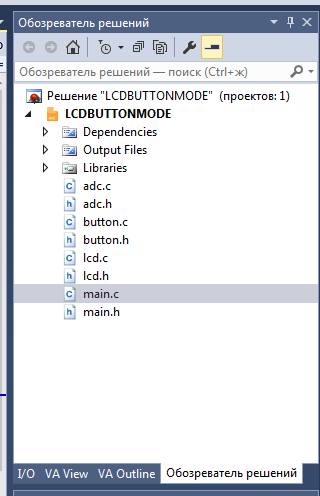
**Модуль LCD 16×2 .Продолжаем работать с кнопками**

На [**прошлом занятии**](https://narodstream.ru/avr-urok-35-modul-lcd-16x2-s-knopkami/) мы научились работать с кнопками модуля дисплея **LCD 16×2**и написали некоторую реакцию на кнопки.

Сегодня же мы постараемся улучшить данный алгоритм работы и написать код реакции на события изменения состояния кнопок с тем учётом, чтобы, во-первых, состояние кнопок отслеживалось в процессах, не мешающим выполнению основного кода, а во-вторых, чтобы мы могли в данных процессах отследить не только состояние кнопок, а ещё и изменение состояния кнопок, то есть процесс отжатия или нажатия кнопки. Причём, как правило, мы все наблюдаем, что реакция всегда происходит не на нажатие кнопки, а именно на отжатие. И данный алгоритм не только звучит тяжело, но и собственно код также не обещает быть лёгким. Поэтому данную поставленную задачу мы будем решать поэтапно.

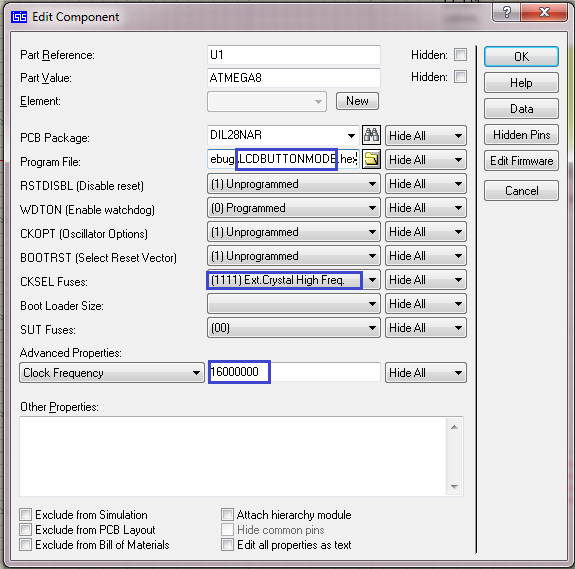
Во-первых, конечно, нам необходим будет проект. Так же, как и в прошлом занятии, мы создадим новый проект, выбрав контроллер Atmega8a, и назовём его LCDBUTTONMODE. Скопируем файлы main.h, lcd.c, lcd.h, adc.c, и adc.h в папку с проектом туда, где лежит файл main.c. Также данные файлы подключим к дереву проекта таким же образом, как и в прошлом занятии. Создадим ещё два новых файла button.c и button.h. Удалим полностью содержимое файла main.c и скопируем туда содержимое одноимённого файла из предыдущего проекта LCDBUTTON.



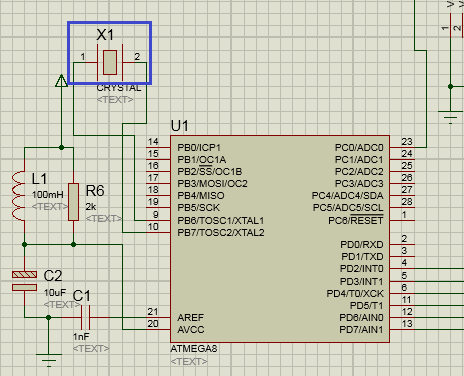
Подключим файл button.h в main.h и в button.c

**#include "button.h"**

Попробуем собрать код. Скопируем, переименуем в имя проекта из предыдущего проекта также файл для протеуса и запустим его. Исправим там путь до прошивки и изменим немного настройки в плане использования кварцевого резонатора на 16 МГц



Также добавим в схему собственно сам резонатор



Если всё нормально собралось и запустилось, то пишем код дальше. Соответственно лучшего решения для отвязывания задачи от выполнения основного алгоритма кода, чем использование таймеров, я не нашёл, поэтому мы и воспользуемся таймером. Таймер возьмём самый простой – нулевой, так как вдруг первый потребуется для более серьёзных задач, чем обработка нажатий и отжатий кнопок. Подключим файл main.h в файле button.h

#define BUTTON\_H\_

**#include "main.h"**

#endif /\* BUTTON\_H\_ \*/

Добавим код для таймера в файл button.c

#include "button.h"

**//—————————————**

**void init\_button\_timer(void)**

**{**

**TIMSK |= (1<<TOIE0);        //устанавливаем бит разрешения прерывания 0-ого счетчика по переполнению**

**TCCR0 |= (0<<CS02)|(1<<CS01)|(1<<CS00); // устанавливаем предделитель 64**

**//тем самым получаем — частота тактирования / предделитель / 256 = 16000000/64/256 = 976,5625 (около милисекунды)**

**//делим на 256, так как таймер нулевой всегда считает от 0x00 до 0xFF и это не регулируется никак**

**}**

**//—————————————**

**ISR (TIMER0\_OVF\_vect)**

**{**

**}**

Для хранения состояния кнопок в файле button.h создадим массив для хранения статусов кнопок

#include "main.h"

**//——————————————-**

**unsigned char button\_state[5];**

**//——————————————-**

Там же создадим макросы для кнопок

unsigned char button\_state[5];

//——————————————-

**#define Button\_Right 0x00**

**#define Button\_Up 0x01**

**#define Button\_Down 0x02**

**#define Button\_Left 0x03**

**#define Button\_Select 0x04**

//——————————————-

И ещё создадим макросы для состояния кнопок. В макроподстановках используем разные биты.

