Άσκηση τέταρτη του Εργαστηρίου

Στη τέταρτη άσκηση εξετάζεται ο τρόπος λειτουργίας της διακοπής από την υπερχείλιση του timer0 στο PIC18F4550 καθώς και η χρήση ειδικών λειτουργιών (functions) που υπάρχουν στο CCS C compiler για τη διαχείριση αυτού του τρόπου διακοπής.

Οι μετρητές ή χρονιστές γενικά.

Στο PIC18F4550 υπάρχουν 4 συνολικά χρονιστές ή μετρητές (timers).

Timer0 Λειτουργεί σε 8 ή 16 bits ως μετρητής χρόνου (Timer) ή ως μετρητής παλμών (Counter).

Timer1 Λειτουργεί σε 16bits ως μετρητής χρόνου (Timer) ή ως μετρητής παλμών (Counter).

Timer2 Λειτουργεί σε 8 bits ως μετρητής χρόνου και σε συνδυασμό με το ενσωματωμένο συγκριτή δημιουργεί παλμούς διαμορφωμένους σε εύρος (PWM).

Timer3 Λειτουργεί σε 8 ή 16 bits ως μετρητής χρόνου (Timer) ή ως μετρητής παλμών (Counter).

Η μονάδα Timer0 ενσωματώνει τα ακόλουθα χαρακτηριστικά:

•Λειτουργία του χρονιστή σε 8 ή 16 bits επιλεγόμενο από το πρόγραμμα και λειτουργία του χρονιστή ως μετρητή χρόνου (timer) ή ως μετρητή παλμών (counter)
•Δυνατότητα ανάγνωσης και εγγραφής των καταχωρητών του χρονιστή.
•Προγραμματιζόμενο διαιρέτη συχνότητας (prescaler) ο οποίος μπορεί να λειτουργεί σε συνδυασμό με τον χρονιστή ή όχι.

•Επιλογή των παλμών ρολογιού (clock) από πηγή (εσωτερική ή εξωτερική)
•Επιλογή της ακμής σκανδαλισμού για το εξωτερικό ρολόι

Η μέθοδος διακοπής από την υπερχείλιση του Timer 0 .

Συναρτήσεις αρχικοποίησης

SETUP_TIMER_0()

Για την αρχικοποίηση της λειτουργίας του timer0 στη παραπάνω συνάρτηση μπορούν να δοθούν οι παρακάτω παράμετροι που καθορίζουν το τρόπο λειτουργίας του

T0_OFF Μετρητής κλειστός

T0_8_BIT Θέτει τον μετρητή σε 8 bits λειτουργία μετρά από 0 έως το 255. Όταν δεν ορίζεται τότε υπονοείται λειτουργία του μετρητή σε 16 bits. Μέτρηση από 0 έως 65535.

T0_INTERNAL Ο μετρητής μετρά τους παλμούς που έρχονται από το εσωτερικό ρολόι και επειδή οι παλμοί έρχονται σε ίσα χρονικά διαστήματα, μετρά χρόνο.

T0_EXT_L_TO_H Ο μετρητής παίρνει παλμούς από εξωτερική πηγή μέσω του RA4 και στη περίπτωση αυτή η λειτουργία του είναι ως μετρητής εξωτερικών παλμών (counter) Η μέτρηση των παλμών αυτών γίνεται στην άνοδό τους (από Low στο High).

T0_EXT_H_TO_L Ο μετρητής παίρνει παλμούς από εξωτερική πηγή μέσω του RA4 και στη περίπτωση αυτή η λειτουργία του είναι ως μετρητής εξωτερικών παλμών (counter) Η μέτρηση των παλμών αυτών γίνεται στη κάθοδο τους (από High στο Low).

T0_DIV_1 /1 Με τις παραμέτρους αυτές καθορίζεται ο λόγος διαίρεσης του prescaler. Με λόγο διαίρεσης /1 ο προγραμματιζόμενος διαιρέτης (prescaler) δεν λειτουργεί με το timer0 και λειτουργεί με το WDT (Watch Dog Timer)

T0_DIV_2 /2

T0_DIV_4 /4

T0_DIV_8 /8

T0_DIV_16 /16

T0_DIV_32 /32

T0_DIV_64 /64

T0_DIV_128 /128

T0_DIV_256 /256

set_timer0(x) Με την συνάρτηση αυτή δίδεται αρχική τιμή στον timer0. Όπου 0<=x<=65535

get_timer0() Με τη συνάρτηση αυτή διαβάζεται η τιμή που έχει ο timer τη χρονική στιγμή της ανάγνωσης.

