσει στο **breakpoint γραμμή 95 πρέπει να αλλαχτεί** η τιμή του RES σε κάτι μεγαλύτερο του 10 για να τρέξει μόνο μια φορά αλλιώς υποθέτει ότι αφού έστριψε ξανά βρήκε τοίχο μπροστά.



Αφού αλλάξουμε την τιμή ξανά πατάμε run :

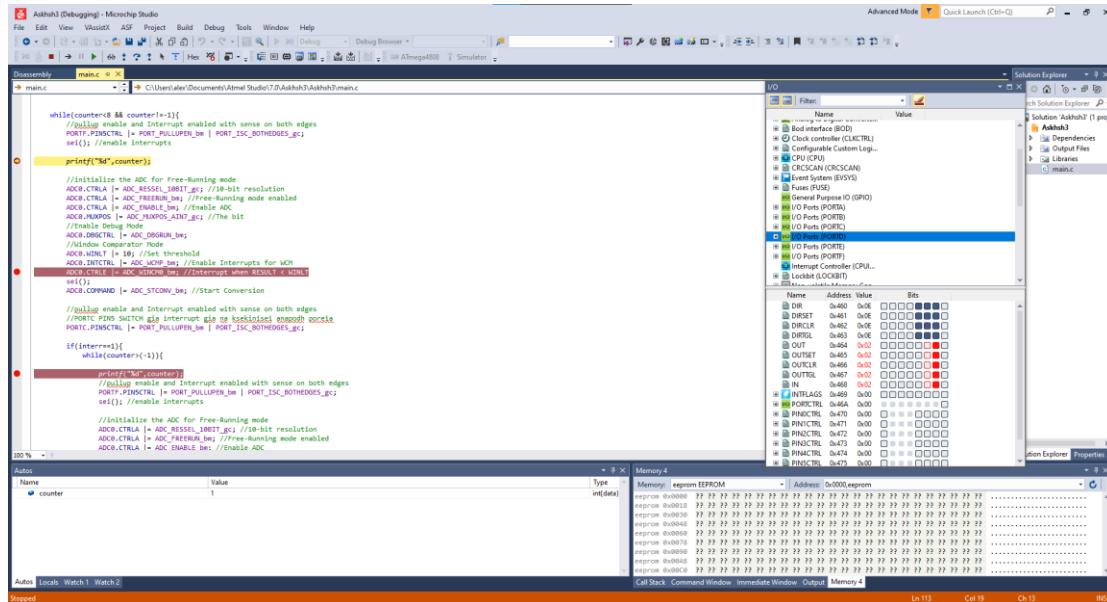
Κλείνει το LED1 που είναι για ευθεία πορεία και ανάβει το LED2 για να στρίψει αριστερά:



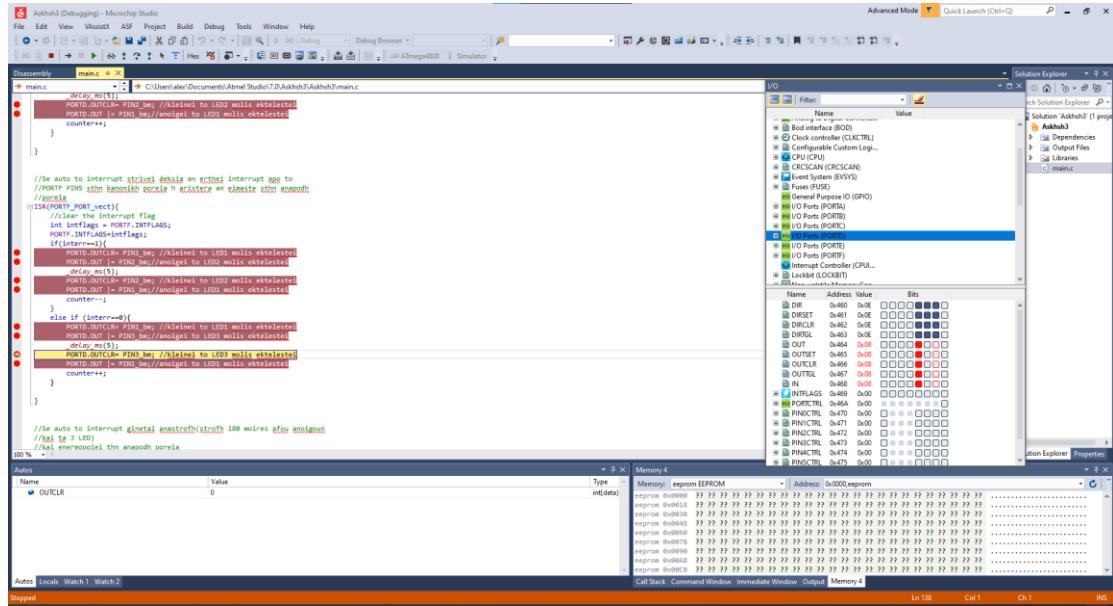
Μόλις τελειώσει το delay (5 ms) και εκτελεστούν οι εντολές

```
PORTD.OUTCLR= PIN2_bm; //kleinei to LED2 molis ektelestei  
PORTD.OUT |= PIN1_bm;//anoigei to LED1 molis ektelestei  
counter++;
```

