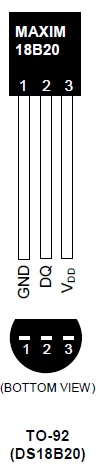
**Подключаем датчик температуры DS18B20**



И сегодняшним представителем будет **DS18B20**, который является датчиком температуры.

Измеряет температуру данный датчик в градусах по цельсию. Измерения могут быть как 9-битной дискретности, так и 12-битной.

Распределяются данные биты следующим образом.

Мы будем измерять температуру в 12-битном виде и самые младшие 4 бита будут хранить показания долей градуса, а остальные старшие 8 – сами градусы.

Подключается данный датчик по однопроводной технологии (1-wire).

Диапазон измерений – от -55 0C до +125 0C, но наивысшая точность показаний достигается в диапазоне от -10 0C до +85 0C/

Погрешность данного датчика – 0,5 градуса, поэтому нам нет смысла использовать все младшие биты и мы их будем просто отсекать.

Режим 1-wire у нас не организован аппаратно в контроллере, поэтому мы будем весь протокол программировать.

Чем удобен для нас данный датчик? Удобен он тем, что мы можем его в часовой модуль, использованный на прошлых занятиях просто впаять и использовать. Для этого на модуле выведен отдельный контакт. У датчика всего 3 ножки. 2 из них ножки питания и одна – ножка данных.

Данный датчик существует в двух видах корпуса – Dip и TO-92. Мы будем использовать второй тип.

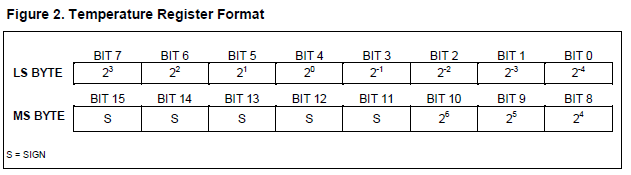
Таких датчиков, как пишут, на 1 провод можно повесить несколько, но я не пробовал.

В связи с этим у каждого датчика есть уникальный 64-битный код, записанный в его ROM, записанного постредством лазерных технологий.

Но, так как мы будем использовать только один датчик, то мы будем к нему обращаться другим способом, не используя ROM.

Мы подаём на датчик команду 44h последовательным кодом, тем самым заставляя датчик конвертировать температуру.

Посмотрим регистры, в которых хранится значение температуры после преобразования



В четырех младших битах младшего регистра хранятся доли градусов, в четырех старших, а также в трех младших битах старшего регистра – целые значения градусов.

В остальных битах – знак. Если 0, то плюс, если 1 – то минус.

Также в технической документации написано, что нужно обязательно подтянуть резистор на информационную ножку датчика к питанию.

Также написан ряд временных и других характеристик, которые мы уже будем рассматривать по мере того, как будем писать код.

Проект был создан новый и назван **My1820LCD**, все файлы проекта [**прошлого занятия**](https://narodstream.ru/avr-urok-19-sobiraem-chasy-na-mikrosxeme-ds1307-i-lcd-1602/) **MyClock1307LCD** были вставлены в проект, а код главного модуля также был скопирован в главный модуль нового проекта. Всё это сделано потому, что мы работаем с тем же модулем, просто мы также впаяли туда датчик температуры, а время мы считывать не перестанем, пусть часы также ходят.

Добавим ещё файлы **DS18B20.c** и **DS18B20.h** для созадния библиотеки работы именно с датчиком температуры и заголовочный файл также подключим в main.h по нашему обыкновению, ну и его же подключим в одноименный c-файл.

Сначала создадим функцию конвертирования температуры в файле DS18B20.c. Надобность данной функции обусловлена тем, что датчик нам передаёт показания в двух байтах в определённом виде, рассмотренном нами выше, который без преобразования будет непонятен

//функция преобразования показаний датчика в температуру

**int dt\_check(void)**

**{**

**}**

Создадим две переменных, которые нам пригодятся в дальнейшем в теле нашей функции

int dt\_check(void)

{

**unsigned char bt;//переменная для считывания байта**

**unsigned int tt=0;**

Сразу сделаем прототип нашей функции в заголовочном файле

#include "main.h"

**int dt\_check(void); //функция преобразования показаний датчика в температуру**

Чтобы нам дальше писать тело функции, нам будет нужна ещё функция. Напишем её выше, чтобы не создавать прототип. Данная функция будет узнавать, есть ли датчик на шине

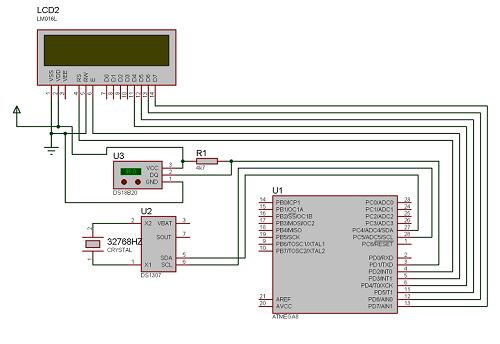
//функция определения датчика на шине

**char dt\_testdevice(void) //dt – digital termomether | определим, есть ли устройство на шине**

**{**

**}**

Теперь, прежде чем писать тело данной функции, самое время посмотреть схему подключения датчика к контроллера. Посмотрим опять же упрощённый вариант в протеусе (нажмите на картинку для увеличения изображения)

[](https://narodstream.ru/wp-content/uploads/2016/12/Image02-11.png)

Мы видим, что датчик своим информационным портом подключен к ножке порта МК PD1, также мы видим подтягивающее сопротивление на 4,7 килоом с данного провода на шину питания.

И вот теперь на данной ножке мы должны как-то иммитировать определённые имплуьсы. То есть мы как-то должны инициализировать то ноль, то единицу. Делать мы это сможем благодаря тому, что у нас есть подтягивающий резистор. Мы будем её то отпускать от общей шины, то притягивать к ней. Соответственно, когда мы её отпустим, то через резистор на ней окажется высокий уровень, а когда подтянем к земле, то низкий.

Теперь вопрос, как это всё сделать. А сделать это можно вполне с помощью определённого бита регистра DDRB, который отвечает, как мы знаем, за направление работы определённых ножек порта. Но мы воспользуемся тем, что когда мы объявляем ножку на выход, она у нас притягивается к земле, понятно что при условии, что соответствующий бит регистра PORTD будет также нулём. А если мы уже в соответствующий бит регистра DDRB записываем значение, при котором соответствующая этому биту ножка объявится на вход, то, соответственно, от общей шины она. ясное дело отпустится, иначе как контроллер определит её состояние. Ну и, как и было написано выше, на этой ножке засчет подтягивающего резистора установится высокий логический уровень.

Также, начиная процесс обмена с датчиком, мы должны запретить прерывания, чтобы не получить искаженных данных. Также, прежде чем запретить прерывания, нам необходимо сохранить регистр статуса контроллера, или стек. Это регистр SREG. Поэтому в самом начале тела нашей функции проверки присутствии датчика на шине мы это и сделаем, сохранив его в определённую переменную, ну и затем уже запретим прерывания

char dt\_testdevice(void) //dt – digital termomether | определим, есть ли устройство на шине

{

**char stektemp=SREG;// сохраним значение стека**

**cli(); //запрещаем прерывания**

Добавим ещё одну переменную в теле данной функции для возврата результата функции

cli(); //запрещаем прерывание

**char dt;**

Теперь для удобства работы с ножками порта давайте напишем некоторые дефайновые замены в заголовочном файле DS18B20.h

#include "main.h"

**#define PORTTEMP PORTD**

**#define DDRTEMP DDRD**

**#define PINTEMP PIND**

**#define BITTEMP 1**

Также здесь давайте сделаем ещё некоторые макроподставновки

#include "main.h"

**#define NOID 0xCC //Пропустить идентификацию**

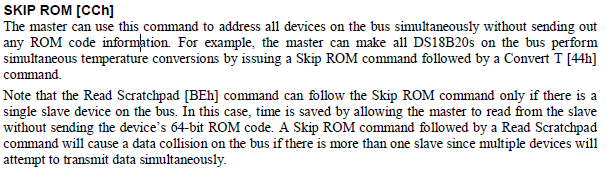
**#define T\_CONVERT 0x44 //Код измерения температуры**

**#define READ\_DATA 0xBE //Передача байтов ведущему**

Здесь мы присваиваем удобные имена командам датчикам.

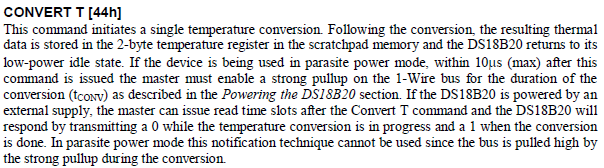
Ну давайте их посмотрим.

Первая команда **0xCC**



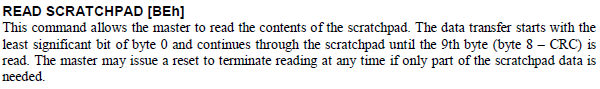
Она заставляет датчик отойти от стандартной процедуры инициализации и позволяет пропутсить считыванеие ROM для определения идентификатора. Нам идентификатор не нужен, так как датчик данный на шине у нас присутствует в единственном экземпляре.

