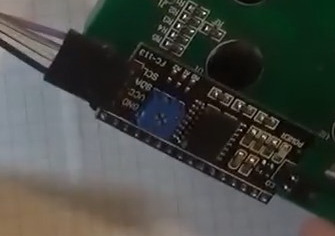
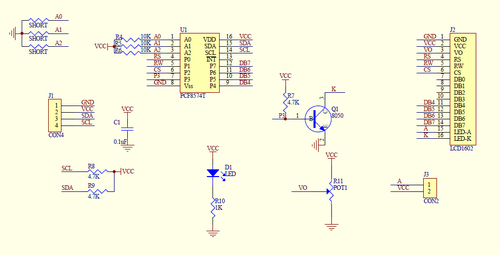
**HAL. I2C. Adapter for LCD 20 × 4**

Today we will try to connect a **20 × 4** character **LCD** display  via an **I2C** bus **adapter** . This will allow us to save the feet of the ports, as well as to get by with fewer wires.

This adapter looks like this and its connector is fully compatible on the legs with a display connector



This is nothing more than a serial to parallel converter based on the **PCF8574** chip  . Here is the adapter circuit (click on the image to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2016/11/Image02-2.png)

According to the scheme we see which feet of the display connector we have the legs of the chip.

We create the project from **MYLCD80** , call it **I2CLCD80** .

Start the Cube. Since the project was created in an earlier version of the Cube, then in response to the dialogue with the three buttons, press "Migrate" so that our project adapts to the new version.

Turn on I2C1. Do not touch anything anywhere, remember the tabs SDA and SCL, generate the project, add the file lcd.c, configure the programmer, collect the code.

Open the file lcd.h and fix everything there according to the requirements of the I2C bus.

We remove the inclusion of the port legs completely, because They are not used by us and this will be done by the microcircuit

#include "stm32f4xx\_hal.h"

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

~~#define d4\_set () HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_4, GPIO\_PIN\_SET)~~

~~#define d5\_set () HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_5, GPIO\_PIN\_SET)~~

~~#define d6\_set () HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_6, GPIO\_PIN\_SET)~~

~~#define d7\_set () HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_7, GPIO\_PIN\_SET)~~

~~#define d4\_reset () HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_4, GPIO\_PIN\_RESET)~~

~~#define d5\_reset () HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_5, GPIO\_PIN\_RESET)~~

~~#define d6\_reset () HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_6, GPIO\_PIN\_RESET)~~

~~#define d7\_reset () HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_7, GPIO\_PIN\_RESET)~~

Before we change, we need a function that will send certain data to the adapter. In order to add it to us, open the file lcd.h and write the function

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

**void WriteByteI2CLCD (uint8\_t bt)**

**{**

**buf [0] = bt;**

**HAL\_I2C\_Master\_Transmit (& hi2c1, (uint16\_t) 0x4E, buf, 1,1000);**

**}**

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

Let's return to **lcd.h**

All the rest is done here on the basis of the above scheme

**#define e\_set () WriteByteI2CLCD (portlcd | = 0x04) // set line E to 1**

**#define e\_reset () WriteByteI2CLCD (portlcd & = ~ 0x04) // set line E to 0**

**#define rs\_set () WriteByteI2CLCD (portlcd | = 0x01) // set the RS line to 1**

**#define rs\_reset () WriteByteI2CLCD (portlcd & = ~ 0x01) // set the RS line to 0**

**#define setled () WriteByteI2CLCD (portlcd | = 0x08) // turn on backlight**

**#define setwrite () WriteByteI2CLCD (portlcd & = ~ 0x02) // set record to the memory of the display**

Now correct the file **lcd.c**

We will write there some defines

#include "lcd.h"

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

**extern I2C\_HandleTypeDef hi2c1;**

**uint8\_t buf [1] = {0};**

**char str1 [100];**

**uint8\_t portlcd = 0; // cell for storing the port data of the expansion chip**

Then change the delay function, we'll do it in microseconds

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

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

**{**

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

**/ \* Wait till done \* /**

**while (micros-);**

**}**

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

We go further in order.

We remove the function LCD\_WriteData.

We remove the functions LCD\_Command and LCD\_Data, instead of them we add other functions

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

**void sendhalfbyte (uint8\_t c)**

**{**

**c << = 4;**

**e\_set (); // turn on the line E**

**DelayMicro (50);**

**WriteByteI2CLCD (portlcd | c);**

**e\_reset (); // turn off the line E**

**DelayMicro (50);**

**}**

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

**void sendbyte (uint8\_t c, uint8\_t mode)**

**{**

**if (mode == 0) rs\_reset ();**

**else rs\_set ();**

**uint8\_t hc = 0;**

**hc = c >> 4;**

**sendhalfbyte (hc); sendhalfbyte (c);**

**}**

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

Next, we correct this function

void LCD\_Clear (void)

{

**sendbyte (0x01, 0);**

**DelayMicro (1500);**

}

This, too, will be corrected

void LCD\_SendChar (char ch)

{

**sendbyte (ch, 1);**

}

This function will be slightly modified

void LCD\_SetPos (uint8\_t x, uint8\_t y)

{

        switch (y)

        {

                case 0:

**sendbyte**(x | 0x80 **, 0**);

                        HAL\_Delay (1);

                        break;

                case 1:

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

                        HAL\_Delay (1);

                        break;

                case 2:

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

                        HAL\_Delay (1);

                        break;

                case 3:

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

                        HAL\_Delay (1);

                        break;

        }

}

Now this

void LCD\_ini (void)

{

**HAL\_Delay (15);**

**sendhalfbyte (0x03);**

**HAL\_Delay (4);**

**sendhalfbyte (0x03);**

**DelayMicro (100);**

**sendhalfbyte (0x03);**

**HAL\_Delay (1);**

**sendhalfbyte (0x02);**

**HAL\_Delay (1);**

**sendbyte (0x28, 0);**// 4bit mode (DL = 0) and 2 lines (N = 1)

**HAL\_Delay (1);**

**sendbyte (0x0C, 0);**// turn on the image on the display (D = 1), we do not turn on the cursors (C = 0, B = 0)

**HAL\_Delay (1);**

**sendbyte (0x6, 0); // the cursor (even though it's invisible) will move to the left**

**HAL\_Delay (1);**

**setled (); // backlight**

**setwrite (); // write**

}

This one is insignificant

void LCD\_String (char \* st)

{

        uint8\_t i = 0;

        while (st [i]! = 0)

        {

**sendbyte**(st [i] **, 1**);

                i ++;

        }

}

Stitch, watch.