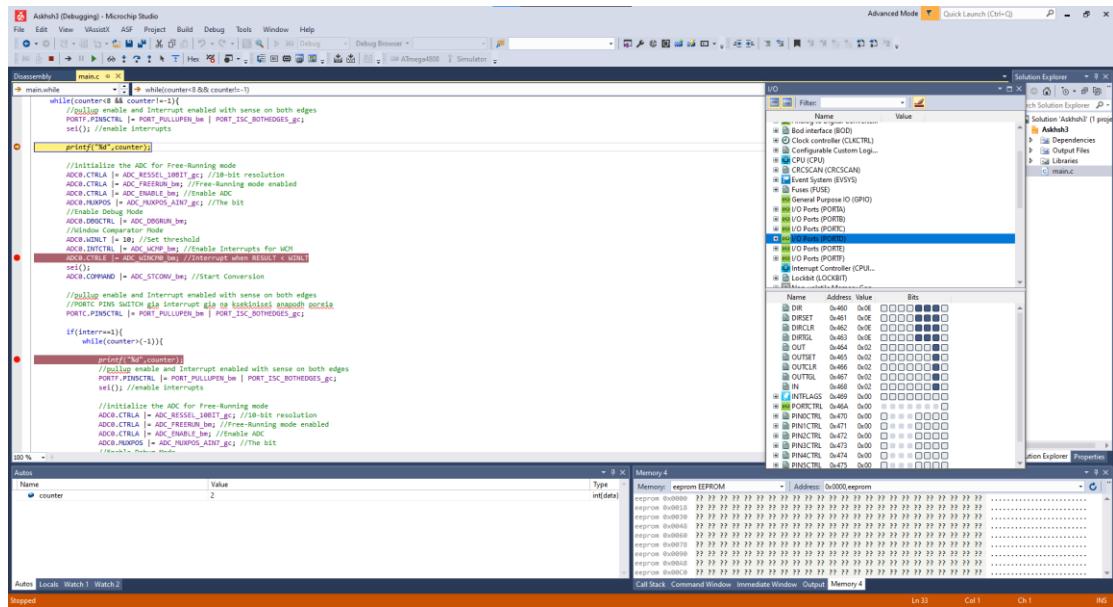
κλείνει το LED2 και ανοίγει το LED1 για ευθεία πορεία και το counter αυξήθηκε κατά ένα που σημαίνει ότι συναντήσαμε γωνία:



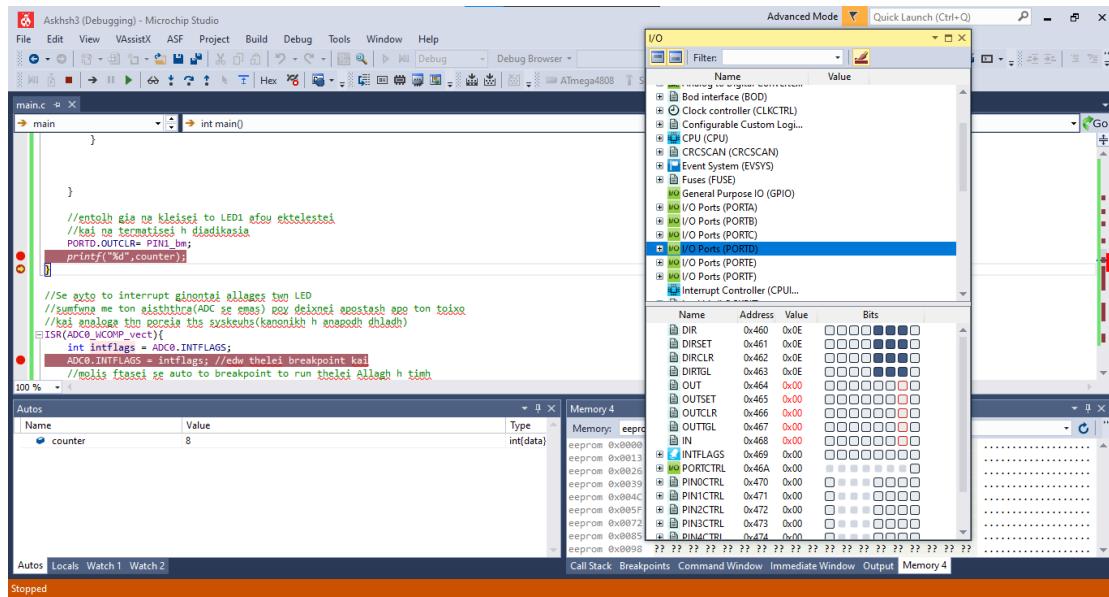
Τώρα έστω ότι δεν υπάρχει τοίχος δεξιά άρα πατάμε το interrupt (PIN5 PORTF) για να στρίψει δεξιά τότε καλείται το interrupt **ISR(PORTF_PORT_vect)** όπου ανοίγει το LED3 για να στρίψει δεξιά και κλείνει το LED1:



