МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«МИРЭА - Российский технологический университет»

РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения
Кафедра КБ-6 «Приборы и информационно-измерительные системы»

КУРСОВАЯ РАБОТА

по дисциплине Методы и средства автоматизации проектирования интеллектуальных измерительных устройств

Тема курсовой работы Разработка и отладка встраиваемого программного обеспечения интеллектуального измерительного прибора

Студента _____ В.Д.Бонвеч
дата, подпись — инициалы и фамилия

Группа БПБО-02-20 шифр 20Б0920

Обозначение работы КР–02068717–12.03.01–КБ-6–19–21

Работа защищена на оценку _____

Руководитель курсовой работы _____ С.А. Канаев
дата, подпись — инициалы и фамилия

Члены комиссии _____ О.В. Москаленко
подпись — инициалы и фамилия

_____
подпись — инициалы и фамилия

Работа представлена к защите «___» _____20__ г.

Допущен к защите «___» _____20__ г.

Москва 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "МИРЭА - РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ"

ИНСТИТУТ КОМПЛЕКСНОЙ БЕЗОПАСНОСТИ И СПЕЦИАЛЬНОГО ПРИБОРОСТРОЕНИЯ
КАФЕДРА КБ-6 «ПРИБОРЫ И ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫЕ СИСТЕМЫ»

УТВЕРЖДАЮ

Заведующий кафедрой __КБ-6__

_____ /А.Б. Снедков/
подпись ФИО

«___»_____ 20___г.

ЗАДАНИЕ

на выполнение курсовой работы по дисциплине

“ Методы и средства автоматизации проектирования интеллектуальных измерительных устройств ”

Студент Бонвеч Виктор Дмитриевич
Ф.И.О.

шифр студенческого билета № 20Б0920 группа БПБО-02-20

1. Тема: Разработка и отладка встраиваемого программного обеспечения интеллектуального измерительного прибора

2. Срок сдачи студентом законченной работы **27 мая 2021г.**

3. Исходные данные для проектирования _Вариант №49, Разработать управляющую программу на языке Си для устройства на базе микроконтроллера с ядром AVR. Тип микроконтроллера ATmega32. Базовая платформа - EasyAVR v7 Development System фирмы mikroElektronika. Наименование: цифровой манометр. Отображаемый параметр: давление. На вход аналого-цифрового преобразователя (АЦП), встроенного в микроконтроллер базовой платформы, подается напряжение, которое формируется схемой включения тензометрического датчика давления. При изменении измеряемого параметра в диапазоне от 0 до 400 кПа, выходное напряжение линейно изменяется в диапазоне от 0 до 5 В. Выход преобразователя подключен к входному каналу АЦП базовой платформы (используется встроенный в МК АЦП). Тип индикатора - АЦЖКИ (2Х16). С помощью кнопочных переключателей реализовать переключение единиц измерений (Па, кПа, МПа). Формат отображения результата измерений на индикатор выбрать самостоятельно. Измеряемый параметр отображать с

разрешающей способностью не хуже 0,1 кПа. Количества кнопочных переключателей не менее 3.

4. Перечень листов графического материала:

Руководитель работы _____ / Канаев С.А. /
подпись Ф.И.О.

Задание принял к исполнению **25 марта 2021г.**

Студент _____ / Бонвеч В.Д. /
подпись Ф.И.О.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью курсового проектирования является разработка управляющей программы на языке Си для устройства на базе микроконтроллера с ядром AVR. Тип микроконтроллера ATmega32. Базовая платформа - EasyAVR v7 фирмы mikroElektronika.

Наименование: цифровой манометр.

Отображаемый параметр: давление.

На вход аналого-цифрового преобразователя (АЦП), встроенного в микроконтроллер базовой платформы, подается напряжение, которое формируется схемой включения тензометрического датчика давления. При изменении измеряемого параметра в диапазоне от 0 до 400 кПа, выходное напряжение линейно изменяется в диапазоне от 0 до 5 В. Отображение значения давления и единицы измерения будет осуществляться на экране АЦЖКИ 2Х16. С помощью кнопочных переключателей реализовать переключение единиц измерений (Па, кПа, МПа).

Измеряемый параметр отображать с разрешающей способностью не хуже 0,1 кПа. Количества кнопочных переключателей не менее 3.

Использованный пакет для написания программы - Atmel Studio 7.0

1 ХАРАКТЕРИСТИКА УСТРОЙСТВА И ПРИНЦИП РАБОТЫ

В данном разделе будет дана краткая характеристика устройства и алгоритм его работы.

1.1 Описание принципа работы устройства

Манометр спроектирован на базовой платформе EasyAVR v7 Development System фирмы mikroElektronika [1], на которой расположены: микроконтроллер, манометр, АЦЖКИ (2Х16).

Структурная схема устройства представлена на рисунке 1.1.

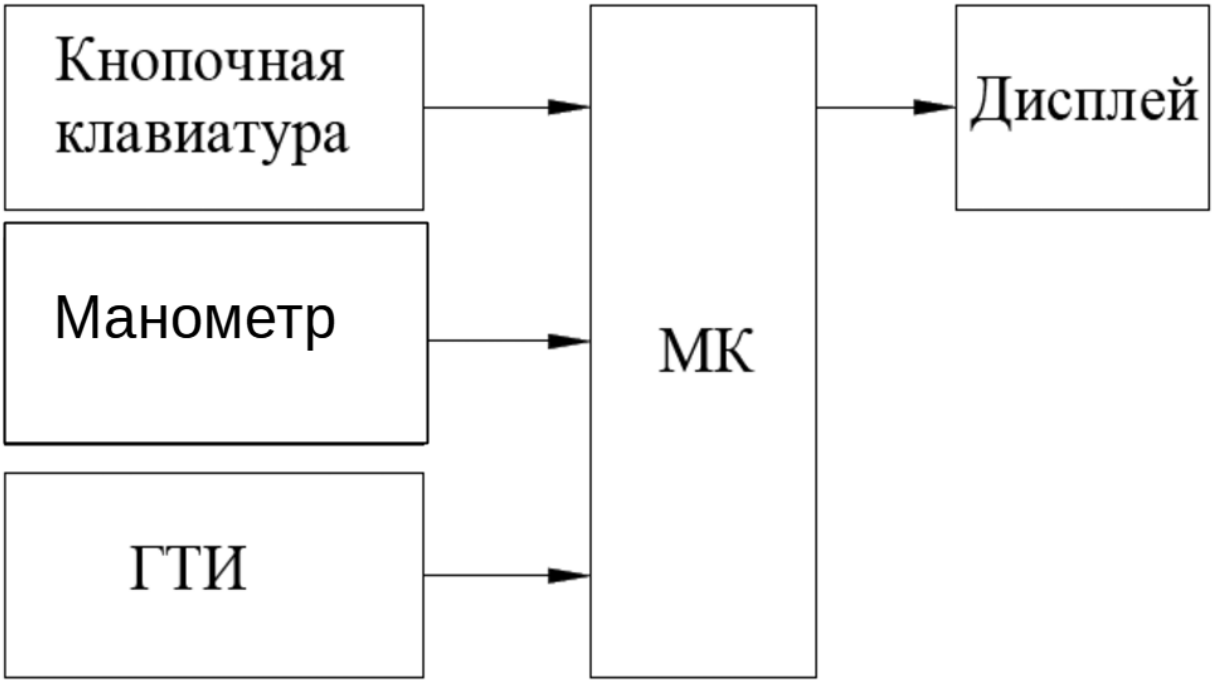

Рисунок 1.1 – Структурная схема устройства

Согласно структурной схеме устройства, принцип работы заключается в следующем: путем изменения сопротивления манометра, изменяется напряжение поступающее на вывод АЦП встроенного в микроконтроллер, который

преобразует напряжение, поступающее на его вход, в эквивалентное значение двоичного кода. Затем микроконтроллер преобразует значение, полученное с АЦП, в значение давления и отображает его на дисплее. При необходимо изменения единицы измерения отображаемой на дисплее, необходимо нажать на кнопочный переключатель, установленный на кнопочной клавиатуре. ГТИ (генератор тактовых импульсов) необходим для работы микроконтроллера (МК).

