

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ»

Кафедра ИиСП

Отчет
по лабораторной работе № 9
по дисциплине «Машинно-зависимые языки программирования»
Вариант 2

Выполнил: ст. гр. ПС-14

Сайфутдинов Э.Р.

Проверил: доцент, доцент
кафедры ИиСП Баев А.А.

г. Йошкар-Ола

2024

Цель работы:

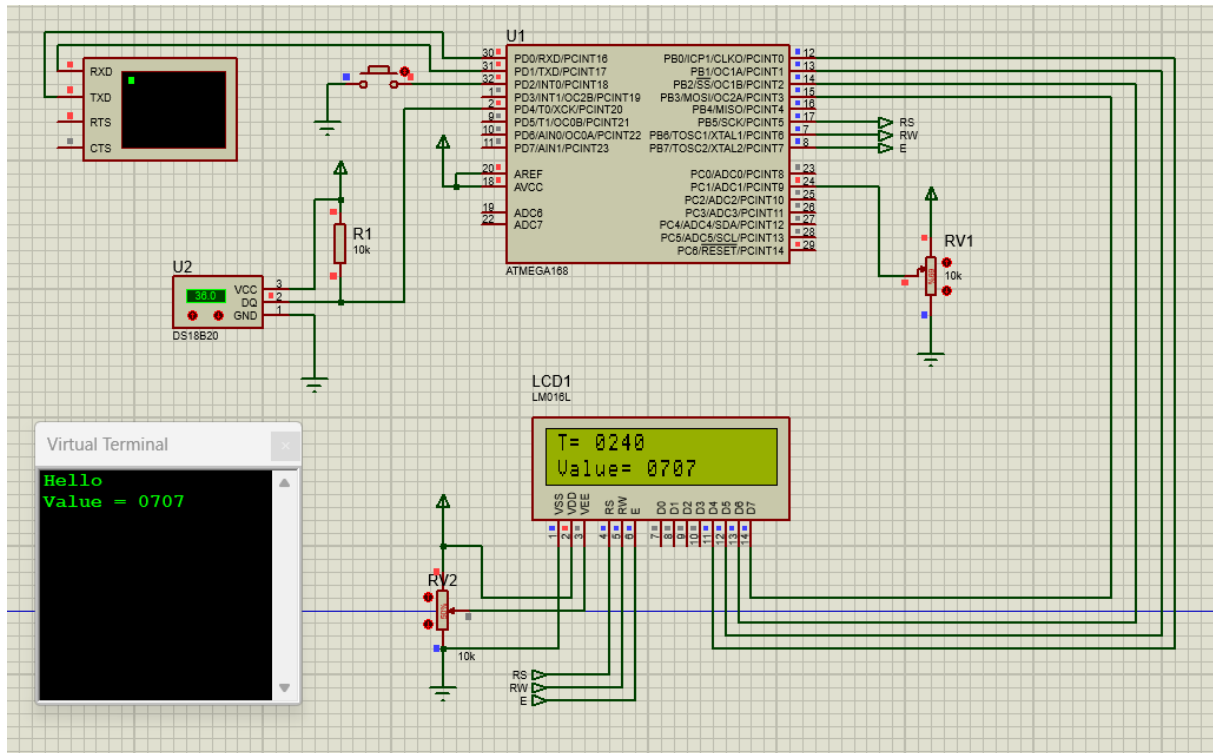
Рассмотреть работу с датчиком DS18B20 по протоколу 1-Wire.

Задания на лабораторную работу:

1. Теоретические сведения

Учебное пособие “Применение микроконтроллеров в радиотехнических и биомедицинских системах”.

2. Практическая часть



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

#define D4 PB0
#define D5 PB1
#define D6 PB2
#define D7 PB3
#define RS PB5
#define RW PB6
#define E PB7
#define CMD 0
#define DATA 1
#define WIRE PD4
#define SET_1_WIRE PORTD |= (1 << WIRE);
#define SET_0_WIRE PORTD &= ~(1 << WIRE);

void InitPorts();
void InitTimer1();
void InitADC();
void InitUSART();
void InitLCD();
void Bin2Dec(uint16_t data);
void DisplayData(uint16_t data);
void SendChar(char symbol);
void SendString(const char *buffer);
void LCD_Write(uint8_t type, char data);
char LCD_Read();
void OneWire_Init(void);
uint8_t OneWire_Read_1_0(void);
uint8_t OneWire_ReadByte(void);
void OneWire_Send_1_0(uint8_t);
void OneWire_SendByte(uint8_t);

volatile uint8_t bcd_buffer[] = {0, 0, 0, 0};
```

```

volatile uint16_t ADC_val, temperature = 0;
const char char_tab[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C',
'D', 'E', 'F'};

