**ΕΡΓΑΣΤΗΡΙΟ ΠΡΟΗΓΜΕΝΟΙ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΕΣ**

**Εργαστηριακή Άσκηση 3**

Δούρου Βασιλική Ευαγγελία – Α.Μ: 1072633

Πουρής Βασίλειος – Α.Μ.: 1072485

**1.** Κατά τη συγγραφή των παρακάτω έγιναν οι ακόλουθες παραδοχές:

* Στο δεύτερο και στο τρίτο ερώτημα αν ο ADC εντοπίσει κάποια τιμή μικρότερη του προκαθορισμένου κατωφλιού, η ροή του προγράμματος θα παραμείνει στην ISR του ADC μέχρι να δοθεί τιμή που ξεπερνάει το κατώφλι.

Ο κώδικας του πρώτου ερωτήματος της άσκησης για τη λειτουργία ενεργοποίησης του ανεμιστήρα μετά την ενεργοποίηση του διακόπτη είναι ο ακόλουθος:

#include <avr/io.h>

#include <avr/interrupt.h>

#define xlepides 20 //1ms

#define xvashs 40 //2ms

int timesl,timesv;

timesl=0;

timesv=0;

int button=0;

int main(){

PORTD.DIR |= PIN0\_bm; //lepides

PORTD.DIR |= PIN1\_bm; //vash

PORTD.OUT |=PIN0\_bm; //pin0 off

PORTD.OUT |=PIN1\_bm; //pin1 off

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

sei();

while(button==0){

}

TCA0.SPLIT.CTRLD = TCA\_SPLIT\_SPLITM\_bm; //ENABLE TCA SPLIT MODE

TCA0.SPLIT.CTRLB = TCA\_SPLIT\_LCMP0EN\_bm|TCA\_SPLIT\_HCMP0EN\_bm;

TCA0.SPLIT.HCNT = 0x02;

TCA0.SPLIT.HPER = xvashs;

TCA0.SPLIT.HCMP0 = 0x16;

TCA0.SPLIT.LCNT = 0x00;

TCA0.SPLIT.LPER = xlepides;

TCA0.SPLIT.LCMP0 = 0x10;

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

TCA0.SPLIT.INTCTRL =0x03; // hunf and lunf INTERRUPT ENABLE

while(1){

;

}

}

ISR(TCA0\_HUNF\_vect){

//clear the interrupt flag

timesv++;

if(timesv%2==0){

PORTD.OUT |= PIN1\_bm; //PIN is off

}

else{

PORTD.OUTCLR = PIN1\_bm; //PIN is on

}

int intflags = TCA0.SPLIT.INTFLAGS;

TCA0.SPLIT.INTFLAGS=intflags;

}

ISR(TCA0\_LUNF\_vect){

//clear the interrupt flag

timesl++;

if(timesl%2==0){

PORTD.OUT |= PIN0\_bm; //PIN is off

}

else{

PORTD.OUTCLR = PIN0\_bm; //PIN is on

}

int intflags = TCA0.SPLIT.INTFLAGS;

TCA0.SPLIT.INTFLAGS=intflags;

}

ISR(PORTF\_PORT\_vect){

//clear the interrupt flag

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

button=1;

}

Ο κώδικας του δεύτερου ερωτήματος μετά την προσθήκη του ADC και τη δυνατότητα επανεκκίνησης του ανεμιστήρα είναι ο ακόλουθος:

#include <avr/io.h>

#include <avr/interrupt.h>

#define xlepides 20 //1ms

#define xvashs 40 //2ms

int timesl,timesv;

int y=0;

timesl=0;

timesv=0;

int button=0;

int main(){

PORTD.DIR |= PIN0\_bm; //lepides

PORTD.DIR |= PIN1\_bm; //vash

PORTD.DIR |= PIN2\_bm; //elegxos gia adc

PORTD.OUT |=PIN0\_bm; //pin0 off

PORTD.OUT |=PIN1\_bm; //pin1 off

PORTD.OUT |=PIN2\_bm; //pin2 off

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

//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 |= 4; //Set threshold

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

ADC0.CTRLE |= ADC\_WINCM0\_bm; //Interrupt when RESULT < WINLT

sei();

while(1){

while(button==0){

}

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

TCA0.SPLIT.CTRLD = TCA\_SPLIT\_SPLITM\_bm; //ENABLE TCA SPLIT MODE

TCA0.SPLIT.CTRLB = TCA\_SPLIT\_LCMP0EN\_bm|TCA\_SPLIT\_HCMP0EN\_bm;

TCA0.SPLIT.HCNT = 0x02;

TCA0.SPLIT.HPER = xvashs;

