5η Εργαστηριακή Αναφορά

Ομάδα 20

Παπαδόπουλος Χαράλαμπος 03120199

Στρίφτης Γεώργιος 03121200

Άσκηση 1

Στόχος της άσκησης είναι η εξοικείωση με το Two-Wire-Interface (TWI) της πλακέτας για σειριακή επικοινωνία. Το TWI επιτρέπει τη διασύνδεση πολλών συσκευών με τον μικροελεγκτή χρησιμοποιώντας μόνο δύο γραμμές (SCL για το ρολόι και SDA για τα δεδομένα), γεγονός που καθιστά την επικοινωνία πιο αποδοτική και απλή. Όλες οι συσκευές συνδέονται παράλληλα και κάθε συσκευή έχει μοναδική διεύθυνση, επιτρέποντας στον μικροελεγκτή να επικοινωνεί συγκεκριμένα με κάθε μία.

Αναλυτικά, για την άσκηση αυτή, οι λογικές συναρτήσεις F0 και F1 που ζητήθηκαν υλοποιήθηκαν με τους εξής τρόπους:

- **F0**: Η συνάρτηση υπολογίστηκε ως το αποτέλεσμα της έκφρασης F0=(A'BC + B'D)' όπου τα A, B, C και D αντιπροσωπεύουν τα πρώτα τέσσερα bits του PORTB. Το bit που λαμβάνεται για το αποτέλεσμα F0 εξάγεται στον ακροδέκτη IO0_0 του ολοκληρωμένου επέκτασης θυρών PCA9555.
- **F1**: Ο υπολογισμός της συναρτήσεως F1=(A+B+C)*(B*D') έγινε με παρόμοιο τρόπο, με την έξοδο να εξάγεται στον ακροδέκτη IO0_1 του PCA9555, συνδεδεμένο στο PD3 της πλακέτας για την απεικόνιση του αποτελέσματος μέσω LED.

Για την υλοποίηση αυτών των συναρτήσεων, ο μικροελεγκτής ATmega328PB χρησιμοποιεί τις εντολές του πρωτοκόλλου TWI για να επικοινωνήσει με το PCA9555 και να ελέγξει την κατάσταση των εισόδων και εξόδων.

Η χρήση των pull-up αντιστάσεων στις γραμμές του διαύλου SDA και SCL είναι κρίσιμη, ώστε οι γραμμές να βρίσκονται σε σταθερό λογικό επίπεδο όταν δεν μεταφέρονται δεδομένα, αποφεύγοντας τα σφάλματα επικοινωνίας. Οι αντιστάσεις πρόσδεσης των 10 kΩ που απαιτούνται συνδέονται μέσω των βραχυκυκλωτήρων J12 και J13 της πλακέτας.

Παραλείπεται ο κώδικας αρχικοποίησης καθώς είναι ο ίδιος με το documentation που δόθηκε.

```
void setup()
{
    DDRB = 0b11110000;                    //PORTB input
    PORTB = 0b00001111;                 // enable pull-up resistors
    DDRD = 0xFF;
    PORTD = 0xFF;
}

int main(void) {
    setup();
    uint8_t input;
    uint8_t A, B, C, D, F0, F1;
```

```
    twi_init();
    PCA9555_0_write(REG_CONFIGURATION_0, 0x00); //Set EXT_PORT0 as output
    while(1)
    {
        input = ~(PINB);
        input &= 0b00001111;
        A = (input & 0b00000001);
        B = (input & 0b00000010) >> 1;
        C = (input & 0b00000100) >> 2;
        D = (input & 0b00001000) >> 3;
        F0 = (~((~A&B&C) | ((~B)&D))) & 1;
        F1 = ((A|B|C) & (B&(~D))) << 1;
        _delay_ms(1);
        F0 |= F1;

        PCA9555_0_write(REG_OUTPUT_0, F0);
    }

}
```

Άσκηση 2

Σε αυτήν την άσκηση χρησιμοποιήθηκε το Port Expander PCA9555 για την ανάγνωση των δεδομένων εισόδου και την απεικόνιση τους μέσω LED.

Ο κώδικας θέτει την IO0_0 έως IO0_3 ως εξόδους, συνδέοντάς τες με τα PD2 έως PD5, έτσι ώστε η λογική κατάσταση να απεικονίζεται οπτικά. Κάθε φορά που ο μικροελεγκτής λαμβάνει ένα συγκεκριμένο σήμα από το PCA9555, ενεργοποιεί το αντίστοιχο LED, ανάλογα με την κατάσταση του πλήκτρου που έχει πιεστεί.