Следующая команда **0x44**

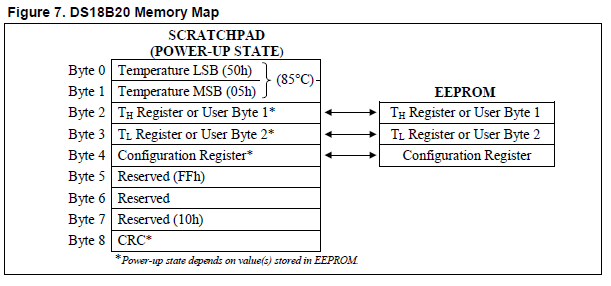


Данная команда инициализирует начало конвертирования данных в цифровой код.

Следующая команда **0xBE**

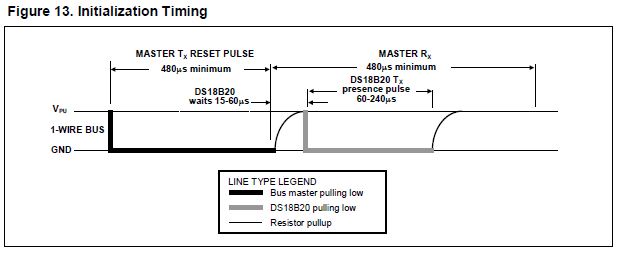


Данная команда читает регистры датчика. Вот эти регистры



Вот так вооще организована вся память в датчике измеерния температуры, и как мы видим, самые младшие два регистра в данной памяти и представляют собой два байта показаний температуры.

Посмотрим теперь все тайминги инициализации датчика



Здесь видно что мы здесь притягиваем шину, дежим её в таком состоянии как минимум 480 микросекунд, затем отпускаем её, ждём максимум 60 мкс (мы подождём побольше – 65 мкс) ответа датчика, датчик должен нам ответить таким же притягиванием шины к земле за данное время. Если обнаружится ноль, то значит устройство на шине есть, а если шина так и останется висеть в воздухе, то данного датчика на шине нет.

Ну. и на остове вышепрочитанного продолжим нашу функцию определения наличия датчика на шине

char dt;

**DDRTEMP |= 1<<BITTEMP; //притягиваем шину**

**\_delay\_us(485); //задержка как минимум на 480 микросекунд**

**DDRTEMP &= ~(1<<BITTEMP); //отпускаем шину**

**\_delay\_us(65); //задержка как минимум на 60 микросекунд**

**if ((PINTEMP & (1<<BITTEMP))==0)//проверяем, ответит ли устройство**

**{**

**dt=1;//устройство есть**

**}**

**else dt=0;//устройства нет**

В условии у нас идёт проверка, опустилась ли шина.

Затем мы стеку вернём его первоначальное состояние

else dt=0;//устройства нет

**SREG = stektemp;// вернем значение стека**

Отдельной командой после этого разрешать прерывания не требуется, так как все эти команды как раз и управляют определёнными битами регистра **SREG**. Если прерывания были разрешены до сохранения стека, то они, ясное дело, опять разрешатся, так как все биты регистра мы возвращаем в прежнее положение.

Дальше. судя по диаграмме, которая у нас находится выше, мы должны выдержать время максимум 480 микросекунд, но так как после последней задержки у нас уже прошло некоторое количество времени, то хватит и 420, тем более задержка эта не фиксированная, а максимальная. Ну, и добавим эту задержку в тело функции далее, и в оконцовке вернём результат функции.

**SREG = stektemp;// вернем значение стека**

**\_delay\_us(420); //задержка как минимум на 480 микросекунд, но хватит и 420, тк это с учетом времени прошедших команд**

**return dt; //вернем результат**

}

Теперь по окончании написания кода данной функции, мы можем продолжить начатую в предыдущей части урока функцию преобразования. Но, всё-таки, чтобы её продолжить, нам нужна будет ещё одна немаловажная, а может и вообще самая важная функция в нашем проекте – функция считывания байта с датчика температуры. Создадим данную функцию, а также в её теле сразу объявим две локальные переменные

**//функция чтения байта с устройства**

**unsigned char dt\_readbyte(void)**

**{**

**char c=0;**

**char i;**

**}**

Прежде чем писать дальше тело данной функции, мы должны вспомнить что каждый байт у нас состоит из восьми бит, поэтому целесообразно для того, чтобы не нагромождать много кода в одну функцию, создать ещё одну выше для чтения отдельного бита с датчика температуры. Создадим пока каркас данной функции с комментарием, объясняющем назначение этой функции

**//функция чтения бита с устройства**

**char dt\_readbit(void)**

**{**

**}**

В данной функции мы также сохраним регистр статусов микроконтроллера, запретим прерывания и создадим переменную для дальнейшего возврата результата. Результат у нас будет в виде обычной знаковой 8-битной величины, по большому счёту мы в ней будем использовать только младший бит

//функция чтения бита с устройства

char dt\_readbit(void)

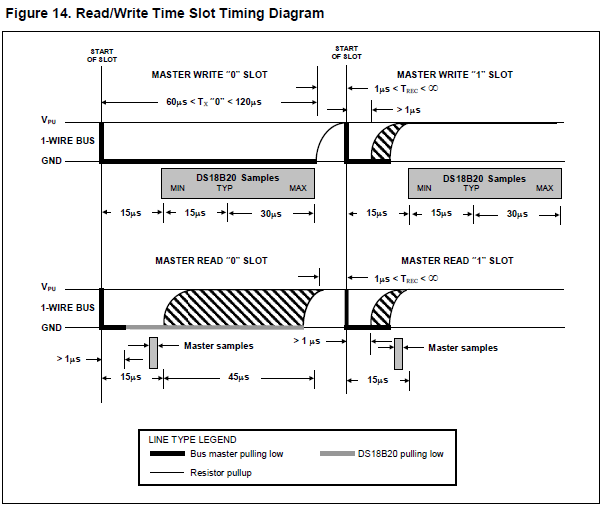
{

**char stektemp=SREG;// сохраним значение стека**

**cli(); //запрещаем прерывание**

**char bt; //переменная хранения бита**

Для написания дальнейшего кода посмотрим временную диаграмму, как вообще читать бит с данного датчика. Данная диаграмма в технической документации универсальна и для чтения и для записи бита в устройство



Мы здесь видим, что после притягивания шины к земле мы ждём минимум одну микросекунду, ну, подождём две микросекунды на всякий случай, перед этим притянув шину к земле

char bt; //переменная хранения бита

**DDRTEMP |= 1<<BITTEMP; //притягиваем шину**

**\_delay\_us(2); //задержка как минимум на 2 микросекунды**

Смотрим дальше. Мы должны шину отпустить от земли, а затем задержка для считывания результата. Здесь стоит общая задержка 15 микросекунд, так как мы уже две подождали, осталось нам 13

\_delay\_us(2); //задержка как минимум на 2 микросекунды

**DDRTEMP &= ~(1<<BITTEMP); //отпускаем шину**

**\_delay\_us(13);**

После данной задержки результат уже на шине должен быть. Считаем его аналогично, как в предыдущей функции мы узнавали присутствие усройства на шине

\_delay\_us(13);

**bt = (PINTEMP & (1<<BITTEMP))>>BITTEMP; //читаем бит**

До считывания следующего байта, судя по диаграмме мы ждём 45 микросекунд, затем возвращаем значение стека в стек и возвращаем результат функции считывания бита на шине

**bt = (PINTEMP & (1<<BITTEMP))>>BITTEMP; //читаем бит**

**\_delay\_us(45);**

**SREG = stektemp;// вернем значение стека**

**return bt; //вернем результат**

}

Теперь, используя данную функцию, допишем тело функции считывания с датчика целого байта, последовательно занося байты в переменную **c**, применяя битовый сдвиг, а затем ещё же и возвращая.

**char i;**

**for(i=0;i<8;i++)**

**c|=dt\_readbit()<<i; //читаем бит**

**return c;**

}

Ну вот. Самая, можно сказать, важная функция написана.

Продолжим функцию конвертирования считанных байтов

**unsigned int tt=0;**

**if(dt\_testdevice()==1) //если устройство нашлось**

**{**

**}**

**return tt;**

}

В теле условия по определению присутствия датчика на шине, прежде чем что-то считать с нашего датчика, мы должны ему будем передать команды.

