ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2024-2025

ΑΘΗΝΑ 1 Νοεμβρίου 2024

5η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"

ΟΜΑΔΑ 23

Συνεργάτες

Νικόλαος Αναγνώστου
03121818

Νικόλαος Λάππας
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.Να σημειωθεί πως ο έλεγχος για την λογική και τον υπολογισμό του F0 και F1 έγινε με 4 έξυπνα ifs τα οποία θα εξηγηθούν αναλυτικά στην εξέταση. 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().Δηλαδή η μόνη διαφορά είναι ότι τώρα μπορούμε να μιλάμε με την lcd μόνο μέσω της port 0 του PCA9555 και συγκεκριμένα μέσω του IO0_2 – IO0_7 . Ο κώδικάς μας σε C με την εκτύπωση των δυο ονομάτων των συμμετεχόντων (τα οποία τυπώνονται διαδοχικά με διαφορά 2 δευτερολέπτων) φαίνεται παρακάτω. Να σημειωθεί πως επιλέξαμε να εκτυπώνουμε στην 1η γραμμή το όνομα και στη 2η γραμμή το επίθετο.

```
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);

  }
}
```