Άσκηση τρίτη του Εργαστηρίου

Στη τρίτη άσκηση δίδεται ο τρόπος λειτουργίας των διακοπών γενικά στο PIC και εξετάζεται αρχικά ο τρόπος λειτουργίας διακοπών από πόδια των θυρών που είναι διαθέσιμα στο PIC18F4550 καθώς και η χρήση ειδικών λειτουργιών (functions) που υπάρχουν στο CCS C compiler.

Διακοπές (interrupts)

Η διακοπή γενικά όπως το περιγράφει και η λέξη είναι λειτουργία του μικροελεγκτή κατά την οποία διακόπτεται η εκτέλεση του τρέχοντος προγράμματος λόγω κάποιου εξωτερικού ή εσωτερικού γεγονότος (π.χ. αλλαγή μετώπου στο πόδι RB0 ή υπερχείλιση του χρονιστή 0) και γίνεται αλλαγή της ροής του προγράμματος σε συγκεκριμένη διεύθυνση της μνήμης προγράμματος για την εκτέλεση ειδικού προγράμματος εξυπηρέτησης της διακοπής και επαναφορά της ροής του προγράμματος στην αμέσως επόμενη εντολή από όπου έγινε η διακοπή μετά την ολοκλήρωση του προγράμματος εξυπηρέτησης της διακοπής.

Γενικά για την επικοινωνία των περιφερειακών συσκευών με τη κεντρική μονάδα υπάρχουν δύο τρόποι.

α. Χρησιμοποιώντας τη μέθοδο των συνεχών ελέγχων (polling).

b. Χρησιμοποιώντας τη μέθοδο της διακοπής (interrupt).

Στη μέθοδο των συνεχών ελέγχων ο μικροελεγκτής ελέγχει συνεχώς κάποιο σήμα που αλλάζει η περιφερειακή συσκευή όταν αυτή θέλει να πάρει ή να δώσει δεδομένα στη κεντρική μονάδα. Αν υπάρχουν περισσότερες συσκευές ο μικροελεγκτής θα πρέπει μία προς μία να ελέγχει αν κάποια από τις συσκευές ζητά εξυπηρέτηση στη περίπτωση αυτή εξυπηρετεί τη συσκευή που ζήτησε και συνεχίζει τον έλεγχο της επομένης. Η σειρά με την οποία ελέγχονται ρυθμίζεται από το πρόγραμμα του χρήστη. Η μέθοδος αυτή απαιτεί από το μικροελεγκτή ένα μεγάλο μέρος από την υπολογιστική του ισχύ για τους συνεχείς ελέγχους και εξαρτάται από τον αριθμό των περιφερειακών συσκευών που είναι συνδεδεμένες με αυτή τη διαδικασία.

Στη μέθοδο της διακοπής ο μικροελεγκτής δεν χρειάζεται να ασχολείται με της περιφερειακές συσκευές και όταν κάποια ζητήσει επικοινωνία τότε συμβαίνει διακοπή και διακλάδωση στο πρόγραμμα εξυπηρέτησής της. Το πρόγραμμα αυτό εξυπηρέτησης ονομάζεται ρουτίνα εξυπηρέτησης της διακοπής (ISR interrupt service routine) ή ρουτίνα διαχείρισης της διακοπής (IH **I**nterrupt **H**andler).

Interrupt έναντι Polling.

Συγκρίνοντας τις δύο μεθόδους προκύπτει η υπεροχή της μεθόδου της διακοπής. Στη μέθοδο των συνεχών ελέγχων υπάρχει μεγάλη σπατάλη χρόνου και υπολογιστικής ισχύος του μικροελεγκτή, δεν μπορεί να αλλάξει η σειρά ελέγχου και εξυπηρέτησης των περιφερειακών συσκευών, δεν μπορεί να σταματήσει ο έλεγχος κάποιας συσκευής κατά τη διάρκεια εκτέλεσης του προγράμματος, υπάρχει κίνδυνος να χαθεί κάποια αίτηση στη περίπτωση που ο χρόνος του παλμού αίτησης είναι πολύ μικρός, δεν μπορούν να ιεραρχηθούν οι ρουτίνες εξυπηρέτησης, δεν μπορεί η μία ρουτίνα εξυπηρέτησης να διακόψει άλλη ρουτίνα. Όλα τα παραπάνω δεν συμβαίνουν στην περίπτωση της μεθόδου της διακοπής.

Η μέθοδος της διακοπής μπορεί διακριθεί σε επικαλυπτόμενη (maskable) και μη επικαλυπτόμενη (non maskable) . Στην επικαλυπτόμενη ο χρήστης έχει τη δυνατότητα μέσω του προγράμματος να καλύψει τη διακοπή, όποτε είναι αναγκαίο, χωρίς να χαθεί το συμβάν που προκάλεσε τη διακοπή και να τη διαχειριστεί όποτε μπορεί, ενώ στη μη επικαλυπτόμενη η διακοπή εκτελείτε όποτε συμβεί χωρίς τη δυνατότητα της επικάλυψης. Επίσης η διακοπή μπορεί να διακριθεί σε εσωτερική και σε εξωτερική. Εσωτερική είναι όταν συμβαίνει από περιφερειακές συσκευές που περιέχονται στον μικροελεγκτή (π.χ. υπερχείλιση κάποιου χρονιστή) και δεν απαιτείται εξωτερικό σήμα και εξωτερική διακοπή όταν η διακοπή προξενείτε από σήμα που προέρχεται από κάποια εξωτερική περιφερειακή συσκευή μέσω των θυρών εισόδου του μικροελεγκτή (π.χ αλλαγή στο πόδι RB0 του μικροελεγκτή).

