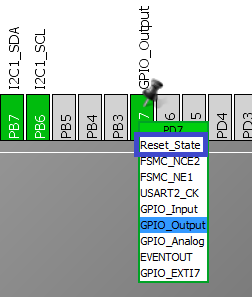
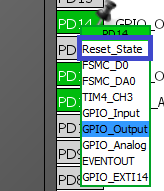
**Display TFT 240 × 320 8bit**

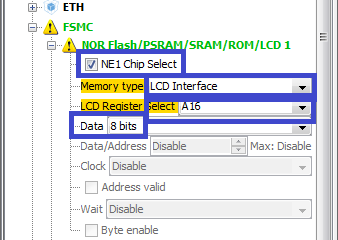
Today we will try to connect a TFT display with a resolution of 240 × 320 to the STM32 controller. The display module is implemented on the ILI9341 controller and connected via an 8-bit bus and we will connect it using FSMC technology. This is a technology that allows you to work with the display controller, as with conventional memory, without even worrying about which control leg and when to jerk.

For this display, I also have a lesson for the AVR controller. I advise you to view it as well. After that, you will become much more comfortable working with him.

We create the project from the project USB\_HOST\_MSC\_FATFS. Let's call it ILI9341. Run the project in the Cube, disable the output of PD7 and PD14, otherwise we will not enable the FSMC.

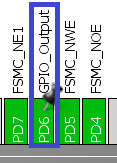
  

We turn on the FSMC as follows

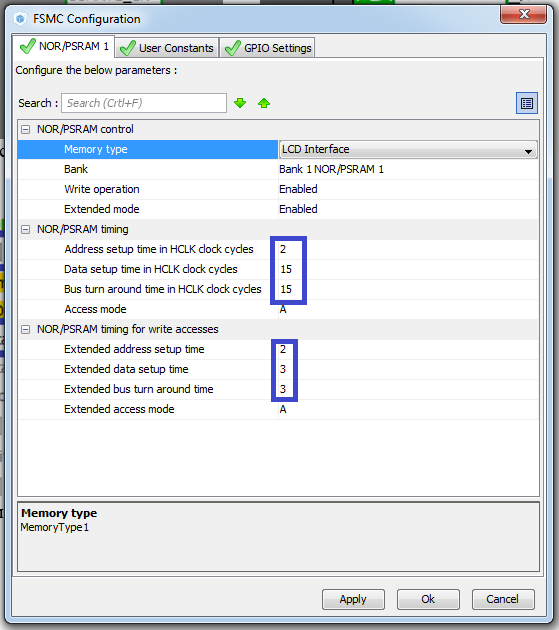


Now we see to which legs of the controller which display legs to connect

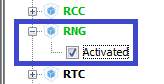
Also, the controller has a presser foot (LCD\_RST). To control this display foot, turn on the tab of the PD6 port to the output.



In Configuration, the FSMC will enable the following timings



Also include RNG for the normal formation of a random number



Generate the project, open it, connect lcd.c. We will collect the project and configure the programmer.

Create and add files ili9341.c and ili9341.h to the appropriate src and inc directory

Connect the file to main.h

#include "lcd.h"

**#include "ili9341.h"**

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

We also connect it in ili9341.c

**#include "ili9341.h"**

In the standard way, fill this file and connect the necessary libraries

#ifndef ILI9341\_H\_

#define ILI9341\_H\_

#include <stdlib.h>

#include "stm32f4xx\_hal.h"

#include "lcd.h"

#include "fatfs.h"

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

Declare the address variables for data and commands to the file ili9341.h

#include "fatfs.h"

**#define ADDR\_CMD \* (uint8\_t \*) 0x60000000**

**#define ADDR\_DATA \* (uint8\_t \*) 0x60010000**

We add here a few more macrosubstitutions

#define ADDR\_DATA \* (uint8\_t \*) 0x60010000

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

**#define RESET\_ACTIVE HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_6, GPIO\_PIN\_RESET);**

**#define RESET\_IDLE HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_6, GPIO\_PIN\_SET);**

**#define BLACK 0x0000**

**#define BLUE 0x001F**

**#define RED 0x0F800**

**#define GREEN 0x07E0**

**#define CYAN 0x07FF**

**#define MAGENTA 0xF81F**

**#define YELLOW 0xFFE0**

**#define WHITE 0xFFFF**

Add the function of initializing the display to file ili9341.c

#include "ili9341.h"

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

**void TFT9341\_ini (void)**

**{**

**}**

Do not forget to add the prototype to the header file.

