A close up of text

Description automatically generated

ΑΘΗΝΑ 8 Νοεμβρίου 2024

**6η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ**

**ΓΙΑ ΤΟ ΜΑΘΗΜΑ “Εργαστήριο Μικροϋπολογιστών”**

**ΟΜΑΔΑ 23**

**Συνεργάτες**

Νικόλαος Αναγνώστου Νικόλαος Λάππας

03121818 03121098

**Ζήτημα 6.1:** Στην συγκεκριμένη άσκηση καλούμαστε να ελέγξουμε την ενεργοποίηση τεσσάρων πλήκτρων από τα δεκαέξι συνολικά πλήκτρα του πληκτρολογίου της πλακέτας. Κάθε φορά που ανιχνεύεται ένα από τα ζητούμενα πλήκτρα, ανάβουμε το αντίστοιχο Led της πύλης PORTB. Αυτό που κάνουμε είναι να χρησιμοποιήσουμε, αρχικά, τις δοσμένες συναρτήσεις που ελέγχουν την εγγραφή και την ανάγνωση ενός από τους καταχωρητές ελέγχου του ολοκληρωμένου PCA9555 και στην συνέχεια ορίζουμε τις τέσσερις συναρτήσεις που ζητάει η εκφώνηση. Θα αναλύσουμε τις ζητούμενες συναρτήσεις με την σειρά που καλούνται στο πρόγραμμά μας. Συγκεκριμένα,

* Η συνάρτηση , παίρνει την 16-bit τιμή που επιστρέφει η και την χωρίζει σε τέσσερις τετράδες. Κάθε τετράδα την αποθηκεύει στα τέσσερα LSB μιας νέας μεταβλητής η οποία δείχνει σε ποια γραμμή βρισκόμαστε. Σε αυτά τα τέσσερα bits που μας ενδιαφέρουν, είναι αποθηκευμένη η πληροφορία των στηλών που έχουν ενεργοποιηθεί στην συγκεκριμένη γραμμή. Επομένως, περνάμε σε έναν έλεγχο μετά αναλόγως σε ποια γραμμή βρισκόμαστε, και εξετάζουμε ποια στήλη έχει ενεργοποιηθεί, και ανάλογα στέλνουμε το κατάλληλο input (αντιστοίχιση που κάνουμε με τα πλήκτρα). Για παράδειγμα, το πλήκτρο ‘6’ έχει κωδική αντιστοίχιση με τον δυαδικό αριθμό ‘1011 1011’, με τα πρώτα τέσσερα MSB να αφορούν την στήλη και τα τέσσερα LSB να αφορούν την γραμμή. Έτσι στην main συνάρτηση κάνουμε τους απαραίτητους ελέγχους προκειμένου να αγνοούμε κάθε άλλο πλήκτρο εκτός των τεσσάρων που ζητούνται, και από αυτά τα τέσσερα κάθε φορά να αντιλαμβανόμαστε ποιο έχει πατηθεί.
* Η συνάρτηση , είναι υπεύθυνη να αντιμετωπίζει το πρόβλημα του σπινθηρισμού. Συγκεκριμένα, παίρνουμε μέσω της το πρώτο κουμπί που θεωρητικά πατιέται, και μετά από 10 msec ξανακαλούμε την ίδια συνάρτηση και παίρνουμε ακόμα μια είσοδο. Από την δεύτερη τιμή αγνοούμε τα bit που διαφέρουν σε σχέση με την πρώτη τιμή εισόδου, και θα συγκρίνουμε το τελικό αποτέλεσμα με την προηγούμενη τιμή που έχουμε αποθηκευμένη σε κάθε νέα κλήση της εν λόγω συνάρτησης. Επιστρέφουμε τελικά την ελεγμένη τιμή πλήκτρου που πατήθηκε.
* Η συνάρτηση , καλεί τέσσερις φορές την συνάρτηση , μια φορά για κάθε γραμμή, και την 8-bit τιμή που επιστρέφεται σε αυτή, την κάνει shift αριστερά τέσσερις θέσεις (τα 4 LSB της 8-bit τιμής μας είναι χρήσιμα), προκειμένου να αποθηκευτεί όλη η χρήσιμη πληροφορία από αυτές τις τέσσερις κλήσεις, σε μια μόνο 16-bit μεταβλητή. Αυτή η μεταβλητή περιέχει τέσσερα συνεχόμενα bit για κάθε σειρά που δείχνουν ποια στήλη κάθε σειρά είναι ενεργοποιημένη (έχει τιμή μηδέν). Για παράδειγμα, αν παίρναμε ‘0111 1011 1101 1110’ θα ξέραμε ότι οι στήλες 4 της γραμμής 1, 3 της γραμμής 2, 2 της γραμμής 1 και 1 της γραμμής 4 έχουν ενεργοποιηθεί.
* Η συνάρτηση , ανάλογα με το ποια γραμμή ελέγχουμε κάθε φορά, μπαίνει στο αντίστοιχο case i, και ενεργοποιεί μια μάσκα, την μάσκα για την αντίστοιχη γραμμή i. Αυτή την μάσκα την χρησιμοποιούμε μετά προκειμένου να ορίσουμε ποιας γραμμής στήλες θέλουμε να εξετάσουμε. Συγκεκριμένα, κάνουμε , το οποίο …. και μετά με το οποίο παίρνουμε το 8-bit δεδομένο εισόδου που αφορά τις ενεργοποιημένες στήλες τις γραμμής. Κάνουμε λογικό και απομονώνουμε τα τέσσερα MSB που αφορούν την είσοδο μας (στήλες) και το κάνουμε shift δεξιά τέσσερις θέσεις για να το έχουμε στα LSB για καλύτερη διαχείριση.

