A close up of text

Description automatically generated

ΑΘΗΝΑ 1 Νοεμβρίου 2024

**5η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ**

**ΓΙΑ ΤΟ ΜΑΘΗΜΑ “Εργαστήριο Μικροϋπολογιστών”**

**ΟΜΑΔΑ 23**

**Συνεργάτες**

Νικόλαος Αναγνώστου Νικόλαος Λάππας

03121818 03121098

**Ζήτημα 5.1:**

Στην συγκεκριμένη άσκηση, ρυθμίσαμε το IO\_PORT\_0 ως έξοδο και μέσα στον βρόχο διαβάζουμε συνεχώς το input από το PIND (A, B, C, D) και υπολογίζουμε τις ζητούμενες λογικές συναρτήσεις. Το αποτέλεσμα το στέλνουμε στο PCA9555 στο IO\_PORT\_0 μέσω του REG\_OUTPUT\_0, αποτέλεσμα το οποίο έχουμε ορίσει μέσω καλωδίων να εμφανίζεται στα LED PD2 και PD3. Παρακάτω ακολουθεί ο κώδικας χωρίς τις δοθείσες από την εκφώνηση συναρτήσεις.

int main(){

DDRC = 0xFF; //is used by the twi -> pc4 pc5

DDRB = 0xF0; //pb0-pb3 is input

int F0,F1;

DDRD = 0xFF; //output

twi\_init();

PCA9555\_0\_write(REG\_CONFIGURATION\_0, 0x00); //IO\_0 AND I0\_1 is set to output

uint8\_t alpha,beta,current,result;

while(1){

PORTD &= 0x0C; //ordering every pd unrelative to the exercise to remain off for eternity

//smart mechanism for doing the logic of the exercise 5.1

current = (~PINB)&(0x0F);

alpha = (current&0x07); //ABC

beta = (current&0x0A); //BD

//check F0

if(alpha != 0x06 && beta != 0x08) F0 = 0x01;

else F0 = 0x00;

//check F1

if(alpha>0x00 && beta==0x02) F1 = 0x02;

else F1 = 0;

//the f0 f1 are now calculated and now its the time to be pushed with twi to IO0\_0 ans IO0\_1

result = F0 | F1;

PCA9555\_0\_write(REG\_OUTPUT\_0, result);

}

}

**Ζήτημα 5.2:**

Στο συγκεκριμένο ερώτημα κληθήκαμε να ανάβουμε τα λαμπάκια PD4, PD5, PD6 και PD7 ξεχωριστά όταν πατιέται το κουμπί ‘\*’, ‘0’, ‘#’ και ‘D’ αντίστοιχα από το πληκτρολόγιο που υπάρχει στην πλακέτα. Την λειτουργία αυτή την υλοποιήσαμε κάνοντας χρήση του I2C επικοινωνιακού διαύλου και συγκεκριμένα συνδέσαμε τους ακροδέκτες IO0\_0-IO0\_3 της θύρας επέκτασης 0 ως εξόδους (REG\_CONFIGURATION\_0) και τα ενώσαμε με καλώδια στα LED\_PD4-LED\_PD7 προκειμένου να απεικονίζεται η έξοδος από το ολοκληρωμένο κύκλωμα PCA9555 στα 4 LED της θύρας D. Επιπλέον, ρυθμίσαμε με τα CONFIGURATIONS τον ακροδέκτη της θύρας επέκτασης 1 IO1\_0 ως έξοδο, ενώ τους ακροδέκτες IO1\_4-IO1\_7 ως είσοδοι, ακροδέκτες οι οποίοι θα καθορίζουν την γραμμή και την στήλη στον ‘πίνακα’ keyboard (REG\_CONFIGURATION\_1). Επομένως, όταν εμείς πατάμε ένα από τα τέσσερα πλήκτρα της τελευταίας γραμμής του πληκτρολογίου, αρχίζει η επικοινωνία με τον I2C δίαυλο και αντιλαμβάνεται το σύστημα για παράδειγμα τον δυαδικό αριθμό 0b11101110 όταν πατάμε το κουμπί ‘\*’. Αυτό έχουμε χρησιμοποιήσει στην main() συνάρτηση, τέσσερεις ελέγχους δηλαδή, έναν για κάθε δυνατό πλήκτρο που πατάμε στην τελευταία γραμμή. Έχουμε υποθέσει ότι μόνο ένα πλήκτρο είναι εφικτό να πατηθεί, και κανένα συνδυαστικά με κάποιο άλλο. Για την υλοποίηση του προγράμματός μας σε γλώσσα C, κάναμε χρήση των συναρτήσεων για την TWI και το ολοκληρωμένο PCA9555, και φτιάξαμε την ακόλουθη int main().

