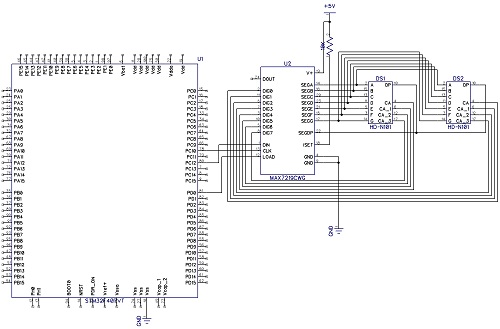
**Lesson 26**

**HAL. SPI. The MAX7219 driver**

We continue to deal with the SPI bus. In the [**last lesson,**](http://narodstream.ru/stm-urok-25-hal-spi-led-dinamicheskaya-indikaciya/) we connected four indicators to two shift registers and achieved discharge control.

And today we will also manage a multi-bit indicator, but with the help of a specialized driver chip, also connected via the SPI interface to the microcontroller. This chip is MAX7219. We will study this microcircuit only in general terms, since we have already thoroughly studied it in a similar [**lesson on AVR**](http://narodstream.ru/avr-urok-28-spi-drajver-led-max7219/) , I think it makes no sense to repeat it. Therefore, today's goal is to learn how to control this chip using the STM32 controller. We still have the same debug card - **STM32 F4 Discovery** .

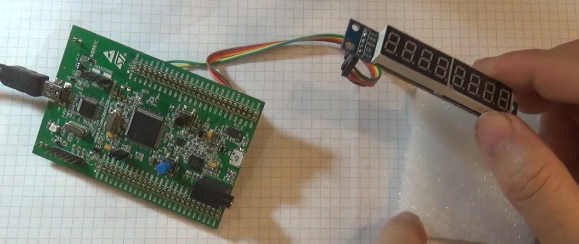
Here is a diagram of connecting this chip to the controller

[](http://narodstream.ru/wp-content/uploads/2017/01/image01-1.jpg)

To the chip we connect two four-digit indicators, and with a common cathode. And it's not we who connect, it's all done as a separate module

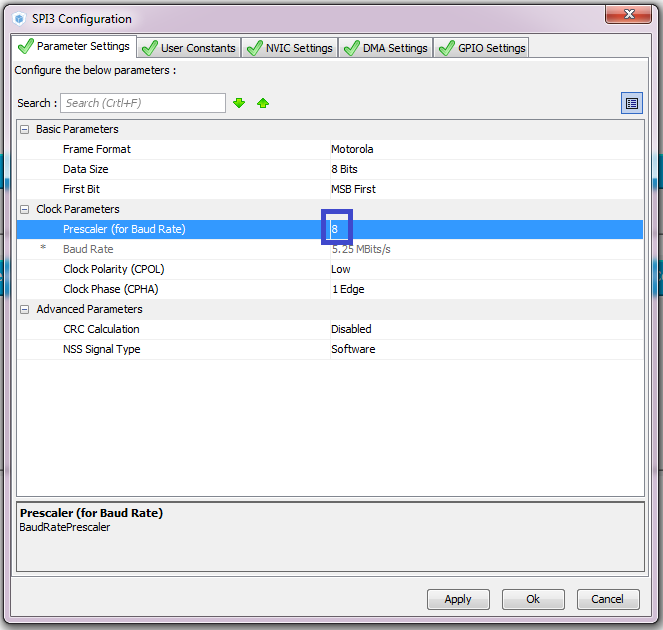


And so, in practice, the entire connection scheme looks like



The project is created from the previous year's project SPI\_595\_LED, since there are no timers there, and they will not be needed. We call it LED7219. The files led.c and led.h are renamed max7219.c and max7219.h respectively.

Start the Cube. Reduce the speed of SPI a bit, because the documentation for the chip costs 10 MHz



Generate and open the project in Keil.

We are looking at the connection of indicators.

We move on to the project. We connect the file max7219.c and configure the programmer.

We will collect the project.

In main.h and max7219.c, we fix the file connection

~~#include "led.h"~~

**#include "max7219.h"**

Correct the headers and remove the files in the file max7219.h. Will remain there that's what

#ifndef **MAX7219**\_H\_

#define **MAX7219**\_H\_

#include "stm32f4xx\_hal.h"

#include "main.h"

#endif / \* **MAX7219 \_H\_**\* /

In the main.c file, remove all variables

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

~~#define cs\_set () HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_0, GPIO\_PIN\_RESET)~~

~~#define cs\_reset () HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_0, GPIO\_PIN\_SET)~~

~~#define cs\_strob () cs\_reset (); cs\_set ()~~

~~uint8\_t aTxBuffer [1] = {0};~~

/ \* USER CODE END PV \* /

Remove one of the defines from the file max7219.c

#define cs\_set () HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_0, GPIO\_PIN\_RESET)

