5a laboratory work

Analog read (Analog to Digital Converter - ADC)

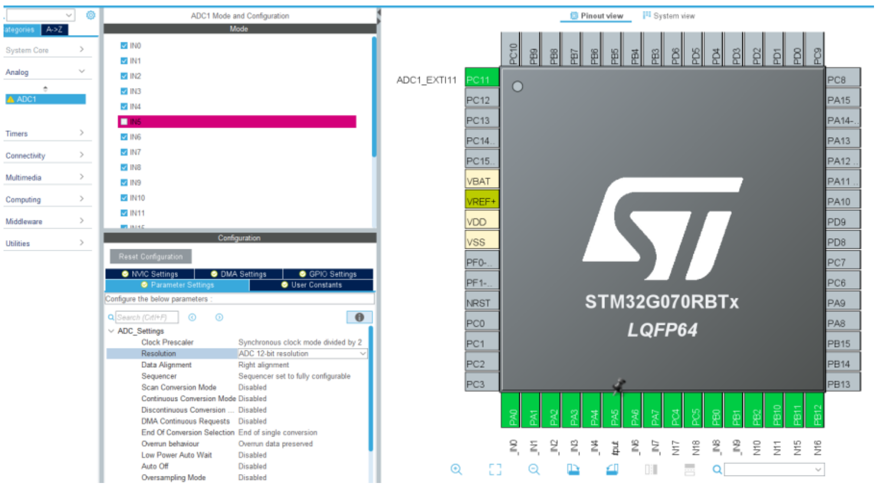
1. Aim

- Learn how to read signals using ADC (Analog to digital converter).
- Learn, how to calculate voltage from ADC values.

1. Theory

ADC converter is used for reading voltage signals from 0V to 3,3V and convert to digital numbers from 0-255 (if ADC resolution is 8 bit) or from 0 to 1023 (if ADC resolution is 10 bit) or from 0 to 4095 (if ADC resolution is 12 bit).

To start using ADC – you need to set up it in CubeMx IDE. Anf first you need to set certain pin of microcontroller as an Analog (ADC) input (Fig. 1).



Configure the below parameters :

ADC_Settings

Clock Prescaler	Synchronous clock mode divided by 2
Resolution	ADC 12-bit resolution
Data Alignment	Right alignment
Sequencer	Sequencer set to fully configurable
Scan Conversion Mode	Disabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion ...	Disabled
DMA Continuous Requests	Disabled
End Of Conversion Selection	End of single conversion
Overrun behaviour	Overrun data preserved
Low Power Auto Wait	Disabled
Auto Off	Disabled
Oversampling Mode	Disabled

Fig. 1 Setup Analog input pin of microcontroller

ADC has several channels; which means you can read several analog signals. When you select some channel, the microcontroller will reserve a dedicated pin for it. So, if you need to read the ADC value from Channel 0 you need to connect your device/sensor as shown in the image below.

