



Πολυτεχνική Σχολή
Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Προηγμένοι Μικροεπεξεργαστές

Εργαστηριακή Άσκηση 3

*Κατσαρός Ανδρέας 1084522
Ποταμιάνος Άγγελος Νικόλαος 1084537*

Πάτρα, 2023-24

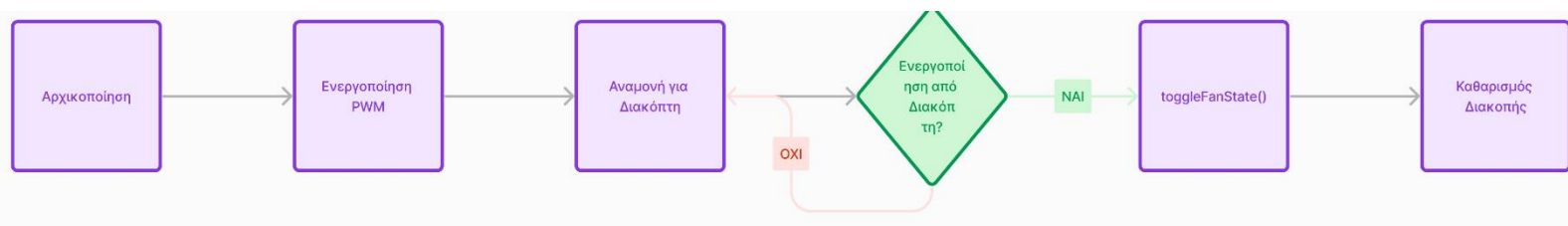
Ερωτήματα Εργαστηριακής Άσκησης 3

- 1 Υλοποιείτε τη λειτουργία ενεργοποίησης του ανεμιστήρα μετά την ενεργοποίηση του διακόπτη (switch 5), δηλαδή οι δύο παλμοί PWM αρχικοποιούνται και οδηγούν τα δύο LEDs.
 - 2 Προσθέστε τη δεύτερη λειτουργία του παλμού της κυκλικής κίνησης των λεπίδων μέσω της ενεργοποίησης του διακόπτη (switch 5).
 - 3 Επίσης, προσθέστε τη λειτουργία απενεργοποίησης του ανεμιστήρα με δυνατότητα ενεργοποίησής του εκ νέου σε επόμενο πάτημα του διακόπτη.
-

Μπορείτε να έχετε πρόσβαση και σε ολόκληρο τον κώδικα στο τέλος της αναφοράς .
Παρακάτω επεξηγούμε αναλυτικά τον κώδικα για την υλοποίηση των ερωτημάτων .

Ερώτημα 1:

Παράθεση Διαγράμματος Ροής :



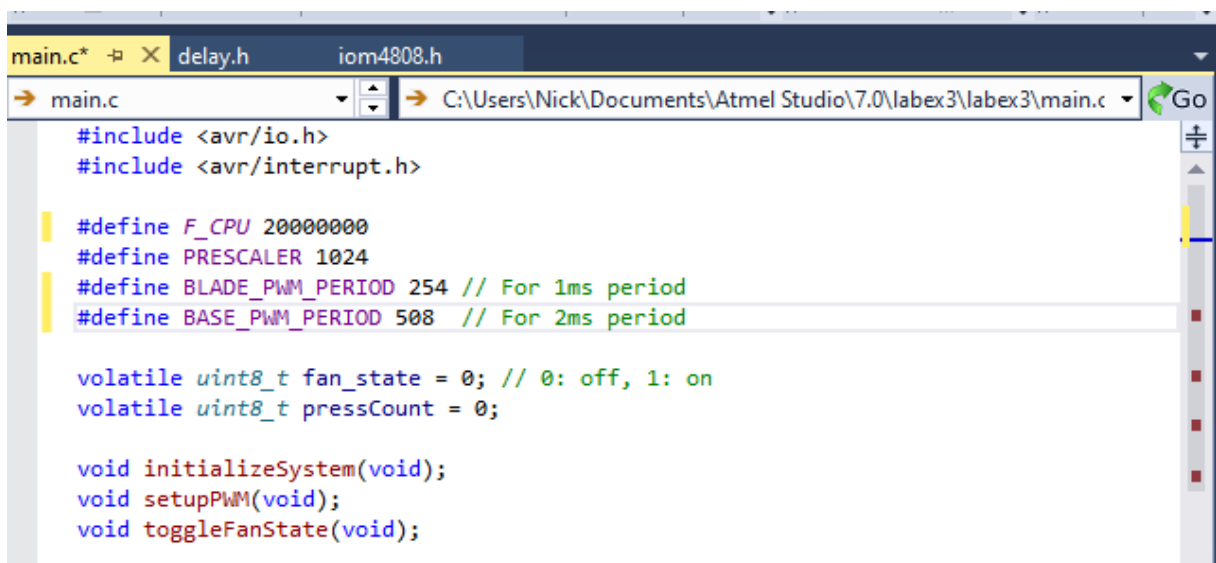
Σκοπός του κώδικα:

Σκοπός μας είναι να επιτρέψουμε την ενεργοποίηση και διαχείριση ενός ανεμιστήρα μέσω της χρήσης PWM (Pulse Width Modulation) σημάτων. Μέσω της χρήσης interrupts, ο

κώδικας επιτρέπει την άμεση ανταπόκριση σε ενέργειες του χρήστη, όπως το πάτημα ενός διακόπτη, αλλάζοντας την κατάσταση του ανεμιστήρα (ενεργοποίηση, αλλαγή ταχύτητας, απενεργοποίηση).

Επεξήγηση βασικών σημείων του κώδικα:

Για την ενεργοποίηση του ανεμιστήρα, απαιτείται η δημιουργία δύο διαφορετικών PWM σημάτων, τα οποία θα ελέγχουν ανεξάρτητα την κίνηση των λεπίδων και της βάσης του ανεμιστήρα. Η λογική πίσω από αυτό είναι η ακριβής ρύθμιση της ταχύτητας και της διεύθυνσης κίνησης, μέσω της προσαρμογής της διάρκειας των υψηλών και χαμηλών σημείων του κάθε παλμού.



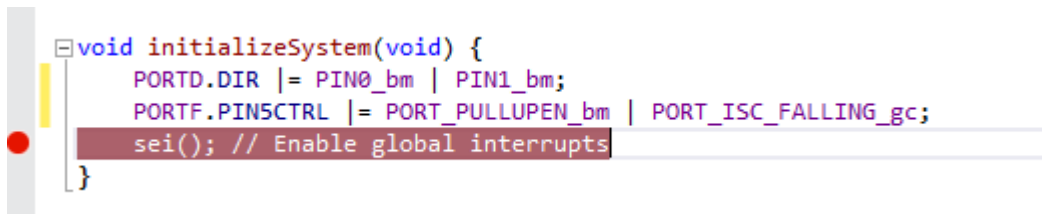
```
#include <avr/io.h>
#include <avr/interrupt.h>

#define F_CPU 2000000
#define PRESCALER 1024
#define BLADE_PWM_PERIOD 254 // For 1ms period
#define BASE_PWM_PERIOD 508 // For 2ms period

volatile uint8_t fan_state = 0; // 0: off, 1: on
volatile uint8_t pressCount = 0;

void initializeSystem(void);
void setupPWM(void);
void toggleFanState(void);
```

Η initializeSystem() είναι υπεύθυνη για την αρχικοποίηση των pins και των χρονοδιακοπτών. Συγκεκριμένα, μέσω της ρύθμισης του TCA0 σε split mode, επιτυγχάνεται η ανεξάρτητη διαμόρφωση των δύο PWM σημάτων με διαφορετικές περιόδους και κύκλους εργασίας. Η συνάρτηση toggleFanState() καλείται κάθε φορά που ανιχνεύεται falling edge, ενεργοποιώντας ή απενεργοποιώντας τον ανεμιστήρα ανάλογα με το πλήθος των πατημάτων.



```
void initializeSystem(void) {
    PORTD.DIR |= PIN0_bm | PIN1_bm;
    PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_FALLING_gc;
    sei(); // Enable global interrupts
}
```

Η υλοποίηση ξεκίνησε με την επιλογή των κατάλληλων pins (σύμφωνα με τις οδηγίες και τις παλαιότερες ασκήσεις) του μικροελεγκτή για την έξοδο των PWM σημάτων. Στη συνέχεια, έγινε η αρχικοποίηση των αντίστοιχων χρονοδιακοπτών (timers) του μικροελεγκτή, ρυθμίζοντας τα σημεία αναφοράς για την περίοδο και το duty cycle κάθε σήματος. Το duty

cycle αναφέρεται στο ποσοστό του χρόνου κατά το οποίο το σήμα βρίσκεται στην υψηλή κατάσταση εντός μίας πλήρους περιόδου.