Αφού στρίψει δεξιά και ολοκληρωθεί η διαδικασία σβήνει το LED3 που είναι για να στρίψει δεξιά και ανάβει το LED1 που είναι για ευθεία πορεία και το counter αυξήθηκε κατά ένα που σημαίνει ότι συναντήσαμε και άλλη γωνία:

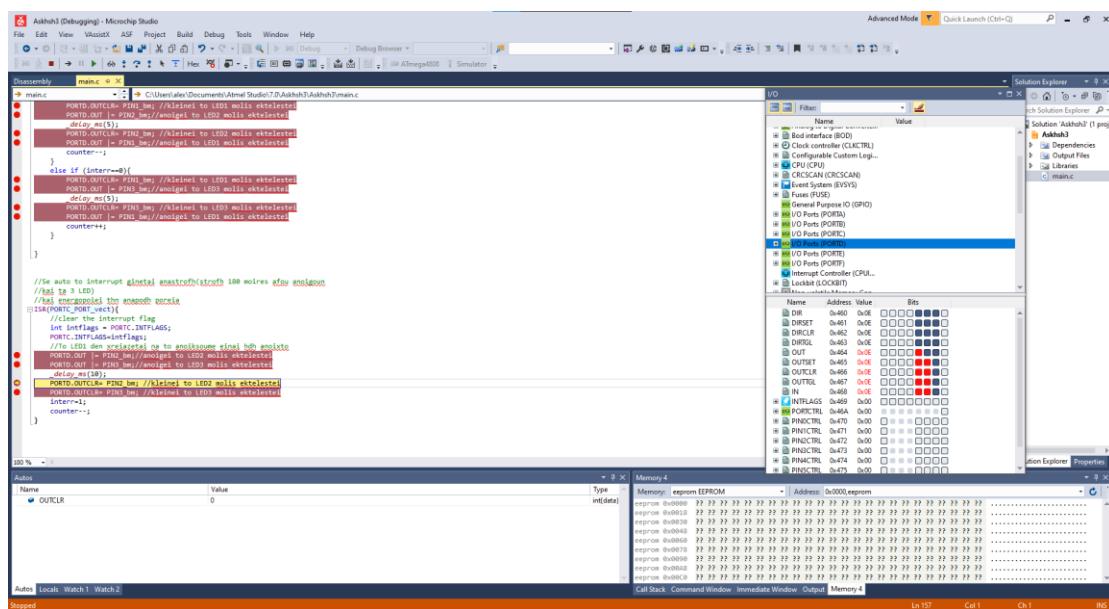


Αφού ολοκληρώσει το περίγραμμα το δωματίου δηλαδή και τις 8 στροφές (όπως φαίνεται και στην μεταβλητή counter που έχουμε ορίσει ώστε να μετράει γωνίες) κλείνουμε το LED1 που είναι για ευθεία πορεία και τελειώνει το πρόγραμμα.



ΑΝΑΠΟΔΗ ΠΟΡΕΙΑ

Έστω ότι έχουμε κάνει τις πρώτες 2 στροφές του δωματίου και θέλουμε να κάνει αναστροφή. Για να ενεργοποιηθεί η ανάποδη πορεία πρέπει να πατηθεί το PIN5 του PORTC και ενεργοποιείται έτσι το interrupt ISR(PORTC_PORT_vect) όπου ανάβουν τα 3 LED ταυτόχρονα για να προσημειωθεί η αναστροφή.



Εδώ έχουμε ένα delay 10 ms για να στρίψει 180 μοίρες η συσκευή και να ξεκινήσει η ανάποδη πορεία του. Μόλις τελειώσει το delay σβήνουν τα LED2 και LED3 και το LED1 παραμένει ανοιχτό και η τιμή του counter που ήταν 2 σε αυτή την περίπτωση κατεβαίνει κατά ένα δηλαδή γίνεται 1 σε αυτή την περίπτωση.