Γενικά η χρήση της μεθόδου της διακοπής εφαρμόζεται οπωσδήποτε σε περιπτώσεις όπου

α. Σε εφαρμογές κρίσιμες στον χρόνο. Όπου δηλαδή υπάρχει κίνδυνος να χαθεί κάποιο συμβάν λόγω του πολύ μικρού χρόνου που είναι διαθέσιμο. π.χ. Στην περίπτωση που χάνεται η τροφοδοσία του κυκλώματος και πρέπει να σωθεί το περιεχόμενο κάποιων καταχωρητών έτσι ώστε όταν επανέλθει να συνεχίσει η εκτέλεση του κυρίως προγράμματος από εκεί που διακόπηκε.

β. Σε περιπτώσεις όπου κάποιες ρουτίνες πρέπει να εκτελούνται επανειλημμένα σε ίσα χρονικά διαστήματα.

γ. Όταν υπάρχουν πολλαπλά προγράμματα που εκτελούνται (multi tasking) και θα πρέπει ο χρόνος να μοιράζεται σε αυτά.

δ. Σε περιπτώσεις της άμεσης εξυπηρέτησης όποτε αυτή ζητηθεί χωρίς χάσιμο χρόνου.

ε. Γενικότερα όποτε κάποιο συμβάν είναι τελείως ασύγχρονο και δεν είναι γνωστό πότε μπορεί να συμβεί αποφεύγεται η χρήση των συνεχών ελέγχων (polling).

Στον μικροελεγκτή PIC18F4550 αλλά και γενικότερα στην PIC18FXXX σειρά υπάρχουν εσωτερικές διακοπές και εξωτερικές διακοπές αλλά όχι μη επικαλυπτόμενες εκτός από το σήμα του Reset που εκτελείτε όποτε συμβεί χωρίς να μπορεί να καλυφθεί.

Στον μικροελεγκτή PIC18F4550 υπάρχουν πολλοί τρόποι διακοπής που οφείλονται σε περιφερειακές συσκευές που περιέχονται στον μικροελεγκτή όπως οι χρονιστές ο μετατροπέας από αναλογικό σε ψηφιακό και άλλοι. Στην σειρά 18FXXX υπάρχουν δύο είδη διακοπής. Οι διακοπή με χαμηλή προτεραιότητα η οποία διακλαδίζει το πρόγραμμα στην διεύθυνση 0x018 της μνήμης του προγράμματος και η διακοπή με υψηλή προτεραιότητα που διακλαδίζει το πρόγραμμα στην διεύθυνση 0X008 της μνήμης του προγράμματος. Πρέπει να τονιστεί ότι μια διακοπή με μεγαλύτερη προτεραιότητα μπορεί να διακόψει μια με μικρότερη προτεραιότητα αλλά όχι το αντίθετο. Η διαχείριση των προτεραιοτήτων για τους διάφορους τρόπους διακοπής γίνεται από τους κατάλληλους καταχωρητές του μικροελεγκτή.

Η μέθοδος διακοπής από αλλαγή μετώπου στο pin 0 ή στο pin 1 ή στο pin 2 της πόρτας Β.

Στην μέθοδο αυτή η διακοπή μπορεί να γίνει μόνο από τα bit 0 bit 1 και bit2 της πόρτας Β και μπορεί να συμβεί, ανάλογα με τον προγραμματισμό του κατάλληλου καταχωρητή, στην άνοδο του παλμού ή στην κάθοδο του παλμού που δίδεται στο pin RB0 ή RB1 ή RB2 (πόρτα Β bit 0 ή 1 ή 2). Η διαδικασία που ακολουθείται για την ενεργοποίηση και την διαχείριση της διακοπής είναι τυποποιημένη και ο κατασκευαστής του C compiler δίνει μια σειρά από συναρτήσεις έτοιμες για τον έλεγχο και διαχείριση της διακοπής. Θα πρέπει να διευκρινιστεί ότι στην περίπτωση της αναπτυξιακής πλακέτας του εργαστηρίου οι διευθύνσεις διακοπής αλλάζουν, με διακοπή με μικρότερη προτεραιότητα λόγω της παρουσίας του boot-loader και από τη διεύθυνση 0x018 στη **0x818** και η διεύθυνση της διακοπής με τη μεγάλη προτεραιότητα από τη διεύθυνση 0x008 στη **0x808**. Οι διαθέσιμες έτοιμες συναρτήσεις του C compiler δίδονται παρακάτω. Διακρίνονται σε δύο κατηγορίες αυτές που αφορούν την αρχικοποίηση και αυτές που αφορούν τη διαχείριση.

