Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εργαστήριο Μικροϋπολογιστών 2021-2022

5η Εργασία (AVR)

Κωνσταντίνος Σιδέρης, Ομάδα 48

Α.Μ.: 03118134

Η υλοποίηση των ρουτινών για την χρήση της οθόνης LCD υλοποιήθηκαν ως εξωτερικές συναρτήσεις assembly σε .S αρχεία.

wait_usec.S

```
#define _SFR_ASM_COMPAT 1
#define __SFR_OFFSET 0
#include <avr/io.h>

.global wait_usec

wait_usec:              ;Υλοποίηση ρουτίνας wait_usec
            sbiw r24 ,1
            nop
            nop
            nop
            nop
            brne wait_usec
            ret
```

wait_msec.S

```
#define _SFR_ASM_COMPAT 1
#define __SFR_OFFSET 0
#include <avr/io.h>

.global wait_msec

wait_msec:              ;Υλοποίηση ρουτίνας wait_msec
            push r24
            push r25
            ldi r24 , lo8(998)
            ldi r25 , hi8(998)
            rcall wait_usec
            pop r25
            pop r24
            sbiw r24 , 1
            brne wait_msec
            ret
```

write_2_nibbles_sim.S

```
#define _SFR_ASM_COMPAT 1
#define __SFR_OFFSET 0
#include <avr/io.h>

.global write_2_nibbles_sim

write_2_nibbles_sim:          ;Υλοποίηση ρουτίνας write_2_nibbles_sim
            push r24
```

```
            push r25
            ldi r24 ,lo8(6000)
            ldi r25 ,hi8(6000)
            rcall wait_usec
            pop r25
            pop r24
            push r24
            in r25, PIND
            andi r25, 0x0f
            andi r24, 0xf0
            add r24, r25
            out PORTD, r24
            sbi PORTD, PD3
            cbi PORTD, PD3
            push r24
            push r25
            ldi r24 ,lo8(6000)
            ldi r25 ,hi8(6000)
            rcall wait_usec
            pop r25
            pop r24
            pop r24
            swap r24
            andi r24 ,0xf0
            add r24, r25
            out PORTD, r24
            sbi PORTD, PD3
            cbi PORTD, PD3
            ret
```

lcd_command_sim.S

```
#define _SFR_ASM_COMPAT 1
#define __SFR_OFFSET 0
#include <avr/io.h>

.global lcd_command_sim

lcd_command_sim:        ;Υλοποίηση ρουτίνας lcd_comand_sim
            push r24
            push r25
            cbi PORTD, PD2
            rcall write_2_nibbles_sim
            ldi r24, 39
            ldi r25, 0
            rcall wait_usec
            pop r25
            pop r24
            ret
```

lcd_data_sim.S

```
#define _SFR_ASM_COMPAT 1
#define __SFR_OFFSET 0
#include <avr/io.h>

.global lcd_data_sim

lcd_data_sim:                ;Υλοποίηση ρουτίνας lcd_data_sim
            push r24
            push r25
            sbi PORTD, PD2
            rcall write_2_nibbles_sim
            ldi r24 ,43
            ldi r25 ,0
            rcall wait_usec
            pop r25
            pop r24
            ret
```

lcd_init_sim.S

```
#define _SFR_ASM_COMPAT 1
#define __SFR_OFFSET 0
#include <avr/io.h>

.global lcd_init_sim

lcd_init_sim:                ;Υλοποίηση ρουτίνας lcd_init_sim
            push r24
            push r25

            ldi r24, 40
            ldi r25, 0
            rcall wait_msec
            ldi r24, 0x30
            out PORTD, r24
            sbi PORTD, PD3
            cbi PORTD, PD3
            ldi r24, 39
            ldi r25, 0
            rcall wait_usec
            push r24
            push r25
            ldi r24,lo8(1000)
            ldi r25,hi8(1000)
            rcall wait_usec
            pop r25
            pop r24
            ldi r24, 0x30
```

```
out PORTD, r24
sbi PORTD, PD3
cbi PORTD, PD3
ldi r24,39
ldi r25,0
rcall wait_usec
push r24
push r25
ldi r24 ,lo8(1000)
ldi r25 ,hi8(1000)
rcall wait_usec
pop r25
pop r24
ldi r24,0x20
out PORTD, r24
sbi PORTD, PD3
cbi PORTD, PD3
ldi r24,39
ldi r25,0
rcall wait_usec
push r24
push r25
ldi r24 ,lo8(1000)
ldi r25 ,hi8(1000)
rcall wait_usec
pop r25
pop r24
ldi r24,0x28
rcall lcd_command_sim
ldi r24,0x0c
rcall lcd_command_sim
ldi r24,0x01
rcall lcd_command_sim
ldi r24, lo8(1530)
ldi r25, hi8(1530)
rcall wait_usec
ldi r24 ,0x06
rcall lcd_command_sim
pop r25
pop r24
ret
```

