Ε.Μ.Π. - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2023-20234

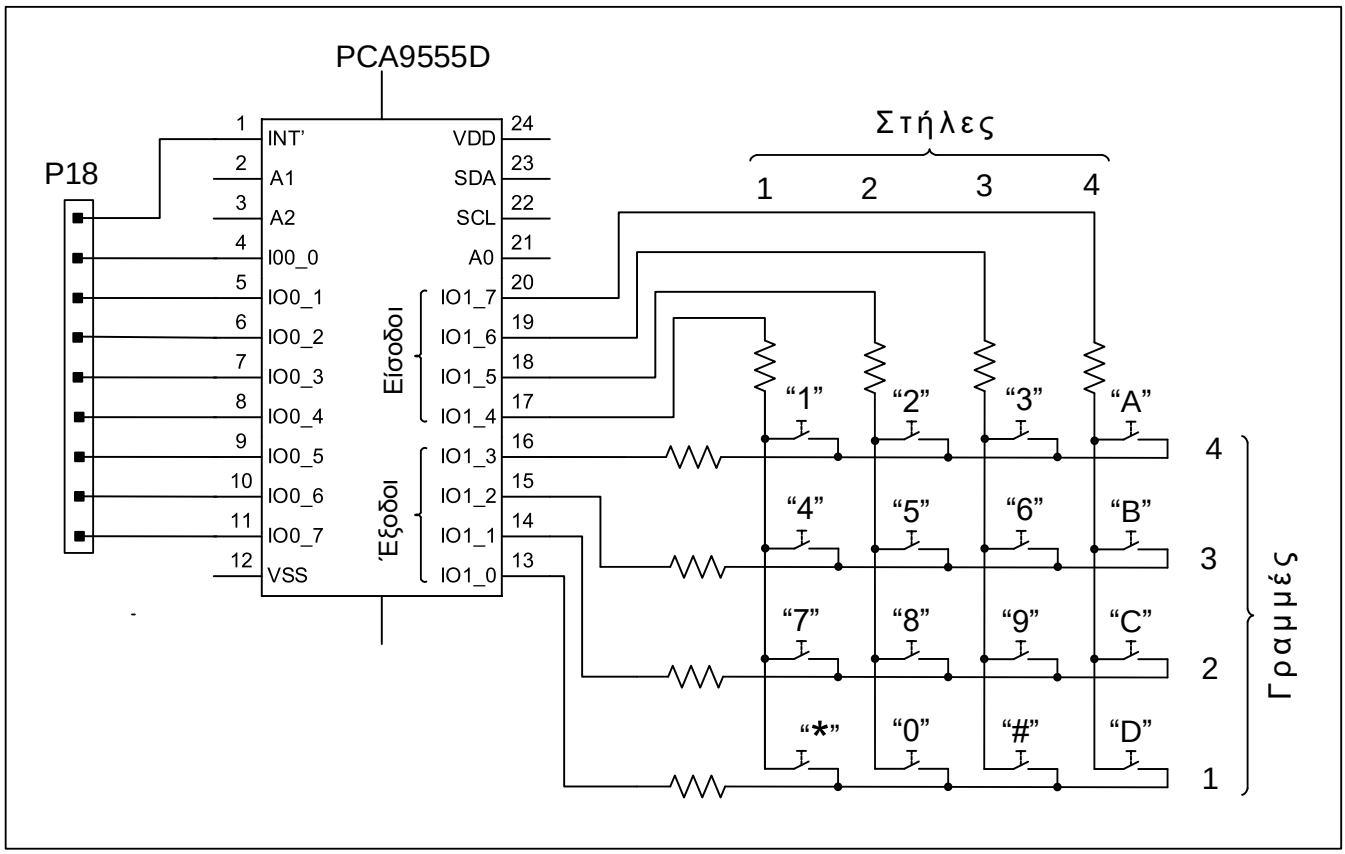
ΑΘΗΝΑ, 10 Νοεμβρίου 2023

6η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ
ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"
Χρήση πληκτρολογίου 4×4 σε θύρα επέκτασης στον AVR

Εξέταση – Επίδειξη: Παρασκευή 24/11/20223
Προθεσμία για παράδοση Έκθεσης: Τρίτη 28/11/2023 (23:59)

