MICROCOMPUTERS LABORATORY

Team 3

Εργαστήριο Μικροϋπολογιστών

5η Εργαστηριακή Άσκηση

Παναγιώτης Ντάγκας el18018

Βικέντιος Βιτάλης el18803

Τα ζητούμενα της 4ης εργαστηριακής άσκησης:

```
#include <xc.h>
#define F_CPU 8000000
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

unsigned char sw_state1[2];
unsigned char prev_state[2];
unsigned char duty = 0;
unsigned char buf[100];

//Write_2_nibbles: Αναλαμβάνει να στείλει σε 2 κομμάτια τα δεδομένα. Πρώτα
//τα msb και μετά τα lsb bits. Αυτή η ρουτίνα αξιοποιείται από την lcd_data
//και την lcd_command.
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);
}
//Η lcd_data θέτει το PD2 = 1 που είναι συνδεδεμένο στο RS, δηλαδή είναι
//δεδομένο το μήνυμα που στέλνεται.
void lcd_data(unsigned char orisma) {
        PORTD |= (1 << PD2);
        write_2_nibbles(orisma);
        _delay_us(43);
}
//Η lcd_command θέτει το PD2 = 0, δηλαδή και το RS = 0.
//Έτσι είναι εντολή αυτό που στέλνεται
void lcd_command(unsigned char orisma1) {
        PORTD &= ~(1 << PD2);
        write_2_nibbles(orisma1);
        _delay_us(39);
}
//Όπου γίνεται χρήση των lcd_data/lcd_command βάζουμε μια επιπλέον καθυστέρηση
//γιατί κάθε εντολή στην οθόνη χρειάζεται διαφορετικό χρόνο για να εκτελεστεί.
//Δηλαδή πρέπει να περιμένεις πριν στείλεις μια δεύτερη εντολή γιατί θα
//αγνοηθεί. Ο απαραίτητος χρόνος εκτέλεσης, άρα και της πρόσθετης απαιτούμενης
//αναμονής, καθορίζεται από τον κατασκευαστή του ελεγκτή.

//Η lcd_init κάνει αρχικοποίηση της οθόνης.
//Την καθαρίζει και την προετοιμάζεται να δείξει δεδομένα.Για RS = 0 -> εντολή
//Για RS = 1 -> χαρακτήρα
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);
}
```

```
unsigned char scan_row(int row){
	unsigned char a = ( 1 << 3 ); //Άσσος στο Bit3 αρχικά
	a = (a << row); //Τότε ο άσσος μετακινείται στον αριθμό της επιθυμητής
//σειράς
	PORTC = a;
	_delay_us(500);
	return PINC & 0x0F; //Επιστροφή εισόδου από τη συγκεκριμένη σειρά
}

//Το πληκτρολόγιο διαβάζεται γραμμή-γραμμή.
//Αυτό γίνεται με την scan_row η οποία καλείται 4 φορές από την scan_keypad και
//έτσι διαβάζονται και οι 4 γραμμές του πληκτρολογίου.
void scan_keypad(){


	sw_state1[0] = (scan_row(1) << 4);

	sw_state1[0] +=  scan_row(2);


	sw_state1[1] = (scan_row(3) << 4);

	sw_state1[1] += scan_row(4);

	PORTC = 0x00; //Το πληκτρολόγιο έχει απενεργοποιηθεί

}
//Η scan_keypad_rising_edge ουσιαστικά καλεί 2 scan_keypad, συγκρίνει τις 2
//καταστάσεις και κρατάει μόνο τα κοινά για να αποφευχθεί ο σπινθηρισμός.
//Σπινθηρισμός εξ αποστάσεως δεν υπάρχει. Στην πλακέτα όμως έχουμε μηχανικό
//πάτημα. Έτσι, ο τρόπος που θα πατηθεί ή θα αφεθεί ένα κουμπί, μπορεί να
//στείλει κι άλλα σήματα. Για να αποφευχθεί αυτό εισάγουμε καθυστερήσεις.
void scan_keypad_rising_edge(){
	unsigned char temp[2];
	scan_keypad();
	temp[0] = sw_state1[0];
	temp[1] = sw_state1[1];

	_delay_ms(0x15);

	scan_keypad();