Συναρτήσεις αρχικοποίησης

enable_interrupts(INT_EXT); Η συνάρτηση αυτή ενεργοποιεί τη διακοπή από αλλαγή μετώπου στην είσοδο από τη πόρτα Β από το bit 0 (RB0).

enable_interrupts(INT_EXT1); Η συνάρτηση αυτή ενεργοποιεί τη διακοπή από αλλαγή μετώπου στην είσοδο από τη πόρτα Β από το bit 1 (RB1).

enable_interrupts(INT_EXT2); Η συνάρτηση αυτή ενεργοποιεί τη διακοπή από αλλαγή μετώπου στην είσοδο από τη πόρτα Β από το bit 2 (RB2).

Αντίστοιχα για απενεργοποίηση των διακοπών υπάρχουν οι αντίστοιχες συναρτήσεις

disable_interrupts(INT_EXT); Η συνάρτηση αυτή απενεργοποιεί τη διακοπή από αλλαγή μετώπου στην είσοδο από τη πόρτα Β από το bit 0 (RB0).

disable_interrupts(INT_EXT1); Η συνάρτηση αυτή απενεργοποιεί τη διακοπή από αλλαγή μετώπου στην είσοδο από τη πόρτα Β από το bit 1 (RB1).

disable_interrupts(INT_EXT2); Η συνάρτηση αυτή απενεργοποιεί τη διακοπή από αλλαγή μετώπου στην είσοδο από τη πόρτα Β από το bit 2 (RB2).

Οι εντολές για το προγραμματισμό του μετώπου.

ext_int_edge(X, L_TO_H); Ρύθμιση του μετώπου για τη διακοπή από λογικό 0 σε λογικό 1 (άνοδος του παλμού) . Ανάλογα τη τιμή Χ μπορεί να είναι από το bit0 ή bit1 ή bit2 Αν Χ=0 από το RB0 , αν Χ=1 από το RB1 και αν Χ=2 από το RB2. Όμοια υπάρχει η εντολή

ext_int_edge(X, H_TO_L); Ρύθμιση του μετώπου για τη διακοπή από λογικό 1 σε λογικό 0 (κάθοδος του παλμού) . Ανάλογα τη τιμή Χ μπορεί να είναι από το bit0 ή bit1 ή bit2 Αν Χ=0 από το RB0 , αν Χ=1 από το RB1 και αν Χ=2 από το RB2.

port_x_pullups(TRUE); Στην περίπτωση που θέλουμε να αποφύγουμε εξωτερικές αντιστάσεις στην πόρτα Α ή Β με την εντολή αυτή μπορούν να ενεργοποιηθούν εσωτερικές (pull-ups) αντιστάσεις στη πόρτα Α ή Β (περίπου 10Κ) ανάλογα με το x αν x=a ή x=b. a=PORTA ,b=PORTB

enable_interrupts(GLOBAL); Η συνάρτηση αυτή ενεργοποιεί τον γενικό διακοπή των διακοπτών. Χωρίς την ενεργοποίηση αυτού δεν είναι δυνατή η διακοπή στο PIC από κανένα τρόπο.

Συναρτήσεις διαχείρισης.

Όπως αναφέρθηκε παραπάνω όταν συμβαίνει διακοπή το πρόγραμμα διακλαδίζεται στη διεύθυνση διακοπής και από εκεί στη ρουτίνα διακοπής που είναι διαφορετική για κάθε περίπτωση και προτεραιότητα. Στον C compiler της CCS οι συναρτήσεις των ρουτινών διακοπής ορίζονται με ξεχωριστό τρόπο για κάθε περίπτωση. Οι ρουτίνες διακοπής που αφορούν τις διακοπές από τα bits RB0, RB1 και RB2 δίδονται παρακάτω.

#INT_EXT Με τον τρόπο αυτό ορίζεται η ρουτίνα εξυπηρέτησης για τη διακοπή από το bit RB0.

#INT_EXT1 Με τον τρόπο αυτό ορίζεται η ρουτίνα εξυπηρέτησης για τη διακοπή από το bit RB1.

#INT_EXT2 Με τον τρόπο αυτό ορίζεται η ρουτίνα εξυπηρέτησης για τη διακοπή από το bit RB2.

Ένα παράδειγμα χρήσης των παραπάνω δίδεται παρακάτω.

Να γραφεί πρόγραμμα για τον μικροελεγκτή PIC 18F4550 με το οποίο να εκτελείται διακοπή από τον ακροδέκτη εξωτερικής διακοπής RB0.

Κάθε φορά που γίνεται μια μετάβαση από λογικό 1 σε λογικό 0 στον ακροδέκτη εξωτερικής διακοπής RB0 να ελαττώνεται κατά 1 η τιμή της θύρας D. Η αρχική τιμή της θύρας D είναι 0x30.

Το αποτέλεσμα της μείωσης εμφανίζεται στην πόρτα D στην οποία είναι συνδεδεμένα 8 LEDs.