Call this function in main.c

        LCD\_Clear ();

**TFT9341\_ini ();**

  / \* USER CODE END 2 \* /

In the FileReadWrite function we leave only this

/ \* USER CODE BEGIN 0 \* /

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

**void FileReadWrite (void)**

**{**

**if (f\_mount (& USBDISKFatFs, (TCHAR const \*) USBH\_Path, 0)! = FR\_OK)**

**{**

**Error\_Handler ();**

**}**

**else**

**{**

**}**

**}**

In file ili9341.c we add two functions for the delay

#include "ili9341.h"

**// ---------------------**

**void TFT9341\_Delay (uint32\_t dly)**

**{**

**uint32\_t i;**

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

**}**

**// ---------------------**

**\_\_STATIC\_INLINE void DelayMicro (\_\_ IO uint32\_t micros)**

**{**

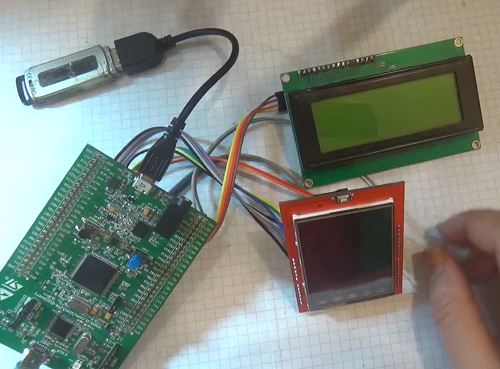
**micros \* = (SystemCoreClock / 1000000) / 5;**

**while (micros-);**

**}**

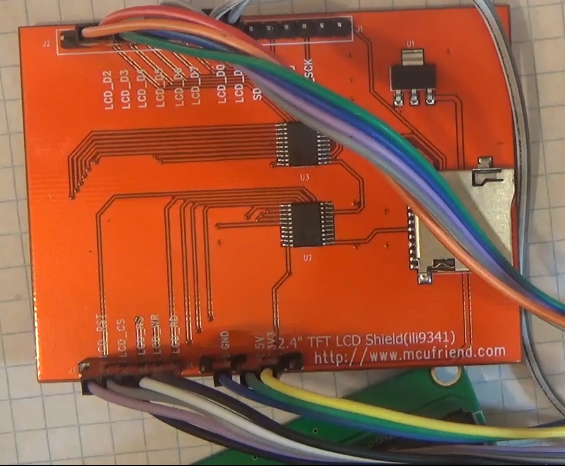
**// ---------------------**

Before writing other functions, let's see how the display is connected to the Discovery board (in general, I also advise you to watch the video tutorial, you'll see it all much better in it, for here basically the explanation of the code)

[](http://narodstream.ru/wp-content/uploads/2016/11/Image00-8.png)

As we can see, we still have a symbolic display via the adapter on the I2C bus for monitoring the read identifier.

Here we can also see the display on the back, to which legs it has wires connected



The first thing we'll write is a function for passing the command to the display

        while (micros-);

}

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

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

**{**

**ADDR\_CMD = cmd;**

**}**

**// ---------------------**

This function is so easy with FSMC technology

A similar function will be used to transfer data to the display controller memory

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

**{**

**ADDR\_DATA = dt;**

**TFT9341\_Delay (1);**

**}**

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

We also add a function for reading the register from the display controller (many things are peeked in datashit, in the video version attached to the lesson from the bottom, it's more obvious)

**uint32\_t TFT9341\_ReadReg (uint8\_t r)**

**{**

**uint32\_t id;**

**uint8\_t x;**

**TFT9341\_SendCommand (r);**

**DelayMicro (50);**

**x = ADDR\_DATA;**

**id = x;**

**id << = 8;**

**DelayMicro (1);**

**x = ADDR\_DATA;**

**id | = x;**

**id << = 8;**

**DelayMicro (1);**

**x = ADDR\_DATA;**

**id | = x;**

**id << = 8;**

**DelayMicro (1);**

**x = ADDR\_DATA;**

**id | = x;**

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

**{**

**id << = 8;**

**DelayMicro (5);**

**x = ADDR\_DATA;**

**id | = x;**

**}**

**DelayMicro (150); // stabilization time**

**return id;**

**}**

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

Add two variables in the file ili9341.c

#include "ili9341.h"

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

**uint16\_t X\_SIZE = 0;**

**uint16\_t Y\_SIZE = 0;**

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

Let's write the display orientation function

**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;**

**}**

**}**

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

Let's write a prototype for this function, we need it later.

Let's write the function of the display reset

**void TFT9341\_reset (void)**

**{**

**RESET\_ACTIVE;**

**HAL\_Delay (2);**

**RESET\_IDLE;**

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

**for (uint8\_t i = 0; i <3; i ++) TFT9341\_SendData (0xFF);**

**}**

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

Add one more variable

uint16\_t Y\_SIZE = 0;

**uint32\_t dtt = 0;**

Let's start writing the initialization of the display and try to read its identifier

void TFT9341\_ini (void)   
{   
**char str [10];   
    TFT9341\_reset ();   
    HAL\_Delay (1000);   
    dtt = TFT9341\_ReadReg (0xD3);   
    LCD\_Clear ();   
    LCD\_SetPos (0,0);   
    sprintf (str, "0x% 08lX", (unsigned long) dtt);   
    LCD\_String (str);**

We compile the code, we'll edit the controller and see the result of reading the identifier on the symbol display.