#define Button\_Select 0x04

//——————————————-

**#define ST\_PRESSED 0b00000001        //кнопка нажата**

**#define ST\_UNPRESSED 0b00000010 //кнопка отжата**

**#define ST\_ERROR 0b00000100     //не удалось считать статус**

**#define ST\_PRESSURE 0b00001000   //процесс нажатия кнопки**

**#define ST\_UNPRESSURE 0b00010000 //процесс отжатия кнопки**

**#define ST\_LOCKED 0b00100000        //кнопка недоступна**

//——————————————-

Попробуем собрать код, а также запустить прошивку в протеусе. Всё у нас работает, то есть мы своим кодом ничего не запортилось.

Дальнейший код реализации задуманных режимов мы продолжим писать в [**следующей части**](https://narodstream.ru/avr-urok-36-modul-lcd-16x2-prodolzhaem-rabotat-s-chast-2/) урока.

# ****Модуль LCD 16×2 .Продолжаем работать с кнопками****

В [**первой части**](https://narodstream.ru/avr-urok-36-modul-lcd-16x2-prodolzhaem-rabotat-s-chast-1/) занятия мы настроили проект, также содали проект для протеуса, настроили таймер и добавили некоторые макроподстановки для хранения определённых статусов.

Сегодня же мы продолжим начатое дело.

Начнём писать работу кнопок.

В файле реализации функций библиотеки button.c создадим функцию для определения текущего статуса кнопки

//—————————————

**unsigned char Read\_Button\_Press(unsigned char b)**

**{**

**switch(b)**

**{**

**case Button\_Right:**

**if(adc\_value<25)                                                return ST\_PRESSED;**

**else        return ST\_UNPRESSED;**

**break;**

**case Button\_Up:**

**if((adc\_value>25)&&(adc\_value<65))                return ST\_PRESSED;**

**else        return ST\_UNPRESSED;**

**break;**

**case Button\_Down:**

**if((adc\_value>65)&&(adc\_value<115))                return ST\_PRESSED;**

**else        return ST\_UNPRESSED;**

**break;**

**case Button\_Left:**

**if((adc\_value>115)&&(adc\_value<180))        return ST\_PRESSED;**

**else        return ST\_UNPRESSED;**

**break;**

**case Button\_Select:**

**if((adc\_value>180)&&(adc\_value<230))        return ST\_PRESSED;**

**else        return ST\_UNPRESSED;**

**break;**

**}**

**return ST\_ERROR;**

**}**

Добавим массив счётчиков в файле button.c

#include "button.h"

//—————————————

**static unsigned char but\_cnt[5]={0};**

Массив счётчиков для каждой кнопки нам потребуется для устранения дребезга кнопок. Но именно такой счётчик нужен для каждой кнопки. По-другому нельзя. Я пробовал, получается каша и ничего не работает.

Теперь напишем функцию обработки состояния кнопки и установки окончательного статуса

**//—————————————**

**void Read\_Button\_State (unsigned char b)**

**{**

**if ((button\_state[b]&ST\_LOCKED)!=0) return;**

**if(Read\_Button\_Press(b)==ST\_UNPRESSED)**

**{**

**if (but\_cnt[b]>0)**

**{**

**but\_cnt[b]—;**

**}**

**else**

**{**

**if ((button\_state[b]&ST\_PRESSED)!=0)**

**{**

**button\_state[b] |= ST\_UNPRESSURE;**

**button\_state[b] &= ~ST\_PRESSED;**

**button\_state[b] |= ST\_UNPRESSED;**

**}**

**}**

**}**

**else**

**{**

**if (but\_cnt[b]<5)**

**{**

**but\_cnt[b]++;**

**}**

**else**

**{**

**if ((button\_state[b]&ST\_UNPRESSED)!=0)**

**{**

**button\_state[b] |= ST\_PRESSURE;**

**button\_state[b] &= ~ST\_UNPRESSED;**

**button\_state[b] |= ST\_PRESSED;**

**}**

**}**

**}**

**}**

**//——————————————**

Чтобы данная функция виделась из функции-обработчика прерывания от таймера, прототип на неё добавим в файле button.c в начале кода

static unsigned char cnt=0, but\_cnt[5]={0};

//—————————————

**void Read\_Button\_State (unsigned char b);**

Напишем функцию инициализации статусов кнопок

//——————————————

void Buttons\_Ini(void)

{

        unsigned char i;

        //обнулим счётчики и сбросим статусы

        for (i=0;i<5;i++)

        {

                button\_state[i] = ST\_UNPRESSED;

        }

        //запустим таймер

        init\_button\_timer();

}

//——————————————

Напишем на неё прототип и вызовем в файле main.c  в главной функции main()

#define ST\_LOCKED 0b00100000        //кнопка недоступна

//——————————————-

**void Buttons\_Ini(void);**

//——————————————-

#endif /\* BUTTON\_H\_ \*/

— — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —

ADC\_Init(); //инициализируем АЦП

**Buttons\_Ini(); //инициализируем состояние кнопок**

Соберём код и опять проверим данный код протеусе, что у нас по-прежнему всё нормально работает и что таймер ни как не мешает и не влияет на работоспособность нашего кода.

Дальнейшую работу с написанными нами функциями для обеспечения отслеживания всех статусов и изменения статусов кнопок мы продолжим в следующей части.

**Часть 3**

# ****Модуль LCD 16×2 .Продолжаем работать с кнопками****

В [**части 1**](https://narodstream.ru/avr-urok-36-modul-lcd-16x2-prodolzhaem-rabotat-s-chast-1/) и [**части 2**](https://narodstream.ru/avr-urok-36-modul-lcd-16x2-prodolzhaem-rabotat-s-chast-2/) занятия мы настроили проект, также содали проект для протеуса, настроили таймер и добавили некоторые макроподстановки для хранения определённых статусов, также написали ряд функций для обеспечения отслеживания данных статусов

Сегодня мы продолжим это дело и попытаемся уже реализовать в работающий процесс

Напишем код для определения статуса пока одной кнопки в функции-обработчике прерывания от таймера

ISR (TIMER0\_OVF\_vect)

