**Lesson 27**

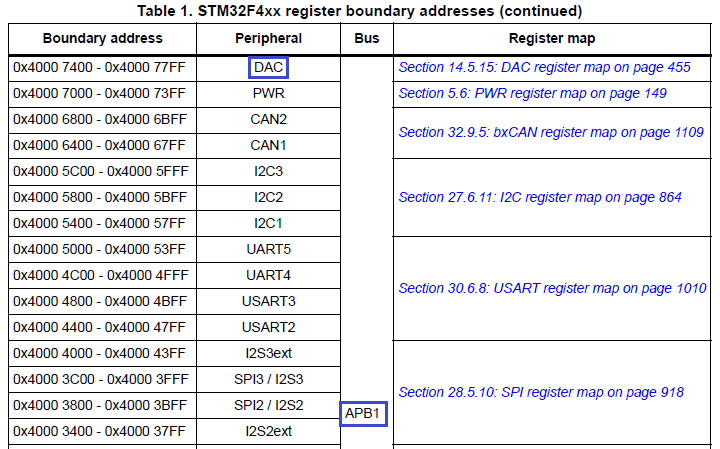
**HAL. DAC**

Today we will begin to study interesting technology - **digital-to-analog converter (DAC)** or English **Digital-to-analog converter (DAC)** .

As can be seen from the name of this converter, it is engaged in converting a digital code or value into a voltage of a certain value, somehow dependent on the given digital code. And also from the name it can be seen that the digital-to-analog converter is dealing with the problem, the inverse of the one that is being handled by the analog-to-digital converter (ADC), which we have already learned, starting with [**lesson 16**](http://narodstream.ru/stm-urok-16-hal-adc-regular-channel/) .

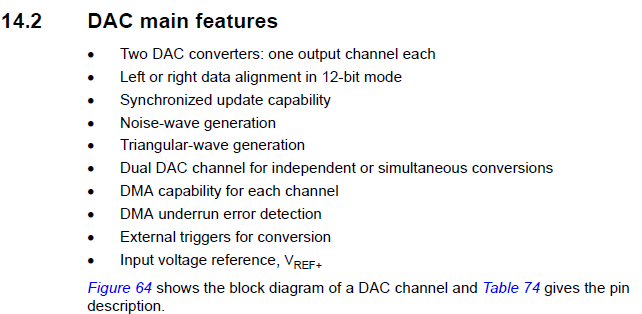
In general, too much detail we will not study the principle of the digital-to-analog converter. It is important for us to learn how to work with it using the STM32 controller.

First, open the Reference Manual and see which bus is controlled by this periphery



As we can see, this is the APB1 bus.

Also in this documentation we will look at the brief characteristics of our DAC



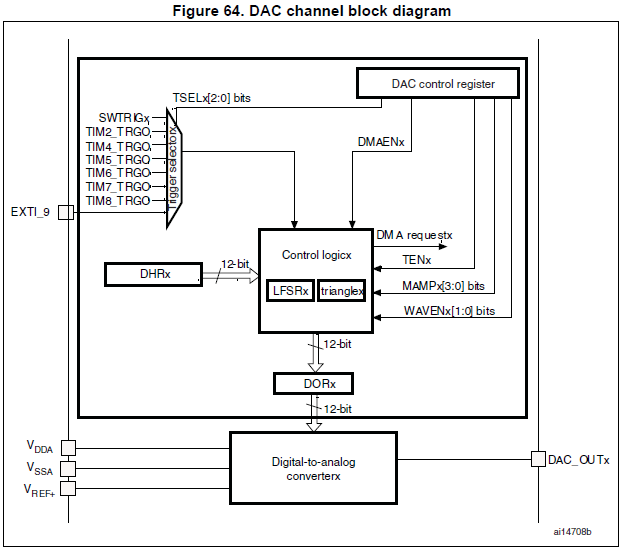
Our DAC is 12-bit.

We see that there are two such DACs, but there is also a remark here, which is not very encouraging, that for each such converter there is only one controller foot and it can not be redefined.

You can also align the data to the right or left edge of a double-byte DHRx register. That is, depending on the specific settings, we assign how we will locate the data in this register for subsequent conversion to an electrical signal.

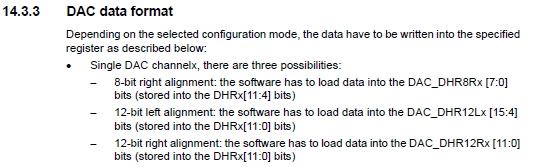
There is also the possibility of synchronous updates, noise generation, triangular pulses, separate channel conversion, and shared use, the use of DMA for each channel, the detection of DMA emptying errors, the use of external triggering conversion control, and the use of a reference voltage.

