**Προηγμένοι Μικροεπεξεργαστές**

**Εργαστηριακή Άσκηση 3**

**Μέλη Ομάδας**

**1ο μέλος: 2ο μέλος:**

**ΟΝΟΜΑ: ΛΟΥΚΑΣ ΟΝΟΜΑ: ΕΥΑΓΓΕΛΟΣ**

**ΕΠΩΝΥΜΟ: ΤΡΙΑΝΤΑΦΥΛΛΟΠΟΥΛΟΣ ΕΠΩΝΥΜΟ: ΜΠΕΝΕΤΟΣ**

**ΕΞΑΜΗΝΟ: 8 ΕΞΑΜΗΝΟ: 10**

**ΑΜ: 1084495 ΑΜ: 1072628**

1) Για την συγκεκριμένη άσκηση μας ζητήθηκε να δημιουργήσουμε, μέσω του προσομοιωτή, ένα πρόγραμμα που να εκτελεί τις βασικές λειτουργίες ενός οικιακού ανεμιστήρα. Οι δύο βασικές λειτουργίες που πρέπει να κάνει ο ανεμιστήρας είναι η περιστροφή των πλαστικών λεπίδων εντός της σχάρας, που παράγουν τον αέρα, και η περιστροφή του σώματος (βάσης). Για να το επιτύχουμε αυτό κάναμε χρήση μόνο δύο χρονιστών, όπως προαπαιτούσε η άσκηση, και συγκεκριμένα τον χρονιστή TCA0.SPLIT. Στον κώδικα που θα παραδώσουμε στην συνέχεια, ορίζουμε τρεις μεταβλητές που θα βοηθήσουν στην εκτέλεση του προγράμματος. Η πρώτη είναι η fan\_button, τύπου ακέραιου, που θα προσδιορίζει εάν είναι ανοικτός ή κλειστός ο ανεμιστήρας, καθώς και ποια λειτουργία εκτελεί όπως θα ζητηθεί παρακάτω. Η δεύτερη μεταβλητή, είναι η x και θα αξιοποιηθεί εντός των συναρτήσεων ISR ώστε να εξερχόμαστε από τις δομές επανάληψης που θα συναντάμε στην main. Η τρίτη μεταβλητή, η aux\_flag, είναι τύπου bool και θα υπάρχει για να εισερχόμαστε στην μεγάλη εξωτερική δομή επανάληψης while.

Για την main αρχικά ορίζουμε τα LEDs που αντιστοιχούν στις λειτουργίες που ζητούνται, δηλαδή το LED0 για τις πλαστικές λεπίδες και το LED1 για την βάση. Έπειτα εισερχόμαστε στην δομή while, όπου εκεί οφείλουμε να πατήσουμε να πατήσουμε το PIN5 στο INTFLAGS του PORTF ώστε να εισέλθει το πρόγραμμα στην συνάρτηση ISR του PIN5 και να αυξήσει τον μετρητή fan\_button κατά 1. Αφού εξέλθει της συνάρτησης αυτής, επιστρέφει στην main και εισέρχεται στην συνθήκη if(fan\_button == 1), όπου ενεργοποιούνται οι δύο χρονιστές, ο πρώτος o High count για τις πλαστικές λεπίδες με περίοδο 1ms (HPER = 20) και HCMP0 =10 (50% Duty Cycle), και ο Low count για την βάση με περίοδο 2ms (LPER = 40) και LCMP0 = 24 (60% Duty Cycle). Για καθένα από τους δύο χρονιστές, μετά την εκτέλεσή τους, το πρόγραμμα μεταβαίνει στις συναρτήσεις ISR για τις πλαστικές λεπίδες (HUNF) και για την βάση (LUNF). Με την εκτέλεση της ISR(TCA0\_HUNF\_vect) ανάβει το LED0 για τις λεπίδες με την εκτέλεση της ISR(TCA0\_LUNF\_vect) ανάβει το LED1 για την βάση.