The screenshot shows the Microchip Studio interface with the assembly view open. The assembly code for main.c includes ADC initialization and configuration. A memory dump window is visible on the right, showing the state of the PORTF registers. The code uses ADC interrupts to control LED states and update a counter.

```

main.c
ADC0.DRSCTR |= ADC_DRSCTR_WDTH_8BITS_gc; //whitecounter=8&8=counter=1
ADC0.CTRLA |= ADC_CTRLA_COMPARATOR_gc; //Window Comparator Mode
ADC0.HINL |= ADC_HINL_THRESHOLD_gc; //Set threshold
ADC0.INTCTRL |= ADC_INT0_WCH_be; //Enable Interrupts for WCH
ADC0.CTRLE |= ADC_WCHNB_WCH_gc; //Interrupt when RESULT < HINL
ADC0.COMPAWD |= ADC_STC0WNB_be; //Start Conversion

//pullup enable and Interrupt enabled with sense on both edges
PORTC_PINS0 |= PORT_PULLUPEN_gc | PORT_ISC_BOTHHEDGES_gc;
if(Intererrw){ //if interrupt
    while(counters<(1)){ //if(counter>0)
        printf("wdt_counter");
        //pullup enable and Interrupt enabled with sense on both edges
        PORTC_PINS0 |= PORT_PULLUPEN_gc | PORT_ISC_BOTHHEDGES_gc;
        sei(); //enable Interrupts
    }
}

//Initialize the ADC for Free-Running mode
ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; //10-bit resolution
ADC0.CTRLA |= ADC_FREERUN_bm; //Free-running mode enabled
ADC0.CTRLA |= ADC_FREERUN_A_gc; //Free-running mode for ADC
ADC0.MUXPOS |= ADC_MUXPOS_A_ADC_gc; //The bit
//enable Debug Mode
ADC0.CTRLB |= ADC_DEBUG_bm;
//Window Comparator Mode
ADC0.WINL |= 10; //Set threshold
ADC0.WINH |= 10; //Set threshold
ADC0.CTRLE |= ADC_WINCH_WCH_gc; //Interrupt when RESULT <=HINL
sei();
ADC0.COMPAWD |= ADC_STC0W_WCH_gc; //Start Conversion

}

```

Έστω τώρα ότι έχει ξεκινήσει η ανάποδη πορεία και έρχεται interrupt PIN5 του PORTF ότι δεν υπάρχει τοίχος αριστερά άρα ανάβει το LED2 για να στρίψει αριστερά και σβήνει το LED1:

The screenshot shows the Microchip Studio interface with the assembly view open. The assembly code for main.c includes handling of PORTF interrupt PIN5. A memory dump window is visible on the right, showing the state of the PORTF registers during the interrupt. The code toggles LED states based on the interrupt flag.

```

main.c
PORTF.INTFLAGS |= PORTF_INTFLAG5_gc; //Set interrupt to LED2 molis ekteleste
delay_ms(5);
PORTF.INTFLAGS |= PORTF_INTFLAG5_gc; //Set interrupt to LED2 molis ekteleste
PORTF.DOUTL |= PORTF_DOUTL5_bm; //Set interrupt to LED2 molis ekteleste
counter++; //counter=2

//Set auto to interrupt strobe debias an erthei interrupt app to
//PORTF 2ND strobe koumponi poteira h ertheia sto 2th snapshot
//noska
ISR(PORTF_PORT_vect){ //Set interrupt flag
    int flags = PORTF.INTFLAGS;
    PORTF.INTFLAGS=flags;
    if(flags & PORTF_INTFLAG5_gc){ //if interrupt
        PORTF.DOUTL |= PORTF_DOUTL5_bm; //Set interrupt to LED2 molis ekteleste
        PORTF.DOUTL |= PORTF_DOUTL4_bm; //Set interrupt to LED3 molis ekteleste
        PORTF.DOUTL |= PORTF_DOUTL3_bm; //Set interrupt to LED2 molis ekteleste
        PORTF.DOUTL |= PORTF_DOUTL2_bm; //Set interrupt to LED3 molis ekteleste
        PORTF.DOUTL |= PORTF_DOUTL1_bm; //Set interrupt to LED2 molis ekteleste
        PORTF.DOUTL |= PORTF_DOUTL0_bm; //Set interrupt to LED3 molis ekteleste
        counter++; //counter=1
    } else if (flags&~0x80){ //if interrupt ~0x80
        PORTF.DOUTL |= PORTF_DOUTL5_bm; //Set interrupt to LED1 molis ekteleste
        PORTF.DOUTL |= PORTF_DOUTL4_bm; //Set interrupt to LED3 molis ekteleste
        PORTF.DOUTL |= PORTF_DOUTL3_bm; //Set interrupt to LED1 molis ekteleste
        PORTF.DOUTL |= PORTF_DOUTL2_bm; //Set interrupt to LED3 molis ekteleste
        PORTF.DOUTL |= PORTF_DOUTL1_bm; //Set interrupt to LED1 molis ekteleste
        PORTF.DOUTL |= PORTF_DOUTL0_bm; //Set interrupt to LED3 molis ekteleste
        counter++; //counter=2
    }

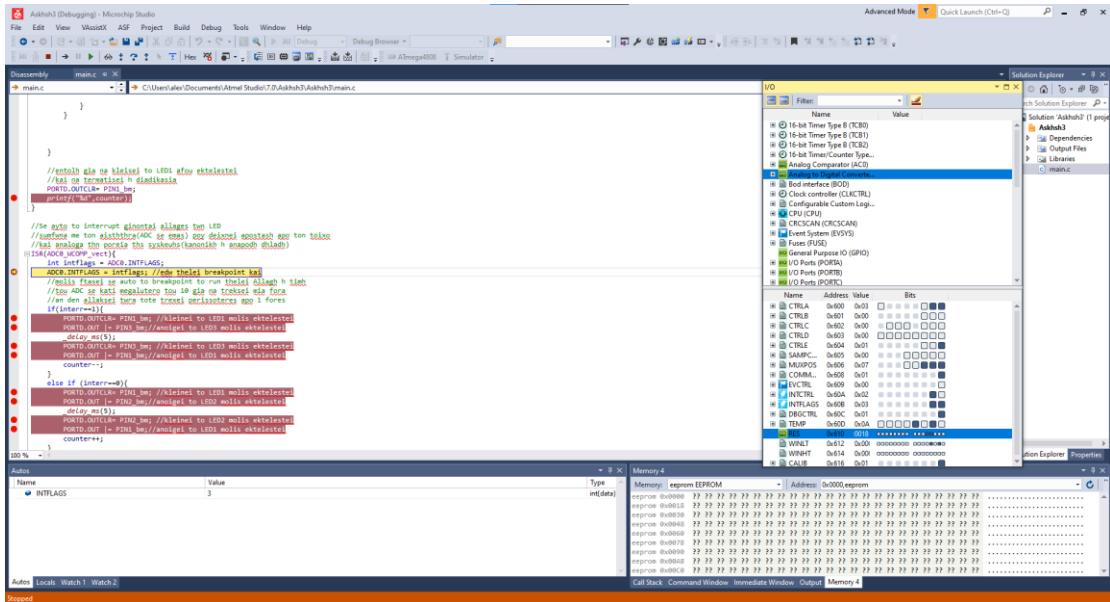
    //Set auto to interrupt matala anastrofhi/strofi 100 moires afou anoiou
}