There is also a block diagram of the DAC



This diagram shows that there are a number of timers that we can use as triggers. We have a control register, the bits of which serve for certain settings, the DHRx data register, well, two of them - DHR1 and DHR2, which we use to store the value that will be converted into an electrical signal, and the numbers correspond to the numbers of the DACs. Well, and directly, the converter itself.

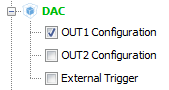
And here it is indicated which modes exist for conversion



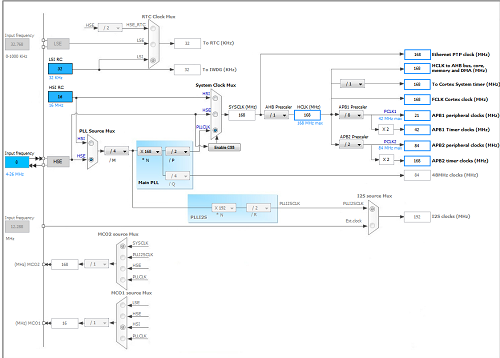
We see that there is also an 8-bit mode and two versions of the 12-bit mode, depending on the alignment of the converted value. We will not use these registers directly in the code, except that we will address them in the debugging process, if we do not get it working.

Well, let's get down to code at last.

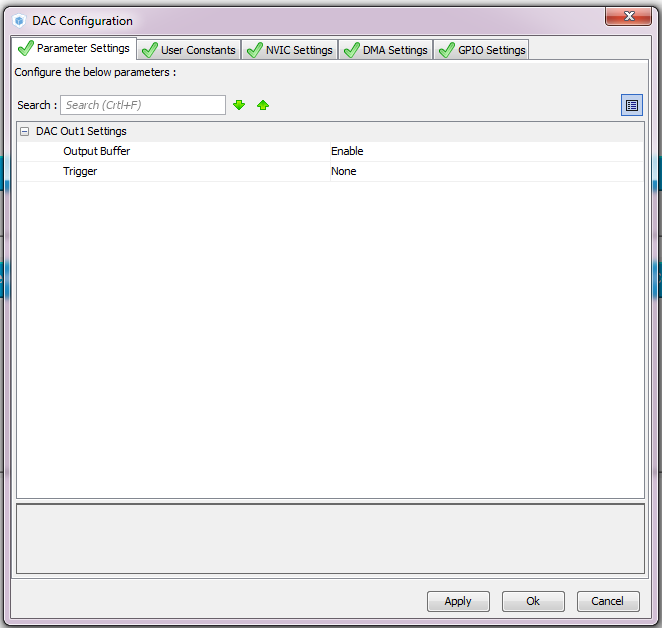
We create the project from the project TEST001, tk. there is nothing extra is not connected. Let's call it DAC. Run the project in the Cube, turn on there DAC OUT1



In Clock Configuration, we make the following settings

[](http://narodstream.ru/wp-content/uploads/2017/01/image00-2.png)

In the DAC settings, the following



Nothing else is included.

Generate and run the project. Let's assemble the code, set the programmer to auto-cut and start writing.

Start the DAC by finding the function in the HAL User Manual documentation on page 215

  / \* USER CODE BEGIN 2 \* /

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

  / \* USER CODE END 2 \* /

Let's try to bring in the DAC some value.

The value that must be recorded in the periphery of the DAC is calculated using the following formula

image07

Well, rather, this formula needs to be reversed, expressing from it DOR. This is the value that we use in the function.

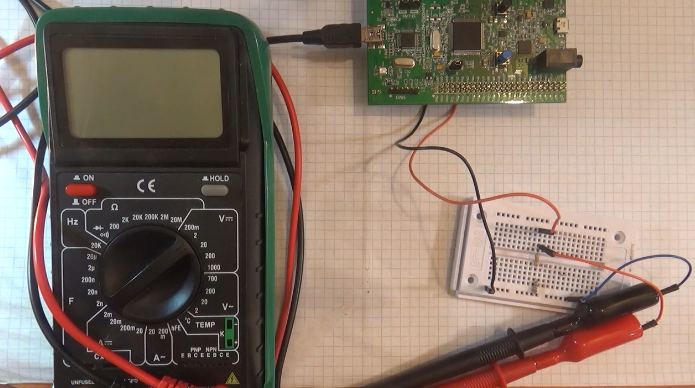
In general, for example, we need to generate a voltage at the output of the DAC with a value of 1 volt. Turning the above formula, we must divide 1 volt by the reference voltage or by 3 and then multiply by 4095. We get 1365.

