**Lesson 28**

**HAL. DAC. Triangle**

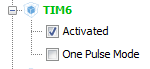
In the [**last lesson,**](http://narodstream.ru/stm-urok-27-hal-dac/) we got acquainted with a **digital-to-analog converter (DAC)**  and studied how it is implemented in **STM32** microcontrollers .

We also achieved the fact that entering a number in the DAC data register now affects the voltage on a certain leg in our project, we also looped the values ​​and got some signal that changes regularly in time and tracked this change on a self-made oscilloscope .

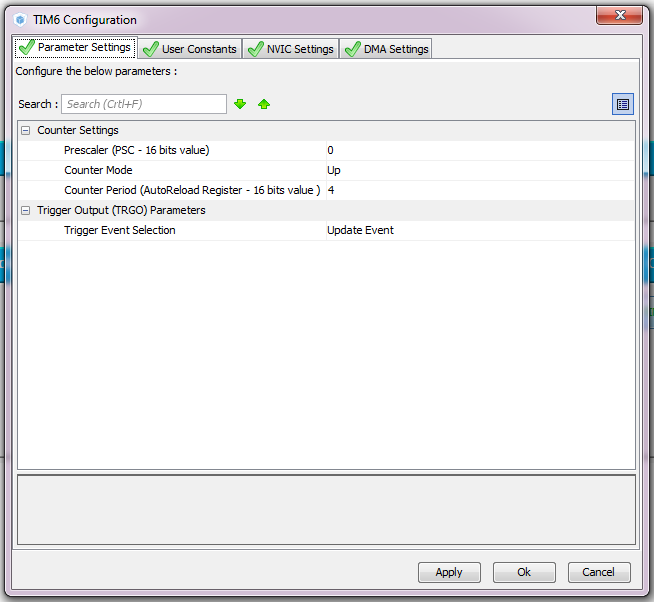
We had something similar to triangular impulses, but nevertheless they were not quite them. Therefore, today we will try to display on our oscilloscope's screen more real triangular pulses. For this we use a trigger, as well as hardware generation of triangular pulses in the periphery of the DAC.

So let's move on to the project.

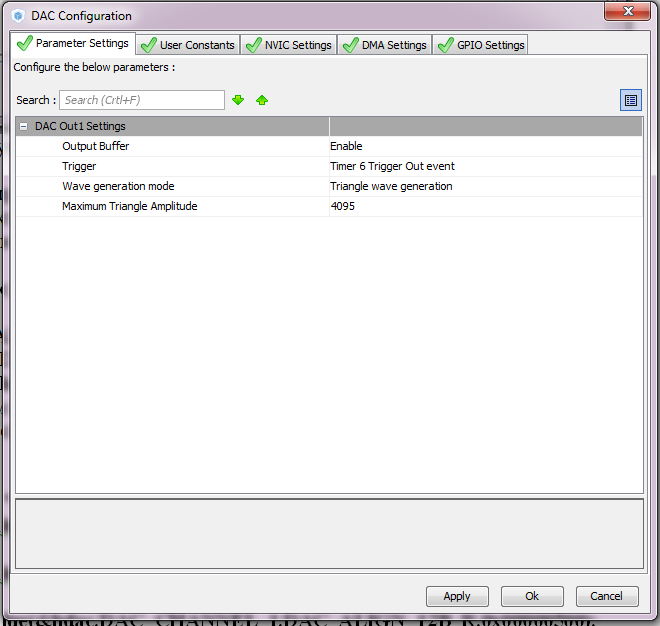
We create the project from the DAC project, because there is nothing extra is not connected. Let's call it DAC\_ TRIANGLE. Run the project in the Cube, turn on the timer 6



In Clock Configuration do not touch anything, go to the timer settings



In the DAC settings, the following



Nothing else is included, interrupts and DMA are not needed.

Generate and run the project. We'll collect the code, set up the programmer and start writing.