Λύση

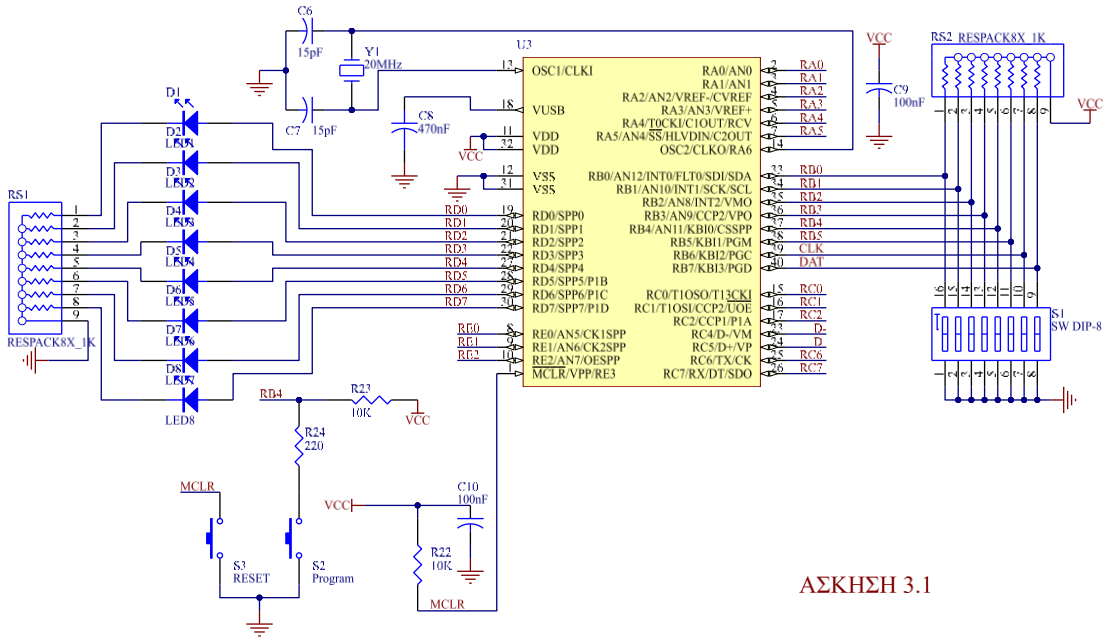
Στην άσκηση αυτή θα πρέπει να γίνουν τα εξής:

- Η πόρτα B να ορισθεί σαν είσοδος και η πόρτα D σαν έξοδος.
- Να ενεργοποιηθεί ο γενικός διακόπτης διακοπών
- Να ενεργοποιηθεί εξωτερική διακοπή που προκαλείται από αλλαγή κατάστασης του ακροδέκτη RB0.
- Να ορισθεί ότι η διακοπή θα προκαλείται κατά την μετάβαση του ακροδέκτη RB0 από λογικό 1 σε λογικό 0 (στο κατερχόμενο μέτωπο του παλμού).

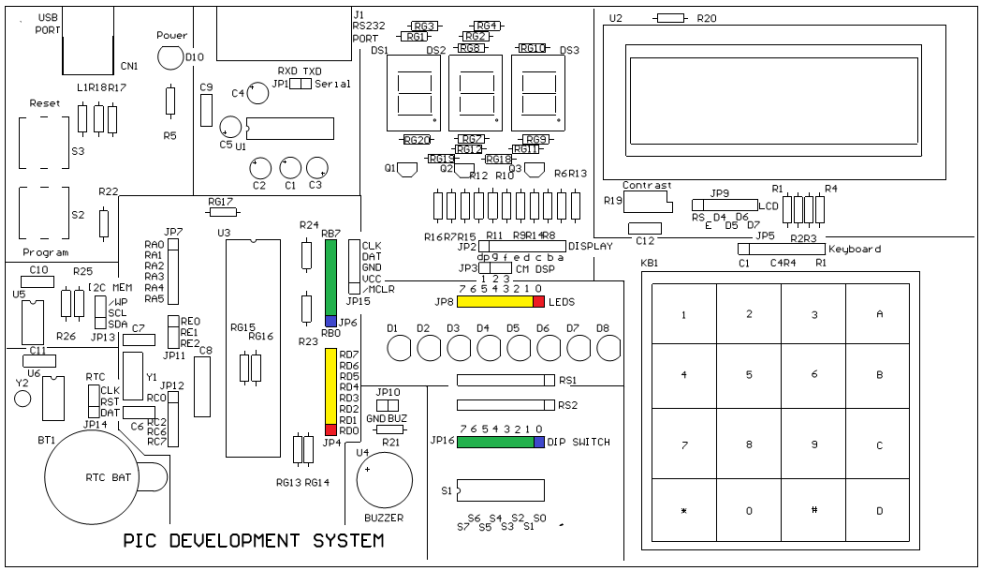
- Να γραφεί το κύριο πρόγραμμα (που δεν κάνει τίποτα και απλά περιμένει να συμβεί μια διακοπή)

- Να γραφεί η ρουτίνα εξυπηρέτησης της διακοπής που προκαλεί ελάττωση του περιεχομένου της πόρτας D κατά 1 και επανατοποθέτηση της πόρτας D στην τιμή 0x30 όταν το αποτέλεσμα της ελάττωσης γίνει 0.

Το σχηματικό της άσκησης



ΑΣΚΗΣΗ 3.1



Η σύνδεση των καλωδιοταινιών στη πλακέτα του εργαστηρίου.