Πληκτρολόγιο 4×4

Το πληκτρολόγιο 4x4 της κάρτας ntuAboard έχει τέσσερις γραμμές και τέσσερις στήλες δηλαδή συνολικά 16 πλήκτρα. Όταν πατηθεί κάποιο πλήκτρο ενώνονται η γραμμή και στήλη που αντιστοιχούν σε αυτό το πλήκτρο. Η ακριβής διάταξη του πληκτρολογίου φαίνεται στο παρακάτω σχήμα:



Σχήμα 6.1 Σύνδεση του πληκτρολογίου 4×4 στην εξωτερική θύρα IO1 του PCA9555

Το πληκτρολόγιο είναι συνδεδεμένο στην εξωτερική θύρα εισόδου/εξόδου γενικής χρήσης IO1 του PCA9555, το οποίο επικοινωνεί με τον μικροελεγκτή ATmega328PB δια μέσω της διεπαφής TWI0 και η διεύθυνση του είναι 0b0100000. Οι δυο γραμμές του διαύλου TWI0 συνδέονται στους ακροδέκτες PC4 (SDA) και PC5 (SCL) του ATmega328PB. Οι αντιστάσεις πρόσδεσης (pull-up resistors) συνδέονται στο δίαυλο με την τοποθέτηση βραχυκυκλωτήρων στους κονέκτορες J12 και J13. Όταν ο δίαυλος TWI είναι ενεργός τότε οι ακροδέκτες PC4(SDA) και PC5(SCL) δεν μπορούν να χρησιμοποιηθούν για άλλο σκοπό. Όταν ένας ακροδέκτης, οποιασδήποτε από τις δύο θύρες του PCA9555, ρυθμιστεί ως είσοδος τότε ο ακροδέκτης αυτός προσδένεται στην τάση VDD μέσω μιας αντίσταση υψηλής τιμής(pull-up resistor).

Ανάγνωση του πληκτρολογίου

Μία μέθοδος ανάγνωσης του πληκτρολογίου είναι η εξής:

Διαδοχικά κάθε μία από τις γραμμές του πληκτρολογίου, οι οποίες έχουν ρυθμιστεί ως έξοδοι και είναι συνδεδεμένες με τους ακροδέκτες IO1[3:0] της θύρας IO1 του PCA9555, τίθεται σε λογικό 0 (χαμηλή στάθμη τάσης) ενώ ταυτόχρονα οι υπόλοιπες γραμμές έχουν τεθεί σε λογικό 1. Για κάθε μία από τις γραμμές, που έχει τεθεί σε λογικό 0, εκτελείται μία μικρή χρονοκαθυστέρηση και μετά διαβάζονται οι ακροδέκτες IO1[7:4] οι οποίοι αντιστοιχούν στις στήλες του πληκτρολογίου και έχουν ρυθμιστεί ως είσοδοι. Αν δεν υπάρχει πιεσμένος διακόπτης, λόγω των pull-up αντιστάσεων, οι ακροδέκτες εισόδου βρίσκονται σε κατάσταση λογικού 1. Τα πλήκτρα που είναι πατημένα κάθε φορά μπορούν να εντοπιστούν γνωρίζοντας πια γραμμή έχει τεθεί σε λογικό 0 και διαβάζοντας ταυτόχρονα ποιες από τις στήλες βρίσκονται σε λογικό 0.

