**Lesson 25**

**HAL. SPI. LED Dynamic display**

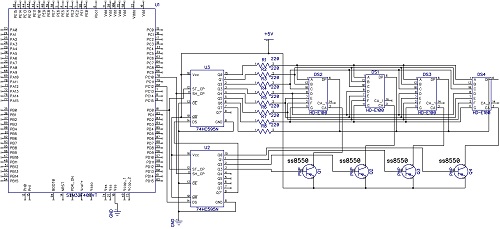
In the  [**previous lesson,**](http://narodstream.ru/stm-urok-24-hal-spi-led-staticheskaya-indikaciya/) we studied the possibility of controlling via the SPI bus a seven-segment LED indicator.

And in this lesson we will try to manage already four such indicators, and we will use dynamic indications already.

What distinguishes the dynamic indication from the usual static, you can learn from the [**lesson 12**](http://narodstream.ru/stm-urok-12-hal-dinamicheskaya-indikaciya/) .

The main difference is that the indicators or discharges glow all at once, but in turn.

Before we go directly to the project, let's look at the connection diagram (click on the image to enlarge the image)

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

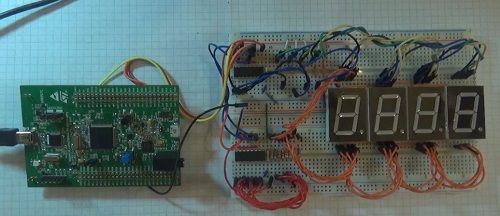
First, we see that there are now two shift registers, since eight ports will not be enough for us. One of the registers will control the segments, and the second - the bits.

We will also apply a cascaded or circular method of connecting to the SPI. Therefore, in parallel we only Chip Select and the synchronization feet. And from the MISO foot of the controller, the signal will go first to the digital input of the lower register, and already on from the digital output of the lower register the signal will go to the digital input of the upper one. MOSI we also do not use, because we do not need a report, we'll see everything on the glow of the indicators. The segments of all four indicators are connected in parallel to the parallel outputs of the upper register, and the common anodes of the indicators are connected through the key transistors to the four lower parallel outputs of the lower register.

In general, first in the lower case we send a byte for the upper case, and then for the lower case, and the byte that was already there will be moved to the upper case, thereby the bytes will all be in their places. And then, having generated a pulse on the SS leg, we will move all the bytes into registers intended for parallel outputs, and thus the signals will arrive at the indicators.

Also, the circuit has a somewhat simplified version and it does not indicate the resistors of the current limit of the base by 2.2 kilo. So do not forget to put these resistors between the base transistors and the four outputs of the register.

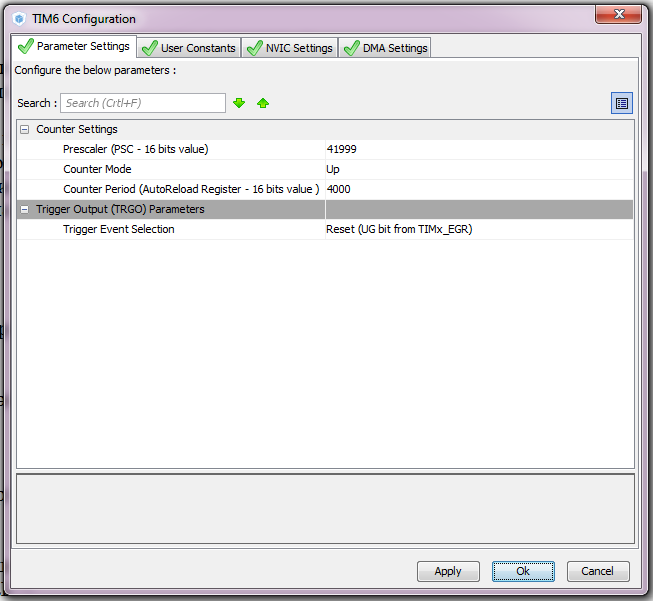
And this is how our scheme looks in practice

[](http://narodstream.ru/wp-content/uploads/2017/01/image02.jpg)

We create the project from the previous project SPI\_595\_LED, we call it SPI\_595\_DYN\_LED.

Start the Cube. Let's add a timer 6.

Configure it so far like this - once per second



Interrupts also need to be enabled.

Generate and open the project in Keil.

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

We will collect the project.