{

**Read\_Button\_State(Button\_Right);**

**Read\_Button\_State(Button\_Up);**

**Read\_Button\_State(Button\_Down);**

**Read\_Button\_State(Button\_Left);**

**Read\_Button\_State(Button\_Select);**

}

Напишем две функции для сброса и установки статуса кнопки

//——————————————

**void ResetButtonState(unsigned char b, unsigned char st)**

**{**

**button\_state[b]  &= ~st;**

**}**

**//——————————————**

**void SetButtonState(unsigned char b, unsigned char st)**

**{**

**button\_state[b]  |= st;**

**}**

//——————————————

void Buttons\_Ini(void)

Добавим прототипы данных функций

void Buttons\_Ini(void);

**void ResetButtonState(unsigned char b, unsigned char st);**

**void SetButtonState(unsigned char b, unsigned char st);**

//——————————————-

#endif /\* BUTTON\_H\_ \*/

Теперь в бесконечном цикле удалим часть старого кода попытаемся определить статус одной кнопки и среагируем на него, также не забываем про флаги. Можно убавить задержку

.

        while(1)

        {

                setpos(0,1);

**if (button\_state[Button\_Right]&ST\_UNPRESSURE)**

**{**

**//предотвратим обращение к кнопке из таймера во время выполнения обработчика**

**SetButtonState(Button\_Right,ST\_LOCKED);**

**str\_lcd("RIGHT KEY ");**

**ResetButtonState(Button\_Right,ST\_UNPRESSURE);**

**SetButtonState(Button\_Right,ST\_PRESSURE);**

**ResetButtonState(Button\_Right,ST\_LOCKED);**

**}**

*\_delay\_ms*(**50**);

}

Скомпилируем код, проверим его пока в протеусе. Сообщение о кнопке должно появляться именно в момент отжатия кнопки.

Определим статус ещё одной кнопки и попробуем также в протеусе понажимать их по очереди

                        ResetButtonState(Button\_Right,ST\_LOCKED);

                }

**if (button\_state[Button\_Up]&ST\_UNPRESSURE)**

**{**

**//предотвратим обращение к кнопке из таймера во время выполнения обработчика**

**SetButtonState(Button\_Up,ST\_LOCKED);**

**str\_lcd("UP KEY    ");**

**ResetButtonState(Button\_Up,ST\_UNPRESSURE);**

**SetButtonState(Button\_Up,ST\_PRESSURE);**

**ResetButtonState(Button\_Up,ST\_LOCKED);**

**}**

Можно, в принципе, попробовать и на живом контроллере, вдруг не пойдёт. Если всё нормально, то добавим теперь обработчики недостающих трёх кнопок и проверим работу также и в протеусе и на настоящем дисплее и контроллере.

                         ResetButtonState(Button\_Up,ST\_LOCKED);

                 }

**if (button\_state[Button\_Down]&ST\_UNPRESSURE)**

**{**

**//предотвратим обращение к кнопке из таймера во время выполнения обработчика**

**SetButtonState(Button\_Down,ST\_LOCKED);**

**str\_lcd("DOWN KEY  ");**

**ResetButtonState(Button\_Down,ST\_UNPRESSURE);**

**SetButtonState(Button\_Down,ST\_PRESSURE);**

**ResetButtonState(Button\_Down,ST\_LOCKED);**

**}**

**if (button\_state[Button\_Left]&ST\_UNPRESSURE)**

**{**

**//предотвратим обращение к кнопке из таймера во время выполнения обработчика**

**SetButtonState(Button\_Left,ST\_LOCKED);**

**str\_lcd("LEFT KEY  ");**

**ResetButtonState(Button\_Left,ST\_UNPRESSURE);**

**SetButtonState(Button\_Left,ST\_PRESSURE);**

**ResetButtonState(Button\_Left,ST\_LOCKED);**

**}**

**if (button\_state[Button\_Select]&ST\_UNPRESSURE)**

**{**

**//предотвратим обращение к кнопке из таймера во время выполнения обработчика**

**SetButtonState(Button\_Select,ST\_LOCKED);**

**str\_lcd("SELECT KEY");**

**ResetButtonState(Button\_Select,ST\_UNPRESSURE);**

**SetButtonState(Button\_Select,ST\_PRESSURE);**

**ResetButtonState(Button\_Select,ST\_LOCKED);**

**}**

Немного усовершенствуем наш код, а то как-то он устрашающе-громоздко смотрится. Создадим счётчик и массив из строк в функции main()

int main(void)

{

**unsigned char i=0;**

**char str1[5][11] =**

**{**

**"RIGHT KEY ",**

**"UP KEY    ",**

**"DOWN KEY  ",**

**"LEFT KEY  ",**

**"SELECT KEY"**

**};**

        port\_ini(); //инициализируем порты

Исправим код в бесконечном цикле

        while(1)

        {

                setpos(0,1);

**for (i=0;i<5;i++)**

                {

                        if (button\_state[**i**]&ST\_UNPRESSURE)

                        {

                                //предотвратим обращение к кнопке из таймера во время выполнения обработчика

                                SetButtonState(**i**,ST\_LOCKED);

                                str\_lcd(**str1[i]**);

                                ResetButtonState(**i**,ST\_UNPRESSURE);

                                SetButtonState(**i**,ST\_PRESSURE);

                                ResetButtonState(**i**,ST\_LOCKED);