Το πρόγραμμα της άσκησης

```
#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=0x30;
// Δήλωση συναρτήσεων

void ext_int0(void);
void init (void);

void main()
{
     init();

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

// Ορισμός συναρτήσεων
#INT_EXT   // Διακοπή με μεγάλη προτεραιότητα
void ext_int0(void){
                        counter--;
                        If (counter == 0){
                                          counter = 0x30;
                                          }
                        PORTD = counter;
                        delay_ms(300);  // Για να αποφύγουμε τις αναπηδήσεις
                                        //του  διακόπτη
                      }

void init (void){
                    set_tris_b(0xff);     // Καθορισμός της πόρτας B ως είσοδος
                    set_tris_d(0x00);      //  Καθορισμός της πόρτας D ως έξοδος
                    PORTD = counter;
                    port_b_pullups(TRUE);
                    ext_int_edge(0, H_TO_L);
```

```
        enable_interrupts(INT_EXT);
        enable_interrupts(GLOBAL);

    }
```

Η μέθοδος διακοπής από αλλαγή επιπέδου σε κάποιο από τα pins RB4,RB5,RB6,RB7.

Στη μέθοδο αυτή η διακοπή συμβαίνει όταν σε κάποιο από τα πόδια RB4,RB5,RB6,RB7 συμβεί αλλαγή επιπέδου δηλαδή σε κάποιο ή κάποια από τα πόδια αυτά αλλάξει η λογική κατάσταση (από λογικό 1 σε λογικό 0 ή από λογικό 0 σε λογικό 1). Η διαφορά σε σχέση με τη προηγούμενη μέθοδο είναι ότι στη μέθοδο αυτή δεν λαμβάνεται υπ' όψη η αλλαγή στο μέτωπο αλλά η αλλαγή στο επίπεδο σε όλα τα περισσότερο σημαντικά bits της πόρτας B. Επειδή η διακοπή συμβαίνει από οποιοδήποτε πόδι μέσα στη ρουτίνα εξυπηρέτησης θα πρέπει να γίνει έλεγχος των pins RB4,RB5,RB6,RB7 για να προσδιοριστεί η πηγή της διακοπής. Οι διαθέσιμες συναρτήσεις του C compiler δίδονται παρακάτω.

Συναρτήσεις αρχικοποίησης

enable_interrupts(INT_RB); Η συνάρτηση αυτή ενεργοποιεί την διακοπή από αλλαγή επιπέδου σε κάποιο ή κάποια πόδια από τη πόρτα εισόδου B RB4,RB5,RB6,RB7.

disable_interrupts(INT_RB); Η συνάρτηση αυτή απενεργοποιεί την διακοπή από αλλαγή επιπέδου σε κάποιο ή κάποια πόδια από τη πόρτα εισόδου B RB4,RB5,RB6,RB7.

Συναρτήσεις διαχείρισης.

 # INT_RB Με τον τρόπο αυτό ορίζεται η ρουτίνα εξυπηρέτησης για τη διακοπή από τα bits RB4,RB5,RB6,RB7.

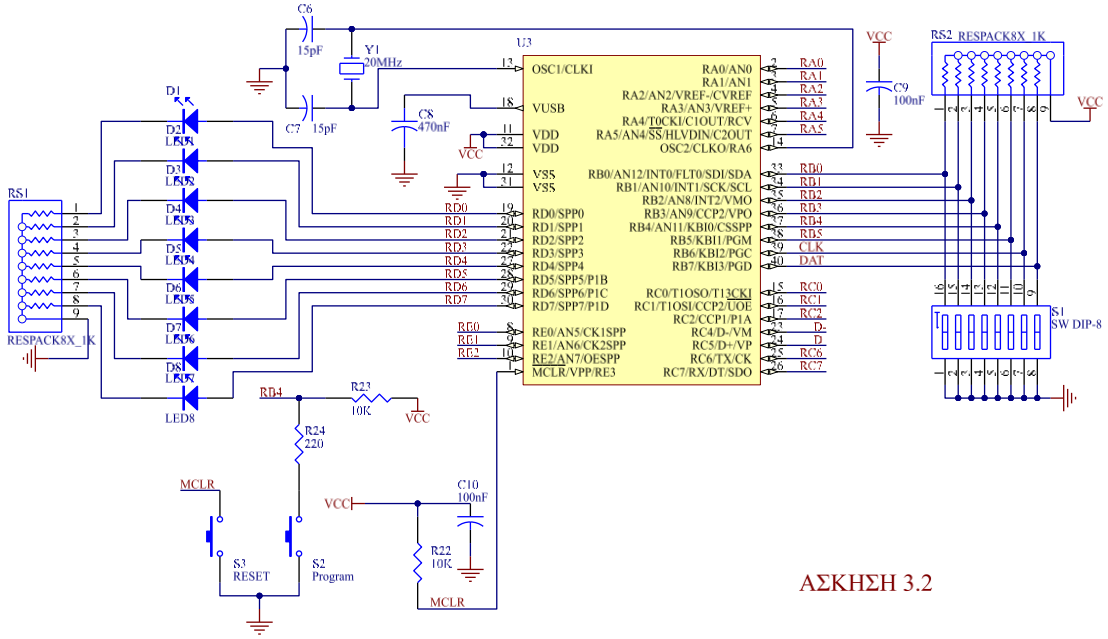
Παράδειγμα της χρήσης της διακοπής από τα RB4,RB5,RB6,RB7.

