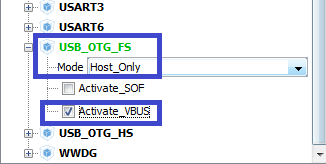
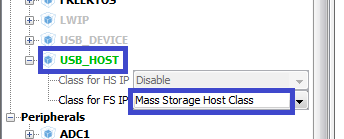
**USB. Host Mass Storage Class**

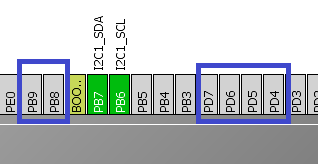
The project is created from the project I2CLCD80. Call it USB\_HOST\_MSC\_FATFS. Run the project in the Cube, enable USB\_OTG\_FS in the Host\_Only mode, turn on Activate\_VBUS.



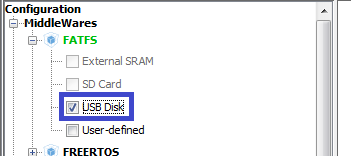
In USB\_DEVICE in the Class For FS IP section, select Mass Storage Host Class



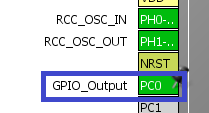
The paws of the ports PD4-PD7, PB8, PB9 are disconnected, this is a relic of past studies



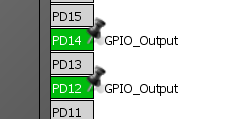
In the FATFS section we will enable USB Disk



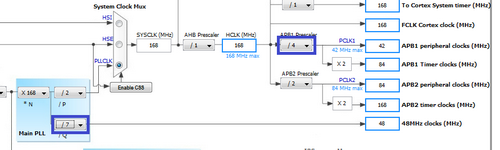
Paw PC0 switch on output



Turn on the output of the ports of the ports responsible for the red and green LEDs

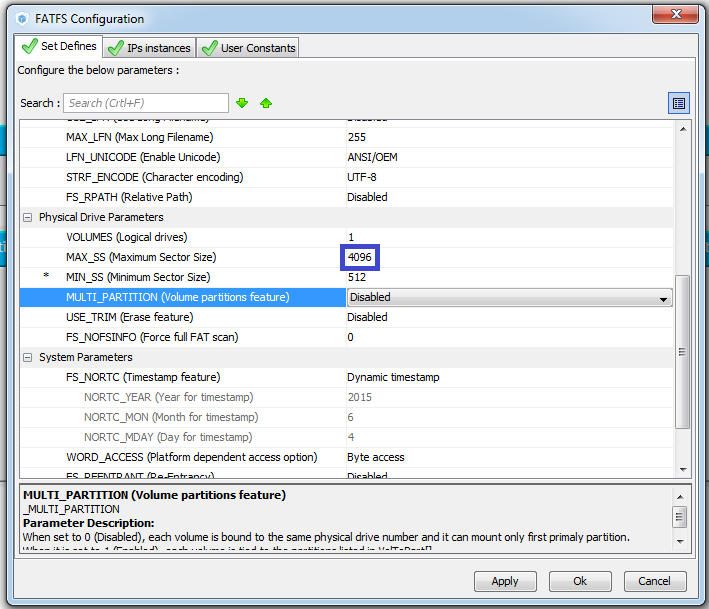


In Clock Configuration, select the following dividers (click on the image to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2016/12/image00_0862.png)

In the configuration, interrupts were displayed there themselves.

The only way to enable cluster support is up to 4 KB



Generate and run the project, connect lcd.c and configure the programmer to autoroute.

We will collect the project.

In main, we remove all user code except for initializing and cleaning the display

  / \* USER CODE BEGIN 2 \* /

        LCD\_ini ();

        LCD\_Clear ();

  / \* USER CODE END 2 \* /

In the file usb\_host.c we connect the file for the file system

/ \* USER CODE BEGIN 0 \* /

**#include "ff\_gen\_drv.h"**

/ \* USER CODE END 0 \* /

Also in this file we will replace something in the function USBH\_UserProcess

static void USBH\_UserProcess (USBH\_HandleTypeDef \* phost, uint8\_t id)

{

  / \* USER CODE BEGIN 2 \* /

  switch (id)

  {

  case HOST\_USER\_SELECT\_CONFIGURATION:

  break;

  case HOST\_USER\_DISCONNECTION:

~~Appli\_state = APPLICATION\_DISCONNECT;~~

**Appli\_state = APPLICATION\_IDLE;**

**f\_mount (NULL, (TCHAR const \*) "", 0);**

 break;

  case HOST\_USER\_CLASS\_ACTIVE:

  Appli\_state = **APPLICATION\_START**;

  break;

  case HOST\_USER\_CONNECTION:

**// Applied\_state = APPLICATION\_START;**

  break;

  default:

  break;

  }

  / \* USER CODE END 2 \* /

}

In the main.c file, we declare some variables

/ \* USER CODE BEGIN PV \* /

/ \* Private variables ------------------- \* /

**extern ApplicationTypeDef Appli\_state;**

**FATFS USBDISKFatFs; / \* File system object for USB disk logical drive \* /**

**FIL MyFile; / \* File object \* /**

**extern USBH\_HandleTypeDef hUsbHostFS;**

/ \* USER CODE END PV \* /

Also, in the main.c file, we create a function for reading, creating and editing files, and we will write down the inclusion of the green LED to see if we are in this function.

/ \* USER CODE BEGIN 0 \* /

**void FileReadWrite (void)**

**{**

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

**}**

/ \* USER CODE END 0 \* /

Let's add code to an infinite loop

    MX\_USB\_HOST\_Process ();

  / \* USER CODE BEGIN 3 \* /

**if (Appli\_state == APPLICATION\_START)**

**{**

**FileReadWrite ();**

**Appli\_state = APPLICATION\_IDLE;**

**}**

**else if (Appli\_state == APPLICATION\_IDLE)**

**{**

**}**

  }

  / \* USER CODE END 3 \* /

We connect the flash drive to the computer, create on it a text file "123.txt" with some content, extract it from the computer.

We connect the controller, flash drive to it through the OTG cable.

We will collect the project, we will sew the controller and see if we get into our function by glowing the green LED.

In the function, we hit, now we can write code here, because just hitting here, we are sure that everything is ready for us. Add some variables to this function

  HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_SET); **// here we see that we are here**

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

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

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

Call the function of connecting our flash drive. The result is also viewed with the help of LEDs

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

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

**{**

**/ \* FatFs Initialization Error \* /**

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

**HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_SET); // red color - an error signal**

**Error\_Handler ();**

**}**

**else**

**{**

**HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_RESET);**

**HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_SET); // here we see that we are here**

**}**

Call the function to open the file for reading. Also, the result will be tracked by the green LED, by transferring the function of calling the port pawl control to another location

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

  {

    / \* FatFs Initialization Error \* /

    Error\_Handler ();

  }

        else

        {

**if (f\_open (& MyFile, "123.txt", FA\_READ)! = FR\_OK)**

**{**

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

**HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_SET); // red color - an error signal**

**Error\_Handler ();**

**}**

**else**

**{**

**HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_14, GPIO\_PIN\_RESET);**

**HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_12, GPIO\_PIN\_SET); // here we see that we are here**

**}**

        }

Now write the code for reading the bytes from the file and displaying them on the display. In this regard, the monitoring of results using LEDs can be removed

                if (f\_open (& MyFile, "123.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);**

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

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

**}**

                }

In the [**next part of**](http://narodstream.ru/stm-urok-36-usb-host-mass-storage-class-chast-2/) this lesson, we will try to create a file and write information to it.