                        }

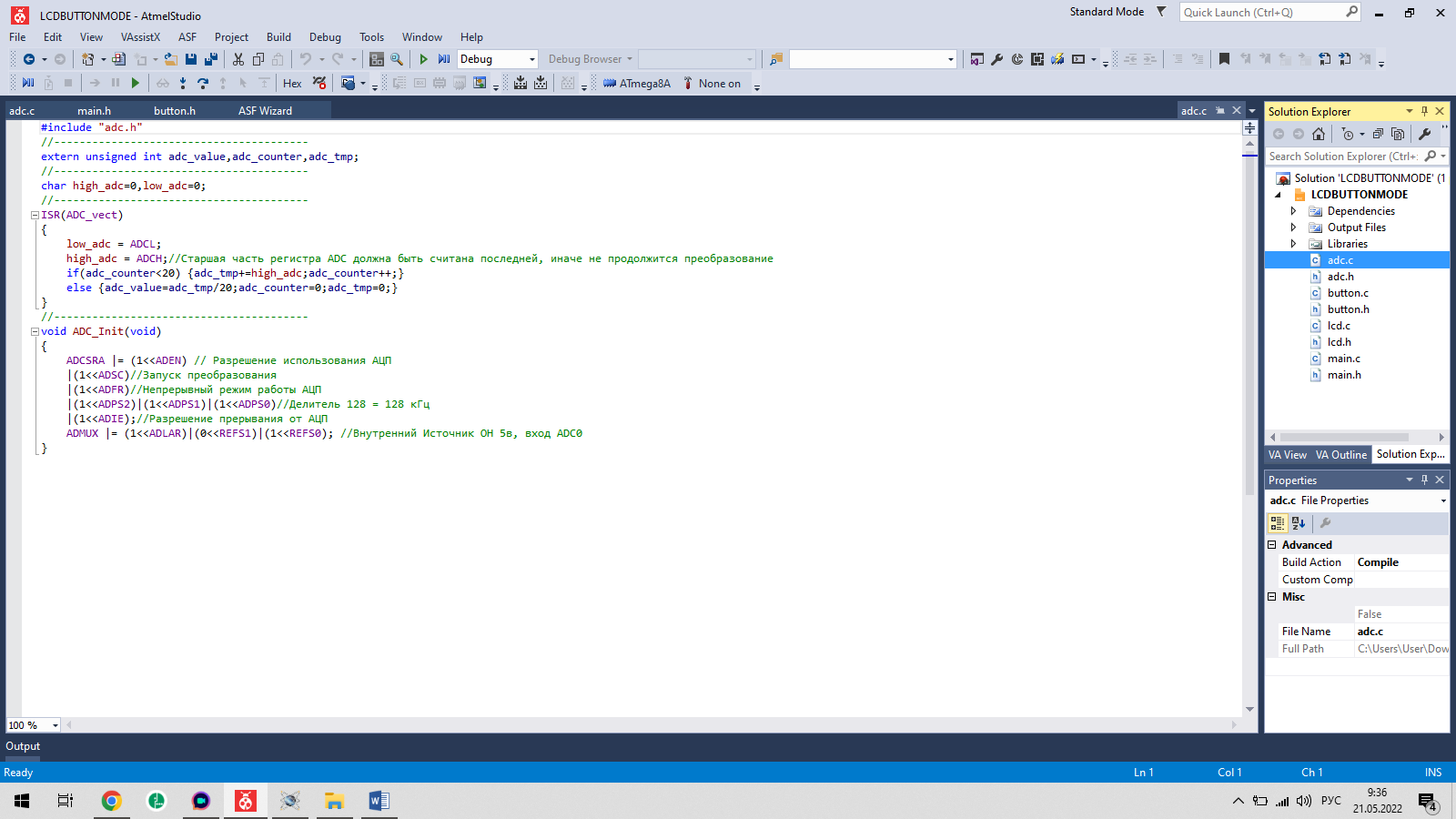
                }

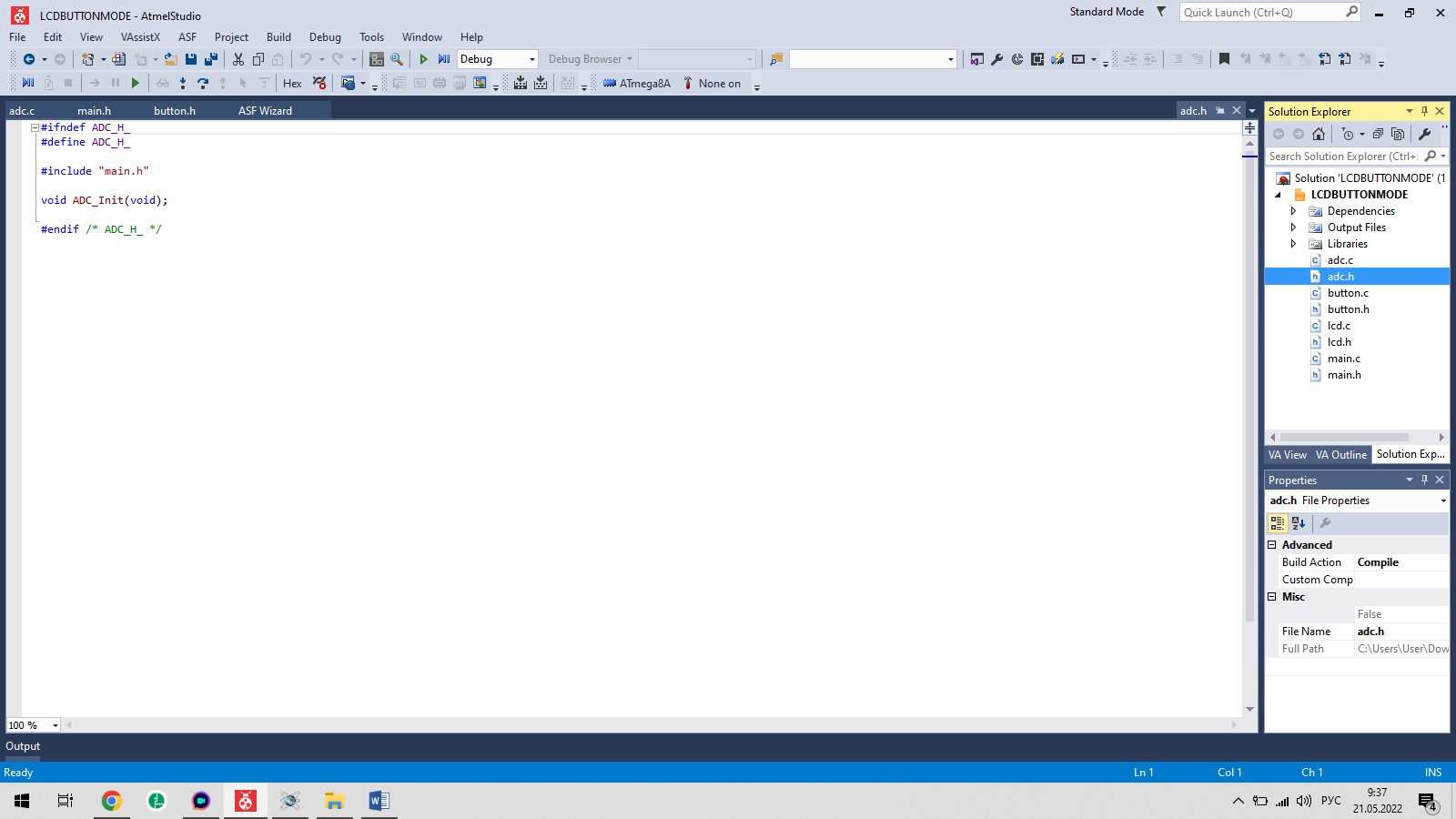
*\_delay\_ms*(50);

