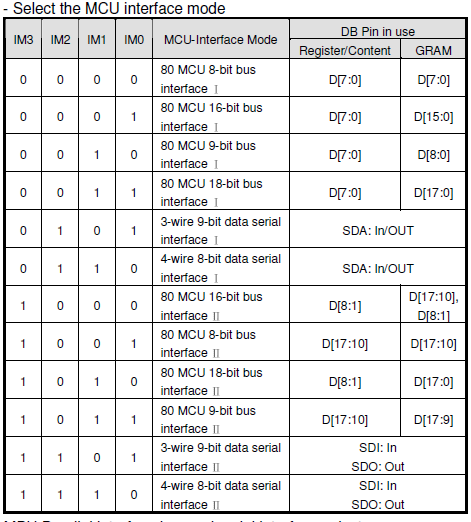
**Дисплей TFT 240×320 8bit**

Сегодня мы попробуем подключить к нашему контроллеру дисплей уже графический цветной разрешением 320 на 240 точек, управляемый по восьмибитному параллельному интерфейсу, Дисплей выполнен с помощью модуля и управляется контроллером **ILI9341**.

Вообще, у данного контроллера дисплея существует несколько режимов подключения к управляющему микроконтроллеру, в том числе есть и SPI, но мне попался дисплей именно с прараллельным способом подключения, о чём я нисколько не жалею. С таким интерфейсом также немало дисплеев, но что обидно, очень мало информации по реализации кода для дисплеев именно с таким интерфейсом. Поэтому наша задача – данный информационный пробел устранить на корню. Вот перечень режимов подключения

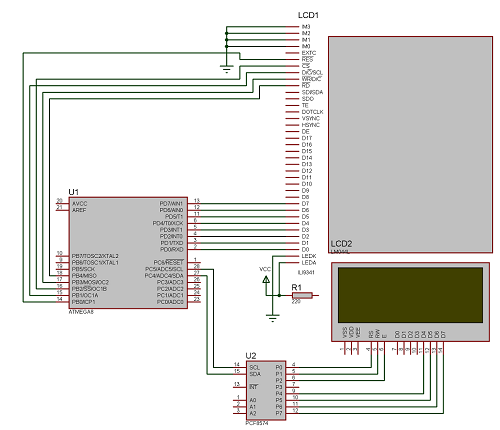


Включить тот или иной режим удаётся благодаря упралению уровнем на ножках IM3:IM0.

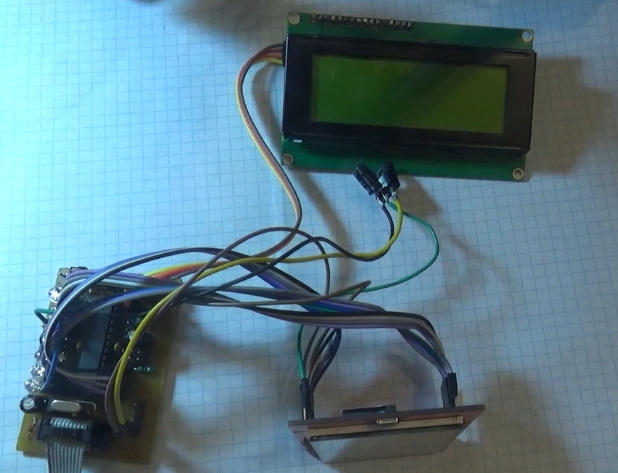
Наш режим, которым мы будем пользоваться – самый первый. Чтобы его включить, нужно на всех данных ножках инициировать низкоуровневой сигнал.

Судя по технической документации, контроллер ILI9341 может управлять 262000 разновидностями цветов.

Вот схема подключения дисплея к контроллеру

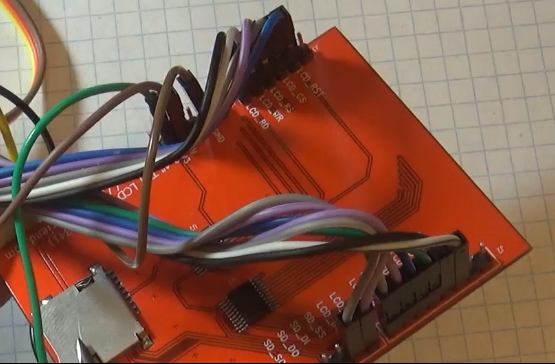
[](https://narodstream.ru/wp-content/uploads/2017/01/image01-3.png)

А вот так схема выглядит на практике



Мы видим также простенький стабилизатор, висящий на проводках, состоящий из микросхемы и конденсатора. Данный стабилизатор служит для подачи питания 3,3 вольта на модуль дисплея, так как такое напряжение для него также требуется. Некоторую информацию, возвращённую из контроллера дисплея мы будем отслеживать с помощью символьного дисплея 20х4, подключенного по интерфейсу **i2c**, который также мы видим на картинке.

А вот так выглядит модуль дисплея снизу



Мы видим ещё на борту данного модуля картоприёмник для карты Micro SD, которая подключается по интерфейсу SPI.

Также мы видим, что все контакты у дисплея подписаны, поэтому с подключением, я думаю сложностей не возникнет.

Первые снизу справа четыре ножки служат для того, чтобы общаться с картой памяти, поэтому они у нас свободные. Используем мы следующие ножки дисплея:

**LCD\_RST** – ножка для перезагрузки дисплея;

**LCD\_CS** – Chip Select (выбор), активный уровень низкий;

**LCD\_RS** – данные/команда, для передачи данных высокий уровень, для передачи команды – низкий;

**LCD\_WR** – включение режима записи, активный уровень низкий;

**LCD\_RD** – включение режима чтения, активный уровень низкий;

**GND** – общий провод;

**5V** – питание 5 вольт;

**3V3** – питание 3,3 вольта;

**LCD\_D0:LCD\_D7** – параллельная 8-разрядная шина данных.

Управление контроллером дисплея происходит посредством команд или другими словами отправкой в определённый регистр определённых величин. В технической документации очень подробно расписаны все регистры, причём есть и краткий перечень команд регистров, а есть и подробный для каждой команды. Техническая документация на контроллер ILI9341 прикреплена внизу данной страницы. Сначала мы отправляем адрес регистра, а затем шлём параметры команды. Причём количество параметров в различных командах также различается, что накладывает некоторые трудности для создания универсальной функции отправки команды. Но ничего, что-нибудь придумаем.

У нас создан проект **TFT9341**. Проект вполне стандартный, подключены к нему наши библиотек, написанные на прошлых занятиях – это **twi** и **lcdtwi**.

Также мы создадим и подключим к проекту ещё 2 файла для библиотеки уже непосредственно нашего нового дисплея – это файлы **ili9341.h** и **ili9341.c**.

Вот так выглядит главный заголовочный файл **main.h**

**#ifndef MAIN\_H\_**

**#define MAIN\_H\_**

**#define F\_CPU 16000000UL**

**#include <avr/io.h>**

**#include <avr/interrupt.h>**

**#include <util/delay.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <avr/pgmspace.h>**

**#include "twi.h"**

**#include "lcdtwi.h"**

**#include "ili9341.h"**

**#endif /\* MAIN\_H\_ \*/**

В файле ili9341.h подключим

#ifndef ILI9341\_H\_

#define ILI9341\_H\_

#include <stdlib.h>

#include <string.h>

**#include "main.h"**

**#include "twi.h"**

**#include "lcdtwi.h"**

Ну и, как вы все знаете, чтобы любой дисплей подключить и его заставить что-то показывать, необходимо пройти определённую процедуру инициализации. По большому счёту это относится не только к дисплеям.

В файле ili9341.c подключим заголовочный файл и напишем каркас для функции инициализации дисплея

**#include "ili9341.h"**

**//————————————————————–**

**void TFT9341\_ini(void)**

**{**

**}**

Не забываем на данную функцию добавить прототип в хедер-файле, а также напишем некоторые макроподстановки, которые нам потребуются в процессе работы с дисплеем

#include "lcdtwi.h"

**#define swap(a,b) {*int16\_t* t=a;a=b;b=t;}**

**#define DATA\_DDR DDRD**

**#define DATA\_PORT PORTD**

**#define DATA\_PIN PIND**

**#define COMMAND\_DDR DDRB**

**#define COMMAND\_PORT PORTB**

**#define LCD\_CS 2//Chip Select**

**#define LCD\_CD 1//Command/Data**

**#define LCD\_WR 3//LCD Write**

**#define LCD\_RD 4//LCD Read**

**#define LCD\_RESET 0//LCD Reset**

**#define RESET\_IDLE COMMAND\_PORT|=(1<<LCD\_RESET)**

**#define CS\_IDLE COMMAND\_PORT|=(1<<LCD\_CS)**

**#define WR\_IDLE COMMAND\_PORT|=(1<<LCD\_WR)**

**#define RD\_IDLE COMMAND\_PORT|=(1<<LCD\_RD)**

**#define RESET\_ACTIVE COMMAND\_PORT&=~(1<<LCD\_RESET)**

**#define CS\_ACTIVE COMMAND\_PORT&=~(1<<LCD\_CS)**

**#define WR\_ACTIVE COMMAND\_PORT&=~(1<<LCD\_WR)**

**#define RD\_ACTIVE COMMAND\_PORT&=~(1<<LCD\_RD)**

**#define CD\_COMMAND COMMAND\_PORT&=~(1<<LCD\_CD)**

**#define CD\_DATA COMMAND\_PORT|=(1<<LCD\_CD)**

**#define BLACK 0x0000**

**#define BLUE 0x001F**

**#define RED 0x0F800**

**#define GREEN 0x07E0**

**#define CYAN 0x07FF**

**#define MAGENTA 0xF81F**

**#define YELLOW 0xFFE0**

**#define WHITE 0xFFFF**

**#define setReadDir() DATA\_DDR=0x00**

**#define setWriteDir() DATA\_DDR=0xFF**

**#define WR\_STROBE {WR\_ACTIVE;WR\_IDLE;}**

**void TFT9341\_ini(void);**

#endif /\* ILI9341\_H\_ \*/

Я думаю, назначение каждого макроса объяснять нет смысла, так как всё ясно из наименований. Но если вдруг кому-то что-то непонятно, то посмотрите видеоурок, кликнув по картинке внизу страницы, там объяснение более подробное.

Но чтобы продолжить писать функцию инициализации, нам необходима будет ещё функция, так как чтобы управлять контроллером дисплея, нам нужно сначала инициализировать как-то уровни на его ножках. Напишем её выше.

**//————————————————————–**

**void port\_ini(void)**

**{**

**DATA\_PORT=0x00;**

**DATA\_DDR=0xFF;//Шина данных на выход**

**COMMAND\_DDR=0x1F;//Командные лапки также все на выход**

**}**

**//————————————————————–**

Вызовем данную функцию в функции инициализации

void TFT9341\_ini(void)

{

**port\_ini();**

Перейдём в главный модуль и немного напишем в главную функцию

int main(void)

{

**I2C\_Init();//Инициализируем TWI**

**LCD\_ini();//инициализируем символьный дисплей**

**clearlcd();//очистим символьный дисплей**

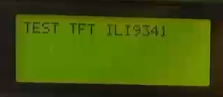
**setpos(0,0);**

**str\_lcd("TEST TFT ILI9341");**

**TFT9341\_ini();**

Мы включили шину I2C, инициализировали и очистили символьный дисплей, спозиционировали курсор и вывели первое сообщение на экран данного дисплея, а затем вызвали и функцию инициализации нашего графического дисплея.

Пока мы, конечно, сделали только заготовку нашего проекта, но, тем не менее, давайте соберём проект и прошьём контроллер для того, чтобы пока проверить работоспособность символьного дисплея



Вернёмся в модуль ili9341.c и напишем ещё две функцию для отправки команд и данных в контроллер дисплея

**//————————————————————–**

**void TFT9341\_SendCommand(unsigned char cmd)**

**{**

**CD\_COMMAND;//лапка в состоянии посылки команды**

**RD\_IDLE;//отключим чтение**

**CS\_ACTIVE;//выбор дисплея**

**DATA\_PORT=cmd;**

**WR\_STROBE;**

**CS\_IDLE;**

**}**

**//————————————————————–**

**void TFT9341\_SendData(unsigned char dt)**

**{**

**CD\_DATA;//лапка в состоянии посылки данных**

**RD\_IDLE;//отключим чтение**

**CS\_ACTIVE;//выбор дисплея**

**DATA\_PORT=dt;**

**WR\_STROBE;**

**CS\_IDLE;**

**}**

**//————————————————————–**

В данных функциях также всё ясно по комментариям. Но немного все равно попытаюсь что-то объяснить.

Сначала мы ножку RS выставляем в уровень, необходимый для нашей задачи (команда или данные), затем отключим чтение, а то вдруго оно включено, инициируя на соответствующей ножке высокий уровень, далее опустим Chip Select, затем на параллельной шине данных выставим уровни, соответствующие значению посылаемого байта, сделаем соответствующий импульс на ножке WR, чтобы данные записались, и поднимем ножку Chip Select.

Далее добавим функцию для полной перезагрузки дисплея

**//————————————————————–**

**void TFT9341\_reset(void)**

**{**

**CS\_IDLE;**

**WR\_IDLE;**

**RD\_IDLE;**

**RESET\_ACTIVE;**

***\_delay\_ms*(2);**

**RESET\_IDLE;**

**CS\_ACTIVE;**

**TFT9341\_SendCommand(0x01); //Software Reset**

**for (*uint8\_t* i=0;i<3;i++) WR\_STROBE;**

**CS\_IDLE;**

**}**

**//————————————————————–**

Начало функции похоже на начало предыдущих функций, но не совсем.

Мы поднимаем ножку выбора, поднимаем ножки записи и чтения, затем делаем аппаратную перезагрузку, формируя импульс на соответствующей ножке, сначала мы её опускаем, затем ждём 2 милисекунда, затем поднимаем, далее опускаем ножку выбора и отправляем команду для программной перезагрузки дисплея. У данной команды параметров нет, поэтому мы их и не отправляем. Затем мы производим три импульса ножкой записи и поднимаем ножку выбора.

Вызовем нашу функцию в функции **TFT9341\_ini**, применив после неё задержку, чтобы контроллер дисплея успел до конца проинициализироваться

port\_ini();

**TFT9341\_reset();**

***\_delay\_ms*(1000);**

Напишем теперь функцию элементарной передачи байта в дисплей

**//————————————————————–**

**void TFT9341\_Write8(unsigned char dt)**

**{**

**DATA\_PORT=dt;**

**WR\_STROBE;**

**}**

**//————————————————————–**

Ну здесь всё понятно, я думаю.

Теперь создадим функцию чтения из регистра контроллера дисплея. Данная функция, ясное дело, будет возвращать результат в виде 32-битной величины. Для начала добавим в эту функцию две переменные

**//————————————————————–**

**unsigned long TFT9341\_ReadReg(unsigned char r)**

**{**

**unsigned long id;**

**unsigned char x;**

**}**

**//————————————————————–**

Далее мы в данной функции опустим кнопку выбора, также инициируем уровень для команды и передадим байт адреса регистра в контроллер дисплея

unsigned char x;

**CS\_ACTIVE;//выбор дисплея**

**CD\_COMMAND;//лапка в состоянии посылки команды**

**TFT9341\_Write8(r);**

Затем воспользуемся макросом setReadDir, который установит все ножки данных контроллера дисплея на выход

TFT9341\_Write8(r);

**setReadDir();**

Потом установим высокий уровень на ножке RS, соответствующий работе с данными и подождём немного

setReadDir();

**CD\_DATA;**

***\_delay\_us*(50);**

Затем переведём в активное состояние ножку чтения и подождём ещё немного

*\_delay\_us*(50);

**RD\_ACTIVE;**

***\_delay\_us*(5);**

После всего этого на порте данных контроллера дисплея у нас должен образоваться байт данных. Заберём его в переменную

*\_delay\_us*(5);

**x=DATA\_PIN;**

Это будет самый старший байт. Поднимем ножку чтения и перепишем данный в нашу 32-разрядную переменную и сдвинем её на 8 пунктов, тем самым наш считанный байт станет немного старше, сместившись вправо. И в последствии мы также и поступим со всеми остальными байтами

x=DATA\_PIN;

**RD\_IDLE;**

**id=x;**

**id<<=8;**

Напишем то же самое для остальных трёх байтов

id<<=8;

**RD\_ACTIVE;**

***\_delay\_us*(5);**

**x=DATA\_PIN;**

**RD\_IDLE;**

**id|=x;**

**id<<=8;**

**RD\_ACTIVE;**

***\_delay\_us*(5);**

**x=DATA\_PIN;**

**RD\_IDLE;**

**id|=x;**

**id<<=8;**

**RD\_ACTIVE;**

***\_delay\_us*(5);**

**x=DATA\_PIN;**

**RD\_IDLE;**

**id|=x;**

Особый случай будет для регистра EFh. Здесь мы сдвигаем всё ещё раз и принимаем ещё один байт

id|=x;

**if(r==0xEF)**

**{**

**id<<=8;**

**RD\_ACTIVE;**

***\_delay\_us*(5);**

**x=DATA\_PIN;**

**RD\_IDLE;**

**id|=x;**

**}**

Далее мы поднимаем ножку выбора и воспользуемся макросом **setWriteDir**, который установит включит все ножки данных на вход

  id|=x;

}

**CS\_IDLE;**

**setWriteDir();**

Немножко подождём для стабилизации и вернём считанные данные

  setWriteDir();

***\_delay\_us*(150);//stabilization time**

**return id;**

}

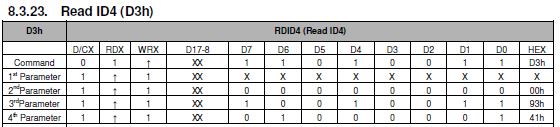
Для того, чтобы узнать, с тем ли дисплеем мы и правильно ли работаем, мы попробуем считать его идентификатор. Для этого в начале файла добавим переменную

#include "ili9341.h"

//————————————————————–

**unsigned long dtt=0;**

Для того, чтобы считать идентификатор, мы воспользуемся данными регистра **D3h**



Считаем данный регистр в функции инициализации дисплея

*\_delay\_ms*(1000);

**dtt=TFT9341\_ReadReg(0xD3);**

Опустим ножку выбора и, применив функцию sprintf, отобразим считанную из регистра величину на символьном дисплее на второй строке

dtt=TFT9341\_ReadReg(0xD3);

**CS\_ACTIVE;**

**setpos(0,1);**

***sprintf*(str,"0x%08lX",dtt);**

**str\_lcd(str);**

Соберём код, прошьём контроллер.