давайте напишем функцию передачи байта в датчик, в её теле мы созадим локальную переменную для дальнейшего цикла передачи каждого байта

**//функция записи байта на устройство**

**void dt\_sendbyte(unsigned char bt)**

**{**

**char i;**

**}**

Ну и, соответственно, чтобы нам передать байт, нам нужно сначала научиться передавать бит, так как без бита не бывает байта. Для этого выше создадим другую функцию, написав её теле пока стандартный код сохранения стека и запрета прерываний

**//функция записи бита на устройство**

**void dt\_sendbit(char bt)**

**{**

**char stektemp=SREG;// сохраним значение стека**

**cli(); //запрещаем прерывание**

**}**

Затем смотрим в диаграмму записи бита в устройство, которую мы смотрели в даташите датчика в предыдущей части. В ней видно, что мы должны также притянуть шину к земле, подождать минимум 1 микросекунду, мы опять же подождём две. Это нужно для того, чтобы датчик "успел понять" что шина притянулась.

cli(); //запрещаем прерывание

**DDRTEMP |= 1<<BITTEMP; //притягиваем шину**

**\_delay\_us(2); //задержка как минимум на 2 микросекунды**

Дальше мы уже будем решать, что именно мы будем посылать. если будем посылать 0, то мы шину отпускать не будем, а, если нам нужно послать 1, то будем шину отпускать

\_delay\_us(2); //задержка как минимум на 2 микросекунды

**if(bt)**

**DDRTEMP &= ~(1<<BITTEMP); //отпускаем шину**

Дальше мы ждём от 60 до 120 микросекунд, ну, подожём 65, как обычно. То есть если мы предаём ноль, то мы просто выжидаем с притянутой шиной данный временной интервал, а если передаём единицу, то делаем то же самое, но отпустив перед этим шину, так как сработает условие в коде. Затем мы отпускаем шину. То есть, если мы передаём единицу, то она уже у нас отпущена, если ноль, то отпускаем. ну и затем возвращаем сохраненное значение в стек

  if(bt)

      DDRTEMP &= ~(1<<BITTEMP); //отпускаем шину

**\_delay\_us(65); //задержка как минимум на 60 микросекунд**

**DDRTEMP &= ~(1<<BITTEMP); //отпускаем шину**

**SREG = stektemp;// вернем значение стека**

}

Теперь продолжим, используя данную функцию, функцию записи (или отправки) байта в шину

  char i;

**for(i=0;i<8;i++)//посылаем отдельно каждый бит на устройство**

**{**

**if((bt & (1<<i)) == 1<<i)//посылаем 1**

**dt\_sendbit(1);**

**else //посылаем 0**

**dt\_sendbit(0);**

**}**

}

Здесь мы также используем цикл и в цикле мы, используя также битовый сдвиг, постепенно по одному биту считываем посланный в функцию в качестве входного аргумента байт и посылаем каждый считанный бит в функцию передачи бита в виде переменной, равной 0 или 1.

Теперь, написав все эти рутинные функции, мы смело можем продолжить так и не дописанную в предыдущих частях функцию конвертирования показания температуры.

Как мы помним, мы написали в данной функции код условия наличия датчика на шине, и теперь можем писать тело данного условия. Начнём с передачи команды разрешения на то, чтобы нам не считывать идентификатор датчика

if(dt\_testdevice()==1) //если устройство нашлось

{

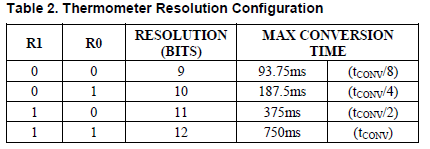
**dt\_sendbyte(NOID); //пропустить идентификацию, тк у нас только одно устройство на шине**

Затем мы уже отправляем команду на считывание температуры

dt\_sendbyte(NOID); //пропустить идентификацию, тк у нас только одно устройство на шине

**dt\_sendbyte(T\_CONVERT); //измеряем температуру**

Данная команда – это только команда, а не само считывание температуры. По данной команде датчик начнёт измерение температуры и конвертацию её в цифровой код и занесение его в регистровую пару. Данный процесс у датчика длится значительно долго, около 750 милисекунд, не микро. Данное время необходимо, чтобы датчик успел записать все 12 бит



Здесь мы видим, что если нам особая точность не нужна, то мы можем считать другое количество бит, тем самым сэкономив значительное количество времени. Но мы напишем идеально 750, так как мы никуда не спешим

dt\_sendbyte(T\_CONVERT); //измеряем температуру

**\_delay\_ms(750); //в 12битном режиме преобразования – 750 милисекунд**

Далее мы опять проделываем те же манипуляции с шиной. как и при проверке наличия устройства на ней, и опять посылаем команду работы без идентификации

\_delay\_ms(750); //в 12битном режиме преобразования – 750 милисекунд

**dt\_testdevice(); //снова используем те же манипуляции с шиной что и при проверке ее присутствия**

**dt\_sendbyte(NOID); //пропустить идентификацию, тк у нас только одно устройство на шине**

И затем мы уже даем команду датчику считывать байты с регистра. Судя по временным диаграммам, они там уже должны быть

dt\_sendbyte(NOID); //пропустить идентификацию, тк у нас только одно устройство на шине

**dt\_sendbyte(READ\_DATA); //даем команду на чтение данных с устройства**

Затем мы считываем байты с регистра и укладываем их в переменную tt, потом, выйдя из тела условия, возвращаем их.

    dt\_sendbyte(READ\_DATA); //даем команду на чтение данных с устройства    bt = dt\_readbyte(); //читаем младший бит

**tt = dt\_readbyte(); //читаем старший бит MS**

**tt = (tt<<8)|bt;//сдвигаем старший влево, младший пишем на его место, тем самым получаем общий результат**

**}**

**return tt;**

**}**

Мы получили хоть и не раздельный результат в виде двух различных байтов, но все равно результат этот сырой, так как в нём есть и биты знака, и также шестнадцатые доли градуса, которые мы не можем показать на дисплее. Поэтому нужна будет ещё одна функция конвертирования результата в удобный для глаза вид.

Мы вообще упростим нашу задачу и не будем использовать доли, а будем использовать только целые показания, и бит отрицания мы также не будем использовать, мы будем измерять только положительные температуры. Вот такая поэтому будет простая у нас функция

**//преобразование температуры в единицы**

**char converttemp (unsigned int tt)**

**{**

**char t = tt>>4;//сдвиг и отсечение части старшего байта**

**return t;**

**}**

Напишем на данную функцию прототип в хедер-файле

int dt\_check(void); //функция преобразования показаний датчика в температуру

**char converttemp (unsigned int tt); //преобразование температуры в единицы**

Теперь мы, наконец-то вернёмся в наш главный модуль в его главную функцию **main()**, а то мы уже тут очень давно не были. Создадим здесь локальную переменную для хранения показаний температуры

int main(void)

{

**unsigned int tt=0; //переменная для хранения температуры**

Напишем код для вызова функции измерения температуры в бесконечном цикле

sendcharlcd(sec%10+0x30);//Преобразуем число в код числа

sendcharlcd(' ');

**tt = converttemp(dt\_check()); //измеряем температуру**

Отобразим показания температуры на дисплее

  tt = converttemp(dt\_check()); //измеряем температуру

**sendcharlcd(tt/10+0x30);//Преобразуем число в код числа**

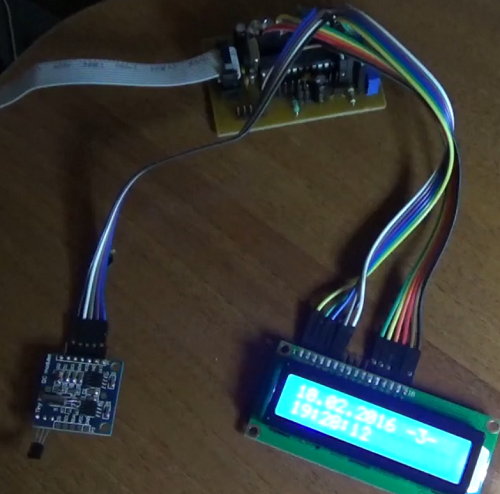
**sendcharlcd(tt%10+0x30);//Преобразуем число в код числа**

**sendcharlcd('\*');**

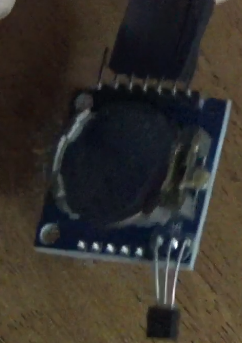
**sendcharlcd('C');**

}

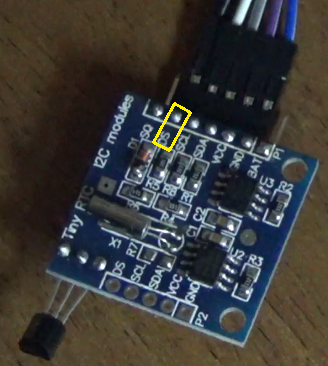
Прежде чем собирать код и прошивать контроллер, посмотрим подключение нашей всей схемы (нажмите на картинку для увеличения изображения)

[](https://narodstream.ru/wp-content/uploads/2016/12/Image10-10.png)