We remove all unnecessary from the project, the delay function in microseconds will be commented out so that there are no unnecessary warnings. From an infinite loop, we also remove everything. It remains only this

/ \* USER CODE BEGIN 2 \* /

        HAL\_DAC\_Start (& hdac, DAC\_CHANNEL\_1);

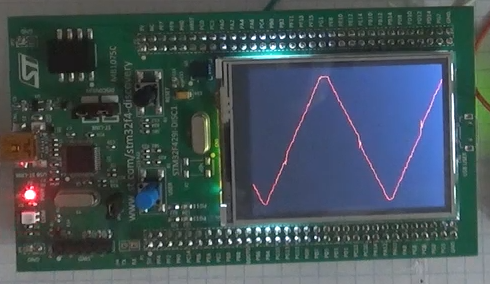
/ \* USER CODE END 2 \* /

Before starting the DAC, you will also need to start the timer

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

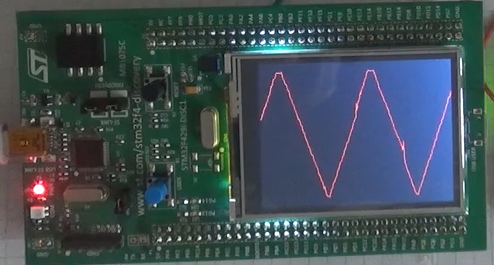
        HAL\_DAC\_Start (& hdac, DAC\_CHANNEL\_1);

We will collect, we will sew and we will look at the result in a self-made oscillograph



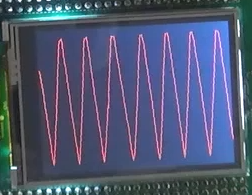
Increase the frequency by entering the timer values ​​in Cuba 1 instead of 4, without forgetting, of course, before closing the project.

Generate, collect, sew, look



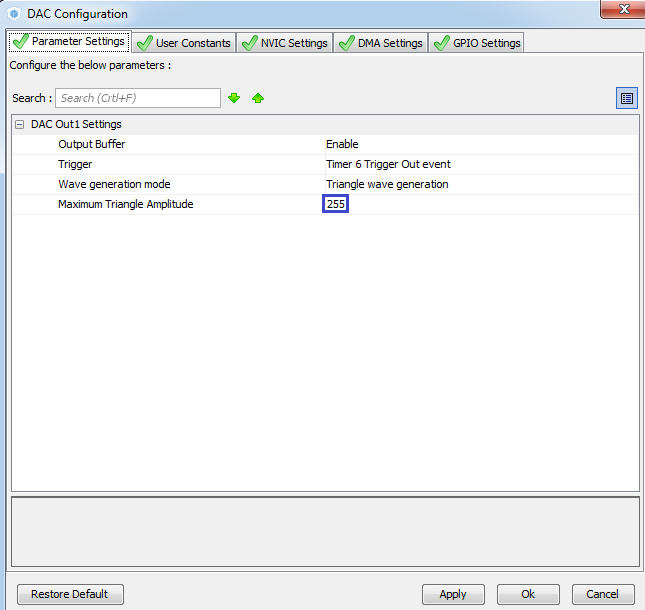
Now the only thing we can do to increase the frequency, in Clock Configuration, increase the frequency on APB1, to which our timer is connected twice.

Generate, collect, sew, look

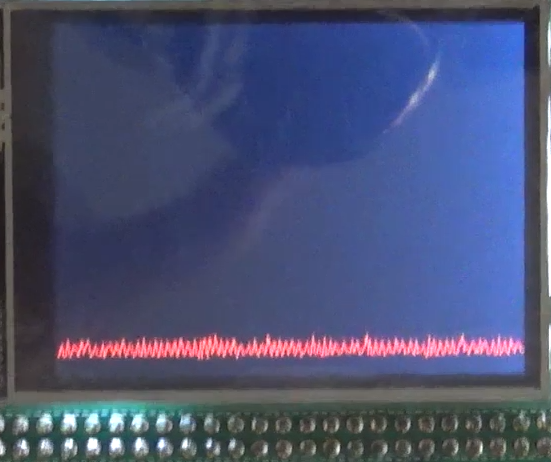


Now the generator operates at maximum frequency.

With a resolution of 12 bits of a higher frequency, we will not achieve, except that with only a lower resolution. We'll try to expose 255. This, however, is not the resolution, but the number to which the timer will count, the resolution is likely to be 256 or 8 bits



Generate the project, open it in Keil, compile the code and patch the controller. We will have this picture



In the [**next lesson**](http://narodstream.ru/stm-urok-29-hal-dac-triangle-dma/) we will try to generate already hardware noise from the DAC.