Παράδειγμα αρχικοποίησης του timer0

SETUP_TIMER_0(T0_INTERNAL | T0_DIV_4) Λειτουργία του timer0 έχοντας παλμούς από το εσωτερικό ρολόι με λόγο διαίρεσης του prescaler /4. Το σύμβολο | ενώνει τις παραμέτρους της ρουτίνας **SETUP_TIMER_0**

Παράδειγμα τοποθέτησης του timer0 σε αρχική τιμή

set_timer0(40000); Δίνει αρχική τιμή στον timer0 τη τιμή 40000, Στην υπερχείλισή του ο μετρητής παίρνει τη 0 αφού φθάσει τη μεγίστη τιμή που είναι η 65535.

Παράδειγμα ανάγνωσης του timer0

int16 temp;

temp = get_timer0(); επιστρέφει στη μεταβλητή temp τη τιμή του μετρητή 0.

Για την ενεργοποίηση και απενεργοποίηση της διακοπής υπάρχουν οι αντίστοιχες συναρτήσεις

enable_interrupts(INT_TIMER0); Η συνάρτηση αυτή ενεργοποιεί τη διακοπή από την υπερχείλιση του timer0.

disable_interrupts(INT_TIMER0); Η συνάρτηση αυτή απενεργοποιεί τη διακοπή από την υπερχείλιση του timer0.

Συνάρτηση διαχείρισης.

Η ρουτίνα διακοπής που αφορά τη διακοπή από την υπερχείλιση του timer0 δίδεται παρακάτω.

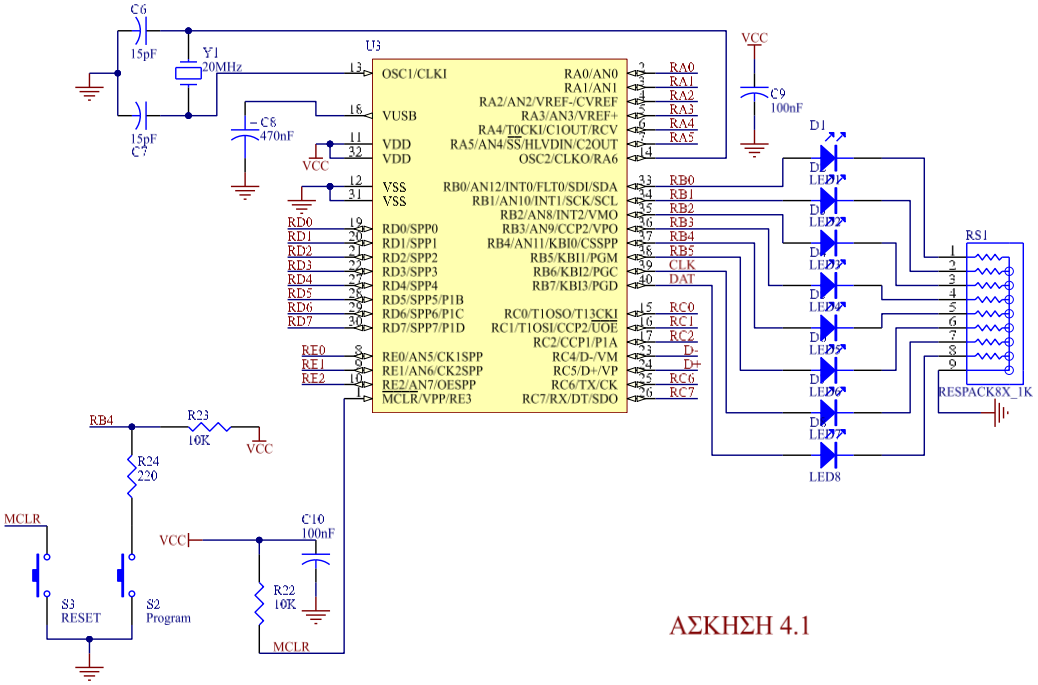
#INT_TIMER0

Ένα παράδειγμα χρήσης των παραπάνω δίδεται παρακάτω.