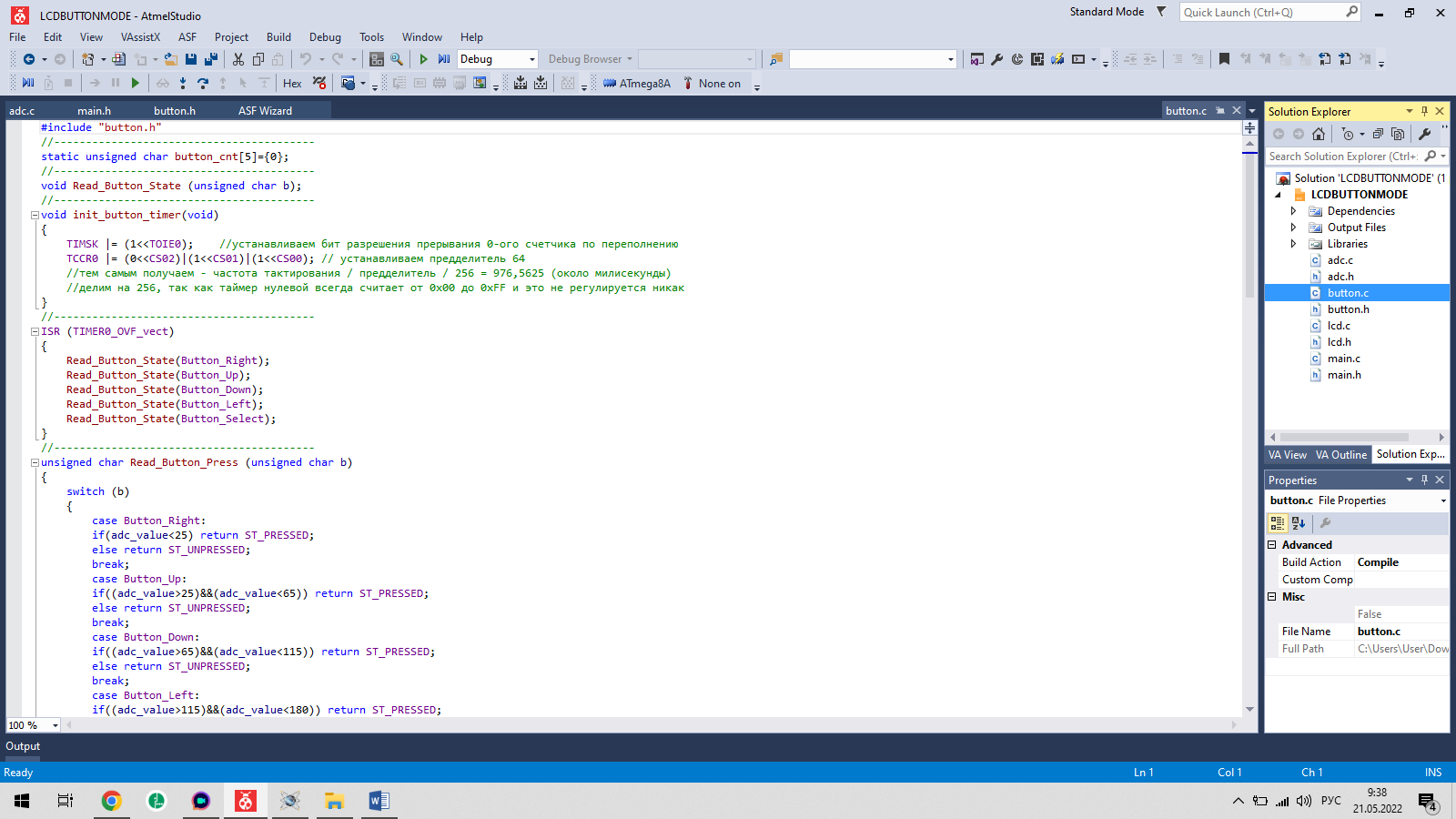
        }

Скомпилируем код прошьём контроллер и посмотрим результат. Если всё работает, то считаю цель занятия достигнутой.

**ПРОГРАММНЫЙ КОД ПРЕДСТАВЛЕН НИЖЕ**







#include "button.h"

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

static unsigned char button\_cnt[5]={0};

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

void Read\_Button\_State (unsigned char b);

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

void init\_button\_timer(void)

{

TIMSK |= (1<<TOIE0); //устанавливаем бит разрешения прерывания 0-ого счетчика по переполнению

TCCR0 |= (0<<CS02)|(1<<CS01)|(1<<CS00); // устанавливаем предделитель 64

//тем самым получаем - частота тактирования / предделитель / 256 = 976,5625 (около милисекунды)

//делим на 256, так как таймер нулевой всегда считает от 0x00 до 0xFF и это не регулируется никак

}

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

ISR (TIMER0\_OVF\_vect)

{

Read\_Button\_State(Button\_Right);

Read\_Button\_State(Button\_Up);

Read\_Button\_State(Button\_Down);

Read\_Button\_State(Button\_Left);

Read\_Button\_State(Button\_Select);

}

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

unsigned char Read\_Button\_Press (unsigned char b)

{

switch (b)

{

case Button\_Right:

if(adc\_value<25) return ST\_PRESSED;

else return ST\_UNPRESSED;

break;

case Button\_Up:

if((adc\_value>25)&&(adc\_value<65)) return ST\_PRESSED;

else return ST\_UNPRESSED;

break;

case Button\_Down:

if((adc\_value>65)&&(adc\_value<115)) return ST\_PRESSED;

else return ST\_UNPRESSED;

break;

case Button\_Left:

if((adc\_value>115)&&(adc\_value<180)) return ST\_PRESSED;

else return ST\_UNPRESSED;

break;

case Button\_Select:

if((adc\_value>180)&&(adc\_value<230)) return ST\_PRESSED;

else return ST\_UNPRESSED;

break;