Αυτό που κάνουμε πρακτικά είναι να αποθηκεύουμε το input που πατήθηκε και με βάση αυτό να στέλνουμε μέσω του διαύλου I2C την πληροφορία που θέλουμε να εμφανιστεί στην έξοδο, δηλαδή τον 8-bit (1 byte) δεκαεξαδικό αριθμό που καθορίζει το LED που θέλουμε να ανάψει. Αυτό το πετυχαίνουμε φορτώνοντας τον καταχωρητή TWDR0 με τον 8-bit αριθμό που θέλουμε κάθε φορά (γίνεται στην συνάρτηση unsigned char twi\_write(unsigned char data)) και ενεργοποιούμε την σημαία διακοπής TWINT και TWEN (Enable bit) προκειμένου να εκκινήσει μια διαδικασία του υλικού να μεταφέρει την πληροφορία που θέλουμε για εγγραφή στην γραμμή SDA. Μόλις γίνει αυτή η μεταφορά δεδομένων γίνεται set ξανά το TWINT υποδηλώνοντας ότι η μετάδοση έχει ολοκληρωθεί και η γραμμή (bus) είναι ελεύθερη για χρήση.

Έχουμε χρησιμοποιήσει την γραμμή IO1\_0 ως γείωση (γράφοντας 0 σε αυτή) και διαβάζουμε κάθε είσοδο με αντίστροφη λογική, δηλαδή κάθε πάτημα κουμπιού αντιστοιχεί σε λογικό 0.

int main(void)

{

twi\_init();

//DDRB = 0x00; // Initialize PORTB as input

PCA9555\_0\_write(REG\_CONFIGURATION\_0, 0x00); // Set EXT\_PORT 0 as output (we care about IO0\_0-IO0\_3)

PCA9555\_0\_write(REG\_CONFIGURATION\_1, 0xFE); // Set IO1\_0 as output and the rest as input (set)

while(1)

{

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

PCA9555\_0\_write(REG\_OUTPUT\_1, 0xFE);

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

if (input == 0b11101110) PCA9555\_0\_write(REG\_OUTPUT\_0, 0x01); // Pressed '\*'

else if (input == 0b11011110) PCA9555\_0\_write(REG\_OUTPUT\_0, 0x02); // Pressed '0'

else if (input == 0b10111110) PCA9555\_0\_write(REG\_OUTPUT\_0, 0x04); // Pressed '#'

else if (input == 0b01111110) PCA9555\_0\_write(REG\_OUTPUT\_0, 0x08); // Pressed 'D'

}

}

**Ζήτημα 5.3:**

Όμοια και σε αυτή την άσκηση χρησιμοποιήσαμε τις συναρτήσεις που δίνονται στην εκφώνηση. Αυτό που διαφέρει είναι η προσθήκη των συναρτήσεων για την χρήση της LCD οθόνης προκειμένου να τυπώσουμε εκεί το αποτέλεσμά μας. Συγκεκριμένα, χρησιμοποιήσαμε τις συναρτήσεις που κατασκευάσαμε στο ζήτημα 4.2 της προηγούμενης σειράς ασκήσεων με την μόνη διαφορά ότι τώρα όπου κάναμε χρήση της PORTD, εδώ χρησιμοποιούμε την PCA9555\_0\_read() και ενεργοποιούμε κάθε εντολή ή δεδομένο μέσω του PCA9555\_0\_write(). Ο κώδικάς μας σε C με την εκτύπωση των δυο ονομάτων των συμμετεχόντων (τα οποία τυπώνονται διαδοχικά με διαφορά 2 δευτερολέπτων) φαίνεται παρακάτω.

void write\_2\_nibbles(uint8\_t lcd\_data) {

uint8\_t temp;

// Send the high nibble

temp = (PCA9555\_0\_read(REG\_OUTPUT\_0) & 0x0F) | (lcd\_data & 0xF0); // Keep lower 4 bits of PIND and set high nibble of lcd\_data

PCA9555\_0\_write(REG\_OUTPUT\_0 , temp); // Output the high nibble to PORTD

PCA9555\_0\_write(REG\_OUTPUT\_0, PCA9555\_0\_read(REG\_OUTPUT\_0) | (1 << PD3)); // Enable pulse high

\_delay\_us(1); // Small delay to let the signal settle

PCA9555\_0\_write(REG\_OUTPUT\_0, PCA9555\_0\_read(REG\_OUTPUT\_0) & ~(1 << PD3)); // Enable pulse low

// Send the low nibble

lcd\_data <<= 4; // Move low nibble to high nibble position

temp = (PCA9555\_0\_read(REG\_OUTPUT\_0) & 0x0F) | (lcd\_data & 0xF0); // Keep lower 4 bits of PIND and set high nibble of new lcd\_data

PCA9555\_0\_write(REG\_OUTPUT\_0 , temp); // Output the low nibble to PORTD

PCA9555\_0\_write(REG\_OUTPUT\_0 , PCA9555\_0\_read(REG\_OUTPUT\_0) | (1 << PD3)); // Enable pulse high

