**Lesson 17**

**HAL. ADC. Regular Channel. Interrupt**

We create the project from **ADC\_REGULAR** , we call it **ADC\_REGULAR\_INT** .

Run the Cube, we'll check just in case the interrupts are turned on.

We generate and launch our project. We also add lcd.c. Let's assemble it.

There, on page 108 of the HAL\_User\_Manual datasheet, you can see how the code is written for the ADC using interrupts. Add the start function to an infinite loop.

        LCD\_String (str);

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

  / \* USER CODE END 2 \* /

And in an endless loop, you can still comment out.

At the beginning of the main module, we add a variable to store the result of the transformation

ADC\_HandleTypeDef hadc1;

/ \* USER CODE BEGIN PV \* /

/ \* Private variables ------------------- \* /

**uint16\_t ADC\_Data = 0;**

/ \* USER CODE END PV \* /

Near the end of the main module, add the function

/ \* USER CODE BEGIN 4 \* /

**void HAL\_ADC\_ConvCpltCallback (ADC\_HandleTypeDef \* hadc1)**

**{**

**ADC\_Data = HAL\_ADC\_GetValue (hadc1);**

**}**

/ \* USER CODE END 4 \* /

You can of course insert this into the interrupt handler, but it's better not to. Library developers advise us exactly this function. It is not performed with any interrupt, but only when the conversion is completed.

You also need to start the conversion in the same function again, we do not have the automatic mode.

void HAL\_ADC\_ConvCpltCallback (ADC\_HandleTypeDef \* hadc1)

{

          ADC\_Data = HAL\_ADC\_GetValue (hadc1);

**HAL\_ADC\_Start\_IT (hadc1); // run the analog-to-digital conversion**

}

Well, let's still display again our data on the display

  while (1)

  {

// sprintf (str, "%. 2fv", u); // convert the result to a string

                sprintf (str, "% 04d", ADC\_Data); // convert the result to a string

                LCD\_SetPos (0,3); // show the result on the LCD display

                LCD\_String (str);

                HAL\_Delay (200); // delay before the next cycle

  / \* USER CODE END WHILE \* /

We will collect the project, we will sew the controller and we will look.

At us conversions occur only once, at start or at all does not work anything.

It may seem that we have not completed something, but that's not the point. In the interrupt handler of the ADC, a function is called

  / \* USER CODE END ADC\_IRQn 0 \* /

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

  / \* USER CODE BEGIN ADC\_IRQn 1 \* /

Let's go into it and see its code. There is a lot of code there and it's just that on the frequency that we configured, physically can not be fulfilled, that's why we do not have a transformation.

Let's set up the divisor in the cube. Instead of 2 we put 8.

We will collect the project, we will sew the controller and see, that all earned. Let's now reduce the divisor a little.

We will make 4.

Again, we collect everything and sew it

It seems that it works too. Thus, you just need to play with frequency and understand what will work.

Now let's turn the readings on the display into a readable view similar to the last one

  while (1)

  {

**u = ((float) ADC\_Data) \* 3/4096;**

**sprintf (str, "%. 2fv", u); // convert the result to a string**

Now the matter is different.

Let's go into Configuration and enable Continuous Conversion Mode

Again, generate the project and collect the code

Now the line in the handler must be removed

void HAL\_ADC\_ConvCpltCallback (ADC\_HandleTypeDef \* hadc1)

{

          ADC\_Data = HAL\_ADC\_GetValue (hadc1);

~~HAL\_ADC\_Start\_IT (hadc1); // run the analog-to-digital conversion~~

}

We will collect, We sew and we will see

Everything is working.