Όταν ο χρήστης πατάει ένα πλήκτρο, αυτό μπορεί να μείνει πιεσμένο για μεγάλο χρονικό διάστημα. Για το λόγο αυτό απαιτείται ιδιαίτερος χειρισμός για να εντοπιστούν τα πλήκτρα που έχουν πατηθεί ως ολοκληρωμένη ενέργεια και όχι απλά αυτά που είναι κάθε φορά πατημένα. Αυτό επιτυγχάνεται εάν κληθεί μία ρουτίνα που διαβάζει την κατάσταση των πλήκτρων και αποθηκεύει το αποτέλεσμα στη μνήμη του μικροελεγκτή. Στη συνέχεια καλείτε ξανά η ίδια ρουτίνα, για νέο έλεγχο της κατάστασης των πλήκτρων. Μια σύγκριση των δύο καταστάσεων του πληκτρολογίου θα αποκαλύψει τις διαφορές στην κατάσταση των διακοπτών. Το χρονικό διάστημα ανάμεσα στις διαδοχικές κλήσεις της συνάρτησης διαβάσματος του πληκτρολογίου είναι κρίσιμο, διότι καθορίζει το χρόνο που θα πρέπει να μείνει πιεσμένος ένας διακόπτης για να καταγραφεί από τον μικροελεγκτή. Αυτό σημαίνει ότι ένα μεγάλο χρονικό διάστημα μεταξύ των διαδοχικών κλήσεων της ρουτίνας θα αναγκάσει τον χρήστη να κρατά πατημένο το πλήκτρο για αντίστοιχα μεγάλο χρονικό διάστημα (ώστε να μπορεί να αναγνωριστεί). Αντίθετα, ένα πολύ μικρό χρονικό διάστημα θα δημιουργήσει προβλήματα λόγω του σπινθηρισμού που παρουσιάζουν οι διακόπτες. Οι τυπικές τιμές καθυστέρησης είναι της τάξης των 10 έως 20 msec.

Ο παρακάτω κώδικας μπορεί να χρησιμοποιηθεί για να γίνει εγγραφή ή ανάγνωση ενός από τους καταχωρητές ελέγχου του ολοκληρωμένου PCA9555:

```
#define F_CPU 16000000UL

#include<avr/io.h>
#include<avr/interrupt.h>
#include<util/delay.h>


#define PCA9555_0_ADDRESS 0x40          //A0=A1=A2=0 by hardware
#define TWI_READ    1                   // reading from twi device
#define TWI_WRITE   0                   // writing to twi device
#define SCL_CLOCK  100000L              // twi clock in Hz

//Fscl=Fcpu/(16+2*TWBR0_VALUE*PRESCALER_VALUE)
#define TWBR0_VALUE ((F_CPU/SCL_CLOCK)-16)/2

// PCA9555 REGISTERS
typedef enum {
  REG_INPUT_0              = 0,
  REG_INPUT_1              = 1,
  REG_OUTPUT_0             = 2,
  REG_OUTPUT_1             = 3,
  REG_POLARITY_INV_0       = 4,
  REG_POLARITY_INV_1       = 5,
  REG_CONFIGURATION_0      = 6,
  REG_CONFIGURATION_1      = 7
} PCA9555_REGISTERS;

//----------- Master Transmitter/Receiver --------------------
#define TW_START                    0x08
#define TW_REP_START                0x10

//---------------- Master Transmitter ----------------------
#define TW_MT_SLA_ACK               0x18
#define TW_MT_SLA_NACK              0x20
#define TW_MT_DATA_ACK              0x28

//---------------- Master Receiver ----------------
#define TW_MR_SLA_ACK               0x40
#define TW_MR_SLA_NACK              0x48
#define TW_MR_DATA_NACK             0x58


#define TW_STATUS_MASK  0b11111000
#define TW_STATUS   (TWSR0 & TW_STATUS_MASK)

//initialize TWI clock
```

```
void twi_init(void)
{
  TWSR0 = 0;                         // PRESCALER_VALUE=1
 TWBR0 = TWBR0_VALUE;  // SCL_CLOCK  100KHz
}

// Read one byte from the twi device (request more data from device)
unsigned char twi_readAck(void)
{
        TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);
        while(!(TWCR0 & (1<<TWINT)));
        return TWDR0;
}

//Read one byte from the twi device, read is followed by a stop condition
unsigned char twi_readNak(void)
{
        TWCR0 = (1<<TWINT) | (1<<TWEN);
        while(!(TWCR0 & (1<<TWINT)));
        return TWDR0;
}

// Issues a start condition and sends address and transfer direction.
// return 0 = device accessible, 1= failed to access device