А вот и результат



Идентификатор считан, значит мы работаем именно с тем контроллером и работаем с параллельной шиной правильно, так как регистры читаются.

Что ж, продолжим. Откроем файл ili9341.c и добавим в него две глобальные переменные для хранения ширины и высоты нашего дисплея в пикселях

#include "ili9341.h"

//————————————————————–

**unsigned int X\_SIZE = 0;**

**unsigned int Y\_SIZE = 0;**

unsigned long dtt=0;

Пока они у нас будут равны нулю, а значения мы им присвоим в процессе инициализации.

Создадим функцию изменения ориентации дисплея

**//————————————————————–**

**void TFT9341\_SetRotation(unsigned char r)**

**{**

**}**

**//————————————————————–**

У нас будет четыре вида ориентации. Два вертикальных – один обычный, один перевёрнутый и подобные два горизонтальных. Условимся обозначать из следующим образом:

**0** – вертикальная обычная ориентация

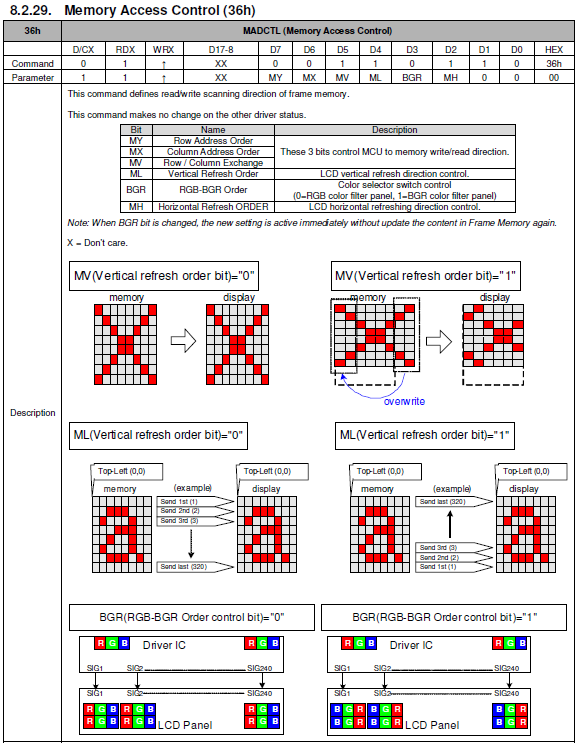
**1** – горизонтальная обычная ориентация

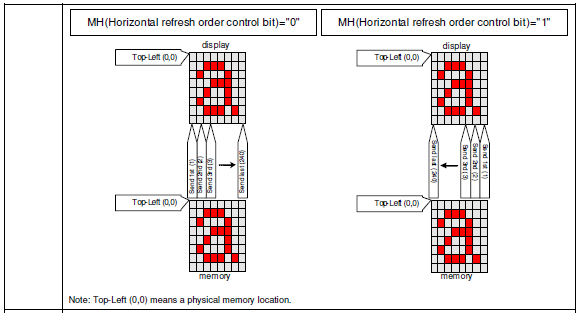
**2** – вертикальная перевёрнутная ориентация

**3** – горизонтальная перевёрнутная ориентация.

Теперь начнём писать тело нашей функции.

Посмотрим в даташите регистр **0x36**





Как мы видим в нотации к данному регистру, что мы можем менять ориетнацию в зависимости от состояния определённых битов в параметре.

Поэтому сначала отправим адрес регистра в контроллер дисплея, а затем создадим ветвление в зависимости от вызываемого режима ориентации

void TFT9341\_SetRotation(unsigned char r)

{

**TFT9341\_SendCommand(0x36);**

**switch(r)**

**{**

**case 0:**

**break;**

**case 1:**

**break;**

**case 2:**

**break;**

**case 3:**

**break;**

**}**

}

Теперь отправим параметр. Содержимое его будет зависеть от определённых битов ориентации. Биты MH и ML всегда будут в нуле, BGR всегда в единице, а вот значение битов MV, MX и MY будут меняться.

При обычной вертикальной ориентации (входной параметр ноль) в единице из этих трёх битов будет только MX (Column Address Order).

Поэтому напишем первый кейс следующим образом, заодно и присвоим нашим глобальным переменным размеров экрана соответствующие значения

case 0:

**TFT9341\_SendData(0x48);**

**X\_SIZE = 240;**

**Y\_SIZE = 320;**

break;

При горизонтальной ориентации (входной параметр 1) в единице будет бит MV (Row / Column Exchange).

Напишем следующий кейс

case 1:

**TFT9341\_SendData(0x28);**

**X\_SIZE = 320;**

**Y\_SIZE = 240;**

**break;**

При вертикальной перевёрнутой ориентации (входной параметр 2) в единице будет уже бит MY (Row Address Order).

Напишем кейс

case 2:

**TFT9341\_SendData(0x88);**

**X\_SIZE = 240;**

**Y\_SIZE = 320;**

break;

При горизонтальной перевёрнутой ориентации (входной параметр 3) в единице будут все три бита

case 3:

**TFT9341\_SendData(0xE8);**

**X\_SIZE = 320;**

**Y\_SIZE = 240;**

break;

Продолжим нашу инициализацию. Допишем в функцию инициализации следующий код, который лишний раз перезагрузит программно наш дисплей, а вернее его контроллер

str\_lcd(str);

**TFT9341\_SendCommand(0x01);//Software Reset**

Следующая команда уже будет требовать данные (целых 5 параметров)

TFT9341\_SendCommand(0x01);//Software Reset

**TFT9341\_SendCommand(0xCB);//Power Control A**

**TFT9341\_SendData(0x39);**

**TFT9341\_SendData(0x2C);**

**TFT9341\_SendData(0x00);**

**TFT9341\_SendData(0x34);**

**TFT9341\_SendData(0x02);**

Ну, я не буду сюда постить весь даташит, дабы не засорять страницу и базу данных, буду объяснять вкратце.

Данный регистр, в которые мы отправили столько данных – это регистр настроек или управления. Все биты, которые мы там включили, вы можете посмотреть в даташите.

Есть ещё один подобный регистр, в который мы отправим три аргумента

TFT9341\_SendData(0x02);

**TFT9341\_SendCommand(0xCF);//Power Control B**

**TFT9341\_SendData(0x00);**

**TFT9341\_SendData(0xC1);**

**TFT9341\_SendData(0x30);**

Следующие 2 команды вносят настройки в регистр управления различными таймингами

TFT9341\_SendData(0x30);

**TFT9341\_SendCommand(0xE8);//Driver timing control A**

**TFT9341\_SendData(0x85);**

**TFT9341\_SendData(0x00);**

**TFT9341\_SendData(0x78);**

**TFT9341\_SendCommand(0xEA);//Driver timing control B**

**TFT9341\_SendData(0x00);**

**TFT9341\_SendData(0x00);**

Также заполним ещё ряд регистров, назначения которых я объяснил в комментариях к каждой команде. Также где-то посередине кода мы вызовем функцию ориентации дисплея и инициализируем там обычный вертикальный режим с индексом 0. На этом этапе у нас и проинициализируются наши глобальные переменные размера нашего экрана

TFT9341\_SendData(0x00);

**TFT9341\_SendCommand(0xED);//Power on Sequence control**

**TFT9341\_SendData(0x64);**

**TFT9341\_SendData(0x03);**

**TFT9341\_SendData(0x12);**

**TFT9341\_SendData(0x81);**

**TFT9341\_SendCommand(0xF7);//Pump ratio control**

**TFT9341\_SendData(0x20);**

**TFT9341\_SendCommand(0xC0);//Power Control 1**

**TFT9341\_SendData(0x10);**

**TFT9341\_SendCommand(0xC1);//Power Control 2**

**TFT9341\_SendData(0x10);**

**TFT9341\_SendCommand(0xC5);//VCOM Control 1**

**TFT9341\_SendData(0x3E);**

**TFT9341\_SendData(0x28);**

**TFT9341\_SendCommand(0xC7);//VCOM Control 2**

**TFT9341\_SendData(0x86);**

**TFT9341\_SetRotation(0);**

**TFT9341\_SendCommand(0x3A);//Pixel Format Set**

**TFT9341\_SendData(0x55);//16bit**

**TFT9341\_SendCommand(0xB1);**

**TFT9341\_SendData(0x00);**

**TFT9341\_SendData(0x18);// Частота кадров 79 Гц**

**TFT9341\_SendCommand(0xB6);//Display Function Control**

**TFT9341\_SendData(0x08);**

**TFT9341\_SendData(0x82);**

**TFT9341\_SendData(0x27);//320 строк**

**TFT9341\_SendCommand(0xF2);//Enable 3G (пока не знаю что это за режим)**

**TFT9341\_SendData(0x00);//не включаем**

**TFT9341\_SendCommand(0x26);//Gamma set**

**TFT9341\_SendData(0x01);//Gamma Curve (G2.2) (Кривая цветовой гаммы)**

**TFT9341\_SendCommand(0xE0);//Positive Gamma Correction**

**TFT9341\_SendData(0x0F);**

**TFT9341\_SendData(0x31);**

**TFT9341\_SendData(0x2B);**

**TFT9341\_SendData(0x0C);**

**TFT9341\_SendData(0x0E);**

**TFT9341\_SendData(0x08);**

**TFT9341\_SendData(0x4E);**

**TFT9341\_SendData(0xF1);**

**TFT9341\_SendData(0x37);**

**TFT9341\_SendData(0x07);**

**TFT9341\_SendData(0x10);**

**TFT9341\_SendData(0x03);**

**TFT9341\_SendData(0x0E);**

**TFT9341\_SendData(0x09);**

**TFT9341\_SendData(0x00);**

**TFT9341\_SendCommand(0xE1);//Negative Gamma Correction**

**TFT9341\_SendData(0x00);**

**TFT9341\_SendData(0x0E);**

**TFT9341\_SendData(0x14);**

**TFT9341\_SendData(0x03);**

**TFT9341\_SendData(0x11);**

**TFT9341\_SendData(0x07);**

**TFT9341\_SendData(0x31);**

**TFT9341\_SendData(0xC1);**

**TFT9341\_SendData(0x48);**

**TFT9341\_SendData(0x08);**

**TFT9341\_SendData(0x0F);**

**TFT9341\_SendData(0x0C);**

**TFT9341\_SendData(0x31);**

**TFT9341\_SendData(0x36);**

**TFT9341\_SendData(0x0F);**

Выйдем из спящего режима и включим дисплей

  TFT9341\_SendData(0x0F);

**TFT9341\_SendCommand(0x11);//Выйдем из спящего режим**

***\_delay\_ms*(150);**

**TFT9341\_SendCommand(0x29);//Включение дисплея**

**TFT9341\_SendData(0x2C);**

***\_delay\_ms*(150);**

}

На этом инициализация закончена.

 Напишем теперь функцию заливки определённой области памяти данными пикселей одного цвета для последующего использования в функции заливки одним цветом прямоугольной области

**//————————————————————–**

**void TFT9341\_Flood(unsigned short color, unsigned long len)**

**{**

**}**

**//————————————————————–**

Первый входной параметр – это цвет пикселей, а второй длина области памяти для заливки. Адресация будет в отдельной функции.

Добавим в функцию несколько переменных, причём две из них сразу проинициализируем определённым образом

void TFT9341\_Flood(unsigned short color, unsigned long len)

{

**unsigned short blocks;**

**unsigned char i, hi = color>>8, lo=color;**

То есть 16 бит цвета мы здесь распределим по двум переменным.

Опустим ножку выбора и ножку команд/данных

unsigned char i, hi = color>>8, lo=color;

**CS\_ACTIVE;**

**CD\_COMMAND;**

Отправим цвет в соответствующий для этого регистр

CD\_COMMAND;

**TFT9341\_Write8(0x2C);**

**CD\_DATA;**

**TFT9341\_Write8(hi);**

**TFT9341\_Write8(lo);**

То есть здесь мы уже не используем специальные функции для отправки команд и данных для экономии времени, здесь уже время очень важно, иначе пострадает скорость работы с дисплеем. Поэтому мы уже пользуемся функциями более низкоуровневыми без лишнего дрыгания ножками.

Продекрементируем длину области, так как отсчёт у нас происходит не с 1, а с 0

TFT9341\_Write8(lo);

**len–;**

Разобьём нашу длину на блоки по 64 пикселя

len–;

**blocks=(unsigned short)(len/64);//64 pixels/block**

Дальше условие, если младшая часть 16-битного цвета равна старшей

blocks=(unsigned short)(len/64);//64 pixels/block

**if (hi==lo)**

**{**

**}**

**else**

**{**

**}**

В теле условия цикл, равный количеству блоков

if (hi==lo)

{

**while(blocks–)**

**{**

**}**

}

В теле цикла ещё один цикл, равный 16

while(blocks–)

{

**i=16;**

**do**

**{**

**WR\_STROBE;WR\_STROBE;WR\_STROBE;WR\_STROBE;//2bytes/pixel**

**WR\_STROBE;WR\_STROBE;WR\_STROBE;WR\_STROBE;//x4 pixel**

**} while (–i);**

}

В данном цикле мы просто стробируем, отправляя один и тот же байт в контроллер дисплея, уровни байта ведь у на ножках порта установлен. Получается, что мы один и тот же байт отправим 128 раз, то есть если разбить на пары, то 64 пары и, следовательно, 64 пикселя. То есть при условии равенства старшего и младшего байтов цвета мы отправили столько блоков, сколько у нас есть, а засчёт того, что мы каждый раз не устанавливаем уровень на ножках порта данных, мы значительно выигрываем во времени.

Выйдя из цикла отправки всех блоков, мы отправим оставшуюся часть байтов, если у нас длина не будет делиться на 64 без остатка

    } while (–i);

  }

**//Fill any remaining pixels(1 to 64)**

**for (i=(unsigned char)len&63;i–;)**

**{**

**WR\_STROBE;**

**WR\_STROBE;**

**}**

}

Вот так. Теперь заходим в "противную часть" нашего условия, когда у нас старший и младший байт цвета будут разными. Я думаю, это будет чаще. Там также будет цикл, равный количеству полных блоков

else

{

**while(blocks–)**

**{**

**}**

}

В цикле также цикл, равный 16, в котором мы передаём весь блок

while(blocks–)

{

**i=16;**

**do**

**{**

**TFT9341\_Write8(hi);TFT9341\_Write8(lo);TFT9341\_Write8(hi);TFT9341\_Write8(lo);**

**TFT9341\_Write8(hi);TFT9341\_Write8(lo);TFT9341\_Write8(hi);TFT9341\_Write8(lo);**

**} while (–i);**

}

Ну, тут понятно, раз байты не равны, передаём их по очереди.

Затем также передаём оставшуюся часть пикселей

      } while (–i);

    }

**//Fill any remaining pixels(1 to 64)**

**for (i=(unsigned char)len&63;i–;)**

**{**

**TFT9341\_Write8(hi);**

**TFT9341\_Write8(lo);**

**}**

}

Ну и в конце поднимем ножку выбора

      TFT9341\_Write8(lo);

    }

  }

**CS\_IDLE;**

}

Теперь напишем функцию записи в регистр 32-битного числа

**//————————————————————–**

**void TFT9341\_WriteRegister32(unsigned char r, unsigned long d)**

**{**

**CS\_ACTIVE;**

**CD\_COMMAND;**

**TFT9341\_Write8(r);**

**CD\_DATA;**

***\_delay\_us*(1);**

**TFT9341\_Write8(d>>24);**

***\_delay\_us*(1);**

**TFT9341\_Write8(d>>16);**

***\_delay\_us*(1);**

**TFT9341\_Write8(d>>8);**

***\_delay\_us*(1);**

**TFT9341\_Write8(d);**

**CS\_IDLE;**

**}**

**//————————————————————–**

Я думаю, тут даже объяснять ничего не надо. Мы сначала передаём адрес регистра, а затем с некотроыми задержками передаём наши данные.

Но, прежде чем отправлять в память байты, нам нужно объявить область памяти, в которую будет вся наша цепочка одинаковых пикселей отправляться. Для этого мы напишем специальную функцию

**//————————————————————–**

**void TFT9341\_SetAddrWindow(unsigned int x1,unsigned int y1,unsigned int x2,unsigned int y2)**

**{**

**unsigned long t;**

**CS\_ACTIVE;**

**t = x1;**

**t<<=16;**

**t |= x2;**

**TFT9341\_WriteRegister32(0x2A,t);//Column Addres Set**

**t = y1;**

**t<<=16;**

**t |= y2;**

**TFT9341\_WriteRegister32(0x2B,t);//Page Addres Set**

**CS\_IDLE;**

**}**

**//————————————————————–**

Ну здесь мы в специализированные регистры **2Ah** и **2Bh** передаём наши коортдинаты сначала начала и окончания вертикальной области, а затем горизонтальной, распределив соответственно эти координаты в четырехбайтовые величины. Отправка в регистры стандартная.