Посмотрим также поближе модуль с припаянным датчиком



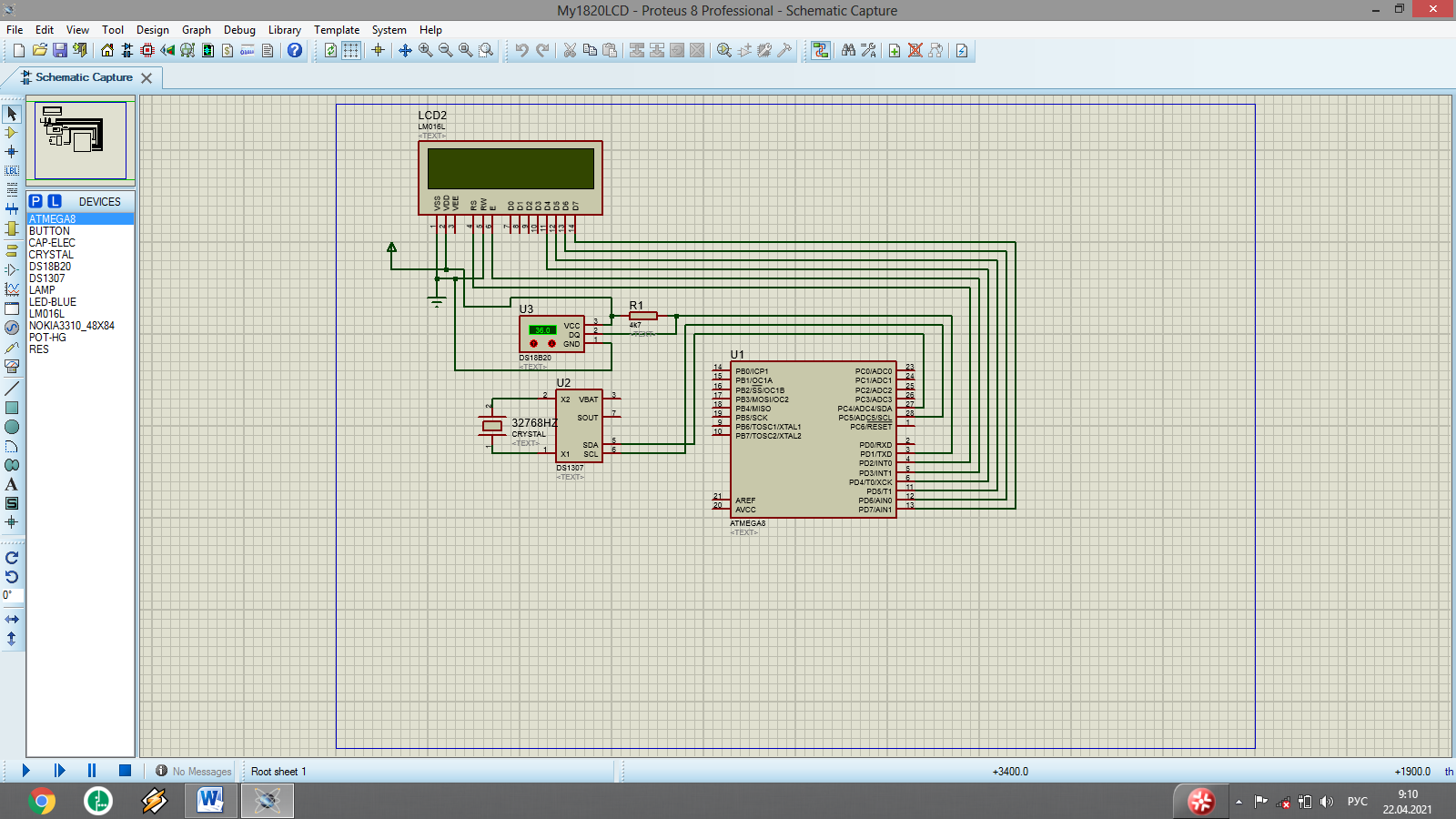
Вот так выглядит модуль с другой стороны, желтым прямоугольником отмечен контакт данных термодатчика

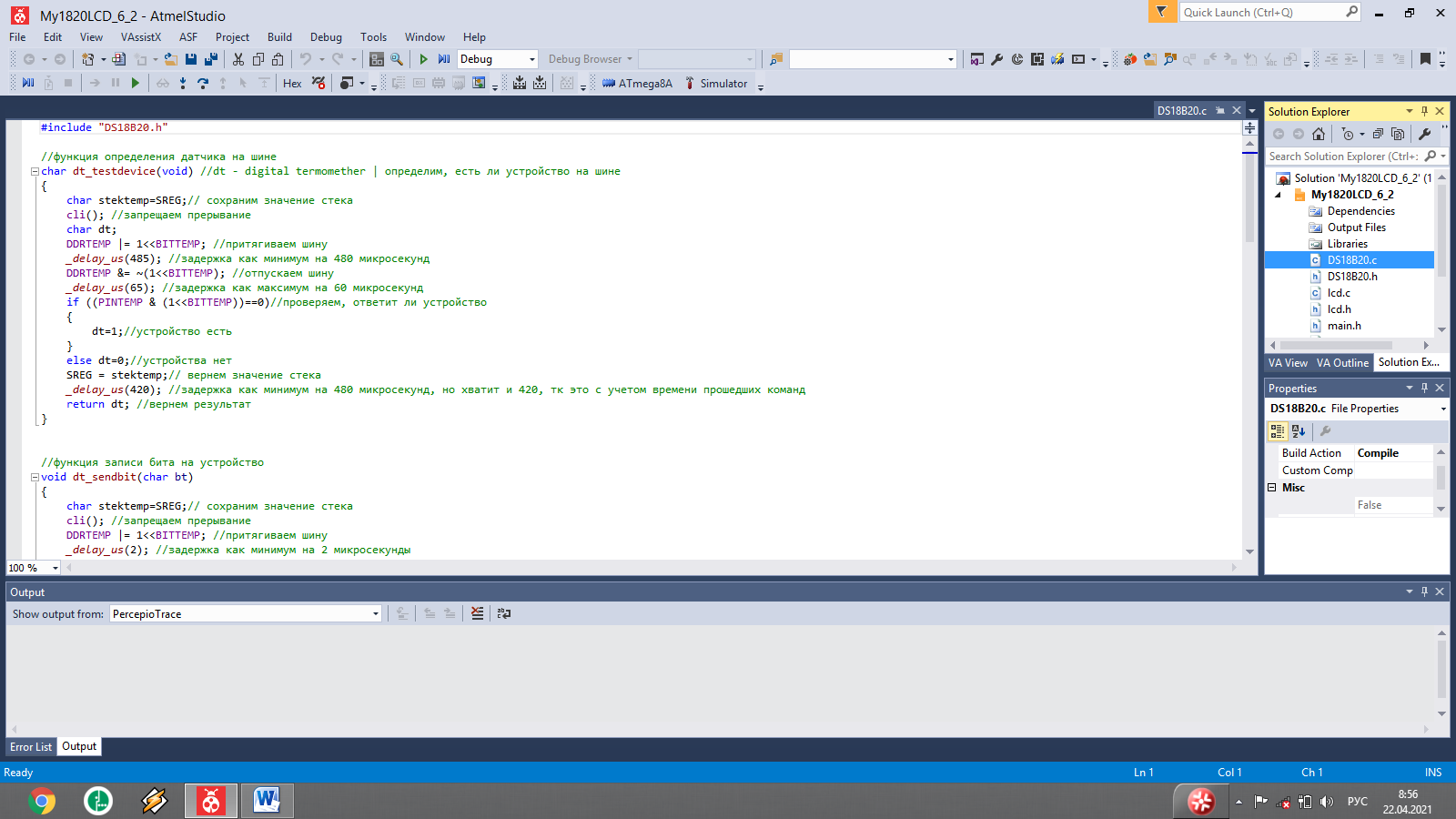


Соберём код и прошьём контроллер. А вот и результат



Так как при измерении температуры у нас происходит очень большая задержка, а мы всё равно отсекаем считанные биты, впоследствии мы. когда будем писать часы с термометром, то мы данную задержку значительно убавим.





#include "DS18B20.h"

//функция определения датчика на шине

char dt\_testdevice(void) //dt - digital termomether | определим, есть ли устройство на шине

{

char stektemp=SREG;// сохраним значение стека

cli(); //запрещаем прерывание

char dt;

DDRTEMP |= 1<<BITTEMP; //притягиваем шину

*\_delay\_us*(485); //задержка как минимум на 480 микросекунд

DDRTEMP &= ~(1<<BITTEMP); //отпускаем шину

*\_delay\_us*(65); //задержка как максимум на 60 микросекунд

if ((PINTEMP & (1<<BITTEMP))==0)//проверяем, ответит ли устройство

{

dt=1;//устройство есть

}

else dt=0;//устройства нет

SREG = stektemp;// вернем значение стека

*\_delay\_us*(420); //задержка как минимум на 480 микросекунд, но хватит и 420, тк это с учетом времени прошедших команд

return dt; //вернем результат

}

//функция записи бита на устройство

void dt\_sendbit(char bt)

{

char stektemp=SREG;// сохраним значение стека

cli(); //запрещаем прерывание

DDRTEMP |= 1<<BITTEMP; //притягиваем шину

*\_delay\_us*(2); //задержка как минимум на 2 микросекунды

if(bt)

DDRTEMP &= ~(1<<BITTEMP); //отпускаем шину

*\_delay\_us*(65); //задержка как минимум на 60 микросекунд

DDRTEMP &= ~(1<<BITTEMP); //отпускаем шину

SREG = stektemp;// вернем значение стека

}

//функция записи байта на устройство

void dt\_sendbyte(unsigned char bt)

{

char i;

for(i=0;i<8;i++)//посылаем отдельно каждый бит на устройство

{

if((bt & (1<<i)) == 1<<i)//посылаем 1

dt\_sendbit(1);

else //посылаем 0

dt\_sendbit(0);

}

}

//функция чтения бита с устройства

char dt\_readbit(void)

{

char stektemp=SREG;// сохраним значение стека

cli(); //запрещаем прерывание

char bt; //переменная хранения бита

DDRTEMP |= 1<<BITTEMP; //притягиваем шину

*\_delay\_us*(2); //задержка как минимум на 2 микросекунды

DDRTEMP &= ~(1<<BITTEMP); //отпускаем шину

*\_delay\_us*(13);

bt = (PINTEMP & (1<<BITTEMP))>>BITTEMP; //читаем бит

*\_delay\_us*(45);

SREG = stektemp;// вернем значение стека

return bt; //вернем результат

}

//функция чтения байта с устройства

unsigned char dt\_readbyte(void)

{

char c=0;

char i;

for(i=0;i<8;i++)

c|=dt\_readbit()<<i; //читаем бит

return c;

}

//функция преобразования показаний датчика в температуру

int dt\_check(void)