Για την κίνηση των λεπίδων χρησιμοποιήθηκε ένας PWM χρονοδιακόπτης με περίοδο 1ms και duty cycle 50%, ενώ για την κίνηση της βάσης επιλέχθηκε ένα δεύτερο PWM σήμα με περίοδο 2ms και duty cycle 60%. Αυτό επιτυγχάνει μια βασική εναλλαγή της ταχύτητας και της κατεύθυνσης, παρέχοντας τη δυνατότητα περιστροφής των λεπίδων και της βάσης με διαφορετικές ταχύτητες.

```
void setupPWM(void) {
    TCA0.SPLIT.CTRLD = TCA_SPLIT_SPLITM_bm; // Enable split mode

    TCA0.SPLIT.LPER = BLADE_PWM_PERIOD; // period blade
    TCA0.SPLIT.HPER = BASE_PWM_PERIOD; // period base
    TCA0.SPLIT.LCMP0 = BLADE_PWM_PERIOD / 2; // 50% blade
    TCA0.SPLIT.HCMP0 = BASE_PWM_PERIOD * 0.6; // 60% base

    TCA0.SPLIT.CTRLB = TCA_SPLIT_LCMP0EN_bm | TCA_SPLIT_HCMP0EN_bm; // Enable PWM

    // Enable hunf lunf
    TCA0.SPLIT.INTCTRL = TCA_SPLIT_LUNF_bm | TCA_SPLIT_HUNF_bm;

    TCA0.SPLIT.CTRLA = TCA_SPLIT_ENABLE_bm | TCA_SPLIT_CLKSEL_DIV1024_gc; // Start
}
```

Κατά την ενεργοποίηση του διακόπτη (switch), οι ρυθμίσεις των PWM χρονοδιακοπών πρέπει να ενεργοποιηθούν, ξεκινώντας έτσι την κίνηση του ανεμιστήρα. Η λογική αυτή υλοποιείται μέσω της συνάρτησης ελέγχου που καλείται από τη διακοπή του διακόπτη (pin5 του portf), ενώ ο κώδικας φροντίζει να επαναφέρει την κατάσταση των LEDs σε κατάσταση αναμονής όταν ο ανεμιστήρας δεν είναι ενεργοποιημένος.

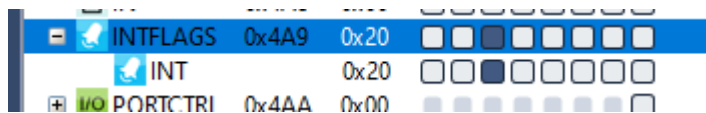


Figure 1: πριν την εκτέλεση

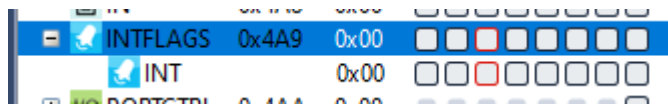


Figure 2: μετά την εκτέλεση

ISR(TCA0_LUNF_vect) και ISR(TCA0_HUNF_vect)

```
ISR(TCA0_LUNF_vect) {  
    //if (lunf_flag=1){  
        PORTD.OUTTGL = PIN0_bm;  
        TCA0.SPLIT.INTFLAGS = TCA_SPLIT_LUNF_bm;  
        lunf_flag=0;  
    }  
    //}  
}  
  
ISR(TCA0_HUNF_vect) {  
    //if (lunf_flag=0){  
        PORTD.OUTTGL = PIN1_bm;  
        TCA0.SPLIT.INTFLAGS = TCA_SPLIT_HUNF_bm;  
        lunf_flag=1;  
    }  
    //}  
}
```

Οι δύο αυτές διακοπές αναφέρονται στα LUNF και HUNF του χρονοδιακόπτη TCA0, αντίστοιχα. Στην πράξη, αυτό σημαίνει ότι κάθε φορά που το τάμερ φτάνει σε ένα συγκεκριμένο σημείο, εκτελείται η αντίστοιχη ISR. Στην περίπτωσή μας, χρησιμοποιούμε αυτές τις διακοπές για να ελέγχουμε την κατάσταση των LEDs που συμβολίζουν την κίνηση των λεπίδων και της βάσης του ανεμιστήρα.