Состав устройства:

- микроконтроллер – Atmega32 [1];
- манометр;
- дисплей - АЦЖКИ 2Х16 [2];
- кнопочная клавиатура.

1.2 Описание компонентов устройства

В данном разделе будет приведены краткая характеристика компонентов устройства и схемы их подключения к микроконтроллеру.

1.2.1 Микроконтроллер ATmega 32

Atmega32 является КМОП 8-битным микроконтроллером построенным на расширенной AVR RISC архитектуре. Используя команды исполняемые за один машинный такт, контроллер достигает производительности в 1 MIPS на рабочей частоте 1 МГц, что позволяет разработчику эффективно оптимизировать потребление энергии за счёт выбора оптимальной производительности.

AVR ядро сочетает расширенный набор команд с 32 рабочими регистрами общего назначения. Все 32 регистра соединены с АЛУ, что обеспечивает доступ к двум независимым регистрам на время исполнения

команды за один машинный такт. Благодаря выбранной архитектуре достигнута наивысшая скорость кода и соответственно высокая производительность в 10 раз превосходящая скорость соответствующего CISC микроконтроллера.ATmega32 содержит 32Кбайт внутрисистемной программируемой FLASH памяти программ, допускающей чтение во время записи, 1024 байт EEPROM, 2К байт SRAM , 32 рабочих регистра, JTAG интерфейс сканирования внутренних регистров, встроенную систему отладки и программирования, три гибких таймера- счётчика с модулем сравнения, внутренние и внешние прерывания, последовательный программируемый интерфейс USART, байт-ориентированный двухпроводный последовательный интерфейс, 8-и канальный, 10-и битный АЦП с дифференциальным программируемым усилителем (только для TQFP), программируемый Watchdog таймер с внутренним генератором, порт SPI и шестью режимами сбережения энергии.

1.2.2 АЦЖКИ 2Х16

В качестве АЦЖКИ используется модуль WH1602B2-TMI-ET. Данный тип индикатора имеет 2 строки по 16 символов, что позволяет выводить до 32 символов одновременно. Вывод необходимой информации на АЦЖКИ 2Х16 осуществляется с помощью трех функций:

- инициализация индикатора;
- установка положения «курсора»;
- запись символа на индикатор.

Для передачи данных используются старшие четыре вывода PC4, PC5, PC6, PC7 порта С микроконтроллера. Для сброса дисплея используется вывод PA2. Для включения дисплея используется вывод PD6. Для регулировки контрастности дисплея используется потенциометр P2.

Схема подключения АЦЖКИ 2Х16 представлена на рисунке 1.2.

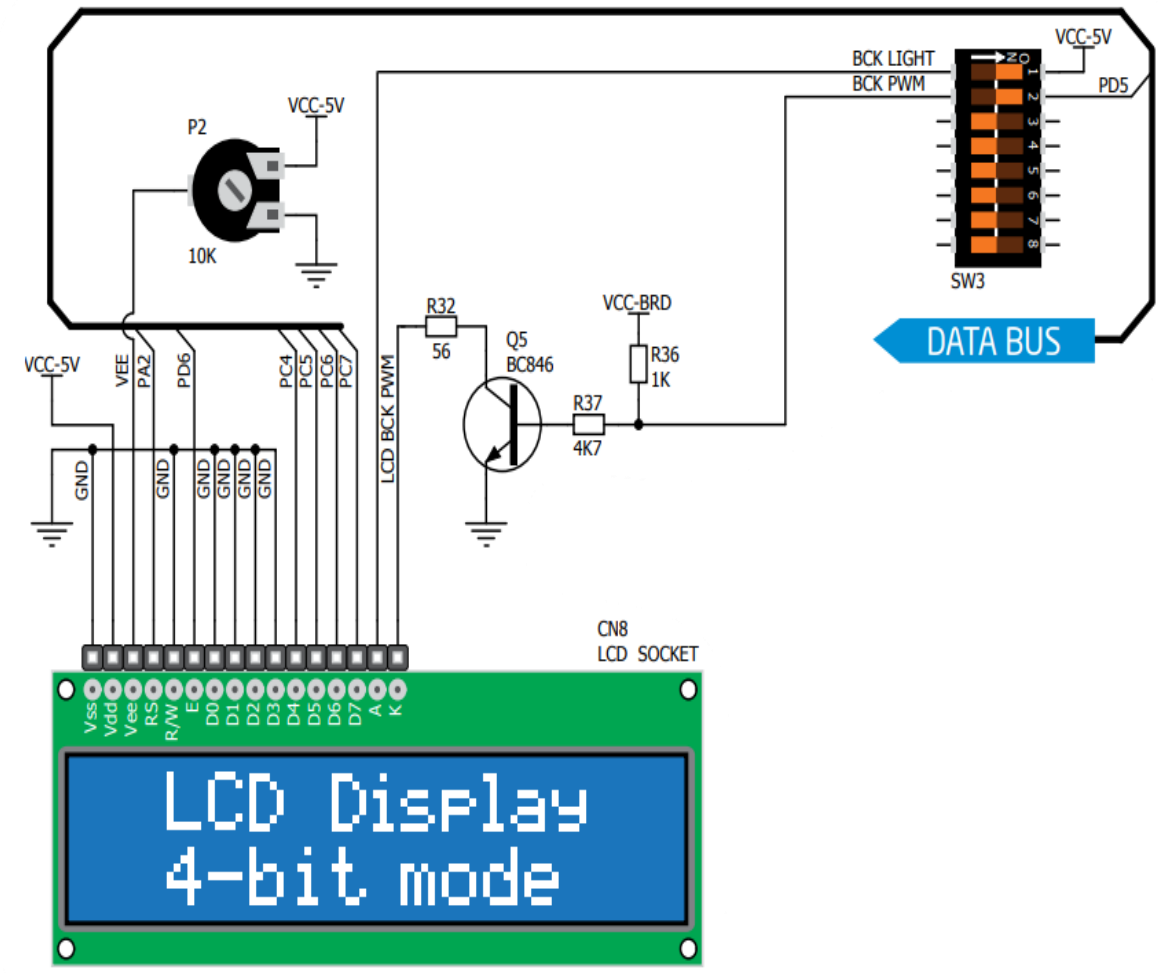

Рисунок 1.2 – Схема подключения АЦЖКИ 2X16

Назначение выводов разъема, к которому подключается АЦЖКИ 2X16, представлено в таблице 1.

