МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
КАФЕДРА САУ

ОТЧЕТ

по Лабораторной работе №6

по дисциплине «МИКРОПРОЦЕССОРНАЯ ТЕХНИКА В МЕХАТРОНИКЕ И РОБОТОТЕХНИКЕ»

тема: Работа с датчиком температуры по протоколу I2C

Студенты гр. 6492 _____ Мурашко А.С.
_____ Огурецкий Д.В.
_____ Спорыш И.В.

Преподаватель _____ Девяткин А.В.

Санкт-Петербург
2019

Цель работы — Изучение работы протокола I2C и работы с датчиком температуры Ds1631.

Задание на лабораторную работу — Написать программу вывода температуры с датчика на символьный дисплей

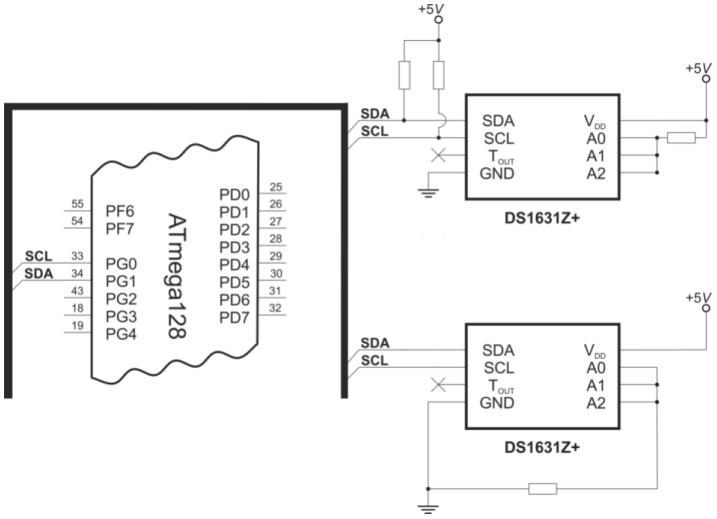

Рис. 1 Схема подключения датчиков температуры

Выполнение работы

Описание функций библиотеки, реализующей работу I2C

Основной особенность: для получения низкого уровня на линии используется замыкание линии на землю, а для получения высокого уровня используется внешняя подтяжка к питанию, то для генерации этих уровней используется не регистр PORTx, а регистр DDRx, который переводит ножку контроллера из состояния входа (высокого сопротивления) в состояние выхода с низким уровнем.

Код

```
#define SDA 1
#define SCL 0
#define DDRX_I2C DDRG
#define HIGH(pin) (DDRX_I2C &= (~_BV(pin)))
#define LOW(pin) (DDRX_I2C |= _BV(pin))
#define PULSE 50 //us время импульса, определяет скорость передачи

void I2C_Start(void)
{
    HIGH(SDA);
    HIGH(SCL);
    delay_us(PULSE);
    LOW(SDA);
    delay_us(PULSE);
}
```

```
16
17 void I2C_SendByte(unsigned char data)
18 {    //время на передачу 1 бита 3*PULSE  1 байт 24*PULSE
19     unsigned char i;
20     LOW(SCL);
21     delay_us(PULSE);
22     for(i=(1<<7);i>0;i=(i>>1))
23     {
24         if(data & i){HIGH(SDA);}else{LOW(SDA);}
25         delay_us(PULSE);
26         HIGH(SCL);
27         delay_us(PULSE);
28         LOW(SCL);
29         delay_us(PULSE);
30     }
31         LOW(SDA); //чтобы при подтверждении не произошла отправка стоп-со-
32 стояния
33         delay_us(PULSE);  //ждем пока ACK бит ведомого станет 1
34         //подтверждаем ACK бит
35         HIGH(SCL);
36         delay_us(PULSE);
37         LOW(SCL);
38         delay_us(PULSE);
39 }
40
41 void I2C_Stop(void) { HIGH(SCL); delay_us(PULSE); HIGH(SDA); de-
42 lay_us(PULSE); }
43
44 unsigned char I2C_ReadByte(unsigned char ack)
45 {
46    unsigned char i,readbyte =  0;
47    HIGH(SDA);
48    for(i=(1<<7) ; i > 0 ; i = (i >>1))
49    {
50         HIGH(SCL);
51         delay_us(PULSE);
52         if(BIT_IS_SET(PING,SDA))
53         readbyte |= i;
54         LOW(SCL);
55         delay_us(PULSE);
56    }
57    if(ack){HIGH(SDA);}else{LOW(SDA);}  //отправляем ACK-бит
58    delay_us(PULSE);
59    HIGH(SCL);
60    delay_us(PULSE);
61    LOW(SCL);
62    delay_us(PULSE);
63    //для окончания приема
64    LOW(SDA);
65    delay_us(PULSE);
      return readbyte;
   }
```