Με όλες αυτές τις λεπτομέρειες που προαναφέρονται, στην συνέχεια παραθέτουμε τον κώδικα που τις εφαρμόζει:

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

int fan\_button = 0; //variable to define whether the device is on or off

int x = 0; //Flag for reset the ISRs

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

int main()

{

PORTD.DIR |= PIN1\_bm; //Set PIN1 as output for rotation of the base

PORTD.OUT |= PIN1\_bm; //LED1 for rotation of base is off

PORTD.DIR |= PIN0\_bm;//Set PIN0 as output for rotation of the fans(plastic blades)

PORTD.OUT |= PIN0\_bm; //LED0 for rotation of fans is off

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; // //Pullup enable bit mask enable bit mask and Sense Both Edges

//We activate the fan by pressing Switch5

TCA0.SPLIT.CTRLD |= TCA\_SPLIT\_SPLITM\_bm; //We enable the Split Mode

TCA0.SPLIT.HCNT = 0; //Clear Counter High Counter

TCA0.SPLIT.LCNT = 20; //Clear Counter Low Counter

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

while(aux\_flag)

{

x = 0;

sei();

if(fan\_button == 1)

{

//we are going to use high count as counter for fans(plastic blades)

TCA0.SPLIT.CTRLB = TCA\_SPLIT\_HCMP0EN\_bm; /\* High Compare 0 Enable bit mask. \*/

TCA0.SPLIT.HPER = 20; // Period value for the fans(plastic blades)

TCA0.SPLIT.HCMP0 = 10; // 50% duty cycle for the fans(plastic blades)

TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm; // Module Enable bit mask

TCA0.SPLIT.INTCTRL = TCA\_SPLIT\_HUNF\_bm; // High Underflow Interrupt Enable bit mask.

while(x == 0)

{

; //similar to a nop function

}

x = 0;

//we are going to use low count as counter for base

TCA0.SPLIT.CTRLB = TCA\_SPLIT\_LCMP0EN\_bm; /\* Low Compare 0 Enable bit mask. \*/

TCA0.SPLIT.LPER = 40; // Period value for the fans(plastic blades)

TCA0.SPLIT.LCMP0 = 24; // 60% duty cycle for the base

TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm; // Module Enable bit mask

TCA0.SPLIT.INTCTRL = TCA\_SPLIT\_LUNF\_bm; //Low Underflow Interrupt Enable bit mask.

while(x == 0)

{

; //similar to a nop function

}

x = 0;

cli();

}

}

ISR(PORTF\_PORT\_vect)

{

int y = PORTF.INTFLAGS; //Procedure to

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

fan\_button++; //fan is turned on

}

ISR(TCA0\_HUNF\_vect)

{

TCA0.SPLIT.CTRLA = 0; //Disable

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

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

if(fan\_button == 1)

{

PORTD.OUTCLR = PIN0\_bm; //LED0 for rotation of fans(plastic blades) is on

}

x = 1; //change flag to get out of the loop

}

ISR(TCA0\_LUNF\_vect)

{

TCA0.SPLIT.CTRLA = 0; //Disable

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

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

if(fan\_button == 1)

{

PORTD.OUTCLR = PIN1\_bm; //LED1 for rotation of base is on

}

x = 1; //change flag to get out of the loop

}

2) Έπειτα μας ζητείται να προσθέσουμε μια επιπλέον λειτουργία στον ανεμιστήρα. Σε περίπτωση που πατήσει κάποιος για δεύτερη φορά το κουμπί, τότε η περίοδος των πλαστικών λεπίδων θα υποδιπλασιαστεί, δηλαδή το HPER από 20 θα γίνει 10 αλλά το Duty Cycle θα είναι και πάλι 50%, άρα το HCMP0 από 10 θα γίνει 5. Το δεύτερο πάτημα του κουμπιού θα πρέπει να γίνει όσο ο debugger βρίσκεται στην δομή if(fan\_button == 1) και συγκεκριμένα μετά την while(x == 0) του χρονιστή της βάσης. Όταν το εκτελέσουμε με αυτόν τον τρόπο, θα εισέλθει για δεύτερη φορά στην ISR του PIN5 και το fan\_button θα αυξηθεί και θα γίνει 2. Μετά την πρώτη if, θα μεταβεί απευθείας στην δεύτερη if, την if(fan\_button == 2) και θα εκτελεστεί η ζητούμενη λειτουργία.