Таблица 1 – Назначение выводов разъема CN8

Наименование	Назначение
GND	Линия «земли»
Vdd	Линия питания 5 В
Vee	Линия регулировки контрастности
E	Линия включения дисплея
D4-D7	Линии данных
A	Линия подсветки (анод)
K	Линия подсветки (катод)

1.2.3 Манометр

Манометр представляет из себя схему резистивного делителя, который необходим для обеспечения изменения напряжения на входе АЦП микроконтроллера ATmega32 в диапазоне от 0 до 5В. Значения напряжение, поступающее от манометра на вход АЦП, преобразуется в двоичный код. Схема подключения потенциометра представлена на рисунке 1.3.

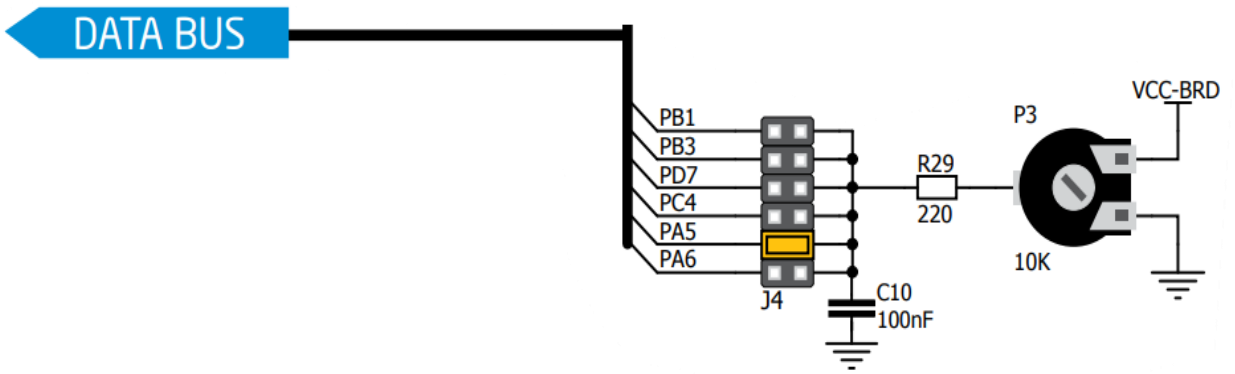

Рисунок 1.3 - Схема подключения манометра

1.2.4 Генератор тактовых импульсов

В базовой платформе EasyAVR v7 Development System фирмы mikroElektronika для синхронизации процессов используется внешний генератор тактовых импульсов с частотой 8 МГц.

1.2.5 Кнопочная панель управления

Кнопочная панель управления содержит кнопочные переключатели и позволяет пользователю управлять устройством. В проектируемом устройстве используется 3 кнопочных переключателя. Влияние дребезга контактов уменьшено программным способом.

2 РАЗРАБОТКА И ОТЛАДКА ВСТРОЕННОГО ПО

В данном разделе приводится описание принципа работы программного обеспечения устройства.

2.1 Настройка выводов АЦЖКИ

```
#define RS 2
#define E 6
```

Рисунок 2.1 – Настройка выводов АЦЖКИ

Вывод RS отвечает за выбор строки, вывод E отвечает за линию действующего дисплея

2.2 Инициализация

На рисунке 2.2 показан фрагмент кода, который отвечает за инициализацию функций.

```
asm ("cli");          //Команда ассемблера "запрет прерываний"
init_PORT();
timer_init();
timer1_init();
init_adc();
init_lcd();
lcd_com(0x81);     //Установка курсора в позицию: строка 1 знакоместо 1
write_str("  R=00000kOhm"); //Вывод названия прибора
asm ("sei");
```

Рисунок 2.2 – Инициализация

2.3 Функция “init_PORT”

Данная функция отвечает за инициализацию и настройку портов устройства. На рисунке 2.3 изображен код функции.

```
Void init_PORT()
{
        PORTA=0b00000000;
        DDRA=0b00000100; /* Data Direction Register;  настройка
линий порта A;  старшие линии порта A работают на ввод */
        PORTC=0x00;
        DDRC=0xff;
        PORTB=0x00;
        DDRB=0xff;
        PORTD=0x00;
        DDRD=0b01000000;
}
```

Рисунок 2.3 – Функция “init_PORT”

2.4 Функция “timer1_init”

Данная функция отвечает за настройку таймера, а также выставления оптимальной частоты срабатывания таймера. Источником тактового сигнала для таймера/счетчика T1 может быть как тактовый сигнал используемый для всего микроконтроллера с использованием предделителя, так и сигнал, поступающий на вход T1 (11 ножка). Если не выбрано ни одного источника тактового сигнала, то таймер/счетчик останавливается.[7] Характеристики таймера/счетчика T1 представлены на рисунке 2.4. Регистры таймера/счетчика T1 обозначены на рисунке 2.5. Код функции “timer1_init” показан на рисунке 2.6.

Два независимых выхода по совпадению

Таймер сброса при совпадении

Один вход захвата

Блок шумоподавления входа захвата

Изменяемый период ШИМ сигнала

Фазовый корректор ШИМ сигнала

Изменяемый период ШИМ сигнала

Тактовый генератор

Три независимых источника прерывания

Рисунок 2.4 – Характеристики таймера/счетчика

TCNT1 - счетный регистр таймера/счетчика T1 (16 бит)

OCR1A - регистр сравнения A (16 бит)

OCR1B - регистр сравнения B (16 бит)

TIMSK1 - регистр маски прерываний для таймера/счетчика T1

TIFR1 - регистр флагов прерываний для таймера/счетчика T1

TCCR1A - регистр управления A

TCCR1B - регистр управления B

TCCR1C - регистр управления C

ICR1 - регистр захвата (16 бит)

Рисунок 2.5 – Регистры таймера/счетчика

```
void timer1_init(void)    {
        TCNT1=0x0000;
        OCR1A=31250; //   число 31250
        TCCR1A=0b00000000;// режим CTC
        TCCR1B=0b00001100;// делитель на 256
        TIFR=0b00111100;
        TIFR=0b000010010;
        TIMSK=0b00010010;}
```

Рисунок 2.6 – Код функции

2.5 Функция “timer_init”

Как и предыдущая функция, эта функция отвечает за настройку таймера[8]. В данной работе этот таймер не несёт никакой функции, однако инициализирован для возможности добавления новых возможностей в проект. Код функции представлен на рисунке 2.7.

```
VOID TIMER_INIT(VOID)    {
        TCNT0=0B00000000;
        OCR0=49; //  ЧИСЛО 50
        TCCR0=0B00001100;// ДЕЛИТЕЛЬ НА 256
        TIFR=0B00000011;
}
```

Функция 2.7 – Функция “timer_init”

2.6 Манометр

В таймере1 находится код, отвечающий за корректную работу манометра. ADCSRA - регистр управления АЦП A. Код функции обработчика прерываний представлен на рисунках 2.8-2.11.

```
        ISR (TIMER1_COMPA_vect)
        {
                ADCSRA=(0b10000111|0b01000000);
                temp_ADC=ADCSRA;
                while((temp_ADC&0b01000000)!=0)
                {
                        temp_ADC=ADCSRA;
                }
```