We continue to write the initialization of the display further, also from time to time looking in the datasheet to the controller (in principle comments near each setting in the code themselves will say):

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

**DelayMicro (1);**

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

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

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

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

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

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

**DelayMicro (1);**

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

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

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

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

**DelayMicro (1);**

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

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

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

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

**DelayMicro (1);**

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

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

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

**DelayMicro (1);**

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

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

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

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

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

**DelayMicro (1);**

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

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

**DelayMicro (1);**

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

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

**DelayMicro (1);**

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

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

**DelayMicro (1);**

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

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

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

**DelayMicro (1);**

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

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

**DelayMicro (1);**

**TFT9341\_SetRotation (0);**

**DelayMicro (1);**

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

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

**DelayMicro (1);**

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

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

**TFT9341\_SendData (0x18); // Frame rate 79 Hz**

**DelayMicro (1);**

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

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

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

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

**DelayMicro (1);**

**TFT9341\_SendCommand (0xF2); // Enable 3G (I do not yet know what this mode is)**

**TFT9341\_SendData (0x00); // do not enable**

**DelayMicro (1);**

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

**TFT9341\_SendData (0x01); // Gamma Curve (G2.2) (Curve of color scale)**

**DelayMicro (1);**

**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);**

**DelayMicro (1);**

**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);**

**DelayMicro (1);**

**TFT9341\_SendCommand (0x11); // Let's get out of sleep mode**

**HAL\_Delay (150);**

**TFT9341\_SendCommand (0x29); // Turning the display on**

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

**HAL\_Delay (150);**

Let's try now to paint the display in some color. For this we need three functions

**// ---------------------**

**void TFT9341\_Flood (uint16\_t color, uint32\_t len)**

**{**

**uint16\_t blocks;**

**uint8\_t i, hi = color >> 8, lo = color;**

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

**TFT9341\_SendData (hi);**

**DelayMicro (1);**

**TFT9341\_SendData (lo);**

**len-;**

**blocks = (uint16\_t) (len / 64); // 64 pixels / block**

**while (blocks-)**

**{**

**i = 16;**

**do**

**{**

**TFT9341\_SendData (hi);**

**TFT9341\_SendData (lo);**

**TFT9341\_SendData (hi);**

**TFT9341\_SendData (lo);**

**TFT9341\_SendData (hi);**

**TFT9341\_SendData (lo);**

**TFT9341\_SendData (hi);**

**TFT9341\_SendData (lo);**

**} while (-i);**

**}**

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

**for (i = (uint8\_t) len & 63; i-;)**

**{**

**TFT9341\_SendData (hi);**

**TFT9341\_SendData (lo);**

**}**

**}**

**// ---------------------**

**void TFT9341\_SetAddrWindow (uint16\_t x1, uint16\_t y1, uint16\_t x2, uint16\_t y2)**

**{**

**TFT9341\_SendCommand (0x2A); // Column Addres Set**

**TFT9341\_SendData (x1 >> 8);**

**TFT9341\_SendData (x1 & 0xFF);**

**TFT9341\_SendData (x2 >> 8);**

**TFT9341\_SendData (x2 & 0xFF);**

**DelayMicro (1);**

**TFT9341\_SendCommand (0x2B); // Page Addres Set**

**TFT9341\_SendData (y1 >> 8);**

**TFT9341\_SendData (y1 & 0xFF);**

**TFT9341\_SendData (y2 >> 8);**

**TFT9341\_SendData (y2 & 0xFF);**

**DelayMicro (1);**

**}**

**// ---------------------**

**void TFT9341\_FillScreen (uint16\_t color)**

**{**

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

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

**}**

**// ---------------------**

For the TFT9341\_FillScreen function, we'll write a prototype and call it in main ()

        TFT9341\_ini ();

**TFT9341\_FillScreen (RED);**

  / \* USER CODE END 2 \* /

We will collect, we will sew, we will check.

At us the display glows red.



Add a global variable to the file ili9341.c

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

**extern RNG\_HandleTypeDef hrng;**

uint16\_t X\_SIZE = 0;

Let's write a function for setting a random color for the display

**uint16\_t TFT9341\_RandColor (void)**

**{**

**return HAL\_RNG\_GetRandomNumber (& hrng) & 0x0000FFFF;**

**}**

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

T

Also create a prototype for this function.

Let's write a small test in the main () function. To do this, first add a variable for the loop counter (at the same time remove the variable str)

  / \* USER CODE BEGIN 1 \* /

**uint16\_t i;**

  / \* USER CODE END 1 \* /

Now the test itself

        TFT9341\_ini ();

**for (i = 0; i <20; i ++)**

**{**

**TFT9341\_FillScreen (TFT9341\_RandColor ());**

**HAL\_Delay (150);**

**}**

**TFT9341\_FillScreen (BLACK);**

**HAL\_Delay (500);**

We will collect and we will sew the project. We should see this on the screen of our TFT display, coloring the display in different colors every 0.15 seconds

Now try to write a function for painting an arbitrary rectangular area of ​​the display, rather than the whole screen

**void TFT9341\_FillRectangle (uint16\_t color, uint16\_t x1, uint16\_t y1,**

**uint16\_t x2, uint16\_t y2)**

**{**

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

**TFT9341\_Flood (color, (uint16\_t) (x2-x1 + 1) \* (uint16\_t) (y2-y1 + 1));**

**}**

**// ---------------------**

Let's make a prototype of the function and write one more light test in main ()

        HAL\_Delay (500);

**for (i = 0; i <20; i ++)**

**{**

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

**HAL\_Delay (100);**

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

**HAL\_Delay (100);**

**TFT9341\_FillRectangle (TFT9341\_RandColor (), 0.160.119.319);**

**HAL\_Delay (100);**

**TFT9341\_FillRectangle (TFT9341\_RandColor (), 120.160.239.319);**

**HAL\_Delay (100);**

**}**

**HAL\_Delay (500);**

**TFT9341\_FillScreen (BLACK);**

We will collect and we will sew the project. If everything is correct, then this picture should look like this



In the [**next part,**](http://narodstream.ru/stm-urok-37-displej-tft-240x320-8bit-chast-4/) we will continue to write other tests and display functions.