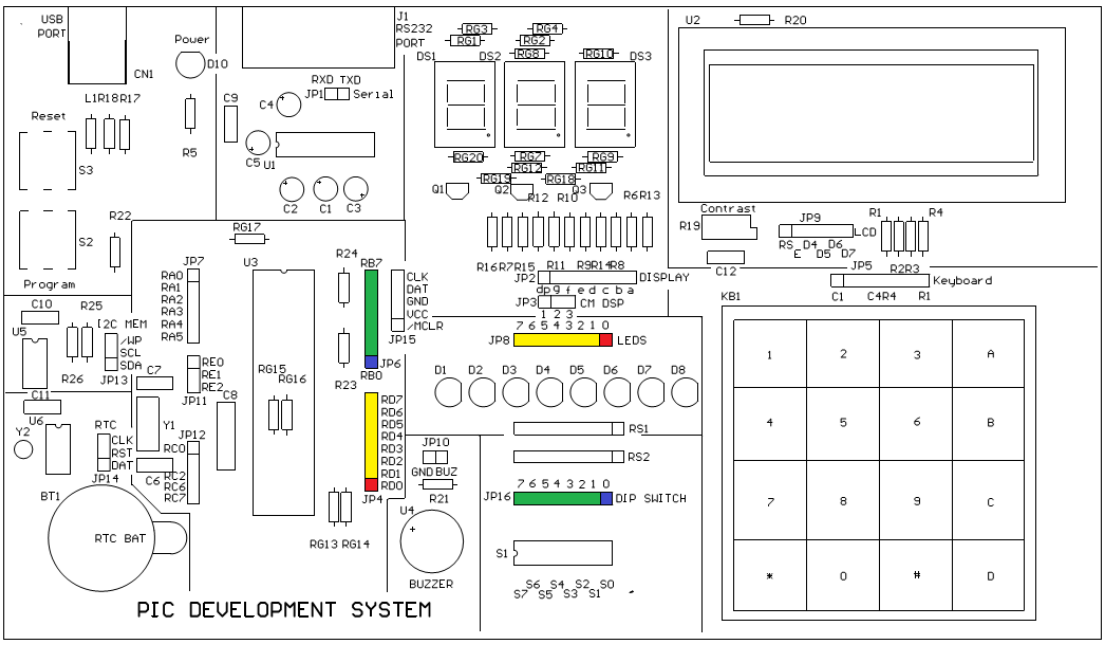
Να γραφεί πρόγραμμα για τον μικροελεγκτή PIC18F4550 που ελέγχει τα bits RB4,RB5,RB6,RB7 με τη μέθοδο της διακοπής και κάθε φορά που αλλάζει η κατάσταση σε κάποιο από τα bits αυτά, να αναβοσβήνει και το αντίστοιχο bit στην έξοδο που θα είναι η πόρτα D, με ρυθμό 1Hz αν η αλλαγή έγινε από λογικό 0 σε λογικό 1 και 5Hz αν η αλλαγή έγινε από λογικό1 σε λογικό 0.

Λύση

Ο πιο εύκολος τρόπος για να εντοπιστεί το bit της αλλαγής είναι να γίνει λογικό XOR μεταξύ της τρέχουσας κατάστασης με τη προηγούμενη. Το bit του αποτελέσματος που τίθεται σε λογικό 1 είναι το bit που άλλαξε. Ο έλεγχος για το αν το bit αυτό ήταν σε λογικό 1 ή σε λογικό 0 μπορεί να γίνει αν ελεγχθεί το bit που άλλαξε και η παρούσα κατάσταση του είναι σε λογικό 1 τότε ήταν σε λογικό 0 άρα η αλλαγή είναι από λογικό 0 σε λογικό 1 ενώ αν βρεθεί σε λογικό 0 τότε το bit αυτό ήταν σε λογικό 1 άρα η αλλαγή που έγινε είναι από λογικό 1 σε λογικό 0.

Το σχηματικό της άσκησης





Η σύνδεση των καλωδιοταινιών στη πλακέτα του εργαστηρίου.