{

unsigned char bt;//переменная для считывания байта

unsigned int tt=0;

if(dt\_testdevice()==1) //если устройство нашлось

{

dt\_sendbyte(NOID); //пропустить идентификацию, тк у нас только одно устройство на шине

dt\_sendbyte(T\_CONVERT); //измеряем температуру

*\_delay\_ms*(750); //в 12битном режиме преобразования - 750 милисекунд

dt\_testdevice(); //снова используем те же манипуляции с шиной что и при проверке ее присутствия

dt\_sendbyte(NOID); //пропустить идентификацию, тк у нас только одно устройство на шине

dt\_sendbyte(READ\_DATA); //даем команду на чтение данных с устройства

bt = dt\_readbyte(); //читаем младший бит

tt = dt\_readbyte(); //читаем старший бит MS

tt = (tt<<8)|bt;//сдвигаем старший влево, младший пишем на его место, тем самым получаем общий результат

}

return tt;

}

//преобразование температуры в единицы

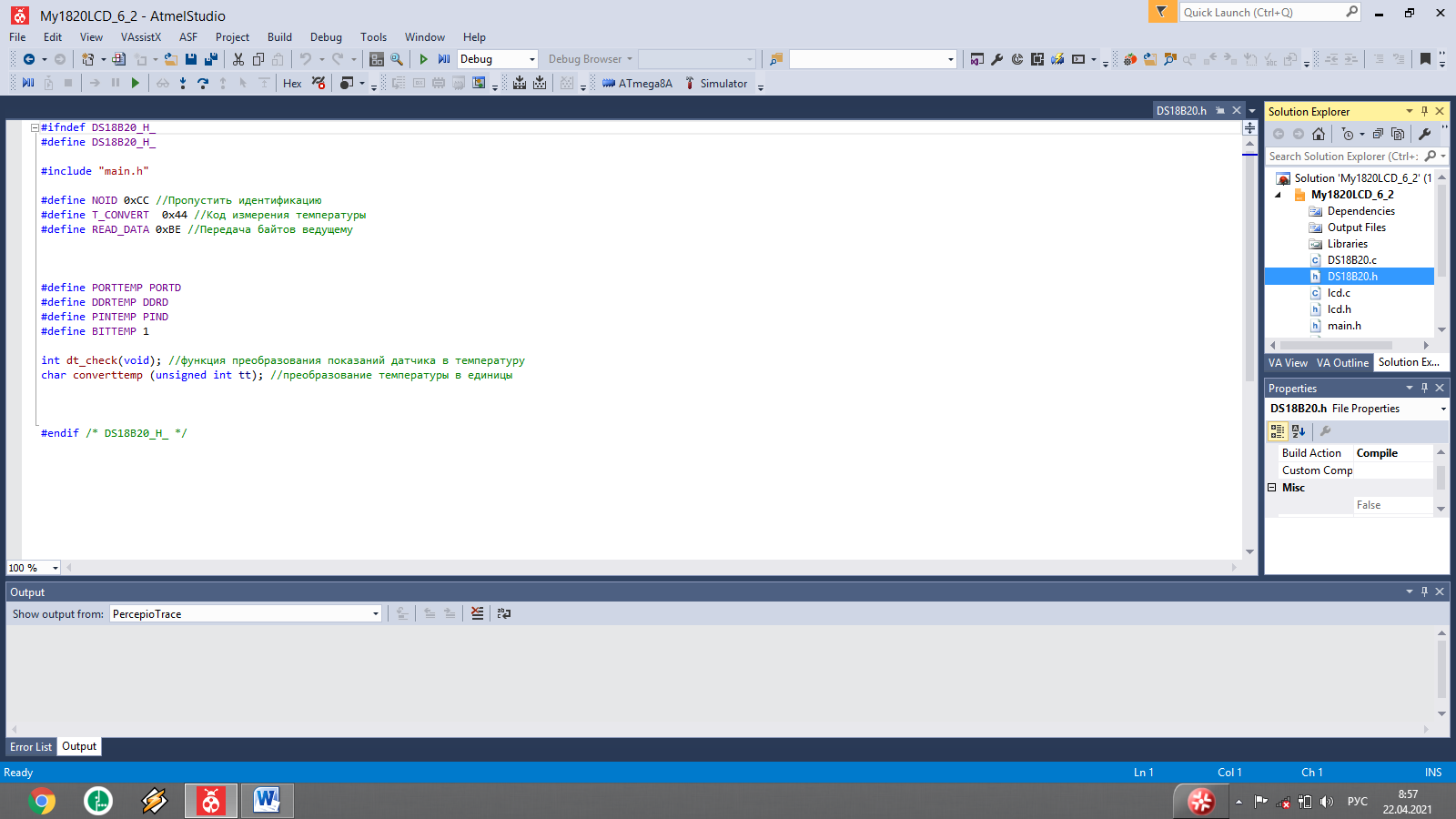
char converttemp (unsigned int tt)

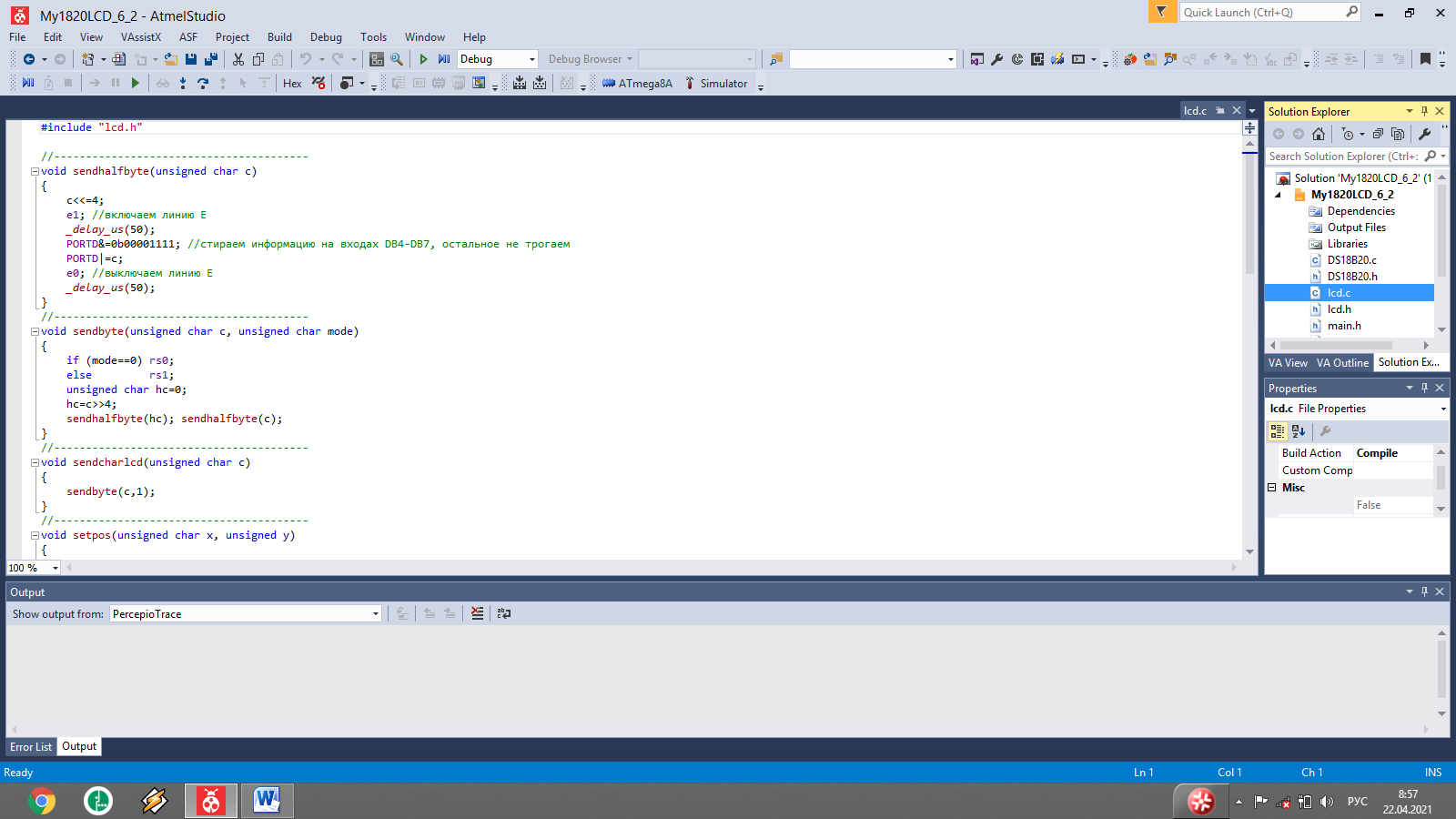
{

char t = tt>>4;//сдвиг и отсечение части старшего байта

return t;

}





#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 sendcharlcd(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(void)

{

sendbyte(0b00000001, 0);

*\_delay\_us*(1500);

}

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

void str\_lcd (char str1[])