Рисунок 2.8 – Таймер1

```
code_ADC=0;
code_ADC=ADCL;
temp_ADC=ADCH;
code_ADC=code_ADC|(temp_ADC<<8);
temp_code_ADC=code_ADC*391.006843; //линейная зависимость
temp_time=temp_code_ADC;
```

Рисунок 2.9 – Таймер1

```
lcd[0]=temp_time%10+48; // смещение на 48 в соответствии с таблицей кодировки АЦЖКИ
temp_time=temp_time/10;
lcd[1]=temp_time%10+48;
temp_time=temp_time/10;
lcd[2]=temp_time%10+48;
temp_time=temp_time/10;
lcd[3]=temp_time%10+48;
temp_time=temp_time/10;
lcd[4]=temp_time%10+48;
lcd[5]=temp_time%10+48;
```

Рисунок 2.10 – Таймер1

```
lcd_com(0x84); //установка курсора в позицию строка 1 место 5
if(pas == 0) // условный оператор кнопки 0
 {
        lcd_data(0x20);
        lcd_data(0x20);
        lcd_data(lcd[5]);
        lcd_data(lcd[4]);
        lcd_data(lcd[3]);          //Вывод значения 3 из массива lcd
```

```
            lcd_data(lcd[2]);          //Вывод значения 2 из массива lcd
            lcd_data(lcd[1]);          //Вывод значения 1 из массива lcd
            lcd_data(lcd[0]);          //Вывод значения 0 из массива lcd
            lcd_data(0x20);
            lcd_data(0x20);}
        else if(pas == 1){             // условный оператор кнопки 1
             lcd_data(0x20);
             lcd_data(lcd[5]);
             lcd_data(lcd[4]);
             lcd_data(lcd[3]);         //Вывод значения 3 из массива lcd
             lcd_data(0x2E);
             lcd_data(lcd[2]);         //Вывод значения 2 из массива lcd
             lcd_data(lcd[1]);         //Вывод значения 1 из массива lcd
             lcd_data(lcd[0]);         //Вывод значения 0 из массива lcd
             lcd_data(0x20);
             lcd_data(0x6B);
        }
        else if(pas == 2){ // условный оператор кнопки 2
             lcd_data(0x30);
             lcd_data(0x2E);
             lcd_data(lcd[5]);
             lcd_data(lcd[4]);
             lcd_data(lcd[3]);         //Вывод значения 3 из массива lcd
             lcd_data(lcd[2]);         //Вывод значения 2 из массива lcd
             lcd_data(lcd[1]);         //Вывод значения 1 из массива lcd
             lcd_data(lcd[0]);         //Вывод значения 0 из массива lcd
             lcd_data(0x20);
             lcd_data(0x4D);}
}
```

Рисунок 2.11 – Таймер1

2.7 Функция “init_lcd”

Инициализация жидкокристаллического индикатора для платформы EASY AVR V7. Код функции с объяснением её работы изображен на рисунках 2.12-2.14.

```
void init_lcd()
{
DDRC=0b11111111;              //Запись значения FF в порт C
PORTC=0x00;      /* Data Direction Register; настройка линий порта C;  линии порта C работают на вывод, PC4-PC7 - данные АЦЖКИ*/
PORTA=0b00000000;
DDRA=0b00000100;/* Data Direction Register; настройка линий порта A; линии порта A работают на ввод, PA2 - сигнал RS АЦЖКИ*/
PORTD=0b00000000;
DDRD=0b01000000;              /* Data Direction Register; настройка линий порта D; линии порта D работают на ввод, PD6 - сигнал E АЦЖКИ*/
_delay_ms(50);     //Функция формирования задержки 50 мс
PORTA&=~(1<<RS);     // Линию RS устанавливаем в 0 (передача команды)
PORTC=0b00110000;    //Вывод в старшую тетраду PORTC команду выбора режима интерфейса
PORTD&=~(1<<E);       // Сбрасываем E в ноль
```

Рисунок 2.12 - Функция “init_lcd”

```
asm("nop");
PORTD|=(1<<E);          // Устанавливаем E в единицу
_delay_us(10);          //Функция формирования задержки 10 мкс
PORTD&=~(1<<E);              // Сбрасываем E в ноль (сформирован импульс E длительностью 10 мкс)
  _delay_ms(1);    //Функция формирования задержки 1 мс
```

```
PORTC=0b00110000;    //Вывод в старшую тетраду PORTC команду выбора режима интерфейса
PORTD&=~(1<<E);      // Сбрасываем E в ноль
asm("nop");
PORTD|=(1<<E);       // Устанавливаем E в единицу
_delay_us(10);       //Функция формирования задержки 10 мкс
PORTD&=~(1<<E);      // Сбрасываем E в ноль (сформирован импульс E длительностью 10 мкс)
_delay_ms(1);        //Функция формирования задержки 1 мс
PORTC=0b00110000;//Вывод в старшую тетраду PORTC команду выбора режима интерфейса
PORTD&=~(1<<E);// Сбрасываем E в ноль
asm("nop");
PORTD|=(1<<E); // Устанавливаем E в единицу
_delay_us(10);       //Функция формирования задержки 10 мкс
PORTD&=~(1<<E);      // Сбрасываем E в ноль (сформирован импульс E длительностью 10 мкс)
_delay_ms(1);//Функция формирования задержки 1 мс
_delay_ms(10);//Функция формирования задержки 10 мс
lcd_com(0x20);   /*Настройка дисплея (установка интерфейса (4хбитный), номера строки (первая строка) и типа шрифта (5 x 8 точек) )*/
```

Рисунок 2.13 - Функция “init_lcd”

```
lcd_com(0x28);   /*Настройка дисплея (установка интерфейса (4хбитный), номера строки (вторая строка) и типа шрифта (5 x 8 точек) )*/

lcd_com(0x08);  /* Управление включениями дисплея (вывод на дисплей, отображение курсора и мигание курсора) Вывод на дисплей отключен, курсор не отображается, мигание курсора отключено */
lcd_com(0x01);   //Очистка дисплея
```

```
lcd_com(0x06);  /*Назначение направление перемещения курсора и включения сдвига всего дисплея сдвиг дисплея выключен, курсор перемещается вправо */
lcd_com(0x0C); /* Управление включениями дисплея (вывод на дисплей, отображение курсора и мигание курсора) Вывод на дисплей включен, курсор не отображается, мигание курсора отключено */
}
```

Рисунок 2.14 – Функция “init_lcd”

2.8 Функция “lcd_com”

Функция осуществляет передачу команды (из списка команд данного ЖКИ). В связи с особенностью подключения ЖКИ, передача происходит в два действия: передача старшей тетрады и передача младшей тетрады. Код представлен на рисунках 2.15-2.16.

```
    void lcd_com (unsigned char lcd)
    {
     unsigned char temp;   //Объявляется беззнаковая символьная переменная
    PORTA&=~(1<<RS); // Линию RS устанавливаем в 0 (передача команды)
    PORTD|=(1<<E);       // Устанавливаем E в единицу
temp=lcd;
```