Το πρόγραμμα της άσκησης.

```
#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
byte  last=0;
byte  old=0;
byte changed_bit=0;
int1 state_p =0;
// Δήλωση συναρτήσεων

void rb (void);
void init (void);

void main()
{
     init();
```

```
    while (1){
                if ((changed_bit ^ 0x10) == 0){
                        if (state_p) {
                                        PORTD = PORTD ^ 0x10;
                                        delay_ms(500);
                                }
                                        else {
                                                PORTD = PORTD ^ 0x10;
                                                delay_ms(100);
                                            }
                                        }
                if ((changed_bit ^ 0x20) == 0){
                        if (state_p) {
                                        PORTD = PORTD ^ 0x20;
                                        delay_ms(500);
                                }
                                        else {
                                                PORTD = PORTD ^ 0x20;
                                                delay_ms(100);
                                            }
                                        }
                if ((changed_bit ^ 0x40) == 0){
                        if (state_p) {
                                        PORTD = PORTD ^ 0x40;
                                        delay_ms(500);
                                }
                                        else {
                                                PORTD = PORTD ^ 0x40;
                                                delay_ms(100);
                                            }
                                        }
                if ((changed_bit ^ 0x80) == 0){
                        if (state_p) {
                                        PORTD = PORTD ^ 0x80;
                                        delay_ms(500);
                                }
                                        else {
                                                PORTD = PORTD ^ 0x80;
                                                delay_ms(100);
                                            }
                                        }

            }
}
```

```
// Ορισμός συναρτήσεων
#INT_RB HIGH  // Διακοπή με μεγάλη προτεραιότητα
void rb (void){
                        last = PORTB ^ old;
                        old =  PORTB;
                        if (bit_test(last,4 )&& !bit_test(old,4)){

 //εάν έχει γίνει αλλαγή στο RB4 και το ανάστροφο της τρέχουσας κατάστασης
// του είναι 1 τότε έγινε αλλαγή του RB4 από 1 σε 0
                        changed_bit = 0x10;
                        state_p = 0;  // αλλαγή από 1 σε 0
                                                        }
                        if (bit_test(last,4 )&& bit_test(old,4)){
                            changed_bit = 0x10;
                            state_p = 1; // αλλαγή από 0 σε 1
                                                        }

                        if (bit_test(last,5)&& !bit_test (old,5)){

//εάν έχει γίνει αλλαγή στο RB5 και το ανάστροφο της τρέχουσας κατάστασης
//του είναι 1 τότε έγινε αλλαγή του RB5 από 1 σε 0
                           changed_bit = 0x20;
                           state_p = 0;  // αλλαγή από 1 σε 0
                                                        }
                        if (bit_test(last,5 )&& bit_test(old,5)){
                            changed_bit = 0x20;
                            state_p = 1; // αλλαγή από 0 σε 1
                                                        }

                        if (bit_test(last,6 )&& !bit_test(old,6)){

//εάν έχει γίνει αλλαγή στο RB6 και το ανάστροφο της τρέχουσας κατάστασης
//του  είναι 1 τότε έγινε αλλαγή του RB6 από 1 σε 0

                        changed_bit = 0x40;
                        state_p = 0;  // αλλαγή από 1 σε 0
                                                        }
                        if (bit_test(last,6 )&& bit_test(old,6)){
                                changed_bit = 0x40;
                                state_p = 1; // αλλαγή από 0 σε 1
                                                        }

                       if (bit_test(last,7 )&& !bit_test(old,7)){

//εάν έχει γίνει αλλαγή στο RB7 και το ανάστροφο της τρέχουσας κατάστασης
//του είναι 1 τότε έγινε αλλαγή του RB7 από 1 σε 0
```

```
                        changed_bit = 0x80;
                        state_p = 0;  // αλλαγή από 1 σε 0
                                                }
                if (bit_test(last,7 )&& bit_test(old,7)){
                        changed_bit = 0x80;
                        state_p = 1; // αλλαγή από 0 σε 1
                                                }

        }
void init (void){
        set_tris_b(0xff);     // Καθορισμός της πόρτας B ως είσοδος
        set_tris_d(0x00);      //  Καθορισμός της πόρτας D ως έξοδος
        PORTD =0;
        port_b_pullups(TRUE);
        enable_interrupts(INT_RB);
        enable_interrupts(GLOBAL);

    }
```

Ασκήσεις που θα πρέπει να ετοιμαστούν για το τρίτο εργαστήριο

1. Να γραφεί πρόγραμμα σε γλώσσα προγραμματισμού C για τον μικροελεγκτή PIC 18F4550 με το οποίο να συμβαίνουν τα εξής:
Η πόρτα D να είναι συνδεδεμένη σαν έξοδος και στους ακροδέκτες της να είναι συνδεδεμένα 8 led και η πόρτα B στους διακόπτες της πλακέτας.
Κάθε φορά που αλλάζει η κατάσταση του ακροδέκτη RB1 από λογικό ‘0’ σε λογικό ‘1’ και το bit RB7 είναι σε λογικό “1” να αυξάνει ένα μετρητή που αρχικά είναι στο μηδέν και να βγάζει τον μετρητή στη πόρτα D. Αν το bit RB7 είναι σε λογικό ‘0’ και αλλάζει η κατάσταση στον ακροδέκτη RB2 από λογικό ‘0΄σε λογικό ΄1΄ να μειώνει κατά 1 την τιμή του μετρητή που επίσης θα πρέπει να βγαίνει στη πόρτα D. Το πρόγραμμα θα πρέπει να ελέγχει αν κατά τη μείωση έχει φθάσει στο μηδέν να μένει στο μηδέν και όταν φθάνει στην μέγιστη τιμή 255 να μένει στη τιμή αυτή. Να χρησιμοποιηθεί η μέθοδος της διακοπής στις εισόδους RB1 και RB2.

2, Να γραφεί πρόγραμμα σε γλώσσα προγραμματισμού C για τον μικροελεγκτή PIC18F4550 που να λειτουργεί σαν σύστημα συναγερμού θέσης (δηλαδή να ελέγχει αν μετακινήθηκε ένα αυτοκίνητο).

Οι τέσσερεις ακροδέκτες RB4, RB5, RB6, RB7 είναι συνδεδεμένου σε 4 υδραργυρικούς διακόπτες οι οποίοι είναι τοποθετημένοι στα άκρα ενός σταυρού και κλείνουν όταν το αυτοκίνητο μετακινηθεί προς μια από τις 4 κατευθύνσεις. Στην πλακέτα ανάπτυξης εφαρμογών του εργαστηρίου αυτοί οι τέσσερις διακόπτες αντιστοιχούν στα 4 dip switches που συνδέονται στα

τέσσερα περισσότερο σημαντικά διακοπτάκια και θα πρέπει η κατάστασή τους να εμφανίζεται στα τέσσερα περισσότερο σημαντικά leds που συνδέονται στη πόρτα D.

