Εικόνα που περιέχει έμβλημα, σύμβολο, κύκλος, Εμπορικό σήμα

Περιγραφή που δημιουργήθηκε αυτόματα

Εργαστήριο Προηγμένοι Μικροεπεξεργαστές

**3η Εργαστηριακή Άσκηση : Εξοικείωση με την Παλμοευρική Διαμόρφωση (Pulse-Width Modulation – PWM)**

***Μέλη Ομάδας :***

***Μουσελέ Χριστιάνα ΑΜ: 1090068***

***Μπαλάσης Ιωάννης ΑΜ: 1084631***

Για την υλοποίηση των ερωτημάτων έχουν υπολογιστεί και ληφθεί υπόψην τα εξής:

* Υπολογισμός τιμής value για τις λεπίδες και τη βάση σύμφωνα με την θεωρία.

Λεπίδες (Blades):

Βάση(Base):

* Οριμός High και Low λειτουργίας σύμφωνα με τα δεδομένα της εκφώνησης

Λεπίδες(Blades): DC 50% οπότε 50% ως High και 50% ως Low

Βάση(Base): DC 60% οπότε 60% ως High και 40% ως Low

* Επίσης έχουν ληφθεί υπόψην οι παρακάτω τύποι:

Βάση δεδομένων και τιμών Value το χρονικό διάστημα της κάθε κίνησης σε High λειτουργία :

**Κώδικας 1ου Ερωτήματος**

#include <avr/io.h>

#include <avr/interrupt.h>

#define bladetime 20 //1ms

#define basetime 40 //2ms

int blade = 0; //for rising edge

int base = 0; // for rising edge

int button = 0; //flag for pressed switch

int main() {

PORTD.DIR |= PIN0\_bm; //Output PIN for Blade

PORTD.DIR |= PIN1\_bm; //Output PIN for Base

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

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

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //Set PIN5 PORTF for Switch

sei(); //Enable Interrupt

//While Button Value is 0 ,PINS are off

while(button==0){

}

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

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

TCA0.SPLIT.HCNT = 0x02; //High Bit Counter , add delay. Not from the beggining

TCA0.SPLIT.HPER = basetime; //Period for Base

TCA0.SPLIT.HCMP0 = 0x10; //50% of Blade Time

TCA0.SPLIT.LCNT = 0x00; //Low Bit Counter

TCA0.SPLIT.LPER = bladetime; //Period for Blade

TCA0.SPLIT.LCMP0 = 0x24; //60% of Base Time (40)

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

TCA0.SPLIT.INTCTRL =0x03; // HUNF and LUNF INTERRUPT ENABLE, it means after 3ms (1 ms for Blade and 2 ms for Base (also Base rotate starting with 1 ms delay))

while(1){

;

}

}

//Base Underflow Interrupt

ISR(TCA0\_HUNF\_vect){

base++; //Increase variable

PORTD.OUT ^= PIN1\_bm; //XOR: Toggle PIN status

//Clear the Interrupt Flag

int intflags = TCA0.SPLIT.INTFLAGS;

TCA0.SPLIT.INTFLAGS=intflags;

}

//Blade Underflow Interrupts

ISR(TCA0\_LUNF\_vect){

blade++; //Increase variable

PORTD.OUT ^= PIN0\_bm; //XOR: Toggle PIN status

//Clear the Interrupt Flag

int intflags = TCA0.SPLIT.INTFLAGS;

TCA0.SPLIT.INTFLAGS=intflags;

}

ISR(PORTF\_PORT\_vect){

//When Switch pressed, clear flags

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

button=1;

