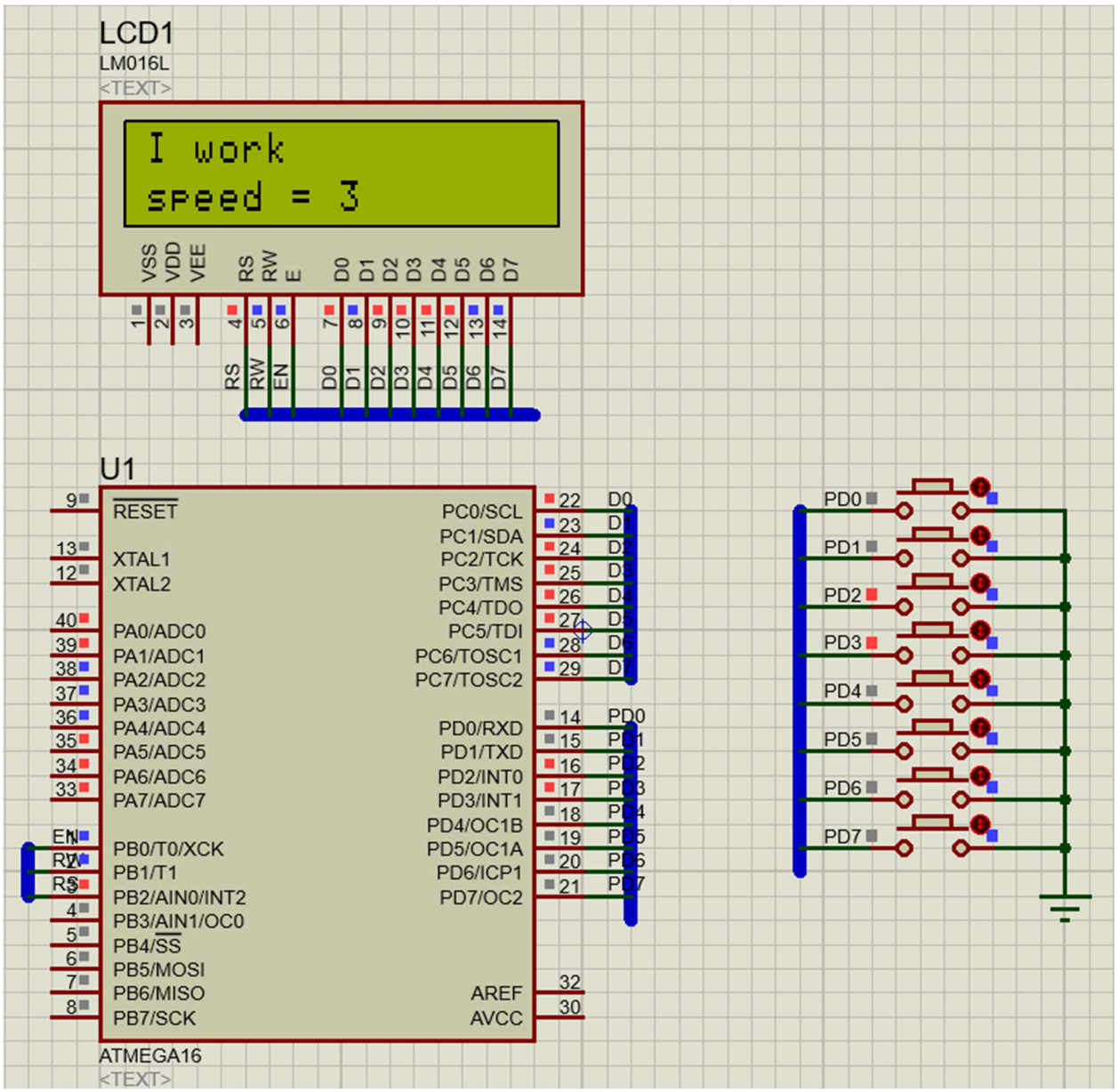
1.1 Разработать функцию, которая в момент включения платы C51 запускает «демо-режим» - выводит на ЖК сообщение с предложением о нажатии кнопок и зажигает на светодиодах платы самостоятельно разработанную студентом последовательность.

1.2 Разработать функцию нажатия свободных кнопок, изменяющую скорость бегущих огней.

2. Разработать функцию вывода на ЖК дисплей таймера от 0 до N, где N-произвольное число. Управление таймером: скорость изменения значений и пауза/запуск счёта задавать через прерывания INT0, INT2.