The image displays two screenshots from the AVR Studio IDE. The left screenshot shows the C code for the interrupt service routines (ISR) for the TCA0 timer. The right screenshot shows the I/O View window, which displays the status of various I/O components.

Left Screenshot (Code):

```
ISR(TCA0_LUNF_vect) {  
    //if (lunf_flag=1){  
        PORTD.OUTTGL = PIN0_bm;  
        TCA0.SPLIT.INTFLAGS = TCA_SPLIT_LUNF_bm;  
        lunf_flag=0;  
    }  
    //}  
}  
  
ISR(TCA0_HUNF_vect) {  
    //if (lunf_flag=0){  
        PORTD.OUTTGL = PIN1_bm;  
        TCA0.SPLIT.INTFLAGS = TCA_SPLIT_HUNF_bm;  
        lunf_flag=1;  
    }  
    //}  
}
```

Right Screenshot (I/O View):

Name	Address	Value	Bits
CTRLFSET	0xA07	0x00	0000
EVCTRL	0xA09	0x00	0000
INTCTRL	0xA0A	0x03	0000
HUNF	0x01	0x01	0000
LCM...	0x00	0x00	0000
LCM...	0x00	0x00	0000
LCM...	0x00	0x00	0000
LUNF	0x01	0x01	0000
INTCTRL	0xA0A	0x03	0000
INTFLAGS	0xA0B	0x63	0000
INTFLAGS	0xA0B	0x63	0000

Η χρήση των ISR για τον έλεγχο των LEDs μας επιτρέπει να δημιουργήσουμε έναν πολύ ακριβή και συγχρονισμένο έλεγχο της κατάστασής τους, ανταποκρινόμενοι άμεσα στις αλλαγές των χρονοδιακοπών, χωρίς να επιβαρύνουμε τον κύριο βρόχο εκτέλεσης. Αυτό σημαίνει ότι ακόμα και ενώ ο μικροελεγκτής ενδεχομένως να εκτελεί άλλες λειτουργίες, ο έλεγχος των LEDs θα συνεχίσει να λειτουργεί ανεξάρτητα και αποτελεσματικά.

Name	Address	Value	Bits
DIR	0x460	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
DIRSET	0x461	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
DIRCLR	0x462	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
DIRTGL	0x463	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
OUT	0x464	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
OUTSET	0x465	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
OUTCLR	0x466	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
OUTTGL	0x467	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
IN	0x468	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
INTFLAGS	0x469	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
PORTCTRL	0x46A	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

-LED αρχικά:

-LED αφού πατήσω το 5 bit του portf και μπει στην lunf:

Name	Address	Value	Bits
DIR	0x460	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
DIRSET	0x461	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
DIRCLR	0x462	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
DIRTGL	0x463	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
OUT	0x464	0x01	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>
OUTSET	0x465	0x01	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>
OUTCLR	0x466	0x01	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>
OUTTGL	0x467	0x01	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>
IN	0x468	0x01	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>
INTFLAGS	0x469	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
PORTCTRL	0x46A	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
PIN0CTRL	0x470	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
PIN1CTRL	0x471	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
PIN2CTRL	0x472	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

-LED αφού βγεί μόνο του από την lunf και μπει στην hunf:

Name	Address	Value	Bits
DIR	0x460	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
DIRSET	0x461	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
DIRCLR	0x462	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
DIRTGL	0x463	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
OUT	0x464	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
OUTSET	0x465	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
OUTCLR	0x466	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
OUTTGL	0x467	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
IN	0x468	0x03	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input checked="" type="checkbox"/>
INTFLAGS	0x469	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
PORTCTRL	0x46A	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
PIN0CTRL	0x470	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
PIN1CTRL	0x471	0x00	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