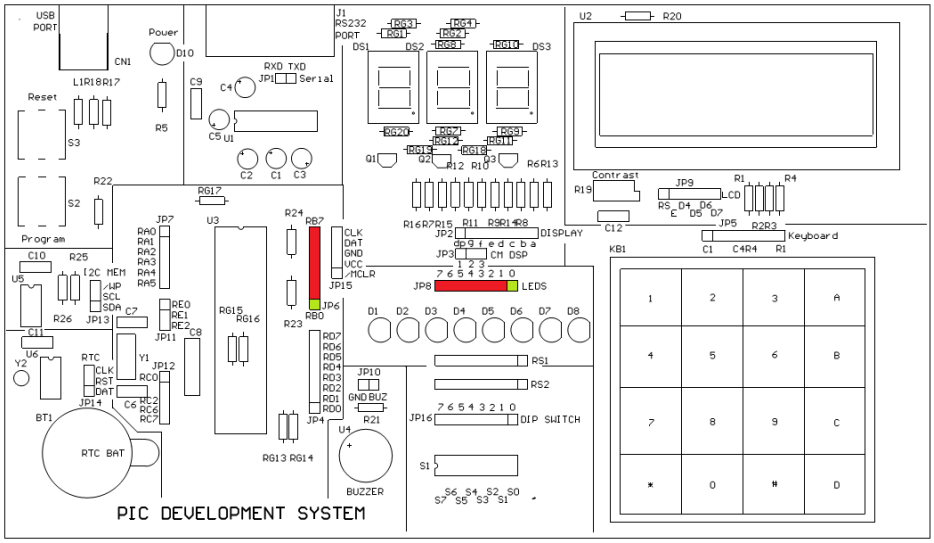
Να γραφεί πρόγραμμα για το μικροελεγκτή PIC 18F4550 με το οποίο να εκτελείται διακοπή από την υπερχείλιση του timer0.

Να αναβοσβήνουν τα Leds που είναι συνδεδεμένα στα bits RB0 και RB7 με ρυθμό 1Hz και τα Leds που είναι συνδεδεμένα στα bits RB4 και RB3 με ρυθμό 5 Hz.

Το σχηματικό του κυκλώματος



Οι συνδέσεις στην πλακέτα του εργαστηρίου



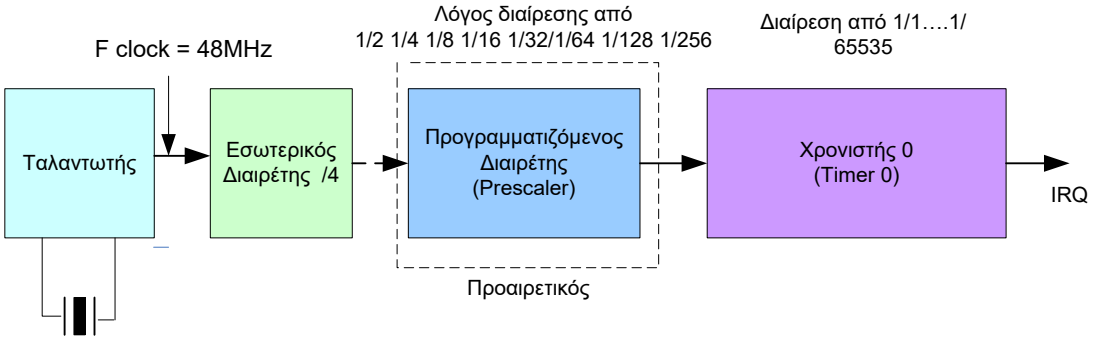
Το πρόγραμμα της άσκησης

Λύση

Στην άσκηση αυτή θα πρέπει να γίνουν τα εξής:

- Η πόρτα Β να ορισθεί σαν έξοδος.
- Να ενεργοποιηθεί ο γενικός διακόπτης διακοπών
- Να ενεργοποιηθεί εσωτερική διακοπή που προκαλείται από την υπερχείλιση του timer0.
- Να γραφεί το κύριο πρόγραμμα (που δεν θα κάνει τίποτα και απλά θα περιμένει να συμβεί η διακοπή)
- Να γραφεί η ρουτίνα εξυπηρέτησης της διακοπής όπου ο χρόνος της διακοπής θα είναι όσος ο χρόνος της ημιπεριόδου του σήματος με τη μεγαλύτερη συχνότητα και θα αλλάζει τη κατάσταση των Leds που είναι συνδεδεμένα στα bits RB3 και RB4 και κάθε 5 φορές που θα αλλάζει το σήμα με τη μεγαλύτερη συχνότητα θα αλλάζει η κατάσταση των bits RB0 και RB7, λόγω του λόγου των δύο συχνοτήτων 5/1.

