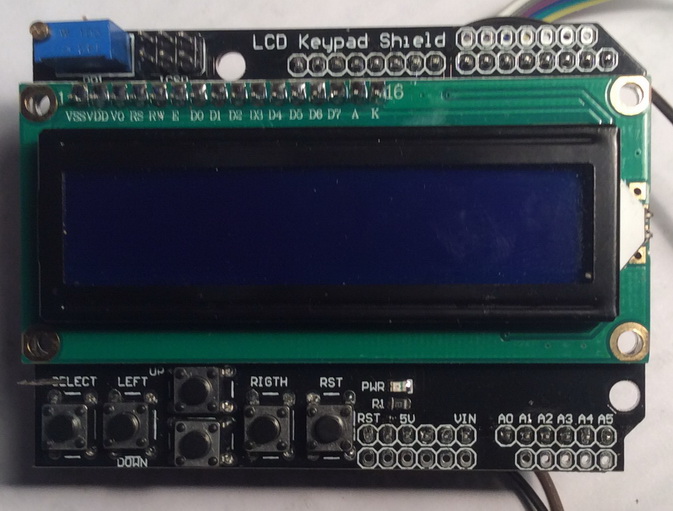
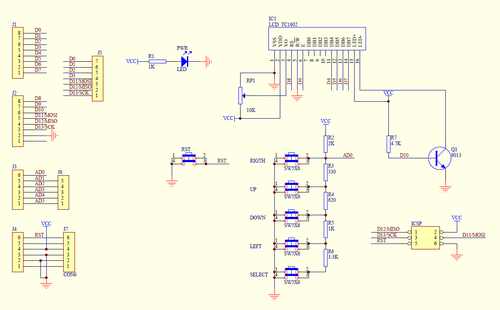
**Модуль LCD 16×2 с кнопками и регулятором**

Сегодня мы после долгого перерыва наконец-то вернёмся к программированию микроконтроллеров AVR. У нас по-прежнему Atmega 8a, а программировать мы будем опять дисплей 16х2, но дисплей данный организован в другом модуле, в котором уже установлен регулятор контрастности, а также установлено 7 кнопок. Данные кнопки подключены не к отдельным ножкам портов, а подключены только к одной ножке и между контактами кнопок установлены резисторы, что позволяет при нажатии определённой кнопки получать различное напряжение, которое затем распознаётся аналого-цифровым преобразователем микроконтроллера. Данный тип подключения позволит нам сэкономить четыре ножки порта. Кнопка Reset не участвует в данной цепочке и выведена на отдельный контакт, поэтому данную кнопку мы рассматривать не будем.

Вот так выглядит дисплей

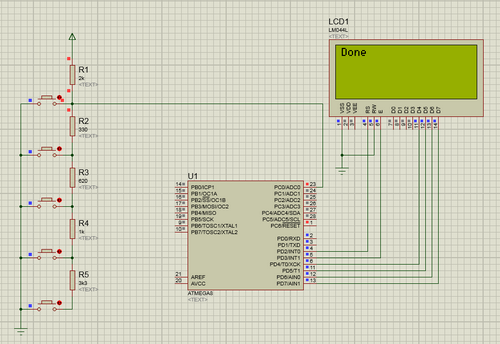


Вот краткая схема модуля (нажмите на картинку для увеличения размера)