Также нам потребуется какой-то шрифт, если мы захотим выводить какой-то текст. Я с помощью определённой программы создал этот шрифт, причём написал не весь шрифт, а несколько букв, ибо памяти у нас не так много. Сделал я их высотой в 16 пикселей

**font 16**

**//————————————————————–**

**//font 16**

**const unsigned char chars16[][32] PROGMEM =**

**{**

**//SPACE**

**{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,**

**0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},**

**//0**

**{0x00, 0x00, 0x00, 0x00, 0x07, 0xc0, 0x0f, 0xe0, 0x0c, 0x60, 0x18, 0x30, 0x18, 0x30, 0x18, 0x30,**

**0x18, 0x30, 0x18, 0x30, 0x18, 0x30, 0x0c, 0x60, 0x0f, 0xe0, 0x07, 0xc0, 0x00, 0x00, 0x00, 0x00},**

**//1**

**{0x00, 0x00, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x01, 0x80, 0x01, 0x80, 0x01, 0x80, 0x01, 0x80,**

**0x01, 0x80, 0x01, 0x80, 0x01, 0x80, 0x01, 0x80, 0x03, 0xc0, 0x03, 0xc0, 0x00, 0x00, 0x00, 0x00},**

**//A**

**{0x00, 0x00, 0x00, 0x00, 0x07, 0xf0, 0x07, 0xf0, 0x01, 0x40, 0x03, 0x60, 0x03, 0x60, 0x06, 0x30,**

**0x07, 0xf0, 0x0f, 0xf8, 0x0c, 0x18, 0x0c, 0x18, 0x3e, 0x3e, 0x3e, 0x3e, 0x00, 0x00, 0x00, 0x00},**

**//B**

**{0x00, 0x00, 0x00, 0x00, 0x0f, 0xe0, 0x0f, 0xf0, 0x06, 0x30, 0x06, 0x30, 0x07, 0xe0, 0x07, 0xf0,**

**0x06, 0x38, 0x06, 0x18, 0x06, 0x18, 0x06, 0x38, 0x0f, 0xf0, 0x0f, 0xe0, 0x00, 0x00, 0x00, 0x00},**

**//C**

**{0x00, 0x00, 0x00, 0x00, 0x07, 0xd8, 0x0f, 0xf8, 0x1c, 0x38, 0x38, 0x18, 0x30, 0x00, 0x30, 0x00,**

**0x30, 0x00, 0x30, 0x08, 0x38, 0x0c, 0x1c, 0x38, 0x0f, 0xf0, 0x07, 0xe0, 0x00, 0x00, 0x00, 0x00}**

**};**

**//————————————————————–**

Ну и далее собственно заливка всей области экрана определённым цветом

**//————————————————————–**

**void TFT9341\_FillScreen(unsigned int color)**

**{**

**TFT9341\_SetAddrWindow(0,0,X\_SIZE-1,Y\_SIZE-1);**

**TFT9341\_Flood(color,(long)X\_SIZE\*(long)Y\_SIZE);**

**}**

**//————————————————————–**

Мы использовали наши вышенаписанные функции, чтобы сначала передать координаты, а затем уже отправить цвет и длину всей области, получив её способом перемножения горизонтального и вертикального размеров.

Напишем на данную функцию прототип и вызовем её в main() после инициализации дисплея

TFT9341\_ini();

**TFT9341\_FillScreen(BLACK);**

Соберём код, пошьём контроллер и посмотрим результат



Как мы видим, дисплей наш окрасился в чёрный цвет.

Поменяем цвет на красный и ещё раз проверим

TFT9341\_ini();

**TFT9341\_FillScreen(RED);**



Давайте теперь окрасим дисплей во все те цвета, которые мы прописали в макросах. Зайдём в функцию main() и напишем там код после вызова функции инициализации дисплея

TFT9341\_ini();

**TFT9341\_FillScreen(RED);**

***\_delay\_ms*(500);**

**TFT9341\_FillScreen(BLUE);**

***\_delay\_ms*(500);**

**TFT9341\_FillScreen(GREEN);**

***\_delay\_ms*(500);**

**TFT9341\_FillScreen(CYAN);**

***\_delay\_ms*(500);**

**TFT9341\_FillScreen(MAGENTA);**

***\_delay\_ms*(500);**

**TFT9341\_FillScreen(YELLOW);**

***\_delay\_ms*(500);**

**TFT9341\_FillScreen(WHITE);**

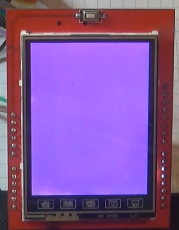
***\_delay\_ms*(500);**

**TFT9341\_FillScreen(BLACK);**

***\_delay\_ms*(500);**

while (1)

Соберём код, прошьём контроллер, посмотрим результат



Продолжаем дальше.

Напишем функцию заливки прямоугольной области экрана, то есть не всю область, а отдельный прямоугольник с заранее заданными координатами. Вернёмся в файл ili9341.c и напишем данную функцию сразу после функции заливки полного экрана

**//————————————————————–**

**void TFT9341\_FillRectangle(unsigned int color,unsigned int x1, unsigned int y1,**

**unsigned int x2, unsigned int y2)**

**{**

**TFT9341\_SetAddrWindow(x1, y1, x2, y2);**

**TFT9341\_Flood(color, (long)(x2–x1+1) \* (long)(y2–y1+1));**

**}**

**//————————————————————–**

Здесь, я думаю, всё предельно понятно. Во входных параметрах мы передаём цвет, координаты левого верхнего угла прямоугольной области и координаты правого нижнего угла. Также мы рассчитываем размер заливаемой области умножив вычисленную длину на высоту.

Напишем также в заголовочный файл прототип данной функции.

Теперь напишем функцию генерирования случайного цвета

**//————————————————————–**

**unsigned int TFT9341\_RandColor(void)**

**{**

**unsigned char c = *rand*()%8;**

**switch(c)**

**{**

**case 0:**

**return BLACK;**

**break;**

**case 1:**

**return BLUE;**

**break;**

**case 2:**

**return RED;**

**break;**

**case 3:**

**return GREEN;**

**break;**

**case 4:**

**return CYAN;**

**break;**

**case 5:**

**return MAGENTA;**

**break;**

**case 6:**

**return YELLOW;**

**break;**

**case 7:**

**return WHITE;**

**break;**

**}**

**return BLACK;**

**}**

**//————————————————————–**

Добавим прототип для данной функции в заголовочный файл и добавим в main() небольшой тест сразу после кода по закрашиванию экрана, который мы только что написали

*\_delay\_ms*(500);

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

**{**

**TFT9341\_FillRectangle(TFT9341\_RandColor(),0,0,119,159);**

***\_delay\_ms*(100);**

**TFT9341\_FillRectangle(TFT9341\_RandColor(),120,0,239,159);**

***\_delay\_ms*(100);**

**TFT9341\_FillRectangle(TFT9341\_RandColor(),0,160,119,319);**

***\_delay\_ms*(100);**

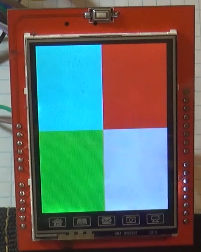
**TFT9341\_FillRectangle(TFT9341\_RandColor(),120,160,239,319);**

***\_delay\_ms*(100);**

**}**

**TFT9341\_FillScreen(BLACK);**

Соберём код, прошьём контроллер и посмотрим результат на дисплее



Опять вернёмся в наш файл с функциями и напишем там функцию вывода точки определённого цвета на экран

**//————————————————————–**

**void TFT9341\_DrawPixel(int x, int y, unsigned int color)**

**{**

**if((x<0)||(y<0)||(x>=X\_SIZE)||(y>=Y\_SIZE)) return;**

**CS\_ACTIVE;**

**TFT9341\_SetAddrWindow(x,y,X\_SIZE-1,Y\_SIZE-1);**

**CS\_ACTIVE;**

**CD\_COMMAND;**

**TFT9341\_Write8(0x2C);**

**CD\_DATA;**

**TFT9341\_Write8(color>>8);TFT9341\_Write8(color);**

**CS\_IDLE;**

**}**

**//————————————————————–**

В данной функции мы сначала проверяем входные параметры на валидность, затем опускаем шину выбора, затем адресуем память области экрана, верхний левый угол будет у нас равен координатам точки, а нижний правый – нижнему правому углу дисплея, затем мы ещё раз опускаем шину выбора, так как в функции адресации она поднимается, затем переходим в режим передачи команд, посылаем адрес регистра для передачи данных в область памяти и передаём только один пиксель, ну и по окончании поднимаем шину выбора.

Напишем для данной функции прототип.

Ну и, соответственно, тест в main()

TFT9341\_FillScreen(BLACK);

**for(i=0;i<15000;i++)**

**{**

**TFT9341\_DrawPixel(*rand*()%240,*rand*()%320,TFT9341\_RandColor());**

**}**

**TFT9341\_FillScreen(BLACK);**

Мы будем выводить здесь 15000 точек случайного цвета в случайное место дисплея.

Соберём код, прошьём контроллер и посмотрим результат нашей работы



Теперь давайте зайдём в файл ili9341.c и напишем вывод на экран дисплея линии с координатами начала и окончания. Так как эта функиция оказалась непростой, то сначала мы её просто создадим, а писать будем постепенно

**//————————————————————–**

**void TFT9341\_DrawLine(unsigned int color,unsigned int x1, unsigned int y1, unsigned int x2, unsigned int y2)**

**{**

**}**

**//—————————————————————**

Как обычно сначала во входных параметрах у нас цвет, а затем координаты начала и окончания линиии.

Применим вот такую конструкцию

void TFT9341\_DrawLine(unsigned int color,unsigned int x1, unsigned int y1, unsigned int x2, unsigned int y2)

{

**int steep = *abs*(y2 – y1) > *abs*(x2 – x1);**

Здесь переменная будет равна единице в случае выполнения условия, а в случае невыполнения будет нулём. В условии у нас неравенство. Мы сравниваем – что у нас больше – разность между вертикальными координатами или между горизонтальными.

И затем уже в зависимости от результата войдём в условие

int steep = *abs*(y2 – y1) > *abs*(x2 – x1);

**if (steep) {**

**swap(x1, y1);**

**swap(x2, y2);**

**}**

Если результат истинный, то мы поменяемся значениями вертикальных и горизонтальных координат.

Теперь, если у нас первая горизонтальная координата больше второй, то поменяемся их значениями, причём поменяемся и вертикальными

  swap(x2, y2);

}

**if (x1 > x2) {**

**swap(x1, x2);**

**swap(y1, y2);**

**}**

Добавим 2 переменных

  swap(y1, y2);

}

**int dx, dy;**

Занесём в них разность координат

int dx, dy;

**dx = x2 – x1;**

**dy = *abs*(y2 – y1);**

В горизонтальных координатах мы не применяем функцию абсолютной величины, так как выше мы уже добились того, что теперь у нас x2 всегда больше или равно x1.

сохраним в переменную половину разницы между вертикальными координатами и создадим ещё одну переменную

dy = *abs*(y2 – y1);

**int err = dx / 2;**

**int ystep;**

Затем, в зависимости от того, какая вертикальная координата меньше, занесём в эту переменную значение

int ystep;

**if (y1 < y2) {**

**ystep = 1;**

**} else {**

**ystep = -1;**

**}**

И в конце функции цикл, в котором мы и будем рисовать нашу прямую линию, постепенно инкрементируя горизонтальную координату

    ystep = -1;

  }

**for (; x1<=x2; x1++) {**

**}**

}

Теперь займёмся телом данного цикла.

А в цикле будет условие, в котором мы нарисуем точку линии в зависимости от значения переменной **steep**

for (; x1<=x2; x1++) {

**if (steep) {**

**TFT9341\_DrawPixel(y1, x1, color);**

**} else {**

**TFT9341\_DrawPixel(x1, y1, color);**

**}**

Далее занесём в переменную разницу вертикальных координат с противоположным знаком

TFT9341\_DrawPixel(x1, y1, color);

}

**err -= dy;**

Ну и в конце цикла условие, в котором мы наращиваем значение y1 или убавляем его в зависимости от значения переменной ystep

    err -= dy;

**if (err < 0) {**

**y1 += ystep;**

**err += dx;**

**}**

  }

}

Вот такая вот интересная функция. Ну, а теперь, конечно, тест в main(), но перед этим мы не забываем про прототип в заголовочном файле. В тесте мы выведем вертикальные линии случайного цвета

TFT9341\_FillScreen(BLACK);

**for(i=0;i<240;i++)**

**{**

**TFT9341\_DrawLine(TFT9341\_RandColor(),i,0,i,319);**

**}**

***\_delay\_ms*(500);**

**TFT9341\_FillScreen(BLACK);**

Скомпилируем наш код, прошьём контроллер и посмотрим на наш дисплей



Далее напишем тест вывода прямых линий со случайными координатами и случайным цветом

TFT9341\_FillScreen(BLACK);

**for(i=0;i<1000;i++)**

**{**

**TFT9341\_DrawLine(TFT9341\_RandColor(),*rand*()%240,*rand*()%320,*rand*()%240,*rand*()%320);**

**}**

***\_delay\_ms*(500);**

**TFT9341\_FillScreen(BLACK);**

Давайте проверим данный тест



Вернёмся в файл с функциями и напишем функцию рисования незакрашенного прямоугольника. Так как прямоугольники у нас будут горизонтальными или вертикальными (не под наклоном), то реализация этй задачи значительно упрощается до вывода четырёх прямых линий

**//—————————————————————**

**void TFT9341\_DrawRect(unsigned int color,unsigned int x1, unsigned int y1, unsigned int x2, unsigned int y2)**

**{**

**TFT9341\_DrawLine(color,x1,y1,x2,y1);**

**TFT9341\_DrawLine(color,x2,y1,x2,y2);**

**TFT9341\_DrawLine(color,x1,y1,x1,y2);**

**TFT9341\_DrawLine(color,x1,y2,x2,y2);**

**}**

**//—————————————————————**

Входные параметры стандартные, как и у закрашенного прямоугольника, а в теле вывод прямых линий: верхней, левой, правой и нижней.

Создадим для функции прототип и напишем интересный тест в main(), в котором выведем наши прямоугольники красивым образом

TFT9341\_FillScreen(BLACK);

**for(i=0;i<120;i++)**

**{**

**TFT9341\_DrawRect(TFT9341\_RandColor(),i,i,239-i,319-i);**

**}**

***\_delay\_ms*(500);**

**TFT9341\_FillScreen(BLACK);**

Проверим тест на практике



Теперь напишем функцию отрисовки окружности с определённым радиусом

**//—————————————————————**

**void TFT9341\_DrawCircle(int x0, int y0, int r, unsigned int color)**

**{**

**int f = 1 – r;**

**int ddF\_x = 1;**

**int ddF\_y = -2 \* r;**

**int x = 0;**

**int y = r;**

**TFT9341\_DrawPixel(x0 , y0+r, color);**

**TFT9341\_DrawPixel(x0 , y0–r, color);**

**TFT9341\_DrawPixel(x0+r, y0 , color);**

**TFT9341\_DrawPixel(x0–r, y0 , color);**

**while (x<y) {**

**if (f >= 0) {**

**y–;**

**ddF\_y += 2;**

**f += ddF\_y;**

**}**

**x++;**

**ddF\_x += 2;**

**f += ddF\_x;**

**TFT9341\_DrawPixel(x0 + x, y0 + y, color);**

**TFT9341\_DrawPixel(x0 – x, y0 + y, color);**

**TFT9341\_DrawPixel(x0 + x, y0 – y, color);**

**TFT9341\_DrawPixel(x0 – x, y0 – y, color);**

**TFT9341\_DrawPixel(x0 + y, y0 + x, color);**

**TFT9341\_DrawPixel(x0 – y, y0 + x, color);**

**TFT9341\_DrawPixel(x0 + y, y0 – x, color);**

**TFT9341\_DrawPixel(x0 – y, y0 – x, color);**

**}**

**}**

**//—————————————————————**

Данная функция была взята из какого-то примера и переработана под наш дисплей.

Во входных параметрах координаты центра, радиус в пикселях и цвет.

Напишем для неё прототип и напишем тест в main(), в котором мы будем выводить в случайное место окружности радиусом в 20 пикселей

TFT9341\_FillScreen(BLACK);

**for(i=0;i<2000;i++)**

**{**

**TFT9341\_DrawCircle(*rand*()%200+20,*rand*()%280+20,20,TFT9341\_RandColor());**

**}**

***\_delay\_ms*(500);**

**TFT9341\_FillScreen(BLACK);**

Запас 20 пикселей в координатах сделан для того, чтобы окружности уместились полностью в видимой областью и не попали в область вне экрана.

Соберём код, прошьём контроллер и посмотрим результат работы нашего теста на практике



В данной заключительной части урока мы поработаем уже с текстом. Это будет нелёгкая работа, но зато успокаивет то что эта чать будет последняя в данном уроке.

В одной из предыдущих частей мы создали массив для шрифта размером в 16 пикселей, правда там не все символы, а только некоторые, так как размещение во флеш-памяти всех символов невозможно из-за её нехватки. Добавим также массив шрифта в 8 пикселей

//————————————————————–

//font 8

const unsigned char chars8[][8] PROGMEM ={

//SPACE

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},

// !