Επομένως, με βάση αυτές τις αλλαγές που έγιναν, παρατίθεται παρακάτω και ο νέος κώδικας:

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

int fan\_button = 0; //variable to define whether the device is on or off

int x = 0; //Flag for reset the ISRs

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

int main()

{

PORTD.DIR |= PIN1\_bm; //Set PIN1 as output for rotation of the base

PORTD.OUT |= PIN1\_bm; //LED1 for rotation of base is off

PORTD.DIR |= PIN0\_bm;//Set PIN0 as output for rotation of the fans(plastic blades)

PORTD.OUT |= PIN0\_bm; //LED0 for rotation of fans is off

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; // //Pullup enable bit mask enable bit mask and Sense Both Edges

//We activate the fan by pressing Switch5

TCA0.SPLIT.CTRLD |= TCA\_SPLIT\_SPLITM\_bm; //We enable the Split Mode

TCA0.SPLIT.HCNT = 0; //Clear Counter High Counter

TCA0.SPLIT.LCNT = 20; //Clear Counter Low Counter

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

while(aux\_flag)

{

x = 0;

sei();

if(fan\_button == 1)

{

//we are going to use high count as counter for fans(plastic blades)

TCA0.SPLIT.CTRLB = TCA\_SPLIT\_HCMP0EN\_bm; /\* High Compare 0 Enable bit mask. \*/

TCA0.SPLIT.HPER = 20; // Period value for the fans(plastic blades)

TCA0.SPLIT.HCMP0 = 10; // 50% duty cycle for the fans(plastic blades)

TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm; // Module Enable bit mask

TCA0.SPLIT.INTCTRL = TCA\_SPLIT\_HUNF\_bm; // High Underflow Interrupt Enable bit mask.

while(x == 0)

{

; //similar to a nop function

}

x = 0;

//we are going to use low count as counter for base

TCA0.SPLIT.CTRLB = TCA\_SPLIT\_LCMP0EN\_bm; /\* Low Compare 0 Enable bit mask. \*/

TCA0.SPLIT.LPER = 40; // Period value for the fans(plastic blades)

TCA0.SPLIT.LCMP0 = 24; // 60% duty cycle for the base

TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm; // Module Enable bit mask

TCA0.SPLIT.INTCTRL = TCA\_SPLIT\_LUNF\_bm; //Low Underflow Interrupt Enable bit mask.

while(x == 0)

{

; //similar to a nop function

}

x = 0;

}

if(fan\_button == 2)

{

//we are going to use high count as counter for fans(plastic blades)

TCA0.SPLIT.CTRLB = TCA\_SPLIT\_HCMP0EN\_bm; /\* High Compare 0 Enable bit mask. \*/

TCA0.SPLIT.HPER = 10; // Period value for the fans(plastic blades)

TCA0.SPLIT.HCMP0 = 5; // 50% duty cycle for the fans(plastic blades)

TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm; // Module Enable bit mask

TCA0.SPLIT.INTCTRL = TCA\_SPLIT\_HUNF\_bm; // High Underflow Interrupt Enable bit mask.

while(x == 0)

{

; //similar to a nop function

}

x = 0;

}

cli();

}

}

ISR(PORTF\_PORT\_vect)

{

int y = PORTF.INTFLAGS; //Procedure to

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

fan\_button++; //fan is turned on

}

ISR(TCA0\_HUNF\_vect)

{

TCA0.SPLIT.CTRLA = 0; //Disable

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

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

if(fan\_button == 1)

{

PORTD.OUTCLR = PIN0\_bm; //LED0 for rotation of fans(plastic blades) is on

}

x = 1; //change flag to get out of the loop

}

ISR(TCA0\_LUNF\_vect)

{

TCA0.SPLIT.CTRLA = 0; //Disable

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

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

if(fan\_button == 1)

{

PORTD.OUTCLR = PIN1\_bm; //LED1 for rotation of base is on

}

x = 1; //change flag to get out of the loop

}

3) Τέλος, προσθέτουμε στο πρόγραμμα και την τρίτο ζητούμενο, που είναι ο τερματισμός της λειτουργίας του ανεμιστήρα. Η προσθήκη που κάναμε στον κώδικα είναι μια ακόμα δομή if, η (if fan\_button == 3), στην οποία σβήνουμε τα δύο LEDs για να συμβολίσουμε τον τερματισμό της συσκευής και παράλληλα θέτουμε με 0 την μεταβλητή fan\_button καθώς και την εντολή TCA0.SPLIT.CTRLA, ώστε να μπορέσουμε να ξανατρέξουμε στο μέλλον το πρόγραμμα του ανεμιστήρα και να γίνονται ορθά οι ζητούμενες λειτουργίες. Για να επιτύχουμε τον τερματισμό του ανεμιστήρα, πατάμε το PIN5 όσο το πρόγραμμα βρίσκεται στην while(x == 0) της if(fan\_button == 2) έτσι ώστε όταν εξέλθει από αυτή την if εισέλθει αμέσως στην καινούργια if.

