**HAL. Dynamic display**

We continue to work with seven-segment indicators. We will just work with several indicators at once, so that we can get numbers from 1 to 9 on them, as this is not really interesting, but also large numbers.

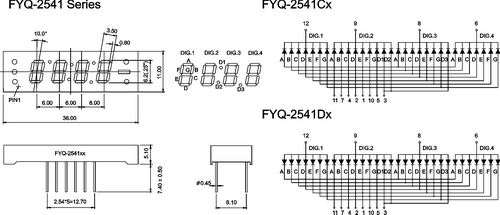
Only we will connect them not to the 8 legs of the port each, but all the available indicators are parallel to only eight legs. **Dynamic indication** will help us in this .

**Dynamic indication** is a kind of information display using LED indicators, in which only one digit of one individual indicator is displayed at a certain moment of time, then the next one, and the previous one is extinguished, and so on. Only all this is shown with such speed that the human eye (or rather the brain) does not have time to estimate that the numbers are burning in turn and it seems to him that the numbers are burning all together simultaneously. This type of indication allows you to save an impressive number of port legs.

This is achieved as follows. On all the legs of all indicators, in sequence, in turn, the levels are given first of one digit, intended, for example, for the first indicator, then also all together, the levels of the digit intended for the second indicator, and so on. Otherwise, it will not work either, because their paws are all paralleled. Only at the time when a figure is sent for a certain indicator, then at its cathode, if it is supplied with a common cathode, a low level is applied, and all other cathodes are high, and consequently the remaining indicators will not light up. And if it is with a common anode, then a high voltage is applied to the required anode, and all others are low and they will also not glow.

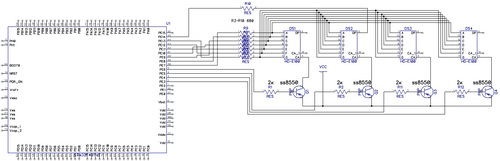
According to the dynamic display, we already had a [**lesson for AVR**](http://narodstream.ru/avr-urok-11-dinamicheskaya-indikaciya/) , and those who are dealing with STM are already more prepared comrades, so it's useless to tell all this in detail.

Let's look at the diagram of the 4-bit 7-segment indicator itself. I have an indicator of this type here (click on the picture to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2016/12/Image00-4.png)

I also have an indicator with a common anode, so let's stick to this scheme for writing code.

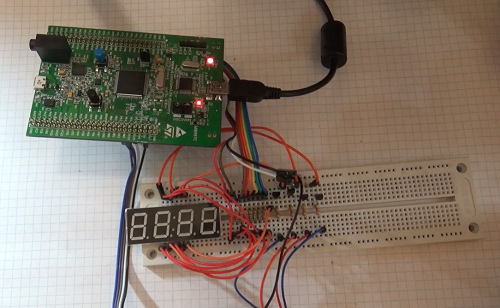
So here we have everything connected to the controller (click on the image to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2016/12/Image01.jpg)

There are several indicators on the diagram, so I did not find a single four-digit program in the schematic drawing program, but it's even clear.

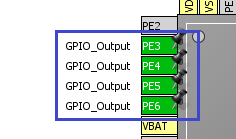
Common anodes are connected via key transistors in inverse mode. current limiting resistors are also used.

This is how the scheme looks in practice (click on the image to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2016/12/Image03-6.png)

As before, we create the project from LED\_STAT, call it LED\_DYN.

Start the cube, turn on the output of 4 more ports to indicate PE3-PE6.



In the settings in Configuration do not touch anything at all.

We generate the project.

Adding the file led.c

We collect, sew, look.



At us all works, but works simultaneously, since we do not manage common anodes in the meantime.

In the main function main (), we change the initialization of the tabs so that the indicators do not shine

  / \* USER CODE BEGIN 2 \* /

**HAL\_ GPIO\_PIN\_4 |**

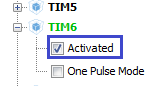
**| GPIO\_PIN\_7 | GPIO\_PIN\_8 | GPIO\_PIN\_9 | GPIO\_PIN\_10**

**| GPIO\_PIN\_11 | GPIO\_PIN\_12 | GPIO\_PIN\_13 | GPIO\_PIN\_14,**

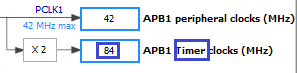
**GPIO\_PIN\_SET);**

Once again we collect, we sew, we look.

Close the project, go into the cube, add the 6th timer

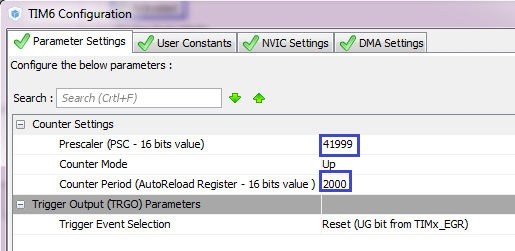


Watch Clock Configuration - the frequency of APB1 costs 84. We leave it unchanged

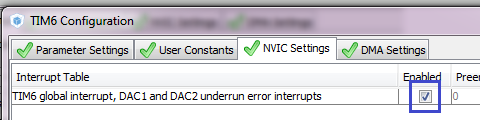


Go to Configuration-> TIM6

We set the parameter Prescaler 41999, Counter Period 2000. It should be 1 second



Turn on interrupts



Apply the changes in the timer, generate the project and open it in Keil.