}

return ST\_ERROR;

}

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

void Read\_Button\_State (unsigned char b)

{

if ((button\_state[b]&ST\_LOCKED)!=0) return;

if (Read\_Button\_Press(b)==ST\_UNPRESSED)

{

if (button\_cnt[b]>0)

{

button\_cnt[b]--;

}

else

{

if((button\_state[b]&ST\_PRESSED)!=0)

{

button\_state[b] |= ST\_UNPRESSURE;

button\_state[b] &= ~ST\_PRESSED;

button\_state[b] |= ST\_UNPRESSED;

}

}

}

else

{

if (button\_cnt[b]<5)

{

button\_cnt[b]++;

}

else

{

if((button\_state[b]&ST\_UNPRESSED)!=0)

{

button\_state[b] |= ST\_PRESSURE;

button\_state[b] &= ~ST\_UNPRESSED;

button\_state[b] |= ST\_PRESSED;

}

}

}

}

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

void ResetButtonState(unsigned char b, unsigned char st)

{

button\_state[b]&=~st;

}

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

void SetButtonState(unsigned char b, unsigned char st)

{

button\_state[b]|=st;

}

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

void Button\_ini(void)

{

unsigned char i;

//сбросим статусы кнопок

for (i=0;i<5;i++)

{

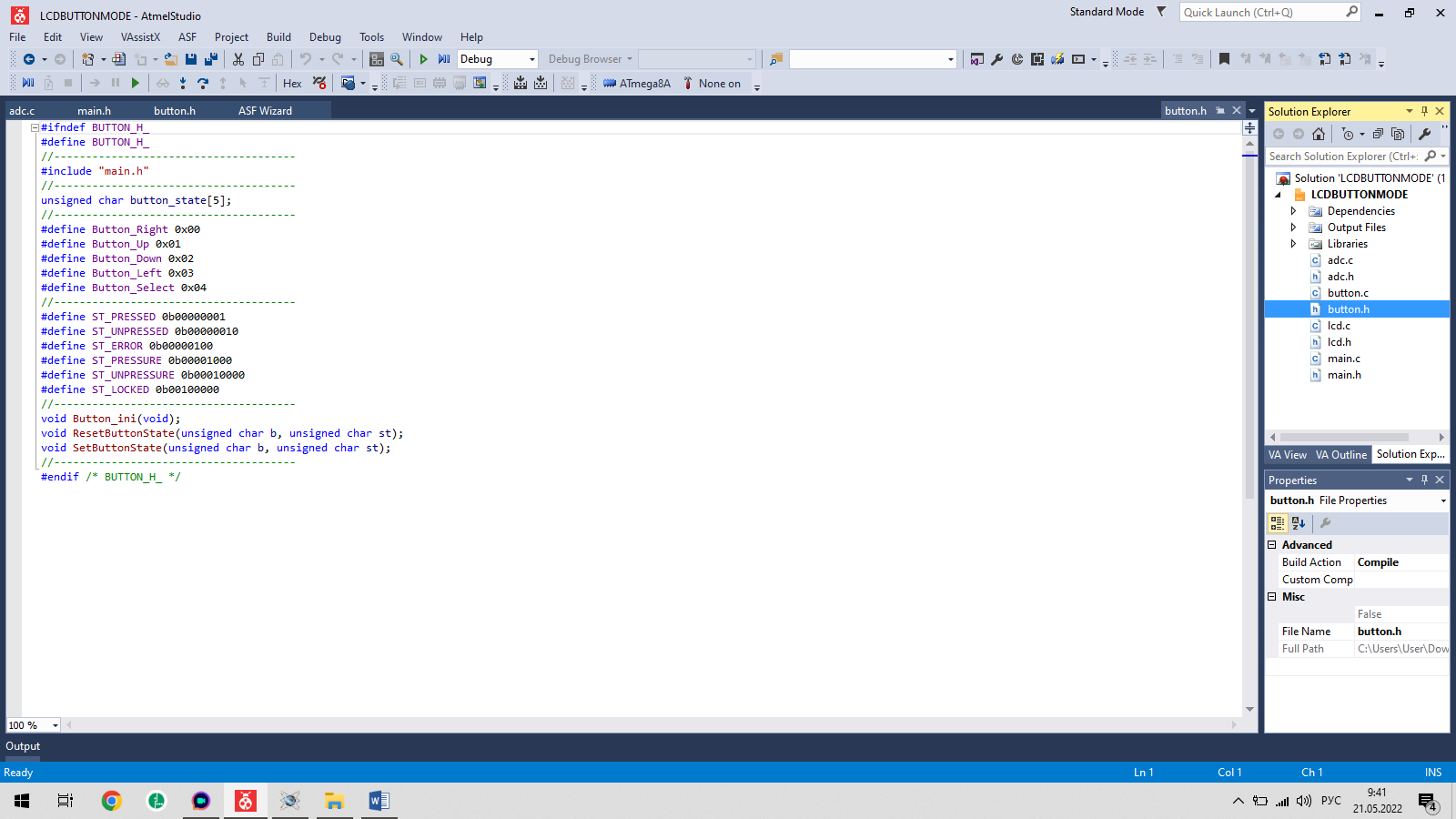
button\_state[i]=ST\_UNPRESSED;