Επομένως, αφού κάνουμε και τις επιπλέον αυτές αλλαγές, παραθέτουμε στην συνέχεια, την τελική μορφή του κώδικα για αυτά που μας ζητήθηκαν:

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

int fan\_button = 0; //variable to define whether the device is on or off

int x = 0; //Flag for reset the ISRs

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

int main()

{

PORTD.DIR |= PIN1\_bm; //Set PIN1 as output for rotation of the base

PORTD.OUT |= PIN1\_bm; //LED1 for rotation of base is off

PORTD.DIR |= PIN0\_bm;//Set PIN0 as output for rotation of the fans(plastic blades)

PORTD.OUT |= PIN0\_bm; //LED0 for rotation of fans is off

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; // //Pullup enable bit mask enable bit mask and Sense Both Edges

//We activate the fan by pressing Switch5

TCA0.SPLIT.CTRLD |= TCA\_SPLIT\_SPLITM\_bm; //We enable the Split Mode

TCA0.SPLIT.HCNT = 0; //Clear Counter High Counter

TCA0.SPLIT.LCNT = 20; //Clear Counter Low Counter

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

while(aux\_flag)

{

x = 0;

sei();

if(fan\_button == 1)

{

//we are going to use high count as counter for fans(plastic blades)

TCA0.SPLIT.CTRLB = TCA\_SPLIT\_HCMP0EN\_bm; /\* High Compare 0 Enable bit mask. \*/

TCA0.SPLIT.HPER = 20; // Period value for the fans(plastic blades)

TCA0.SPLIT.HCMP0 = 10; // 50% duty cycle for the fans(plastic blades)

TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm; // Module Enable bit mask

TCA0.SPLIT.INTCTRL = TCA\_SPLIT\_HUNF\_bm; // High Underflow Interrupt Enable bit mask.

while(x == 0)

{

; //similar to a nop function

}

x = 0;

//we are going to use low count as counter for base

TCA0.SPLIT.CTRLB = TCA\_SPLIT\_LCMP0EN\_bm; /\* Low Compare 0 Enable bit mask. \*/

TCA0.SPLIT.LPER = 40; // Period value for the fans(plastic blades)

TCA0.SPLIT.LCMP0 = 24; // 60% duty cycle for the base

TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm; // Module Enable bit mask

TCA0.SPLIT.INTCTRL = TCA\_SPLIT\_LUNF\_bm; //Low Underflow Interrupt Enable bit mask.

while(x == 0)

{

; //similar to a nop function

}

x = 0;

}

if(fan\_button == 2)

{

//we are going to use high count as counter for fans(plastic blades)

TCA0.SPLIT.CTRLB = TCA\_SPLIT\_HCMP0EN\_bm; /\* High Compare 0 Enable bit mask. \*/

TCA0.SPLIT.HPER = 10; // Period value for the fans(plastic blades)

TCA0.SPLIT.HCMP0 = 5; // 50% duty cycle for the fans(plastic blades)

TCA0.SPLIT.CTRLA |= TCA\_SPLIT\_ENABLE\_bm; // Module Enable bit mask

TCA0.SPLIT.INTCTRL = TCA\_SPLIT\_HUNF\_bm; // High Underflow Interrupt Enable bit mask.

while(x == 0)

{

; //similar to a nop function

}

x = 0;

}

if(fan\_button == 3)

{

// We turn off the fans

PORTD.OUT |= PIN1\_bm; //LED1 for rotation of base is off

PORTD.OUT |= PIN0\_bm; //LED0 for rotation of fans is off

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

fan\_button = 0; // We reset the variable, so the fans can

// start all over again

}

cli();

}

}

ISR(PORTF\_PORT\_vect)

{

int y = PORTF.INTFLAGS; //Procedure to

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

fan\_button++; //fan is turned on

}

ISR(TCA0\_HUNF\_vect)

{

TCA0.SPLIT.CTRLA = 0; //Disable

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

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

if(fan\_button == 1)

{

PORTD.OUTCLR = PIN0\_bm; //LED0 for rotation of fans(plastic blades) is on

}

x = 1; //change flag to get out of the loop

}

ISR(TCA0\_LUNF\_vect)

{

TCA0.SPLIT.CTRLA = 0; //Disable

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

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

if(fan\_button == 1)

{

PORTD.OUTCLR = PIN1\_bm; //LED1 for rotation of base is on

}

x = 1; //change flag to get out of the loop

}