Therefore, we will continue to work with it and write in the file ili9341.c the function for outputting a point of a certain color on the screen

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

**void TFT9341\_DrawPixel (int x, int y, uint16\_t color)**

**{**

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

**TFT9341\_SetAddrWindow (x, y, x, y);**

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

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

**TFT9341\_SendData (color & 0xFF);**

**}**

We will also add a prototype to it, well, and in honor of this, of course, the test in main (), only first we copy the delay function to main.c

/ \* USER CODE BEGIN 0 \* /

**\_\_STATIC\_INLINE void DelayMicro (\_\_ IO uint32\_t micros)**

**{**

**micros \* = (SystemCoreClock / 1000000) / 5;**

**while (micros-);**

**}**

**// ---------------------**

void FileReadWrite (void)

Well, actually the test.

        TFT9341\_FillScreen (BLACK);

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

**{**

**TFT9341\_DrawPixel (HAL\_RNG\_GetRandomNumber (& hrng)% 240, HAL\_RNG\_GetRandomNumber (& hrng)% 320, TFT9341\_RandColor ());**

**DelayMicro (100);**

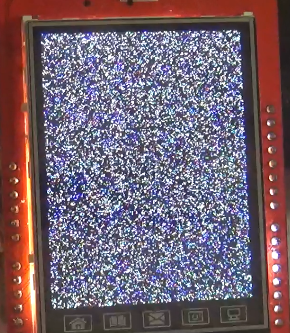
**}**

**HAL\_Delay (500);**

**TFT9341\_FillScreen (BLACK);**

We will collect and we will sew the project.

Here's how our test on the display screen looks



Write the function of outputting a straight line to the screen

**void TFT9341\_DrawLine (uint16\_t color, uint16\_t x1, uint16\_t y1,**

**uint16\_t x2, uint16\_t 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;**

**}**

**}**

**}**

**// ---------------------**

Let's write the prototype and test in main ()

First, add one more variable to main ()

  / \* USER CODE BEGIN 1 \* /

        uint16\_t i, **j**;

  / \* USER CODE END 1 \* /

And actually the test itself

        TFT9341\_FillScreen (BLACK);

**for (j = 0; j <20; j ++)**

**{**

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

**{**

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

**}**

**HAL\_Delay (100);**

**}**

**HAL\_Delay (500);**

**TFT9341\_FillScreen (BLACK);**

This is how the test looks after the build and firmware



We will write one more test, for this we will not need new functions

        TFT9341\_FillScreen (BLACK);

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

**{**

**TFT9341\_DrawLine (TFT9341\_RandColor (), HAL\_RNG\_GetRandomNumber (& hrng)% 240,**

**HAL\_RNG\_GetRandomNumber (& hrng)% 320, HAL\_RNG\_GetRandomNumber (& hrng)% 240,**

**HAL\_RNG\_GetRandomNumber (& hrng)% 320);**

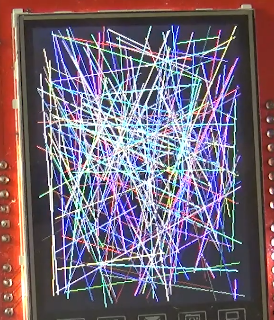
**HAL\_Delay (1);**

**}**

**HAL\_Delay (1000);**

**TFT9341\_FillScreen (BLACK);**

We will collect the code, we will impose the controller and we will receive here such result



Today we will also continue this interesting business.

We add the function of drawing a rectangle, but not already painted

**void TFT9341\_DrawRect (uint16\_t color, uint16\_t x1, uint16\_t y1,**

**uint16\_t x2, uint16\_t 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);**

**}**

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

We will write the prototype and, accordingly, the test in main ()

        TFT9341\_FillScreen (BLACK);

**for (j = 0; j <20; j ++)**

**{**

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

**{**

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

**}**

**HAL\_Delay (100);**

**if (j <19) TFT9341\_FillScreen (BLACK);**

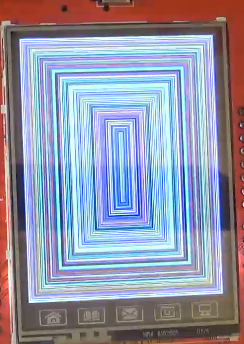
**}**

**HAL\_Delay (1000);**

**TFT9341\_FillScreen (BLACK);**

Before assembling the project and flashing the controller, we will comment out the previous tests so that we do not wait and do not waste time.

We will collect the project, we will impose the controller and we will look at the result



Now write the drawing function on the display of a circle with a certain radius

**void TFT9341\_DrawCircle (uint16\_t x0, uint16\_t y0, int r, uint16\_t 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);**

**}**

**}**

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

Let's create a prototype of this function, and, accordingly, the test in main ()

        TFT9341\_FillScreen (BLACK);

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

**{**

**TFT9341\_DrawCircle (HAL\_RNG\_GetRandomNumber (& hrng)% 200 + 20,**

**HAL\_RNG\_GetRandomNumber (& hrng)% 280 + 20, 20,**

**TFT9341\_RandColor ());**

**}**

**HAL\_Delay (1000);**

        TFT9341\_FillScreen (BLACK);

Let's see the result of this test, having pre-assembled the project and flashing the controller



Now let's try to work with symbols. In order not to occupy a lot of operational and flash memory of the controller, we connect a flesh-drive to it via USB OTG FS, to which we copy the previously created files of five fonts of different heights. These fonts are given in the Keil example folder, but they are given in text arrays, which I converted to binary files, by the way also with the help of the stm32 controller and the FATFS library (the files will be attached at the end of the part).

