ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εργαστήριο Μικροϋπολογιστών
6η Εργαστηριακή Άσκηση

Ομάδα: 64
Μέλη: Κωνσταντίνος Ιωάννου (ΑΜ: 031-19840)
Γιώργος Μυτιληναίος (ΑΜ: 031-19841)

Ημερομηνία εργαστηριακής εξέτασης: Τρίτη 7 Δεκεμβρίου 2022
Ημερομηνία υποβολής: Κυριακή 11Δεκεμβρίου 2022

Ζήτημα 6.1

```
int *scan_row(int i) //NEW EDITION
{
  static int row[4]  = {0,0,0,0};
  uint8_t y ,z;
  int x;
  //x =pow(2,i);
  row[1]=row[2]=row[3]=row[0]=0;

PCA9555_0_write(REG_OUTPUT_1, i);
y=PCA9555_0_read(REG_INPUT_1);
y=y&0xF0;
y=y>>4;
//y = 0000y3y2y1y0
  x=0b00000001;

  for (int j=0;j<4;j++) {
      z=x&y ;
      x=x*2;
      if(z== 0b000000000)    {  row[j]=1; }
    }
return row;
}

 int *scan_keypad()
  { int z = 0 ;
   static int  keypad[16] ;
    int *temp;

     int i ;
     for(int x=0;x<4;x++)
   {

         switch(x) {
         case 0:
             i=0b1110;
             break;
         case 1:
              i=0b1101;
             break;
         case 2:
             i=0b1011;
             break;
         case 3:
             i=0b0111;
             break;
         }

        //i = pow(2,x) ;
        temp=scan_row(i);

         for(int j=0;j<4;j++){
          keypad[z+j]= *(temp+j);
   }
        z=z+4;
     }


return keypad;
}
```

Περιγραφή κώδικα :

Η **συνάρτηση scan_row(int i)** δέχεται ως όρισμά έναν δυαδικό αριθμό I που κάθε φορά ενεργοποιεί την συγκεκριμένη σειρά που θέλουμε στο πληκτρολόγιο (πχ 0b1110 ενεργοποιούμε την πρώτη γραμμή στο πληκτρολόγιο).Αυτό το κάνουμε μέσω της εντολής PCA9555_0_**write.** Ενώ μέσω της εντολής **read** διαβάζουμε την είσοδο και μεταφέρουμε αυτά τα 4MSB στα LSB και τα απομονώνουμε. Έχουμε λοιπόν ένα byte που έχει 0 μηδενικά όπου το κουμπί της γραμμής που κοιτάμε ήταν πατημένο και 1 διαφορετικά. Τέλος μέσω ενός loop απομονώνουμε ένα ένα αυτά τα τέσσερα bits και κοιτάζουμε ποια από αυτά ήταν μηδέν δηλαδή ποια ήταν πατημένα και όσα ήταν τα περνάμε σε ένα πίνακα ως άσσο και τα υπόλοιπα ως μηδέν . Αυτόν τον πίνακα τεσσάρων θέσεων είναι που επιστρέφει η συνάρτηση μας ώστε να ξέρουμε σε μια γραμμή ποια μποτών ήταν πατημένα και ποια όχι.

Η συνάρτηση **scan_keypad()** ουσιαστικά καλεί την scan_row 4 φορές μία για κάθε γραμμή και αποθηκεύει το αποτέλεσμα κάθε γραμμής σε ένα μονοδιάστατο πίνακα 16 θέσεων (βοηθά να τον σκεφτούμε ως δισδιάστατο [4][4]) όπου οι πρώτες τέσσερις θέσεις είναι για την πρώτη γραμμή οι επόμενες τέσσερις για την δεύτερη γραμμή και πάει λέγοντας. Επίσης έχουμε ένα switch ώστε ανάλογα σε ποια επανάληψη βρισκόμαστε να καλούμε την scan_row για το σωστό i δηλαδή ενεργοποιώντας την σωστή γραμμή .Αυτή η συνάρτηση επιστρέφει τον πίνακα 16 θέσεων που αναφέραμε πριν όπου σε όποια θέση έχει άσσο σημαίνει ότι το κουμπί στο πληκτρολόγιο ήταν πατημένο διαφορετικά έχει μηδέν.

```
int *scan_keypad_rising_edge()
{
    static int templ[16] ;
    static int temp2[16] ;
    int *t;

     t = scan_keypad();
     for(int j=0;j<16;j++){
          templ[j]= *(t+j);
     }
    _delay_ms(15);  // (10-20ms);

     t = scan_keypad();
     for(int j=0;j<16;j++){
          temp2[j]= *(t+j);
     }


for(int i =0 ;i<16;i++) {
    if(templ[i] != temp2[i]) templ[i] = 0;
    }


     return templ;
}

const char keypad_to_ascii(int temp[])
{
   static int j ;
   int flag=0;
   char str ;
   char keypad_layout[16]={ '*','0','#','D',
                            '7','8','9','C',
                           '4','5','6','B',
                           '1','2','3','A' }  ;

   for( j=00;j<16;j++) {
      if(temp[j]==1) {
          flag=1;
       goto key_found;
    }
   }

key_found:
       if(flag==1){
       str = keypad_layout[j];
       return str;
       }
       else return ' ';
}
```

