Εργαστήριο Μικροϋπολογιστών
Θοδωρής Παπαρρηγόπουλος
el18040
Ομάδα 21

5ο Εργαστήριο

```
#define F_CPU 8000000 // FREQUENCY OF ATMEGA16

#include <xc.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>


/* Global Variables */
unsigned char mem[2], key_reg[2];
unsigned char value;
char duty = 0;


// read from keypad
static unsigned char scan_row(int i);
static unsigned char swap(unsigned char x);
static void scan_keypad();
static int scan_keypad_rising_edge();
static unsigned char keypad_to_ascii();

// ADC related
static void ADC_init(void);

void PWM_init();

static int increase_duty(int duty);
static int decrease_duty(int duty);

void write_2_nibbles(unsigned char b);
void lcd_data(unsigned char orisma);
void lcd_command(unsigned char orisma1);
void lcd_init();
void lcd_write(unsigned int adc);

ISR(ADC_vect) { // ADC Interrupt routine
       lcd_init();
       lcd_data('V');
       lcd_data('o');
       lcd_data('1');
       lcd_data('\n');

       unsigned int final = (5 * ADC / 1024.0 * 100);

       unsigned char a = 48 + final/100;
       unsigned char b = 48 + (final%100)/10;
       unsigned char c = 48 + (final%100)%10;

       lcd_data(a);
       lcd_data('.');
       lcd_data(b);
       lcd_data(c);
```

```
}
int main(void)
{
	DDRD = 0xff;
	DDRC = 0xf0; // necessary for keypad

	ADC_init();
	PWM_init();

	asm("sei"); // enable interupts

	lcd_init();


	while(1)
	{

		mem[0] = 0; // INITIALIZE RAM
		mem[1] = 0;
		if (scan_keypad_rising_edge()) {
			value = keypad_to_ascii();
			if (value == '1') {
				duty = increase_duty(duty);
				} else if (value == '2') {
				duty = decrease_duty(duty);
			}
			OCR0 = duty;
			ADCSRA |= 1 << ADSC;
			_delay_ms(8);
		}

	}
}



void ADC_init(void) // Initialize ADC
{
	ADMUX = 0x40;
	ADCSRA = (1<<ADEN) | (1<<ADIE)| (1<<ADPS2)| (1<<ADPS1)| (1<<ADPS0);
}

void PWM_init()
{
	//set TMR0 in fast PWM mode with non-inverted output, prescale=8
	TCCR0 = (1<<WGM00) | (1<<WGM01) | (1<<COM01) | (1<<CS01);
	DDRB|=(1<<PB3); //set PB3 pin as output
}


static int increase_duty(int duty) {
	++duty;
	if (duty > 255)
	duty = 0;
	return duty;
}

static int decrease_duty(int duty){
	--duty;
	if (duty < 0)
```

```
	duty = 255;
	return duty;
}

void lcd_write(unsigned int adc) {
	lcd_init();
	lcd_data('V');
	lcd_data('o');
	lcd_data('1');
	lcd_data('\n');

	char result[8];
	sprintf(result, "%d", duty);

	for (int i = 0; i < 8; ++i)
	lcd_data((char)result[i]);

}

unsigned char scan_row(int i) { // i = 1,2,3,4
	unsigned char a = ( 1 << 3 ); // SKIP 3 LSB
	a = (a << i); // SELECT ROW ACCORDING TO FUNCTION INPUT i
	PORTC = a; // WE SELECT ROW BY SETTING CORRESPONDING BIT TO 1
	_delay_us(500); // DELAY FOR REMOTE USAGE
	return PINC & 0x0F; // WE READ THE 4 LSB, '1' INDICATES SWITCH PUSHED
}
/* FUNCTION TO SWAP LO WITH HO BITS */
unsigned char swap(unsigned char x) {
	return ((x & 0x0F) << 4 | (x & 0xF0) >> 4);
}
/* SCAN ROWS(1..4) *DIFFERENT ORDER FROM EXERSISE DOCUMENT*
* FIRST ROW: PC4->PC0: 1, PC4->PC1: 2, PC4->PC2: 3, PC4->PC3: A
* SECOND ROW: PC5->PC0: 4, PC5->PC1: 5, PC5->PC2: 6, PC5->PC3: B
* THIRD ROW: PC6->PC0: 7, PC6->PC1: 8, PC6->PC2: 9, PC6->PC3: C
* FOURTH ROW: PC7->PC0: *, PC7->PC1: 0, PC7->PC2: #, PC7->PC3: D
*/
void scan_keypad() {
	unsigned char i;
	// check row 1, 0b0001-ROW CORRESPONDING TO PC4
	i = scan_row(1);
	key_reg[1] = swap(i); //key_reg[1] = first_row(4 MSB)-0000
	// check row 2, 0b0010-ROW CORRESPONDING TO PC5
	i = scan_row(2);
	key_reg[1] += i; //key_reg[1] = first_row(4 MSB)-second_row(4 LSB)