We connect USB Flash to Discovery via OTG-cable in the same way as we did in [**lesson 36**](http://narodstream.ru/stm-urok-36-usb-host-mass-storage-class-chast-1/) , when we studied the USB Mass Storage class.

Let's write some more variables in file ili9341.c

uint32\_t dtt = 0;

**uint32\_t TextColor;**

**uint32\_t BackColor;**

**uint8\_t \* pchar, \* str2;**

**extern FIL MyFile; / \* File object \* /**

**extern void Error\_Handler (void);**

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

void TFT9341\_Delay (uint32\_t dly)

Now add two structures and a few more variables.

extern void Error\_Handler (void);

**typedef struct \_tFont**

**{**

**uint16\_t Width;**

**uint16\_t Height;**

**} sFONT;**

**typedef struct**

**{**

**uint32\_t TextColor;**

**uint32\_t BackColor;**

**sFONT \* pFont;**

**} LCD\_DrawPropTypeDef;**

**sFONT Font24;**

**sFONT Font20;**

**sFONT Font16;**

**sFONT Font12;**

**sFONT Font8;**

**LCD\_DrawPropTypeDef lcdprop;**

Let's write the function of installing the default fonts

**void TFT9341\_FontsIni (void)**

**{**

**Font8.Height = 8;**

**Font8.Width = 5;**

**Font12.Height = 12;**

**Font12.Width = 7;**

**Font16.Height = 16;**

**Font16.Width = 11;**

**Font20.Height = 20;**

**Font20.Width = 14;**

**Font24.Height = 24;**

**Font24.Width = 17;**

**lcdprop.BackColor = BLACK;**

**lcdprop.TextColor = GREEN;**

**lcdprop.pFont = & Font16;**

**}**

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

Call it in the initialization at the very end of the function

        HAL\_Delay (150);

        TFT9341\_FontsIni ();

}

**Lesson 37**

**Part 6**

# Display TFT 240 × 320 8bit

In the [**last part of**](http://narodstream.ru/stm-urok-37-displej-tft-240x320-8bit-chast-5/) this lesson, we finished writing tests to output primitives to the display screen, and also started working with different fonts, writing several structures and variables, and adding a function to initialize the fonts at program startup

Today we will also continue working with fonts.

Let's write the symbol output function on the display screen (you can hear all the explanations of the code while watching the video version of the lesson attached below)