{0x04,0x04,0x04,0x04,0x00,0x04,0x00,0x00},

// "

{0x0A,0x0A,0x0A,0x00,0x00,0x00,0x00,0x00},

// #

{0x0A,0x0A,0x1F,0x0A,0x1F,0x0A,0x0A,0x00},

// $

{0x04,0x0F,0x14,0x0E,0x05,0x1E,0x04,0x00},

// %

{0x18,0x19,0x02,0x04,0x08,0x13,0x03,0x00},

// &

{0x0C,0x12,0x14,0x08,0x14,0x12,0x0D,0x00},

// '

{0x0C,0x04,0x08,0x00,0x00,0x00,0x00,0x00},

// (

{0x02,0x04,0x08,0x08,0x08,0x04,0x02,0x00},

// )

{0x08,0x04,0x02,0x02,0x02,0x04,0x08,0x00},

// \*

{0x00,0x04,0x15,0x0E,0x15,0x04,0x00,0x00},

// +

{0x00,0x04,0x04,0x1F,0x04,0x04,0x00,0x00},

// ,

{0x00,0x00,0x00,0x00,0x0C,0x04,0x08,0x00},

// –

{0x00,0x00,0x00,0x1F,0x00,0x00,0x00,0x00},

// .

{0x00,0x00,0x00,0x00,0x00,0x0C,0x0C,0x00},

// /

{0x00,0x01,0x02,0x04,0x08,0x10,0x00,0x00},

// 0

{0x0E,0x11,0x13,0x15,0x19,0x11,0x0E,0x00},

// 1

{0x04,0x0C,0x04,0x04,0x04,0x04,0x0E,0x00},

// 2

{0x0E,0x11,0x01,0x02,0x04,0x08,0x1F,0x00},

// 3

{0x1F,0x02,0x04,0x02,0x01,0x11,0x0E,0x00},

// 4

{0x02,0x06,0x0A,0x12,0x1F,0x02,0x02,0x00},

// 5

{0x1F,0x10,0x1E,0x01,0x01,0x11,0x0E,0x00},

// 6

{0x06,0x08,0x10,0x1E,0x11,0x11,0x0E,0x00},

// 7

{0x1F,0x01,0x02,0x04,0x08,0x08,0x08,0x00},

// 8

{0x0E,0x11,0x11,0x0E,0x11,0x11,0x0E,0x00},

// 9

{0x0E,0x11,0x11,0x0F,0x01,0x02,0x0C,0x00},

// :

{0x00,0x01,0x02,0x04,0x08,0x10,0x00,0x00},

// ;

{0x00,0x0C,0x0C,0x00,0x0C,0x04,0x08,0x00},

// <

{0x02,0x04,0x08,0x10,0x08,0x04,0x02,0x00},

// =

{0x00,0x00,0x1F,0x00,0x1F,0x00,0x00,0x00},

// >

{0x08,0x04,0x02,0x01,0x02,0x04,0x08,0x00},

// ?

{0x0E,0x11,0x01,0x02,0x04,0x00,0x04,0x00},

// @

{0x0E,0x11,0x01,0x0D,0x15,0x15,0x0E,0x00},

// A

{0x0E,0x11,0x11,0x11,0x1F,0x11,0x11,0x00},

// B

{0x1E,0x11,0x11,0x1E,0x11,0x11,0x1E,0x00},

// C

{0x0E,0x11,0x10,0x10,0x10,0x11,0x0E,0x00},

// D

{0x1C,0x12,0x11,0x11,0x11,0x12,0x1C,0x00},

// E

{0x1F,0x10,0x10,0x1E,0x10,0x10,0x1F,0x00},

// F

{0x1F,0x10,0x10,0x1E,0x10,0x10,0x10,0x00},

// G

{0x0E,0x11,0x10,0x17,0x11,0x11,0x0E,0x00},

// H

{0x11,0x11,0x11,0x1F,0x11,0x11,0x11,0x00},

// I

{0x0E,0x04,0x04,0x04,0x04,0x04,0x0E,0x00},

// J

{0x07,0x02,0x02,0x02,0x02,0x12,0x0C,0x00},

// K

{0x11,0x12,0x14,0x18,0x14,0x12,0x11,0x00},

// L

{0x10,0x10,0x10,0x10,0x10,0x10,0x1F,0x00},

// M

{0x11,0x1B,0x15,0x15,0x11,0x11,0x11,0x00},

// N

{0x11,0x11,0x19,0x15,0x13,0x11,0x11,0x00},

// O

{0x0E,0x11,0x11,0x11,0x11,0x11,0x0E,0x00},

// P

{0x1E,0x11,0x11,0x1E,0x10,0x10,0x10,0x00},

// Q

{0x0E,0x11,0x11,0x11,0x15,0x12,0x0D,0x00},

// R

{0x1E,0x11,0x11,0x1E,0x14,0x12,0x11,0x00},

// S

{0x0F,0x10,0x10,0x0E,0x01,0x01,0x1E,0x00},

// T

{0x1F,0x04,0x04,0x04,0x04,0x04,0x04,0x00},

// U

{0x11,0x11,0x11,0x11,0x11,0x11,0x0E,0x00},

// V

{0x11,0x11,0x11,0x11,0x11,0x0A,0x04,0x00},

// W

{0x11,0x11,0x11,0x11,0x15,0x15,0x0E,0x00},

// X

{0x11,0x11,0x0A,0x04,0x0A,0x11,0x11,0x00},

// Y

{0x11,0x11,0x11,0x0A,0x04,0x04,0x04,0x00},

// Z

{0x1F,0x01,0x02,0x04,0x08,0x10,0x1F,0x00},

// [

{0x0E,0x08,0x08,0x08,0x08,0x08,0x0E,0x00},

//

{0x11,0x0A,0x1F,0x04,0x1F,0x04,0x04,0x00},

// ]

{0x0E,0x02,0x02,0x02,0x02,0x02,0x0E,0x00},

// ^

{0x04,0x0A,0x11,0x00,0x00,0x00,0x00,0x00},

// \_

{0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0x00},

// '

{0x08,0x04,0x00,0x00,0x00,0x00,0x00,0x00},

// a

{0x00,0x00,0x0E,0x01,0x0F,0x11,0x0F,0x00},

// b

{0x10,0x10,0x1E,0x11,0x11,0x11,0x1E,0x00},

// c

{0x00,0x00,0x0E,0x10,0x10,0x11,0x0E,0x00},

// d

{0x01,0x01,0x0D,0x13,0x11,0x11,0x0F,0x00},

// e

{0x00,0x00,0x0E,0x11,0x1F,0x10,0x0E,0x00},

// f

{0x06,0x09,0x08,0x1C,0x08,0x08,0x08,0x00},

// g

{0x00,0x0F,0x11,0x11,0x0F,0x01,0x0E,0x00},

// h

{0x10,0x10,0x16,0x19,0x11,0x11,0x11,0x00},

// i

{0x04,0x00,0x0C,0x04,0x04,0x04,0x0E,0x00},

// j

{0x02,0x00,0x06,0x02,0x02,0x12,0x0C,0x00},

// k

{0x10,0x10,0x12,0x14,0x18,0x14,0x12,0x00},

// l

{0x18,0x08,0x08,0x08,0x08,0x08,0x1C,0x00},

// m

{0x00,0x00,0x1A,0x15,0x15,0x11,0x11,0x00},

// n

{0x00,0x00,0x16,0x19,0x11,0x11,0x11,0x00},

// o

{0x00,0x00,0x0E,0x11,0x11,0x11,0x0E,0x00},

// p

{0x00,0x00,0x1E,0x11,0x1E,0x10,0x10,0x00},

// q

{0x00,0x00,0x0F,0x11,0x0F,0x01,0x01,0x00},

// r

{0x00,0x00,0x16,0x19,0x10,0x10,0x10,0x00},

// s

{0x00,0x00,0x0E,0x10,0x0E,0x01,0x1E,0x00},

// t

{0x08,0x08,0x1C,0x08,0x08,0x09,0x06,0x00},

// u

{0x00,0x00,0x11,0x11,0x11,0x13,0x0D,0x00},

// v

{0x00,0x00,0x11,0x11,0x11,0x0A,0x04,0x00},

// w

{0x00,0x00,0x11,0x11,0x11,0x15,0x0A,0x00},

// x

{0x00,0x00,0x11,0x0A,0x04,0x0A,0x11,0x00},

// y

{0x00,0x00,0x11,0x11,0x0F,0x01,0x0E,0x00},

// z

{0x00,0x00,0x1F,0x02,0x04,0x08,0x1F,0x00}

};

//————————————————————–

Теперь добавим функцию по выводу символа определённой высоты в определённое место экрана

//—————————————————————

void TFT9341\_Draw\_Char(int x, int y, unsigned int color, unsigned int phone, unsigned char charcode, unsigned char size)

{

}

//—————————————————————

Во входных параметрах будут координаты вывода символа, цвет символа, цвет фона, код символа и размер символа.

Добавим возможность отслеживания размера с помощью операции ветвления

void TFT9341\_Draw\_Char(int x, int y, unsigned int color, unsigned int phone, unsigned char charcode, unsigned char size)

{

  switch(size)

  {

    int i,h;

    case 1:

    break;

    case 2:

    break;

    case 3:

    break;

    case 4:

    break;

  }

}

вариант 1 будет у нас для шрифта размером 8 пикселей, а вариант 2 для 16 пиксельного, остальные два варианта пока будут в запасе.

Начнем заполнять первый кейс, написав туда двухмерный цикл

case 1:

  for(h=0;h<8;h++)

  {

    for(i=0;i<8;i++)

    {

    }

  }

  break;

В принципе, несложно догадаться, что мы из элемента массива собираем пиксели по горизонтали (i) и по вертикали (h).

Теперь внутри внутреннего цикла будет условие, которое в зависимости от значения проверяемого бита в элементе массива будет отрисовывать точку определённого цвета. При единице будет цвет символа, а при нуле – цвет фона

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

{

  if ((pgm\_read\_byte(&chars8[charcode-0x20][h])>>(7-i))&0x01)

  {

    TFT9341\_DrawPixel(x+i,y+h,color);

  }

  else

  {

    TFT9341\_DrawPixel(x+i,y+h,phone);

  }

}

В случае второго варианта (размер шрифта 16 пикселей) будет также двухмерный цикл, только h уже будет доходить до 16, а не до восьми

case 2:

  for(h=0;h<16;h++)

  {

    for(i=0;i<8;i++)

    {

    }

  }

  break;

Далее в теле внутреннего цикла у нас будет уже два условия, одно – для первого байта горизонтали, а второе – для второго

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

{

  if ((pgm\_read\_byte(&chars16[charcode-0x20][h\*2])>>(7-i))&0x01)

  {

    TFT9341\_DrawPixel(x+i,y+h,color);

  }

  else

  {

    TFT9341\_DrawPixel(x+i,y+h,phone);

  }

  if ((pgm\_read\_byte(&chars16[charcode-0x20][h\*2+1])>>(7-i))&0x01)

  {

    TFT9341\_DrawPixel(x+i+8,y+h,color);

  }

  else

  {

    TFT9341\_DrawPixel(x+i+8,y+h,phone);

  }

}

Создадим прототип нашей функции и попробуем в функции main() что-то написать на экране

TFT9341\_FillScreen(BLACK);

TFT9341\_Draw\_Char(10,10,RED,GREEN,0x21,2);

TFT9341\_Draw\_Char(26,10,RED,GREEN,0x22,2);

TFT9341\_Draw\_Char(42,10,RED,GREEN,0x23,2);

TFT9341\_Draw\_Char(58,10,RED,GREEN,0x24,2);

TFT9341\_Draw\_Char(74,10,RED,GREEN,0x25,2);

TFT9341\_Draw\_Char(10,26,RED,GREEN,0x21,1);

TFT9341\_Draw\_Char(18,26,RED,GREEN,0x22,1);

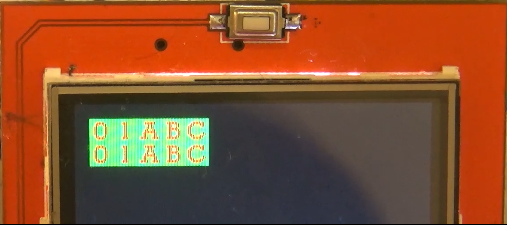
TFT9341\_Draw\_Char(26,26,RED,GREEN,0x23,1);

TFT9341\_Draw\_Char(34,26,RED,GREEN,0x24,1);

TFT9341\_Draw\_Char(42,26,RED,GREEN,0x25,1);

\_delay\_ms(1000);

Соберём код, прошьём контроллер и посмотрим результат



Теперь попробуем написать функцию отрисовки целой строки на экране

//—————————————————————

void TFT9341\_String(unsigned int x, unsigned int y, unsigned int color,unsigned int phone,char \*str, unsigned char size)

{

  while(\*str)

  {

    if((x + (size\*8)) > X\_SIZE)

    {

      x = 1;

      y = y + (size\*8);

    }

    TFT9341\_Draw\_Char(x, y, color, phone,\*str, size);

    x += size\*8;

    \*str++;

  }

}

//—————————————————————

При кажущейся сложности отрисовки строки, функция оказалась не такой и тяжёлой. Постепенно сдвигая указатель на следующий символ в строке и отслеживая конец строки, мы потихоньку выводим символы слева направо. Но также не забываем отслеживать правый прай дисплея,чтобы не уйти за него.

Также напишем прототип данной функции и в main() напишем тест для проверки данной функции

\_delay\_ms(1000);

TFT9341\_FillScreen(BLACK);

TFT9341\_String(1, 1, RED,GREEN,"12345ABCDE", 1);

TFT9341\_String(1, 9, RED,GREEN,"EDCBA54321", 1);

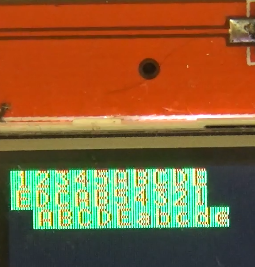
TFT9341\_String(10, 17, RED,GREEN,"ABCDEabcde", 1);

\_delay\_ms(1000);

TFT9341\_FillScreen(BLACK);

while (1)

Соберём код, прошьём контроллер и посмотрим наши строки на практике



А также в бесконечный цикл мы напишем тест вывода случайных нулей и единиц размером 8 пикселей в случайные места экрана

  TFT9341\_FillScreen(BLACK);

  while (1)

  {

    TFT9341\_Draw\_Char((rand()%15)\*16,(rand()%20)\*16,GREEN,BLACK,0x21+(rand()%2),2);