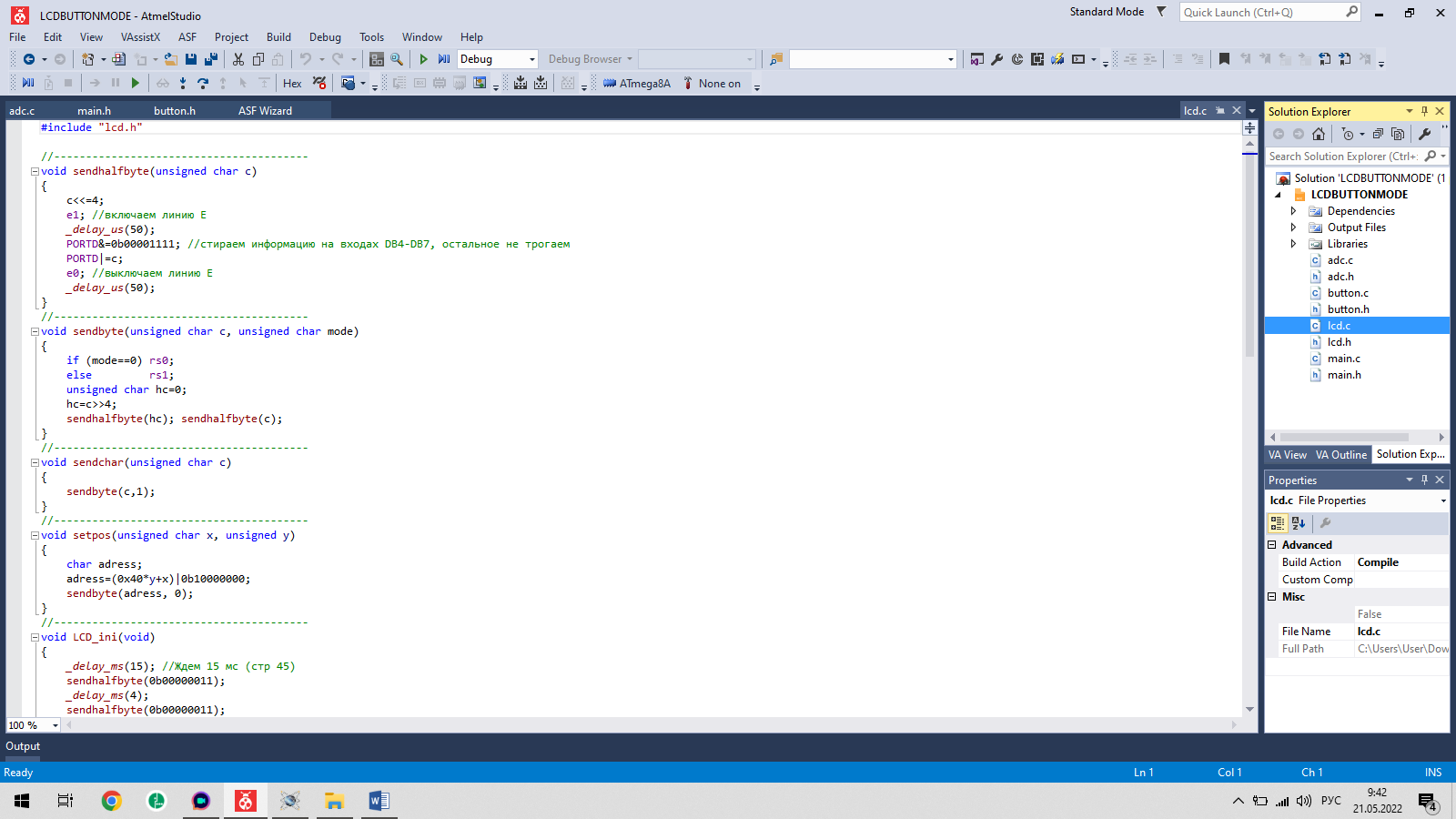
}

//запустим таймер

init\_button\_timer();

}





#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[])

{

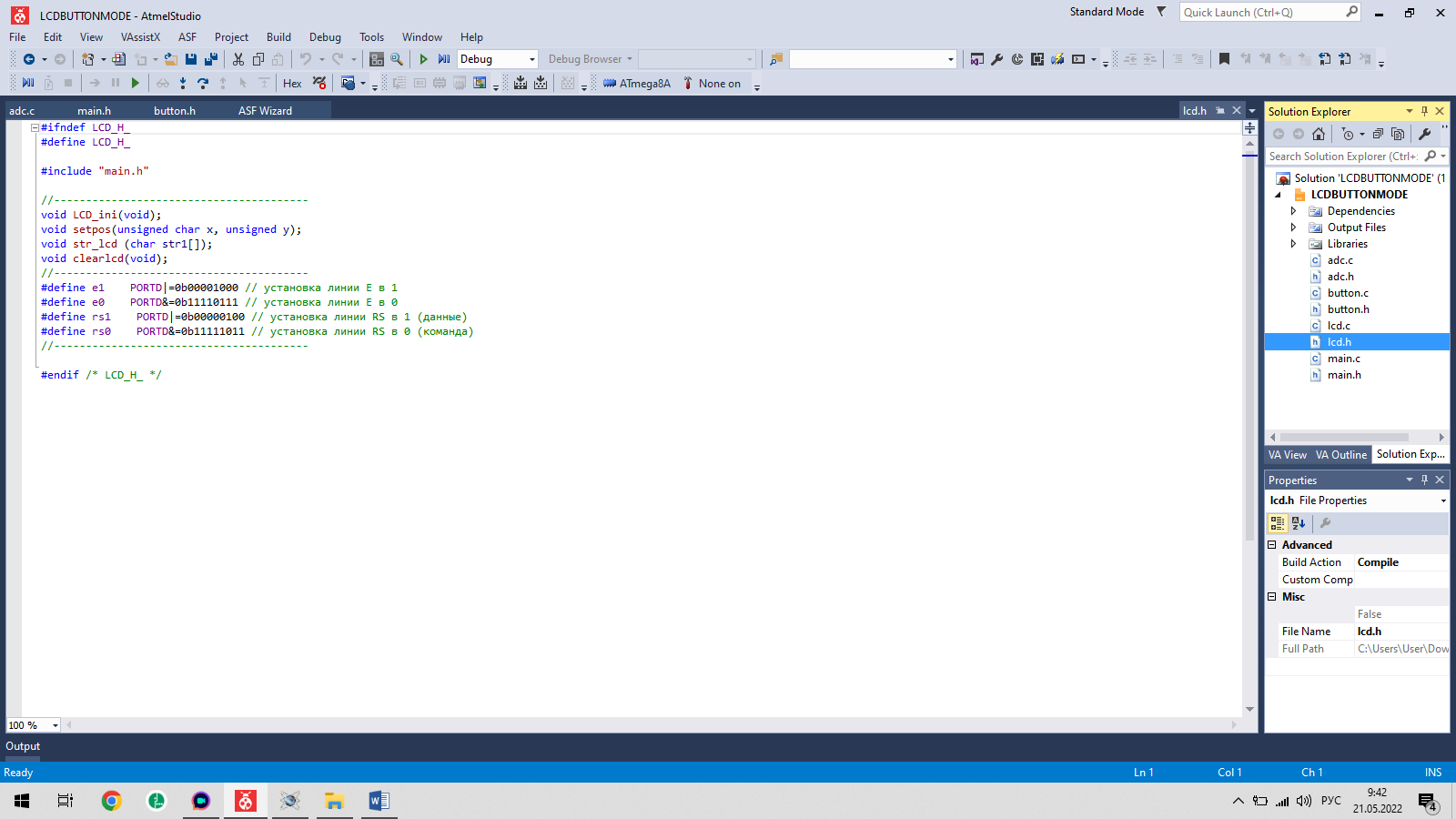
wchar\_t n;