**void TFT9341\_DrawChar (uint16\_t x, uint16\_t y, uint8\_t s)**

**{**

**FRESULT res; / \* FatFs function common result code \* /**

**uint32\_t bytesread;**

**uint8\_t charbuf [100];**

**uint32\_t i = 0, j = 0;**

**uint16\_t height, width;**

**width = lcdprop.pFont-> Width;**

**height = lcdprop.pFont-> Height;**

**uint16\_t fontsize = 0; // character size in bytes in the table**

**uint32\_t line = 0; // horizontal line of pixels of the font**

**uint8\_t offset1; // difference between real width and width in bytes**

**uint16\_t offsetfile = 0; // Offset the character in the font file**

**if ((x + lcdprop.pFont-> Width)> = X\_SIZE) return;**

**switch (lcdprop.pFont-> Height)**

**{**

**case 8:**

**offset1 = 3;**

**fontsize = 8; // character size in bytes**

**HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_SET);**

**offsetfile = (s- '') \* fontsize;**

**f\_lseek (& MyFile, offsetfile);**

**res = f\_read (& MyFile, charbuf, 8, (void \*) & bytesread);**

**if ((bytesread == 0) || (res! = FR\_OK))**

**{**

**Error\_Handler ();**

**}**

**break;**

**case 12:**

**offset1 = 1;**

**fontsize = 12; // character size in bytes**

**HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_SET);**

**offsetfile = (s- '') \* fontsize;**

**f\_lseek (& MyFile, offsetfile);**

**res = f\_read (& MyFile, charbuf, 12, (void \*) & bytesread);**

**if ((bytesread == 0) || (res! = FR\_OK))**

**{**

**Error\_Handler ();**

**}**

**break;**

**case 16:**

**offset1 = 5;**

**fontsize = 32; // character size in bytes**

**HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_SET);**

**offsetfile = (s- '') \* fontsize;**

**f\_lseek (& MyFile, offsetfile);**

**res = f\_read (& MyFile, charbuf, 32, (void \*) & bytesread);**

**if ((bytesread == 0) || (res! = FR\_OK))**

**{**

**Error\_Handler ();**

**}**

**break;**

**case 20:**

**offset1 = 2;**

**fontsize = 40; // character size in bytes**

**HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_SET);**

**offsetfile = (s- '') \* fontsize;**

**f\_lseek (& MyFile, offsetfile);**

**res = f\_read (& MyFile, charbuf, 40, (void \*) & bytesread);**

**if ((bytesread == 0) || (res! = FR\_OK))**

**{**

**Error\_Handler ();**

**}**

**break;**

**case 24:**

**offset1 = 7;**

**fontsize = 72; // character size in bytes**

**HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_SET);**

**offsetfile = (s- '') \* fontsize;**

**f\_lseek (& MyFile, offsetfile);**

**res = f\_read (& MyFile, charbuf, 72, (void \*) & bytesread);**

**if ((bytesread == 0) || (res! = FR\_OK))**

**{**

**Error\_Handler ();**

**}**

**break;**

**}**

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

**{**

**if ((height == 8) || (height == 12)) line = charbuf [i];**

**else if ((height == 16) || (height == 20)) line = (charbuf [i \* 2] << 8) | (charbuf [i \* 2 + 1]);**

**else line = (charbuf [i \* 3] << 16) | (charbuf [i \* 3 + 1] << 8) | (charbuf [i \* 3 + 2]);**

**// line = 0xFFFF;**

**for (j = 0; j <width; j ++)**

**{**

**if (line & (1 << (width-j + offset1-1)))**

**{**

**TFT9341\_DrawPixel (x + j, y, lcdprop.TextColor);**

**}**

**else**

**{**

**TFT9341\_DrawPixel (x + j, y, lcdprop.BackColor);**

**}**

**}**

**y ++;**

**}**

**}**

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

**Lesson 37**

**Part 7**

# Display TFT 240 × 320 8bit

In the [**last part of**](http://narodstream.ru/stm-urok-37-displej-tft-240x320-8bit-chast-6/) this lesson we wrote a very complicated function for working with fonts, since we will download the fonts not from arrays, but from external USB Flesh, which we connected to the motherboard with the OTG cable.

Well, since the function is so complicated, then there were errors. Let's correct them. Their whole 8. In general, the whole process is most interesting to watch in the video tutorial, because the text of the function posted in the last lesson was written without errors.

Work on bugs:

1.

typedef struct \_tFont

{

        uint16\_t Width;

        uint16\_t ~~Heigth~~**Height**;

} sFONT;

2.

void TFT9341\_FontsIni (void)

{

        Font8. ~~Heigth~~**Height**= 8;

        Font8.Width = 5;

        Font12. ~~Heigth~~**Height**= 12;

        Font12.Width = 7;

        Font16. ~~Heigth~~**Height**= 16;

        Font16.Width = 11;

        Font20. ~~Heigth~~**Height**= 20;

        Font20.Width = 14;

        Font24. ~~Heigth~~**Height**= 24;

        Font24.Width = 17;

3.

        if ( **(x +**width **)**> = X\_SIZE) return;

        switch (height)

4.

        height = lcdprop.pFont-> ~~Heigth~~**Height**;

        uint16\_t fontsize = 0; // character size in bytes in the table

5.

~~uint16\_t~~**uint32\_t** line = 0; // horizontal line of pixels of the font

6.

                else if ((height == 16) || (height == 20)) line = (charbuf [i \* 2] << 8) | (charbuf [i \* 2 + 1]);

                else line = (charbuf [i \* 3] << 16) ~~|~~| (charbuf [i \* 3 + 1] << 8) | (charbuf [i \* 3 + 2]); (instead of two || one |)

7.

                                TFT9341\_DrawPixel (x + j, y, lcdprop.TextColor);

                        }

                        else

~~if (line & (1 << (width-j + offset1-1)))~~

                        {

8.

                                TFT9341\_DrawPixel (x + j, y, lcdprop.BackColor);

                        }

~~y ++;~~

                }

**y ++;**

        }

Let's write the prototype of this function in the header file

Let's write three more functions for setting the font size, font color and background color of the symbol

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

**void TFT9341\_SetFont (uint8\_t f)**

**{**

**switch (f)**

**{**

**case 8:**

**f\_close (& MyFile);**

**if (f\_open (& MyFile, "font8.bin", FA\_READ)! = FR\_OK)**

**{**

**Error\_Handler ();**

**}**

**else**

**{**

**lcdprop.pFont = & Font8;**

**}**

**break;**

**case 12:**

**f\_close (& MyFile);**

**if (f\_open (& MyFile, "font12.bin", FA\_READ)! = FR\_OK)**

**{**

**Error\_Handler ();**

**}**

**else**

**{**

**lcdprop.pFont = & Font12;**

**}**

**break;**

**case 16:**

**f\_close (& MyFile);**

**if (f\_open (& MyFile, "font16.bin", FA\_READ)! = FR\_OK)**

**{**

**Error\_Handler ();**

**}**

**else**

**{**

**lcdprop.pFont = & Font16;**

**}**

**break;**

**case 20:**

**f\_close (& MyFile);**

**if (f\_open (& MyFile, "font20.bin", FA\_READ)! = FR\_OK)**

**{**

**Error\_Handler ();**

**}**

**else**

**{**

**lcdprop.pFont = & Font20;**

**}**

**break;**

**case 24:**

**f\_close (& MyFile);**

**if (f\_open (& MyFile, "font24.bin", FA\_READ)! = FR\_OK)**

**{**

**Error\_Handler ();**

**}**

**else**

**{**

**lcdprop.pFont = & Font24;**

**}**

**break;**

**}**

**}**

**// ---------------------**

**void TFT9341\_SetTextColor (uint16\_t color)**

**{**

**lcdprop.TextColor = color;**

**}**

**// ---------------------**

**void TFT9341\_SetBackColor (uint16\_t color)**

**{**

**lcdprop.BackColor = color;**

**}**

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

Add to them all prototypes

Let's write a test for displaying symbols on the screen in the body of the **FileReadWrite**function