int main(void)
{
    InitPorts();
    InitTimer1();
    EIMSK |= (1 << INT0);
    EICRA |= (1 << ISC01);
    InitADC();
    InitUSART();
    InitLCD();
    sei();
    SendString("Hello\r\n");
    LCD_Write(DATA, 'T');
    LCD_Write(DATA, '=');
    LCD_Write(DATA, 0x20);
    LCD_Write(CMD, 0x40 | 0x80);
    LCD_Write(DATA, 'V');
    LCD_Write(DATA, 'a');
    LCD_Write(DATA, 'l');
    LCD_Write(DATA, 'u');
    LCD_Write(DATA, 'e');
    LCD_Write(DATA, '=');
    LCD_Write(DATA, 0x20);
    OneWire_Init();
    OneWire_SendByte(0xCC); // Skip ROM
    OneWire_SendByte(0x4E); // Write scratchpad
    OneWire_SendByte(0x00); // TH
    OneWire_SendByte(0x00); // TL
    OneWire_SendByte(0x3F); // config
    while (1)
    {
        Bin2Dec(ADC_val);
        LCD_Write(CMD, 0x47 | 0x80);
        LCD_Write(DATA, 0x30 + bcd_buffer[3]);
        LCD_Write(DATA, 0x30 + bcd_buffer[2]);
        LCD_Write(DATA, 0x30 + bcd_buffer[1]);
        LCD_Write(DATA, 0x30 + bcd_buffer[0]);
        OneWire_Init();
        OneWire_SendByte(0xCC); // Skip ROM
        OneWire_SendByte(0x44); // Convert T
        while (OneWire_Read_1_0() == 0);
        OneWire_Init();
        OneWire_SendByte(0xCC); // Skip ROM
        OneWire_SendByte(0xBE); // Read Scratchpad
        temperature = 0x00FF & OneWire_ReadByte();
        temperature |= OneWire_ReadByte() << 8;
        LCD_Write(CMD, 0x03 | 0x80);
        LCD_Write(DATA, char_tab[(temperature >> 12) & 0x0F]);
        LCD_Write(DATA, char_tab[(temperature >> 8) & 0x0F]);
        LCD_Write(DATA, char_tab[(temperature >> 4) & 0x0F]);
        LCD_Write(DATA, char_tab[(temperature >> 0) & 0x0F]);
    }
}

//-----
ISR(TIMER1_COMPB_vect)
{
    // DisplayData(0x1E61);
}
ISR(INT0_vect)
{

```

```

        Bin2Dec(ADC_val);
        SendString("Value = ");
        SendChar(0x30 + bcd_buffer[3]);
        SendChar(0x30 + bcd_buffer[2]);
        SendChar(0x30 + bcd_buffer[1]);
        SendChar(0x30 + bcd_buffer[0]);
        SendString("\r\n");
    }
    ISR(ADC_vect)
    {
        ADC_val = ADC;
    }
    ISR(USART_RX_vect)
    {
        if (UDR0 == 0x20)
        {
            SendString("Roger that\r\n");
        }
    }
    //-----
    void InitPorts(void)
    {
        DDRB = 0xFF; // настройка выводов управления дисплеем
        PORTB = 0;
        DDRD = (0 << PD2 | 1 << PD4); // настройка вывода кнопки
        PORTD |= (1 << PD2 | 1 << PD4);
    }
    void InitTimer1(void)
    {
        TCCR1A = 0;
        TCCR1B = (1 << CS11 | 1 << CS10 | 1 << WGM12);
        TCNT1 = 0;
        TIMSK1 |= (1 << OCIE1B);
        OCR1A = 12500;
        OCR1B = 12500;
    }
    void Bin2Dec(uint16_t data)
    {
        bcd_buffer[3] = (uint8_t)(data / 1000);
        data = data % 1000;
        bcd_buffer[2] = (uint8_t)(data / 100);
        data = data % 100;
        bcd_buffer[1] = (uint8_t)(data / 10);
        data = data % 10;
        bcd_buffer[0] = (uint8_t)(data);
    }
    void DisplayData(uint16_t data)
    {
        Bin2Dec(data);
    }
    void InitADC(void)
    {
        ADMUX = (1 << MUX0); // Align left, ADC1
        ADCSRB = (1 << ADTS2 | 1 << ADTS0); // Start on Timer1 COMPB
        ADCSRA = (1 << ADEN | 1 << ADSC | 1 << ADIF); // Enable, auto update, IRQ enable
    }
    void InitUSART()
    {
        UCSR0B = (1 << RXEN0 | 1 << TXEN0 | 1 << RXCIE0);
        UCSR0C = (1 << UCSZ01 | 1 << UCSZ00);
        UBRR0H = 0;
        UBRR0L = 0x67;
    }
    void SendChar(char symbol)
    {

```