unsigned char twi_start(unsigned char address)
{
   uint8_t   twi_status;

        // send START condition
        TWCR0 = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);

        // wait until transmission completed
        while(!(TWCR0 & (1<<TWINT)));

        // check value of TWI Status Register.
        twi_status = TW_STATUS & 0xF8;
        if ( (twi_status != TW_START) && (twi_status != TW_REP_START)) return 1;

        // send device address
        TWDR0 = address;
        TWCR0 = (1<<TWINT) | (1<<TWEN);

        // wail until transmission completed and ACK/NACK has been received
        while(!(TWCR0 & (1<<TWINT)));

        // check value of TWI Status Register.
        twi_status = TW_STATUS & 0xF8;
        if ( (twi_status != TW_MT_SLA_ACK) && (twi_status != TW_MR_SLA_ACK) )
   {
      return 1;
   }
```

```
        return 0;
}

// Send start condition, address, transfer direction.
// Use ack polling to wait until device is ready
void twi_start_wait(unsigned char address)
{
   uint8_t   twi_status;

   while ( 1 )
   {
        // send START condition
        TWCR0 = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);

        // wait until transmission completed
        while(!(TWCR0 & (1<<TWINT)));

        // check value of TWI Status Register.
        twi_status = TW_STATUS & 0xF8;
        if ( (twi_status != TW_START) && (twi_status != TW_REP_START)) continue;

        // send device address
        TWDR0 = address;
        TWCR0 = (1<<TWINT) | (1<<TWEN);

        // wail until transmission completed
        while(!(TWCR0 & (1<<TWINT)));

        // check value of TWI Status Register.
        twi_status = TW_STATUS & 0xF8;
        if ( (twi_status == TW_MT_SLA_NACK )||(twi_status ==TW_MR_DATA_NACK) )
        {
           /* device busy, send stop condition to terminate write operation */
              TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);

              // wait until stop condition is executed and bus released
              while(TWCR0 & (1<<TWSTO));

           continue;
        }
        break;
    }
}


//  Send one byte to twi device,  Return 0 if write successful or 1 if write failed
unsigned char twi_write( unsigned char data )
{
        // send data to the previously addressed device
        TWDR0 = data;
```

```
        TWCR0 = (1<<TWINT) | (1<<TWEN);

        // wait until transmission completed
        while(!(TWCR0 & (1<<TWINT)));
        if( (TW_STATUS & 0xF8) != TW_MT_DATA_ACK) return 1;
        return 0;
}

// Send repeated start condition, address, transfer direction
//Return:  0 device accessible
//         1 failed to access device
unsigned char twi_rep_start(unsigned char address)
{
   return twi_start( address );
}


// Terminates the data transfer and releases the twi bus
void twi_stop(void)
{
        // send stop condition
        TWCR0 = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);

        // wait until stop condition is executed and bus released
        while(TWCR0 & (1<<TWSTO));
}


void PCA9555_0_write(PCA9555_REGISTERS reg, uint8_t value)
{
    twi_start_wait(PCA9555_0_ADDRESS + TWI_WRITE);
    twi_write(reg);
    twi_write(value);
    twi_stop();
}

uint8_t PCA9555_0_read(PCA9555_REGISTERS reg)
{
uint8_t ret_val;

    twi_start_wait(PCA9555_0_ADDRESS + TWI_WRITE);
    twi_write(reg);
    twi_rep_start(PCA9555_0_ADDRESS + TWI_READ);
    ret_val = twi_readNak();
    twi_stop();

    return ret_val;
}
```