Η συνάρτηση **scan_keypad_rising_edge()** έχει ως σκοπό να αποφεύγεται ο σπινθηρισμός . Αρχικά καλεί την scan_keypad() (περνάει τις τιμές σε έναν πίνακα από τον δείκτη για ευκολία στην χρήση) και παίρνει ένα στιγμιότυπο του πληκτρολογίου (temp1 πίνακας) .Έπειτα γίνεται μια καθυστέρηση και καλούμε ξανά την scan_keypad και παίρνουμε ένα νέο στιγμιότυπο του πληκτρολογίου (temp2 πίνακας).Συγκρίνουμε λοιπόν αυτά τα δύο στιγμιότυπα και δημιουργούμε έναν νέο πίνακα που κρατάει ως άσους (δηλαδή πατημένα κουμπιά) μόνο όπου και τα δυο στιγμιότυπα είχαν άσσο. Με αυτόν τον τρόπο εξασφαλίζουμε ότι αν ένα μπουτόν πατηθεί λόγω σπινθηρισμού για ένα ελάχιστο χρονικό διάστημα στο πρώτο στιγμιότυπο μετά την καθυστέρηση δεν θα είναι πατημένο οπότε και δεν θα το λάβουμε υπόψιν μας.

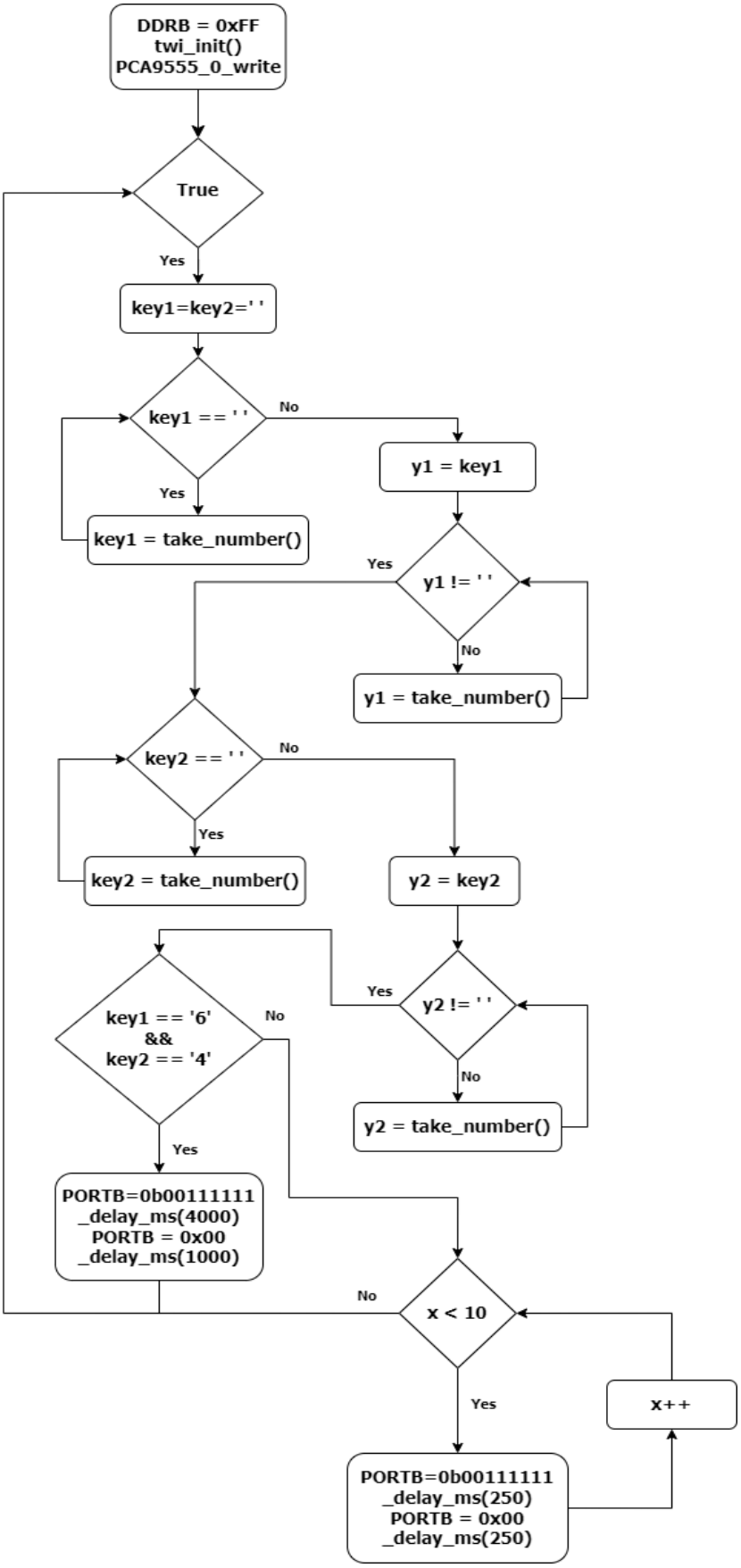
Η συνάρτηση **keypad_to_ascii(int temp [])** δέχεται ως όρισμα έναν πίνακα συγκεκριμένα αυτό που επιστρέφει η scan_keypad_rising_edge() και μέσω ενός loop τσεκάρει τα 16 στοιχεία του πίνακα μέχρι να βρει άσσο .Όταν βρει άσσο σπάει από το loop κρατάει την θέση που βρήκε τον πίνακα και επιστρέφει τον χαρακτήρα που βρίσκεται στην αντίστοιχη θέση του πίνακα keypad_layout ο οποίος είναι ένας πίνακας 16 θέσεων ο οποίος αναπαριστά τα σύμβολα που έχει το πληκτρολόγιο με την σειρά. Αν τώρα δεν βρει άσσο δηλαδή πατημένο κουμπί στο πληκτρολόγιο απλώς επιστρέφει τον κενό ‘ ‘ χαρακτήρα.