Для задания 1:



```
#define F_CPU 12000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define RS 2
#define RW 1
#define EN 0

void button_init(void);
void timer_init(void);
void lcd_init(void);

void lcd_com(unsigned char);
void lcd_dat(unsigned char);
void lcd_string(char*);

void next_step(int i);

uint8_t mode = 0;       // Текущие состояние
uint8_t mode_last = 0;  // Предидущие состояние

uint8_t speed_mode = 1; // Скоросной режим
char speed_char[] = "speed = 1"; // Строка для отображения скорости на дисплее

uint16_t initial_timer_value = 55000; // Изначальное значение для таймера
uint8_t pattern = 0b11111000; // Паттерн бегущих огней
uint8_t direction = 0; // Режим бега влево/вправо
uint8_t step = 0; // Шаг для бегущих огней

int main(void)
{
        timer_init();
    lcd_init();
    button_init();
    sei();

        int running_line_step = 0;
        int messageLength = 61;

    while (1)
    {
                switch (mode)
```

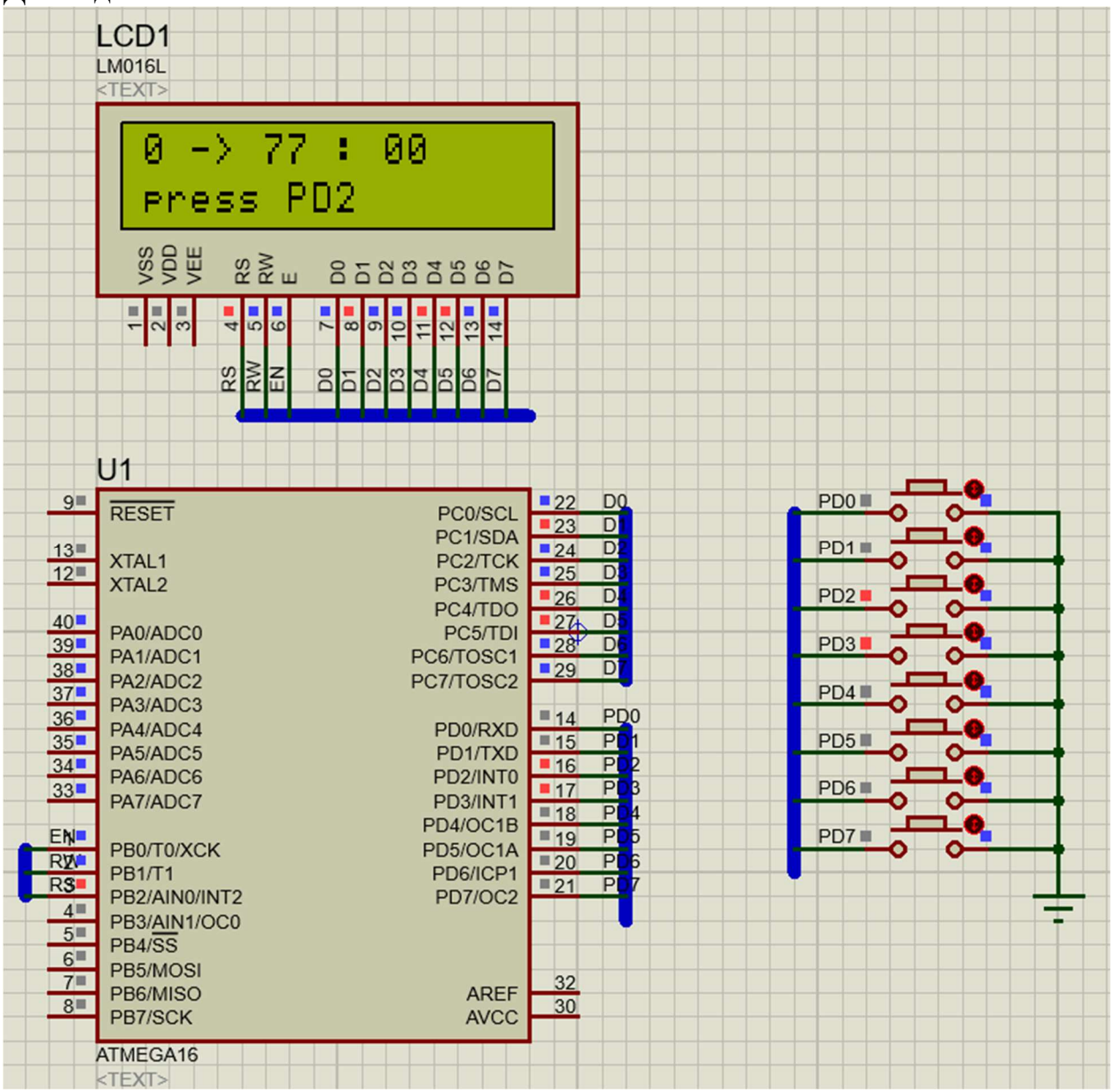
```
            {
                case (0):{
                    if (mode_last == 1) {
                        mode_last = 0;
                        running_line_step = 0;
                    }
                    next_step(running_line_step);
                    running_line_step++;
                    if (running_line_step >= messageLength){
                        running_line_step = 0;
                    }
                    break;
                }
                case(1):{
                    if (mode_last == 0) {
                        mode_last = 1;

                        lcd_com(0x01);
                        lcd_com(0x80);
                        lcd_string("I work");
                        lcd_com(0xC0);
                    }

                    lcd_com(0xC0);
                    lcd_string(speed_char);
                    break;
                }
            }
    }
}



ISR(INT0_vect){
    mode_last = mode;
    mode = !mode;
    TCNT1 = initial_timer_value;
};

ISR(INT1_vect){
    speed_mode += 1;
     if (speed_mode > 3){
        speed_mode = 1;
     }
```

```
        switch (speed_mode)
        {
                case (3):{ speed_char[8] = '3'; initial_timer_value = 65000; break; }
                case (2):{ speed_char[8] = '2'; initial_timer_value = 60000; break; }
                case (1):{ speed_char[8] = '1'; initial_timer_value = 55000; break; }
        }
};

ISR(TIMER1_OVF_vect)
{
        if (mode == 0) {
                PORTA = 0b00000000;
                return;
        }

        PORTA = pattern;
        if (direction == 0) {
                pattern = (pattern << 1) | (pattern >> 7); // Сдвиг вправо
                } else {
                pattern = (pattern >> 1) | (pattern << 7); // Сдвиг влево
        }
        step++;
        if(step > 4) {
                direction = !direction;
                step = 0;
        }
        TCNT1 = initial_timer_value;
}

void next_step(int i) {
        char staticText[] = "Instruction";
        char message[] = "  Press the button! 1 - instructions or run; 2 - lights
speed;";
        int messageLength = 62;

        char message16[16]; // массив для хранения 16 символов +
завершающий нуль