{

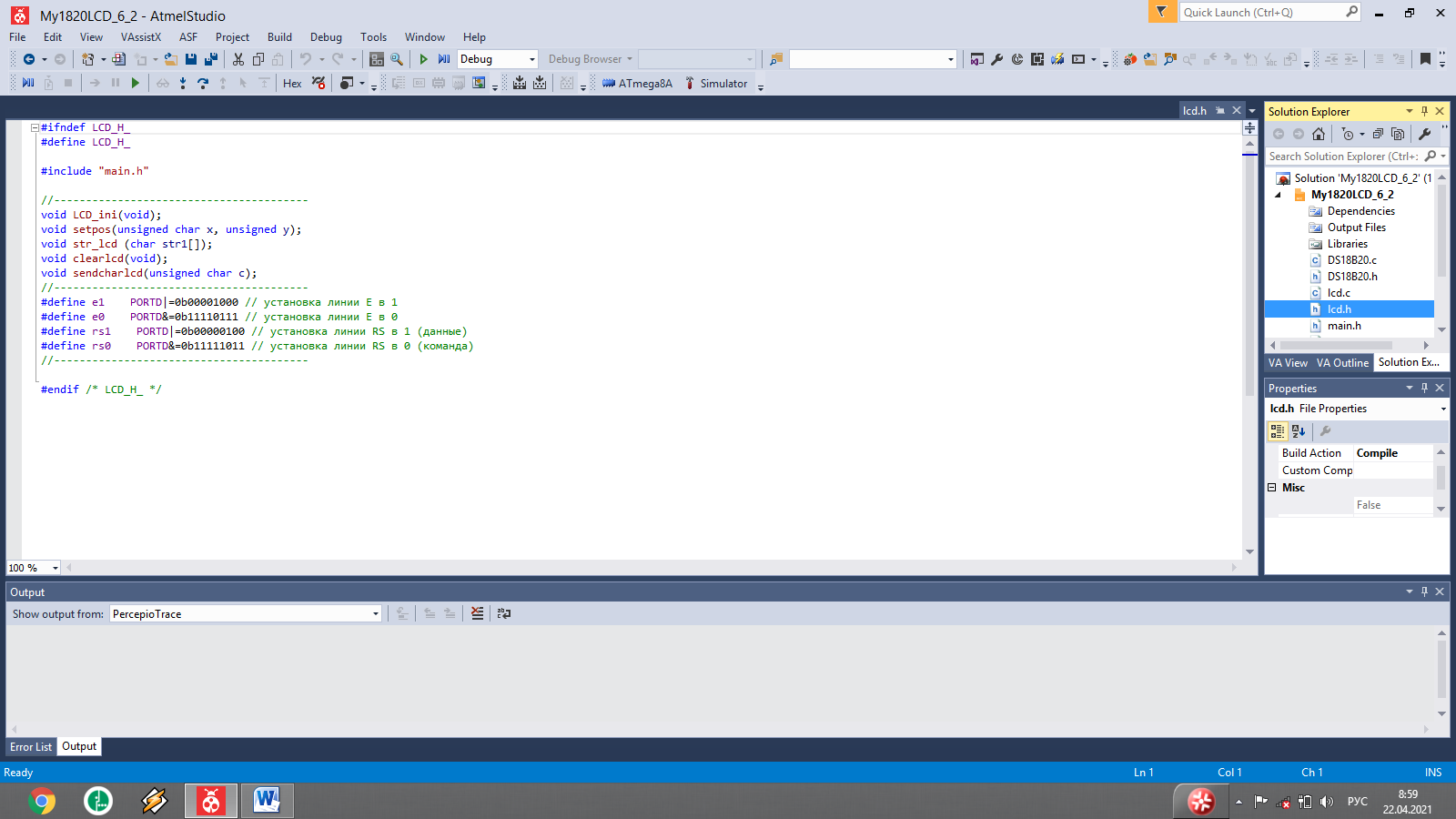
wchar\_t n;

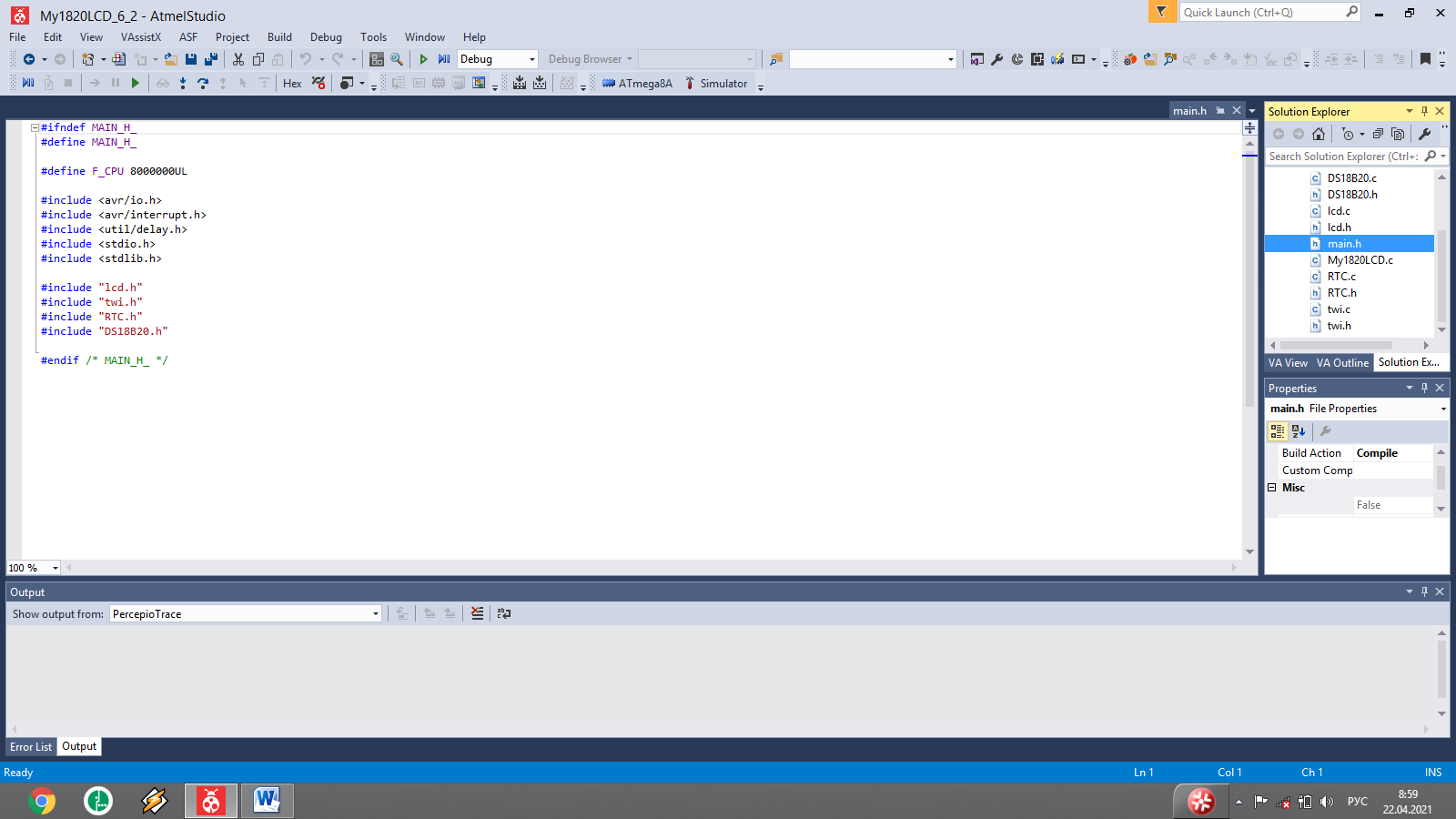
for(n=0;str1[n]!='\0';n++)

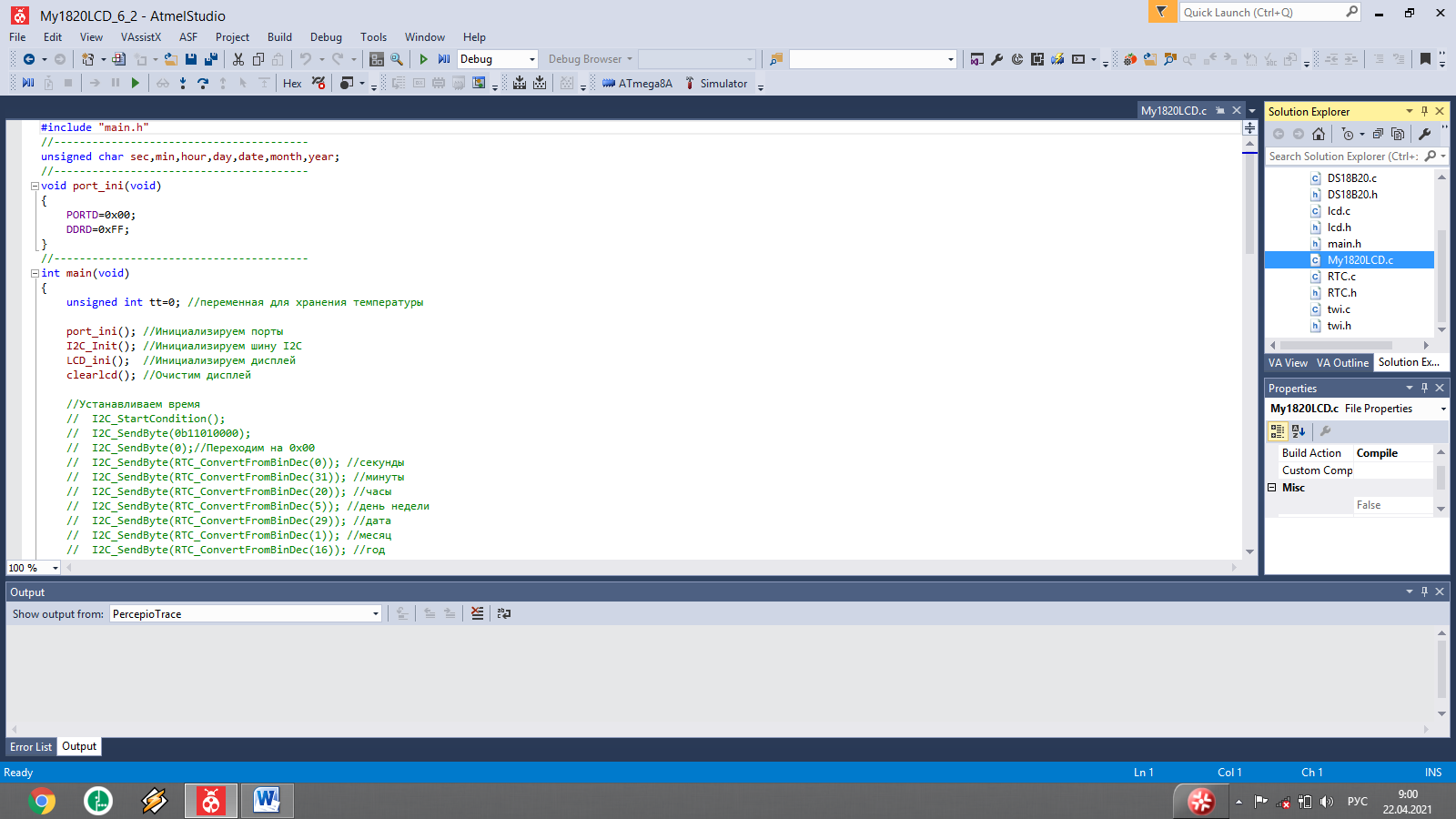
sendcharlcd(str1[n]);