```

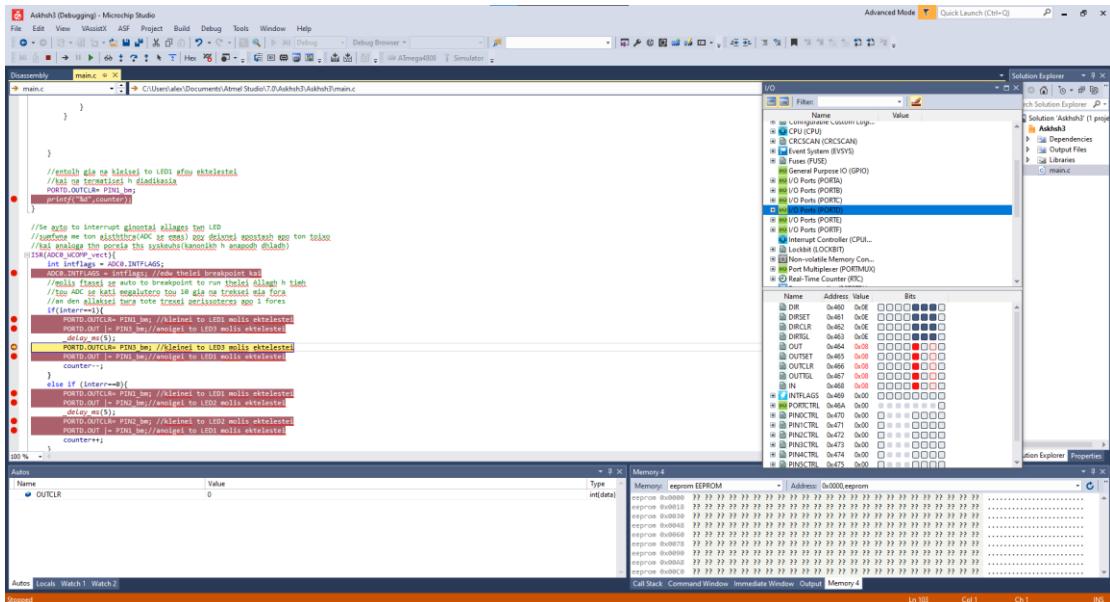
Μετά το delay (5 ms) σβήνει το LED2 και ανοίγει το LED1 για ευθεία πορεία και η τιμή του counter μειώνεται κατά ένα(όπως φαίνεται και στο screenshot γίνεται 0 από 1 που ήταν).

Έστω τώρα ότι εντοπίζει τοίχο ευθεία σε κοντινή απόσταση (δηλαδή $RES < 10$)

Μόλις η ροή φτάσει στο **breakpoint γραμμή 95** πρέπει να αλλαχτεί η τιμή του RES σε κάτι μεγαλύτερο του 10 για να τρέξει μόνο μια φορά αλλιώς υποθέτει ότι αφού έστριψε ξανά βρήκε τοίχο μπροστά.



Αφού μπει στην if του interrupt που είναι για την ανάποδη πορεία ανάβει το LED3 για να στρίψει δεξιά , (ανάβει LED3 για 5 ms) σβήνει και ανάβει το LED1 για ευθεία πορεία. Έπειτα το counter γίνεται -1 που σημαίνει ότι η συσκευή έφτασε στην αρχική θέση, σβήνει το LED1 και τερματίζει το πρόγραμμα.