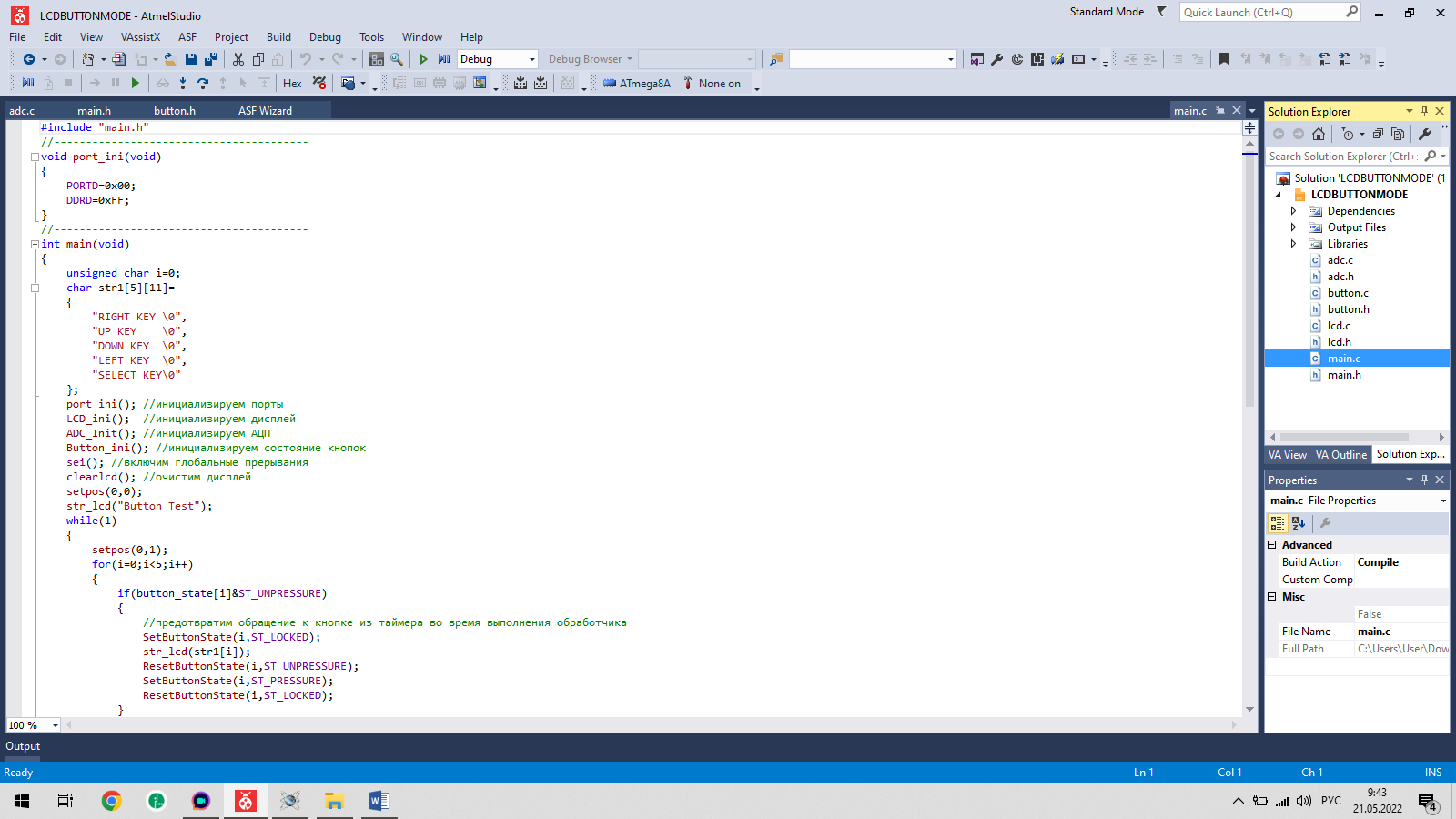
for(n=0;str1[n]!='\0';n++)

sendchar(str1[n]);

}

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





#include "main.h"

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

void port\_ini(void)

{

PORTD=0x00;

DDRD=0xFF;

}

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

int main(void)

{

unsigned char i=0;

char str1[5][11]=

{

"RIGHT KEY \0",

"UP KEY \0",

"DOWN KEY \0",

"LEFT KEY \0",

"SELECT KEY\0"

};

port\_ini(); //инициализируем порты

LCD\_ini(); //инициализируем дисплей

ADC\_Init(); //инициализируем АЦП

Button\_ini(); //инициализируем состояние кнопок

sei(); //включим глобальные прерывания

clearlcd(); //очистим дисплей

setpos(0,0);

str\_lcd("Button Test");

while(1)

{

setpos(0,1);

for(i=0;i<5;i++)

{

if(button\_state[i]&ST\_UNPRESSURE)

{

//предотвратим обращение к кнопке из таймера во время выполнения обработчика

SetButtonState(i,ST\_LOCKED);

str\_lcd(str1[i]);

ResetButtonState(i,ST\_UNPRESSURE);

SetButtonState(i,ST\_PRESSURE);

ResetButtonState(i,ST\_LOCKED);

}

}

*\_delay\_ms*(50);

}

}