}

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







#include "main.h"

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

unsigned char sec,min,hour,day,date,month,year;

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

void port\_ini(void)

{

PORTD=0x00;

DDRD=0xFF;

}

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

int main(void)

{

unsigned int tt=0; //переменная для хранения температуры

port\_ini(); //Инициализируем порты

I2C\_Init(); //Инициализируем шину I2C

LCD\_ini(); //Инициализируем дисплей

clearlcd(); //Очистим дисплей

//Устанавливаем время

// I2C\_StartCondition();

// I2C\_SendByte(0b11010000);

// I2C\_SendByte(0);//Переходим на 0x00

// I2C\_SendByte(RTC\_ConvertFromBinDec(0)); //секунды

// I2C\_SendByte(RTC\_ConvertFromBinDec(31)); //минуты

// I2C\_SendByte(RTC\_ConvertFromBinDec(20)); //часы

// I2C\_SendByte(RTC\_ConvertFromBinDec(5)); //день недели

// I2C\_SendByte(RTC\_ConvertFromBinDec(29)); //дата

// I2C\_SendByte(RTC\_ConvertFromBinDec(1)); //месяц

// I2C\_SendByte(RTC\_ConvertFromBinDec(16)); //год

// I2C\_StopCondition();

while(1)

{

//Читаем времени

I2C\_SendByteByADDR(0,0b11010000); //переходим на адрес 0

*\_delay\_ms*(200);

I2C\_StartCondition(); //Отправим условие START

I2C\_SendByte(0b11010001); //отправим в устройство бит чтения

sec = I2C\_ReadByte();

min = I2C\_ReadByte();

hour = I2C\_ReadByte();

day = I2C\_ReadByte();

date = I2C\_ReadByte();

month = I2C\_ReadByte();

year = I2C\_ReadLastByte();

I2C\_StopCondition(); //Отправим условие STOP

sec = RTC\_ConvertFromDec(sec); //Преобразуем в десятичный формат

min = RTC\_ConvertFromDec(min); //Преобразуем в десятичный формат

hour = RTC\_ConvertFromDec(hour); //Преобразуем в десятичный формат

day = RTC\_ConvertFromDec(day); //Преобразуем в десятичный формат

year = RTC\_ConvertFromDec(year); //Преобразуем в десятичный формат

month = RTC\_ConvertFromDec(month); //Преобразуем в десятичный формат

date = RTC\_ConvertFromDec(date); //Преобразуем в десятичный формат

setpos(0,0); //Ставим курсор на начало координат

sendcharlcd(date/10+0x30);//Преобразуем число в код числа

sendcharlcd(date%10+0x30);//Преобразуем число в код числа

sendcharlcd('.');

sendcharlcd(month/10+0x30);//Преобразуем число в код числа

sendcharlcd(month%10+0x30);//Преобразуем число в код числа

sendcharlcd('.');

sendcharlcd('2');

sendcharlcd('0');

sendcharlcd(year/10+0x30);//Преобразуем число в код числа

sendcharlcd(year%10+0x30);//Преобразуем число в код числа

sendcharlcd(' ');

sendcharlcd('-');

sendcharlcd(day+0x30);//Преобразуем число в код числа

sendcharlcd('-');

setpos(0,1); //Ставим курсор на начало координат

sendcharlcd(hour/10+0x30);//Преобразуем число в код числа

sendcharlcd(hour%10+0x30);//Преобразуем число в код числа

sendcharlcd(':');

sendcharlcd(min/10+0x30);//Преобразуем число в код числа

sendcharlcd(min%10+0x30);//Преобразуем число в код числа

sendcharlcd(':');

sendcharlcd(sec/10+0x30);//Преобразуем число в код числа

sendcharlcd(sec%10+0x30);//Преобразуем число в код числа

sendcharlcd(' ');

tt = converttemp(dt\_check()); //измеряем температуру

sendcharlcd(tt/10+0x30);//Преобразуем число в код числа

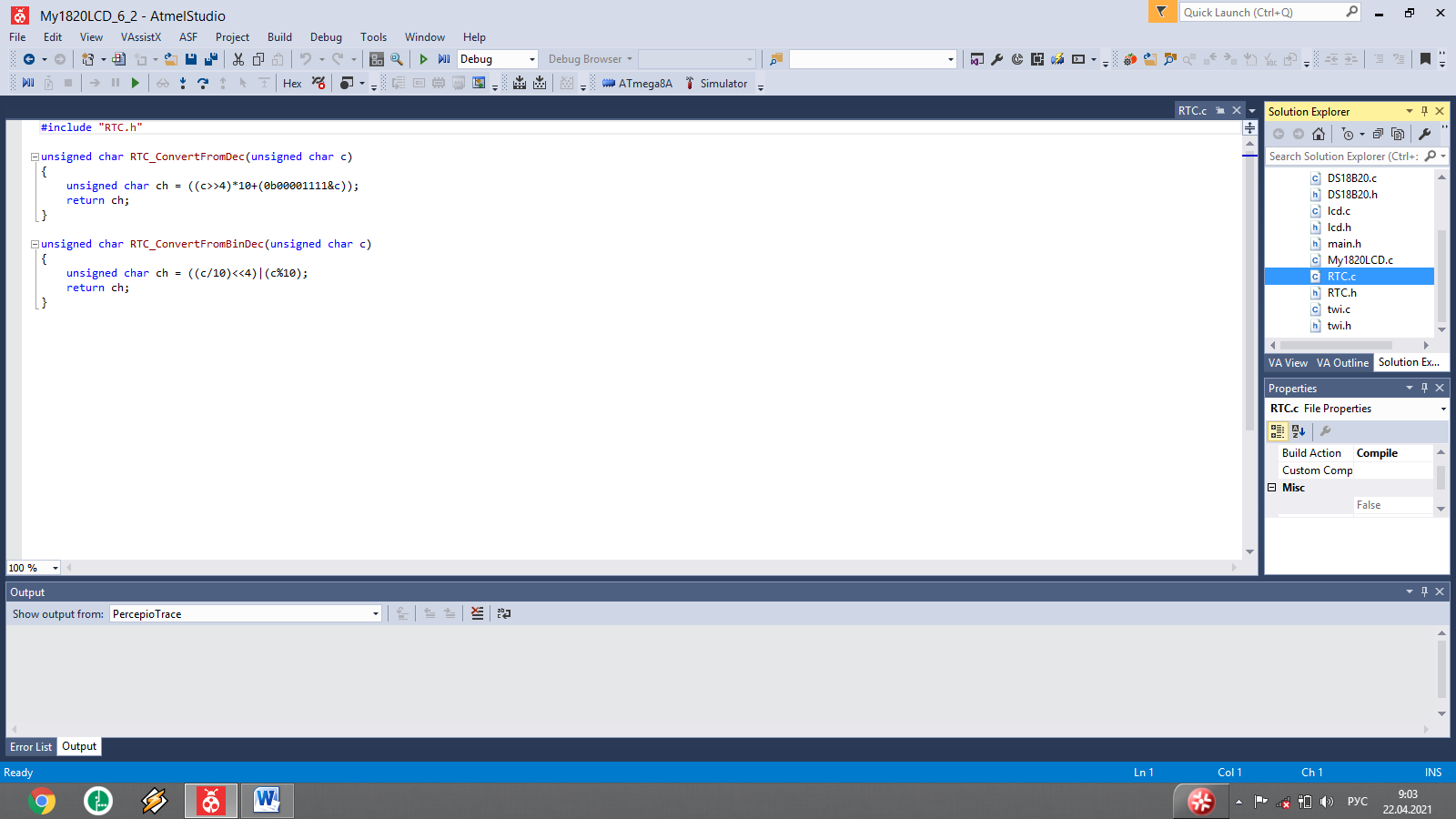
sendcharlcd(tt%10+0x30);//Преобразуем число в код числа