\_delay\_us(1); // Small delay to let the signal settle

PCA9555\_0\_write(REG\_OUTPUT\_0, PCA9555\_0\_read(REG\_OUTPUT\_0) & ~(1 << PD3)); // Enable pulse low

}

void lcd\_data(uint8\_t data)

{

PCA9555\_0\_write(REG\_OUTPUT\_0, PCA9555\_0\_read(REG\_OUTPUT\_0) | 0x04); // LCD\_RS = 1, (PD2 = 1) -> For Data

write\_2\_nibbles(data); // Send data

\_delay\_ms(5);

return;

}

void lcd\_command(uint8\_t data)

{

PCA9555\_0\_write(REG\_OUTPUT\_0, PCA9555\_0\_read(REG\_OUTPUT\_0) & 0xFB); // LCD\_RS = 0, (PD2 = 0) -> For Instruction

write\_2\_nibbles(data); // Send data

\_delay\_ms(5);

return;

}

void lcd\_clear\_display()

{

uint8\_t clear\_disp = 0x01; // Clear display command

lcd\_command(clear\_disp);

\_delay\_ms(5); // Wait 5 msec

return;

}

void lcd\_init() {

\_delay\_ms(200);

// Send 0x30 command to set 8-bit mode (three times)

PCA9555\_0\_write(REG\_OUTPUT\_0,0x30); // Set command to switch to 8-bit mode

PCA9555\_0\_write(REG\_OUTPUT\_0, PCA9555\_0\_read(REG\_OUTPUT\_0) | (1 << PD3)); // Enable pulse

\_delay\_us(1);

PCA9555\_0\_write(REG\_OUTPUT\_0, PCA9555\_0\_read(REG\_OUTPUT\_0) & ~(1 << PD3)); // Clear enable

\_delay\_us(30); // Wait 250 Âµs

PCA9555\_0\_write(REG\_OUTPUT\_0,0x30); // Repeat command to ensure mode set

PCA9555\_0\_write(REG\_OUTPUT\_0, PCA9555\_0\_read(REG\_OUTPUT\_0) | (1 << PD3));

\_delay\_us(1);

PCA9555\_0\_write(REG\_OUTPUT\_0, PCA9555\_0\_read(REG\_OUTPUT\_0) & ~(1 << PD3));

\_delay\_us(30);

PCA9555\_0\_write(REG\_OUTPUT\_0,0x30); // Repeat once more

PCA9555\_0\_write(REG\_OUTPUT\_0, PCA9555\_0\_read(REG\_OUTPUT\_0) | (1 << PD3));

\_delay\_us(1);

PCA9555\_0\_write(REG\_OUTPUT\_0, PCA9555\_0\_read(REG\_OUTPUT\_0) & ~(1 << PD3));

\_delay\_us(30);

// Send 0x20 command to switch to 4-bit mode

PCA9555\_0\_write(REG\_OUTPUT\_0,0x20);

PCA9555\_0\_write(REG\_OUTPUT\_0, PCA9555\_0\_read(REG\_OUTPUT\_0) | (1 << PD3));

\_delay\_us(1);

PCA9555\_0\_write(REG\_OUTPUT\_0, PCA9555\_0\_read(REG\_OUTPUT\_0) & ~(1 << PD3));

\_delay\_us(30);

// Set 4-bit mode, 2 lines, 5x8 dots

lcd\_command(0x28);

// Display ON, Cursor OFF

lcd\_command(0x0C);

// Clear display

lcd\_clear\_display();

// Entry mode: Increment cursor, no display shift

lcd\_command(0x06);

}

int main(){

DDRB = 0xff;

DDRC = 0xff;

DDRD = 0xff;

twi\_init();

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

lcd\_init();

while (1){

lcd\_clear\_display();

lcd\_command(0x80);

lcd\_data('N');

lcd\_data('I');

lcd\_data('K');

lcd\_data('O');

lcd\_data('L');

lcd\_data('A');

lcd\_data('O');

lcd\_data('S');

lcd\_command(0xC0);

lcd\_data('A');

lcd\_data('N');

lcd\_data('A');

lcd\_data('G');

lcd\_data('N');

lcd\_data('O');

lcd\_data('S');

lcd\_data('T');

lcd\_data('O');

lcd\_data('U');

\_delay\_ms(2000);

lcd\_clear\_display();

lcd\_command(0x80);

lcd\_data('N');

lcd\_data('I');

lcd\_data('K');

lcd\_data('O');

lcd\_data('L');

lcd\_data('A');

lcd\_data('O');

lcd\_data('S');

lcd\_command(0xC0);

lcd\_data('L');

lcd\_data('A');

lcd\_data('P');

lcd\_data('P');

lcd\_data('A');

lcd\_data('S');

\_delay\_ms(2000);

}

}