Рисунок 2.15 - Функция “lcd_com”

```
PORTC=temp;    //Передача полученной последовательности в порт C, к которому подключен ЖКИ
asm("nop");       //Команда из ассемблера "nop" (No OPeration), создает задержку в один такт
PORTD&=~(1<<E);  // Сбрасываем E в ноль
asm("nop");
PORTD|=(1<<E);     // Устанавливаем E в единицу
```

```
temp=lcd*16;       /*Содержимое переменной lcd подвергается сдвигу на 4 разряда влево (lcd*16)
В результате содержимое старшей тетрады замещается содержимым младшей*/
PORTC=temp;      //Передача полученной последовательности в порт С, к которому подключен ЖКИ
asm("nop");       //Команда из ассемблера "nop" (No OPeration), создает задержку в один такт
PORTD&=~(1<<E);// Сбрасываем E в ноль
asm("nop");
 _delay_ms(2);      //Функция формирования задержки 2 мс
        }
```

Рисунок 2.16 – Функция “lcd_com”

2.9 Функция “lcd_data”

Функция осуществляет передачу команды (из списка команд данного ЖКИ). В связи с особенностью подключения ЖКИ, передача происходит в два действия: передача старшей тетрады и передача младшей тетрады. Код функции представлен на рисунках 2.17-2.18.

```
        void lcd_data (unsigned char lcd)
        {
        unsigned char temp;       //Объявляется беззнаковая символьная переменная
```

Рисунок 2.17 - Функция “lcd_data”

```
        PORTA|=(1<<RS);        // Линию RS устанавливаем в 1 (передача данных)
        PORTD|=(1<<E);         // Устанавливаем E в единицу
        temp=lcd;
        PORTC=temp;            //Передача полученной последовательности в порт С, к которому подключен ЖКИ
```

```
        asm("nop");          //Команда из ассемблера "nop" (No OPeration), создает задержку в один такт
PORTD&=~(1<<E);// Сбрасываем E в ноль
asm("nop");
PORTD|=(1<<E); // Устанавливаем E в единицу
temp=lcd*16;        /*Содержимое переменной lcd подвергается сдвигу на 4 разряда влево (lcd*16) В результате содержимое старшей тетрады замещается содержимым младшей*/
PORTC=temp;//Передача полученной последовательности в порт C, к которому подключен ЖКИ
asm("nop"); //Команда из ассемблера "nop" (No OPeration), создает задержку в один такт
PORTD&=~(1<<E);       // Сбрасываем E в ноль
asm("nop");
_delay_ms(2);       //Функция формирования задержки 2 мс
        }
```

Рисунок 2.18 – Функция “lcd_data”

2.10 Функция “write_str”

Для удобства вывода текста на АЦЖКИ используется функция “write_str”. Функция преобразует переданные данные (символы) в массив, который выводится на ЖКИ через функцию вывода данных. Таким образом, можно передать данные без предварительной кодировки, размер которых превышает 1 байт. Код функции представлен на рисунке 2.19.

```
        void write_str(const char *str)
{
unsigned char k;    //Объявление локальной переменной
for (k=0;k<255;k++)
{
```

```
if (str[k]==0) // Если k-тый элемент принятой строки не содержит информации (равен нулю)
return; // Выход из цикла if
else lcd_data(str[k]); //Иначе происходит вызов функции передачи данных, которой передается k-тый элемент строки
}
```

Рисунок 2.19 – Функция “write_str”

2.11 Функция “init_adc”

Данная функция необходима для настройки входов аналого-цифрового преобразователя. Регистр ADMUX управляет выбором опорного напряжения, а также входа мультиплексора. Регистр ADMUX представлен на рисунке 2.20.

Код функции представлен ниже.

```
void init_adc(void)
{
DDRA=0b00001111;                    // Настройка входов АЦП
PORTA=0b00000000;
ADMUX=0b00000101;                   // Настройка регистра АЦП
ADCSRA=0b10000111;  /*бит 7 - включение АЦП; биты 2,1,0-выбор предделителя=128; частота преобразования = 62500 Гц*/
ADCSRA=(0b10000111|0b01000000); //бит 6 - запуск преобразования
}
```

Рисунок 2.20 – Функция “init_adc”

3 ОТЛАДКА И РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Данная программа написана в среде Atmel Studio 7.0, разрабатывалась для отладочной платы Easy AVR v7 Development System фирмы mikroElectronika.

При подключении платы к источнику питания происходит настройка портов и инициализация всех аппаратных средств. На экран АЦЖКИ выводится строка: “P = … Pa” (кнопка 0 по умолчанию), отображающая давление в диапазоне от 0 до 400 кПа. Пример показан на рисунке 3.1.

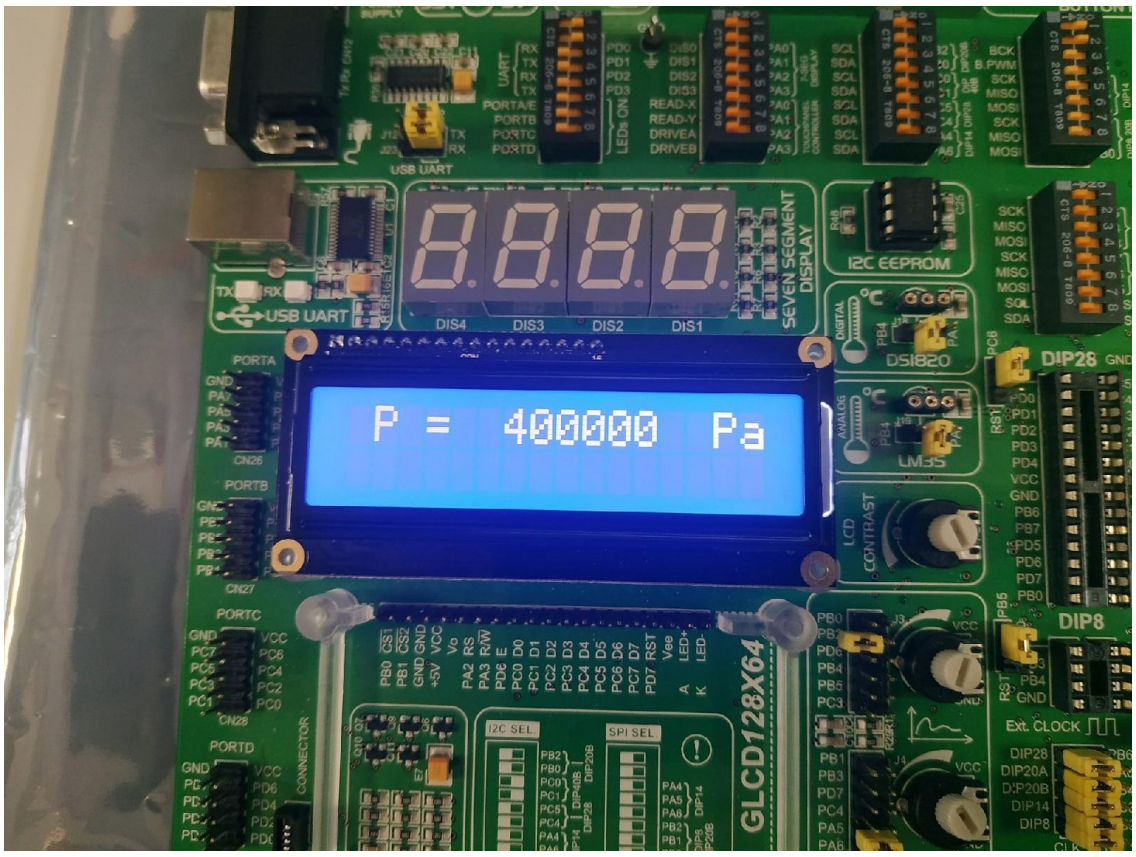

Рисунок 3.1 – Пример работоспособности программы (кнопка 0)

При нажатии кнопки 1, единицы измерения меняются на кПа. Пример показан на рисунке 3.2.