}

**Εξήγηση Κώδικα 1ου Ερωτήματος**

Στην αρχή του κώδικα, ορίζονται οι χρονικές περίοδοι για τις περιστροφικές κινήσεις των λεπίδων και της βάσης του ανεμιστήρα, καθώς και τις μεταβλητές base, blade και button. Οι δύο πρώτες χρησιμοποιούνται για την καταγραφή του Rising Edge κάθε κινήσεως, ενώ η τρίτη για χρησιμοποιείται ως Flag για τον διακόπτη. Στη συνέχεια ορίζονται οι έξοδοι PIN0 και PIN2 που θα ελέγχουν τις περιστροφικές κινήσεις και θέτονται ίσες με 1 έτσι ώστε να είναι κλειστά τα αντίστοιχα LED. Έπειτα , κάνουμε Enable το PULLUP και το INTERRUΡΤ του PIN5 PORTF, για να το ορίσουμε ως διακόπτη (κουμπί) ενεργοποίησης του ανεμιστήρα. Ενεργοποιούνται οι διακοπές και για όσο διάστημα δεν έχει ενεργοποιηθεί ο διακόπτης (κουμπί) παραμένουμε μέσα σε ένα While Loop. Γίνεται Enable SPLIT MODE στον ΤCA και επιλέχθηκε η σύγκριση τιμών των δύο 8-bit Timers να γίνεται στο HCMP0 και LCMP0. Αρχικοποιούμε τον High Byte Counter με μια μικρή καθυστέρηση (ξεκινάει πρώτα η περιστροφική κίνηση λεπίδων) , την περίοδο του ίση με το παλμό της περιστροφικής κίνησης της βάσης και το ΗCMP0 με το Duty Cycle των λεπίδων. Με τον ίδιο τρόπο ορίζουμε τον Low Byte Counter ίσο με 0 , την περίοδο των low bytes ίση με το παλμό των λεπίδων και το LCMP0 ίσο με το 60% του Base. Τέλος, ορίζεται η διακοπή για τις περιστροφικές κινήσεις (LUNF-HUNF) , ορίζεται η τιμή του Prescaler και γίνεται Enable ο Τimer. Όταν παρουσιαστεί Rising Edge θα γίνει κλήση της ΙSR του LUNF (Λεπίδες-UnderFlow Timer Low Bytes) κατά την οποία θα γίνει καθαρισμός των Flags και θα γίνει αύξηση της μεταβλητής blade και η κατάσταση του PIN εναλλάσσεται χρησιμοποιώντας την λογική XOR (^=). Αυτό σημαίνει ότι αν η κατάσταση του PIN είναι ήδη HIGH, τότε γίνεται LOW, και αν είναι LOW, τότε γίνεται HIGH. Αυτό δημιουργεί μια τακτική αλλαγή της κατάστασης του PIN κάθε φορά που εκτελείται η διακοπή υπερχείλισης.(εναλλαγή της κατάστασης του LED). Το ίδιο γίνεται και όταν γίνει κλήση της ISR του HUNF (Βάση-UnderFlow Timer High Bytes).

**Διάγραμμα Ροής 1ου Ερωτήματος**

**Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, κύκλος, διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα**

**Κώδικας 2ου Ερωτήματος**

#include <avr/io.h>

int main(void)

{

#include <avr/io.h>

#include <avr/interrupt.h>

#define bladetime 20 //1ms

#define basetime 40 //2ms

#define newbladetime 10 // divisible by 2

int blade = 0; //for rising edge

int base = 0; // for rising edge

int button = 0; //flag for pressed switch

int main() {

PORTD.DIR |= PIN0\_bm; //Output PIN for Blade

PORTD.DIR |= PIN1\_bm; //Output PIN for Base

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

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

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //Set PIN5 PORTF for Switch

sei(); //Enable Interrupt

while(button==0){

}

//TIMER

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

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

TCA0.SPLIT.HCNT = 0x01; //High Bit Counter , add delay means that's start after 1. Not from the beggining

TCA0.SPLIT.HPER = basetime; //Period for Base

TCA0.SPLIT.HCMP0 = 0x10; //50% of Blade Time

TCA0.SPLIT.LCNT = 0x00; //Low Bit Counter

TCA0.SPLIT.LPER = bladetime; //Period for Blade

TCA0.SPLIT.LCMP0 = 0x24; //60% of Base Time

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

TCA0.SPLIT.INTCTRL =0x03; // HUNF and LUNF INTERRUPT ENABLE, it means after 3ms (1 ms for Blade and 2 ms for Base (also Base rotate starting with 1 ms delay))

while(1){

;

}

//Base Underflow Interrupt

ISR(TCA0\_HUNF\_vect){

base++; //Increase variable

PORTD.OUT ^= PIN1\_bm; //XOR: Toggle PIN status

//Clear the Interrupt Flag

int intflags = TCA0.SPLIT.INTFLAGS;

TCA0.SPLIT.INTFLAGS=intflags;

}

//Blade Underflow interrupts

ISR(TCA0\_LUNF\_vect){

blade++; //Increase variable

PORTD.OUT ^= PIN0\_bm; //XOR: Toggle PIN status

//Clear the Interrupt Flag

int intflags = TCA0.SPLIT.INTFLAGS;

TCA0.SPLIT.INTFLAGS=intflags;

}

ISR(PORTF\_PORT\_vect){

//When Switch pressed, clear flags

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

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

button=++;

if(button=2){

TCA0\_SPLIT\_LPER = newbladetime;

TCA0\_SPLIT\_LCMP0 = 0x05; //50% of the new time

}

}

**Εξήγηση Κώδικα 2ου Ερωτήματος**

Ο κώδικας στο δεύτερο ερώτημα είναι ο ίδιος με τον κώδικα του πρώτου , ωστόσο στην ISR(PORTF\_PORT\_vect) , η μεταβλητή η οποία αναφέρεται στο Switch αυξάνεται κατά 1 κάθε φορά και μέσα από έναν βρόγχο if ελέγχουμε τις λεπίδες. Σύμφωνα με την εκφώνηση της άσκησης όταν πατηθεί δεύτερη φορά το κουμπί ο παλμός της υποδιπλασιάζεται ενώ ότι αφορά την περιστροφική κίνηση της βάσης παραμένει ως έχει.