Ο οπλισμός του συστήματος, δηλαδή η ενεργοποίηση του, γίνεται από τον ακροδέκτη RB2. Όταν RB2=0 το σύστημα είναι οπλισμένο (ενεργοποιημένο). Όταν RB2=1 το σύστημα είναι απενεργοποιημένο.

Αν το σύστημα είναι ενεργοποιημένο και αλλάξει η κατάσταση σε τουλάχιστον έναν από τους ακροδέκτες RB4,…RB7 και το σύστημα συνεχίζει να είναι ενεργοποιημένο και για τα επόμενα 4 sec, τότε να χτυπά μια σειρήνα για 5 sec και στη συνέχεια το σύστημα περιμένει για νέα αλλαγή κατάστασης στους ακροδέκτες RB4 έως PB7.

Στη πλακέτα ανάπτυξης εφαρμογών του εργαστηρίου η σειρήνα προσομοιώνεται με ένα led το οποίο είναι συνδεδεμένο στον ακροδέκτη RD1 της πόρτας D.

Υπόδειξη:
Θα ενεργοποιηθεί διακοπή που προκαλείται από αλλαγή κατάστασης στους ακροδέκτες RB4 έως RB7.

Το κύριο πρόγραμμα δείχνει τη κατάσταση των εισόδων στα leds της πόρτα D .

Στη ρουτίνα εξυπηρέτησης της διακοπής θα ελέγχεται αν είναι ενεργοποιημένος ο συναγερμός, στη συνέχεια θα υπάρχει μια αναμονή για 4 sec και νέος έλεγχος για το αν συνεχίζει να είναι ενεργοποιημένος ο συναγερμός. Αν συνεχίζει να είναι ενεργοποιημένος ο συναγερμός τότε θα χτυπάει μια σειρήνα για 5 sec, δηλαδή στη πλακέτα θα ανάβει το led που είναι συνδεδεμένο στο RD1 για 5 sec.

3. Να γραφεί πρόγραμμα σε γλώσσα προγραμματισμού C για το μικροελεγκτή PIC18F4550 που να λειτουργεί ως εξής. Το σύστημα θα δέχεται δύο διαφορετικές διακοπές από τις εισόδους RB0, RB2, στη κάθοδο του παλμού. Όταν δέχεται διακοπή από την είσοδο RB0 να αναβοσβήνει τα Leds που θα είναι συνδεδεμένα στην πόρτα D του PIC με ρυθμό 4 Hz. Ενώ όταν δέχεται διακοπή από την είσοδο RB2, να αναβοσβήνει τα Leds που θα είναι συνδεδεμένα στην πόρτα D του PIC με ρυθμό 10 Hz.

Ερωτήσεις που θα πρέπει να απαντηθούν (Προαιρετικά). Οι απαντήσεις στέλνονται στο Email angmicro2@gmail.com

1.Σε ένα πρόγραμμα που παίρνει 3 δείγματα από την αναλογική είσοδο ενός μικροελεγκτή με τιμές του κάθε δείγματος 0-100 και διαιρεί δια 3 για να βγάλει

τον μέσο όρο ποιος είναι ο κατάλληλος τύπος της μεταβλητής για το αποτέλεσμα

a. int1

b. int16

c. int8

d. int32

e. float

2. Ποιο το αποτέλεσμα της πράξης 2+3*5 σε C πρόγραμμα.

a. 25

b. 17

c, 15.

d. 18

3. Στο παρακάτω πρόγραμμα γίνεται μετατροπή ενός τριψήφιου ASCII σε ένα ακέραιο διορθώστε τα λάθη .

int16 value;

char c1,c2,c3;

c1='3';

c2='2';

c3='1';

value = (c1- '0') * 100 + (c2-'0') * 10 + (c3-'0') ;

4. Αν όλες οι μεταβλητές είναι τύπου int8 ποιες από τις παρακάτω εκφράσεις θα δώσουν διαφορετικό αποτέλεσμα

a. x % 8

b. x & 7

c. (x<<5)>>5

d. x>7 ? x-8 : x

e. όλες οι εκφράσεις θα δώσουν διαφορετικό αποτέλεσμα.

5. Αν η αρχική τιμή του x είναι 0 ποια θα είναι η τιμή του x μετά την πράξη

x= ++++++--x;

a. 2
b. 258
c. 3
d. 0
e. Δεν είναι σωστή η έκφραση στην C

6. Σε πόσους διαφορετικούς δρόμους οδηγεί το παρακάτω πρόγραμμα.

```
If (aa)
  If (bb)
 cc=1;
    else
        if (dd)
            cc=2;
        else
            cc=3;
```

a. Σε κανένα
b. Σε 2
c. Σε 3
d. Σε 4
e. Δεν είναι σωστή έκφραση στην C.

7. Ποια θα είναι η τιμή του x μετά την εκτέλεση του παρακάτω κώδικα

```
x=0;
for(i=1;i <=5; i++)
    for(j=1; j<=5; j++,x++)
        if(j==i)
            break;
```

a. 0

b. 5

c. 10

d. 25

e. Το πρόγραμμα δεν είναι σωστό.