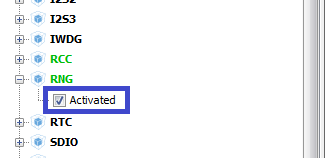
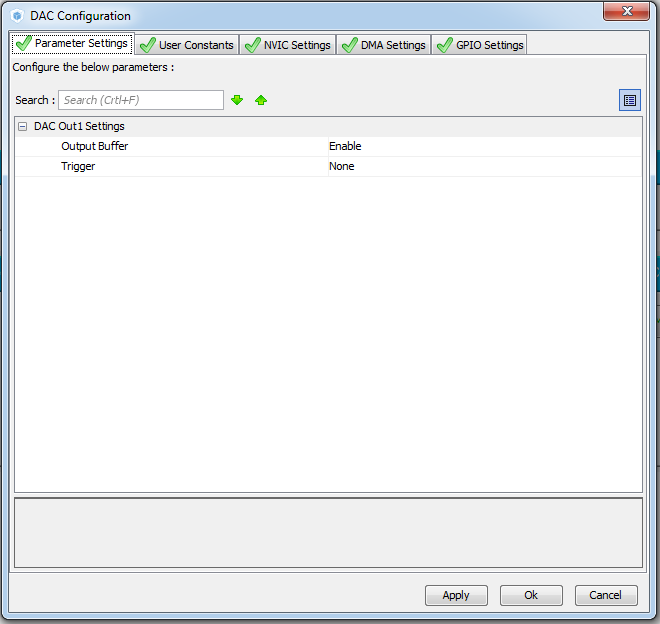
**DAC. Noise. RNG**

We create the project from the DAC\_NOISE project. Let's call it DAC\_NOISE\_RNG. Run the project in the Cube, turn on the RNG.



In Clock Configuration, we perform an automatic regeneration

In Configuration in DAC, turn off the hardware noise (trigger).



In the timer settings, enable the interrupts. Divisor and period do not touch.

Generate the project, open it, compile and configure the programmer.

In connection with the inclusion of timer interrupts, we will slightly change the function of its start in main ()

  / \* USER CODE BEGIN 2 \* /

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

We insert into main.c the interrupt handler at the end of the timer account

/ \* USER CODE BEGIN 4 \* /

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

**{**

**}**

/ \* USER CODE END 4 \* /

Let's start writing a function call in this handler.

In the HAL User manual, open page 661 and take there function **HAL\_RNG\_GetRandomNumber**

It will be like this

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

{

**HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, (HAL\_RNG\_GetRandomNumber (& hrng)) & 0x00000FFF);**

}

We will collect the project, we will impose the controller and we will look at the result. As we see, the shelves are completely different from us than with the use of a pseudo-random number, there are practically no repetitions.

Well, now we will do the same with frequencies as in the last lesson. You can not go into the Cube MX - right in Keil.

We first set the divisor and period values ​​to 41/2, and then 20/1.

Now you can flash the project of the last lesson with the same indicators and compare the result.

s