מערכות משובצות מחשב – סיכום

Configurations:

```
#define LCD_SCREEN_SIZE 16
#define LCD_FIRST_LINE 0x80
#define LCD_SECOND_LINE 0xc0

#define KEYBOARD_LENGTH 16
```

```
char keyboard_values[KEYBOARD_LENGTH * 2] = {0x44, '1', 0x34, '2', 0x24, '3', 0x43, '4',
0x33, '5', 0x23, '6', 0x42, '7', 0x32, '8',0x22, '9', 0x41, '0', 0x14, 'A', 0x13, 'B',
0x12, 'C', 0x11, 'D', 0x21, 'E', 0x31, 'F'};
```

```
// LED configurations
TRISA &= 0xff00;
PORTA = 0x0; // lights off at start
// audio configurations
TRISBbits.TRISB14 = OFF;
ANSELBbits.ANSB14 = OFF;
// LCD configurations
char control[] = {0x38, 0x38, 0x38, 0xe, 0x6, 0x1, 0x80};
TRISE &= 0xff00;
TRISDbits.TRISD4 = 0; // RD4 (DISP_EN) set as an output
TRISDbits.TRISD5 = 0; // RD5 (DISP_RW) set as an output
TRISBbits.TRISB15 = 0; // RB15 (DISP_RS) set as an output
PORTDbits.RD5 = 0; //w=0
PORTBbits.RB15 = 0; //rs=0
ANSELEbits.ANSE2 = 0;
ANSELEbits.ANSE4 = 0;
ANSELEbits.ANSE5 = 0;
ANSELEbits.ANSE6 = 0;
ANSELEbits.ANSE7 = 0;
ANSELBbits.ANSB15 = 0;
lcd_init(control, 7);
```

```
// switch configurations
TRISFbits.TRISF3 = ON; // RF3 (SW0) configured as input
TRISFbits.TRISF5 = ON; // RF5 (SW1) configured as input
TRISFbits.TRISF4 = ON; // RF4 (SW2) configured as input
TRISDbits.TRISD15 = ON; // RD15 (SW3) configured as input
TRISDbits.TRISD14 = ON; // RD14 (SW4) configured as input
TRISBbits.TRISB11 = ON; // RB11 (SW5) configured as input
TRISBbits.TRISB10 = ON; // RB10 (SW6) configured as input
TRISBbits.TRISB9 = ON; // RB9 (SW7) configured as input
ANSELBbits.ANSB11 = OFF; // RB11 (SW5) disabled analog
ANSELBbits.ANSB10 = OFF; // RB10 (SW6) disabled analog
ANSELBbits.ANSB9 = OFF; // RB9 (SW7) disabled analog
```

```
// keyboard configurations
TRISCbits.TRISC2 = OFF; //RC2
TRISCbits.TRISC1 = OFF; //RC1
TRISCbits.TRISC4 = OFF; //RC4
TRISGbits.TRISG6 = OFF; //RG6
TRISCbits.TRISC3 = _ON; //RC3
TRISGbits.TRISG7 = _ON; //RG7
TRISGbits.TRISG8 = _ON; //RG8
TRISGbits.TRISG9 = _ON; //RG9
ANSELGbits.ANSG6 = OFF;
ANSELGbits.ANSG7 = OFF;
ANSELGbits.ANSG8 = OFF;
ANSELGbits.ANSG9 = OFF;
CNPUCbits.CNPUC3;
CNPUGbits.CNPUG7;
CNPUGbits.CNPUG8;
CNPUGbits.CNPUG9;
```

```
// RGB configurations
TRISDbits.TRISD2 = OFF;  // R LED8_R AN25/RPD2/RD2 output RED
TRISDbits.TRISD12 = OFF; // G LED8_G RPD12/PMD12/RD12 output GREEN
TRISDbits.TRISD3 = OFF;  // B LED8_B AN26/RPD3/RD3 output BLUE
```