**Διάγραμμα Ροής 2ου Ερωτήματος**

Εικόνα που περιέχει στιγμιότυπο οθόνης, κείμενο, διάγραμμα, σχεδίαση

Περιγραφή που δημιουργήθηκε αυτόματα

**Κώδικας 3ου Ερωτήματος**

#include <avr/io.h>

#include <avr/interrupt.h>

#define bladetime 20 //1ms

#define basetime 40 //2ms

#define newbladetime 10 // divisible by 2

int blade = 0; //for rising edge

int base = 0; // for rising edge

int button = 0; //flag for pressed switch

int main() {

PORTD.DIR |= PIN0\_bm; //Output PIN for Blade

PORTD.DIR |= PIN1\_bm; //Output PIN for Base

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

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

PORTF.PIN5CTRL |= PORT\_PULLUPEN\_bm | PORT\_ISC\_BOTHEDGES\_gc; //Set PIN5 PORTF for Switch

sei(); //Enable Interrupt

while(1) {

while(button==0){

;

}

//TIMER

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

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

TCA0.SPLIT.HCNT = 0x02; //High Bit Counter , add delay means that's start after 1. Not from the beggining

TCA0.SPLIT.HPER = basetime; //Period for Base

TCA0.SPLIT.HCMP0 = 0x10; //50% of Blade Time

TCA0.SPLIT.LCNT = 0x00; //Low Bit Counter

TCA0.SPLIT.LPER = bladetime; //Period for Blade

TCA0.SPLIT.LCMP0 = 0x24; //60% of Base Time

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

TCA0.SPLIT.INTCTRL =0x03; // HUNF and LUNF INTERRUPT ENABLE, it means after 3ms (1 ms for Blade and 2 ms for Base (also Base rotate starting with delay))

while(button==1 | button==2){

;

}

}

}

//Base Underflow Interrupt

ISR(TCA0\_HUNF\_vect){

cli();

base++; //Increase variable

PORTD.OUT ^= PIN1\_bm; //XOR: Toggle PIN status

//Clear the Interrupt Flag

int intflags = TCA0.SPLIT.INTFLAGS;

TCA0.SPLIT.INTFLAGS=intflags;

sei();

}

//Blade Underflow Interrupt

ISR(TCA0\_LUNF\_vect){

cli();

blade++; //Increase variable

PORTD.OUT ^= PIN0\_bm; //XOR: Toggle PIN status

//Clear the Interrupt Flag

int intflags = TCA0.SPLIT.INTFLAGS;

TCA0.SPLIT.INTFLAGS=intflags;

sei();

}

ISR(PORTF\_PORT\_vect){

//When Switch pressed, clear flags

int y = PORTF.INTFLAGS;

PORTF.INTFLAGS=y;

button++;

if(button==2){

TCA0\_SPLIT\_LPER = newbladetime;

TCA0\_SPLIT\_LCMP0 = 0x05; //50% of the new time

}

if(button==3){

TCA0.SPLIT.CTRLA=0;

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

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

button=0;