Ακολουθεί ο κώδικας του κύριου προγράμματος (C) με σχόλια:

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

char reg[2], mem[2], key1, key2, d[5];
volatile float VPA0, previous_VPA0 = 0.0;

extern void lcd_init_sim();
extern void lcd_data_sim(char c);

char swap(char x) { //Υλοποίηση εντολής swap
	return ((x & 0x0F) << 4 | (x & 0xF0) >>4);
}

char scan_row(int r) { //Υλοποίηση ρουτίνας scan_row_sim
	char o = 0x08;
	o = (o << r);	//Ολίσθηση r φορές του o = 00001000
	PORTC = o;  //Θέτουμε την γραμμή που σκανάρουμε σε 1
	_delay_us(500);   //Καθυστέρηση 500ms για απομακρυσμένη λειτουργία
	return PINC & 0x0F; //Επιστροφή τεσσάρων  LSB της PORTC (στήλες
πληκτρολογίου)
}

void scan_keypad() { //Υλοποίηση ρουτίνας scan_keypad_sim
	char c;
	c = scan_row(1); //Σκανάρισμα πρώτης σειράς
	reg[1] = swap(c); //Αποθήκευση στα 4 MSB των πρώτων 8 bit του 16-bit
register reg
	c = scan_row(2); //Σκανάρισμα δεύτερης σειράς
	reg[1] = reg[1] + c; //Αποθήκευση στα 4 LSB των πρώτων 8 bit του
16-bit register reg
	c = scan_row(3); //Σκανάρισμα τρίτης σειράς
	reg[0] = swap(c); //Αποθήκευση στα 4 MSB των δεύτερων 8 bit του 16-bit
register reg
	c = scan_row(4); //Σκανάρισμα τέταρτης σειράς
	reg[0] = reg[0] + c; //Αποθήκευση στα 4 LSB των δεύτερων 8 bit του
16-bit register reg
	PORTC = 0x00;
}

int scan_keypad_rising_edge() { //Υλοποίηση ρουτίνας scan_row_rising_edge_sim
	char tmp[2];

	scan_keypad();	//Σκανάρισμα πληκτρολογίου
	tmp[0] = reg[0]; //Προσωρινή αποθήκευση του αποτελέσματος
```

```
	tmp[1] = reg[1];

	_delay_ms(15);	//Αναμονή 15ms λόγω σπινθηρισμών

	scan_keypad();	//Δεύτερο σκανάρισμα πληκτρολογίου
	reg[0] = reg[0] & tmp[0]; //Απόρριψη πλήκτρων που εμφάνισαν
σπινθηρισμό
	reg[1] = reg[1] & tmp[1];

	tmp[0] = mem[0]; //Λήψη προηγούμενης κατάστασης διακοπτών από την  RAM
	tmp[1] = mem[1];

	mem[0] = reg[0]; //Αποθήκευση τωρινής κατάστασης διακοπτών από την
RAM
	mem[1] = reg[1];

	reg[0] = reg[0] & (~tmp[0]); //Εύρεση διακοπτών που έχουν μόλις πατηθεί
	reg[1] = reg[1] & (~tmp[1]);

	return (reg[0] || reg[1]); //Επιστροφή των διακοπτών που μόλλις
πατήθηκαν (0 αν δεν έχει πατηθεί πλήκτρο)
}

char keypad_to_ascii() { //Υλοποίηση ρουτίνας keypad_to_ascii_sim
	if ((reg[0]&0x01) == 0x01)
	return '*'; //Εύρεση πατημένου πλήκτρου και επιστροφή του κωδικού ASCII
που του αντιστοιχεί
	if ((reg[0]&0x02) == 0x02)
	return '0';
	if ((reg[0]&0x04) == 0x04)
	return '#';
	if ((reg[0]&0x08) == 0x08)
	return 'D';
	if ((reg[0]&0x10) == 0x10)
	return '7';
	if ((reg[0]&0x20) == 0x20)
	return '8';
	if ((reg[0]&0x40) == 0x40)
	return '9';
	if ((reg[0]&0x80) == 0x80)
	return 'C';
	if ((reg[1]&0x01) == 0x01)
	return '4';
	if ((reg[1]&0x02) == 0x02)
	return '5';
	if ((reg[1]&0x04) == 0x04)
	return '6';
	if ((reg[1]&0x08) == 0x08)
```