  }

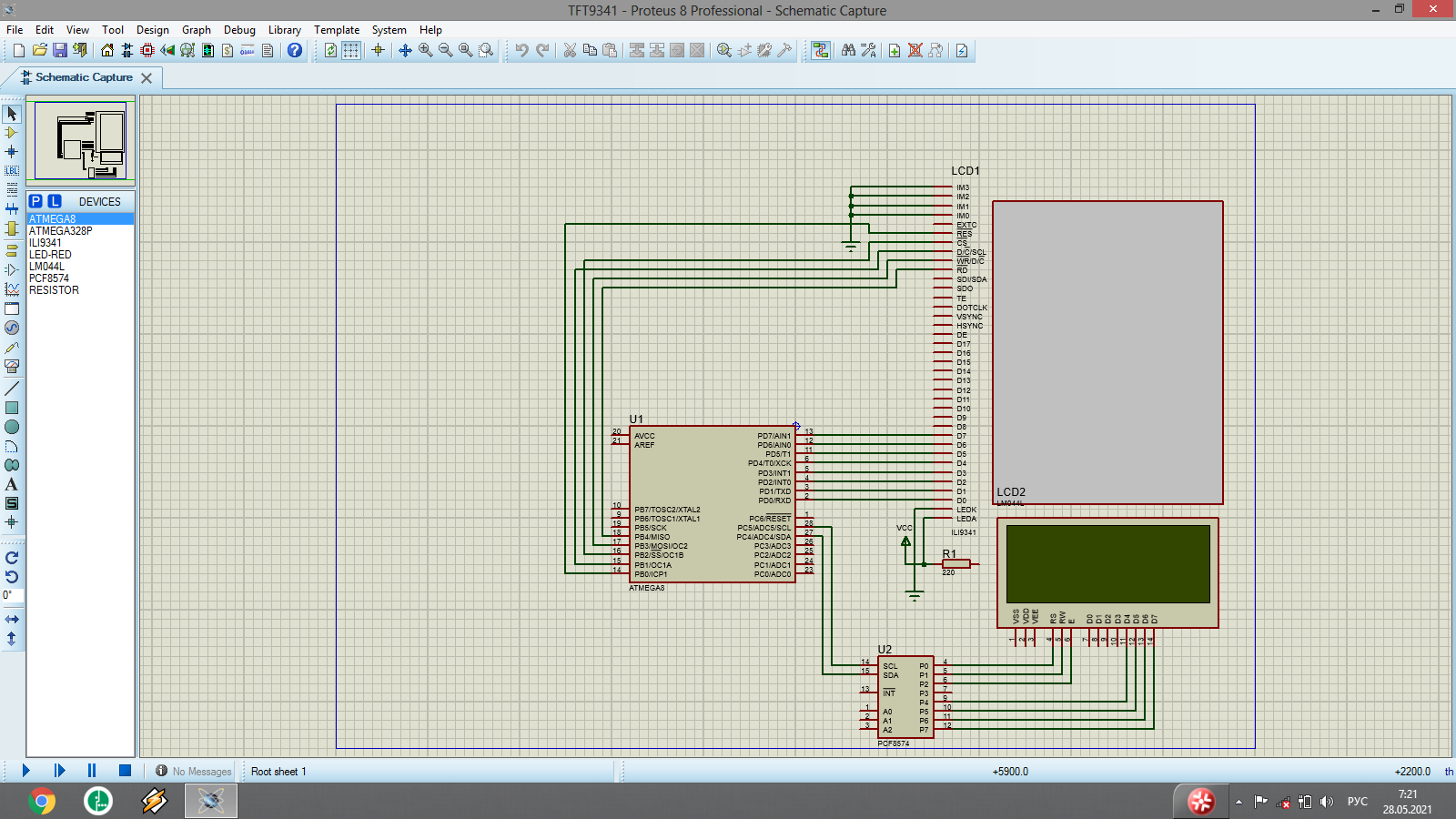
}

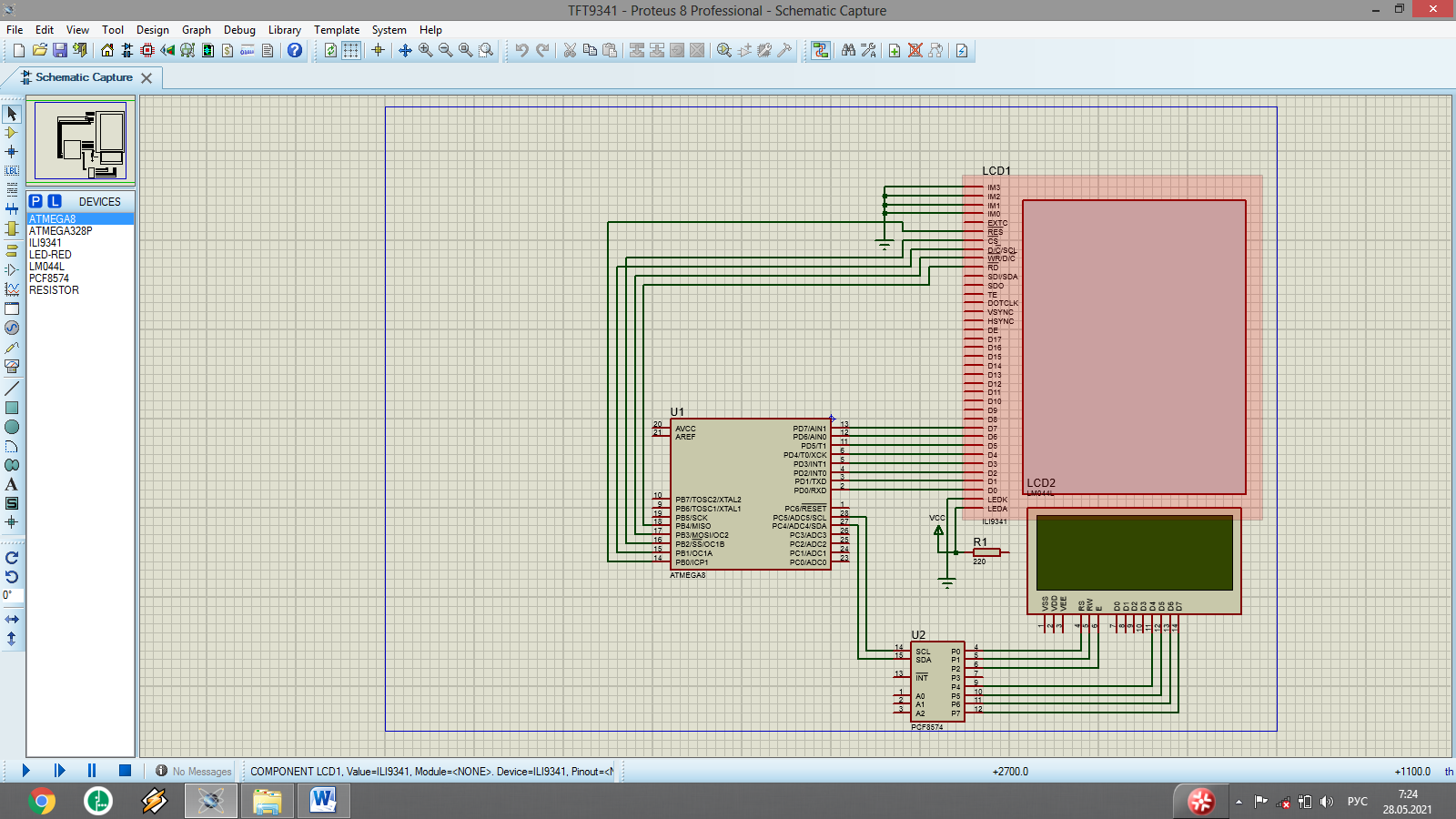
Скомпилируем наш код, прошьём контроллер и посмотрим результат

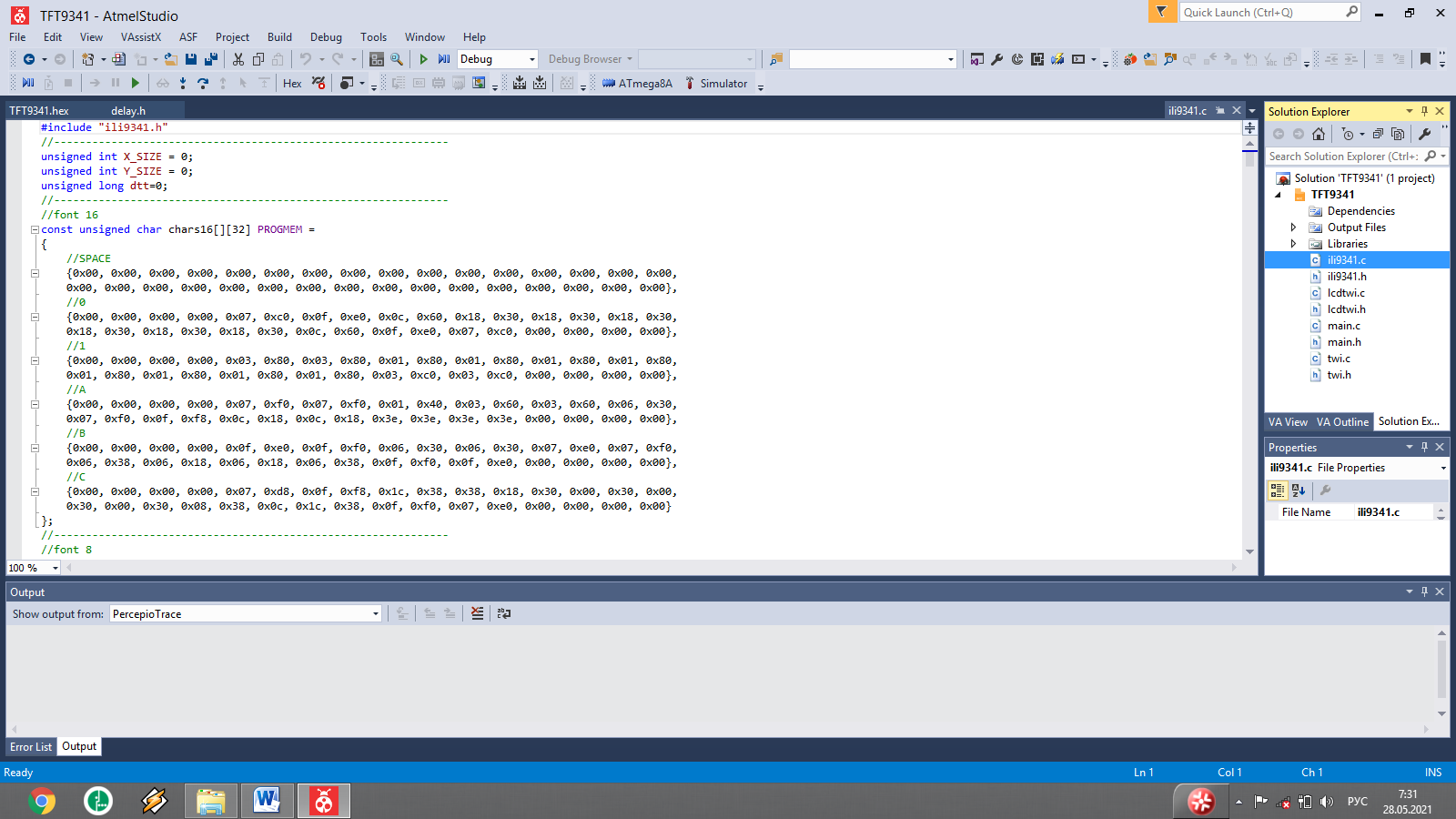


  Я думаю, просмотрев полностью данное занятие, вы стали гораздо грамотнее в программировании микроконтроллеров, а также немного начали разбираться в написании кода для контроллеров дисплеев с параллельным подключением.

**ПРОГРАММНЫЙ КОД ВНИЗУ**







#include "ili9341.h"

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

unsigned int X\_SIZE = 0;

unsigned int Y\_SIZE = 0;

unsigned long dtt=0;

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

//font 16

const unsigned char chars16[][32] PROGMEM =

{

//SPACE

{0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},

//0

{0x00, 0x00, 0x00, 0x00, 0x07, 0xc0, 0x0f, 0xe0, 0x0c, 0x60, 0x18, 0x30, 0x18, 0x30, 0x18, 0x30,

0x18, 0x30, 0x18, 0x30, 0x18, 0x30, 0x0c, 0x60, 0x0f, 0xe0, 0x07, 0xc0, 0x00, 0x00, 0x00, 0x00},

//1

{0x00, 0x00, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x01, 0x80, 0x01, 0x80, 0x01, 0x80, 0x01, 0x80,

0x01, 0x80, 0x01, 0x80, 0x01, 0x80, 0x01, 0x80, 0x03, 0xc0, 0x03, 0xc0, 0x00, 0x00, 0x00, 0x00},

//A

{0x00, 0x00, 0x00, 0x00, 0x07, 0xf0, 0x07, 0xf0, 0x01, 0x40, 0x03, 0x60, 0x03, 0x60, 0x06, 0x30,

0x07, 0xf0, 0x0f, 0xf8, 0x0c, 0x18, 0x0c, 0x18, 0x3e, 0x3e, 0x3e, 0x3e, 0x00, 0x00, 0x00, 0x00},

//B

{0x00, 0x00, 0x00, 0x00, 0x0f, 0xe0, 0x0f, 0xf0, 0x06, 0x30, 0x06, 0x30, 0x07, 0xe0, 0x07, 0xf0,

0x06, 0x38, 0x06, 0x18, 0x06, 0x18, 0x06, 0x38, 0x0f, 0xf0, 0x0f, 0xe0, 0x00, 0x00, 0x00, 0x00},

//C

{0x00, 0x00, 0x00, 0x00, 0x07, 0xd8, 0x0f, 0xf8, 0x1c, 0x38, 0x38, 0x18, 0x30, 0x00, 0x30, 0x00,

0x30, 0x00, 0x30, 0x08, 0x38, 0x0c, 0x1c, 0x38, 0x0f, 0xf0, 0x07, 0xe0, 0x00, 0x00, 0x00, 0x00}

};

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

//font 8

const unsigned char chars8[][8] PROGMEM ={

//SPACE

{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},

// !

{0x04,0x04,0x04,0x04,0x00,0x04,0x00,0x00},

// "

{0x0A,0x0A,0x0A,0x00,0x00,0x00,0x00,0x00},

// #

{0x0A,0x0A,0x1F,0x0A,0x1F,0x0A,0x0A,0x00},

// $

{0x04,0x0F,0x14,0x0E,0x05,0x1E,0x04,0x00},

// %

{0x18,0x19,0x02,0x04,0x08,0x13,0x03,0x00},

// &

{0x0C,0x12,0x14,0x08,0x14,0x12,0x0D,0x00},

// '

{0x0C,0x04,0x08,0x00,0x00,0x00,0x00,0x00},

// (

{0x02,0x04,0x08,0x08,0x08,0x04,0x02,0x00},

// )

{0x08,0x04,0x02,0x02,0x02,0x04,0x08,0x00},

// \*

{0x00,0x04,0x15,0x0E,0x15,0x04,0x00,0x00},

// +

{0x00,0x04,0x04,0x1F,0x04,0x04,0x00,0x00},

// ,

{0x00,0x00,0x00,0x00,0x0C,0x04,0x08,0x00},

// -

{0x00,0x00,0x00,0x1F,0x00,0x00,0x00,0x00},

// .

{0x00,0x00,0x00,0x00,0x00,0x0C,0x0C,0x00},

// /

{0x00,0x01,0x02,0x04,0x08,0x10,0x00,0x00},

// 0

{0x0E,0x11,0x13,0x15,0x19,0x11,0x0E,0x00},

// 1

{0x04,0x0C,0x04,0x04,0x04,0x04,0x0E,0x00},

// 2

{0x0E,0x11,0x01,0x02,0x04,0x08,0x1F,0x00},

// 3

{0x1F,0x02,0x04,0x02,0x01,0x11,0x0E,0x00},

// 4

{0x02,0x06,0x0A,0x12,0x1F,0x02,0x02,0x00},

// 5

{0x1F,0x10,0x1E,0x01,0x01,0x11,0x0E,0x00},

// 6

{0x06,0x08,0x10,0x1E,0x11,0x11,0x0E,0x00},

// 7

{0x1F,0x01,0x02,0x04,0x08,0x08,0x08,0x00},

// 8

{0x0E,0x11,0x11,0x0E,0x11,0x11,0x0E,0x00},

// 9

{0x0E,0x11,0x11,0x0F,0x01,0x02,0x0C,0x00},

// :

{0x00,0x01,0x02,0x04,0x08,0x10,0x00,0x00},

// ;

{0x00,0x0C,0x0C,0x00,0x0C,0x04,0x08,0x00},

// <

{0x02,0x04,0x08,0x10,0x08,0x04,0x02,0x00},

// =

{0x00,0x00,0x1F,0x00,0x1F,0x00,0x00,0x00},

// >

{0x08,0x04,0x02,0x01,0x02,0x04,0x08,0x00},

// ?

{0x0E,0x11,0x01,0x02,0x04,0x00,0x04,0x00},

// @

{0x0E,0x11,0x01,0x0D,0x15,0x15,0x0E,0x00},

// A

{0x0E,0x11,0x11,0x11,0x1F,0x11,0x11,0x00},

// B

{0x1E,0x11,0x11,0x1E,0x11,0x11,0x1E,0x00},

// C

{0x0E,0x11,0x10,0x10,0x10,0x11,0x0E,0x00},

// D

{0x1C,0x12,0x11,0x11,0x11,0x12,0x1C,0x00},

// E

{0x1F,0x10,0x10,0x1E,0x10,0x10,0x1F,0x00},

// F

{0x1F,0x10,0x10,0x1E,0x10,0x10,0x10,0x00},

// G

{0x0E,0x11,0x10,0x17,0x11,0x11,0x0E,0x00},

// H

{0x11,0x11,0x11,0x1F,0x11,0x11,0x11,0x00},

// I

{0x0E,0x04,0x04,0x04,0x04,0x04,0x0E,0x00},

// J

{0x07,0x02,0x02,0x02,0x02,0x12,0x0C,0x00},

// K

{0x11,0x12,0x14,0x18,0x14,0x12,0x11,0x00},

// L

{0x10,0x10,0x10,0x10,0x10,0x10,0x1F,0x00},

// M

{0x11,0x1B,0x15,0x15,0x11,0x11,0x11,0x00},

// N

{0x11,0x11,0x19,0x15,0x13,0x11,0x11,0x00},

// O

{0x0E,0x11,0x11,0x11,0x11,0x11,0x0E,0x00},

// P

{0x1E,0x11,0x11,0x1E,0x10,0x10,0x10,0x00},

// Q

{0x0E,0x11,0x11,0x11,0x15,0x12,0x0D,0x00},

// R

{0x1E,0x11,0x11,0x1E,0x14,0x12,0x11,0x00},

// S

{0x0F,0x10,0x10,0x0E,0x01,0x01,0x1E,0x00},

// T

{0x1F,0x04,0x04,0x04,0x04,0x04,0x04,0x00},

// U

{0x11,0x11,0x11,0x11,0x11,0x11,0x0E,0x00},

// V

{0x11,0x11,0x11,0x11,0x11,0x0A,0x04,0x00},

// W

{0x11,0x11,0x11,0x11,0x15,0x15,0x0E,0x00},

// X

{0x11,0x11,0x0A,0x04,0x0A,0x11,0x11,0x00},

// Y

{0x11,0x11,0x11,0x0A,0x04,0x04,0x04,0x00},

// Z

{0x1F,0x01,0x02,0x04,0x08,0x10,0x1F,0x00},

// [

{0x0E,0x08,0x08,0x08,0x08,0x08,0x0E,0x00},

//

{0x11,0x0A,0x1F,0x04,0x1F,0x04,0x04,0x00},

// ]

{0x0E,0x02,0x02,0x02,0x02,0x02,0x0E,0x00},

// ^

{0x04,0x0A,0x11,0x00,0x00,0x00,0x00,0x00},

// \_

{0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0x00},

// '