In the main () function, run the timer

  / \* USER CODE BEGIN 2 \* /

**HAL\_TIM\_Base\_Start\_IT (& htim6);**

In an endless loop, while everything is commented on

  while (1)

  {

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

// {

// segchar (i);

// HAL\_Delay (1000);

//}

Few adjust the start code, because we have two chips

        cs\_set ();

        aTxBuffer [0] = 0xFF; **// cancel all segments**

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

        cs\_strob ();

**aTxBuffer [0] = 0x0F; // cancel all digits**

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

**cs\_strob ();**

        HAL\_Delay (200);

  / \* USER CODE END 2 \* /

Add the interrupt function from the timer to the main module of the program

/ \* USER CODE BEGIN 4 \* /

**void HAL\_TIM\_PeriodElapsedCallback (TIM\_HandleTypeDef \* htim6)**

**{**

**}**

Add the count counter

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

**uint8\_t n\_count = 0;**

/ \* USER CODE END PV \* /

In led.c we add variables for digits

uint8\_t portseg = 0;

**uint8\_t R1 = 0, R2 = 0, R3 = 0, R4 = 0;**

Projecting them in main.c

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

**extern uint8\_t R1, R2, R3, R4;**

uint8\_t n\_count = 0;

The function ledprint for the file led.c is taken from the same project file LED\_DYN from [**lesson 12**](http://narodstream.ru/stm-urok-12-hal-dinamicheskaya-indikaciya/) .

// ==============================

**void ledprint (uint16\_t number)**

**{**

**R1 = number% 10;**

**R2 = number% 100/10;**

**R3 = number% 1000/100;**

**R4 = number / 1000;**

**}**

Also do not forget about the prototype in led.h

void segchar (uint8\_t seg);

**void ledprint (uint16\_t number);**

In the body of the function HAL\_TIM\_PeriodElapsedCallback in main.c we also copy the code from the project LED\_DYN, but from the function TIM6\_DAC\_IRQHandler of the file stm32f4xx\_it.c

**if (n\_count == 0)**

**{**

**HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_3, GPIO\_PIN\_RESET);**

**HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_4 | GPIO\_PIN\_5 | GPIO\_PIN\_6, GPIO\_PIN\_SET);**

**segchar (R1);**

**}**

**if (n\_count == 1)**

**{**

**HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_4, GPIO\_PIN\_RESET);**

**HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_3 | GPIO\_PIN\_5 | GPIO\_PIN\_6, GPIO\_PIN\_SET);**

**segchar (R2);**

**}**

**if (n\_count == 2)**

**{**

**HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_5, GPIO\_PIN\_RESET);**

**HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_3 | GPIO\_PIN\_4 | GPIO\_PIN\_6, GPIO\_PIN\_SET);**

**segchar (R3);**

**}**

**if (n\_count == 3)**

**{**

**HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_6, GPIO\_PIN\_RESET);**

**HAL\_GPIO\_WritePin (GPIOE, GPIO\_PIN\_3 | GPIO\_PIN\_4 | GPIO\_PIN\_5, GPIO\_PIN\_SET);**

**segchar (R4);**

**}**

**n\_count ++;**

**if (n\_count> 3) n\_count = 0;**

Let's change this code slightly in light of the requirements of the chip and the SPI bus

        if (n\_count == 0)

        {

                segchar (R1);

**aTxBuffer [0] = 0x0E;**

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

**cs\_strob ();**

        }

        if (n\_count == 1)

        {

                segchar (R2);

**aTxBuffer [0] = 0x0D;**

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

**cs\_strob ();**

        }

        if (n\_count == 2)

        {

                segchar (R3);

**aTxBuffer [0] = 0x0B;**

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

**cs\_strob ();**

        }

        if (n\_count == 3)

        {

                segchar (R4);

**aTxBuffer [0] = 0x07;**

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

**cs\_strob ();**

        }

        n\_count ++;

        if (n\_count> 3) n\_count = 0;

We will collect the project, we will sew it and we will look at switching zeros



Now try to print a number

        HAL\_Delay (200);

**ledprint (1234);**

  / \* USER CODE END 2 \* /



Let's try to increase the speed in the Cube, for this period we will reduce the Counter Period from 4000 to 1000

Let's sew and see the result, it's faster, but it's not quite right.

Now change to 200

Let's sew and see the result, again not enough. But nothing, then we'll add.

Now try to count

To do this, we change the type of the variable for the account

  / \* USER CODE BEGIN 1 \* /

**uint16\_t** i = 0;

  / \* USER CODE END 1 \* /

Also uncomment the code in an infinite loop and make some changes, applying another function and increasing the number of numbers, and also reducing the delay

                for (i = 0; i <= **9999**; i ++)

                {

**ledprint**(i);

                        HAL\_Delay ( **100**);

                }

This, accordingly, can be removed

        HAL\_Delay (200);

**~~ledprint (1234);~~**

Again, we'll reduce the period by 10 times in the Cube, setting it at 20.

Let's see the result, we also see a flicker.

If we remove 4 times more, we will get the same result as in the project LED\_DYN

Therefore, we now set the Counter Period 5

Let's see the result, like normal



In the [**next lesson,**](http://narodstream.ru/stm-urok-26-hal-spi-drajver-indikatora-max7219/) we will try to connect a dedicated microcircuit for digital multi-bit LED indicators, also using the SPI interface.