	sw_state1[0] &= temp[0];
	sw_state1[1] &= temp[1];

	temp[0] = ~prev_state[0];
	temp[1] = ~prev_state[1];

	prev_state[0] = sw_state1[0];
	prev_state[1] = sw_state1[1];

	sw_state1[0] &= temp[0];
	sw_state1[1] &= temp[1];

}
```

```
//Η keypad_to_ascii βλέπει ποιο πλήκτρο πατήθηκε και επιστρέφει τον ascii
//κωδικό. Ξεκινάει από το lsb, το πρώτο bit που βρίσκει άσσο είναι το
//κουμπί που πατήθηκε κι επιστρέφει τον αντίστοιχο χαρακτήρα.
unsigned char keypad_to_ascii(unsigned char scan[2]) {
	if (scan[1] & 0x01)
	return '*';

	if (scan[1] & 0x02)
	return '0';

	if (scan[1] & 0x04)
	return '#';

	if (scan[1] & 0x08)
	return 'D';

	if (scan[1] & 0x10)
	return '7';

	if (scan[1] & 0x20)
	return '8';

	if (scan[1] & 0x40)
	return '9';

	if (scan[1] & 0x80)
	return 'C';

	if (scan[0] & 0x01)
	return '4';

	if (scan[0] & 0x02)
	return '5';

	if (scan[0] & 0x04)
	return '6';

	if (scan[0] & 0x08)
	return 'B';

	if (scan[0] & 0x10)
	return '1';

	if (scan[0] & 0x20)
	return '2';

	if (scan[0] & 0x40)
	return '3';

	if (scan[0] & 0x80)
	return 'A';

	// Δεν βρέθηκε τίποτα
	return 0;
}


void init_adc() {
	ADMUX = 0x40;
	ADCSRA = (1<<ADEN)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
}
```

```
void print_vo1() {
	lcd_data('V');
	lcd_data('o');
	lcd_data('1');
	lcd_data('\n');
}

void PWM_init()
{
	//Θέτουμε τον TMR0 σε γρήγορο PWM(Pulse Width Modulation) με μη-
	//ανεστραμμένη είσοδο, prescale = 8
	TCCR0 = (1<<WGM00) | (1<<WGM01) | (1<<COM01) | (1<<CS01);
	DDRB|=(1<<PB3); //Θέτουμε το PB3 ‘pin’ σαν έξοδο
}

ISR(ADC_vect) {
	lcd_init();
	print_vo1();
	unsigned int digital_voltage = ADC;
	float analog_voltage = digital_voltage * 5.0/1024;
	unsigned int final = (unsigned int)(analog_voltage*100);
	unsigned char first = 48 + (final / 100);
	unsigned char second = 48 + ((final % 100)/10);
	unsigned char third = 48 + ((final % 100)%10);
	lcd_data(first);
	lcd_data('.');
	lcd_data(second);
	lcd_data(third);
}

int main(void)
{
	DDRD = 0xFF;
	DDRC = 0xF0;
	init_adc();
	PWM_init();
	asm("sei");
	lcd_init();
	print_vo1();
	while(1)
	{
		prev_state[0] = 0; //Αρχικοποίηση προηγούμενης κατάστασης στο 00
		prev_state[1] = 0;
		scan_keypad_rising_edge();
		if((sw_state1[0] != 0) | (sw_state1[1] != 0)) {
			//Αν πατηθεί το 1, το duty cycle αυξάνεται κατά 1
			if(keypad_to_ascii(sw_state1) == '1') {
				if(duty == 255) {
					duty = 0;
				}
				else {
					duty++;
				}
				OCR0 = duty;
				ADCSRA |= 1 << ADSC;
				_delay_ms(8);
			}
			//Αν πατηθεί το 2, το duty cycle μειώνεται κατά 1
			else if(keypad_to_ascii(sw_state1) == '2') {
				if(duty == 0) {
					duty = 255;
				}
```

```
                        else {
                                duty--;
                        }
                        OCR0 = duty;
                        ADCSRA |= 1 << ADSC;
                        _delay_ms(8);
                }
        }
}
```