-Κ.Ο.Κ

Επιλογή Περιόδου

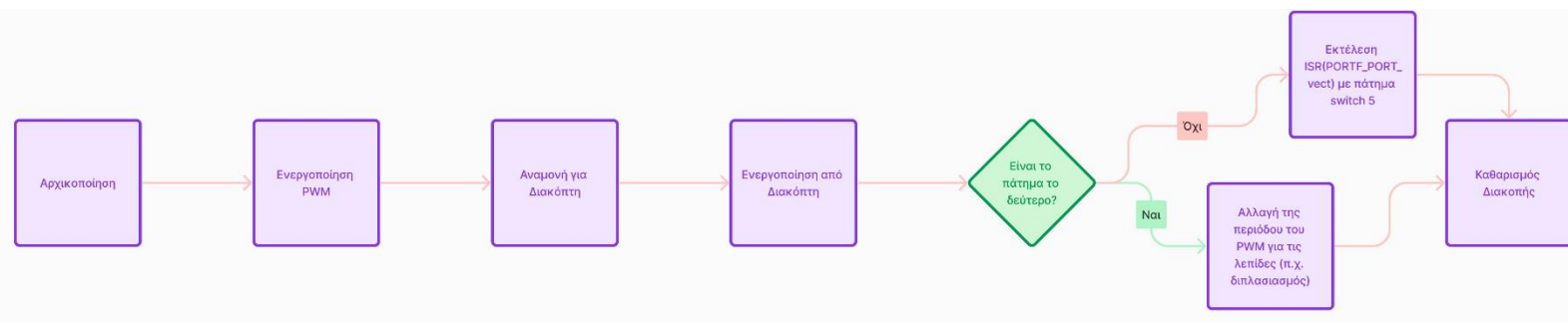
Η συχνότητα του PWM σήματος (f_{PWM_SS}) εξαρτάται από τη συχνότητα του ρολογιού περιφερειακού f_{CLK_PER} (), τον prescaler (N), και την τιμή του PER (που ορίζει την περίοδο του τάιμερ).

Στην περίπτωση που f_{CLK_PER} είναι 20MHz και ο prescaler N=1024, ο τύπος μας δίνει έναν τρόπο να υπολογίσουμε την τελική συχνότητα του PWM σήματος βάσει της επιθυμητής περιόδου του τάιμερ (PER). Ουσιαστικά, αυτό μας επιτρέπει να προσαρμόσουμε την ταχύτητα κίνησης του ανεμιστήρα ανάλογα με τις ανάγκες μας.

$$f_{PWM_SS} = \frac{f_{CLK_PER}}{N(PER + 1)}$$

Ερώτημα 2:

Παράθεση Διαγράμματος Ροής:



Σκοπός του κώδικα:

Προσθέτουμε μια νέα λειτουργία που αφορά τη διαχείριση της κυκλικής κίνησης των λεπίδων του ανεμιστήρα. Η ενεργοποίηση αυτής της λειτουργίας γίνεται μέσω της εκ νέου ενεργοποίησης του διακόπτη (pin 5).

Επεξήγηση βασικών σημείων του κώδικα:

Τι Προσθέσαμε στον Κώδικα:

Για την υλοποίηση της νέας λειτουργίας, προσθέτουμε λογική ελέγχου εντός της συνάρτησης toggleFanState(), η οποία ανταποκρίνεται στις διακοπές που προκύπτουν από το πάτημα του διακόπτη. Με το δεύτερο πάτημα του διακόπτη, η λειτουργία αυτή διπλασιάζει την περίοδο

του PWM σήματος που ρυθμίζει την κίνηση των λεπίδων, ενώ διατηρεί σταθερό το duty cycle.