TCA0.SPLIT.HCMP0 = 0x16;

TCA0.SPLIT.LCNT = 0x00;

TCA0.SPLIT.LPER = xlepides;

TCA0.SPLIT.LCMP0 = 0x10;

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

TCA0.SPLIT.INTCTRL =0x03; // hunf and lunf INTERRUPT ENABLE

while(y==0){

}

button=0;

y=0;

}

}

ISR(TCA0\_HUNF\_vect){

//clear the interrupt flag

timesv++;

if(timesv%2==0){

PORTD.OUT |= PIN1\_bm; //PIN is off

}

else{

PORTD.OUTCLR = PIN1\_bm; //PIN is on

}

int intflags = TCA0.SPLIT.INTFLAGS;

TCA0.SPLIT.INTFLAGS=intflags;

}

ISR(TCA0\_LUNF\_vect){

//clear the interrupt flag

timesl++;

if(timesl%2==0){

PORTD.OUT |= PIN0\_bm; //PIN is off

}

else{

PORTD.OUTCLR = PIN0\_bm; //PIN is on

}

int intflags = TCA0.SPLIT.INTFLAGS;

TCA0.SPLIT.INTFLAGS=intflags;

}

ISR(PORTF\_PORT\_vect){

//clear the interrupt flag

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

PORTD.OUT |= PIN2\_bm; //PIN is off

button=1;

}

ISR(ADC0\_WCOMP\_vect){

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

TCA0.SPLIT.CTRLA=0;

PORTD.OUTCLR = PIN2\_bm; //PIN is on

PORTD.OUT |= PIN0\_bm; //PIN is off

PORTD.OUT |= PIN1\_bm; //PIN is off

timesl=0;

timesv=0;

y=1;

}

Ο κώδικας του τρίτου ερωτήματος μετά την προσθήκη της δεύτερης λειτουργίας του παλμού της κυκλικής κίνησης των λεπίδων μέσω της ενεργοποίησης του διακόπτη και της απενεργοποίησης του ανεμιστήρα είναι ο ακόλουθος:

#include <avr/io.h>

#include <avr/interrupt.h>

#define xlepides 20 //1ms

#define xlepidesepi2 40 //2ms

#define xvashs 40 //2ms

int timesl,timesv;

int x=0;

timesl=0;

timesv=0;

int button=0;

int main(){

PORTD.DIR |= PIN0\_bm; //lepides

PORTD.DIR |= PIN1\_bm; //vash

PORTD.DIR |= PIN2\_bm; //elegxos gia adc

PORTD.OUT |=PIN0\_bm; //pin0 off

PORTD.OUT |=PIN1\_bm; //pin1 off

PORTD.OUT |=PIN2\_bm; //pin2 off

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

//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 |= 4; //Set threshold

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

ADC0.CTRLE |= ADC\_WINCM0\_bm; //Interrupt when RESULT < WINLT

sei();

while(1){

while(button==0){

}

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

TCA0.SPLIT.CTRLD = TCA\_SPLIT\_SPLITM\_bm; //ENABLE TCA SPLIT MODE

TCA0.SPLIT.CTRLB = TCA\_SPLIT\_LCMP0EN\_bm|TCA\_SPLIT\_HCMP0EN\_bm;

TCA0.SPLIT.HCNT = 0x02;

TCA0.SPLIT.HPER = xvashs;

TCA0.SPLIT.HCMP0 = 0x16;

TCA0.SPLIT.LCNT = 0x00;

TCA0.SPLIT.LPER = xlepides;

TCA0.SPLIT.LCMP0 = 0x10;

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

TCA0.SPLIT.INTCTRL =0x03; // hunf and lunf INTERRUPT ENABLE

while(x==0){

}

button=0;

x=0;

}

}

ISR(TCA0\_HUNF\_vect){

//clear the interrupt flag

timesv++;

if(timesv%2==0){

PORTD.OUT |= PIN1\_bm; //PIN is off

}

else{

PORTD.OUTCLR = PIN1\_bm; //PIN is on

}

int intflags = TCA0.SPLIT.INTFLAGS;

TCA0.SPLIT.INTFLAGS=intflags;

}

ISR(TCA0\_LUNF\_vect){

//clear the interrupt flag

timesl++;

if(timesl%2==0){

PORTD.OUT |= PIN0\_bm; //PIN is off

}

else{

PORTD.OUTCLR = PIN0\_bm; //PIN is on

}

int intflags = TCA0.SPLIT.INTFLAGS;

TCA0.SPLIT.INTFLAGS=intflags;

}

ISR(PORTF\_PORT\_vect){

//clear the interrupt flag

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

PORTD.OUT |= PIN2\_bm; //PIN is off

button++;

if(button==2){

TCA0.SPLIT.LPER = xlepidesepi2;

