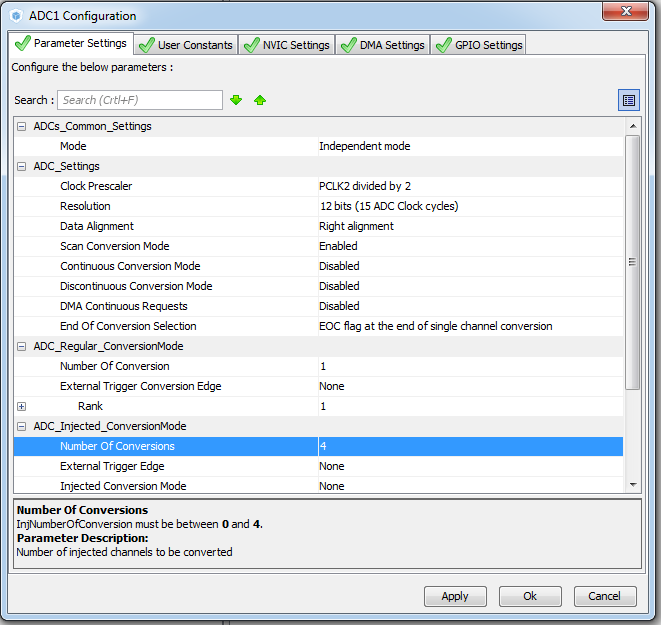
**HAL. ADC Injected Channel**

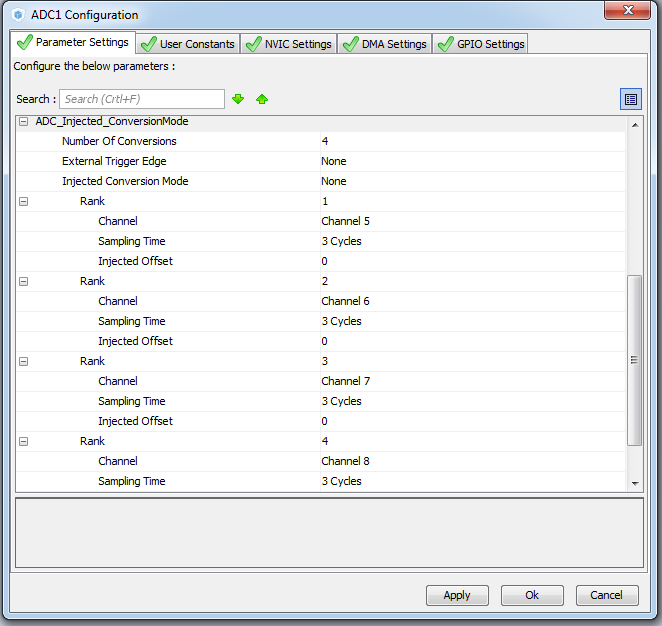
We create the project from **ADC\_REGULAR\_DMA** , we call it **ADC\_INJECTED** .

Start the **Cube** . Go to **Configuration** . We will disconnect there everywhere **DMA** .

Also we will disconnect interrupts from **ADC** , if those are included. We do not need them yet. This is the topic of the next lesson.

Set up there like this





Generate and run the project.

As always, configure the programmer and connect the file lcd.h

Delete all **cnt** counters  from anywhere, including **SysTick**

~~volatile uint32\_t cnt1, cnt2;~~

/ \* USER CODE END PV \* /

/ \* USER CODE BEGIN 2 \* /

~~cnt1 = 0; cnt2 = 0;~~

~~sprintf (str, "% 10u", cnt1); // convert the result to a string~~

~~LCD\_SetPos (6.0); // show the result on the LCD display~~

~~LCD\_String (str);~~

~~sprintf (str, "% 10u", cnt2);~~

~~LCD\_SetPos (6.1); // show the result on the LCD display~~

~~LCD\_String (str);~~

~~cnt1 ++;~~

// ADC\_Data [0] = HAL\_ADC\_GetValue (hadc1);

/ \* USER CODE BEGIN 0 \* /

~~extern volatile uint32\_t cnt2;~~

/ \* USER CODE END 0 \* /

void SysTick\_Handler (void)

{

  / \* USER CODE BEGIN SysTick\_IRQn 0 \* /

~~cnt2 ++;~~

  / \* USER CODE END SysTick\_IRQn 0 \* /

We also remove the ADC from DMA

        LCD\_Clear ();

~~HAL\_ADC\_Start\_DMA (& hadc1, (uint32\_t \*) & ADC\_Data, 4);~~

  / \* USER CODE END 2 \* /

We also remove the for loop from an infinite loop, leaving the contents of the loop

All injector functions are now not on the 108 page, but on the 126

We write in an infinite cycle

  while (1)

  {

                HAL\_ADCEx\_InjectedStart (& hadc1); // run the analog-to-digital conversion

                HAL\_ADC\_PollForConversion (& hadc1,100); // wait for the end of the transformations

                u [0] = ((float) HAL\_ADCEx\_InjectedGetValue (& hadc1, ADC\_INJECTED\_RANK\_1)) \* 3/4096; // enter the result of the transformations into a variable

                u [1] = ((float) HAL\_ADCEx\_InjectedGetValue (& hadc1, ADC\_INJECTED\_RANK\_2)) \* 3/4096; // enter the result of the transformations into a variable

                u [2] = ((float) HAL\_ADCEx\_InjectedGetValue (& hadc1, ADC\_INJECTED\_RANK\_3)) \* 3/4096; // enter the result of the transformations into a variable

                u [3] = ((float) HAL\_ADCEx\_InjectedGetValue (& hadc1, ADC\_INJECTED\_RANK\_4)) \* 3/4096; // enter the result of the transformations into a variable

Add a variable to the main () for the string. And the existing little increase. The counter-variable i can be removed

  / \* USER CODE BEGIN 1 \* /

        float u [4];

**char str [21];**

**char str1 [9];**

  / \* USER CODE END 1 \* /

We add the following code to an infinite loop

**u [2] = ((float) HAL\_ADCEx\_InjectedGetValue (& hadc1, ADC\_INJECTED\_RANK\_3)) \* 3/4096; // enter the result of the transformations into a variable**

**u [3] = ((float) HAL\_ADCEx\_InjectedGetValue (& hadc1, ADC\_INJECTED\_RANK\_4)) \* 3/4096; // enter the result of the transformations into a variable**

**sprintf (str, "%. 2fv", u [0]); // convert the result to a string**

**sprintf (str1, "%. 2fv", u [1]); // convert the result to a string**

**strcat (str, str1);**

**sprintf (str1, "%. 2fv", u [2]); // convert the result to a string**

**strcat (str, str1);**

**sprintf (str1, "%. 2fv", u [3]); // convert the result to a string**

**strcat (str, str1);**

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

**LCD\_String (str);**

**HAL\_ADCEx\_InjectedStop (& hadc1); // stop the conversions**

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

  / \* USER CODE END WHILE \* /

But to use the strcat function, we'll connect the library

/ \* USER CODE BEGIN Includes \* /

#include "main.h"

**#include  <string.h>**

/ \* USER CODE END Includes \* /