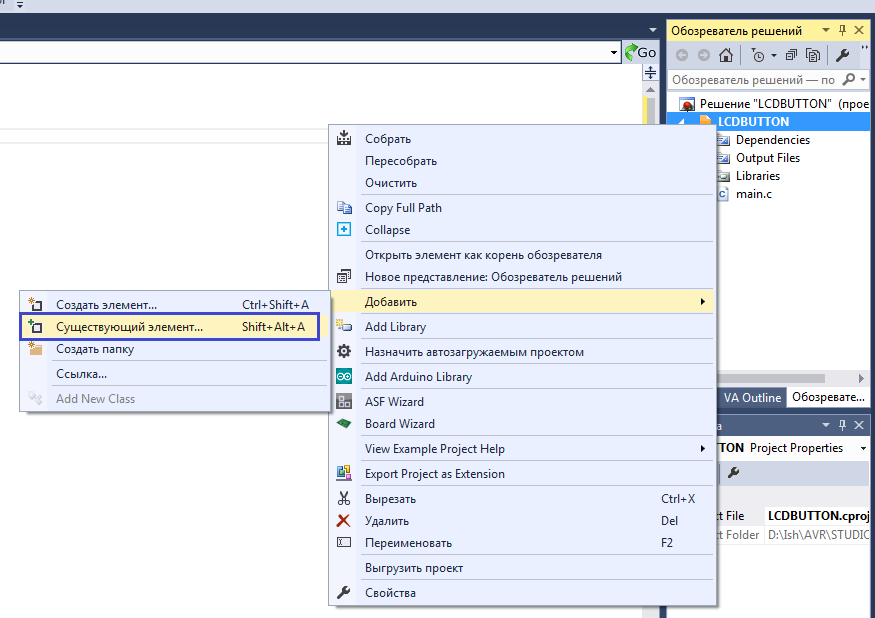
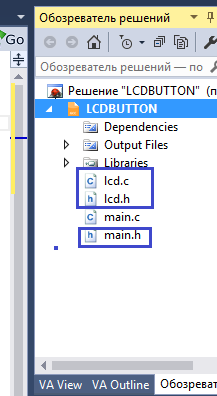
[](https://narodstream.ru/wp-content/uploads/2016/11/image00_1184.png)

Ножку AD0 мы подключим к ножке 23 микроконтроллера, что соответствует ножке PC0 (ADC0).

Остальные ножки дисплея мы подключим также как в уроке по подключению подобного дисплея в уроке 12 по его подключению 4-битным способом. Вот схема в протеусе (нажмите на картинку для увеличения размера)

[](https://narodstream.ru/wp-content/uploads/2016/11/image02_0934.png)

Как всегда, создадим новый проект в Atmel Studio 7, назовём его LCDBUTTON, выберем контроллер ATmega8A, зайдём в обозреватель решений и откроем файл main.c. Из проекта Test09 скопируем файлы lcd.h, lcd.c и main.h в папку с новым проектом, где лежит файл main.c. Подключим их в проект

А в файл нового проекта main.c скопируем содержимое файла Test09.c из проекта Test09.

Добавим в lcd.h прототип функции очистки дисплея

void str\_lcd (char

str1[]);

**void clearlcd(void);**

//—————————————-

Немного изменим текст в фунции main(), убрав оттуда лишнее и добавив на всякий случай очистку дисплея

int main(void)

{

                port\_ini(); //Инициализируем порты

                LCD\_ini();  //Инициализируем дисплей

**clearlcd();  //Очистим дисплей**

        setpos(0,0);

                str\_lcd("**Button Test**");

                while(1)

                {

                }

}

Соберем проект, прошьём контроллер и посмотрим, работает ли наш дисплей.

Теперь займёмся кнопками.

Так как распознавание нажатых кнопок происходит с использованием АЦП, то возьмём за основу исходный код из урока 23 по АЦП, а именно 2й проект – **MyADCISRLCD**. Здесь мы работали с АЦП по прерываниям. Файлы adc.c и adc.h скопируем в наш проект и подключим их в дерево проекта.

В файле main.h подключим adc.h и добавим некоторые переменные для АЦП и включим частоту 16 МГц

#define

F\_CPU **16**000000UL

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <stdio.h>

#include <stdlib.h>

**unsigned int adc\_value,adc\_counter,adc\_tmp;**

#include "lcd.h"

**#include "adc.h"**

В функции main() вызовем инициализацию нашего АЦП

**LCD\_ini();  //Инициализируем дисплей**

**ADC\_Init(); //Инициализируем АЦП**

В принципе нам нет необходимости в точном измерении напряжения, главное, чтобы оно попало в определённый предел, поэтому мы будем использовать только 8 бит АЦП. Тем не менее считать мы обязаны оба байта, иначе не начнется следующее преобразование. В связи с этим немного изменим код функции-обработчика прерывания от АЦП в файле adc.c

low\_adc =

ADCL;

high\_adc =

ADCH;//Верхняя часть регистра ADC должна быть считана последней, иначе не продолжится преобразование