        lcd_com(0x80); // Устанавливаем курсор на первую строку
        lcd_string(staticText);
        lcd_com(0xC0); // Устанавливаем курсор на вторую строку

        for (int j = 0; j < 16; j++) {
                message16[j] = message[(j + i) % messageLength];
        }
```

```
        lcd_string(message16);
        _delay_ms(100); // Доп задержка для чтения бегущей строки
}

void button_init(void){
        DDRD &=~(1<<PD2);
        DDRD &=~(1<<PD3);
        PORTD |=(1<<PD2)|(1<<PD3);
        GICR |= (1<<INT0)|(1<<INT1);
        MCUCR &=~ (1<<ISC01)|(1<<ISC00);
        MCUCR &=~ (1<<ISC11)|(1<<ISC10);
}

void lcd_com(unsigned char p){
        PORTB &= ~(1<<RS);
        PORTB |= (1<<EN);
        PORTC = p;
        _delay_us(500);
        PORTB &=~(1<<EN);
        _delay_us(500);
}

void lcd_dat(unsigned char p){
        PORTB|=(1<<RS)|(1<<EN);
        PORTC=p;
        _delay_us(500);
        PORTB&=~(1<<EN);
        _delay_us(500);
}

void lcd_init(void){
        // Устанавливаем пины RS, RW и EN порта DDRB как выходные
        DDRB |= (1<<RS)|(1<<RW)|(1<<EN);
        PORTB=0x00; // Обнуляем порт B
        DDRC=0xFF; // Устанавливаем порт C как порт вывода
        PORTC=0x00; // Обнуляем порт C
        _delay_us(500); // Задержка 500 микросекунд
        lcd_com(0x08); // Инициализация дисплея: выключаем дисплей
        _delay_us(500); // Задержка 500 микросекунд
        lcd_com(0x3C); // Установка режима 8 бит данных, 2 строки, 5x8 точек
        _delay_us(500); // Задержка 500 микросекунд
        lcd_com(0x01); // Очистка дисплея
        _delay_us(500); // Задержка 500 микросекунд
```

```
	lcd_com(0x06); // Установка направления пути записи, увеличение адреса на 1
	_delay_us(900); // Задержка 900 микросекунд
	lcd_com(0x0C); // Включаем дисплей без курсора
}

void timer_init(void){
	DDRA = 0xFF;
	TCCR1B |= (1 << CS12) | (1 << CS10);
	TIMSK |= (1 << TOIE1);
	TCNT1 = initial_timer_value;
}

void lcd_string(char *str){
	char data=0;
	while(*str){
		data=*str++;
		lcd_dat(data);
	}
}
```

Для задания 2:




```
#define F_CPU 12000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define RS 2
#define RW 1
#define EN 0

void button_init(void);
void timer_init(void);
void lcd_init(void);

void lcd_com(unsigned char);
```

```
void lcd_dat(unsigned char);
void lcd_string(char*);

uint8_t mode = 0;
uint8_t last_mode = 0;

uint8_t speed_mode = 1;
uint8_t last_speed_mode = 0;

char current_time_char[] = "0 -> 77 : 00 ";
char speed_char[] = "Speed - 1";

uint16_t initial_timer_value = 55000;

uint8_t N = 77;
uint8_t step = 0;


int main(void)
{
        timer_init();
    lcd_init();
    button_init();
    sei();

        lcd_com(0x80);
        lcd_string(current_time_char);
        lcd_com(0xC0);
        lcd_string("press PD2");

    while (1)
    {
                switch (mode)
                {
                        case (0):{
                                if (last_mode == 1) {
                                        last_mode = 0;
                                        lcd_com(0x80);
                                        lcd_string(current_time_char);
                                        lcd_com(0xC0);