Askkh3 (Debugging) - Microchip Studio

File Edit View VausiX ASF Project Build Debug Tools Window Help

Disassembly main.c x

```

main:
    .intn 0x00000000
    .if(Inter==1)
        .while(counter<1)
            .printf("0x%02x\r\n",counter)
            //ADC enable and Interrupt enabled with sense on both edges
            //PORTF_INTERRUPT |= PORT_ISC_BOTHEDGES_bm | PORT_ISC_BOTHEDGES_B8;
            //Initialize the ADC for Free-running mode
            ADC_CTRLA |= ADC_FREERUN_bm; //Free-running mode enabled
            ADC_CTRLB |= ADC_FREERUN_bm; //Free-running mode enabled
            ADC_MPOS |= ADC_MPOS_ASNG_gc; //The bit
            //Enable Debug Mode
            ADC_WHLT |= 10; //Set threshold
            //Enable Comparator Mode
            ADC_WHLT |= ADC_WHLT_CMPLR_bm; //Enable Interrupts for WCH
            ADC_CTRLA |= ADC_WHLCN_bm; //Interrupt when RESULT < WCH
            set();
            ADC_CDHAND |= ADC_STCONV_bm; //Start Conversion
        }
    }
    //enable pin as digital to LED1 pfou ektelestel
    //hal na tecmelihi h diadiakia
    PORTD_OUTCLEAR = PIN1_bm;
    printf("0x%02x\r\n",counter)
}

//Se avto to interrupt ginalles ton LED
//sumfema we ton aliththra(ADC se emes) moy dekneli apostroph app to taiso
//hal analogi th porwta ta sygkoumata(kanoumata h enapothi th ledou)
//128=4096
int intFlag = ADC_INTFLAG;
    ADCB_INTFLAG = intFlag; //mills tisefi se auto to breakpoint to run thei Allach h tisai
    100 % / //mills tisefi se auto to breakpoint to run thei Allach h tisai

```

Autos Locals Watch 1 Watch 2

Memory 4

Name	Address	Value	Type
OUTCLR	0x00000000	0	int[4]
counter	0x00000000	-1	int[4]

Call Stack Command Window Immediate Window Output Memory 4

Advanced Mode Quick Launch (Ctrl+Q)

Solution Explorer

Askkh3 (1 project)

- Dependencies
- Output Files
- Libraries
- main.c

Askkh3 (Debugging) - Microchip Studio

File Edit View VausiX ASF Project Build Debug Tools Window Help

Disassembly main.c x

```

main:
    .intn 0x00000000
    .if(Inter==1)
        .while(counter<1)
            .printf("0x%02x\r\n",counter)
            //ADC enable and Interrupt enabled with sense on both edges
            //PORTF_INTERRUPT |= PORT_ISC_BOTHEDGES_bm | PORT_ISC_BOTHEDGES_B8;
            //Initialize the ADC for Free-running mode
            ADC_CTRLA |= ADC_FREERUN_bm; //Free-running mode enabled
            ADC_CTRLB |= ADC_FREERUN_bm; //Free-running mode enabled
            ADC_MPOS |= ADC_MPOS_ASNG_gc; //The bit
            //Enable Debug Mode
            ADC_WHLT |= 10; //Set threshold
            //Enable Comparator Mode
            ADC_WHLT |= ADC_WHLT_CMPLR_bm; //Enable Interrupts for WCH
            ADC_CTRLA |= ADC_WHLCN_bm; //Interrupt when RESULT < WCH
            set();
            ADC_CDHAND |= ADC_STCONV_bm; //Start Conversion
        }
    }
    //enable pin as digital to LED1 pfou ektelestel
    //hal na tecmelihi h diadiakia
    PORTD_OUTCLEAR = PIN1_bm;
    printf("0x%02x\r\n",counter)
}

//Se gto to interrupt ginalles ton LED
//sumfema we ton aliththra(ADC se emes) moy dekneli apostroph app to taiso
//hal analogi th porwta ta sygkoumata(kanoumata h enapothi th ledou)
//128=4096
int intFlag = ADC_INTFLAG;
    ADCB_INTFLAG = intFlag; //mills tisefi se auto to breakpoint to run thei Allach h tisai
    100 % / //mills tisefi se auto to breakpoint to run thei Allach h tisai

```

Autos Locals Watch 1 Watch 2

Memory 4

Name	Address	Value	Type
OUTCLR	0x00000000	0	int[4]
counter	0x00000000	-1	int[4]

Call Stack Command Window Immediate Window Output Memory 4

Advanced Mode Quick Launch (Ctrl+Q)

Solution Explorer

Askkh3 (1 project)

- Dependencies
- Output Files
- Libraries
- main.c

BONUS

Παραδοχές :

- Για να τρέξει ο κώδικας μας θα πρέπει η συγκεκριμένη συσκευή να τοποθετείται με τοίχο δεξιά της.
- Θα πρέπει το σπίτι να μην έχει οξείες ή αμβλείες γωνίες και να μην είναι κυκλικό.
- Για το testing του κώδικα βάλαμε breakpoint στα εξής σημεία: γραμμή 35, 47, 58, 73, 89, 90, 92, 99, 104, 105, 107, 108, 112, 113, 115, 116, 132, 133, 135, 136, 140, 141, 143, 144, 160, 161, 163, 164.