        if(

adc\_counter<20) {

adc\_tmp+=**high\_adc**;

adc\_counter++;}

else {

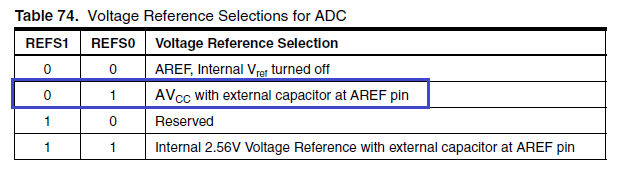
adc\_value=

adc\_tmp/20;

adc\_counter=0;

adc\_tmp=0;}

Также, чтобы в старшем байте было не только старших 4 бита, а было старших 8 байт, не забываем про ADLAR. Также некоторые изменения должны произойти в связи с использованием кварцевого резонатора на 16 МГц, но мы всё-таки оставим тот же делитель, просто в комментарии укажем другую частоту. А самые главные изменения будут связаны с тем, что напряжения некоторые используются большие чем 2,56 вольт, и данного опорного напряжения нам не хватит. Поэтому используем напряжение 5 В в качестве опорного. Для этого мы выключим бит  REFS1 в регистре ADMUX (см. даташит на контроллер стр. 199).



В результате получим следующий код функции инициализации

void ADC\_Init()

{

ADCSRA |= (1<<

ADEN) // Разрешение использования АЦП

                |(1<<

ADSC)//Запуск преобразования

                |(1<<

ADFR)//Непрерывный режим работы АЦП

                |(1<<

ADPS2)|(1<<

ADPS1)|(1<<

ADPS0)//Делитель 128 = **128** кГц

                |(1<<

ADIE);//Разрешение прерывания от АЦП

ADMUX |=**(1<<ADLAR)|**(**0**<<

REFS1)|(1<<

REFS0); //Внутренний Источник ОН 5в, вход ADC0

}

Также не забываем включить глобальные прерывания в main()

        ADC\_Init(); //Инициализируем АЦП

**sei();**

Ну, вроде бы все подготовительные операции завершены. Осталось только добавить код распознавания кнопок в бесконечный цикл функции main()