Описание функций библиотеки, реализующей работу LCD дисплея

Код

```
#include <io.h>
#include <mega128a.h>
#include <delay.h>

#define CMD 1
#define DATA 0
#define DISPLAY 1
#define CURSOR 0
#define RHIGHT 1
#define LEFT   0
#define ON 1
#define OFF 0
#define EIGHT 1
#define FOUR 0
#define ONE 0
#define TWO 1
#define LOWERCASE  0
#define UPPERCASE  1
#define RS 7 // выбор регистра
#define E 6 // строб передачи

//установка адреса DDRAM (позиция символа)
//pos = 0 ... 39
//line = 0 — first line
//line = 1 — second line
void LCD_pos(char pos, char line)
{
   LCD_message(((1<<7)|(pos+line*64)),CMD);
}

//Вывод числового значения (max 5 десятичных разрядов и больше нуля)
void LCD_uint(int value)
{
    int i ;
    unsigned char flag_first_num = 0 ;
    unsigned char number;
    if (value<0) LCD_message(0b10010110,DATA);//выводим минус
    for(i=1; i<=5; i++)
    {
        number = Digit(value,i);
        if(number != 0)    //появление перового символа
        {
           flag_first_num = 1;
        }

        if(flag_first_num == 1)
        {

            LCD_message(number+'0',DATA);//выводим цифру
        }

    }
```

```
53 }
54
55 void LCD_message(char message,int type)
56 {
57     //[]-----------------------------------------------[]
58     //| Назначение: запись кодов в регистр команд ЖКИ |
59     //| Входные параметры: message - сообщение |
60     //| Входные параметры: type - тип сообщения (код или данные) |
61     //[]-----------------------------------------------[]
62
63     if(type) // 1
64     PORTD &= ~(1<<RS); // выбор регистра команд RS=0
65     else
66     PORTD |= (1<<RS); // выбор регистра данных RS=1
67     PORTC=message; // записать команду в порт PORTC
68     PORTD|= (1<<E); // \ сформироватьна
69     delay_us(5); // |выводе E строб 1-0
70     PORTD&= ~(1<<E); // / передачи команды
71     delay_ms(3); // задержка для завершения записи
72
73 }
74
75 void LCD_init(void)
76 {
77     //[]-----------------------------------------------[]
78     //| Назначение: инициализация ЖКИ |
79     //[]-----------------------------------------------[]
80     DDRC = 0xFF;            // все разряды PORTC на выход
81     DDRD|= ((1<<E)|(1<<RS));// разряды PORTD на выход
82     delay_ms (100); // задержка для установления
83                     // напряжения питания
84     LCD_message(0x30,CMD); // \ вывод
85     LCD_message(0x30,CMD); //| трех
86     LCD_message(0x30,CMD); // / команд 0x30
87     LCD_message(0x38,CMD); //8 разр.шина, 2 строки, 5 ? 7 точек
88     LCD_message(0x0E,CMD); // включить ЖКИ и курсор, без мерцания
89     LCD_message(0x06,CMD); //инкремент курсора, без сдвига экрана
90     LCD_message(0x01,CMD); // очистить экран, курсор в начало
91 }
92
93 //вывод строковой информации на дисплей
94 void LCD_string(unsigned char* str)
95 {
96     while(*str != '\0')
97     {
98         LCD_message(Code(*str++),DATA);
99         //*str извлечение элемента по адресу в указателе str
100        //str++ увеличение значение указателя на 1 (т.е. переход к следу-
101 ющему элементу массива)
102    }
103 }
104
105 unsigned char Digit (unsigned int d, unsigned char m)
106 {
107    //[]-----------------------------------------------[]