Previous tests while commenting

        else

        {

**TFT9341\_FillScreen (BLACK);**

**while (1)**

**{**

**TFT9341\_FillScreen (BLACK);**

**TFT9341\_SetTextColor (YELLOW);**

**TFT9341\_SetBackColor (BLUE);**

**TFT9341\_SetFont (24);**

**TFT9341\_DrawChar (10, 10, (uint8\_t) 'S');**

**TFT9341\_DrawChar (27, 10, (uint8\_t) 't');**

**TFT9341\_DrawChar (44, 10, (uint8\_t) 'm');**

**TFT9341\_DrawChar (61, 10, (uint8\_t) '3');**

**TFT9341\_DrawChar (78, 10, (uint8\_t) '2');**

**TFT9341\_SetTextColor (GREEN);**

**TFT9341\_SetBackColor (RED);**

**TFT9341\_SetFont (20);**

**TFT9341\_DrawChar (10, 34, (uint8\_t) 'S');**

**TFT9341\_DrawChar (24, 34, (uint8\_t) 't');**

**TFT9341\_DrawChar (38, 34, (uint8\_t) 'm');**

**TFT9341\_DrawChar (52, 34, (uint8\_t) '3');**

**TFT9341\_DrawChar (66, 34, (uint8\_t) '2');**

**TFT9341\_SetTextColor (BLUE);**

**TFT9341\_SetBackColor (YELLOW);**

**TFT9341\_SetFont (16);**

**TFT9341\_DrawChar (10, 54, (uint8\_t) 'S');**

**TFT9341\_DrawChar (21, 54, (uint8\_t) 't');**

**TFT9341\_DrawChar (32, 54, (uint8\_t) 'm');**

**TFT9341\_DrawChar (43, 54, (uint8\_t) '3');**

**TFT9341\_DrawChar (54, 54, (uint8\_t) '2');**

**TFT9341\_SetFont (12);**

**TFT9341\_SetTextColor (CYAN);**

**TFT9341\_SetBackColor (BLACK);**

**TFT9341\_DrawChar (10, 70, (uint8\_t) 'S');**

**TFT9341\_DrawChar (17, 70, (uint8\_t) 't');**

**TFT9341\_DrawChar (24, 70, (uint8\_t) 'm');**

**TFT9341\_DrawChar (31, 70, (uint8\_t) '3');**

**TFT9341\_DrawChar (38, 70, (uint8\_t) '2');**

**TFT9341\_SetFont (8);**

**TFT9341\_SetTextColor (RED);**

**TFT9341\_SetBackColor (GREEN);**

**TFT9341\_DrawChar (10, 82, (uint8\_t) 'S');**

**TFT9341\_DrawChar (15, 82, (uint8\_t) 't');**

**TFT9341\_DrawChar (20, 82, (uint8\_t) 'm');**

**TFT9341\_DrawChar (25, 82, (uint8\_t) '3');**

**TFT9341\_DrawChar (30, 82, (uint8\_t) '2');**

**HAL\_Delay (5000);**

                }

        }

We will collect, we will sew, we will see



Write a function to output a string of characters

**void TFT9341\_String (uint16\_t x, uint16\_t y, char \* str)**

**{**

**while (\* str)**

**{**

**TFT9341\_DrawChar (x, y, str [0]);**

**x + = lcdprop.pFont-> Width;**

**(void) \* str ++;**

**}**

**}**

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

Let's write a prototype and try to raise the lines in main.c in the FileReadWrite function

Let's create a variable

void FileReadWrite (void)

{

**uint8\_t i = 0;**

        if (f\_mount (& USBDISKFatFs, (TCHAR const \*) USBH\_Path, 0)! = FR\_OK)

Previous code for the output of individual characters is commented out

Here is the actual test

// HAL\_Delay (10000);

                        TFT9341\_SetTextColor (YELLOW);

                        TFT9341\_SetBackColor (BLUE);

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

                        {

                                TFT9341\_SetFont (24);

                                TFT9341\_FillScreen (BLACK);

                                TFT9341\_String (1, 100, "ABCDEF12345678");

                                TFT9341\_SetFont (20);

                                TFT9341\_String (1, 124, "ABCDEFGHI12345678");

                                TFT9341\_SetFont (16);

                                TFT9341\_String (1, 144, "ABCDEFGHIKL 123456789");

                                TFT9341\_SetFont (12);

                                TFT9341\_String (1, 160, "ABCDEFGHIKLMNOPQRSTUVWXY 123456789");

                                TFT9341\_SetFont (8);

                                TFT9341\_String (1, 172, "ABCDEFGHIKLMNOPQRSTUVWXYZ 123456789 ABCDEFGHIKL");

                                HAL\_Delay (2000);

                                TFT9341\_SetRotation (i% 4);