        {

**setpos(0,1);**

**if(adc\_value<25) str\_lcd("RIGHT KEY");**

**else if(adc\_value<65) str\_lcd("UP KEY     ");**

**else if(adc\_value<115) str\_lcd("DOWN KEY   ");**

**else if(adc\_value<180) str\_lcd("LEFT KEY   ");**

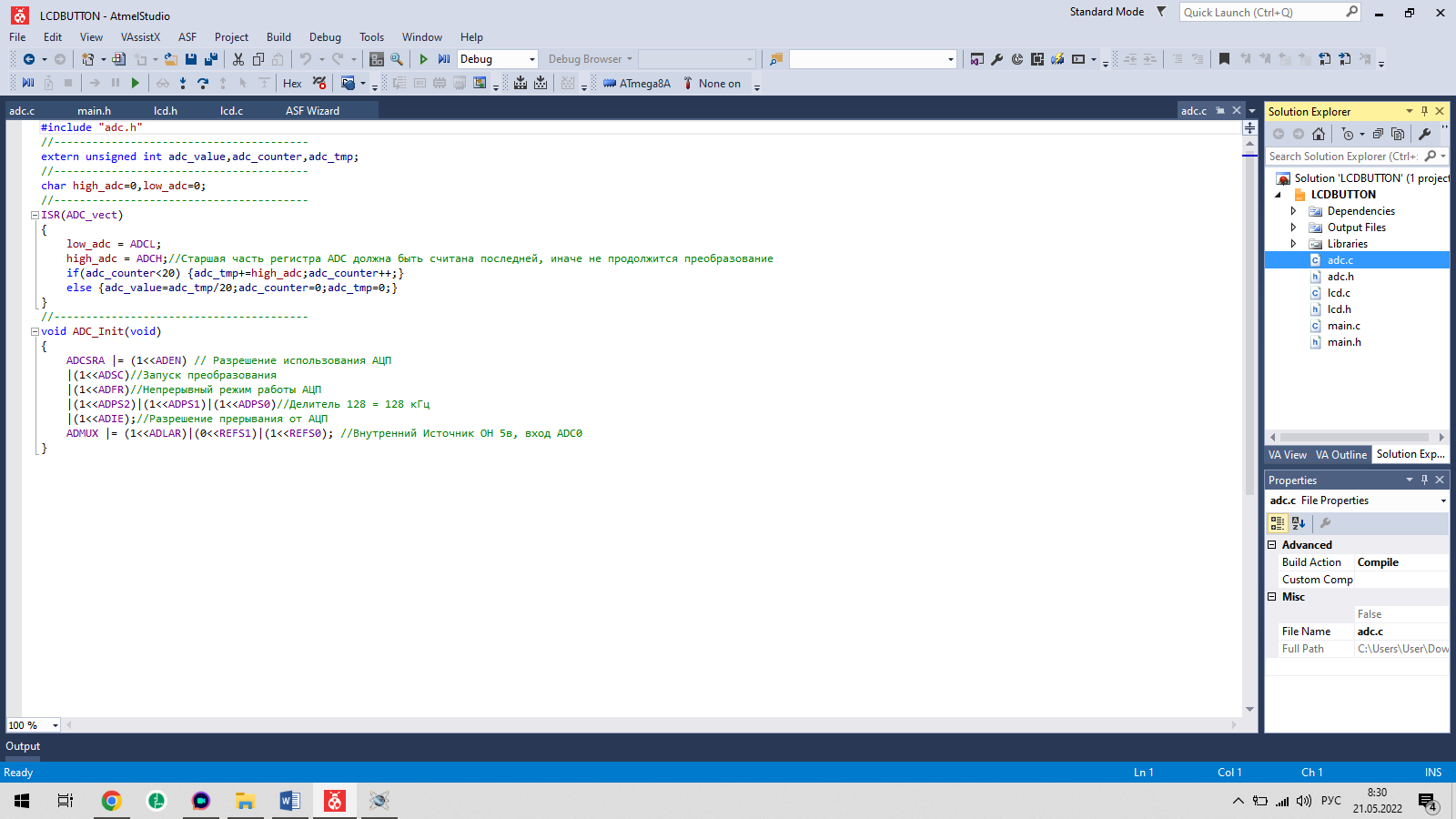
**else if(adc\_value<230) str\_lcd("SELECTT KEY");**