In the main.c file we start the timer

  HAL\_ GPIO\_PIN\_4 |

        | GPIO\_PIN\_7 | GPIO\_PIN\_8 | GPIO\_PIN\_9 | GPIO\_PIN\_10

          | GPIO\_PIN\_11 | GPIO\_PIN\_12 | GPIO\_PIN\_13 | GPIO\_PIN\_14,

        GPIO\_PIN\_SET);

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

In the file stm32f4xx\_it.c we find the function for interrupting TIM6\_DAC\_IRQHandler and write there to check the timer

HAL\_TIM\_IRQHandler (& htim6);

/ \* USER CODE BEGIN TIM6\_DAC\_IRQn 1 \* /

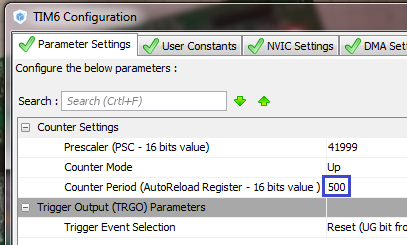
**HAL\_GPIO\_TogglePin (GPIOE, GPIO\_PIN\_3 | GPIO\_PIN\_4 | GPIO\_PIN\_5 | GPIO\_PIN\_6);**

All this is just for an experiment, to check if the timer is working at all.

Once again we collect, we sew, we look.

Our indicators blink with a period of 2 seconds, since this is a switching function, and at first the indicators turn off, turn on in a second, and then turn off again after a second. Therefore, we see that the figures in our country are blinking and are shown through one. I will not even show it, it's very ugly. But you, of course, can see it in the video version of this lesson, which is attached at the very bottom of this page with a lesson.

Close the project, go into the cube, change the Counter Period to 500



We generate the project.

In the file stm32f4xx\_it.c add a variable for the bit counter, and at the same time connect the file led.h

#include "stm32f4xx\_it.h"

/ \* USER CODE BEGIN 0 \* /

**#include "led.h"**

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

We remove our code from TIM6\_DAC\_IRQHandler function and write the following code:

  / \* USER CODE BEGIN TIM6\_DAC\_ IRQn 1 \* /

**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);**

**}**

**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);**

**}**

**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);**

**}**

**n\_count ++;**

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

Once again we collect, we sew, we look



We see that the numbers now turn on in turn.

Of course, while the timer's speed is small, we can watch it. In general, it should not be so that the eye saw the process of dynamic indication, but I specifically increased only 4 times the speed. Now we, on the contrary, are interested in following this process.

Now our task is to transfer to each digit a specially designed figure for it.

In the file led.c we add variables for digits

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

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

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

There we add a new function

**// ---------------**

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

**{**

**R1 = number% 10;**

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

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

**R4 = number / 1000;**

**}**

Let's create a prototype for it.

In the file stm32f4xx\_it.c, the variables are

uint8\_t n\_count = 0;

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

/ \* USER CODE END 0 \* /

Add the lines in the function TIM6\_DAC\_IRQHandler

  / \* USER CODE BEGIN TIM6\_DAC\_IRQn 1 \* /

        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;

In the file main.c we comment everything in an infinite loop.

We add there function call

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

**ledprint (1234);**

Once again we collect, we sew, we look



We see that the number "1234" is displayed, but due to the low rate of change of digits, the numbers are shown in turn.

Close the project, go into the cube, change the Counter Period to 20, it will be 1/100 of a second.

Generate the project, open it in Keil.

We remove all the code from the infinite loop, so that we do not get artifacts in the first digit or comment it out.

We collect the project, we sew it, we look and see that there is a flicker (here it is not very noticeable, in the video too, it is visible only with the natural eye)



Again we go into the cube, change the Counter Period to **10** .

We generate the project, collect, flash, look and see that the flickers are gone. Here there is no point to show it, especially since the video on the contrary, in this case flickers more. than at the period of 20, apparently, this is due to the mismatch of the timer frequency with the frame rate of the camera.

Now remove the line

~~ledprint (1234);~~

Change the type of the variable i

  / \* USER CODE BEGIN 1 \* /

**uint16\_t i = 0;**

  / \* USER CODE END 1 \* /

Uncomment the counter in an infinite loop and add the limit there and change the function of calling the output of the character to the function of calling the output of a four-digit number and decreasing the delay by an order of magnitude.

**for (i = 0; i <10000; i ++)**

**{**

**ledprint (i);**

**HAL\_Delay (100);**

**}**

Once again we collect, we sew, we look