TCA0.SPLIT.LCMP0 = 0x10;

}

if(button==3){

TCA0.SPLIT.CTRLA=0;

x=1;

PORTD.OUT |= PIN0\_bm; //PIN is off

PORTD.OUT |= PIN1\_bm; //PIN is off

timesl=0;

timesv=0;

}

}

ISR(ADC0\_WCOMP\_vect){

int intflags = ADC0.INTFLAGS;

ADC0.INTFLAGS = intflags;

TCA0.SPLIT.CTRLA=0;

PORTD.OUTCLR = PIN2\_bm; //PIN is on

PORTD.OUT |= PIN0\_bm; //PIN is off

PORTD.OUT |= PIN1\_bm; //PIN is off

timesl=0;

timesv=0;

x=1;

}

**2.** Το διάγραμμα ροής του πρώτου ερωτήματος:

Εικόνα που περιέχει διάγραμμα, τεχνικό σχέδιο, Σχέδιο, σκίτσο/σχέδιο

Περιγραφή που δημιουργήθηκε αυτόματα

Το διάγραμμα ροής του δεύτερου ερωτήματος:

Εικόνα που περιέχει διάγραμμα, τεχνικό σχέδιο, κείμενο, Σχέδιο

Περιγραφή που δημιουργήθηκε αυτόματα

Το διάγραμμα ροής του τρίτου ερωτήματος:

Εικόνα που περιέχει διάγραμμα, τεχνικό σχέδιο, Σχέδιο, σκίτσο/σχέδιο

Περιγραφή που δημιουργήθηκε αυτόματα

Στον κώδικα του πρώτου ερωτήματος, αρχικά, ορίζονται οι τρεις μεταβλητές timesl, timesv και button, που αρχικοποιούνται με μηδέν. Οι δύο πρώτες μεταβλητές χρησιμοποιούνται για τον υπολογισμό των rising edge του κάθε παλμού, ενώ η μεταβλητή button χρησιμοποιείται ως flag για το πότε έχει πατηθεί το κουμπί. Επίσης ορίζονται το PIN1 και το PIN0 του PORTD ως έξοδοι και τίθενται ίσα με 1 για να είναι κλειστά τα LED τους. Στη συνέχεια, γίνεται enable το pullup και το interrupt για το PIN5 του PORTF, ώστε αυτό να χρησιμοποιηθεί ως το κουμπί ενεργοποίησης του ανεμιστήρα. Έπειτα, ενεργοποιούνται οι διακοπές και όσο δεν έχει πατηθεί το κουμπί του ανεμιστήρα, η ροή του προγράμματος παραμένει σε ένα while loop. Μόλις πατηθεί, θα γίνει διακοπή και θα καλεστεί η ISR του όπου η τιμή της μεταβλητής button θα γίνει ίση με 1. Ακολούθως, γίνεται enable το split mode στον TCA και επιλέγεται η σύγκριση των τιμών των δύο 8-bit timers να γίνεται στα κανάλια HCMP0 και LCMP0. Αρχικοποιείται ο high byte counter με μία μικρή καθυστέρηση, η περίοδος των high bytes με την τιμή της περιόδου του παλμού των λεπίδων και το HCMP0 με την μισή τιμή της περιόδου (duty cycle 50%). Παρόμοια, αρχικοποιείται ο low byte counter με μηδέν, η περίοδος των low bytes με την τιμή της περιόδου του παλμού της βάσης και το LCMP0 με το 40% της τιμής της περιόδου (duty cycle 40%). Τέλος, ορίζεται η τιμή του prescaler και γίνεται enable o timer και το interrupt για το HUNF και το LUNF. Όταν γίνει underflow στα low bytes του timer, δηλαδή όταν έχουμε rising edge, θα καλεστεί η ISR του LUNF, στη διάρκεια της οποίας θα καθαριστούν τα flags, θα αυξηθεί η τιμή της μεταβλητής timesl κατά ένα και με τη χρήση της συνάρτησης mod2 επιτυγχάνεται η κατάλληλη ενεργοποίηση και απενεργοποίηση του LED, όπως ορίζεται από την εκφώνηση της άσκησης (σε κάθε rising edge εναλλαγή της κατάστασης του LED). Με παρόμοιο τρόπο, όταν γίνει underflow στα high bytes του timer, δηλαδή όταν έχουμε rising edge, θα καλεστεί η ISR του HUNF, στη διάρκεια της οποίας θα καθαριστούν τα flags, θα αυξηθεί η τιμή της μεταβλητής timesv κατά ένα και με τη χρήση της συνάρτησης mod2 επιτυγχάνεται η κατάλληλη ενεργοποίηση και απενεργοποίηση του LED.