Για τη καλύτερη κατανόηση της λειτουργίας του timer0 ως μετρητή χρόνου σχεδιάζουμε το εσωτερικό διάγραμμα του μετρητή.



Στην αναπτυξιακή πλακέτα η συχνότητα του κρυστάλλου είναι 20MHz αλλά εσωτερικά στο PIC υπάρχει ένα κύκλωμα πολλαπλασιασμού συχνότητας (PLL) από το οποίο μετά από πολλαπλασιασμό και διαίρεση προκύπτει συχνότητα λειτουργίας 48MHz. (20MHz / 5 =4MHz μετά το PLL 96MHz / 2 = 48MHz). Επίσης θα πρέπει να τονιστεί ότι εσωτερικά στο PIC υπάρχει ένας διαιρέτης που μπαίνει πάντα σε σειρά με το ρολόι του συστήματος και δημιουργεί τους 4 κύκλους που χρειάζεται ο μικροελεγκτής για να εκτελέσει την τρέχουσα εντολή και να ετοιμάσει την επόμενη.

Ο κύκλος μηχανής = 4 * περίοδοι του ρολογιού. Η περίοδος του ρολογιού είναι 1/f

όπου f η συχνότητα λειτουργίας του ρολογιού (48MHz).

Επομένως στην περίπτωση της αναπτυξιακής πλακέτας 1MC (Ένας κύκλος μηχανής) = 4 * 1/(48*10^6) sec.= 83,333 nsec.

Στις ασκήσεις που ζητείται η παραγωγή σημάτων με διαφορετικές συχνότητες με την μέθοδο της διακοπής διακρίνονται δύο περιπτώσεις .

1. Οι ημιπερίοδοι των παραγόμενων σημάτων έχουν ακέραιο λόγο. Στη περίπτωση αυτή ρυθμίζουμε το χρόνο της διακοπής να γίνεται κάθε ημιπερίοδο του σήματος με την μεγαλύτερη συχνότητα και το σήμα με την μικρότερη συχνότητα παράγεται από τον πολλαπλασιασμό του μικρότερου χρόνου ημιπεριόδου με τον ακέραιο (λόγος διαίρεσης) που προκύπτει από την διαίρεση των δύο συχνοτήτων.

2. Οι ημιπερίοδοι των παραγόμενων σημάτων δεν έχουν ακέραιο λόγο. Στη περίπτωση αυτή λαμβάνουμε αυθαίρετα ως χρόνο διακοπής ένα χρόνο μικρότερο της μικρότερης ημιπεριόδου που όμως διαιρεί τους χρόνους των ημιπεριόδων των δύο σημάτων ακριβώς.

Ο χρόνος που θα συμβαίνει το interrupt T_{INT} = MC (Machine Cycle) * (Διαίρεση του prescaler) * (65536 - (αρχική τιμή του μετρητή))

Στην άσκηση αυτή οι περίοδοι και ημιπερίοδοι των παραγομένων σημάτων είναι:

$T_1 = {}^1/_{f_1} = {}^1/_{1Hz}$ = 1sec => ${}^{T_1}/_2$ = 0.5sec = 500msec και

$T_2 = {}^1/_{f_2} = {}^1/_{5Hz}$= 0.2sec => ${}^{T_2}/_2$ = 0.1sec = 100msec. Παρατηρούμε ότι ο λόγος των δύο χρόνων είναι ακέραιος $\left.{}^{T_1}/_2\right/_{{}^{T_2}/_2} = {}^{500msec}/_{100msec}$ = 5. Στη περίπτωση αυτή λαμβάνουμε ως χρόνο διακοπής το μικρότερο χρόνο δηλαδή τα 100msec. Αν αυτή τη τιμή τη τοποθετήσουμε στη παραπάνω σχέση που δίνει τον χρόνο της διακοπής σε σχέση με τη διαίρεση του prescaler και της αρχικής τιμής του μετρητή έχουμε