108    //| Назначение: выделение цифр из разрядов пятиразрядного |
109    //| десятичного положительного числа |
110    //| Входные параметры: |
111    //| d - целое десятичное положительное число |
```

```
112     //| m - номер разряда (от 1 до 5, слева направо) |
113     //| Функция возвращает значение цифры в разряде m числа d |
114     //[]-----------------------------------------------------[]
115     unsigned char i = 5, a;
116     while(i)
117     {
118         a = d%10; //если d < 0 то a<0
119         if(i-- == m) break;
120         d /= 10;
121     }
122     return(a);
123 }
124
125 unsigned char Code(unsigned char symb)
126 {
127     //[]-----------------------------------------------[]
128     //| Назначение: перекодировка символов кириллицы |
129     //| Входные параметры: symb - символ ASCII |
130     //| Функция возвращает код отображения символа |
131     //[]-----------------------------------------------[]
132       unsigned char TabCon[] =
133     {0x41,0xA0,0x42,0xA1,0xE0,0x45,0xA3,0xA4,0xA5,0xA6,0x4B,
134 0xA7,0x4D,0x48,0x4F,0xA8,0x50,0x43,0x54,0xA9,0xAA,0x58,
135     0xE1,0xAB,0xAC,0xE2,0xAD,0xAE,0x62,0xAF,0xB0,0xB1,0x61,
136     0xB2,0xB3,0xB4,0xE3,0x65,0xB6,0xB7,0xB8,0xB9,0xBA,0xBB,
137     0xBC,0xBD,0x6F,0xBE,0x70,0x63,0xBF,0x79,0x5C,0x78,0xE5,
138     0xC0,0xC1,0xE6,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7};//коды символов в ки-
139 риллице
140     return (symb >= 192 ? TabCon[symb-192]: symb);
141 }
142
143 //очистить дисплей и установить курсор в нулевую позицию(адрес 0 )
144 void clear_display()
145 {
146     LCD_message(1,CMD);
147 };
148
149 //установить курсор в нулевую позицию, дисплей относительно буфера DDRAM
150 в начальную позицию
151 void default_display()
152 {
153     LCD_message(2,CMD);
154 };
155
156 //установить направление сдвига курсора при записи кода в DDRAM
157 // и разрешить(запретить) сдвиг окна вместе с курсором
158 // ID = RIGHT
159 // ID = LEFT
160 // S = ON разрешить сдвиг окна вместе с курсором
161 // S = OFF запретить сдвиг окна вместе с курсором
162 void shift_direction(int ID,int S)
163 {
164   LCD_message((1<<2)|(ID<<1)|S,CMD);
165 }
166
167 //Включить(выключить) индикатор,зажечь(погасить) курсор.Сделать курсор
168 мигающим
169 // B = ON  курсор мигает
170 // B = OFF курсор не  мигает
```

```
171 // D = ON  включить индикатор
172 // D = OFF выключить индикатор
173 // C = ON  зажечь курсор
174 // C = OFF погасить курсор
175 void switch_display(int B,int D,int C)
176 {
177     LCD_message((1<<3)|(D<<2)|(C<<1)|B,CMD);
178 }
179
180 //сдвиг курсора или дисплея вправо или влево
181 // choose = DISPLAY
182 // choose = CURSOR
183 // direction = RIGHT
184 // direction = LEFT
185 void shift(int choose,int direction)
186 {
187     LCD_message((1<<4)|(choose<<3)|(direction<<2),CMD);
188 }
189
190 //установить разрядность шины данных, количество строк, шрифт
191 //data_bus_width = EIGHT 8 бит
192 //data_bus_width = FOUR 4 бит
193 //line_number = ONE 1 строка
194 //line_number = TWO 2 строки
195 //font = LOWERCASE  строчные
    ////font = UPPERCASE  заглавные
    void display_setting(int data_bus_width,int line_number,int font)
    {

    LCD_message((1<<5)|(data_bus_width<<4)|(line_number<<3)|(font<<2),CMD);
    }
```