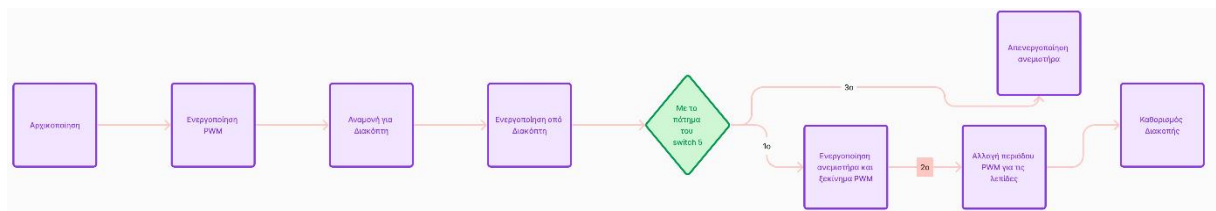
```
} else if (pressCount == 2) {  
    TCA0.SPLIT.LPER = BLADE_PWM_PERIOD * 2; // Double blade  
    TCA0.SPLIT.LCMP0 = (BLADE_PWM_PERIOD * 2) / 2; // Adjust duty
```

Η βασική ιδέα πίσω από αυτή την προσθήκη είναι να επιτρέψουμε την ρύθμιση της ταχύτητας των λεπίδων του ανεμιστήρα μέσω της εναλλαγής της περιόδου του σχετικού PWM σήματος. Με τον διπλασιασμό της περιόδου, ουσιαστικά μειώνουμε τη συχνότητα του σήματος, πράγμα που οδηγεί σε πιο αργή περιστροφή των λεπίδων. Αυτό παρέχει στον χρήστη μια επιπλέον επιλογή ελέγχου της ταχύτητας του ανεμιστήρα, αυξάνοντας την ευελιξία της συσκευής.

Η λογική ελέγχου που προστέθηκε στην toggleFanState(), λοιπόν, ανιχνεύει τον αριθμό των πατημάτων του διακόπτη και ανάλογα προσαρμόζει τις ρυθμίσεις του PWM. Με το πρώτο πάτημα ενεργοποιεί τον ανεμιστήρα, με το δεύτερο πάτημα διπλασιάζει την περίοδο του PWM για τις λεπίδες, και με το τρίτο πάτημα θα απενεργοποιούσε τον ανεμιστήρα (σύμφωνα με το επόμενο υποερώτημα).

Ερώτημα 3:

Παράθεση Διαγράμματος Ροής:



Σκοπός του κώδικα:

Εισάγουμε μια λειτουργία που επιτρέπει την απενεργοποίηση του ανεμιστήρα και τη δυνατότητα ενεργοποίησής του εκ νέου με το επόμενο πάτημα του διακόπτη (switch 5). Αυτό χτίζει πάνω στην υπάρχουσα λογική που έχουμε αναπτύξει στα προηγούμενα υποερωτήματα 1 και 2

Επεξήγηση βασικών σημείων του κώδικα:

Στον κώδικα, ενσωματώνουμε λογική ελέγχου για να ανιχνεύουμε την τρίτη ενεργοποίηση του διακόπτη και να απενεργοποιήσουμε αναλόγως τον ανεμιστήρα. Αυτό σημαίνει την

προσθήκη ενός ακόμη βήματος στην συνάρτηση toggleFanState(), που είναι υπεύθυνη για την αλλαγή των καταστάσεων του ανεμιστήρα.

```
    } else {  
        // Third press - turn off fan  
        fan_state = 0;  
        TCA0.SPLIT.CTRLA &= ~TCA_SPLIT_ENABLE_bm; // Disable PWM  
        PORTD.OUTCLR = PIN0_bm | PIN1_bm; // Turn off LEDs  
        pressCount = 0; // Reset press count to allow reactivation  
    }
```

Λογική της Υλοποίησης

Η λογική πίσω από αυτή την προσθήκη είναι η εξής:

1. Πρώτο Πάτημα του Διακόπτη: Ενεργοποιεί τον ανεμιστήρα και ξεκινά τα PWM σήματα που οδηγούν τα LEDs (Λειτουργία από το υποερώτημα 1).
2. Δεύτερο Πάτημα του Διακόπτη: Αλλάζει την περίοδο του PWM σήματος για τις λεπίδες, διπλασιάζοντας την περίοδο και διατηρώντας το duty cycle (Λειτουργία από το υποερώτημα 2).
3. Τρίτο Πάτημα του Διακόπτη: Απενεργοποιεί τον ανεμιστήρα. Αυτό επιτυγχάνεται σταματώντας τα PWM σήματα και σβήνοντας τα LEDs, προσφέροντας μια πλήρη απενεργοποίηση της συσκευής.