	// check row 3, 0b0100-ROW CORRESPONDING TO PC6
	i = scan_row(3);
	key_reg[0] = swap(i); //key_reg[0] = third_row(4 MSB) -0000
	// check row 4, 0b1000-ROW CORRESPONDING TO PC7
	i = scan_row(4);
	key_reg[0] += i; //key_reg[0] = third_row(4 MSB)-fourth_row(4 LSB)
	PORTC = 0x00; // added for remote usage
}
int scan_keypad_rising_edge() {
	// CHECK KEYPAD
	scan_keypad(); // RETURNS RESULTS IN key_reg
	// ADD TEMPORARY VARIABLES
	unsigned char tmp_keypad[2];
	tmp_keypad[0] = key_reg[0]; //tmp_keypad HOLD ACQUIRED DATA FROM SCAN_KEYPAD()
	tmp_keypad[1] = key_reg[1];
	_delay_ms(0x15); // APOFYGH SPINTHIRISMOU
```

```
	scan_keypad();
	key_reg[0] &= tmp_keypad[0]; // APPORIPSE TIS TIMES POU EMFANISAN SPINTHIRISMO
	key_reg[1] &= tmp_keypad[1];
	tmp_keypad[0] = mem[0]; // BRING LAST STATE OF SWITCHES FROM RAMTO tmp_keypad
	tmp_keypad[1] = mem[1];
	mem[0] = key_reg[0]; // STORE NEW KEYPAD STATE IN RAM FOR FUTURE CALL
	mem[1] = key_reg[1];
	key_reg[0] &= ~tmp_keypad[0]; // FIND KEYPAD SWITCHES THAT HAVE JUST BEEN PRESSED
	key_reg[1] &= ~tmp_keypad[1];
	return (key_reg[0] || key_reg[1]); // 16 BIT VALUE INDICATING FRESHLY PRESSED SWITCHES -
RETURNS 0 IF NO SWITCH PRESSED
}
/* CONVERT VALUE TO ASCII CODE *CHECK COMMENT ABOVE SCAN_KEYPAD FOR CORRESPONDENCE
* key_reg[0] = third_row(4 MSB)-fourth_row(4 LSB)
* key_reg[1] = first_row(4 MSB)-second_row(4 LSB)
* LSB -> MSB == LEFT -> RIGHT IN KEYPAD */
unsigned char keypad_to_ascii() {
	if (key_reg[0] & 0x01)
	return '*';
	if (key_reg[0] & 0x02)
	return '0';
	if (key_reg[0] & 0x04)
	return '#';
	if (key_reg[0] & 0x08)
	return 'D';
	if (key_reg[0] & 0x10)
	return '7';
	if (key_reg[0] & 0x20)
	return '8';
	if (key_reg[0] & 0x40)
	return '9';
	if (key_reg[0] & 0x80)
	return 'C';
	if (key_reg[1] & 0x01)
	return '4';
	if (key_reg[1] & 0x02)
	return '5';
	if (key_reg[1] & 0x04)
	return '6';
	if (key_reg[1] & 0x08)
	return 'B';
	if (key_reg[1] & 0x10)
	return '1';
	if (key_reg[1] & 0x20)
	return '2';
	if (key_reg[1] & 0x40)
	return '3';
	if (key_reg[1] & 0x80)
	return 'A';
	// Nothing Found
	return 0;
}

void write_2_nibbles(unsigned char b) {
	_delay_us(6000);
	unsigned char sth = PIND;
	sth = sth & 0x0F;
	unsigned char tmp1 = b & 0xF0;
	tmp1 += sth;
	PORTD = tmp1;
```

```
        PORTD |=  (1 << PD3);
        PORTD &= ~(1 << PD3);
        _delay_us(6000);
        unsigned char tmp2 = b & 0x0F;
        tmp2 = tmp2 << 4;
        unsigned char tmp3 = b & 0xF0;
        tmp3 = tmp3 >> 4;
        b = tmp2 + tmp3;
        b = b & 0xF0;
        b = b + sth;
        PORTD = b;
        PORTD |=  (1 << PD3);
        PORTD &= ~(1 << PD3);
}

void lcd_data(unsigned char orisma) {
        PORTD |= (1 << PD2);
        write_2_nibbles(orisma);
        _delay_us(43);
}

void lcd_command(unsigned char orisma1) {
        PORTD &= ~(1 << PD2);
        write_2_nibbles(orisma1);
        _delay_us(39);
}

void lcd_init() {
        _delay_ms(40);
        PORTD = 0x30;
        PORTD |= (1 << PD3);
        PORTD &= ~(1 << PD3);
        _delay_us(39);
        _delay_us(1000);
        PORTD = 0x30;
        PORTD |= (1 << PD3);
        PORTD &= ~(1 << PD3);
        _delay_us(39);
        _delay_us(1000);
        PORTD = 0x20;
        PORTD |= (1 << PD3);
        PORTD &= ~(1 << PD3);
        _delay_us(39);
        _delay_us(1000);
        unsigned char arg = 0x28;
        lcd_command(arg);
        arg = 0x0c;
        lcd_command(arg);
        arg = 0x01;
        lcd_command(arg);
        _delay_us(1530);
        arg = 0x06;
        lcd_command(arg);
}
```