```
int main()
{
    DDRD |= 0b11111111;  // output for LCD
    lcd_init();

int keypad[16];
int *temp;
char key=' ';
char temp_key;

twi_init();
PCA9555_0_write(REG_CONFIGURATION_1, 0xF0);



while(1){

temp = scan_keypad_rising_edge();

      for(int j=0;j<16;j++){
        keypad[j]= *(temp+j); }


       temp_key = keypad_to_ascii(keypad);


    if(temp_key!=' ') key =temp_key;

        lcd_command(0x01);
       _delay_ms(3);
       lcd_data(key);
       _delay_ms(25);

}
return 0;
}
```

Η **main()** αρχικά θέτει την οθόνη lcd ως output και την αρχικοποιεί ,ενώ στην συνέχεια αρχικοποιεί το πληκτρολόγιο και μέσω της εντολής PCA9555_0_write(REG_CONFIGURATION_1, 0xF0) Θέτει (11110000) τις θύρες IO1_0-IO1_3 ως output και τις θύρες IO1_4-IO1_7 ως input για όλη την διάρκεια του κώδικα. Τέλος μέσα σε μια συνεχόμενη loop καλούμε την scan_keypad_rising_edge() για να δούμε ποια κουμπιά είναι πατημένα στο πληκτρολόγιο , έπειτα μέσω της keypad_to_ascii παίρνουμε τον χαρακτήρα από το πρώτο κουμπί που είναι πατημένο και με την εντολή lcd_data το εκτυπώνουμε στην οθόνη. Τονίζουμε ότι χρησιμοποιούμε την temp_key ώστε όταν δεν πατάμε κάποιο κουμπί να εκτυπώνει συνεχώς (με μικρή καθυστέρηση) τον ίδιο χαρακτήρα με πριν μέχρι να πατήσουμε εμείς νέο κουμπί .Δηλαδή η οθόνη δείχνει πιο κουμπί πατήσαμε τελευταίο.