Ακολουθεί ο κώδικας με τις παραπάνω συναρτήσεις και την main loop μας, χωρίς τον κώδικα για το ολοκληρωμένο, ο οποίος έχει δοθεί.

#define ROW\_1\_MASK 0xFE // Row 1 low to check only that

#define ROW\_2\_MASK 0xFD // Row 2 low to check only that

#define ROW\_3\_MASK 0xFB // Row 3 low to check only that

#define ROW\_4\_MASK 0xF7 // Row 4 low to check only that

uint8\_t scan\_row(uint8\_t row\_to\_check)

{

uint8\_t row\_mask;

switch (row\_to\_check)

{

case 1: row\_mask = ROW\_1\_MASK; break;

case 2: row\_mask = ROW\_2\_MASK; break;

case 3: row\_mask = ROW\_3\_MASK; break;

case 4: row\_mask = ROW\_4\_MASK; break;

default: return 0; // If invalid row

}

PCA9555\_0\_write(REG\_OUTPUT\_1, row\_mask);

uint8\_t input = PCA9555\_0\_read(REG\_INPUT\_1); // Read the input

// Mask to take the column of keypad from input pressed

// Return, the shifted to the right, result representing the active columns of the specific row

return (input & 0xF0) >> 4;

}

uint16\_t scan\_keypad()

{

uint16\_t bottoms\_pressed = 0; // None pressed at the beginning

for (uint8\_t row = 1; row <= 4; row++)

{

uint8\_t columns\_selected = scan\_row(row); // Get the columns activated from each row

bottoms\_pressed |= columns\_selected;

// For each row, save the columns pressed into the 16-bit bottoms\_pressed

// For row 1, the 4-bit answer will be in the 4 MSB bits of bottoms\_pressed

// For row 2, the 4-bit answer will be in the 4 LSB of the upper half (8-bit) and so on ...

if (row != 4) {bottoms\_pressed = bottoms\_pressed << 4;}

\_delay\_ms(1);

}

return bottoms\_pressed; /\* e.g if bottoms\_pressed = 0111 1011 1101 1110 means that

\* from row 1, column 4 is activated

\* from row 2, column 3 is activated

\* from row 3, column 2 is activated

\* from row 4, column 1 is activated

\*/

}

uint16\_t pressed\_keys\_tempo = 0xFFFF; // First key pressed is 'none', save previous pressed key

uint16\_t scan\_keypad\_rising\_edge()

{

uint16\_t pressed\_keys = scan\_keypad();

\_delay\_ms(10);

uint16\_t recheck\_pressed\_keys = scan\_keypad(); // Check again (de-bouncing)

recheck\_pressed\_keys |= pressed\_keys; // Get rid of the 'bits' that weren't previously pressed

recheck\_pressed\_keys |= ~pressed\_keys\_tempo; // Compare with the previous pressed keys

pressed\_keys\_tempo = pressed\_keys; // Save the current pressed keys into tempo for future use

return recheck\_pressed\_keys; // Return verified key

}

uint16\_t key;

uint8\_t keypad\_to\_ascii()

{

key = scan\_keypad\_rising\_edge(); // 16-bit

if (key == 0xFFFF) {return 0x00;} // Nothing is pressed

uint8\_t row1 = (key >> 12) & 0x0F; // Extract bits 12-15 (Row 1)

uint8\_t row2 = (key >> 8) & 0x0F; // Extract bits 8-11 (Row 2)

uint8\_t row3 = (key >> 4) & 0x0F; // Extract bits 4-7 (Row 3)

uint8\_t row4 = key & 0x0F; // Extract bits 0-3 (Row 4)

for (uint8\_t row = 1; row <= 4; row++)

{

switch (row)