100.000.000 nsec = 83,333 nsec * 128 * (65536 – αρχική τιμή του μετρητή) =>

αρχική τιμή του μετρητή = 56161. Η τιμή του διαιρέτη λαμβάνεται αυθαίρετα 128. Θα μπορούσε να ληφθεί και κάποια άλλη τιμή. Αν ληφθεί η μεγαλύτερη τιμή (256) τότε η αρχική τιμή θα προκύψει δεκαδική οπότε θα έχουμε σφάλμα στρογγυλοποίησης. Τονίζεται ότι η τιμές της διαίρεσης του προγραμματιζόμενου διαιρέτη είναι συγκριμένες όπως φαίνονται στις παραμέτρους παραπάνω.

Έχοντας όλες τις παραπάνω πληροφορίες μπορεί να γραφεί το πρόγραμμα χρησιμοποιώντας τις έτοιμες συναρτήσεις του C Compiler.

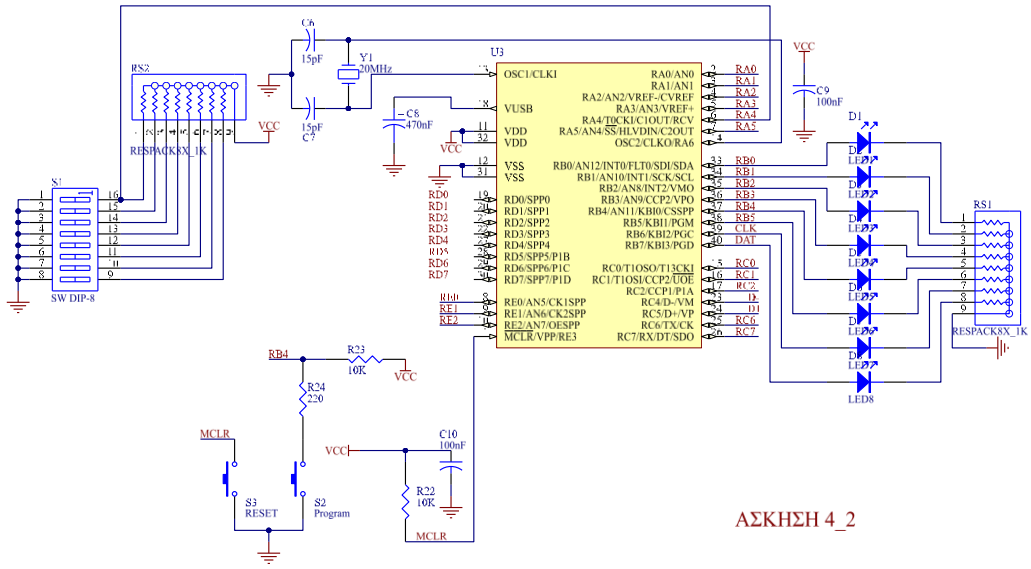
```
#include <main.h>
```

```
#use standard_io ( A )   // Standard είσοδοι και έξοδοι   // Standard είσοδοι και έξοδοι
#use standard_io ( B)
#use standard_io ( C )
#byte PORTA        =0xF80  //ορισμός των θυρών με την θέση τους στην
                           //μνήμη
#byte PORTB        =0xF81
#byte PORTC        =0xF82
#byte PORTD        =0xF83
#byte PORTE        =0xF84
#define Toggle_Led0 PORTB^=0x01; // αντιστροφή του bit RB0
#define Toggle_Led7 PORTB^=0x80; // αντιστροφή του bit RB7
#define Toggle_Led3 PORTB^=0x08; // αντιστροφή του bit RB3
#define Toggle_Led4 PORTB^=0x10; // αντιστροφή του bit RB4
int8  counter;

// Δήλωση συναρτήσεων

void timer0_int(void);
void init (void);
                        // Κύρια ρουτίνα
void main()
{
     init();

    while (1){                //  Περιμένει το Interrupt
               }
}

// Ορισμός συναρτήσεων

#INT_TIMER0 HIGH   // Διακοπή με μεγάλη προτεραιότητα
void timer0_int(void){
                          set_timer0(56161); // αρχική τιμή του μετρητή
                          Toggle_Led3;          // Αλλαγή στην κατάσταση του
                                        // Led3
                          Toggle_Led4;          // Αλλαγή στην κατάσταση του
                                        // Led4
                          counter--;                // Μείωση του counter κατά 1
                          if (counter == 0) {   // Αν ο counter  γίνει μηδέν
                                              counter = 5; // Επανατοποθέτηση στην
                                                        //  αρχική τιμή
                                              Toggle_Led7; //Αλλαγή στην κατάσταση
                                                         //του Led7
                                              Togge_Led0;    // Αλλαγή στην κατάσταση
                                                             //του Led0
                                               }
                          }

void init (void){
                     set_tris_b(0x00);     // Καθορισμός της πόρτας Β ως έξοδος
```