Описание функций по работе с датчиком и main()

Работа программы подробно описана в блок-схеме.

Код

```
1  /*
2   * Created: 11.09.2019 11:46:22
3   * Author: Student
4   * lab6
5   * Variant 8
6   *
7   */
8
9  #include <define.h>
10
11 // Коды команд
12 #define START_CONVERT_T 0x51     //байт управления с битом записи,
13 #define READ_TEMPERATURE 0xAA    // байт управления с битом записи
14 #define ACCESS_CONFIG 0xAC       // байт  управления с битом записи
15 // Биты регистра конфигурации
16 #define R1 3   //разрядность предоставляемого результата и
17 //время единичного измерения
18 #define R0 2
19 #define SHOT 0   // режим одиночного измерения. Если этот
20 //бит сброшен, то датчик работает в режиме постоянного измерения.
```

```
#define W 0     //запись
#define R 1     //чтение

// Запуск измерения температуры
void startConvert(unsigned char address)
{
  I2C_Start();
  I2C_SendByte( 0b10010000 |(address<<1) | W);
  I2C_SendByte(START_CONVERT_T);
  I2C_Stop();
}

// Функция чтения температуры с датчика
int readTemperature(unsigned char address)
{
  int result;
  I2C_Start();
  I2C_SendByte( 0b10010000 |(address<<1) | W);
  I2C_SendByte(READ_TEMPERATURE);
  I2C_Start();
  I2C_SendByte( 0b10010000 |(address<<1) | R);
  result = I2C_ReadByte(0);
  result <<= 8;
  result += I2C_ReadByte(1);
  I2C_Stop();
  return result/256;
}

// Настройка датчика температуры на одиночное
// измерение и 9-и битный результат
void setDs1631(unsigned char address)
{
  I2C_Start();
  I2C_SendByte(  0b10010000 |(address<<1) | W);  //
  I2C_SendByte(ACCESS_CONFIG);
  I2C_SendByte(_BV(SHOT));
  I2C_Stop();
}

void main(void)
{
    LCD_init();
    LCD_string("temp(0) = ");
    LCD_pos(0,1); //перенос курсора в 0 позицию второй строки
    LCD_string("temp(7) = ");
    while(1)
    {
        setDs1631(0);
        startConvert(0);
        //вывод результата на дисплей
        LCD_pos(10,0);
        LCD_uint(readTemperature(0));
        setDs1631(7);
        startConvert(7);
        //вывод результата на дисплей
        LCD_pos(10,1);
        LCD_uint(readTemperature(7));
        delay_ms(200);
    }
```

```
80 }
```

Блок-схема

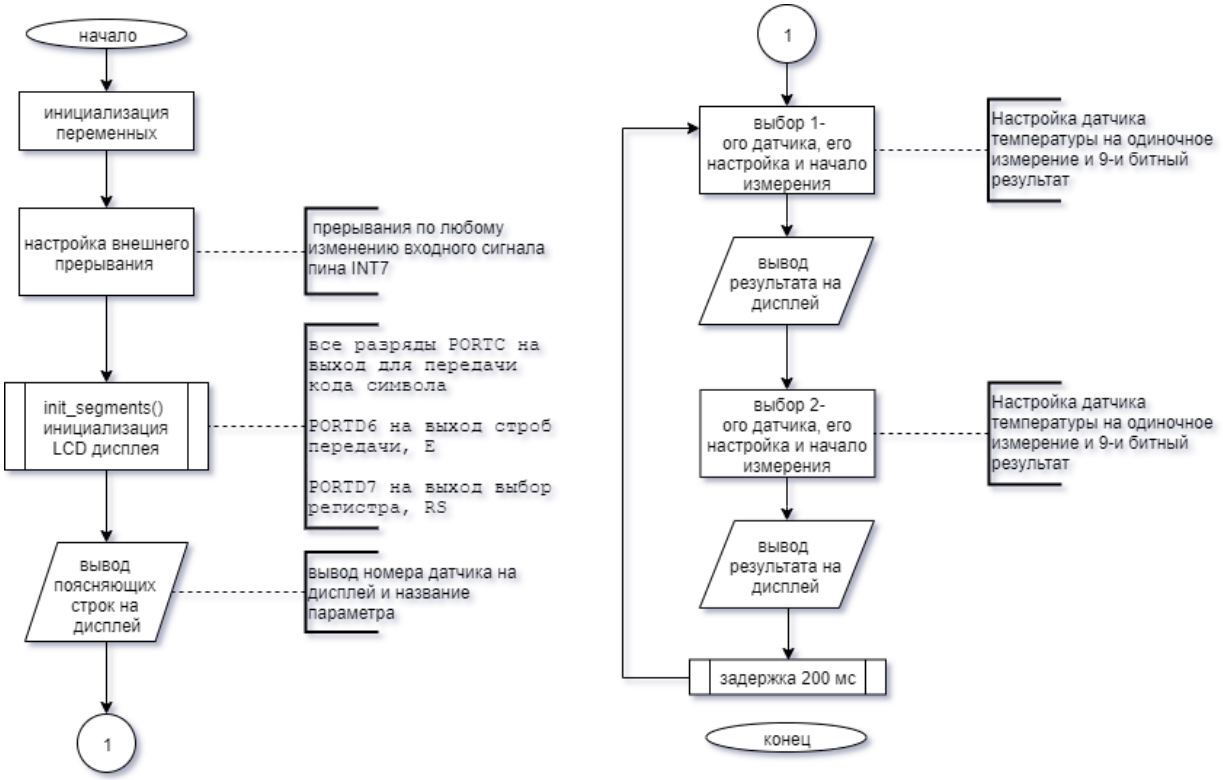

Рис. 5 Блок-схема main()

Вывод: с помощью микроконтроллера можно управлять дисплеем и считывать показания с датчика температуры. Таким образом можно быстро создать простую систему умного дома.