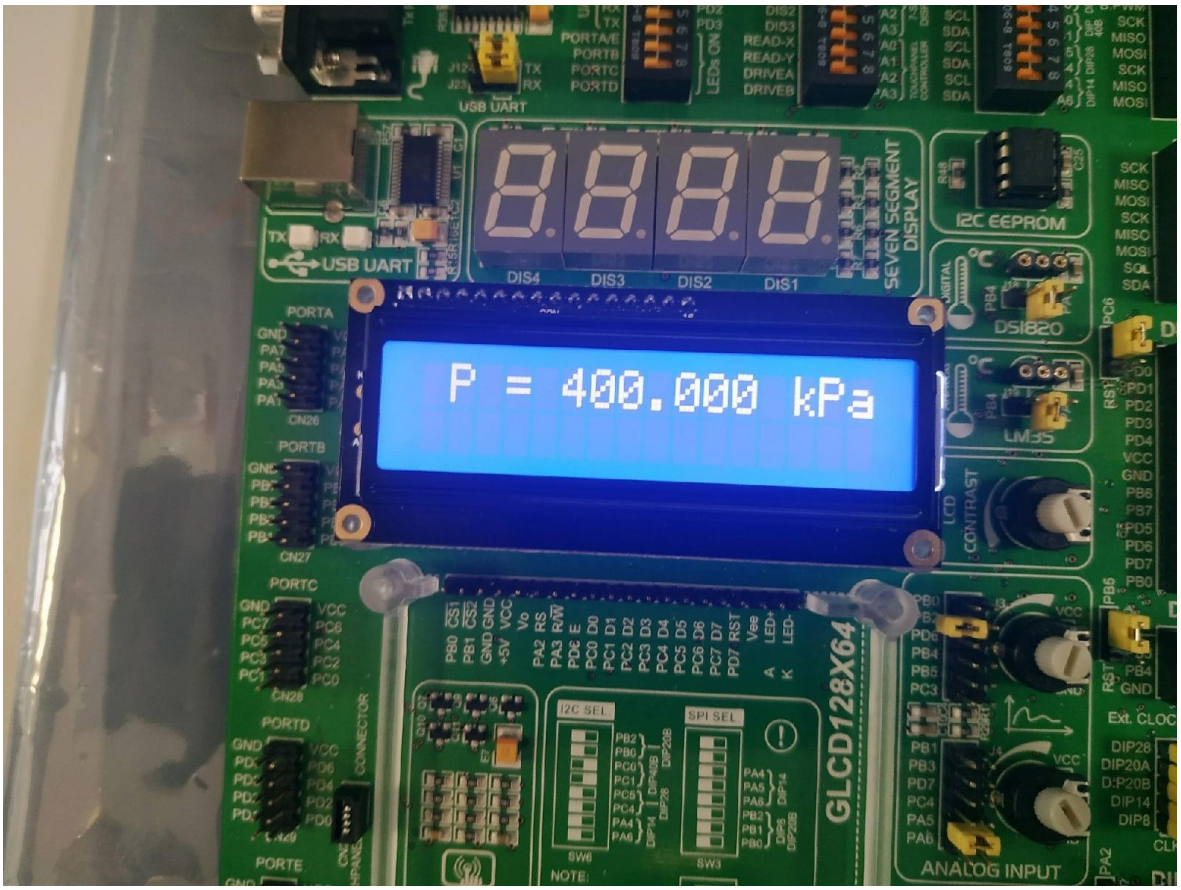

Рисунок 3.2 – Пример работоспособности программы (кнопка 1)

При нажатии кнопки 2, единицы измерения меняются на МПа. Пример показан на рисунке 3.3.

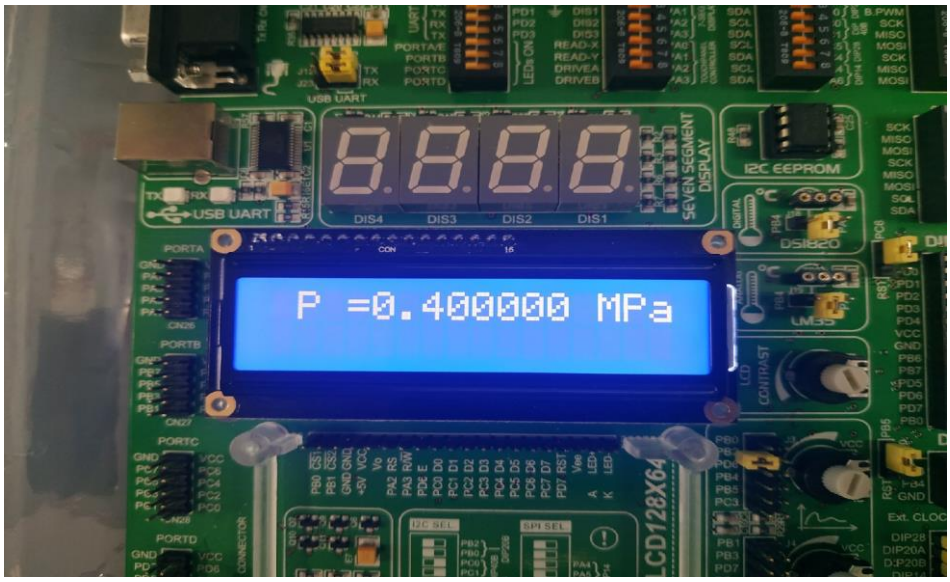

Рисунок 3.2 – Пример работоспособности программы (кнопка 2)

ЗАКЛЮЧЕНИЕ

Во время выполнения курсовой работы были получены навыки работы с интегрированной средой разработки Atmel Studio 7.0. Получены практические навыки написания и отладки программы на языке Си для микроконтроллера ATmega32 базовой платформы EasyAVR v7, а также умение оформлять курсовые работы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. EasyAVR v7 Development System - Руководство пользователя [интернет-ресурс] https://static.chipdip.ru/lib/002/DOC001002769.pdf
2. АЦЖКИ 2x16 datasheet [интернет-ресурс] https://static.chipdip.ru/lib/777/DOC000777905.pdf
3. EasyAVR v7 Schematics [интернет-ресурс] https://static.chipdip.ru/lib/826/DOC000826528.pdf

