Εργαστηριο μικρουπολογιστών 5η ασκηση/AVR
Φαρδέλας Ιωάννης 03113190
Παραλίκας Ηλίας 013116606
Ομάδα 28

```
#define F_CPU 8000000UL
#include "avr/io.h"
#define SPARK_DELAY_TIME 20
#include <util/delay.h>
#include <avr/interrupt.h>
//OC0 is connected to pin PB3
//OC1A is connected to pin PD5
//OC2 is connected to pin PD7

///****ΣΥΝΑΡΤΗΣΕΙΣ ΓΙΑ ΔΙΑΒΑΣΜΑ ΑΠΟ ΤΟ KEYPAD ΙΔΙΕΣ ΜΕ ΠΡΟΗΓΟΥΜΕΝΕΣ
ΣΕΙΡΕΣ***

unsigned int previous_keypad_state = 0; //hold the state of the keyboard
0x0000
int ascii[16];                                   //Is the ascii code for
each key on the keyboard

unsigned char scan_row_sim(int row)
{
	unsigned char temp;
	volatile unsigned char pressed_row;

	temp = 0x08;
	PORTC = temp << row;
	_delay_us(500);
	asm("nop");
	asm("nop");
	pressed_row = PINC & 0x0f;

	return pressed_row;
}
unsigned int scan_keypad_sim(void)
{
	volatile unsigned char pressed_row1, pressed_row2, pressed_row3,
pressed_row4;
	volatile unsigned int pressed_keypad = 0x0000;

	pressed_row1 = scan_row_sim(1);
	pressed_row2 = scan_row_sim(2);
	pressed_row3 = scan_row_sim(3);
	pressed_row4 = scan_row_sim(4);

	pressed_keypad = (pressed_row1 << 12 | pressed_row2 << 8) |
(pressed_row3 << 4) | (pressed_row4);
	PORTC =0x00;
	return pressed_keypad;
}
unsigned int scan_keypad_rising_edge_sim(void)
```

```
{
    unsigned int pressed_keypad1, pressed_keypad2,
current_keypad_state, final_keypad_state;

    pressed_keypad1 = scan_keypad_sim();
    _delay_ms(SPARK_DELAY_TIME);
    pressed_keypad2 = scan_keypad_sim();
    current_keypad_state = pressed_keypad1 & pressed_keypad2;
    final_keypad_state = current_keypad_state & (~
previous_keypad_state);
    previous_keypad_state = current_keypad_state;

    return final_keypad_state;
}
unsigned char keypad_to_ascii_sim(unsigned int final_keypad_state)
{
    volatile int j;
    volatile unsigned int temp;

    for (j=0; j<16; j++)
    {
        temp = 0x01;
        temp = temp << j;
        if (final_keypad_state & temp) //if you find the only pressed
key then return
        {
            return ascii[j];
        }
    }
    //should not reach here
    return 1;
}
void initialize_ascii(void)
{
    ascii[0] = '*';
    ascii[1] = '0';
    ascii[2] = '#';
    ascii[3] = 'D';
    ascii[4] = '7';
    ascii[5] = '8';
    ascii[6] = '9';
    ascii[7] = 'C';
    ascii[8] = '4';
    ascii[9] = '5';
    ascii[10] = '6';
    ascii[11] = 'B';
    ascii[12] = '1';
    ascii[13] = '2';
    ascii[14] = '3';
    ascii[15] = 'A';
}
unsigned char read4x4(void)
{
    unsigned int keypad_state;
    unsigned char ascii_code;

    keypad_state = scan_keypad_rising_edge_sim(); // read the state of
the keyboard
```

```
     if (!keypad_state)
     {
          return 0;
     }
     ascii_code = keypad_to_ascii_sim(keypad_state); // encode it to
ascii code

     return ascii_code;
}


///ΑΡΧΙΚΟΠΟΙΗΣΗ PWM ΟΠΩς ΦΑΙΝΕΤΑΙ ΣΤΟΝ ΕΡΓΑΣΤΗΡΙΑΚΌ ΟΔΗΓΟ
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


}
//ΣΥΝΑΡΤΗΣΕΙΣ ΓΙΑ ΛΕΙΤΟΥΡΓΙΑ lcd  ΙΔΙΕΣ ΜΕ ΠΡΟΗΓΟΥΜΕΝΕΣ
ΑΣΚΗΣΕΙΣ ////ΜΕΤΆΦΡΑΣΗ ΤΩΝ ΕΤΟΙΜΩΝ ΣΥΝΑΡΤΗΣΕΩΝ ΠΟΥ ΔΙΝΟΝΤΑΙ ΣΤΟΝ ΟΔΗΓΟ //
ΑΠΟ ΑΣΣΕΜΠΛΙ //ΣΕ C
unsigned char swapNibbles(unsigned char x)
{
     return ((x & 0x0F) << 4 | (x & 0xF0) >> 4);
}
void write_2_nibbles_sim(unsigned char data)
{
     _delay_us(6000);

     unsigned char temp, Nibble_data;

     temp = PIND;
     temp = temp & 0x0f;
     Nibble_data = data & 0xf0;
     Nibble_data = temp + Nibble_data;
     PORTD = Nibble_data;

     PORTD = PORTD | 0x08;
     PORTD = PORTD & 0xf7;
     _delay_us(6000);