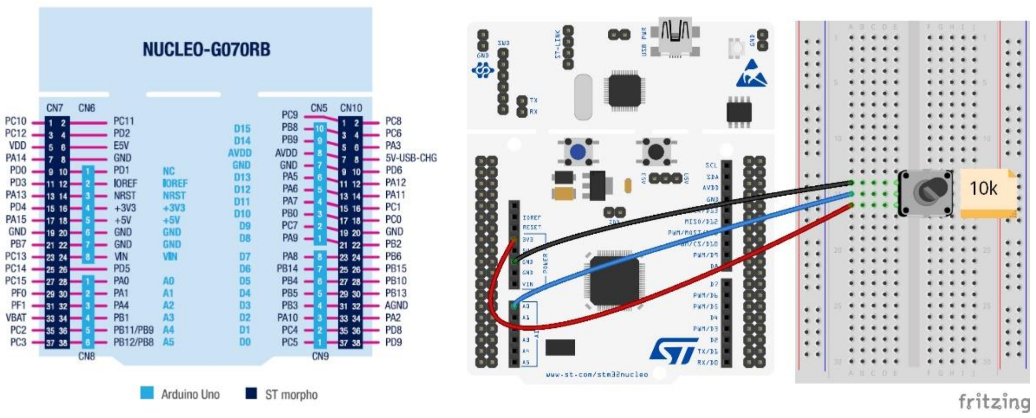
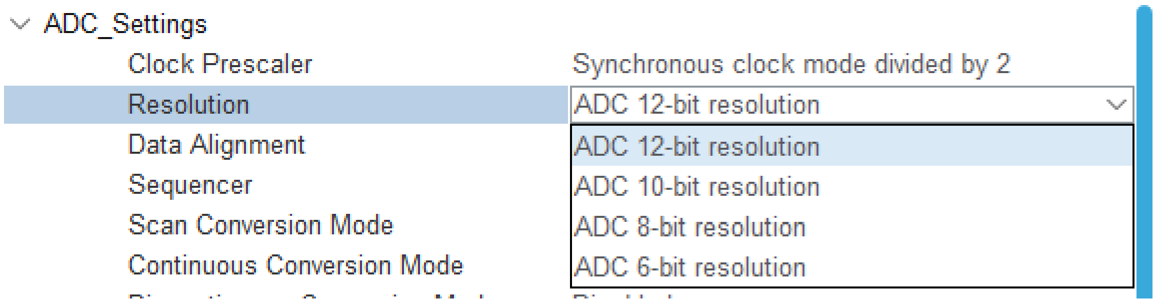


Fig. 2 Potentiometer connection to the microcontroller A0 input

From ADC settings you can also select – the resolution of ADC.



ADC_Settings

Clock Prescaler	Synchronous clock mode divided by 2
Resolution	ADC 12-bit resolution
Data Alignment	ADC 12-bit resolution
Sequencer	ADC 10-bit resolution
Scan Conversion Mode	ADC 8-bit resolution
Continuous Conversion Mode	ADC 6-bit resolution

Fig. 3 ADC resolution settings

If 12-bit ADC resolution is selected and the stm32 microcontroller is powered by 3.3, ADC has conversions that is represented on the picture below:

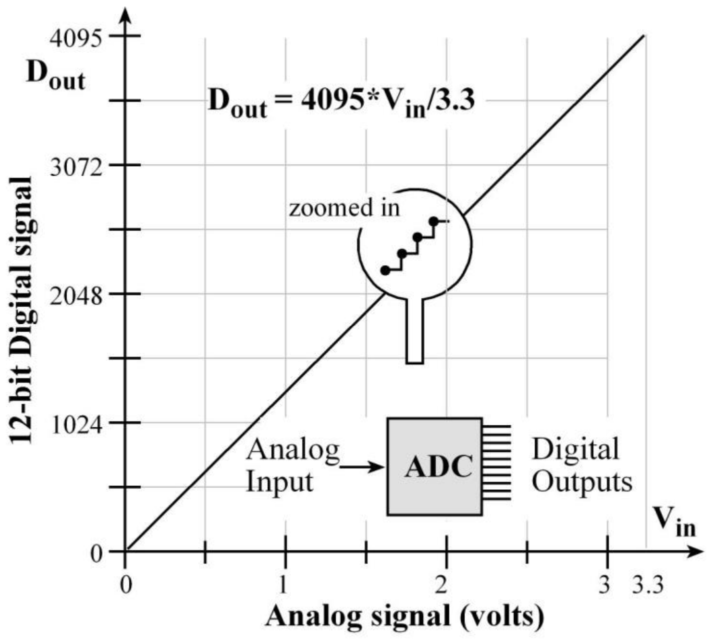
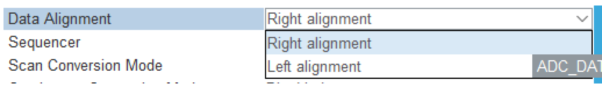


Fig. 4 Voltage (from 0-3.3V) conversion characteristics to digit (12 bit)

You can use adjustment of ADC value in two bytes if the resolution is 10 or 12 bit.