ПРИЛОЖЕНИЕ А

ТЕКСТ ПРОГРАММЫ

```
/* */
//////////////////////////
#define F_CPU 8000000UL                         // Определение тактовой частоты ( 8 МГц)
#define RS 2
#define E 6
#include <avr/io.h>
#include <util/delay.h>                         //
#include <avr/interrupt.h>                //
void timer1_init(void);
void timer_init(void);
void init_PORT(void);
// Прототип функции настройки АЦП
void init_adc(void);
unsigned long int temp_code_ADC=0;
volatile unsigned int code_ADC=0;
volatile unsigned char temp_ADC;

// Прототипы функций инициализации и вывода на АЦЖКИ
void init_lcd(void);
void lcd_com(unsigned char lcd);
void lcd_data(unsigned char lcd);
void write_str(const char* str);
//Глобальные переменные
volatile unsigned char sec_flag=0;

volatile unsigned long int time_count=0;
unsigned long int temp_time=0;

unsigned char lcd[6]={0,0,0,0,0,0};
// Элементы программы опроса кнопок
volatile unsigned char gread_key=0, gkey_up_counter=0, gkey_status=0, gkey_down_counter=0;
volatile unsigned char gkey=0,pas=0;
void scan_key(void);
int main(void)
{
	/*  ИНИЦИАЛИЗАЦИЯ */
	asm ("cli");               //Команда ассемблера "запрет прерываний"

	init_PORT();
	timer_init();
	timer1_init();
	init_adc();
	init_lcd();

	lcd_com(0x81);                    //Установка курсора в позицию: строка 1 знакоместо 1
	write_str("P =  000000  Pa");     //Вывод названия прибора
	asm ("sei");

	 while(1)
{
	if ((gkey_status&0b10000000)!=0)
	{
		gkey_status=gkey_status&0b01000000;
		if (gkey==0)
		{
			pas = 0;
```

```
                PORTB=0b00000000;
            }

            if (gkey==1)
            {
                pas = 1;
                PORTB=0b00000000;
            }

            if(gkey == 2)
            {
                pas = 2;
                PORTB=0b00000000;
            }
            }
        }
        return 0;
 }

void init_PORT()
{
        PORTA=0b00000000;
        DDRA=0b00000100;          /* Data Direction Register;
                                        настройка линий порта A;
                                        старшие линии порта A работают на ввод */
        PORTC=0x00;
        DDRC=0xff;
        PORTB=0x00;
        DDRB=0xff;
        PORTD=0x00;
        DDRD=0b01000000;
}
void timer1_init(void)    {
        TCNT1=0x0000;
        OCR1A=31250; //   число 31250
        TCCR1A=0b00000000;// режим CTC
        TCCR1B=0b00001100;// делитель на 256
        TIFR=0b00111100;
        TIFR=0b000010010;
        TIMSK=0b00010010;
}

void timer_init(void)    {
        TCNT0=0b00000000;
        OCR0=49; //   число 50
        TCCR0=0b00001100;// делитель на 256
        TIFR=0b00000011;


}
ISR (TIMER0_COMP_vect) {
        scan_key();
}
ISR (TIMER1_COMPA_vect)
{
            ADCSRA=(0b10000111|0b01000000);
            temp_ADC=ADCSRA;
            while((temp_ADC&0b01000000)!=0)
            {
                temp_ADC=ADCSRA;
            }
            code_ADC=0;
            code_ADC=ADCL;
            temp_ADC=ADCH;
            code_ADC=code_ADC|(temp_ADC<<8);
            temp_code_ADC=code_ADC*391.006843;
            temp_time=temp_code_ADC;
```

```
                lcd[0]=temp_time%10+48;   //и смещение на 48 в соответствии с таблицей кодировки
АЦЖКИ
                temp_time=temp_time/10;
                lcd[1]=temp_time%10+48;
                temp_time=temp_time/10;
                lcd[2]=temp_time%10+48;
                temp_time=temp_time/10;
                lcd[3]=temp_time%10+48;
                /////new
                temp_time=temp_time/10;
                lcd[4]=temp_time%10+48;
                temp_time=temp_time/10;
                lcd[5]=temp_time%10+48;
                /////new
                lcd_com(0x84);                          //Установка курсора в позицию: строка 1
знакоместо 6

                if(pas == 0)
                {
                      lcd_data(0x20);
                      lcd_data(0x20);
                      lcd_data(lcd[5]);
                      lcd_data(lcd[4]);
                      /////new
                      lcd_data(lcd[3]);          //Вывод значения 3 из массива lcd
                      lcd_data(lcd[2]);          //Вывод значения 2 из массива lcd
                      lcd_data(lcd[1]);          //Вывод значения 1 из массива lcd
                      lcd_data(lcd[0]);          //Вывод значения 0 из массива lcd
                      lcd_data(0x20);
                      lcd_data(0x20);
                }
                else if(pas == 1)
                {
                       lcd_data(0x20);
                       lcd_data(lcd[5]);
                       lcd_data(lcd[4]);
                       lcd_data(lcd[3]);         //Вывод значения 3 из массива lcd
                       lcd_data(0x2E);
                       lcd_data(lcd[2]);         //Вывод значения 2 из массива lcd
                       lcd_data(lcd[1]);         //Вывод значения 1 из массива lcd
                       lcd_data(lcd[0]);         //Вывод значения 0 из массива lcd
                       lcd_data(0x20);
                       lcd_data(0x6B);
                }
                else if(pas == 2)
                {
                       lcd_data(0x30);
                       lcd_data(0x2E);
                       lcd_data(lcd[5]);
                       lcd_data(lcd[4]);
                       lcd_data(lcd[3]);         //Вывод значения 3 из массива lcd
                       lcd_data(lcd[2]);         //Вывод значения 2 из массива lcd
                       lcd_data(lcd[1]);         //Вывод значения 1 из массива lcd
                       lcd_data(lcd[0]);         //Вывод значения 0 из массива lcd
                       lcd_data(0x20);
                       lcd_data(0x4D);
                }
                }

void init_lcd()
{
// Инициализация жидкокристаллического индикатора версия для платформы EASY AVR V7
// Формирование инициализационной последовательности: стр.18 "13.Initializing of LCM"
(для 4хбитного интерфейса)

       DDRC=0b11111111;                  //Запись значения FF в порт C
       PORTC=0x00;                /* Data Direction Register;
                                        настройка линий порта C;
```

```
                                             линии порта C работают на вывод, PC4-PC7 - данные
АЦЖКИ*/
       PORTA=0b00000000;
       DDRA=0b00000100;            /* Data Direction Register;
                                   настройка линий порта A;
                                   линии порта A работают на ввод, PA2 - сигнал RS АЦЖКИ*/
       PORTD=0b00000000;
       DDRD=0b01000000;            /* Data Direction Register;
                                   настройка линий порта D;
                                   линии порта D работают на ввод, PD6 - сигнал E АЦЖКИ*/
              _delay_ms(50);       //Функция формирования задержки 50 мс
//////////////////