Μπορείτε να σκεφτείτε έναν τρόπο ώστε και χωρίς να ξέραμε τις γωνίες του δωματίου η συσκευή μας να καταλάβαινε πότε ολοκλήρωσε το περίγραμμα του δωματίου;

Απάντηση: Για να καταλαβαίνουμε πότε ολοκλήρωσε το περίγραμμα του δωματίου σκεφτήκαμε να χρησιμοποιήσουμε έναν έξτρα counter(μεταβλητή counter2) για να δούμε πότε η συσκευή έκανε ένα θεωρητικό τετράγωνο με 4 αριστερές στροφές. Ουσιαστικά αναθέσαμε “βάρη(+1 αριστερές, -1 δεξιές)” στις αριστερές γωνίες και τις δεξιές όπου ο counter2 αν γίνει 4 έπειτα από τις προσθαφαιρέσεις σημαίνει ότι το θεωρητικό τετράγωνο ολοκληρώθηκε άρα σταματάει ολοκληρώθηκε το περίγραμμα του δωματίου.

Κατά πόσο ο κώδικάς σας μπορεί να αξιοποιηθεί και σε τελείως άγνωστα δωμάτια;

Απάντηση: Ο κώδικας μας μπορεί να αξιοποιηθεί σε οποιοδήποτε άγνωστο δωμάτιο αρκεί το δωμάτιο να μην έχει μη ορθές γωνίες και η συσκευή να τοποθετείται σε σημείο εκκίνησης με τοίχο δεξιά . Δηλαδή σε κυκλικά δωμάτια ή δωμάτια με μη ορθές γωνίες δεν μπορεί να λειτουργήσει.

Προσθέστε το στον κώδικα και εξηγήστε τον τρόπο σκέψη σας (Δεν χρειάζεται να αλλάξετε τον κώδικα της ανάποδης πορείας καθώς αυτό το ερώτημα αναφέρεται μόνο στην κανονική πορεία της συσκευής).

Απάντηση: Αρχικά ορίσαμε 2 counter , η μεταβλητή counter έχει την ίδια λειτουργία με πριν δηλαδή μετράει πλήθος γωνιών. Η μεταβλητή counter2 αυξάνεται κατά ένα σε αριστερές στροφές και μειώνεται κατά ένα σε δεξιές στροφές. Έτσι ουσιαστικά δεν χρειάζεται να ξέρουμε τις γωνίες εξ αρχής αλλά, όταν η μεταβλητή counter2 φτάσει την τιμή 4 σημαίνει ότι η συσκευή έχει κάνει ένα πλήρη “κύκλο” του δωματίου και έφτασε στην αρχική της θέση, άρα και τερματίζει το πρόγραμμα. Η ανάποδη εξακολουθεί να λειτουργεί όπως το αρχικό πρόγραμμα οπότε δεν χρειάστηκε κάποια αλλαγή.

Πείραμα

Ο κώδικας της άσκησης είναι ο ακόλουθος:

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>

int counter=0;
int interr=0;
int counter2=0;
int main(){

    //LED1 gia eutheia
    PORTD.DIR |= PIN1_bm; //PIN is output
    PORTD.OUT |= PIN1_bm;

    //LED2 gia aristera
    PORTD.DIR |= PIN2_bm; //PIN is output

    //LED3 gia dexia
    PORTD.DIR |= PIN3_bm; //PIN is output


while(counter!=-1 && counter2<4){
    //pullup enable and Interrupt enabled with sense on both edges
    PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
    sei(); //enable interrupts

    printf("%d",counter);
    printf("%d",counter2);

    //initialize the ADC for Free-Running mode
    ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; //10-bit resolution
    ADC0.CTRLA |= ADC_FREERUN_bm; //Free-Running mode enabled
    ADC0.CTRLA |= ADC_ENABLE_bm; //Enable ADC
    ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc; //The bit
    //Enable Debug Mode
    ADC0.DBGCTRL |= ADC_DBGRUN_bm;
    //Window Comparator Mode
    ADC0.WINLT |= 10; //Set threshold
    ADC0.INTCTRL |= ADC_WCMP_bm; //Enable Interrupts for WCM
    ADC0.CTRLE |= ADC_WINCM0_bm; //Interrupt when RESULT < WINLT
    sei();
    ADC0.COMMAND |= ADC_STCONV_bm; //Start Conversion

    //pullup enable and Interrupt enabled with sense on both edges
    //PORTC PIN5 SWITCH gia interrupt gia na ksekinisei anapodh poreia
    PORTC.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;

    if(interr==1){
        while(counter>(-1)){

            printf("%d",counter);
            //pullup enable and Interrupt enabled with sense on both edges
            PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_BOTHEDGES_gc;
```