7Segment:

```
#include "ssd.h"

// 7Segment configuration
SSD_Init();
SSD_WriteDigits(0, 0, 0, 0, 0, 0, 0, 0);

// Button configuration
TRISBbits.TRISB1 = _ON; // RB1 (BTNU) configured as input
TRISBbits.TRISB0 = _ON; // RB0 (BTNL) configured as input
TRISFbits.TRISF0 = _ON; // RF0 (BTNC) configured as input
TRISBbits.TRISB8 = _ON; // RB8 (BTNR) configured as input
TRISAbits.TRISA15 = _ON; // RA15 (BTND) configured as input
ANSELBbits.ANSB1 = OFF; // RB1 (BTNU) disabled analog
ANSELBbits.ANSB0 = OFF; // RB0 (BTNL) disabled analog
ANSELBbits.ANSB8 = OFF; // RB8 (BTNR) disabled analog

SSD_WriteDigits(second, tens_second, minute, tens_minute, 0, 0, 0, 0);
```

Interrupt:

```
// interrupt configuration
PR4 = 0xffff; //             set period register, generates one interrupt every 1 ms
TMR4 = 0; //             initialize count to 0
T4CONbits.TCKPS0 = 1; //            1:256 prescale value
T4CONbits.TCKPS1 = 1;
T4CONbits.TCKPS2 = 1;
T4CONbits.TGATE = 0; //             not gated input (the default)
T4CONbits.TCS = 0; //             PCBLK input (the default)
T4CONbits.ON = 1; //             turn on Timer1
IPC4bits.T4IP = 2; //             priority
IPC4bits.T4IS = 0; //             subpriority
IFS0bits.T4IF = 0; //             clear interrupt flag
IEC0bits.T4IE = 1;
INTCONbits.MVEC = 1; //vector interrupt
IPTMR = 0; //INTERRUPT PROXIMITY TIMER REGISTER
asm("ei"); //on interrupt
```

```
void __ISR(_TIMER_4_VECTOR, ipl2) Timer4SR(void) {
    key_code = check_keyboard();
    wait_to_release_keyboard();
    IFS0bits.T4IF = 0; // clean interrupt flag
}
```

Switches:

```
switches[0] = PORTFbits.RF3;
switches[1] = PORTFbits.RF5;
switches[2] = PORTFbits.RF4;
switches[3] = PORTDbits.RD15;
switches[4] = PORTDbits.RD14;
switches[5] = PORTBbits.RB11;
switches[6] = PORTBbits.RB10;
switches[7] = PORTBbits.RB9;
```

Sound (audio):

```
void mode_sound() {
    int j, i;
    for (i = 0; i < 100; i++) {
        PORTBbits.RB14 = ON;
        for (j = 0; j < 500; j++);
        PORTBbits.RB14 = OFF;
    }
}
```

LCD:

```
void lcd_init(char control[], int size) {
    int j;
    for (j = 0; j < size; j++) {
        PORTE = control[j];
        PORTDbits.RD4 = 1;
        PORTDbits.RD4 = 0;
        busy();
    }
    lcd_clear();
}

void lcd_clear() {
    PORTE = 0x01; // set instruction 0x01
    PORTBbits.RB15 = 0; // rs = 0 -> instruction
    PORTDbits.RD5 = 0; // w = 0
    PORTDbits.RD4 = 1; // enable on
    PORTDbits.RD4 = 0; // enable off
    busy();
}
```

```c
void lcd_set_text(char first_line[], char second_line[]) {
    busy();
    lcd_clear();
    lcd_set_cursor(LCD_FIRST_LINE);
    lcd_write_line(first_line);
    lcd_set_cursor(LCD_SECOND_LINE);
    lcd_write_line(second_line);
}

void lcd_write_line(char string[]) {
    int j;
    int length = strlen(string);
    PORTBbits.RB15 = 1; //rs = 1 -> data transfer
    PORTDbits.RD5 = 0; //w = 0
    for (j = 0; j < length; j++) {
        PORTE = string[j];
        PORTDbits.RD4 = 1;
        PORTDbits.RD4 = 0;
        busy();
    }
}

void lcd_set_cursor(int line) {
    PORTE = line;
    PORTBbits.RB15 = 0; // rs = 0 -> instruction
    PORTDbits.RD5 = 0; // w = 0
    PORTDbits.RD4 = 1;
    PORTDbits.RD4 = 0;
    busy();
}
```

```c
void busy(void) {
    char RD, RS;
    int STATUS_TRISE;
    int portMap;
    RD = PORTDbits.RD5;
    RS = PORTBbits.RB15;
    STATUS_TRISE = TRISE;
    PORTDbits.RD5 = 1; //w/r
    PORTBbits.RB15 = 0; //rs
    portMap = TRISE;
    portMap |= 0x80;
    TRISE = portMap;
    do {
        PORTDbits.RD4 = 1; //enable=1
        PORTDbits.RD4 = 0; //enable=0
    } while (PORTEbits.RE7); // BF ?????
    PORTDbits.RD5 = RD;
    PORTBbits.RB15 = RS;
    TRISE = STATUS_TRISE;
}
```