Let's do this.

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

**HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, 1365);**

Before flashing the controller, let's see how we hooked up everything



From the foot PA4, corresponding to the output of our converter, I pulled the wire to the prototyping board, on which the voltage divider was assembled, in order to listen to the converted signal in the active columns. In the meantime, we connect the voltmeter to the leg PA4 and the common wire.

We collect, we sew and measure the tension on the foot PA4



The tension at us completely corresponds to the declared with very small error.

Let's write the following code in an infinite loop, which will gradually increase the output voltage

  while (1)

  {

  / \* USER CODE END WHILE \* /

**HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, 0x00000000);**

**HAL\_Delay (2);**

**HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, 0x00000280);**

**HAL\_Delay (2);**

**HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, 0x00000500);**

**HAL\_Delay (2);**

**HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, 0x00000780);**

**HAL\_Delay (2);**

**HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, 0x00000999);**

**HAL\_Delay (2);**

**HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, 0x00000780);**

**HAL\_Delay (2);**

**HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, 0x00000500);**

**HAL\_Delay (2);**

**HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, 0x00000280);**

**HAL\_Delay (2);**

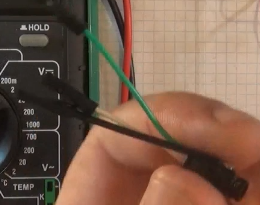
We collect, sew, listen in the columns. In order for you to hear this, watch the video version of the lesson.

Let's try to double the frequency by reducing the delay to 1 millisecond in all 8 locations

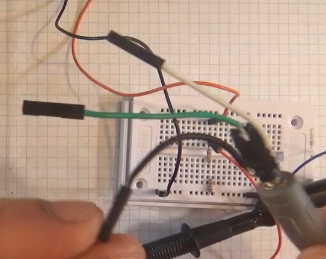
                HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, 0x00000280);

                HAL\_Delay ( **1**);

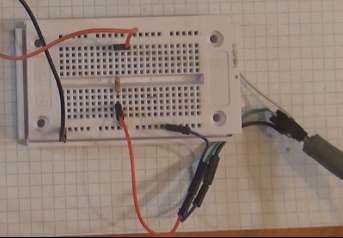
In order to connect the active speakers, I soldered such an uncomplicated connector-adapter



Connect it to the speaker wire



On the common wire of the speakers we connect the common wire from the breadboard, and on the active, for example, right channel of the amplifier of the active speakers, we will supply voltage from the divider. The divider is compiled with the calculation of 1 to 10, that is, the maximum voltage from it will be removed to a maximum of 0.3 volts, which is enough for us to hear the sound in the loudspeaker channel.



We collect the code, we flash the controller, we listen to the sound in the columns.

To further increase the frequency, we will not have enough delay function in milliseconds. It is necessary to write a delay function in microseconds well, or it is easier to take it from the project lcd.c file on the adapter with I2C to the liquid crystal display 20x4

/ \* USER CODE BEGIN 0 \* /

**\_\_STATIC\_INLINE void DelayMicro (\_\_ IO uint32\_t micros)**

**{**

**micros \* = (SystemCoreClock / 1000000) / 5;**

**while (micros-);**

**}**

/ \* USER CODE END 0 \* /

First, check this function by entering instead of the standard delay functions HAL\_Delay ( **1**) in all eight places of DelayMicro (1000)

                HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, 0x00000999);

**DelayMicro (1000);**

We sewed and listen again.

Now let's try to double the frequency by writing again in 8 places

                HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, 0x00000999);

**DelayMicro (500);**

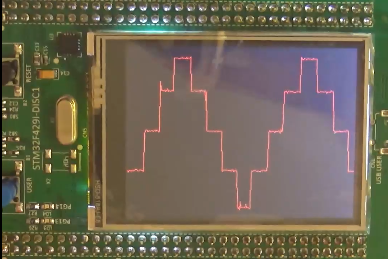
We will gather, sew and listen.

Increase five times the frequency

                HAL\_DAC\_SetValue (& hdac, DAC\_CHANNEL\_1, DAC\_ALIGN\_12B\_R, 0x00000999);

**DelayMicro (100);**

We will collect, we will sew, we will listen and we will look at the self-made oscillograph organized on a board 429



In the [next lesson,](http://narodstream.ru/stm-urok-28-hal-dac-triangle/) we will try using the DAC to generate triangular pulses in a hardware fashion.