Στο δεύτερο ερώτημα, αρχικά, ορίζονται οι τέσσερις μεταβλητές timesl, timesv, y και button, που αρχικοποιούνται με μηδέν. Οι δύο πρώτες μεταβλητές χρησιμοποιούνται για τον υπολογισμό των rising edge του κάθε παλμού, ενώ η μεταβλητή y χρησιμοποιείται ως flag για το πότε έχει έρθει τιμή μικρότερη του threshold στον ADC και η button χρησιμοποιείται ως flag για το πότε έχει πατηθεί το κουμπί. Επίσης, ορίζονται το PIN2, το PIN1 και το PIN0 του PORTD ως έξοδοι και τίθενται ίσα με 1 για να είναι κλειστά τα LED τους. Στη συνέχεια, γίνεται enable το pullup και το interrupt για το PIN5 του PORTF, ώστε αυτό να χρησιμοποιηθεί ως το κουμπί ενεργοποίησης του ανεμιστήρα, και αρχικοποιείται ο ADC σε free-running mode και ενεργοποιούνται οι διακοπές όταν ισχύει RES<WINLT. Ακολούθως, ενεργοποιούνται οι διακοπές και το ακόλουθο κομμάτι του κώδικα επαναλαμβάνεται διαρκώς. Αρχίζει η μετατροπή στον ADC και ο κώδικας είναι ίδιος με το πρώτο ερώτημα, με μοναδική προσθήκη την απενεργοποίηση του LED2 (PIN2 του PORTD=1) που γίνεται στην ISR του PORTF, μέχρι να υπάρξει interrupt από τον ADC. Στην περίπτωση που προκληθεί διακοπή από τον ADC θα κληθεί η ISR του, στη διάρκεια της οποίας θα καθαριστούν τα flags, θα απενεργοποιηθεί ο TCA, θα γίνει ίσο με μηδέν το PIN2 του PORTD (για να ανάψει το LED), θα γίνουν ίσα με ένα τα PIN1 και PIN0 του PORTD (για να σβήσουν τα αντίστοιχα LED), θα μηδενιστούν οι timesl και timesv και η τιμή του y θα γίνει ίση με ένα προκειμένου να βγούμε από το while loop. Τότε θα μηδενιστούν οι τιμές των μεταβλητών button και y.

Ο κώδικας του τρίτου ερωτήματος είναι ίδιος με αυτού του δεύτερου ερωτήματος με μοναδική διαφορά την ISR του PORTF. Σε αυτή την περίπτωση, σε περίπτωση διακοπής, το button δεν γίνεται ίσο με ένα αλλά αυξάνεται κατά ένα κάθε φορά. Αν η τιμή της μεταβλητής button είναι ίση με δύο, δηλαδή τη δεύτερη φορά που πατηθεί το κουμπί, αλλάζει η περίοδος των low bytes του TCA και το LCMP0 (για να έχουμε duty cycle 50%). Αν η τιμή της μεταβλητής button είναι ίση με τρία, δηλαδή την τρίτη φορά που πατηθεί το κουμπί, απενεργοποιείται ο TCA, τίθενται οι τιμές του PIN0 και του PIN1 ίσες με ένα για να απενεργοποιηθούν τα δύο LED και μηδενίζονται οι μεταβλητές timesv και timesl.

Οι τιμές που προέκυψαν για την κυκλική κίνηση της βάσης και των λεπίδων με Tb=2ms και Tl=1ms αντίστοιχα είναι οι εξής:

και . Οι τιμές αυτές προέκυψαν από τον τύπο , όπου Τ είναι ο χρόνος του χρονιστή και είναι η συχνότητα του ρολογιού, όπως προκύπτει από τον τύπο , με =20MHz και Ν=1024.

**3.** Οι παλμοί που προέκυψαν είναι οι ακόλουθοι:

Εικόνα που περιέχει διάγραμμα, γραμμή, μοτίβο

Περιγραφή που δημιουργήθηκε αυτόματα

Στον παλμό 1, δηλαδή τον παλμό της κυκλικής κίνησης των λεπίδων, ισχύει duty cycle=50%.

Στον παλμό 2, δηλαδή τον παλμό της κυκλικής κίνησης της βάσης, ισχύει duty cycle=40%.

Στη διακεκομμένη γραμμή είναι η τιμή του CMP. Στα “Χ” γίνεται match με την τιμή του CMP, ώστε να προκύψει falling edge, ενώ στα “Ο” γίνεται update, αφού έχει περάσει μία περίοδος και προκύπτει rising edge.