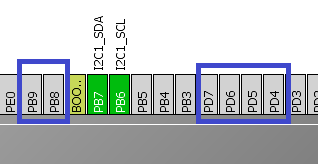
**Lesson 44**

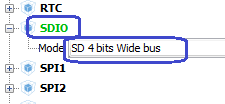
**SDIO. FATFS**

The project is created from the project I2CLCD80. Let's call it SDIO\_FATFS. Run the project in the Cube.

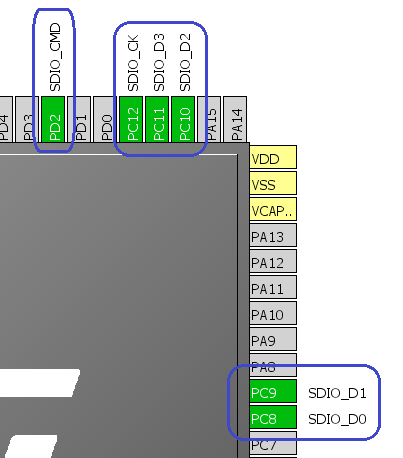
The paws of the ports PD4-PD7, PB8, PB9 are disconnected, this is a relic of past studies



Turn on SDIO



We will include the following port lugs

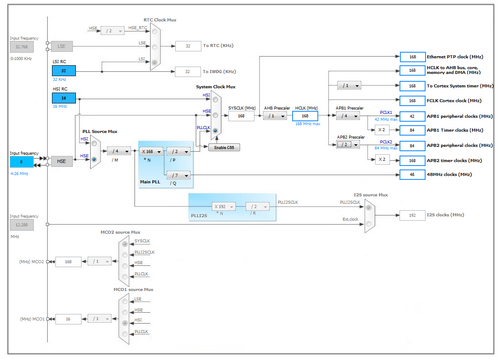


Based on this information, we will make a table for connecting the legs of the Micro SD

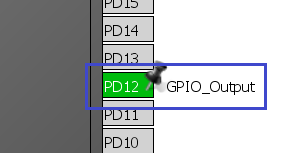
**How to connect Micro SD to the STM32F407VG microcontroller**

|  |  |  |
| --- | --- | --- |
| **Contact Micro SD** | **The port of the port of MK** | **Assignment of output** |
| **1** | **PC10** | **SDIO\_D2** |
| **2** | **PC11** | **SDIO\_D3** |
| **3** | **PD2** | **SDIO\_CMD** |
| **4** | **3V** | **+3.3 V (VDD)** |
| **5** | **PC12** | **SDIO\_CK** |
| **6th** | **GND** | **GND (VSS)** |
| **7th** | **PC8** | **SDIO\_D0** |
| **8** | **PC9** | **SDIO\_D1** |

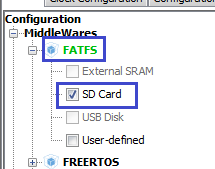
In the Clock Configuration, configure the dividers (click on the image to enlarge)

[](http://narodstream.ru/wp-content/uploads/2016/11/image07_1151.png)

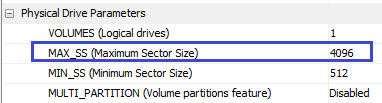
Enable PD12 on the output



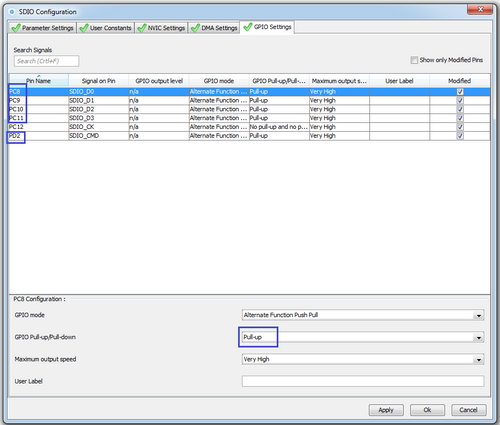
Turn on FATFS / SD Card



In FATFS Configuration, enable the support of large sectors just in case



And most importantly, those who do not have in the module or on the board on which the holder is mounted, tighten the resistance, turn it on the five main tabs SDIO (on all but CS) (click on the image to enlarge).

[](http://narodstream.ru/wp-content/uploads/2016/11/image02_0906.png)

Generate and open the project, connect lcd.c, configure the programmer for auto-cutting.

Let's run the controller to check whether we have connected everything correctly and whether the display is working.