       PORTA&=~(1<<RS);     // Линию RS устанавливаем в 0 (передача команды)
       PORTC=0b00110000;    //Вывод в старшую тетраду PORTC команду выбора режима интерфейса
       PORTD&=~(1<<E);             // Сбрасываем E в ноль
       asm("nop");
       PORTD|=(1<<E);              // Устанавливаем E в единицу
       _delay_us(10);              //Функция формирования задержки 10 мкс
       PORTD&=~(1<<E);             // Сбрасываем E в ноль (сформирован импульс E длительностью 10
мкс)
       _delay_ms(1); //Функция формирования задержки 1 мс
///////////////////
       PORTC=0b00110000;    //Вывод в старшую тетраду PORTC команду выбора режима интерфейса
       PORTD&=~(1<<E);             // Сбрасываем E в ноль
       asm("nop");
       PORTD|=(1<<E);              // Устанавливаем E в единицу
       _delay_us(10);              //Функция формирования задержки 10 мкс
       PORTD&=~(1<<E);             // Сбрасываем E в ноль (сформирован импульс E длительностью 10
мкс)
       _delay_ms(1); //Функция формирования задержки 1 мс
//////////////////
       PORTC=0b00110000;    //Вывод в старшую тетраду PORTC команду выбора режима интерфейса
       PORTD&=~(1<<E);             // Сбрасываем E в ноль
       asm("nop");
       PORTD|=(1<<E);              // Устанавливаем E в единицу
       _delay_us(10);              //Функция формирования задержки 10 мкс
       PORTD&=~(1<<E);             // Сбрасываем E в ноль (сформирован импульс E длительностью 10
мкс)
       _delay_ms(1); //Функция формирования задержки 1 мс
//////////////////
       _delay_ms(10);       //Функция формирования задержки 10 мс

       lcd_com(0x20);              /*Настройка дисплея (установка интерфейса (4хбитный),
                                          номера строки (первая строка) и типа шрифта (5 x 8 то-
чек) )*/
       lcd_com(0x28);              /*Настройка дисплея (установка интерфейса (4хбитный),
                                          номера строки (вторая строка) и типа шрифта (5 x 8 то-
чек) )*/

       lcd_com(0x08);              /* Управление включениями дисплея (вывод на дисплей, отображе-
ние курсора и мигание курсора)

                                          Вывод на дисплей отключен, курсор не отображается, мига-
ние курсора отключено */
       lcd_com(0x01);              //Очистка дисплея
       lcd_com(0x06);              /*Назначение направление перемещения курсора и включения сдвига
всего дисплея

                                          сдвиг дисплея выключен, курсор перемещается вправо */
       lcd_com(0x0C);              /* Управление включениями дисплея (вывод на дисплей, отображе-
ние курсора и мигание курсора)

                                          Вывод на дисплей включен, курсор не отображается, мига-
ние курсора отключено */

// Конец инициализации
}

void lcd_com (unsigned char lcd)
{
```

```
	/* Функция осуществляет передачу команды (из списка команд данного ЖКИ: стр.15
"11.Instruction Table").
	В связи с особенностью подключения ЖКИ, передача происходит в два действия:
	передача старшей тетрады и передача младшей тетрады.*/

	unsigned char temp;		//Объявляется беззнаковая символьная переменная

	PORTA&=~(1<<RS);		// Линию RS устанавливаем в 0 (передача команды)
	PORTD|=(1<<E);			// Устанавливаем E в единицу
	temp=lcd;			//
	PORTC=temp;			//Передача полученной последовательности в порт C, к ко-
торому подключен ЖКИ

	asm("nop");			//Команда из ассемблера "nop" (No OPeration), создает
задержку в один такт
	PORTD&=~(1<<E);			// Сбрасываем E в ноль
	asm("nop");
	////////////////////
	PORTD|=(1<<E);				// Устанавливаем E в единицу
	temp=lcd*16;			/*Содержимое переменной lcd подвергается сдвигу на 4
разряда влево (lcd*16)
								В результате содержимое старшей тетрады за-
мещается содержимым младшей*/
	PORTC=temp;			//Передача полученной последовательности в порт C, к ко-
торому подключен ЖКИ
	asm("nop");			//Команда из ассемблера "nop" (No OPeration), создает
задержку в один такт
	PORTD&=~(1<<E);			// Сбрасываем E в ноль
	asm("nop");
	 _delay_ms(2);			//Функция формирования задержки 2 мс
}
void lcd_data (unsigned char lcd)
{
	/* Функция осуществляет передачу команды (из списка команд данного ЖКИ: стр.15
"11.Instruction Table").
	В связи с особенностью подключения ЖКИ, передача происходит в два действия:
	передача старшей тетрады и передача младшей тетрады.*/

	unsigned char temp;		//Объявляется беззнаковая символьная переменная

	PORTA|=(1<<RS);			// Линию RS устанавливаем в 1 (передача данных)
	PORTD|=(1<<E);			// Устанавливаем E в единицу
	temp=lcd;			/* */
	PORTC=temp;			//Передача полученной последовательности в порт C, к ко-
торому подключен ЖКИ

	asm("nop");			//Команда из ассемблера "nop" (No OPeration), создает
задержку в один такт
	PORTD&=~(1<<E);			// Сбрасываем E в ноль
	asm("nop");
	////////////////////
	PORTD|=(1<<E);				// Устанавливаем E в единицу
	temp=lcd*16;			/*Содержимое переменной lcd подвергается сдвигу на 4
разряда влево (lcd*16)
								В результате содержимое старшей тетрады за-
мещается содержимым младшей*/
		PORTC=temp;			//Передача полученной последовательности в порт C,
к которому подключен ЖКИ
	asm("nop");			//Команда из ассемблера "nop" (No OPeration), создает
задержку в один такт
	PORTD&=~(1<<E);			// Сбрасываем E в ноль
	asm("nop");
	 _delay_ms(2);			//Функция формирования задержки 2 мс
}
void write_str(const char *str)
{
	/* Функция преобразует переданные данные (символы) в массив, который выводится на ЖКИ че-
рез функцию вывода данных
```

```
	Таким образом, можно передать данные без предварительной кодировки, размер которых превышает 1 байт*/

	unsigned char k;		//Объявление локальной переменной

	for (k=0;k<255;k++)
	{
		if (str[k]==0)			// Если k-тый элемент принятой строки не содержит информации (равен нулю)
		return;				// Выход из цикла if
		else lcd_data(str[k]);	//Иначе происходит вызов функции передачи данных, которой передается k-тый элемент строки
	}
}
void scan_key(void)
{
	gread_key=PIND;
	gread_key=(~gread_key)&0x0f;
	if ((gread_key!=0)&&(gkey_status==0))
	{
		gkey_up_counter++;
		if (gkey_up_counter>10)
		{
			gkey_up_counter=0;
			gkey_status=0b11000000;
			gkey=0;
			////////////
			if ((gread_key&0x01)!=0) gkey = 0;
			if ((gread_key&0x02)!=0) gkey = 1;
			if ((gread_key&0x04)!=0) gkey = 2;
			if ((gread_key&0x08)!=0) gkey = 3;
		}
	}
	else gkey_up_counter=0;
	////////
	if ((gread_key==0)&&((gkey_status&0b01000000))!=0)
	{
		gkey_down_counter++;
		if (gkey_down_counter>10)
		{
			gkey_down_counter=0;
			gkey_status=gkey_status&0b10000000;
		}
	}
	else gkey_down_counter=0;
}
void init_adc(void)
{
	DDRA=0b00001111;			// Настройка входов АЦП
	PORTA=0b00000000;
	ADMUX=0b00000101;			// Настройка регистра АЦП
	ADCSRA=0b10000111;		/*бит 7 - включение АЦП;
						биты 2,1,0-выбор предделителя=128;
						частота преобразования = 62500 Гц*/

	ADCSRA=(0b10000111|0b01000000);
						//бит 6 - запуск преобразования
}
```