#define cs\_reset () HAL\_GPIO\_WritePin (GPIOD, GPIO\_PIN\_0, GPIO\_PIN\_SET)

~~#define cs\_strob () cs\_reset (); cs\_set ()~~

Instead of portseg, add another variable

~~uint8\_t portseg = 0;~~

**char dg = 8;**

a bit change the variable aTxBuffer [1], remove from it extern and change the name a little

uint8\_t **aTxBuf**[1] **= {0}**;

Let's see the datasheet on the microcircuit max7219 - how data is transferred, what registers exist.

remove the code of the function segchar well, or instead of it we'll do another function

**void Send\_7219 (uint8\_t rg, uint8\_t dt)**

**{**

**cs\_set ();**

**aTxBuf [0] = rg;**

**HAL\_SPI\_Transmit (& hspi3, (uint8\_t \*) aTxBuf, 1, 5000);**

**aTxBuf [0] = dt;**

**HAL\_SPI\_Transmit (& hspi3, (uint8\_t \*) aTxBuf, 1, 5000);**

**cs\_reset ();**

**}**

Let's make a prototype

**void Send\_7219 (uint8\_t rg, uint8\_t dt);**

Let's write the function of clearing the indicator

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

**void Clear\_7219 (void)**

**{**

**uint8\_t i = dg;**

**do**

**{**

**Send\_7219 (i, 0xF); // the emptiness symbol**

**} while (-i);**

**}**

Write the function of outputting a number to the indicator

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

**void Number\_7219 (volatile long n)**

**{**

**uint8\_t ng = 0; // variable for minus**

**if (n <0)**

**{**

**ng = 1;**

**n \* = - 1;**

**}**

**Clear\_7219 ();**

**if (n == 0)**

**{**

**Send\_7219 (0x01,0); // to the first digit write 0**

**return;**

**}**

**uint8\_t i = 0;**

**do**

**{**

**Send\_7219 (++ i, n% 10);**

**n / = 10;**

**} while (n);**

**if (ng)**

**{**

**Send\_7219 (i + 1,0xA); // character -**

**}**

**}**

Also write the function of initialization of the chip

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

**void Init\_7219 (void)**

**{**

**Send\_7219 (0x09,0xFF); // enable the decoding mode**

**Send\_7219 (0x0B, dg-1); // number of bits used**

**Send\_7219 (0x0A, 0x02); // the intensity of the glow**

**Send\_7219 (0x0C, 0x01); // turn on the indicator**

**Clear\_7219 ();**

**}**

We will write prototypes for added functions

void Send\_7219 (uint8\_t rg, uint8\_t dt);

**void Init\_7219 (void);**

**void Number\_7219 (volatile long n);**

**void Clear\_7219 (void);**

In the file main.c, we correct the type of the counter variable

        uint **32**\_t i = 0;

Here we remove everything except the delay

  / \* USER CODE BEGIN 2 \* /

~~cs\_set ();~~

~~aTxBuffer [0] = 0xFF;~~

~~HAL\_SPI\_Transmit (& hspi3, (uint8\_t \*) aTxBuffer, 1, 5000);~~

~~cs\_strob ();~~

        HAL\_Delay (200);

  / \* USER CODE END 2 \* /

In an infinite loop, the code is still commented out

  while (1)

  {

// for (i = 0; i <= 9; i ++)

// {

// segchar (i);

// HAL\_Delay (1000);

//}

  / \* USER CODE END WHILE \* /

After the delay, we will write the code for displaying the digits on the display

 HAL\_Delay (200);   
**Init\_7219 ();   
 Send\_7219 (1.0 × 01); // 1   
 Send\_7219 (2,0 × 02); // 2   
 Send\_7219 (3,0 × 03); // 3   
 Send\_7219 (4,0 × 04); // 4   
 Send\_7219 ( 5.0 × 05); // 5   
 Send\_7219 (6.0 × 06); // 6   
 Send\_7219 (7.0 × 07); // 7   
 Send\_7219 (8.0 × 08); // 8**

Then wait 2 seconds and try to output a negative number

 Send\_7219 (8.0 × 08); // 8   
**HAL\_Delay (2000);   
 Clear\_7219 ();   
 Number\_7219 (-4356); // try to print a negative value**

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

Wait another 2 seconds and clean the display

 Number\_7219 (-4356); // try to print a negative value of   
**HAL\_Delay (2000);   
 Clear\_7219 ();**  
  / \* USER CODE END 2 \* /

In an infinite loop, remove the for loop. We can not physically exceed the number of indicator symbols, even if we wait for the maximum value of the type of the variable counter. Reduce the delay, the code is also slightly modified.

  while (1)

  {

**Number\_7219 (i);**

                HAL\_Delay ( **200**);

**i ++;**

  / \* USER CODE END WHILE \* /

We collect, sew, look