LCD font:

```c
#define LCD_FIRST_LINE_START 0x80
#define LCD_FIRST_LINE_END 0x85
#define LCD_SECOND_LINE_START 0xc0
#define LCD_SECOND_LINE_END 0xc5
#define LCD_LINES_DIFFERENCE 0x40

int CG_gal[16] = {0x1f, 0x1, 0x1, 0x1, 0x3, 0x5, 0x9, 0x11,
    0x30, 0x30, 0x30, 0x3f, 0x21, 0x21, 0x22, 0x22}; // gal

int str_gal[2] = {1, 0};

int CG_naor[32] = {0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x5f,
    0x09, 0x0a, 0x0c, 0x0c, 0x0c, 0x0a, 0x09, 0x09,
    0x86, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82, 0x82,
    0xbf, 0xa1, 0xa1, 0xa1, 0xa1, 0xa1, 0xa1, 0xa1}; // naor

int str_naor[4] = {5, 4, 3, 2};
```

```
void lcd_add_cgram() {
    PORTE = 0x40; // set CGRAM=0x40
    PORTDbits.RD4 = ON;
    PORTDbits.RD4 = OFF;
    busy();

    PORTBbits.RB15 = ON; //rs
    lcd_add_font(CG_gal, sizeof (CG_gal) / sizeof (int));
    lcd_add_font(CG_naor, sizeof (CG_naor) / sizeof (int));
    lcd_add_font(CG_natan, sizeof (CG_natan) / sizeof (int));
}

void lcd_add_font(int font_code[], int arr_size) {
    int i;
    for (i = 0; i < arr_size; i++) {
        PORTE = font_code[i];
        PORTDbits.RD4 = ON; //enable=1
        PORTDbits.RD4 = OFF; //enable=0
        busy();
    }
}
void lcd_print_word(int string[], int size) {
    int j;
    PORTBbits.RB15 = ON; //rs = 1 -> data transfer
    PORTDbits.RD5 = OFF; //w = 0
    for (j = 0; j < size; j++) {
        PORTE = string[j];
        PORTDbits.RD4 = ON;
        PORTDbits.RD4 = OFF;
        busy();
    }
}

void lcd_print_space() {
    PORTBbits.RB15 = ON; //rs = 1 -> data transfer
    PORTDbits.RD5 = OFF; //w = 0
    PORTE = ' ';
    PORTDbits.RD4 = ON;
    PORTDbits.RD4 = OFF;
    busy();
}
```

Keyboard:

```
char code_to_ascii(int key_code) {
    int i;
    for (i = 0; i < KEYBOARD_LENGTH * 2; i += 2) {
        if (keyboard_values[i] == key_code)
            return keyboard_values[i + 1];
    }
}

void wait_to_release_keyboard() {
    PORTCbits.RC2 = OFF;
    PORTCbits.RC1 = OFF;
    PORTCbits.RC4 = OFF;
    PORTGbits.RG6 = OFF;
    while (!PORTCbits.RC3);
    while (!PORTGbits.RG7);
    while (!PORTGbits.RG8);
    while (!PORTGbits.RG9);
}
```

```
int in_y(int a) {
    int j = 1, flag = 0;
    if (!PORTCbits.RC3) {
        flag = 1;
        j = 1;
    } else if (!PORTGbits.RG7) {
        flag = 1;
        j = 2;
    } else if (!PORTGbits.RG8) {
        flag = 1;
        j = 3;
    } else if (!PORTGbits.RG9) {
        flag = 1;
        j = 4;
    }
    if (flag == 0)
        return NONE;
    else
        return (j | (a << 4));
}
```

```c
int check_keyboard() {
    int xy;
    PORTCbits.RC2 = _ON;
    PORTCbits.RC1 = _ON;
    PORTCbits.RC4 = _ON;
    PORTGbits.RG6 = _ON;
    PORTCbits.RC2 = OFF;
    xy = in_y(1);
    if (xy != NONE)
        return xy;
    PORTCbits.RC2 = _ON;
    PORTCbits.RC1 = OFF;
    xy = in_y(2);
    if (xy != NONE)
        return xy;
    PORTCbits.RC1 = _ON;
    PORTCbits.RC4 = OFF;
    xy = in_y(3);
    if (xy != NONE)
        return xy;
    PORTCbits.RC4 = _ON;
    PORTGbits.RG6 = OFF;
    xy = in_y(4);
    PORTGbits.RG6 = _ON;
    return xy;
}
```