Αυτό το υλοποιούμε με την χρήση μιας μεταβλητής για την καταμέτρηση των πατημάτων του διακόπτη, καθώς και ελέγχων συνθήκης για τον καθορισμό της τρέχουσας κατάστασης και της ανάλογης αντίδρασης του συστήματος.

ΠΛΗΡΗΣ ΚΩΔΙΚΑΣ:

1.c:

```
#include <avr/io.h>  
#include <avr/interrupt.h>  
  
#define F_CPU 20000000  
#define PRESCALER 1024  
#define BLADE_PWM_PERIOD 254  
#define BASE_PWM_PERIOD 508  
  
volatile uint8_t fan_state = 0;  
volatile uint8_t pressCount = 0;
```

```

void initializeSystem(void);
void setupPWM(void);
void toggleFanState(void);

ISR(PORTF_PORT_vect) {
    if (PORTF.INTFLAGS & PIN5_bm) {
        toggleFanState();
        PORTF.INTFLAGS = PIN5_bm;
    }
}

ISR(TCA0_LUNF_vect) {
    PORTD.OUTTGL = PIN0_bm;
    TCA0.SPLIT.INTFLAGS = TCA_SPLIT_LUNF_bm;
}

ISR(TCA0_HUNF_vect) {
    PORTD.OUTTGL = PIN1_bm;
    TCA0.SPLIT.INTFLAGS = TCA_SPLIT_HUNF_bm;
}

void toggleFanState(void) {
    pressCount++;

    if (pressCount == 1) {
        fan_state = 1;
        setupPWM();
    } else if (pressCount == 2) {

    } else {

    }
}

void initializeSystem(void) {
    PORTD.DIR |= PIN0_bm | PIN1_bm;
    PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_FALLING_gc;
    sei();
}

void setupPWM(void) {
    TCA0.SPLIT.CTRLD = TCA_SPLIT_SPLITM_bm;

    TCA0.SPLIT.LPER = BLADE_PWM_PERIOD;
    TCA0.SPLIT.HPER = BASE_PWM_PERIOD;
    TCA0.SPLIT.LCMP0 = BLADE_PWM_PERIOD / 2;
    TCA0.SPLIT.HCMP0 = BASE_PWM_PERIOD * 0.6;

    TCA0.SPLIT.CTRLB = TCA_SPLIT_LCMP0EN_bm | TCA_SPLIT_HCMP0EN_bm;

    TCA0.SPLIT.INTCTRL = TCA_SPLIT_LUNF_bm | TCA_SPLIT_HUNF_bm;

    TCA0.SPLIT.CTRLA = TCA_SPLIT_ENABLE_bm | TCA_SPLIT_CLKSEL_DIV1024_gc;
}

int main(void) {
    initializeSystem();

    while (1) {

```

```
}  
}
```

2.c

```
#include <avr/io.h>  
#include <avr/interrupt.h>  
  
#define F_CPU 20000000  
#define PRESCALER 1024  
#define BLADE_PWM_PERIOD 254  
#define BASE_PWM_PERIOD 508  
  
volatile uint8_t fan_state = 0;  
volatile uint8_t pressCount = 0;  
  
void initializeSystem(void);  
void setupPWM(void);  
void toggleFanState(void);  
  
ISR(PORTF_PORT_vect) {  
    if (PORTF.INTFLAGS & PIN5_bm) {  
        toggleFanState();  
        PORTF.INTFLAGS = PIN5_bm;  
    }  
}  
  
ISR(TCA0_LUNF_vect) {  
    PORTD.OUTTGL = PIN0_bm;  
    TCA0.SPLIT.INTFLAGS = TCA_SPLIT_LUNF_bm;  
}  
  
ISR(TCA0_HUNF_vect) {  
    PORTD.OUTTGL = PIN1_bm;  
    TCA0.SPLIT.INTFLAGS = TCA_SPLIT_HUNF_bm;  
}  
  
void toggleFanState(void) {  
    pressCount++;  
  
    if (pressCount == 1) {  
        fan_state = 1;  
        setupPWM();  
    } else if (pressCount == 2) {  
        TCA0.SPLIT.LPER = BLADE_PWM_PERIOD * 2;  
        TCA0.SPLIT.LCMP0 = (BLADE_PWM_PERIOD * 2) / 2;  
    } else {  
  
    }  
}  
  
void initializeSystem(void) {  
    PORTD.DIR |= PIN0_bm | PIN1_bm;  
    PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_FALLING_gc;  
    sei();  
}  
  
void setupPWM(void) {
```

```

TCA0.SPLIT.CTRLD = TCA_SPLIT_SPLITM_bm;

TCA0.SPLIT.LPER = BLADE_PWM_PERIOD;
TCA0.SPLIT.HPER = BASE_PWM_PERIOD;
TCA0.SPLIT.LCMP0 = BLADE_PWM_PERIOD / 2;
TCA0.SPLIT.HCMP0 = BASE_PWM_PERIOD * 0.6;

TCA0.SPLIT.CTRLB = TCA_SPLIT_LCMP0EN_bm | TCA_SPLIT_HCMP0EN_bm;

TCA0.SPLIT.INTCTRL = TCA_SPLIT_LUNF_bm | TCA_SPLIT_HUNF_bm;

TCA0.SPLIT.CTRLA = TCA_SPLIT_ENABLE_bm | TCA_SPLIT_CLKSEL_DIV1024_gc;
}

int main(void) {
    initializeSystem();

    while (1) {
    }
}