Τα ζητούμενα της 6ης εργαστηριακής άσκησης

Ζήτημα 6.1

Να υλοποιηθεί κώδικας για το μικροελεγκτή ATmega328PB, σε γλώσσα C, ο οποίος να διαβάζει το πληκτρολόγιο της ntuAboard_G1, και να έχει τη δομή που περιγράφεται ακολούθως:

α) Να δημιουργηθεί μια συνάρτηση με όνομα scan_row η οποία να ελέγχει μια γραμμή του πληκτρολογίου για πιεσμένους διακόπτες.

β) Να δημιουργηθεί μια συνάρτηση με όνομα scan_keypad η οποία θα καλεί διαδοχικά την scan_row 4 φορές και θα ελέγχει ολόκληρο το πληκτρολόγιο.

γ) Να δημιουργηθεί μια συνάρτηση με όνομα scan_keypad_rising_edge η οποία θα καλείτε συνεχόμενα και θα κρατάει στη μνήμη του μικροελεγκτή μία εικόνα των πατημένων πλήκτρων. Η συνάρτηση scan_keypad_rising_edge θα καλεί τη συνάρτηση scan_keypad και θα αποθηκεύει τα πλήκτρα που είναι πατημένα σε μια 16μπιτη μεταβλητή. Προκειμένου να αντιμετωπιστεί το πρόβλημα του σπινθηρισμού, που παρουσιάζουν οι διακόπτες, θα καλεί ξανά την scan_keypad, μετά από χρονικό διάστημα της τάξης των 10 έως 20 mSec και θα συγκρίνει την τρέχουσα εικόνα των πατημένων πλήκτρων με αυτή που είναι αποθηκευμένη στη 16μπιτη μεταβλητή. Όσα πλήκτρα διαφέρουν θα απορρίπτονται και η τελική εικόνα των πατημένων πλήκτρων θα συγκρίνεται με την εικόνα που είναι αποθηκευμένη στη μνήμη του μικροελεγκτή από προηγούμενη κλίση έτσι ώστε να εντοπίσει ποια πλήκτρα έχουν πατηθεί μόλις τώρα, απορρίπτοντας τα πλήκτρα που ήταν πατημένα και στην προηγούμενη κλήση της scan_keypad_rising_edge. Τέλος η τωρινή εικόνα των πατημένων πλήκτρων θα αντικαθιστά την προηγούμενη εικόνα στη μνήμη του μικροελεγκτή για χρήση στην επόμενη κλήση.

δ) Να δημιουργηθεί μια συνάρτηση με όνομα *keypad_to_ascii* που εντοπίζει τον διακόπτη που έχει πατηθεί και επιστρέφει τον κωδικό ascii του χαρακτήρα που αντιστοιχεί στον διακόπτη. Αν δεν είναι πιεσμένος κανένας διακόπτης επιστρέφει την τιμή 0. Θεωρείστε δεδομένο ότι κάθε φορά μπορεί να είναι πατημένο μόνο ένα πλήκτρο.

ε) Να αντιστοιχήσετε τέσσερα από τα πλήκτρα του πληκτρολογίου σε τέσσερα led σύμφωνα με τον παρακάτω πίνακα:

“1”	“5”	“9”	“D”
Led στο PB0	Led στο PB1	Led στο PB2	Led στο PB3

Κάθε φορά που θα πιέζεται κάποιο από τα παραπάνω τέσσερα πλήκτρα να ανάβει το αντίστοιχο led. Το led θα παραμένει αναμμένο όσο διάστημα το πλήκτρο είναι πατημένο.

Θεωρείστε δεδομένο ότι κάθε φορά μπορεί να είναι πατημένο μόνο ένα πλήκτρο.

Ζήτημα 6.2

Να υλοποιηθεί κώδικας για το μικροελεγκτή ATmega328PB, σε γλώσσα C, ο οποίος θα κάνει χρήση των συναρτήσεων του ζητήματος 6.1 και θα απεικονίζει στην οθόνη LCD 2x16 το χαρακτήρα που αντιστοιχεί στο πλήκτρο που πατήθηκε τελευταίο. Θεωρείστε δεδομένο ότι κάθε φορά μπορεί να πατηθεί μόνο ένα πλήκτρο.

Ζήτημα 6.3

Γράψτε ένα πρόγραμμα «ηλεκτρονικής κλειδαριάς» το οποίο να ανάβει τα 6 leds PB0 έως PB5 για 4 sec, όταν δοθεί από το keypad 4×4 ο διψήφιος αριθμός της ομάδας σας (π.χ. 09). Εάν κάποιος σπουδαστής δεν έχει αριθμό ομάδας να κάνει χρήση του αριθμού 99. Αν δεν έχουν δοθεί σωστά τα δύο ψηφία του αριθμού της ομάδας τότε τα leds PB0 έως PB5 να αναβοσβήνουν(250 mSec ανάμενα, 250 mSec σβηστά) για 5 sec. Ανεξάρτητα από το χρονικό διάστημα για το οποίο θα μείνει πατημένο ένα πλήκτρο, το πρόγραμμά θα πρέπει να θεωρεί ότι πατήθηκε μόνο μια φορά. Μετά το πάτημα δύο αριθμών το πρόγραμμα να μην δέχεται για 5 sec άλλον αριθμό. Το πρόγραμμα να είναι συνεχόμενης λειτουργίας. Δώστε το διάγραμμα ροής και το πρόγραμμα σε γλώσσα C.