```
	return 'B';
	if ((reg[1]&0x10) == 0x10)
	return '1';
	if ((reg[1]&0x20) == 0x20)
	return '2';
	if ((reg[1]&0x40) == 0x40)
	return '3';
	if ((reg[1]&0x80) == 0x80)
	return 'A';
	return 0;   //Εάν δεν έχει πατηθεί κάποιο πλήκτρο επιστρέφει 0
}

void ADC_init() {
	ADMUX = (1 << REFS0);   //Vref = Vcc και επιλογή A0 για είσοδο
	ADCSRA = (1 << ADEN | 1 << ADIE | 1 << ADPS2 | 1 << ADPS1 | 1 << ADPS0);
//Ενεργοποίηση ADC, διακοπών και ρύθμιση prescaler CK/128=62.5Khz
}

ISR(ADC_vect) {
	VPA0 = (ADC/204.8);     //Υπολογισμός του Vout1 (ADC*Vcc/1024)
	if (VPA0 != previous_VPA0) { //Εάν έχουμε καινούργια μέτρηση τότε την
εμφανίζουμε στην οθόνη
		previous_VPA0 = VPA0;
		dtostrf(VPA0, 4, 2, d); //Μετατροπή Vout1 από float σε
string(αποθηκεύεται ένα ψηφίο σε κάθε θέση του πίνακα d[5])
		lcd_init_sim();
		lcd_data_sim('V');
		lcd_data_sim('o');
		lcd_data_sim('1');
		lcd_data_sim('\n');
		lcd_data_sim(d[0]);
		lcd_data_sim(d[1]);
		lcd_data_sim(d[2]);
		lcd_data_sim(d[3]);
	}
}

ISR(TIMER1_OVF_vect) {
	key2 = 0;
	if (scan_keypad_rising_edge() != 0) { //Διάβσμα πληκτολογίου μέσα στην
ρουτίνα διακοπής του TIMER1
		key2 = keypad_to_ascii(); //Μετατροπή σε ψηφίο ASCII
	}
	if (key2 == '1') { //Αν πατήθηκε το 1 αυξάνουμε το Duty Cycle κατά 1
		OCR0++;
	}
	else if (key2 == '2') { //Αν πατήθηκε το 2 μειώνουμε το Duty Cycle κατά
1
```

```
            OCR0--;
      }
      ADCSRA |= (1<<ADSC); //Ενεργοποίηση μετατροπής ADC
      TCNT1 = 0xF3CB; //Επαναρύθμιση του TCNT1 για υπερχείλιση μετά από 0.4
sec
}

int main(void) {
      DDRD = 0xFF; //Αρχικοποίηση PORTD ως έξοδο
      DDRC = 0xF0; //Αρχικοποίηση 4ων MSB της PORTC ως έξοδο και 4ων LSB της
PORTC ως είσοδο
      DDRB |= (1<<PB3); //Αρχικοποίηση PB3 ως έξοδο

      ADC_init();

      TIMSK = (1 << TOIE1); //Ενεργοποίηση διακοπής υπερχείλισης του χρονιστή
TCNT1 για τον TIMER/COUNTER1
      TCCR1B = (1<<CS12) | (0<<CS11) | (1<<CS10); //CK/1024
      TCNT1 = 0xF3CB; //Αρχικοποίηση του TCNT1 για υπερχείλιση μετά από 0.4 sec

      TCCR0 = (1<<WGM00) | (1<<WGM01) | (1<<COM01) | (1<<CS01); //Ρύθμιση
TIMER/COUNTER0 για παραγωγή μη ανάστροφης FAST PWM συχνότητας 4KHz
      OCR0 = 0xFF; //Αρχικοποίηση του Duty Cycle σε 255(δηλαδή 100%)
      sei(); //Ενεργοποίηση διακοπών

      while (1) {
            mem[0] = 0x00;    //Αρχικοποίηση μεταβλητής μνήμης RAM
            mem[1] = 0x00;

            while (1) {
                  if (scan_keypad_rising_edge() != 0) { //Διάβασμα
πληκτρολογίου μέχρι να πατηθεί πλήκτρο
                        key1 = keypad_to_ascii(); //Μετατροπή σε ψηφίο ASCII
                        break;
                  }
            }

            if (key1 == '1') {//Αν πατήθηκε το 1 αυξάνουμε το Duty Cycle κατά
1
                  OCR0++;
            }
            else if (key1 == '2') { //Αν πατήθηκε το 2 μειώνουμε το Duty Cycle
κατά 1
                  OCR0--;
            }
      }
      return 0;
}
```