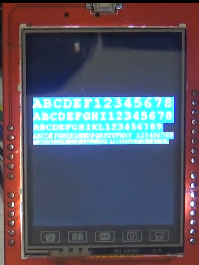
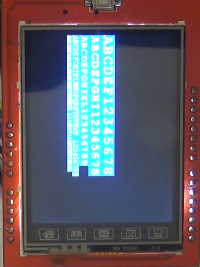
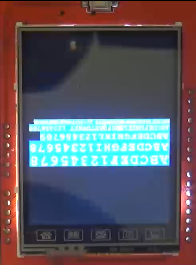
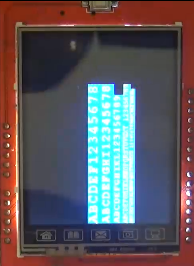
                        }

# Display TFT 240 × 320 8bit

In the [**last part of**](http://narodstream.ru/stm-urok-37-displej-tft-240x320-8bit-chast-7/) this lesson we wrote a number of functions for working with fonts. Also we did not finish the function of outputting lines to the screen in the video version, in the text it is fully laid out.

Therefore, we will check this function already in this part.

We will collect the code, we will tell the controller and we will see the result

Now let's get down to an enjoyable affair - beautiful images on the display from the image files.

To do this, I copy several drawings to the flash drive, one of my own, the rest - files from the sample folders, renamed into convenient names, so as not to get confused.

In the file ili9341.h, add a macro to convert the pixels into a 16-bit format.

#define WHITE 0xFFFF

**#define convert24to16 (x) (((x & 0x00f80000) >> 8) | ((x & 0x0000fc00) >> 5) | ((x & 0x000000f8) >> 3))**

Let's write a function for displaying an image

**void TFT9341\_DrawBitmap (uint16\_t x, uint16\_t y, char \* s)**

**{**

**uint16\_t i, h;**

**uint32\_t index = 0, width = 0, height = 0, bitpixel = 0;**

**FRESULT res; / \* FatFs function common result code \* /**

**uint32\_t bytesread;**

**uint32\_t tmpcolor;**

**uint8\_t \* bufbmp = NULL;**

**f\_close (& MyFile);**

**if (f\_open (& MyFile, s, FA\_READ)! = FR\_OK)**

**{**

**Error\_Handler ();**

**}**

**else**

**{**

**bufbmp = (uint8\_t \*) malloc (100);**

**res = f\_read (& MyFile, bufbmp, 90, (void \*) & bytesread);**

**if ((bytesread == 0) || (res! = FR\_OK))**

**{**

**Error\_Handler ();**

**}**

**else**

**{**

**index = bufbmp [10]; // the address of the beginning of the data in the file**

**index | = bufbmp [11] << 8;**

**index | = bufbmp [12] << 16;**

**index | = bufbmp [13] << 24;**

**width = bufbmp [18]; // width of the raster**

**width | = bufbmp [19] << 8;**

**width | = bufbmp [20] << 16;**

**width | = bufbmp [21] << 24;**

**height = bufbmp [22]; // height of the raster**

**height | = bufbmp [23] << 8;**

**height | = bufbmp [24] << 16;**

**height | = bufbmp [25] << 24;**

**bitpixel = bufbmp [28]; // pixel format (bit / pixel)**

**bitpixel | = bufbmp [29] << 8;**

**for (h = 0; h <height; h ++)**

**{**

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

**{**

**switch (bitpixel)**

**{**

**case 24:**

**f\_lseek (& MyFile, index + ((height-h-1) \* width \* 3) + (i \* 3));**

**res = f\_read (& MyFile, bufbmp, 3, (void \*) & bytesread);**

**if ((bytesread == 0) || (res! = FR\_OK))**

**{**

**Error\_Handler ();**

**}**

**tmpcolor = bufbmp [0];**

**tmpcolor | = bufbmp [1] << 8;**

**tmpcolor | = bufbmp [2] << 16;**

**break;**

**}**

**if (((i + x) <width) | ((h + y) <height))**

**TFT9341\_DrawPixel (i + x, h + y, (uint16\_t) convert24to16 (tmpcolor));**

**}**

**}**

**f\_close (& MyFile);**

**}**

**free (bufbmp);**

**}**

**}**

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

**Lesson 37**

**Part 9**

# Display TFT 240 × 320 8bit

In the [**last part of**](http://narodstream.ru/stm-urok-37-displej-tft-240x320-8bit-chast-8/) this lesson, we finished working with fonts and strings and started working with images already, writing the function of reading the image from USB Flash and displaying it on the screen.

Today we will continue this work.

Let's write a prototype of this function.

Add a test to display the pictures on the screen again in the body of the FileReadWrite function

                        TFT9341\_SetRotation (i% 4);

                        }

**TFT9341\_SetRotation (0);**

**TFT9341\_DrawBitmap (0, 0, "001.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "002.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "003.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "004.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "005.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "006.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "007.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "008.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "009.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "010.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "011.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "012.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "013.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "014.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "015.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "016.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "017.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "018.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "019.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "020.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "021.bmp");**

**HAL\_Delay (5000);**

**TFT9341\_DrawBitmap (0, 0, "022.bmp");**

**HAL\_Delay (5000);**

We will collect the code, we will sew the controller and check in practice



The same way all the other drawings are displayed. You will see the whole test in the video version attached below.

Uncomment the previous tests and check the code completely.