```

3.c:

```

#include <avr/io.h>
#include <avr/interrupt.h>

#define F_CPU 20000000
#define PRESCALER 1024
#define BLADE_PWM_PERIOD 254
#define BASE_PWM_PERIOD 508

volatile uint8_t fan_state = 0;
volatile uint8_t pressCount = 0;
volatile uint8_t lunf_flag = 1;

void initializeSystem(void);
void setupPWM(void);
void toggleFanState(void);

ISR(PORTF_PORT_vect) {
    if (PORTF.INTFLAGS & PIN5_bm) {
        toggleFanState();
        PORTF.INTFLAGS = PIN5_bm;
    }
}

ISR(TCA0_LUNF_vect) {
    PORTD.OUTTGL = PIN0_bm;
    TCA0.SPLIT.INTFLAGS = TCA_SPLIT_LUNF_bm;
    lunf_flag = 0;
}

ISR(TCA0_HUNF_vect) {
    PORTD.OUTTGL = PIN1_bm;
    TCA0.SPLIT.INTFLAGS = TCA_SPLIT_HUNF_bm;
    lunf_flag = 1;
}

```

```

}

void toggleFanState(void) {
    pressCount++;

    if (pressCount == 1) {
        fan_state = 1;
        setupPWM();
    } else if (pressCount == 2) {
        TCA0.SPLIT.LPER = BLADE_PWM_PERIOD * 2;
        TCA0.SPLIT.LCMP0 = (BLADE_PWM_PERIOD * 2) / 2;
    } else {
        fan_state = 0;
        TCA0.SPLIT.CTRLA &= ~TCA_SPLIT_ENABLE_bm;
        PORTD.OUTCLR = PIN0_bm | PIN1_bm;
        pressCount = 0;
    }
}

void initializeSystem(void) {
    PORTD.DIR |= PIN0_bm | PIN1_bm;
    PORTF.PIN5CTRL |= PORT_PULLUPEN_bm | PORT_ISC_FALLING_gc;
    sei();
}

void setupPWM(void) {
    TCA0.SPLIT.CTRLD = TCA_SPLIT_SPLITM_bm;

    TCA0.SPLIT.LPER = BLADE_PWM_PERIOD;
    TCA0.SPLIT.HPER = BASE_PWM_PERIOD;
    TCA0.SPLIT.LCMP0 = BLADE_PWM_PERIOD / 2;
    TCA0.SPLIT.HCMP0 = BASE_PWM_PERIOD * 0.6;

    TCA0.SPLIT.CTRLB = TCA_SPLIT_LCMP0EN_bm | TCA_SPLIT_HCMP0EN_bm;

    TCA0.SPLIT.INTCTRL = TCA_SPLIT_LUNF_bm | TCA_SPLIT_HUNF_bm;

    TCA0.SPLIT.CTRLA = TCA_SPLIT_ENABLE_bm | TCA_SPLIT_CLKSEL_DIV1024_gc;
}

int main(void) {
    initializeSystem();

    while (1) {
    }
}

```

ΤΕΛΟΣ ΑΝΑΦΟΡΑΣ