Interrupt example:

Main:

```
void main() {

    sys_config();
    while (1){};
}
```

Configuration:

+ RGB configuration

```
// Timer 1 configuration
PR1 = 0xc35; // set period register interrupt each 10 ms
TMR1 = 0; // initialize count to 0
T1CONbits.TGATE = OFF; // not gated input (the default)
T1CONbits.TCS = OFF; // PCBLK input (the default
T1CONbits.TCKPS0 = _ON; // 1:256 prescale value
T1CONbits.TCKPS1 = _ON;
T1CONbits.TSYNC = _ON;
T1CONbits.ON = _ON; // turn on Timer1
IPC1bits.T1IP = 2; // interrupt vector priority
IPC1bits.T1IS = 2; // interrupt vector subpriority
IEC0bits.T1IE = _ON;
IFS0bits.T1IF = OFF; // clear interrupt flag
```

```
// Timer 2-3 configuration
PR2 = 0x8968; //PR2x interrupt each 2 second
PR3 = 0x9; //PR2y
TMR2 = 0; //TMRx
TMR3 = 0; //TMRy
T2CONbits.TGATE = OFF; // not gated input (the default)
T2CONbits.TCS = OFF; // PCBLK input (the default)
T2CONbits.TCKPS0 = _ON; // 1:256 prescale value
T2CONbits.TCKPS1 = _ON;
T2CONbits.TCKPS2 = _ON;
T2CONbits.T32 = _ON; // mode 32bit
T2CONbits.ON = _ON; // turn on Timer2/3
IPC2bits.T2IP = 2; // interrupt vector priority
IPC2bits.T2IS = 1; // interrupt vector subpriority
IPC3bits.T3IP = 2; // interrupt vector priority
IPC3bits.T3IS = 1; // interrupt vector subpriority
IEC0bits.T2IE = _ON;
IEC0bits.T3IE = _ON;
IFS0bits.T3IF = OFF; // clear interrupt flag

// Timer 4 configuration
PR4 = 0x7a12; // set period register interrupt each 100 ms
TMR4 = 0; // initialize count to 0
T4CONbits.TGATE = OFF; // not gated input (the default)
T4CONbits.TCS = OFF; // PCBLK input (the default)
T4CONbits.TCKPS0 = _ON; // 1:256 prescale value
T4CONbits.TCKPS1 = _ON;
T4CONbits.TCKPS2 = _ON;
T4CONbits.ON = _ON; // turn on Timer4
IPC4bits.T4IP = 2; // interrupt vector priority
IPC4bits.T4IS = 0; // interrupt vector subpriority
IEC0bits.T4IE = _ON;
IFS0bits.T4IF = OFF; // clear interrupt flag

INTCONbits.MVEC = _ON; //vector interrupt
IPTMR = OFF; //INTERRUPT PROXIMITY TIMER REGISTER
asm("ei"); //on interrupt
```

Functions:

```
void __ISR(_TIMER_1_VECTOR, ipl2) Timer1SR(void) {
    PORTDbits.RD2 = !PORTDbits.RD2;
    IFS0bits.T1IF = OFF; // clean interrupt flag
}

void __ISR(_TIMER_3_VECTOR, ipl2) Timer3SR(void) {
    PORTDbits.RD12 = !PORTDbits.RD12;
    IFS0bits.T3IF = OFF; // clean interrupt flag
}

void __ISR(_TIMER_4_VECTOR, ipl2) Timer4SR(void) {
    PORTDbits.RD3 = !PORTDbits.RD3;
    IFS0bits.T4IF = OFF; // clean interrupt flag
}
```

Every few seconds there is an Interrupt that turns on the RGB.