{0x08,0x04,0x00,0x00,0x00,0x00,0x00,0x00},

// a

{0x00,0x00,0x0E,0x01,0x0F,0x11,0x0F,0x00},

// b

{0x10,0x10,0x1E,0x11,0x11,0x11,0x1E,0x00},

// c

{0x00,0x00,0x0E,0x10,0x10,0x11,0x0E,0x00},

// d

{0x01,0x01,0x0D,0x13,0x11,0x11,0x0F,0x00},

// e

{0x00,0x00,0x0E,0x11,0x1F,0x10,0x0E,0x00},

// f

{0x06,0x09,0x08,0x1C,0x08,0x08,0x08,0x00},

// g

{0x00,0x0F,0x11,0x11,0x0F,0x01,0x0E,0x00},

// h

{0x10,0x10,0x16,0x19,0x11,0x11,0x11,0x00},

// i

{0x04,0x00,0x0C,0x04,0x04,0x04,0x0E,0x00},

// j

{0x02,0x00,0x06,0x02,0x02,0x12,0x0C,0x00},

// k

{0x10,0x10,0x12,0x14,0x18,0x14,0x12,0x00},

// l

{0x18,0x08,0x08,0x08,0x08,0x08,0x1C,0x00},

// m

{0x00,0x00,0x1A,0x15,0x15,0x11,0x11,0x00},

// n

{0x00,0x00,0x16,0x19,0x11,0x11,0x11,0x00},

// o

{0x00,0x00,0x0E,0x11,0x11,0x11,0x0E,0x00},

// p

{0x00,0x00,0x1E,0x11,0x1E,0x10,0x10,0x00},

// q

{0x00,0x00,0x0F,0x11,0x0F,0x01,0x01,0x00},

// r

{0x00,0x00,0x16,0x19,0x10,0x10,0x10,0x00},

// s

{0x00,0x00,0x0E,0x10,0x0E,0x01,0x1E,0x00},

// t

{0x08,0x08,0x1C,0x08,0x08,0x09,0x06,0x00},

// u

{0x00,0x00,0x11,0x11,0x11,0x13,0x0D,0x00},

// v

{0x00,0x00,0x11,0x11,0x11,0x0A,0x04,0x00},

// w

{0x00,0x00,0x11,0x11,0x11,0x15,0x0A,0x00},

// x

{0x00,0x00,0x11,0x0A,0x04,0x0A,0x11,0x00},

// y

{0x00,0x00,0x11,0x11,0x0F,0x01,0x0E,0x00},

// z

{0x00,0x00,0x1F,0x02,0x04,0x08,0x1F,0x00}

};

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

void port\_ini(void)

{

DATA\_PORT=0x00;

DATA\_DDR=0xFF;//Шина данных на выход

COMMAND\_DDR=0x1F;//Командные лапки также все на выход

}

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

void TFT9341\_SendCommand(unsigned char cmd)

{

CD\_COMMAND;//лапка в состоянии посылки команды

RD\_IDLE;//отключим чтение

CS\_ACTIVE;//выбор дисплея

DATA\_PORT=cmd;

WR\_STROBE;

CS\_IDLE;

}

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

void TFT9341\_SendData(unsigned char dt)

{

CD\_DATA;//лапка в состоянии посылки данных

RD\_IDLE;//отключим чтение

CS\_ACTIVE;//выбор дисплея

DATA\_PORT=dt;

WR\_STROBE;

CS\_IDLE;

}

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

void TFT9341\_Write8(unsigned char dt)

{

DATA\_PORT=dt;

WR\_STROBE;

}

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

unsigned long TFT9341\_ReadReg(unsigned char r)

{

unsigned long id;

unsigned char x;

CS\_ACTIVE;//выбор дисплея

CD\_COMMAND;//лапка в состоянии посылки команды

TFT9341\_Write8(r);

setReadDir();

CD\_DATA;

*\_delay\_us*(50);

RD\_ACTIVE;

*\_delay\_us*(5);

x=DATA\_PIN;

RD\_IDLE;

id=x;

id<<=8;

RD\_ACTIVE;

*\_delay\_us*(5);

x=DATA\_PIN;

RD\_IDLE;

id|=x;

id<<=8;

RD\_ACTIVE;

*\_delay\_us*(5);

x=DATA\_PIN;

RD\_IDLE;

id|=x;

id<<=8;

RD\_ACTIVE;

*\_delay\_us*(5);

x=DATA\_PIN;

RD\_IDLE;

id|=x;

if(r==0xEF)

{

id<<=8;

RD\_ACTIVE;

*\_delay\_us*(5);

x=DATA\_PIN;

RD\_IDLE;

id|=x;

}

CS\_IDLE;

setWriteDir();

*\_delay\_us*(150);//stabilization time

return id;

}

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

void TFT9341\_reset(void)

{

CS\_IDLE;

WR\_IDLE;

RD\_IDLE;

RESET\_ACTIVE;

*\_delay\_ms*(2);

RESET\_IDLE;

CS\_ACTIVE;

TFT9341\_SendCommand(0x01); //Software Reset

for (*uint8\_t* i=0;i<3;i++) WR\_STROBE;

CS\_IDLE;

}

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

void TFT9341\_SetRotation(unsigned char r)

{

TFT9341\_SendCommand(0x36);

switch(r)

{

case 0:

TFT9341\_SendData(0x48);

X\_SIZE = 240;

Y\_SIZE = 320;

break;

case 1:

TFT9341\_SendData(0x28);

X\_SIZE = 320;

Y\_SIZE = 240;

break;

case 2:

TFT9341\_SendData(0x88);

X\_SIZE = 240;

Y\_SIZE = 320;

break;

case 3:

TFT9341\_SendData(0xE8);

X\_SIZE = 320;

Y\_SIZE = 240;

break;

}

}

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

void TFT9341\_Flood(unsigned short color, unsigned long len)

{

unsigned short blocks;

unsigned char i, hi = color>>8, lo=color;

CS\_ACTIVE;

CD\_COMMAND;

TFT9341\_Write8(0x2C);

CD\_DATA;

TFT9341\_Write8(hi);

TFT9341\_Write8(lo);

len--;

blocks=(unsigned short)(len/64);//64 pixels/block

if (hi==lo)

{

while(blocks--)

{

i=16;

do

{

WR\_STROBE;WR\_STROBE;WR\_STROBE;WR\_STROBE;//2bytes/pixel

WR\_STROBE;WR\_STROBE;WR\_STROBE;WR\_STROBE;//x4 pixel

} while (--i);

}

//Fill any remaining pixels(1 to 64)

for (i=(unsigned char)len&63;i--;)

{

WR\_STROBE;

WR\_STROBE;

}

}

else

{

while(blocks--)

{

i=16;

do

{

TFT9341\_Write8(hi);TFT9341\_Write8(lo);TFT9341\_Write8(hi);TFT9341\_Write8(lo);

TFT9341\_Write8(hi);TFT9341\_Write8(lo);TFT9341\_Write8(hi);TFT9341\_Write8(lo);

} while (--i);

}

//Fill any remaining pixels(1 to 64)

for (i=(unsigned char)len&63;i--;)

{

TFT9341\_Write8(hi);

TFT9341\_Write8(lo);

}

}

CS\_IDLE;

}

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

void TFT9341\_WriteRegister32(unsigned char r, unsigned long d)

{

CS\_ACTIVE;

CD\_COMMAND;

TFT9341\_Write8(r);

CD\_DATA;

*\_delay\_us*(1);

TFT9341\_Write8(d>>24);

*\_delay\_us*(1);

TFT9341\_Write8(d>>16);

*\_delay\_us*(1);

TFT9341\_Write8(d>>8);

*\_delay\_us*(1);

TFT9341\_Write8(d);

CS\_IDLE;

}

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

void TFT9341\_SetAddrWindow(unsigned int x1,unsigned int y1,unsigned int x2,unsigned int y2)

{

unsigned long t;

CS\_ACTIVE;

t = x1;

t<<=16;

t |= x2;

TFT9341\_WriteRegister32(0x2A,t);//Column Addres Set

t = y1;

t<<=16;

t |= y2;

TFT9341\_WriteRegister32(0x2B,t);//Page Addres Set

CS\_IDLE;

}

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

void TFT9341\_FillScreen(unsigned int color)

{

TFT9341\_SetAddrWindow(0,0,X\_SIZE-1,Y\_SIZE-1);

TFT9341\_Flood(color,(long)X\_SIZE\*(long)Y\_SIZE);

}

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

void TFT9341\_FillRectangle(unsigned int color,unsigned int x1, unsigned int y1,

unsigned int x2, unsigned int y2)

{

TFT9341\_SetAddrWindow(x1, y1, x2, y2);

TFT9341\_Flood(color, (long)(x2-x1+1) \* (long)(y2-y1+1));

}

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

void TFT9341\_DrawPixel(int x, int y, unsigned int color)

{

if((x<0)||(y<0)||(x>=X\_SIZE)||(y>=Y\_SIZE)) return;

CS\_ACTIVE;

TFT9341\_SetAddrWindow(x,y,X\_SIZE-1,Y\_SIZE-1);

CS\_ACTIVE;

CD\_COMMAND;

TFT9341\_Write8(0x2C);

CD\_DATA;

TFT9341\_Write8(color>>8);TFT9341\_Write8(color);

CS\_IDLE;

}

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

void TFT9341\_DrawLine(unsigned int color,unsigned int x1, unsigned int y1,

unsigned int x2, unsigned int y2)

{

int steep = *abs*(y2-y1)>*abs*(x2-x1);

if (steep)

{

swap(x1,y1);

swap(x2,y2);

}

if(x1>x2)

{

swap(x1,x2);

swap(y1,y2);

}

int dx,dy;

dx=x2-x1;

dy=*abs*(y2-y1);

int err=dx/2;

int ystep;

if(y1<y2) ystep = 1;

else ystep = -1;

for (;x1<=x2;x1++)

{

if (steep) TFT9341\_DrawPixel(y1,x1,color);

else TFT9341\_DrawPixel(x1,y1,color);

err-=dy;

if (err<0)

{

y1 += ystep;

err=dx;

}

}

}

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

void TFT9341\_DrawRect(unsigned int color,unsigned int x1, unsigned int y1,

unsigned int x2, unsigned int y2)

{

TFT9341\_DrawLine(color,x1,y1,x2,y1);

TFT9341\_DrawLine(color,x2,y1,x2,y2);

TFT9341\_DrawLine(color,x1,y1,x1,y2);

TFT9341\_DrawLine(color,x1,y2,x2,y2);

}

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

void TFT9341\_DrawCircle(unsigned int x0, unsigned int y0, int r, unsigned int color)

{

int f = 1-r;

int ddF\_x=1;

int ddF\_y=-2\*r;

int x = 0;

int y = r;

TFT9341\_DrawPixel(x0,y0+r,color);

TFT9341\_DrawPixel(x0,y0-r,color);

TFT9341\_DrawPixel(x0+r,y0,color);

TFT9341\_DrawPixel(x0-r,y0,color);

while (x<y)

{

if (f>=0)

{

y--;

ddF\_y+=2;

f+=ddF\_y;

}

x++;

ddF\_x+=2;

f+=ddF\_x;

TFT9341\_DrawPixel(x0+x,y0+y,color);

TFT9341\_DrawPixel(x0-x,y0+y,color);

TFT9341\_DrawPixel(x0+x,y0-y,color);

TFT9341\_DrawPixel(x0-x,y0-y,color);

TFT9341\_DrawPixel(x0+y,y0+x,color);

TFT9341\_DrawPixel(x0-y,y0+x,color);

TFT9341\_DrawPixel(x0+y,y0-x,color);

TFT9341\_DrawPixel(x0-y,y0-x,color);

}

}

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

unsigned int TFT9341\_RandColor(void)

{

unsigned char c = *rand*()%8;

switch(c)

{

case 0:

return BLACK;

break;

case 1:

return BLUE;

break;

case 2:

return RED;

break;

case 3:

return GREEN;

break;

case 4:

return CYAN;

break;

case 5:

return MAGENTA;

break;

case 6:

return YELLOW;

break;

case 7:

return WHITE;

break;

}

return BLACK;

}

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

void TFT9341\_Draw\_Char(int x, int y, unsigned int color, unsigned int phone,

unsigned char charcode, unsigned char size)

{

switch(size)

{

int i,h;

case 1:

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

{

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

{

if ((pgm\_read\_byte(&chars8[charcode-0x20][h])>>(7-i))&0x01)

{

TFT9341\_DrawPixel(x+i,y+h,color);

}

else

{

TFT9341\_DrawPixel(x+i,y+h,phone);

}

}

}

break;

case 2:

for(h=0;h<16;h++)

{

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

{

if ((pgm\_read\_byte(&chars16[charcode-0x20][h\*2])>>(7-i))&0x01)

{

TFT9341\_DrawPixel(x+i,y+h,color);

}

else

{

TFT9341\_DrawPixel(x+i,y+h,phone);

}

if ((pgm\_read\_byte(&chars16[charcode-0x20][h\*2+1])>>(7-i))&0x01)

{

TFT9341\_DrawPixel(x+i+8,y+h,color);

}

else

{

TFT9341\_DrawPixel(x+i+8,y+h,phone);

}

}

}

break;

}

}

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

void TFT9341\_String(unsigned int x, unsigned int y, unsigned int color, unsigned int phone,

char \*str, unsigned char size)

{

while (\*str)

{

if ((x+(size\*8))>X\_SIZE)

{

x = 1;

y = y + (size\*8);

}

TFT9341\_Draw\_Char(x,y,color,phone,\*str,size);

x += size\*8;

\*str++;

}

}

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

void TFT9341\_ini(void)

{

char str[10];

port\_ini();

TFT9341\_reset();

*\_delay\_ms*(1000);

dtt=TFT9341\_ReadReg(0xD3);

CS\_ACTIVE;

setpos(0,1);

*sprintf*(str,"0x%08lX",dtt);

str\_lcd(str);

TFT9341\_SendCommand(0x01);//Software Reset

TFT9341\_SendCommand(0xCB);//Power Control A

TFT9341\_SendData(0x39);

TFT9341\_SendData(0x2C);

TFT9341\_SendData(0x00);

TFT9341\_SendData(0x34);

TFT9341\_SendData(0x02);

TFT9341\_SendCommand(0xCF);//Power Control B

TFT9341\_SendData(0x00);

TFT9341\_SendData(0xC1);

TFT9341\_SendData(0x30);

TFT9341\_SendCommand(0xE8);//Driver timing control A

TFT9341\_SendData(0x85);

TFT9341\_SendData(0x00);

TFT9341\_SendData(0x78);

TFT9341\_SendCommand(0xEA);//Driver timing control B

TFT9341\_SendData(0x00);

TFT9341\_SendData(0x00);

TFT9341\_SendCommand(0xED);//Power on Sequence control

TFT9341\_SendData(0x64);

TFT9341\_SendData(0x03);

TFT9341\_SendData(0x12);

TFT9341\_SendData(0x81);

TFT9341\_SendCommand(0xF7);//Pump ratio control

TFT9341\_SendData(0x20);

TFT9341\_SendCommand(0xC0);//Power Control 1

TFT9341\_SendData(0x10);

TFT9341\_SendCommand(0xC1);//Power Control 2

TFT9341\_SendData(0x10);

TFT9341\_SendCommand(0xC5);//VCOM Control 1

TFT9341\_SendData(0x3E);

TFT9341\_SendData(0x28);

TFT9341\_SendCommand(0xC7);//VCOM Control 2

TFT9341\_SendData(0x86);

TFT9341\_SetRotation(0);

TFT9341\_SendCommand(0x3A);//Pixel Format Set

TFT9341\_SendData(0x55);//16bit

TFT9341\_SendCommand(0xB1);

TFT9341\_SendData(0x00);

TFT9341\_SendData(0x18);// Частота кадров 79 Гц

TFT9341\_SendCommand(0xB6);//Display Function Control

TFT9341\_SendData(0x08);

TFT9341\_SendData(0x82);

TFT9341\_SendData(0x27);//320 строк

TFT9341\_SendCommand(0xF2);//Enable 3G (пока не знаю что это за режим)

TFT9341\_SendData(0x00);//не включаем

TFT9341\_SendCommand(0x26);//Gamma set

TFT9341\_SendData(0x01);//Gamma Curve (G2.2) (Кривая цветовой гаммы)

TFT9341\_SendCommand(0xE0);//Positive Gamma Correction

TFT9341\_SendData(0x0F);

TFT9341\_SendData(0x31);

TFT9341\_SendData(0x2B);

TFT9341\_SendData(0x0C);

TFT9341\_SendData(0x0E);

TFT9341\_SendData(0x08);

TFT9341\_SendData(0x4E);

TFT9341\_SendData(0xF1);

TFT9341\_SendData(0x37);

TFT9341\_SendData(0x07);

TFT9341\_SendData(0x10);

TFT9341\_SendData(0x03);

TFT9341\_SendData(0x0E);

TFT9341\_SendData(0x09);

TFT9341\_SendData(0x00);

TFT9341\_SendCommand(0xE1);//Negative Gamma Correction

TFT9341\_SendData(0x00);

TFT9341\_SendData(0x0E);

TFT9341\_SendData(0x14);

TFT9341\_SendData(0x03);

TFT9341\_SendData(0x11);

TFT9341\_SendData(0x07);

TFT9341\_SendData(0x31);

TFT9341\_SendData(0xC1);

TFT9341\_SendData(0x48);

TFT9341\_SendData(0x08);

TFT9341\_SendData(0x0F);

TFT9341\_SendData(0x0C);

TFT9341\_SendData(0x31);

TFT9341\_SendData(0x36);

TFT9341\_SendData(0x0F);