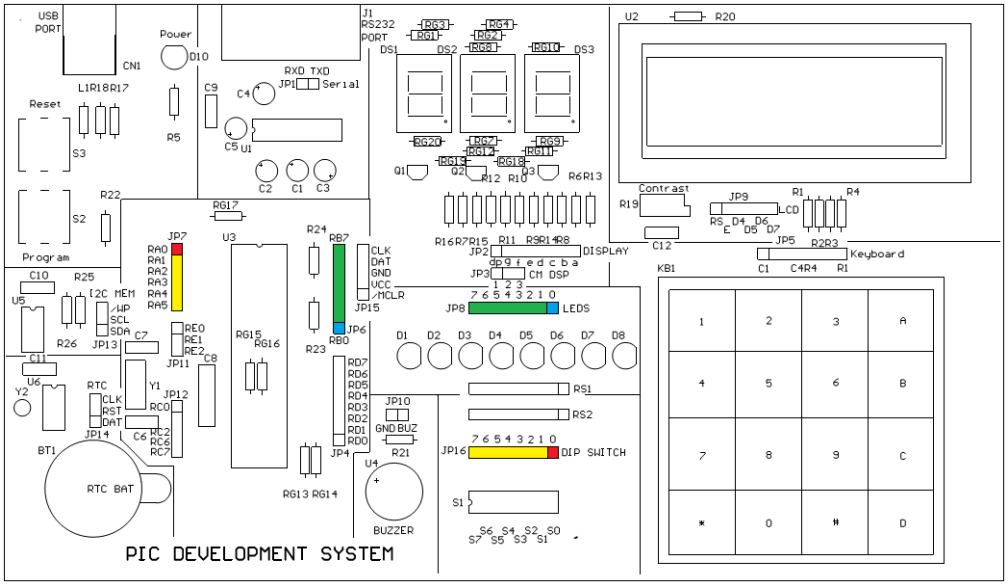
```
                counter=5;     // Αρχική τιμή του counter
                SETUP_TIMER_0(T0_INTERNAL | T0_DIV_128 );
                 set_timer0(56161);   //  Αρχική τιμή του  hardware μετρητή
                enable_interrupts(INT_TIMER0); // Ενεργοποίηση της
                                               // διακοπής του timer0
                enable_interrupts(GLOBAL);       // Ενεργοποίηση του γενικού
                                                 // διακόπτη των διακοπών

        }
```

Ένα παράδειγμα της λειτουργίας του timer0 ως μετρητή παλμών δίδεται παρακάτω. Να γραφεί πρόγραμμα σε γλώσσα προγραμματισμού C για τον μικροελεγκτή PIC 18F4550 που να μετρά τον αριθμό των παλμών που δίδονται από το bit RA4 με αλλαγή στο μέτωπο από λογικό1 σε λογικό 0 (κάθοδο του παλμού) και να δείχνει τον αριθμό αυτό στα Leds που θα είναι συνδεδεμένα στη πόρτα B.