Lab example:

מהלך הניסוי:

בניסוי זה נפעיל את תצוגת הכרטיס (LCD) כתלות במקש שנלחץ בלוח המקשים, נדגום את הכניסות האנלוגיות את קריאת מתח הכניסה ונוציא במוצא האנלוגי צורות גלים שונות.

אם לחיצה על מקש מסוים בלוח המקשים, ישמע צפצוף קצר ויוצג **בשורה השנייה** של התצוגה הכיתוב הבא:

- מקש 1 – יוצג הכיתוב הבא ונדגום את מתח הכניסה האנלוגית מהנגד המשתנה .

Analog 1 – <x.dd>V

- מקש 2 – יוצג הכיתוב הבא ונדגום את המתח בכניסה האנלוגית של המיקרופון

Analog 2 – <x.dd>V

Configuration:

+ LCD configureration

```
// ADC configurations
AD1CON1 = OFF;
AD1CON1bits.SSRC = 7; // Internal counter ends sampling and starts conversion (auto convert)
AD1CON1bits.FORM = 0; // Integer 16-bit
// Setup for manual sampling
AD1CSSL = 0;
AD1CON3 = 0x0002; // ADC Conversion Clock Select bits: TAD = 6 TPB
AD1CON2 = 0;
AD1CON2bits.VCFG = OFF; // Voltage Reference Configuration bits: VREFH = AVDD and VREFL = AVSS
// Turn on ADC
AD1CON1bits.ON = _ON;

// potentiometer configurations
TRISBbits.TRISB2 = _ON;
ANSELBbits.ANSB2 = _ON;

// microphone configurations
TRISBbits.TRISB4 = _ON;
ANSELBbits.ANSB4 = _ON;
```

```
// keyboard configurations
TRISCbits.TRISC2 = OFF; //RC2
TRISCbits.TRISC1 = OFF; //RC1
TRISCbits.TRISC4 = OFF; //RC4
TRISGbits.TRISG6 = OFF; //RG6
TRISCbits.TRISC3 = _ON; //RC3
TRISGbits.TRISG7 = _ON; //RG7
TRISGbits.TRISG8 = _ON; //RG8
TRISGbits.TRISG9 = _ON; //RG9
ANSELGbits.ANSG6 = OFF;
ANSELGbits.ANSG7 = OFF;
ANSELGbits.ANSG8 = OFF;
ANSELGbits.ANSG9 = OFF;
CNPUCbits.CNPUC3;
CNPUGbits.CNPUG7;
CNPUGbits.CNPUG8;
CNPUGbits.CNPUG9;
```

```c
void main() {
    char key_ascii; // current asccii value for pressed key
    int prev_key = NONE;
    int key_code = NONE; // current key code for pressed key

    double voltage_div_value = 1023 / 3.3;
    int voltage = 0;
    char message[LCD_SCREEN_SIZE];
    sys_config();

    while (1) {
        key_code = check_keyboard();
        if (key_code == prev_key)
            continue;

        key_ascii = code_to_ascii(key_code);
        switch (key_ascii) {
            case '1':
                voltage = ADC_AnalogRead(2); //IN analog RB2
                sprintf(message, "Analog 1 - %.2fV", voltage / voltage_div_value);
                lcd_set_text("Lab6:", message);
                break;
            case '2':
                voltage = ADC_AnalogRead(4); //IN analog RB4
                sprintf(message, "Analog 2 - %.2fV", voltage / voltage_div_value);
                lcd_set_text("Lab6:", message);
                break;
        }
        hold_program();
    }
```

```c
unsigned int ADC_AnalogRead(unsigned char analogPIN) {
    int adc_val = 0;

    IEC0bits.T2IE = 0;
    AD1CHS = analogPIN << 16; // AD1CHS<16:19> controls which analog pin goes to the ADC

    AD1CON1bits.SAMP = 1; // Begin sampling
    while (AD1CON1bits.SAMP); // wait until acquisition is done
    while (!AD1CON1bits.DONE); // wait until conversion is done

    adc_val = ADC1BUF0;
    IEC0bits.T2IE = 1;
    return adc_val;
}
```