TFT9341\_SendCommand(0x11);//Выйдем из спящего режим

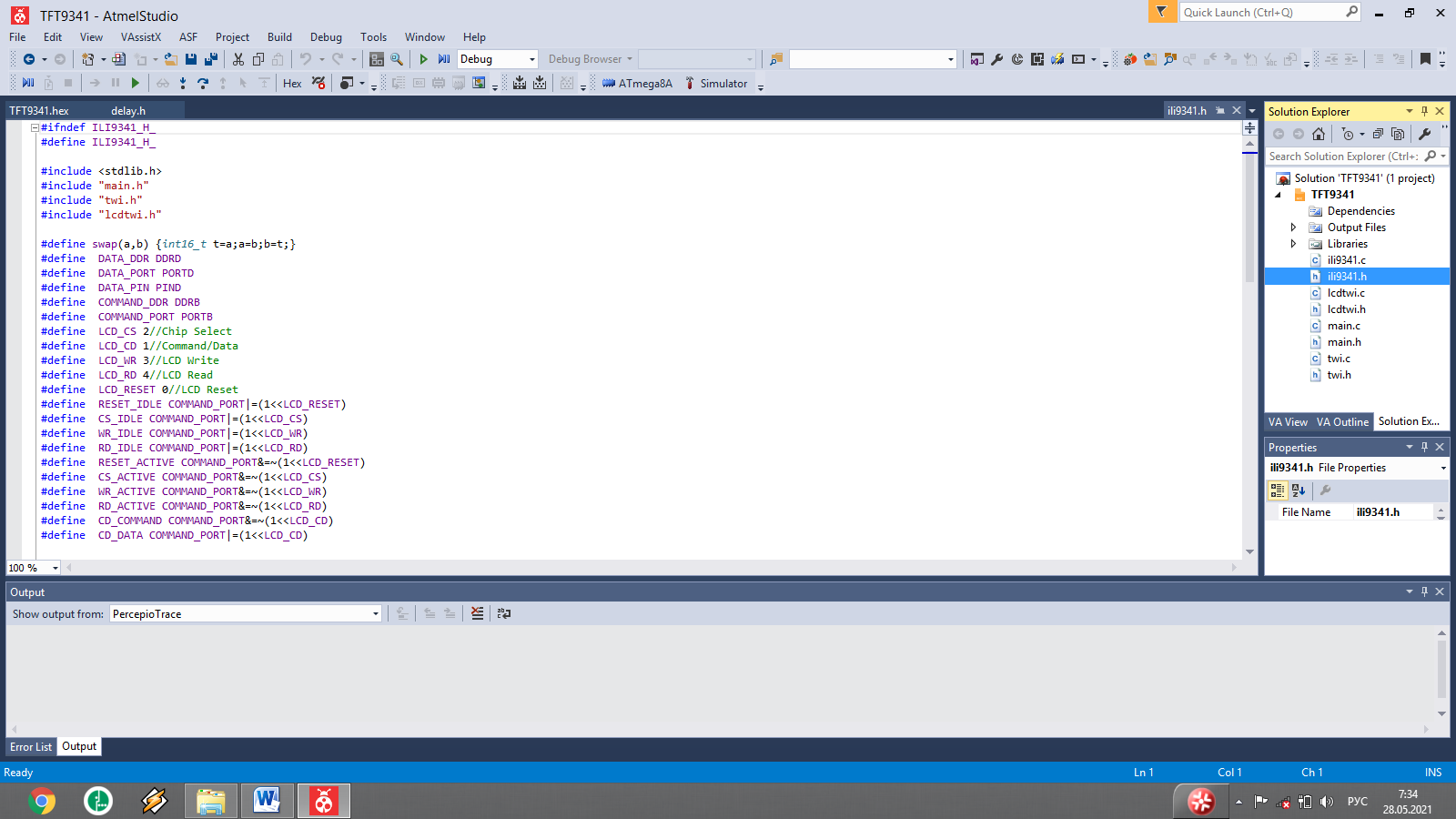
*\_delay\_ms*(150);

TFT9341\_SendCommand(0x29);//Включение дисплея

TFT9341\_SendData(0x2C);

*\_delay\_ms*(150);

}



#ifndef ILI9341\_H\_

#define ILI9341\_H\_

#include <stdlib.h>

#include "main.h"

#include "twi.h"

#include "lcdtwi.h"

#define swap(a,b) {*int16\_t* t=a;a=b;b=t;}

#define DATA\_DDR DDRD

#define DATA\_PORT PORTD

#define DATA\_PIN PIND

#define COMMAND\_DDR DDRB

#define COMMAND\_PORT PORTB

#define LCD\_CS 2//Chip Select

#define LCD\_CD 1//Command/Data

#define LCD\_WR 3//LCD Write

#define LCD\_RD 4//LCD Read

#define LCD\_RESET 0//LCD Reset

#define RESET\_IDLE COMMAND\_PORT|=(1<<LCD\_RESET)

#define CS\_IDLE COMMAND\_PORT|=(1<<LCD\_CS)

#define WR\_IDLE COMMAND\_PORT|=(1<<LCD\_WR)

#define RD\_IDLE COMMAND\_PORT|=(1<<LCD\_RD)

#define RESET\_ACTIVE COMMAND\_PORT&=~(1<<LCD\_RESET)

#define CS\_ACTIVE COMMAND\_PORT&=~(1<<LCD\_CS)

#define WR\_ACTIVE COMMAND\_PORT&=~(1<<LCD\_WR)

#define RD\_ACTIVE COMMAND\_PORT&=~(1<<LCD\_RD)

#define CD\_COMMAND COMMAND\_PORT&=~(1<<LCD\_CD)

#define CD\_DATA COMMAND\_PORT|=(1<<LCD\_CD)

#define BLACK 0x0000

#define BLUE 0x001F

#define RED 0x0F800

#define GREEN 0x07E0

#define CYAN 0x07FF

#define MAGENTA 0xF81F

#define YELLOW 0xFFE0

#define WHITE 0xFFFF

#define setReadDir() DATA\_DDR=0x00

#define setWriteDir() DATA\_DDR=0xFF

#define WR\_STROBE {WR\_ACTIVE;WR\_IDLE;}

void TFT9341\_ini(void);

void TFT9341\_FillScreen(unsigned int color);

void TFT9341\_FillRectangle(unsigned int color,unsigned int x1, unsigned int y1,

unsigned int x2, unsigned int y2);

unsigned int TFT9341\_RandColor(void);

void TFT9341\_DrawPixel(int x, int y, unsigned int color);

void TFT9341\_DrawLine(unsigned int color,unsigned int x1, unsigned int y1,

unsigned int x2, unsigned int y2);

void TFT9341\_DrawRect(unsigned int color,unsigned int x1, unsigned int y1,

unsigned int x2, unsigned int y2);

void TFT9341\_DrawCircle(unsigned int x0, unsigned int y0, int r, unsigned int color);

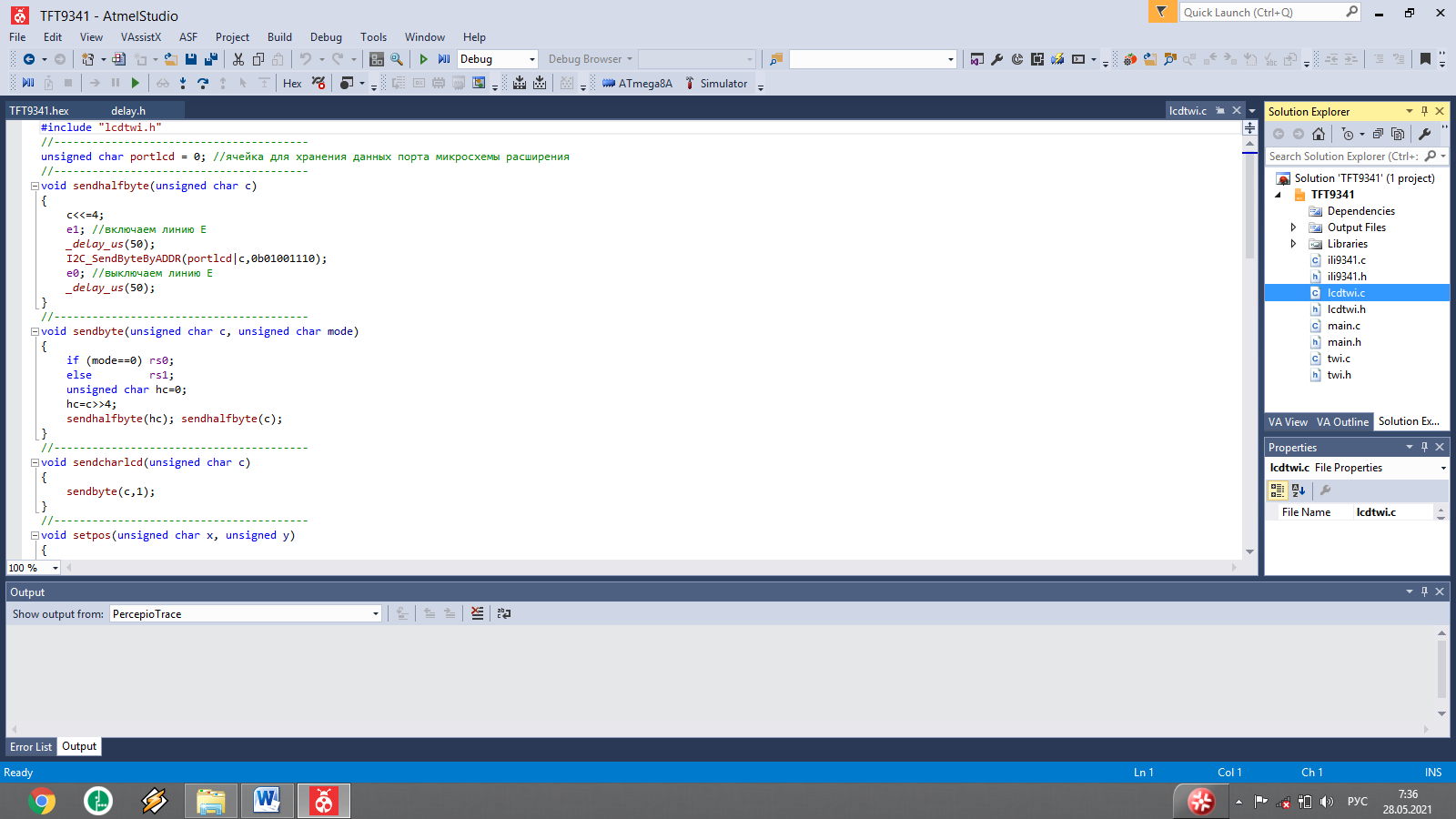
void TFT9341\_Draw\_Char(int x, int y, unsigned int color, unsigned int phone,

unsigned char charcode, unsigned char size);

void TFT9341\_String(unsigned int x, unsigned int y, unsigned int color, unsigned int phone,

char \*str, unsigned char size);

#endif /\* ILI9341\_H\_ \*/



#include "lcdtwi.h"

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

unsigned char portlcd = 0; //ячейка для хранения данных порта микросхемы расширения

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

void sendhalfbyte(unsigned char c)

{

c<<=4;

e1; //включаем линию Е

*\_delay\_us*(50);

I2C\_SendByteByADDR(portlcd|c,0b01001110);

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)

{

switch(y)

{

case 0:

sendbyte(x|0x80,0);

break;

case 1:

sendbyte((0x40+x)|0x80,0);

break;

case 2:

sendbyte((0x14+x)|0x80,0);

break;

case 3:

sendbyte((0x54+x)|0x80,0);

break;

}

}

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

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);

setled();//подсветка

setwrite();//запись

}

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

void clearlcd(void)

{

sendbyte(0b00000001, 0);

*\_delay\_us*(1500);

}

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

void str\_lcd (char str1[])

{

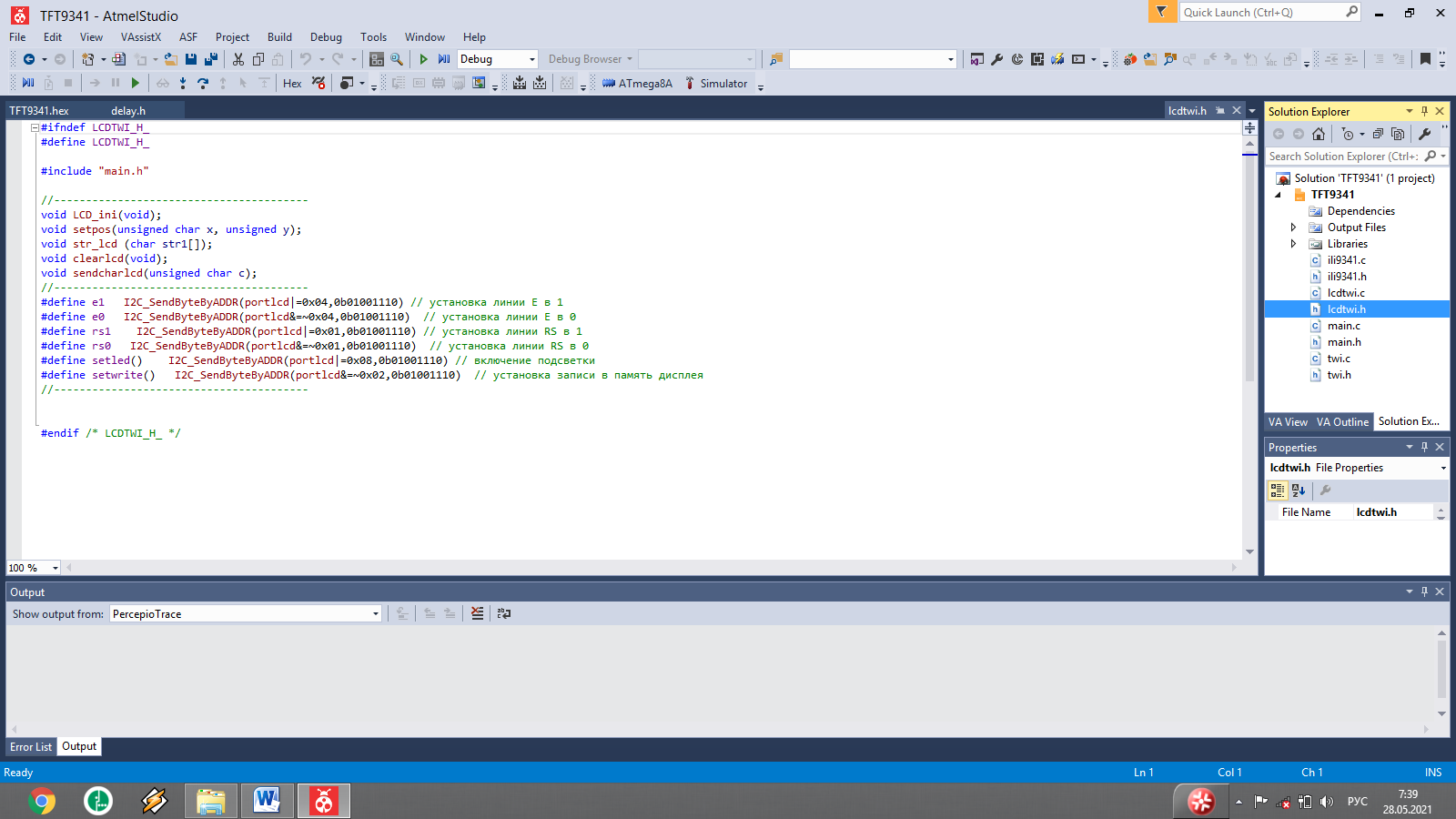
wchar\_t n;

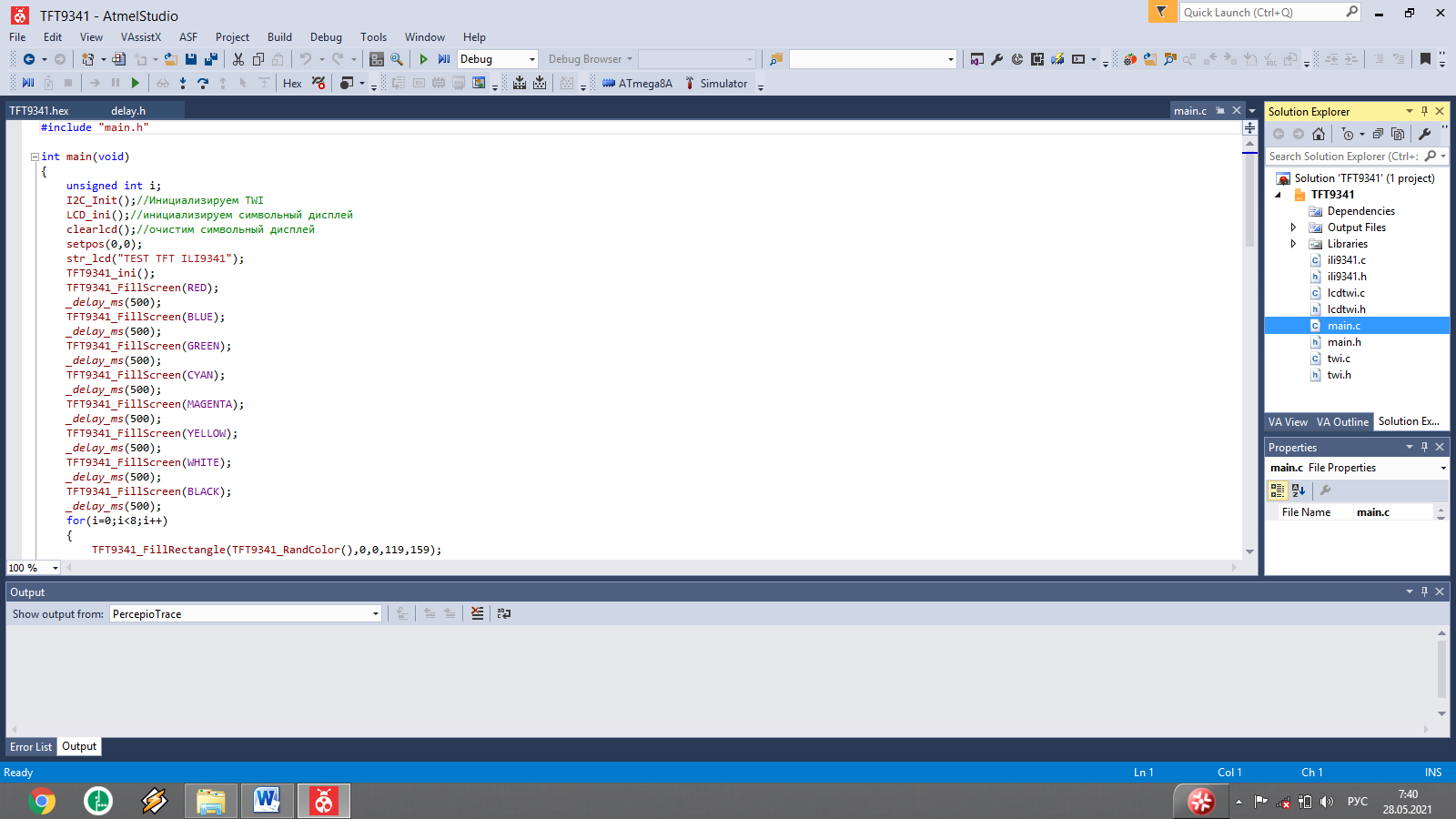
for(n=0;str1[n]!='\0';n++)

sendcharlcd(str1[n]);