Το σχηματικό του κυκλώματος



Οι συνδέσεις στη πλακέτα του εργαστηρίου



Το πρόγραμμα της άσκησης

Λύση

Στην άσκηση αυτή ο μετρητής λειτουργεί ως μετρητής παλμών. Οι παλμοί έρχονται στο μετρητή μέσω του pin RA4 και σε αυτή τη περίπτωση δεν θα πρέπει να υπάρχει ενδιάμεσα ο prescaler.

```
#include <main.h>

#use standard_io ( A )   // Standard είσοδοι και έξοδοι   // Standard είσοδοι και έξοδοι
#use standard_io ( B)
#use standard_io ( C )
#byte PORTA        =0xF80  //ορισμός των θυρών με την θέση τους στην
                           //μνήμη
#byte PORTB        =0xF81
#byte PORTC        =0xF82
#byte PORTD        =0xF83
#byte PORTE        =0xF84
int8 counter;

// Δήλωση συναρτήσεων

void init (void);

void main()
{
     init();

    while (1){
```

```
                counter = get_timer0(); // ανάγνωση του μετρητή
                delay_ms(100);
                PORTB=counter;  // Απεικόνιση του μετρητή στα Leds
                if (counter >255) {
                                counter = 0; // Αν ο μετρητής γίνει
                                        // μεγαλύτερος από το 255 τότε
                                        // κάνε τον μετρητή ξανά 0
                                }

        }
}

// Ορισμός συναρτήσεων

void init (void){
                set_tris_a(0xff);       // Καθορισμός της πόρτας Α ως είσοδος
                SETUP_ADC(NO_ANALOGS); // Disable τις αναλογικές εισ.
                setup_comparator(NC_NC_NC_NC);//Disable τους συγκριτές
                                                //της πόρτας Α
                set_tris_b(0x00);     // Καθορισμός της πόρτας Β ως έξοδος
                counter=0;              // Αρχική τιμή του counter
                PORTB=0;              // Αρχική τιμή εξόδου
                SETUP_TIMER_0(T0_EXT_H_TO_L | T0_DIV_1); // Θέσε τον
                                                //hardware μετρητή να
                                                //παίρνει παλμούς από το
                                                //RA4 και να μετρά τους
                                                //παλμούς στην αλλαγή
                                                //από High σε Low και ο
                                                //λόγος διαίρεσης θα είναι
                                                //1
                 set_timer(0);          // Μηδενισμός του counter
        }
```

Ασκήσεις που θα πρέπει να ετοιμαστούν για το τέταρτο εργαστήριο

1. Να γραφεί πρόγραμμα σε γλώσσα προγραμματισμού C για το μικροελεγκτή PIC 18F4550 με το οποίο το σύστημα θα λειτουργεί ως σήμανση τεχνικού έργου.
Το σύστημα θα λειτουργεί ως εξής: Όταν ένα αυτοκίνητο εισέρχεται στη ζώνη σήμανσης αλλάζει η κατάσταση της εισόδου RB1 από λογικό 1 σε λογικό 0. Αρχικά στο έργο το Led που αναβοσβήνει θα είναι το led που είναι συνδεδεμένο στο bit RB4, με ρυθμό 2Hz ενώ όταν αλλάξει η κατάσταση στο bit RB1 σταματά να αναβοσβήνει το Led RB4 και αρχίζει να αναβοσβήνει το led που είναι συνδεδεμένο με το bit RB7, με ρυθμό 4Hz. Να χρησιμοποιηθεί η μέθοδος της διακοπής από την υπερχείλιση του timer0 για το συγχρονισμό όλων των παραπάνω.

2, Να γραφεί πρόγραμμα σε γλώσσα προγραμματισμού C για το μικροελεγκτή PIC18F4550 που να λειτουργεί ως εξής:
a. Να αναβοσβήνει το led0 που θα είναι συνδεδεμένο στο RB7 με ρυθμό 10Hz.
b. Να αναβοσβήνει το led1 που θα είναι συνδεδεμένο στο RB6 με ρυθμό 4Hz.

c Να αναβοσβήνει το led2 που θα είναι συνδεδεμένο στο RB5 με ρυθμό 2Hz.
d. Να αναβοσβήνει το led3 που θα είναι συνδεδεμένο στο RB4 με ρυθμό 1Hz.
e. Να αναβοσβήνει το led4 που θα είναι συνδεδεμένο στο RB3 με ρυθμό 0.25Hz.
Να χρησιμοποιηθεί η μέθοδος της διακοπής από την υπερχείλιση του timer0 για το χρονισμό όλων των παραπάνω.