}

}

**Εξήγηση Κώδικα 3ου Ε ρωτήματος**

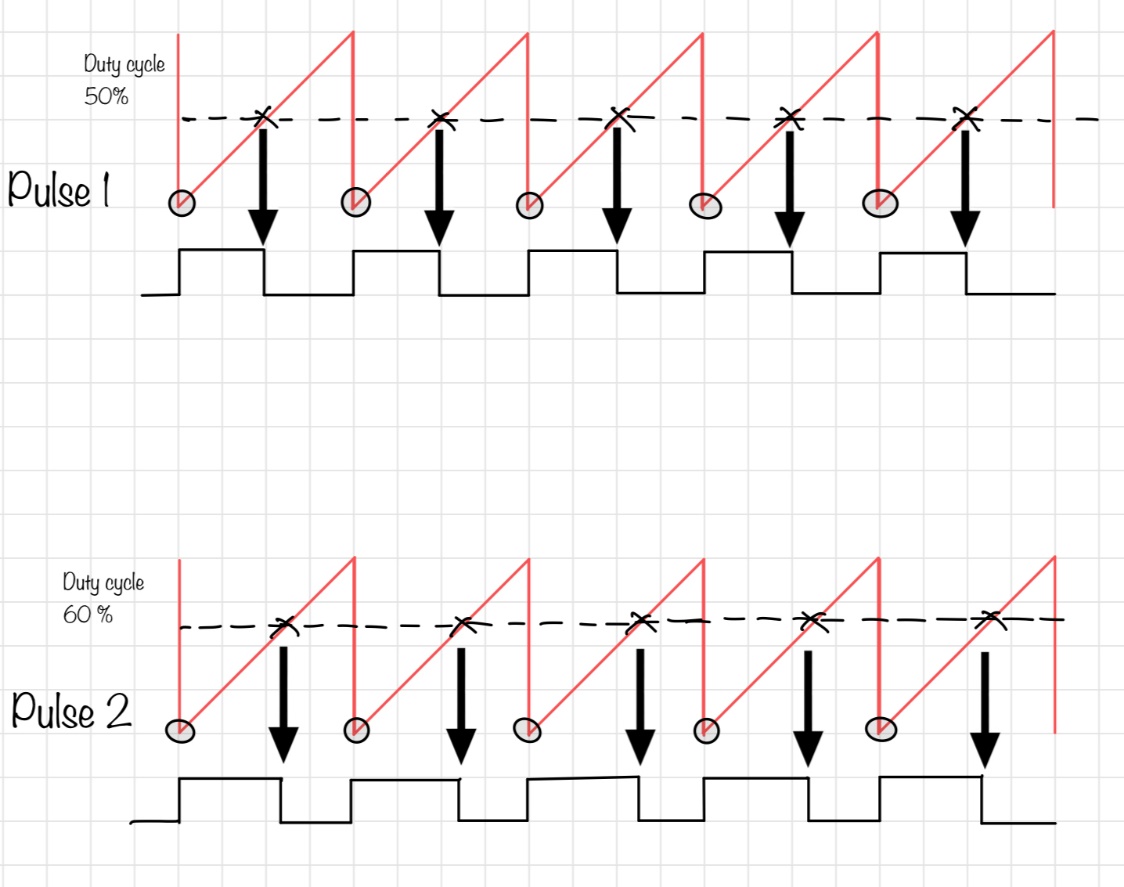
Στο τρίτο ερώτημα της άσκησης έπρεπε να προσθέσουμε την λειτουργία της απενεργοποίησης του ανεμιστήρα την τρίτη φορά που θα πατηθεί το κουμπί, και δυνατότητα ενεργοποίησης του έπειτα. Αρχικά συμπεριλάβαμε τα while(button==0) και while(button==1 | button==2) σε ένα αρχικό while(1) έτσι ώστε να μπορέσει μετά την απενεργοποίηση να ξανά ενεργοποιηθεί σε οποιαδήποτε στιγμή ο ανεμιστήρας. Έπειτα για την απενεργοποίηση του χρησιμοποιήσαμε στην ISR του PORTF\_PORT\_vect έναν επιπλέον έλεγχο if ο οποίος ελέγχει αν το button=3 , δηλαδή αν έχει πατηθεί τρίτη φορά το κουμπί και αν ναι τότε μηδενίζει τον TCA0 , κλείνει τα led και θέτει το button=0 . Αυτό έχει ως αποτέλεσμα να βγούμε από το δεύτερο while και να οδηγηθούμε πάλι στο πρώτο ( while(button==0) ). Το πρόγραμμα μας θα παραμείνει εκεί μέχρι να πατηθεί πάλι το κουμπί και να ενεργοποιηθεί εξ αρχής ο ανεμιστήρας.

**Διάγραμμα Ροής 3ου Ερωτήματος**

**Εικόνα που περιέχει στιγμιότυπο οθόνης, διάγραμμα, ασπρόμαυρο, σχεδίαση

Περιγραφή που δημιουργήθηκε αυτόματα**

Οι παλμοί που προέκυψαν είναι οι ακόλουθοι :



Ο παλμός 1 είναι ο παλμός της κυκλικής κίνησης των λεπίδων και ο παλμός 2 είναι της κυκλικής κίνησης της βάσης . Τα duty cycles είναι 50% και 60% αντίστοιχα .Στην διακεκομμένη γραμμή είναι η τιμή του CMP όπου και δηλώνονται τα duty cycles. Όταν είμαστε στα Χ σημαίνει ότι έχουμε φτάσει στην τιμή του duty cycle και άρα γίνεται falling edge. Στα Ο ξεκινάει η καινούργια περίοδος και έχουμε raising edge.