```

        while (!(UCSR0A & (1 << UDRE0)))
        ;
        UDR0 = symbol;
    }
void SendString(const char *buffer)
{
    while (*buffer != 0)
    {
        SendChar(*buffer++);
    }
}
void InitLCD(void)
{
    uint8_t BF = 0x80;
    _delay_ms(40);
    PORTB &= ~(1 << RS);
    PORTB = (0x30 >> 4);
    PORTB |= (1 << E);
    asm("nop");
    asm("nop");
    asm("nop");
    PORTB &= ~(1 << E);
    PORTB = 0;
    _delay_ms(5);
    PORTB &= ~(1 << RS);
    PORTB = (0x30 >> 4);
    PORTB |= (1 << E);
    asm("nop");
    asm("nop");
    asm("nop");
    PORTB &= ~(1 << E);
    PORTB = 0;
    _delay_us(150);
    PORTB &= ~(1 << RS);
    PORTB = (0x30 >> 4);
    PORTB |= (1 << E);
    asm("nop");
    asm("nop");
    asm("nop");
    PORTB &= ~(1 << E);
    PORTB = 0;
    _delay_ms(5);
    do
    {
        BF = (0x80 & LCD_Read());
    } while (BF == 0x80);
    PORTB &= ~(1 << RS);
    PORTB = (0x20 >> 4);
    PORTB |= (1 << E);
    asm("nop");
    asm("nop");
    asm("nop");
    PORTB &= ~(1 << E);
    PORTB = 0;
    do
    {
        BF = (0x80 & LCD_Read());
    } while (BF == 0x80);
    // заработал 4-проводной интерфейс
    LCD_Write(CMD, 0x28); // 2 строки, 5*8
    LCD_Write(CMD, 0x0C); // display on, cursor on
    LCD_Write(CMD, 0x06); // cnt++, shift enabled
}
void LCD_Write(uint8_t type, char data)
{

```

```

uint8_t BF = 0x80;
do
{
    BF = 0x80 & LCD_Read();
} while (BF == 0x80);
PORTB |= (type << RS);
PORTB |= (1 << E);
PORTB &= ~(0x0F);
PORTB |= (0x0F & (data >> 4)); // старшая тетрада
PORTB &= ~(1 << E);
asm("nop");
asm("nop");
asm("nop");
PORTB |= (1 << E);
PORTB &= ~(0x0F);
PORTB |= (0x0F & data); // младшая тетрада
PORTB &= ~(1 << E);
PORTB = 0;
}
char LCD_Read(void)
{
    char retval = 0;
    PORTB &= ~(1 << RS);
    PORTB |= (1 << RW);
    DDRB &= ~(1 << D4 | 1 << D5 | 1 << D6 | 1 << D7);
    PORTB |= (1 << E);
    asm("nop");
    asm("nop");
    retval = (PINB & 0x0F) << 4;
    PORTB &= ~(1 << E);
    asm("nop");
    asm("nop");
    asm("nop");
    PORTB |= (1 << E);
    asm("nop");
    asm("nop");
    retval |= (PINB & 0x0F);
    PORTB &= ~(1 << E);
    DDRB |= (1 << D4 | 1 << D5 | 1 << D6 | 1 << D7);
    PORTB = 0;
    return retval;
}

void OneWire_Init(void)
{
    DDRD |= (1 << WIRE); // шину на выход
    SET_0_WIRE;          // установить 0
    _delay_us(500);       // ждать не менее 480 мкс
    SET_1_WIRE;
    DDRD &= ~(1 << WIRE); // шину на вход
    _delay_us(60);
    if ((PIND & 1 << WIRE) == 0)
    { // ожидается импульс сброса
        while ((PIND & 1 << WIRE) == 0)
        {
            _delay_us(420);
        }
    }
    else
    {
        SendString("No response\r\n");
    }
}

void OneWire_Send_1_0(uint8_t bit)
{
    _delay_us(2);
}

```



```

        DDRD |= (1 << WIRE);
        SET_0_WIRE;
        _delay_us(10);
        if (bit != 0)
        {
            SET_1_WIRE;
            _delay_us(90);
        }
        else
        {
            SET_0_WIRE;
            _delay_us(90);
            SET_1_WIRE
        }
    }
}

uint8_t OneWire_Read_1_0(void)
{
    uint8_t bit = 0;
    _delay_us(2);
    DDRD |= (1 << WIRE);
    SET_0_WIRE;
    _delay_us(2);
    SET_1_WIRE;
    DDRD &= ~(1 << WIRE);
    _delay_us(2);
    if ((PIND & 1 << WIRE) == 0)
    {
        bit = 0;
    }
    else
    {
        bit = 1;
    }
    _delay_us(90);
    return bit;
}

void OneWire_SendByte(uint8_t data)
{
    for (uint8_t i = 0; i < 8; i++)
    {
        OneWire_Send_1_0(0x01 & data);
        data = data >> 1;
    }
}

uint8_t OneWire_ReadByte(void)
{
    uint8_t retval = 0;
    for (uint8_t i = 0; i < 8; i++)
    {
        retval |= OneWire_Read_1_0() << i;
    }
    return retval;
}

```

Вывод:

Таким образом, рассмотрена работа с датчиком DS18B20 по протоколу 1-Wire.