Σημειώνουμε ότι τις έτοιμες συναρτήσεις για το πληκτρολόγιο όσο και για την οθόνη που μας δόθηκαν στις εκφωνήσεις των ασκήσεων δεν τις δείχνουμε εδώ λόγο χώρου αλλά και για να υπάρχει συνοπτικά η ιδέα και η υλοποίηση του προγράμματος που εμείς γράψαμε, προφανώς στον κώδικα υπάρχουν οι ανάλογες συναρτήσεις .

Ζήτημα 6.2



Διάγραμμα ροής για το ζήτημα 6.2 στην συνέχεια ακολουθεί και η παρουσίαση του κώδικα για αυτό το ζήτημα

```
char  take_number() {

   int keypad[16];
   int *temp;
   char key;

   temp = scan_keypad_rising_edge();
    for(int j=0;j<16;j++){
            keypad[j]= *(temp+j);
        }

    key = keypad_to_ascii(keypad);

   return key;

  }


int main()
{
DDRB = 0xFF; // B as output

twi_init();
PCA9555_0_write(REG_CONFIGURATION_1, 0xF0);

char key1 ,key2,y1,y2;

while(1){

  key1=key2=' ';

 while(key1==' ') {
  key1 = take_number();
  }
  y1=key1;
  while(y1!=' ')
  {
   y1 = take_number();
  }

  while(key2==' ') {
  key2 = take_number();
  }
  y2 = key2;
  while(y2!=' ')
  {
   y2 = take_number();
  }

  if(key1=='6' && key2 == '4') {
    PORTB=0b00111111;
     _delay_ms(4000);
     PORTB = 0x0;
     _delay_ms(1000);
     }

  else { //blink();

    for (int x=0;x<10;x++ ) {
    PORTB=0b00111111;
     _delay_ms(250);
     PORTB = 0x0;
     _delay_ms(250);
    }
    }

  }
return 0;
 }
```

Αρχικά χρησιμοποιούμε όλες τι συναρτήσεις που δημιουργήσαμε στο προηγούμενο ζήτημα 6.1

Η συνάρτηση **take_number()** επιστρέφει τον χαρακτήρα του κουμπιού που είναι πατημένο στο πληκτρολόγιο, σε αυτήν την άσκηση πατάμε ένα κουμπί την κάθε φορά.

Η **main()** αρχικά κάνει ότι έκανε και στην προηγούμενη άσκηση ,αλλά τώρα ορίζει τέσσερεις μεταβλητές χαρακτήρες char1,char2 ,y1,y2 .Στην συνεχόμενη loop ορίζουμε αρχικά τα key1 και key2 την κενή συμβολοσειρά. Στο πρώτο while(key1= = ‘ ‘) διαβάζουμε συνεχώς το πληκτρολόγιο μέχρι να πατήσουμε ένα κουμπί και να δώσουμε τον χαρακτήρα από αυτό το κουμπί στο key1.Με αυτήν την πρακτική περιμένει το πρόγραμμα μας μέχρι να πατήσουμε κάποιο κουμπί. Με το δεύτερο while(y1 != ‘ ‘) όπου y1 δίνουμε τον χαρακτήρα του key1 αποτρέπουμε το όσο είναι πατημένο ένα κουμπί να διαβαστεί πολλαπλές φορές καθώς αυτή η loop περιμένει μέχρι να δίνει η take_number() τον κενό χαρακτήρα δηλαδή μέχρι να μην πατάμε κανένα πλήκτρο στο πληκτρολόγιο. Στην συνέχεια κάνουμε ακριβώς το ίδιο και για το δεύτερο μέρος χαρακτήρα του κωδικού με τα key2 και y2. Πλέον μέσα από το if εξετάζουμε αν δώσαμε τον σωστό κωδικό ή όχι εξετάζοντας αν τα key1 και key2 έχουν τους σωστούς χαρακτήρες. Αν τους έχουν τότε ανάβουν τα led της portb που θέλουμε για 4 sec μετά σβήνουν και μετά έχουμε καθυστέρηση για ακόμη 1 sec ώστε να μην μπορούμε να πατήσουμε νέο πλήκτρο για 5 sec όπως λέει η εκφώνηση .Αν τώρα είναι λάθος οι χαρακτήρες που δώσαμε τα led της portb αναβοσβήνουν μέσα σε ένα loop με την κατάλληλη συχνότητα (2Hz) για 5 sec.Μετά τα key1,key2 παίρνουν την κενή συμβολοσειρά και το κεντρικό loop γίνεται από την αρχή.