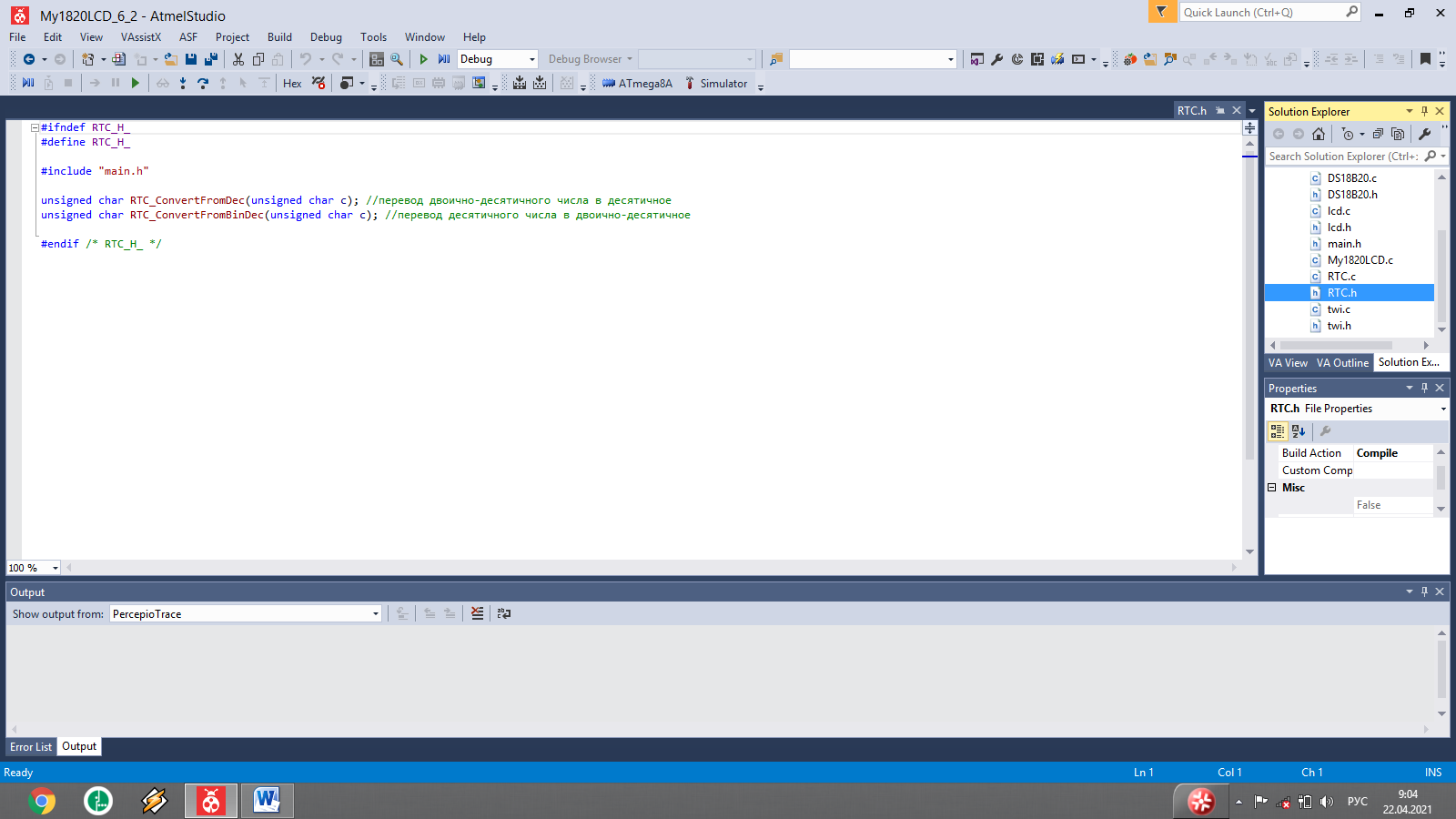
sendcharlcd('\*');

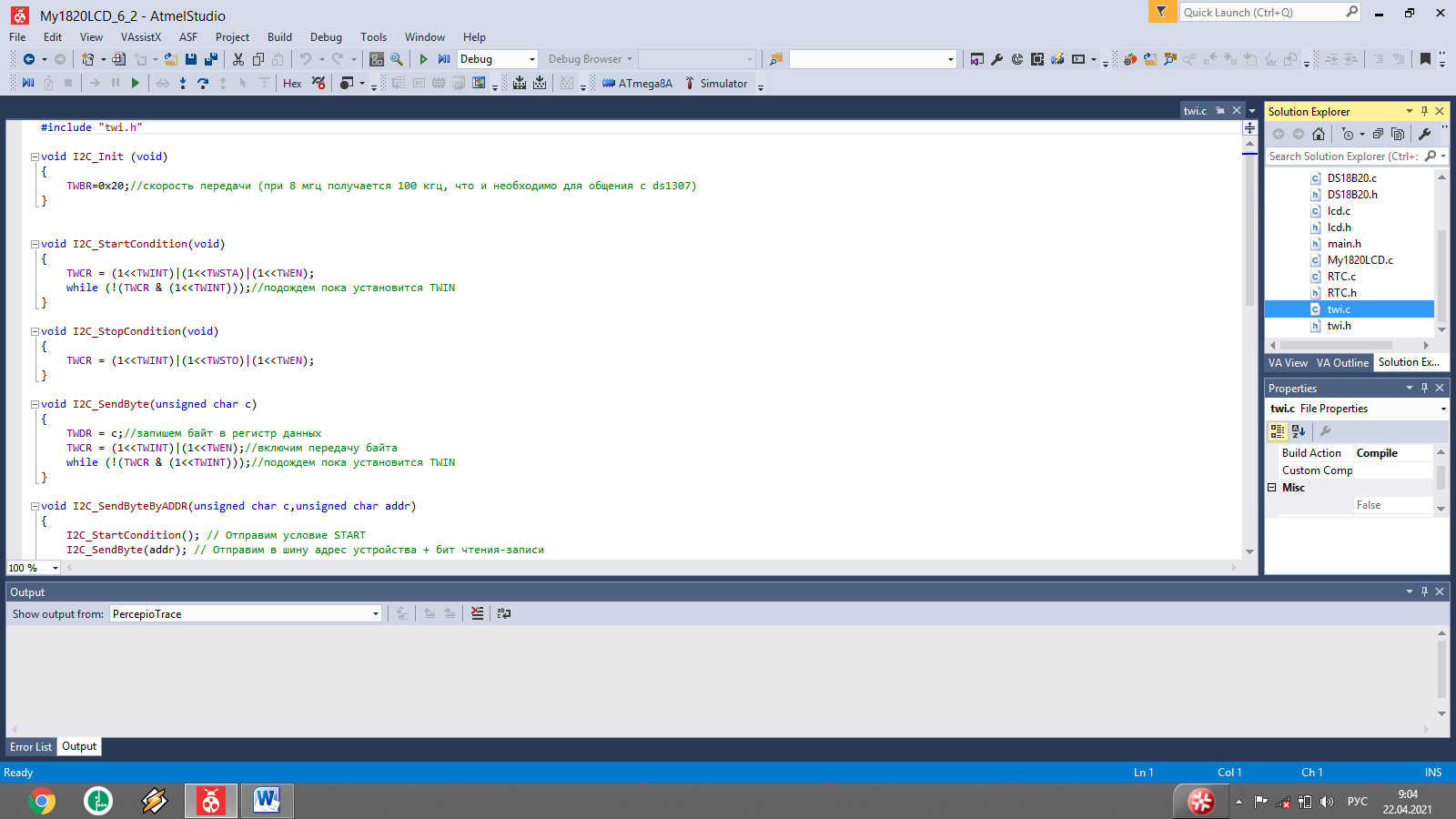
sendcharlcd('C');

}

}







#include "twi.h"

void I2C\_Init (void)

{

TWBR=0x20;//скорость передачи (при 8 мгц получается 100 кгц, что и необходимо для общения с ds1307)

}

void I2C\_StartCondition(void)

{

TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);

while (!(TWCR & (1<<TWINT)));//подождем пока установится TWIN

}

void I2C\_StopCondition(void)

{

TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);

}

void I2C\_SendByte(unsigned char c)

{

TWDR = c;//запишем байт в регистр данных

TWCR = (1<<TWINT)|(1<<TWEN);//включим передачу байта

while (!(TWCR & (1<<TWINT)));//подождем пока установится TWIN

}

void I2C\_SendByteByADDR(unsigned char c,unsigned char addr)

{

I2C\_StartCondition(); // Отправим условие START

I2C\_SendByte(addr); // Отправим в шину адрес устройства + бит чтения-записи

I2C\_SendByte(c);// Отправим байт данных

I2C\_StopCondition();// Отправим условие STOP

}

unsigned char I2C\_ReadByte(void)

{

TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);

while (!(TWCR & (1<<TWINT)));//ожидание установки бита TWIN

return TWDR;//читаем регистр данных

}

unsigned char I2C\_ReadLastByte(void)

{

TWCR = (1<<TWINT)|(1<<TWEN);

while (!(TWCR & (1<<TWINT)));//ожидание установки бита TWIN

return TWDR;//читаем регистр данных

}