Data Alignment	Right alignment
Sequencer	Right alignment
Scan Conversion Mode	Left alignment

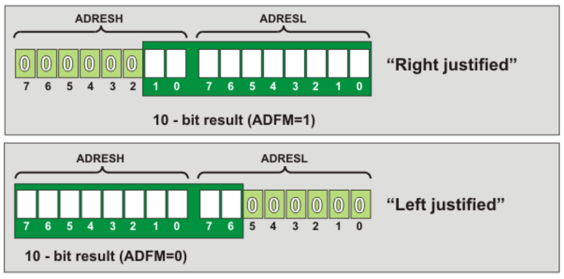


Fig. 5 ADC alignment settings and meaning

The ADC supports several conversion modes.

- **Single mode**, which converts only one channel, in Single-shot or Continuous mode.
- **Scan mode**, which converts a complete set of predefined programmed input channels, in Single-shot or Continuous mode.
- **Discontinuous mode**, converts only a single channel at each trigger signal from the list of pre-defined programmed input channels.

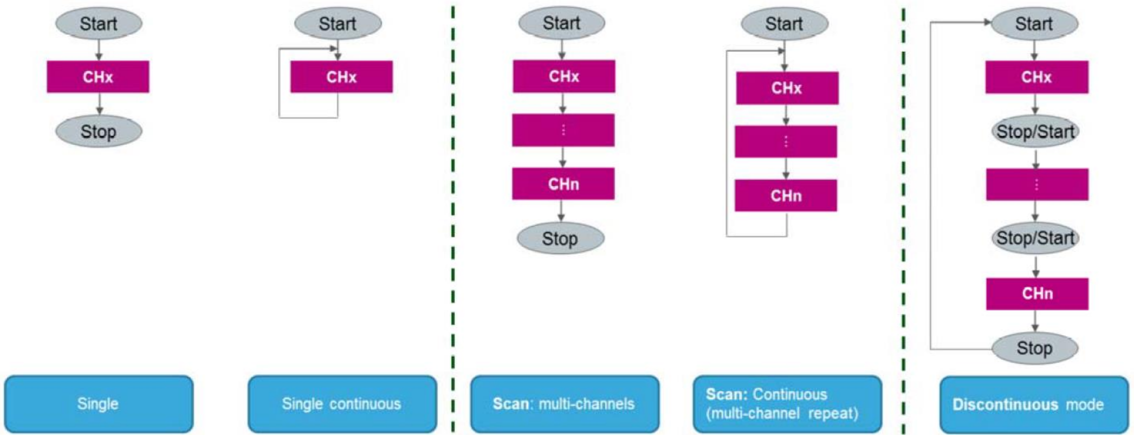


Fig. 6 ADC conversion modes

Single ADC read mode is done by starting ADC and then doing poll for conversion function:

```
  /* USER CODE BEGIN 2 */
  HAL_ADC_Start(&hadc1); // start the adc
  /* USER CODE END 2 */

/* USER CODE BEGIN WHILE */
  while (1)
  {
              HAL_ADC_PollForConversion(&hadc1, 100); // poll for conversion
              adc_val = HAL_ADC_GetValue(&hadc1); // get the adc value
              HAL_ADC_Stop(&hadc1); // stop adc
              HAL_Delay (500); // wait for 500ms
    /* USER CODE END WHILE */
    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

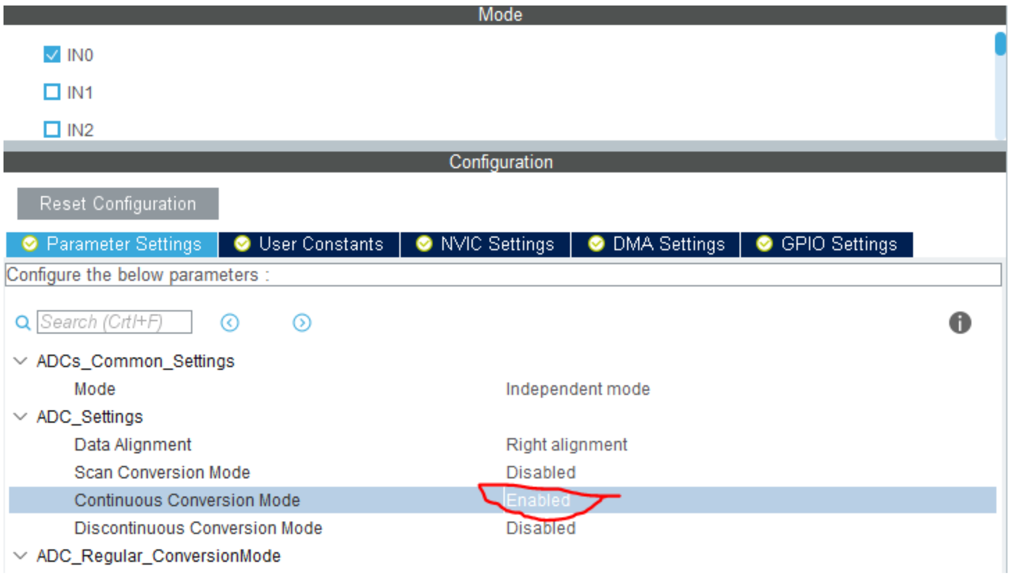
Expl. 1 ADC conversion using the poll method

For ADC value conversion to voltage, you need to take into account resolutions of ADC and ADC voltage.