     data = swapNibbles(data);
     Nibble_data = data & 0xf0;
     Nibble_data = Nibble_data + temp;
     PORTD = Nibble_data;

     PORTD = PORTD | 0x08;
     PORTD = PORTD & 0xf7;
     return;
}
void lcd_data_sim(unsigned char data)
{
     PORTD = PORTD | 0x04;
     write_2_nibbles_sim(data);
     _delay_us(43);
     return;
```

```
}
void lcd_command_sim(unsigned char data)
{
      PORTD = PORTD & 0xfb;
      write_2_nibbles_sim(data);
      _delay_us(39);
      return;
}
void lcd_init_sim()
{
      _delay_ms(40);
      for (int i = 1; i <= 2; i++)
      {
            PORTD = 0x30;
            PORTD = PORTD | 0x08;
            PORTD = PORTD & 0xf7;

            _delay_us(39);
            _delay_us(1000);
      }

      PORTD = 0x20;
      PORTD = PORTD | 0x08;
      PORTD = PORTD & 0xf7;

      _delay_us(39);
      _delay_us(1000);

      lcd_command_sim(0x28);
      lcd_command_sim(0x0C);
      lcd_command_sim(0x01);

      _delay_us(1530);

      lcd_command_sim(0x06);

      return;
}

//ΑΡΧΙΚΟΠΟΙΗΣΗ ADC ΟΠΩΣ ΣΤΗΝ ΠΡΟΗΓΟΥΜΕΝΗ ΑΣΚΗΣΗ
void ADC_init()
{ //initialize the ADC with CK/128,Vref=Vcc ,A0 port to take the ADC
      ADCSRA = (1 << ADEN) | (1 << ADIE) | (1 << ADPS2) | (1 << ADPS1) |
(1 << ADPS0);
      ADMUX = (1 << REFS0);
}
void initialize_timer_interrupts()
{
      TCNT1 = 0xfcf3;
//init to specific number for 0.1sec overflow
      TCCR1B = (1 << CS12) | (0 << CS11) | (1 << CS10); //CLK/1024  //
Timer mode with 1024 prescler
      TIMSK = (1 << TOIE1);                                   //enable
Timer1
}
```

```
//INTERRUPT ΣΥΝΑΡΤΗΣΗ, ΔΙΑΒΑΖΕΙ ΣΕ ΚΑΘΕ ΙΝΤΕΡΑΠΤ ΤΗΝ ADC STHN PA0
ΜΕΤΑ ΥΠΟΛΟΓΖΕΙ VIN ΑΠΟ VIN=VREF*ADC/1024, ΑΠΟΜΟΝΩΝΕΙ ΤΑ 2 ΔΕΚΑΔΙΚΑ ΣΕ 2
ΜΕΤΑΒΛΗΤΕΣ ΚΑΙ ΤΑ ΕΜΦΑΝΙΖΕΙ ΣΤΟΝ LCD
ISR(TIMER1_OVF_vect) // Timer1 ISR
{

      ADCSRA |= (1 << ADSC); //start the ADC transformation
      _delay_us(10);           //wait for the transformation
      int  A,B,C;
      unsigned char a1,b1,c1,h;
      double Vin,Ain, AinLow;
      cli();                         // close the interrupts when we read
the ADC
      AinLow = ADCL;      //read the ADCL
      Ain = ADCH * 256; //read the ADCH and mul with the 256 to correct
the number
      sei();                         //enable the interrupts
      Ain = Ain + AinLow;      //add the 2 ADCL ADCH
      Vin=(5*Ain)/1024;

      A=(int)Vin;
      B=(Vin-A)*10;

      h=Vin*100-A*100-B*10;

      C=(int)h;

      a1=A+'0';
      b1=B+'0';
      c1=C+'0';

      lcd_init_sim();
      lcd_data_sim('V');
      lcd_data_sim('o');
      lcd_data_sim('1');
      lcd_data_sim('\n');
      lcd_data_sim(a1);
      lcd_data_sim('.');
      lcd_data_sim(b1);
      lcd_data_sim(c1);

      TCNT1 = 0xfcf3;
}
ISR(ADC_vect)
{ //just refresh the ADCH,ADCL
}

//ΑΜΑ ΡΑΤΑΜΕ 1 ΑΥΞΑΝΕΕΤΑΙ Ο OCR0 ΑΜΑ ΠΑΤΑΜΕ 2 ΜΙΚΡΑΙΝΕΙ ΚΑΤΑ 1


int main ()
{
      volatile unsigned char number, duty  ;
      PWM_init();
      uint8_t step=1 ;
      OCR0=150;
      DDRB = 0Xff; // B for output
      DDRC = 0xf0;
```

```
    DDRD = 0xff;
    initialize_ascii();
    ADC_init();
    initialize_timer_interrupts();
    lcd_init_sim();
    sei();

    while (1)
    {
        do
        {
            number = read4x4(); // wait for the number to be pushed
            _delay_ms(8) ;
        }while(!number);

        switch (number){
        case '1':
        if(OCR0<255){
        OCR0+=step;
        _delay_ms(8);
        }
        break;
        case '2':
        if(OCR0>0){
       OCR0-=step;
        _delay_ms(8);
        break;
        }
        }

    }
}
```