}

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





#include "main.h"

int main(void)

{

unsigned int i;

I2C\_Init();//Инициализируем TWI

LCD\_ini();//инициализируем символьный дисплей

clearlcd();//очистим символьный дисплей

setpos(0,0);

str\_lcd("TEST TFT ILI9341");

TFT9341\_ini();

TFT9341\_FillScreen(RED);

*\_delay\_ms*(500);

TFT9341\_FillScreen(BLUE);

*\_delay\_ms*(500);

TFT9341\_FillScreen(GREEN);

*\_delay\_ms*(500);

TFT9341\_FillScreen(CYAN);

*\_delay\_ms*(500);

TFT9341\_FillScreen(MAGENTA);

*\_delay\_ms*(500);

TFT9341\_FillScreen(YELLOW);

*\_delay\_ms*(500);

TFT9341\_FillScreen(WHITE);

*\_delay\_ms*(500);

TFT9341\_FillScreen(BLACK);

*\_delay\_ms*(500);

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

{

TFT9341\_FillRectangle(TFT9341\_RandColor(),0,0,119,159);

*\_delay\_ms*(100);

TFT9341\_FillRectangle(TFT9341\_RandColor(),120,0,239,159);

*\_delay\_ms*(100);

TFT9341\_FillRectangle(TFT9341\_RandColor(),0,160,119,319);

*\_delay\_ms*(100);

TFT9341\_FillRectangle(TFT9341\_RandColor(),120,160,239,319);

*\_delay\_ms*(100);

}

TFT9341\_FillScreen(BLACK);

for(i=0;i<15000;i++)

{

TFT9341\_DrawPixel(*rand*()%240,*rand*()%320,TFT9341\_RandColor());

}

TFT9341\_FillScreen(BLACK);

for(i=0;i<240;i++)

{

TFT9341\_DrawLine(TFT9341\_RandColor(),i,0,i,319);

}

*\_delay\_ms*(500);

TFT9341\_FillScreen(BLACK);

for(i=0;i<1000;i++)

{

TFT9341\_DrawLine(TFT9341\_RandColor(),*rand*()%240,*rand*()%320,*rand*()%240,*rand*()%320);

}

*\_delay\_ms*(500);

TFT9341\_FillScreen(BLACK);

for(i=0;i<120;i++)

{

TFT9341\_DrawRect(TFT9341\_RandColor(),i,i,239-i,319-i);

}

*\_delay\_ms*(500);

TFT9341\_FillScreen(BLACK);

for(i=0;i<2000;i++)

{

TFT9341\_DrawCircle(*rand*()%200+20, *rand*()%280+20, 20, TFT9341\_RandColor());

}

*\_delay\_ms*(500);

TFT9341\_FillScreen(BLACK);

TFT9341\_Draw\_Char(10,10,RED,GREEN,0x21,2);

TFT9341\_Draw\_Char(26,10,RED,GREEN,0x22,2);

TFT9341\_Draw\_Char(42,10,RED,GREEN,0x23,2);

TFT9341\_Draw\_Char(58,10,RED,GREEN,0x24,2);

TFT9341\_Draw\_Char(74,10,RED,GREEN,0x25,2);

TFT9341\_Draw\_Char(10,26,RED,GREEN,0x21,2);

TFT9341\_Draw\_Char(26,26,RED,GREEN,0x22,2);

TFT9341\_Draw\_Char(42,26,RED,GREEN,0x23,2);

TFT9341\_Draw\_Char(58,26,RED,GREEN,0x24,2);

TFT9341\_Draw\_Char(74,26,RED,GREEN,0x25,2);

*\_delay\_ms*(1000);

TFT9341\_FillScreen(BLACK);

TFT9341\_String(1,1,RED,GREEN,"12345ABCDE",1);

TFT9341\_String(1,9,RED,GREEN,"EDCAB54321",1);

TFT9341\_String(10,17,RED,GREEN,"ABCDEabcde",1);

*\_delay\_ms*(1000);

TFT9341\_FillScreen(BLACK);

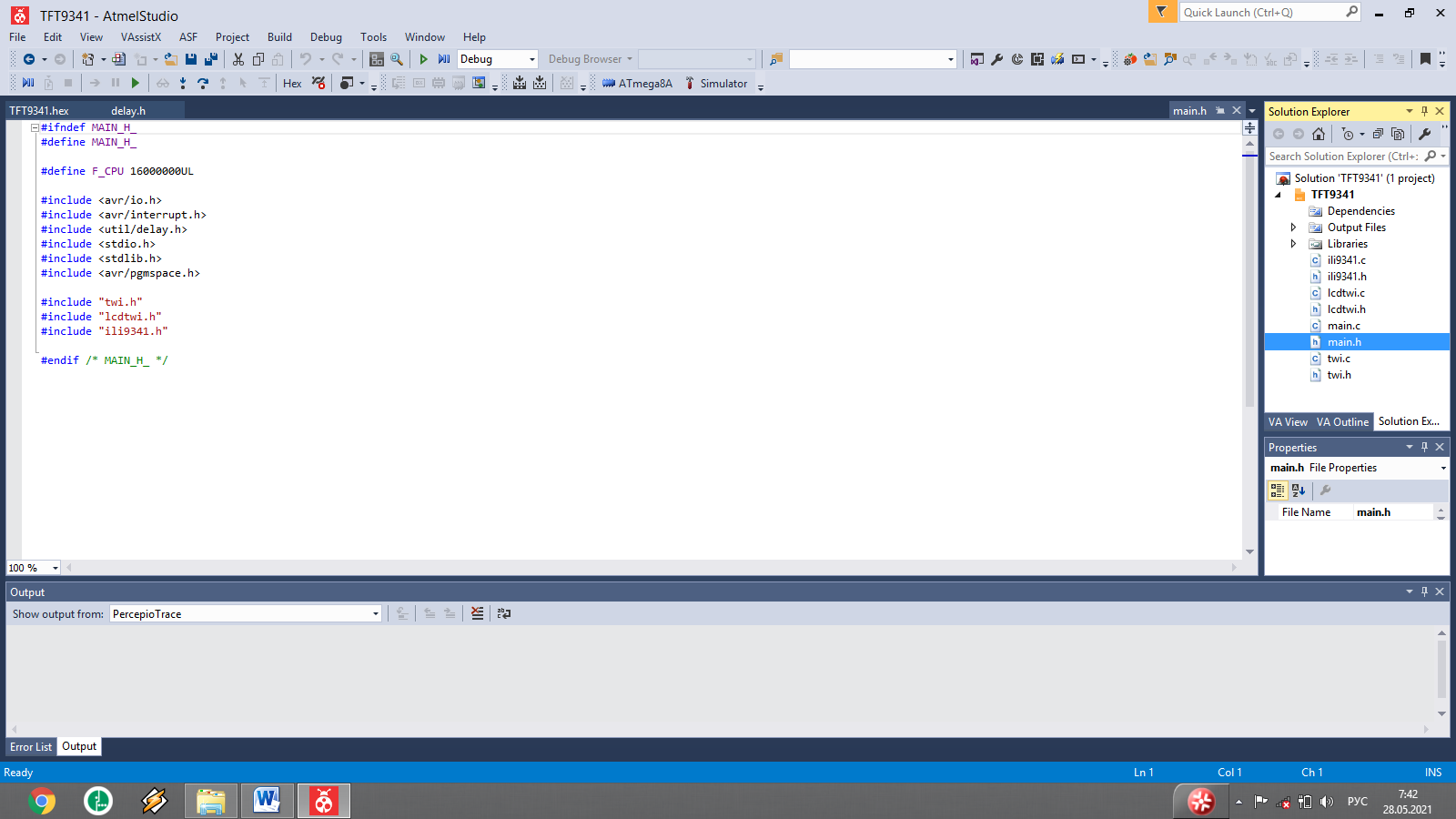
while (1)

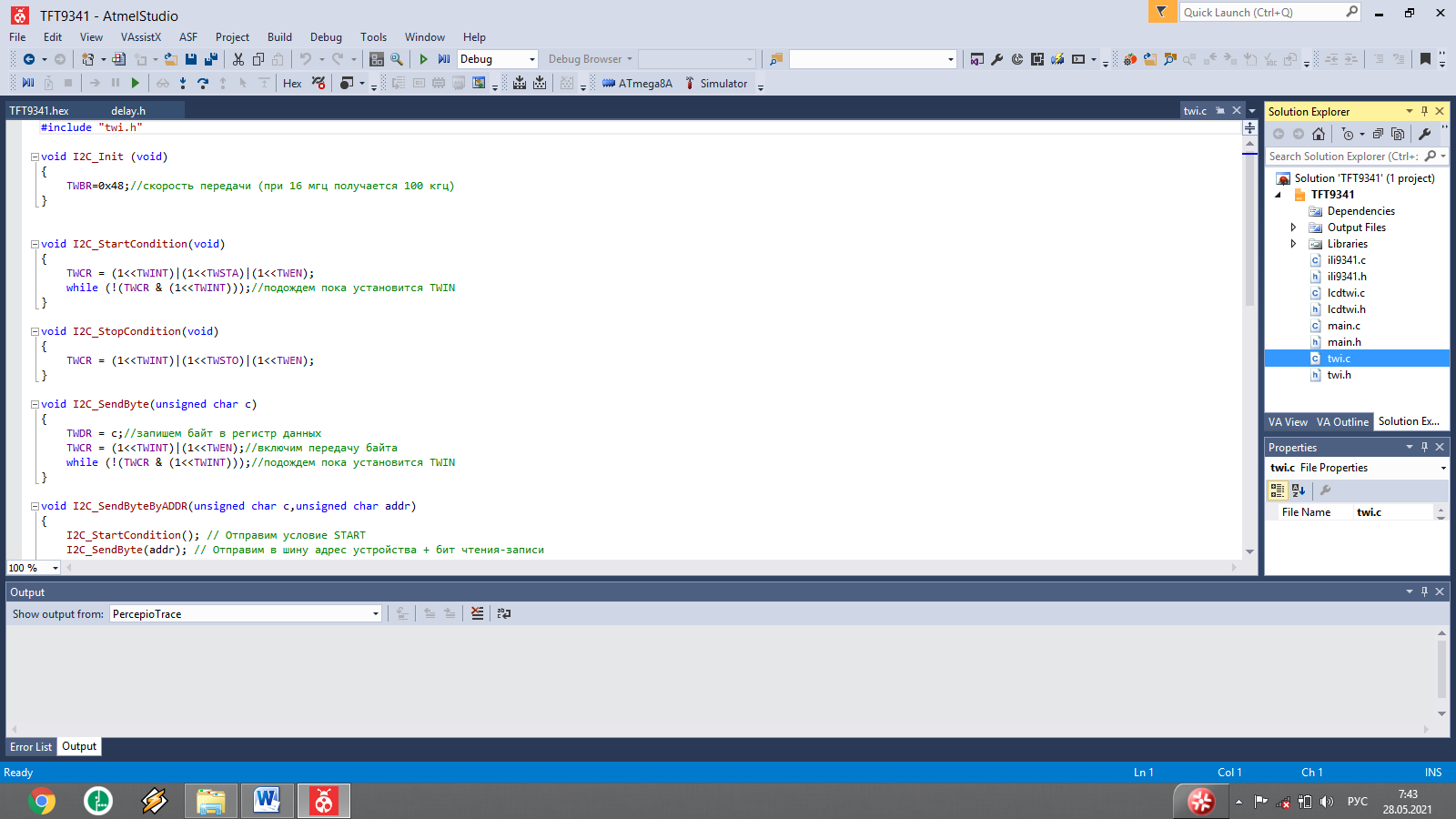
{

TFT9341\_Draw\_Char((*rand*()%15)\*16,(*rand*()%20)\*16,GREEN,BLACK,*rand*()%2+0x21,2);

}

}





#include "twi.h"

void I2C\_Init (void)

{

TWBR=0x48;//скорость передачи (при 16 мгц получается 100 кгц)

}

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)

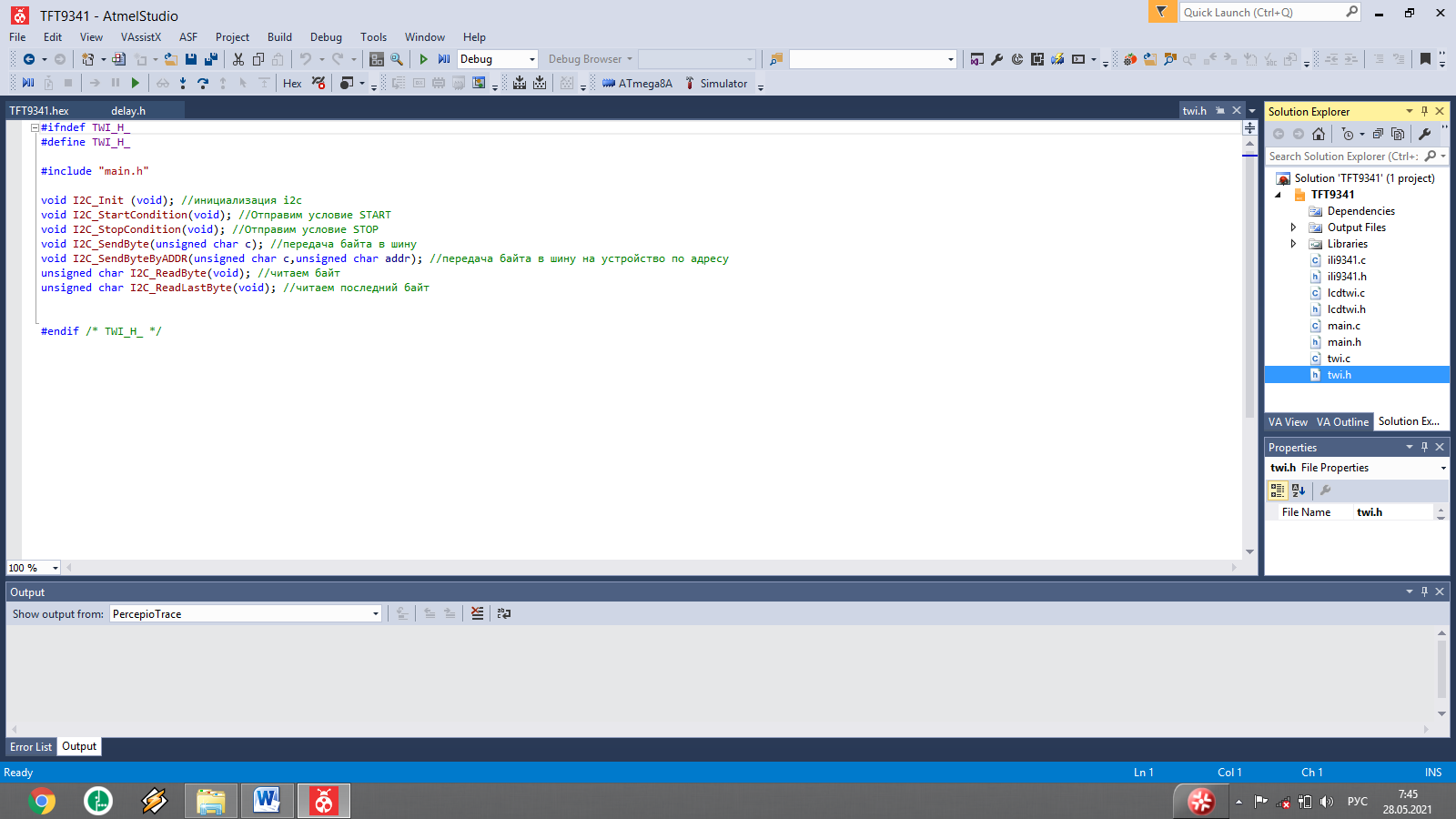
{

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

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

return TWDR;//читаем регистр данных

}



#ifndef TWI\_H\_

#define TWI\_H\_

#include "main.h"

void I2C\_Init (void); //инициализация i2c

void I2C\_StartCondition(void); //Отправим условие START

void I2C\_StopCondition(void); //Отправим условие STOP

void I2C\_SendByte(unsigned char c); //передача байта в шину

void I2C\_SendByteByADDR(unsigned char c,unsigned char addr); //передача байта в шину на устройство по адресу

unsigned char I2C\_ReadByte(void); //читаем байт

unsigned char I2C\_ReadLastByte(void); //читаем последний байт

#endif /\* TWI\_H\_ \*/