3. Να γραφεί πρόγραμμα σε γλώσσα προγραμματισμού C για το μικροελεγκτή PIC18F4550 που να λειτουργεί ως εξής:
Θα αναβοσβήνει τα 4 περισσότερο σημαντικά leds που συνδέονται στη πόρτα B (RB7,RB6,RB5,RB4) με τα υπόλοιπα σβηστά με ρυθμό 5 Hz όταν το bit RD0 είναι σε λογικό '1' , ενώ όταν είναι σε λογικό ΄0΄ να αναβοσβήνει και τα υπόλοιπα leds που συνδέονται στα 4 λιγότερο σημαντικά leds της πόρτα B με ρυθμό 2 Hz. Να χρησιμοποιηθεί η μέθοδος της διακοπής από την υπερχείλιση του timer0 για τον χρονισμό όλων των παραπάνω.

Ερωτήσεις που πρέπει να απαντηθούν. **(Προαιρετικά) στο Email** angmicro2@gmail.com

Α. Να γραφεί πρόγραμμα που να ελέγχει το bit RA1 και αν είναι σε λογικό 0 να θέτει το bit RB0 σε λογικό 0 ενώ αν το RA1 είναι σε λογικό 1 να θέτει το RB0 σε λογικό 1.

Β. Δίδεται το παρακάτω πρόγραμμα να διορθωθούν τα λάθη έτσι ώστε το interrupt να έρχεται κάθε 20 msec και να αλλάζει την κατάσταση στο bit RC0.

```
#include <main.h>
#byte PORTC        =0xF81
#define Toggle_Led0  PORTC^=0x02; // αντιστροφή του bit RC0
// Δήλωση συναρτήσεων
void timer0_int(void);
void init (void);
                          // Κύρια ρουτίνα
void main()
{
     init();

    while (1);        //  Περιμένει το Interrupt

}

// Ορισμός συναρτήσεων
#INT_TIMER0 HIGH   // Διακοπή με μεγάλη προτεραιότητα
void timer0_int(void){
                        set_timer0(20);   // αρχική τιμή του μετρητή
                        Toggle_Led0;      // Αλλαγή στην κατάσταση του
                                          // Led0
```

```
                    }
void init (void){
                    set_tris_c(0x00);     // Καθορισμός της πόρτας Β ως έξοδος
                    SETUP_TIMER_0(T0_INTERNAL | T0_DIV_256 );
                    set_timer0(20);   //  Αρχική τιμή του  hardware μετρητή
                    enable_interrupts(INT_TIMER0); // Ενεργοποίηση της
                                                   // διακοπής του timer0
                    enable_interrupts(GLOBAL);       // Ενεργοποίηση του γενικού
                                                   // διακόπτη των διακοπών
                }
```

C. Στο παρακάτω πρόγραμμα ποια είναι η τιμή του x μετά την εκτέλεση του προγράμματος.

```
int32 x;
char s[20] = “000000”;
for (int i=0; i<=4; i++){
s[i] = ‘0’ + i + 1;
}
 s >> x;
```

a. 12345
b. 0
c. 000000
d. 123450
e. 12340

D. Ένα όχημα έχει ρόδες με διάμετρο 5 πόδια. Στην ρόδα υπάρχει ένας αισθητήρας που δίνει ένα παλμό σε κάθε περιστροφή της ρόδας. Η έξοδος του αισθητήρα συνδέεται με τον PIC και δίνει τους παλμούς στην είσοδο RC0 του PIC. Κάθε δευτερόλεπτο το πρόγραμμα διαβάζει τον μετρητή παλμών (timer1) και τον μηδενίζει για να ξεκινήσει ένα νέος κύκλος. Ο μετρητής του PIC παίρνει τους παλμούς χωρίς την μεσολάβηση του διαιρέτη συχνότητας όπως στην 2η λυμένη άσκηση. Ποιος είναι ο τύπος που δίνει την ταχύτητα του οχήματος σε μίλια ανά ώρα (MPH).
Υπ. Ο timer1 είναι χρονιστής των 16 ή των 8 bits.

a. MPH = (5281/ get_timer1() * 5)*60*60
b. MPH = 60*60*5*get_timer1()/5281
c. MPH = (60*60*5*get_timer1()/4)/5281
d. MPH = (5281/get_timer1())*60*60
e. MPH = (5281/get_timer1()/4)*5*4000000