                                        lcd_string("Pause    ");
                                }
                                break;
                        }
                        case(1):{
```

```
                                    lcd_com(0x80);
                                    lcd_string(current_time_char);

                                    if (speed_mode != last_speed_mode || last_mode
== 0){
                                        last_speed_mode = speed_mode; last_mode
= 0;
                                        lcd_com(0xC0);
                                        lcd_string(speed_char);
                                    }
                                break;
                        }
                }
    }
}



ISR(INT0_vect){
        last_mode = mode;
        mode = !mode;
};

ISR(INT1_vect){
        if (mode == 0) { return; }


        last_speed_mode = speed_mode;
        speed_mode += 1;
         if (speed_mode > 3){
                speed_mode = 1;
         }

         switch (speed_mode)
         {
                case (3):{ speed_char[8] = '3'; initial_timer_value = 65000; break; }
                case (2):{ speed_char[8] = '2'; initial_timer_value = 60000; break; }
                case (1):{ speed_char[8] = '1'; initial_timer_value = 55000; break; }
        }
};

ISR(TIMER1_OVF_vect)
{
        if (step < N) {
                current_time_char[10] = '0' + step / 10;
```

```
                current_time_char[11] = '0' + step % 10;
                step++;
                } else {
                step = 0;
        }

        TCNT1 = initial_timer_value;
}

void button_init(void){
        DDRD &=~(1<<PD2);
        DDRD &=~(1<<PD3);
        PORTD |=(1<<PD2)|(1<<PD3);
        GICR |= (1<<INT0)|(1<<INT1);
        MCUCR &=~ (1<<ISC01)|(1<<ISC00);
        MCUCR &=~ (1<<ISC11)|(1<<ISC10);
}

void lcd_com(unsigned char p){
        PORTB &= ~(1<<RS);
        PORTB |= (1<<EN);
        PORTC = p;
        _delay_us(500);
        PORTB &=~(1<<EN);
        _delay_us(500);
}

void lcd_dat(unsigned char p){
        PORTB|=(1<<RS)|(1<<EN);
        PORTC=p;
        _delay_us(500);
        PORTB&=~(1<<EN);
        _delay_us(500);
}

void lcd_init(void){
        // Устанавливаем пины RS, RW и EN порта DDRB как выходные
        DDRB |= (1<<RS)|(1<<RW)|(1<<EN);
        PORTB=0x00; // Обнуляем порт B
        DDRC=0xFF; // Устанавливаем порт C как порт вывода
        PORTC=0x00; // Обнуляем порт C
        _delay_us(500);
        lcd_com(0x08); // Инициализация дисплея: выключаем дисплей
        _delay_us(500);
        lcd_com(0x3C); // Установка режима 8 бит данных, 2 строки, 5x8 точек
```

```
        _delay_us(500);
        lcd_com(0x01); // Очистка дисплея
        _delay_us(500);
        lcd_com(0x06); // Установка направления пути записи, увеличение
адреса на 1
        _delay_us(900);
        lcd_com(0x0C); // Включаем дисплей без курсора
}

void timer_init(void){
        DDRA = 0xFF;
        TCCR1B |= (1 << CS12) | (1 << CS10);
        TIMSK |= (1 << TOIE1);
        TCNT1 = initial_timer_value;
}

void lcd_string(char *str){
        char data=0;
        while(*str){
                data=*str++;
                lcd_dat(data);
        }
}
```