{

case 1:

if (row1 == 0b00001110) {return 0b11101110; break;} // "\*"

else if (row1 == 0b00001101) {return 0b11011110; break;}// "0"

else if (row1 == 0b00001011) {return 0b10111110; break;}// "#"

else if (row1 == 0b00000111) {return 0b01111110; break;}// "D"

case 2:

if (row2 == 0b00001110) {return 0b11101101; break;} // "7"

else if (row2 == 0b00001101) {return 0b11011101; break;}// "8"

else if (row2 == 0b00001011) {return 0b10111101; break;}// "9"

else if (row2 == 0b00000111) {return 0b01111101; break;}// "C"

case 3:

if (row3 == 0b00001110) {return 0b11101011; break;} // "4"

else if (row3 == 0b00001101) {return 0b11011011; break;}// "5"

else if (row3 == 0b00001011) {return 0b10111011; break;}// "6"

else if (row3 == 0b00000111) {return 0b01111011; break;}// "B"

case 4:

if (row4 == 0b00001110) {return 0b11100111; break;} // "\*"

else if (row4 == 0b00001101) {return 0b11010111; break;}// "0"

else if (row4 == 0b00001011) {return 0b10110111; break;}// "#"

else if (row4 == 0b00000111) {return 0b01110111; break;}// "D"

}

}

}

int main(void)

{

twi\_init();

DDRB = 0xFF; // Initialize PORTB as output

PCA9555\_0\_write(REG\_CONFIGURATION\_1, 0xF0); // Set IO1[0:3] as output and IO1[4:7] as input (set)

while(1)

{

uint8\_t input = keypad\_to\_ascii(); // Read input

if (input == 0b01110111) PORTB = 0x01; // Pressed 'A'

else if (input == 0b10111011) PORTB = 0X04; // Pressed '6'

else if (input == 0b11011101) PORTB = 0x02; // Pressed '8'

else if (input == 0b11101110) PORTB = 0x08; // Pressed '\*'

else PORTB = 0x00; // None of the desired keys is pressed

}

}

**Ζήτημα 6.2:** Η συγκεκριμένη άσκηση αποτελεί μια παραλλαγή της προηγούμενης, και συγκεκριμένα, αντί να ανάβουμε προκαθορισμένα Led της θύρας εξόδου PORTB, χρησιμοποιούμε την LCD οθόνη και εκτυπώνουμε εκεί το αποτέλεσμα μας. Πέρα από τις συναρτήσεις που χρειαστήκαμε στην άσκηση 6.1, σε αυτή εισαγάγαμε και τις συναρτήσεις για την οθόνη από την 4η σειρά, όπου χρησιμοποιούσε την θύρα D και όχι το ολοκληρωμένο κύκλωμα για την οπτικοποίηση των δεδομένων. Αυτό που κάνουμε είναι να προσθέσουμε έξτρα το , το οποίο …. και μετά ανάλογα με την τιμή που θα επιστρέψει η συνάρτηση , θα μπούμε σε έναν από τους 16 συνολικούς ελέγχους και θα τυπώσουμε στην οθόνη τον αντίστοιχο χαρακτήρα.

Ακολουθεί η main συνάρτηση μας.

int main(){

DDRD = 0xFF;

twi\_init();

PCA9555\_0\_write(REG\_CONFIGURATION\_1, 0xF0);

PCA9555\_0\_write(REG\_CONFIGURATION\_0, 0x00);

lcd\_init();

while(1)

{

//lcd\_clear\_display();

lcd\_command(0x80);

uint8\_t input = keypad\_to\_ascii(); // Read input

if (input == 0b01110111) lcd\_data('A'); // Pressed 'A'

else if (input == 0b10110111) lcd\_data('3'); // Pressed '3'

else if (input == 0b11010111) lcd\_data('2'); // Pressed '2'

else if (input == 0b11100111) lcd\_data('1'); // Pressed '1'

else if (input == 0b01111011) lcd\_data('B'); // Pressed 'B'

else if (input == 0b10111011) lcd\_data('6'); // Pressed '6'

else if (input == 0b11011011) lcd\_data('5'); // Pressed '5'

else if (input == 0b11101011) lcd\_data('4'); // Pressed '4'

else if (input == 0b01111101) lcd\_data('C'); // Pressed 'C'

else if (input == 0b10111101) lcd\_data('9'); // Pressed '9'

else if (input == 0b11011101) lcd\_data('8'); // Pressed '8'

else if (input == 0b11101101) lcd\_data('7'); // Pressed '7'

else if (input == 0b01111110) lcd\_data('D'); // Pressed 'D'

else if (input == 0b10111110) lcd\_data('#'); // Pressed '#'

else if (input == 0b11011110) lcd\_data('0'); // Pressed '0'

else if (input == 0b11101110) lcd\_data('\*'); // Pressed '\*'

//else lcd\_clear\_display(); // None of the desired keys is pressed

//\_delay\_ms(2000);

}

}

**Ζήτημα 6.3:**