From the main () function, remove the extra on the display, leaving only the initialization and cleaning

  / \* USER CODE BEGIN 2 \* /

        LCD\_ini ();

        LCD\_Clear ();

  / \* USER CODE END 2 \* /

You can delete the string variable

  / \* USER CODE BEGIN 1 \* /

~~char str [100];~~

  / \* USER CODE END 1 \* /

Let's add some variables to main.c

/ \* USER CODE BEGIN 0 \* /

FATFS SDFatFs; / \* File system object for SD card logical drive \* /

FIL MyFile; / \* File object \* /

extern char SD\_Path [4]; / \* SD logical drive path \* /

/ \* USER CODE END 0 \* /

Mount the partition from the map

        LCD\_Clear ();

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

**{**

**Error\_Handler ();**

**}**

**else**

**{**

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

**}**

  / \* USER CODE END 2 \* /

We will collect the project, we will sew the controller, we will check up, that at us all was mounted.

Open the file

  else

  {

**if (f\_open (& MyFile, "my001.txt", FA\_READ)! = FR\_OK)**

**{**

**Error\_Handler ();**

**}**

**else**

**{**

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

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

                }

        }

  / \* USER CODE END 2 \* /

Let's add some variables to main ()

  / \* USER CODE BEGIN 1 \* /

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

**uint32\_t byteswritten, bytesread; / \* File write / read counts \* /**

**uint8\_t wtext [] = "Hello from Stm32 !!!"; / \* File write buffer \* /**

**uint8\_t rtext [100]; / \* File read buffer \* /**

  / \* USER CODE END 1 \* /

Try to read the text from the file and display it on our display

                else

                {

**res = f\_read (& MyFile, rtext, sizeof (rtext), (void \*) & bytesread);**

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

**{**

**Error\_Handler ();**

**}**

**else**

**{**

**LCD\_Clear ();**

**LCD\_SetPos (0,0);**

**rtext [bytesread] = 0;**

**LCD\_String ((char \*) rtext);**

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

**}**

                }

We will collect the project, we will sew the controller and see the result of our work on the display.

Now let's try to write something down

The code, which is intended for reading, is duplicated, one double is recommended and the comment above is substituted

        // Read

// if (f\_mount (& SDFatFs, (TCHAR const \*) SD\_Path, 0)! = FR\_OK)

// {

// Error\_Handler ();

//}

// else

// {

// if (f\_open (& MyFile, "my001.txt", FA\_READ)! = FR\_OK)

// {

// Error\_Handler ();

//}

// else

// {

// res = f\_read (& MyFile, rtext, sizeof (rtext), (void \*) & bytesread);

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

// {

// Error\_Handler ();

//}

// else

// {

// LCD\_Clear ();

// LCD\_SetPos (0,0);

// rtext [bytesread] = 0;

// LCD\_String ((char \*) rtext);

// f\_close (& MyFile);

//}

//}

//}

And another duplicate will be corrected for writing, something also in it, first commenting, the function f\_close () will be transferred below for another bracket, so that the file should be closed strictly, we also change the filename and the flags of the file's opening and light, if successful, the LED

        // Write

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

  {

    Error\_Handler ();

  }

  else

  {

                if (f\_open (& MyFile, " **mywrite.txt**", **FA\_CREATE\_ALWAYS | FA\_WRITE**)! = FR\_OK)

                {

                        Error\_Handler ();

                }

                else

                {

// res = f\_read (& MyFile, rtext, sizeof (rtext), (void \*) & bytesread);

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

// {

// Error\_Handler ();

//}

// else

// {

// LCD\_Clear ();

// LCD\_SetPos (0,0);

// rtext [bytesread] = 0;

// LCD\_String ((char \*) rtext);

//}

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

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

}

If the LED is lit, then the functions of the display will be deleted, and also uncomment and correct the following lines

                else

                {

                        res = **f\_write**(& MyFile, **wtext**, sizeof (wtext), (void \*) & **byteswritten**);

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

                        {

                                Error\_Handler ();

                        }

                        else

                        {

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

                        }

                        f\_close (& MyFile);

**Let's**fix the text in the **wtext**variable **slightly**so that we do not think it's text from another file

  uint8\_t wtext [] = " **Hello write from Stm32 to Micro SD !!!**"; / \* File write buffer \* /

We will collect the project, we will sew the controller, disconnect the card, remove the flash card from the card holder, insert it into the card reader of the PC and make sure that the new file with the contents on our map appeared.