```

        sei(); //enable interrupts

        //initialize the ADC for Free-Running mode
        ADC0.CTRLA |= ADC_RESSEL_10BIT_gc; //10-bit resolution
        ADC0.CTRLA |= ADC_FREERUN_bm; //Free-Running mode enabled
        ADC0.CTRLA |= ADC_ENABLE_bm; //Enable ADC
        ADC0.MUXPOS |= ADC_MUXPOS_AIN7_gc; //The bit
        //Enable Debug Mode
        ADC0.DBGCTRL |= ADC_DBGRUN_bm;
        //Window Comparator Mode
        ADC0.WINLT |= 10; //Set threshold
        ADC0.INTCTRL |= ADC_WCMP_bm; //Enable Interrupts for WCM
        ADC0.CTRLE |= ADC_WINCM0_bm; //Interrupt when RESULT < WINLT
        sei();
        ADC0.COMMAND |= ADC_STCONV_bm; //Start Conversion

    }

}

//entolh gia na kleisei to LED1 afou ektelestei
//kai na termatisei h diadikasia
PORTD.OUTCLR= PIN1_bm;
printf("%d",counter);
printf("%d",counter2);

}

//Se ayto to interrupt ginontai allages twn LED
//sumfwna me ton aisththora(ADC se emas) poy deixnei apostash apo ton toixo
//kai analoga thn poreia ths syskeuhs(kanonikh h anapodh dhladh)
ISR(ADC0_WCOMP_vect){
    int intflags = ADC0.INTFLAGS;
    ADC0.INTFLAGS = intflags; //edw thelei breakpoint kai
    //molis ftasei se auto to breakpoint to run thelei Allagh h timh
    //tou ADC se kati megalutero tou 10 gia na treksei mia fora
    //an den allaksei twra tote trexei perissoteres apo 1 fores
    if(interr==1){
        PORTD.OUTCLR= PIN1_bm; //kleinei to LED1 molis ektelestei
        PORTD.OUT |= PIN3_bm;//anoigei to LED3 molis ektelestei
        _delay_ms(5);
        PORTD.OUTCLR= PIN3_bm; //kleinei to LED3 molis ektelestei
        PORTD.OUT |= PIN1_bm;//anoigei to LED1 molis ektelestei
        counter--;
    }
    else if (interr==0){
        PORTD.OUTCLR= PIN1_bm; //kleinei to LED1 molis ektelestei
        PORTD.OUT |= PIN2_bm;//anoigei to LED2 molis ektelestei
        _delay_ms(5);
        PORTD.OUTCLR= PIN2_bm; //kleinei to LED2 molis ektelestei
        PORTD.OUT |= PIN1_bm;//anoigei to LED1 molis ektelestei
        counter++;
        counter2++;
    }
}

```

```

//Se auto to interrupt strivei deksia an erthei interrupt apo to
//PORTF PIN5 sthn kanonikh poreia h aristera an eimaste sthn anapodh
//poreia
ISR(PORTF_PORT_vect){
    //clear the interrupt flag
    int intflags = PORTF.INTFLAGS;
    PORTF.INTFLAGS=intflags;
    if(interr==1){
        PORTD.OUTCLR= PIN1_bm; //kleinei to LED1 molis ektelestei
        PORTD.OUT |= PIN2_bm;//anoigei to LED2 molis ektelestei
        _delay_ms(5);
        PORTD.OUTCLR= PIN2_bm; //kleinei to LED2 molis ektelestei
        PORTD.OUT |= PIN1_bm;//anoigei to LED1 molis ektelestei
        counter--;
    }
    else if (interr==0){
        PORTD.OUTCLR= PIN1_bm; //kleinei to LED1 molis ektelestei
        PORTD.OUT |= PIN3_bm;//anoigei to LED3 molis ektelestei
        _delay_ms(5);
        PORTD.OUTCLR= PIN3_bm; //kleinei to LED3 molis ektelestei
        PORTD.OUT |= PIN1_bm;//anoigei to LED1 molis ektelestei
        counter++;
        counter2--;
    }
}

//Se auto to interrupt ginetai anastrofth(strofth 180 moires afou anoigoun
//kai ta 3 LED)
//kai energopoieit hn anapodh poreia
ISR(PORTC_PORT_vect){
    //clear the interrupt flag
    int intflags = PORTC.INTFLAGS;
    PORTC.INTFLAGS=intflags;
    //To LED1 den xreiazetai na to anoikoume einai hdh anoixto
    PORTD.OUT |= PIN2_bm;//anoigei to LED2 molis ektelestei
    PORTD.OUT |= PIN3_bm;//anoigei to LED3 molis ektelestei
    _delay_ms(10);
    PORTD.OUTCLR= PIN2_bm; //kleinei to LED2 molis ektelestei
    PORTD.OUTCLR= PIN3_bm; //kleinei to LED3 molis ektelestei
    interr=1;
    counter--;
}

```

Αποτελέσματα Εκτέλεσης

Τα αποτελέσματα είναι τα ίδια με το βασικό μέρος της άσκησης, δηλαδή όλα λειτουργούν όπως πρέπει σε οποιοδήποτε σχήμα(που υπακούει στις παραδοχές που κάναμε στην αρχή). Τα σχήματα που ελέγχαμε την λειτουργία είναι τα εξής:

