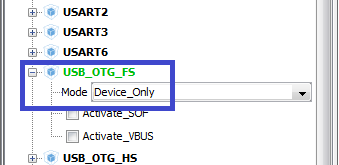
**Lesson 33**

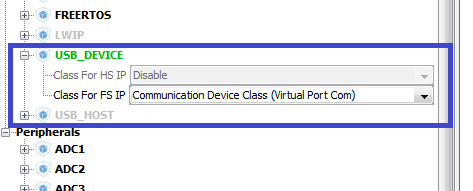
**HAL. USB. Virtual Com Port**

The debug card is the same:  **STM32F4-DISCOVERY** .

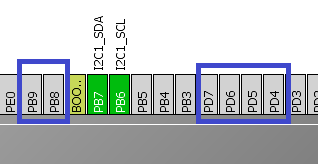
The project is created from the project I2CLCD80. Call it USB\_OTG\_CDC. Run the project in Cube, enable USB\_OTG\_FS in Device\_Only mode



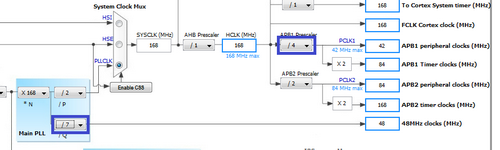
In USB\_DEVICE in the Class For FS IP section, select Communication Device Class (Virtual Port Com).



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



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

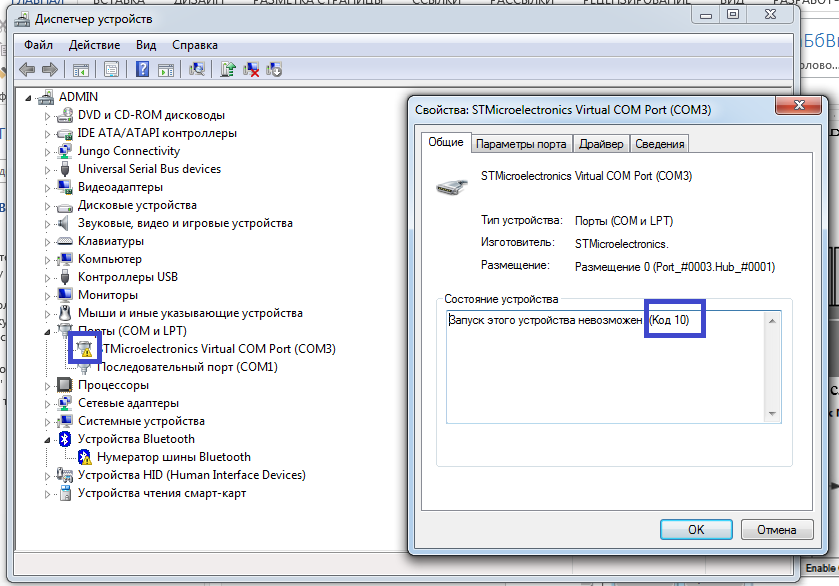
[](http://narodstream.ru/wp-content/uploads/2016/11/image04_0862.png)

In Configuration, we do not touch anything; interrupts there exhibited themselves.

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

We will collect the project. Let's sew the controller. We will have an unknown device, download the driver to our virtual usb device. To do this, go to the site st.com, in the search field there enter virtual com port, download and install the driver. Then it is desirable to go into the folder with the driver installed, select the folder corresponding to the bit depth of our operating system, and start the installation from there.

At us most likely the device will be installed with an error (the code 10)



There are several types of solutions, I liked this one. more simple: in the file usbd\_cdc.h we replace the packet size, instead of 512 we write 256 in this line:

#define CDC\_DATA\_HS\_MAX\_PACKET\_SIZE                 **256** / \* Endpoint IN & OUT Packet size \* /

We will collect, sew and see that the error has disappeared.

Let's start writing the code.

First, we'll try to transfer the data to the PC.

To do this, we first open the file usbd\_cdc\_if.c and fix it there in 2 lines 4 to 64

/ \* It's up to user to redefine and / or remove those define \* /

#define APP\_RX\_DATA\_SIZE   **64**

#define APP\_TX\_DATA\_SIZE   **64**

In the main.c file, we comment out all user code except for initializing and cleaning the display

  / \* USER CODE BEGIN 2 \* /

**LCD\_ini ();**

// sprintf (str, "Stm32F407VG");

// LCD\_String (str);

// LCD\_SetPos (10, 2);

// sprintf (str, "ARM mc");

// LCD\_String (str);

// HAL\_Delay (2000);

**LCD\_Clear ();**

// LCD\_SetPos (4, 0);

// LCD\_SendChar ('s');

// LCD\_SetPos (8, 1);

// LCD\_SendChar ('t');

// LCD\_SetPos (12, 2);

// LCD\_SendChar ('m');

// LCD\_SetPos (16, 3);

// LCD\_SendChar ('3');

// LCD\_SendChar ('2');

// HAL\_Delay (2000);

  / \* USER CODE END 2 \* /

Also in main.c we connect the file **usbd\_cdc\_if.h**for the visibility of the receiving and sending functions

/ \* USER CODE BEGIN Includes \* /

#include "main.h"

**#include "usbd\_cdc\_if.h"**

Let's change the string variable slightly in the main function, reducing the size in it and adding the prefix tx

  / \* USER CODE BEGIN 1 \* /

**char str\_tx [21];**

  / \* USER CODE END 1 \* /

In the file usbd\_cdc\_if.c we add the prototype of the transfer function by copying the advertisement from the implementation of this function in the same file

/ \* USER CODE BEGIN PRIVATE\_FUNCTIONS\_DECLARATION \* /

**uint8\_t CDC\_Transmit\_FS (uint8\_t \* Buf, uint16\_t Len);**

/ \* USER CODE END PRIVATE\_FUNCTIONS\_DECLARATION \* /

In main () we add the data to a string

**sprintf (str\_tx, "USB Transmit \ r \ n");**

  / \* USER CODE END 2 \* /

In an infinite loop, we will try to send this data to the USB port using the function whose prototype we added

  while (1)

  {

**CDC\_Transmit\_FS ((unsigned char \*) str\_tx, strlen (str\_tx));**

                HAL\_Delay (500);

  / \* USER CODE END WHILE \* /

We will collect the code, we will sew the controller and see the result in the terminal program.

It seems to us that something has succeeded. Now let's try to accept something. It's a bit more complicated here. for this, the interrupt handler is already used, which is the function CDC\_Receive\_FS in the file usbd\_cdc\_if.c.

Let's add one more string global variable in main ()

/ \* USER CODE BEGIN PV \* /

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

**char str\_rx [21];**

/ \* USER CODE END PV \* /

We also declare it in the file usbd\_cdc\_if.c

/ \* USER CODE BEGIN PRIVATE\_VARIABLES \* /

**extern char str\_rx [21];**

/ \* USER CODE END PRIVATE\_VARIABLES \* /

In the function CDC\_Receive\_FS in the same file we add some code and something we comment out

static int8\_t CDC\_Receive\_FS (uint8\_t \* Buf, uint32\_t \* Len)

{

  / \* USER CODE BEGIN 6 \* /

**//**USBD\_CDC\_SetRxBuffer (& hUsbDeviceFS, & Buf [0]);

**strncpy (str\_rx, (char \*) Buf, \* Len);**

**str\_rx [\* Len] = 0;**

  USBD\_CDC\_ReceivePacket (& hUsbDeviceFS);

  return (USBD\_OK);

Let's add a variable to main ()

  / \* USER CODE BEGIN 1 \* /

**uint8\_t i = 0;**

        char str\_tx [21];

Entered into our buffer data, try to display, for this in an infinite loop in the main () function, add a specific code

  while (1)

  {

                CDC\_Transmit\_FS ((unsigned char \*) str\_tx, strlen (str\_tx));

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

**LCD\_String (str\_rx);**

**for (i = 1; i <((uint8\_t) (20-strlen (str\_rx))); i ++) LCD\_SendChar ('');**

                HAL\_Delay (500);

We will collect the project. Let's code the code and see the result, typing in the terminal program and sending some lines to the USB port.