Ο κώδικας χειρίζεται την είσοδο μέσω των καταχωρητών REG_INPUT_0 και REG_INPUT_1 του PCA9555, ενώ η ρύθμιση των εξόδων γίνεται με τους καταχωρητές REG_OUTPUT_0 και REG_OUTPUT_1. Με αυτόν τον τρόπο, ο μικροελεγκτής ATmega328PB μπορεί να ελέγξει την κατάσταση του διαύλου TWI και να ελέγξει τις συσκευές μέσω του διαύλου, εξασφαλίζοντας αξιόπιστη διαχείριση των δεδομένων εισόδου/εξόδου.

```
int main(void) {
        setup();
        twi_init();
        uint8_t input;
        PCA9555_0_write(REG_CONFIGURATION_0, 0x00); //Set EXT_PORT0 as
output
        PCA9555_0_write(REG_CONFIGURATION_1, 0xF0);       // bit_0 input,
rest output
        PCA9555_0_write(REG_OUTPUT_1, 0x0);              // check last
line
        while(1)
        {
```

```
                input = PCA9555_0_read(REG_INPUT_1);
                input &= 0xF0;
                input = ~(input >> 4);
                PCA9555_0_write(REG_OUTPUT_0, input);
        }

}
```

Άσκηση 3

Η άσκηση αυτή προϋπέθετε τη σύνδεση και τη διαχείριση μιας οθόνης LCD μέσω του TWI, χρησιμοποιώντας το PCA9555. Η αρχικοποίηση της LCD έγινε με τη λειτουργία lcd_init, η οποία χρησιμοποιεί τη συνάρτηση write_2_nibbles για να γράψει τα δεδομένα σε 2 nibbles (4-bit κάθε φορά), επιτρέποντας την απεικόνιση των χαρακτήρων στην οθόνη με τη μέθοδο σειριακής επικοινωνίας.

Για την αποστολή εντολών και δεδομένων στην οθόνη χρησιμοποιήθηκαν οι συναρτήσεις lcd_command και lcd_data, αντίστοιχα. Οι συναρτήσεις αυτές συνδυάζουν τα δεδομένα και τις εντολές και τα μεταφέρουν με τη χρήση του πρωτοκόλλου I2C στον δίαυλο, ώστε η οθόνη να προβάλλει το όνομα και το επώνυμο όπως ζητείται.

Η επικοινωνία με την LCD απαιτεί τη σωστή διαχείριση του παλμού ενεργοποίησης (Enable Pulse), το οποίο εξασφαλίζει ότι η συσκευή λαμβάνει σωστά τις εντολές.

```
void lcd_command(uint8_t cmd) {
    uint8_t portd = PCA9555_0_read(REG_OUTPUT_0);
    portd &= 0b11111011;      //PORTD &= ~(1 << PD2)

    PCA9555_0_write(REG_OUTPUT_0, portd);


    write_2_nibbles(cmd);
    _delay_us(50);          // Short delay for LCD
}

void lcd_data(char data) {
    uint8_t portd = PCA9555_0_read(REG_OUTPUT_0);
    portd |= 0b00000100;


    PCA9555_0_write(REG_OUTPUT_0, portd);

    write_2_nibbles(data);
    _delay_us(50);
}

void write_2_nibbles(uint8_t data) {
    uint8_t portd = PCA9555_0_read(REG_OUTPUT_0); //read portd
    uint8_t low_bits = portd & 0x0F;    // Mask PORTD lower bits


    portd = (data & 0xF0) | low_bits;
    PCA9555_0_write(REG_OUTPUT_0, portd);


    ///enable pulse
    portd = PCA9555_0_read(REG_OUTPUT_0); //read portd
    portd |= 0b00001000;                    // Enable pulse
```

```
    PCA9555_0_write(REG_OUTPUT_0, portd);
    _delay_us(1);
    portd = PCA9555_0_read(REG_OUTPUT_0); //read portd
    portd &= 0b11110111;

    PCA9555_0_write(REG_OUTPUT_0, portd);

    _delay_us(50);                        // Delay between nibbles

    // Send low nibble

    portd = ((data << 4) & 0xF0) | low_bits;
    PCA9555_0_write(REG_OUTPUT_0, portd);


    portd = PCA9555_0_read(REG_OUTPUT_0); //read portd
    portd |= 0b00001000;                    // Enable pulse

    PCA9555_0_write(REG_OUTPUT_0, portd);
    _delay_us(1);

    portd = PCA9555_0_read(REG_OUTPUT_0); //read portd
    portd &= 0b11110111;
    PCA9555_0_write(REG_OUTPUT_0, portd);
}


void setup() {
    twi_init();
    _delay_ms(100);
    PCA9555_0_write(REG_CONFIGURATION_0, 0x00);
    lcd_init();
    _delay_ms(100);       // Delay 100 mS
    lcd_clear_display();
}


int main(void) {
        setup();

    while(1) {
        char name[] = "GIORGOS STRIFTIS";
        for(int i = 0; i < 16; i++) {
            lcd_data(name[i]);
        }
        _delay_ms(1000);
        lcd_clear_display();

    }
}
```