```
HAL_ADC_PollForConversion(&hadc1, 100); // poll for conversion
adc_val = HAL_ADC_GetValue(&hadc1); // get the adc value
voltage = ((float) adc_val * 4096) / 3.3;
HAL_Delay (100); // wait for 500ms
HAL_ADC_Stop(&hadc1); // stop adc
```

Expl. 2 Recalculate ADC value to the voltage value

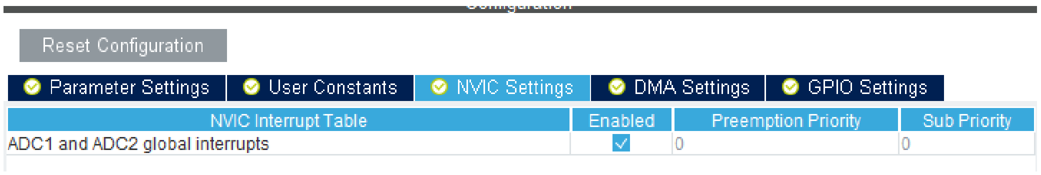
You can use continuous mode and ADC interrupt when conversion is complete. For this you need to select this mode from ADC settings menu.


Mode
☑ IN0
☐ IN1
☐ IN2
Configuration
Reset Configuration
Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings
Configure the below parameters :

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Data Alignment	Right alignment
Scan Conversion Mode	Disabled
Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled
ADC_Regular_ConversionMode	

Fig. 7 ADC continues mode enable

Then you need to setup NVIC.


Reset Configuration
Parameter Settings | User Constants | NVIC Settings | DMA Settings | GPIO Settings

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
ADC1 and ADC2 global interrupts	✓	0	0

Fig. 8 ADC conversion interrupt enable

In **main.c** code you need to use **`HAL_ADC_ConvCpltCallback`** function for do action when ADC read is complete.

```
/* USER CODE BEGIN Includes */
#include <stdio.h>
/* USER CODE END Includes */


/* USER CODE BEGIN PM */
uint16_t adc_value;
/* USER CODE END PM */


/* USER CODE BEGIN 2 */
HAL_ADC_Start_IT(hadc1);
/* USER CODE END 2 */

/* USER CODE BEGIN 4 */
 void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc1)
{
adc_value = HAL_ADC_GetValue(hadc1);
}
/* USER CODE END 4 */
```

Expl. 3 Callback function is called when ADC completes its conversion

2. Tasks

2.1. Write code for read voltage from ADC 1 channel and print out voltage into UART if it changes (increased/decreased) more than 1 V.

2.2. Write the code for read ADC value, turn on Green LED if value is less than 1V, turn on Yellow LED if value is more than 1V and less than 2V, turn on the RED LED if value is more than 2V.

2.3. Write a code to read ADC 20 values and calculate mean value. Please use interrupt for this task.

3. Report content

1) Title.
2) Main blocks of source code for tasks with comments.
3) Conclusions.

1. References

1. https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-mainstream-mcus/stm32g0-series/stm32g0x0-value-line/stm32g070rb.html