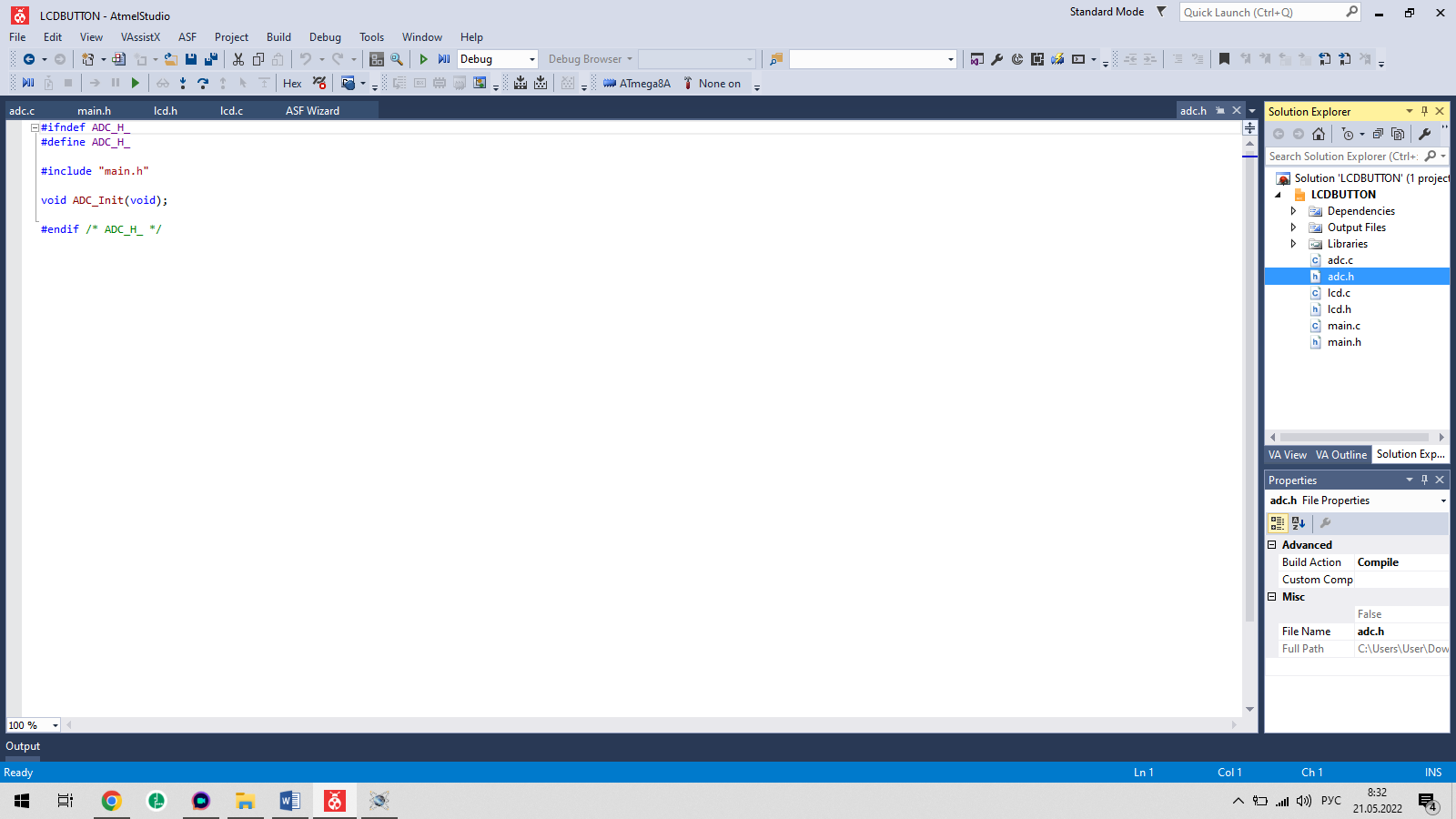
**\_delay\_ms(200);**

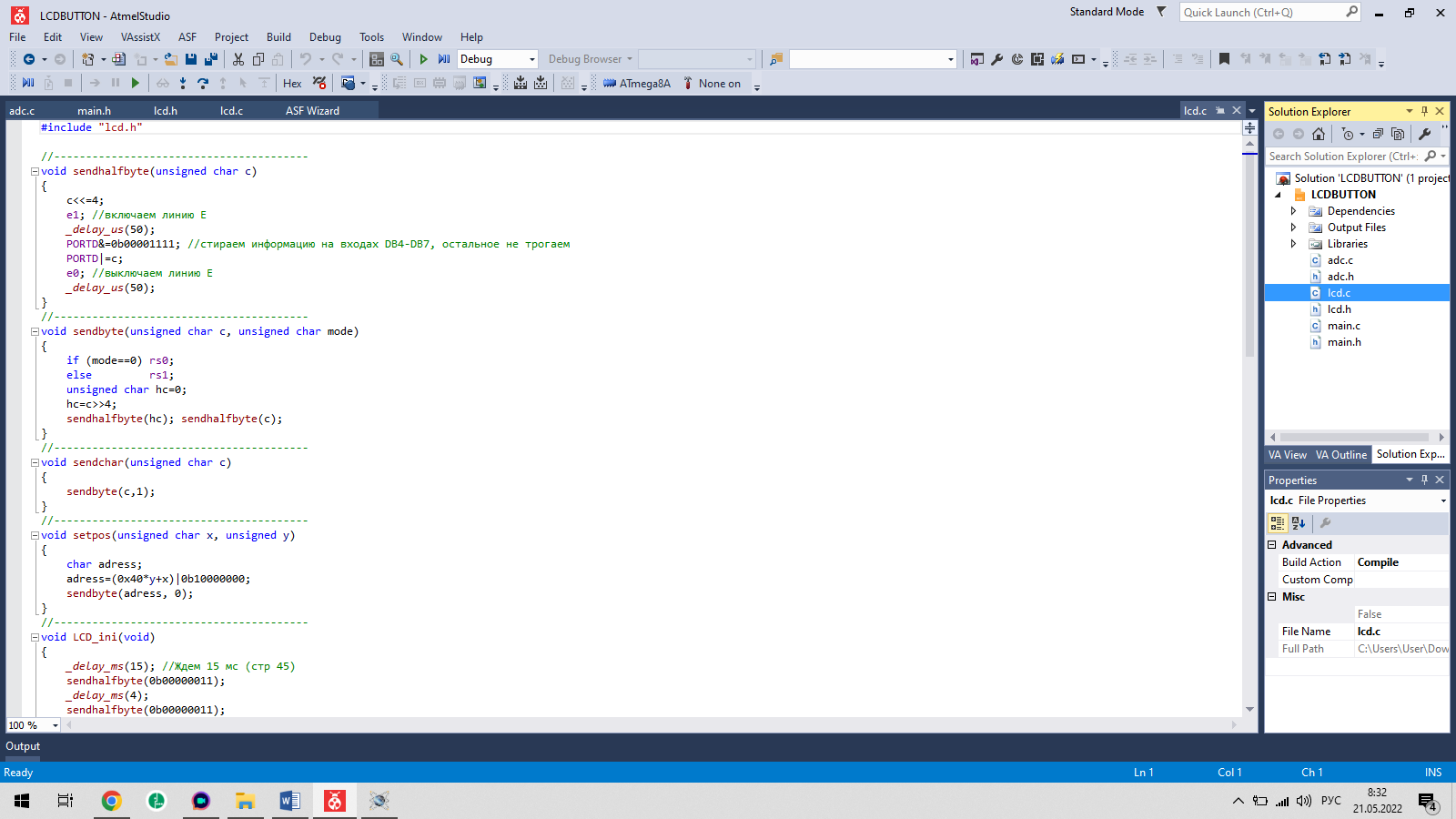
}

Скомпилируем код, прошьём контроллер и посмотрим результат нашей работы.

**КОД ПРЕДСТАВЛЕН НИЖЕ**







#include "lcd.h"

//----------------------------------------

void sendhalfbyte(unsigned char c)

{

c<<=4;

e1; //включаем линию Е

*\_delay\_us*(50);

PORTD&=0b00001111; //стираем информацию на входах DB4-DB7, остальное не трогаем

PORTD|=c;

e0; //выключаем линию Е

*\_delay\_us*(50);

}

//----------------------------------------

void sendbyte(unsigned char c, unsigned char mode)

{

if (mode==0) rs0;

else rs1;

unsigned char hc=0;

hc=c>>4;

sendhalfbyte(hc); sendhalfbyte(c);

}

//----------------------------------------

void sendchar(unsigned char c)

{

sendbyte(c,1);

}

//----------------------------------------

void setpos(unsigned char x, unsigned y)

{

char adress;

adress=(0x40\*y+x)|0b10000000;

sendbyte(adress, 0);

}

//----------------------------------------

void LCD\_ini(void)

{

*\_delay\_ms*(15); //Ждем 15 мс (стр 45)

sendhalfbyte(0b00000011);

*\_delay\_ms*(4);

sendhalfbyte(0b00000011);

*\_delay\_us*(100);

sendhalfbyte(0b00000011);

*\_delay\_ms*(1);

sendhalfbyte(0b00000010);

*\_delay\_ms*(1);

sendbyte(0b00101000, 0); //4бит-режим (DL=0) и 2 линии (N=1)

*\_delay\_ms*(1);

sendbyte(0b00001100, 0); //включаем изображение на дисплее (D=1), курсоры никакие не включаем (C=0, B=0)

*\_delay\_ms*(1);

sendbyte(0b00000110, 0); //курсор (хоть он у нас и невидимый) будет двигаться влево

*\_delay\_ms*(1);

}

//----------------------------------------

void clearlcd()

{

sendbyte(0b00000001, 0);

*\_delay\_us*(1500);

}

//----------------------------------------

void str\_lcd (char str1[])

{

wchar\_t n;

for(n=0;str1[n]!='\0';n++)

sendchar(str1[n]);

}

//----------------------------------------

