**Lesson 16**

**HAL. ADC. Regular Channel**

Today we are starting a series of lessons on programming an analog-to-digital converter, or, as it's shortly said, **ADC** .

I think everyone who is engaged in the progarmming of microcontrollers knows firsthand what an **ADC is** and what it does. And if anyone does not know, then look at my lessons on programming **AVR** controllers - there I'm very detailed about the **ADC** .

Here we will touch on the principle of the **ADC** operation very briefly.

An analog-to-digital converter is a converter that converts the magnitude of an electrical signal into a digital code.

Well and as always **STM** company did not stay aside and included hardware **ADC** support in all its controllers , and so it included that we will devote more than one lesson to this.

And we will start, as always, with the project.

We create the project from MYLCD80, we call it ADC\_REGULAR. Why exactly from this project, because this display we will use with you today in our lesson with the ADC to display the converted results of our conversions.

Start Cube, turn on ADC1 for example 5 input (IN5). We see that the foot for measuring the voltage will be PA5. Let's go first to Clock Configuration and make sure that everything is fine there, nothing is red. By the way for ADC1, if you look at the plate in Reference Manual, the APB2 bus answers, that is, this periphery will work at the maximum frequent 84 MHz.

Go to Configuration. Here we have a lot of parameters. Just in case, we'll first enable the interrupts, then go back to the parameters. And ... do not touch anything here, tk. we will use all the default settings today.But you, of course, immediately have to ask a question. So for what we came here then, if we do not touch anything. And for the fact that I'll tell you about the parameters anyway.

Now we will generate and run our project. We also add lcd.c. Let's assemble it.

Configure our programmer to not press the RESET button. This is done as follows. The Flash-> Configure Flash Tool menu, or the Option For Target-> Utilites tab, the Settings button and put the checkbox in "Reset and Run".

In the main function main, we will slightly reduce the dimension of the array str. We have enough and 9. Also, the variable i will be 16-bit.

  / \* USER CODE BEGIN 1 \* /

**char str [9] = {0};**

**uint16\_t i;**

  / \* USER CODE END 1 \* /

Open the manual STM32F4\_HAL\_User Manual. On page 108 there, the entire process of starting the transformations, waiting for the result and stopping the transformations is written.

Let's find the function to start the conversion in the file **stm32f4xx\_hal\_adc.h (line 544)**. In an infinite loop we write

  while (1)

  {

**HAL\_ADC\_Start (& hadc1);**

Then we need to wait until the end of the transformation.

        HAL\_ADC\_Start (& hadc1);

**HAL\_ADC\_PollForConversion (& hadc1,100);**

We take the result and store it in a variable

        HAL\_ADC\_PollForConversion (& hadc1,100);

**i = (uint32\_t) HAL\_ADC\_GetValue (& hadc1);**

We stop the transformations

        i = (uint32\_t) HAL\_ADC\_GetValue (& hadc1);

**HAL\_ADC\_Stop (& hadc1);**

And we will display the result

        HAL\_ADC\_Stop (& hadc1);

**sprintf (str, "% 04d", i);**

**LCD\_SetPos (0, 3);**

**LCD\_String (str);**

Transformations are too frequent for us, so insert a small delay at the end of an infinite loop

        LCD\_String (str);

**HAL\_Delay (500);**

        / \* USER CODE END WHILE \* /

Let's sew and see the result of our actions with the code.

You can twist the resistor and see how the readings change.

But in this form it is not interesting for us to look at the testimony.

To do this, we first add a variable of another type instead of i.

~~uint16\_t i;~~

**float u;**

And change the code a little in an infinite loop.

~~i = (uint32\_t) HAL\_ADC\_GetValue (& hadc1);~~

**u = ((float) HAL\_ADC\_GetValue (& hadc1)) \* 3/4096;**

HAL\_ADC\_Stop (& hadc1);

~~sprintf (str, "% 04d", i);~~

**sprintf (str, "%. 2fv", u);**

Stitch and watch. Now it's another matter!