ADC Peripheral and Features

23 June 2025 11:50

- → It has up to 19 multiplexed channels allowing it to measure signals from 16 external sources, two internal sources, and the VBAT channel.
- → The A/D conversion of the channels can be performed in single, continuous, scan or discontinuous mode.
- → The result of the ADC is stored into a left- or right-aligned 16-bit data register.
- → 12-bit, 10-bit, 8-bit or 6-bit configurable resolution.
- → Scan mode for automatic conversion of channel 0 to channel ‘n’.
  Scan mode in ADC automatically reads multiple input channels (from channel 0 to channel ‘n’) one after the other in a single conversion cycle, without needing to trigger each one manually. It’s useful for reading multiple sensors efficiently.
- → ADC supply requirements: 2.4 V to 3.6 V at full speed and down to 1.8 V at slower speed.

ADC in stm32f429zi

23 June 2025 11:50

→ There are 3 12-bit ADCs available on this board.
→ 12 bit meaning, value ranges from 0 to 2095, these values represent reference vtg.
→ A step is the smallest vtg change, the ADC can detect.
It depends on the reference vtg and the resolution, i.e bit(12 bit or 8 bit).
→ The step size = (Vref) / (2^n -1);
Here Vref is the reference vtg(3.3V), n is the resolution(12 bit)
The reference vtg, is the maximum vtg the ADC can convert into a digital value.
→ The ADC is powered on by setting the ADON bit in the ADC_CR2 register.
→ Conversion starts when either the SWSTART or the JSWSTART bit is set.
→ The ADC features two clock schemes:
Clock for the analog circuitry: ADCCLK, common to all ADCs.
→ Clock for the digital interface (used for registers read/write access) This clock is equal to the APB2 clock.
→ For the STM32F42x and STM32F43x devices, the temperature sensor is internally connected to ADC1_IN18 channel which is shared with VBAT.
→ The internal reference voltage VREFINT is connected to ADC1_IN17.

ADC Configuration Needs

23 June 2025 14:55

- → The ADC is powered on by setting the ADON bit in the ADC_CR2 register.
- → Vdda is 3.3V by default.
- → We can check the maximum clk frequency by referring to the data sheet, table 74, pg 158, stm32f427vg..
- → In the table , it shows the max clk frequency is 36MHz.
- → Each channel is an input line that ADC can read.
- → A **sequence** is a **list of channels** that the ADC will convert **one after the other**, **automatically** (in scan mode).
  This is useful when you want to monitor **multiple analog signals** in one go.
- → The ADC has several operating modes, depending on your needs:

Use Case	**Mode(s) to Choose**	**Explanation**
Reading 1 analog value manually	Single Conversion	ADC converts one channel once when you tell it to
Reading 1 analog input repeatedly	Continuous Conversion	ADC keeps converting the same channel over and over
Reading multiple channels one after another	Scan Mode + Continuous/Single	ADC steps through your channel list automatically
Need non-blocking read or multiple results	Scan Mode + DMA	ADC does conversions and DMA stores the data in memory
Trigger conversion on external signal (like timer)	External Trigger Mode	ADC waits for a trigger to start conversion

- → Why is the data alignment needed, because we want to store a 12 bit value, in 16 bit register, So we either use left or right alignment for proper reading purpose.(refer 13.4 for data alignment).
- → **Sampling time** is needed to give the ADC's internal capacitor enough time to "capture" the input voltage.(sec->13.5, reference manual).
  Tconv = Sampling time + 12 cycles.
  Choose based on:
  - Source impedance
  - Desired accuracy
  - Speed trade-off

NOTE: In datasheet, the pinout diagram which says AD123_Inx means, that pin can be used by ADC1, ADC2, ADC3 by the channel Inx.

Temperature Sensor in ADC ch.18

24 June 2025 09:35

The temperature sensor can be used to measure the junction temperature (TJ) of the device.

- → VSENSE is input to ADC1_IN16 for the STM23F40x and STM32F41x devices and to ADC1_IN18 for the STM32F42x and STM32F43x devices.
- → Reading the temperature To use the sensor: 3. Select ADC1_IN16 or ADC1_IN18 input channel.
- → Select a sampling time greater than the minimum sampling time specified in the datasheet.
- → Set the TSVREFE bit in the ADC_CCR register to wake up the temperature sensor from power down mode
- → Start the ADC conversion by setting the SWSTART bit (or by external trigger)
- → Read the resulting VSENSE data in the ADC data register
- → Calculate the temperature using the following formula:
- → Temperature (in °C) = {(VSENSE – V25) / Avg_Slope} + 25
  Where:– V25 = VSENSE value for 25° C
  Avg_Slope = average slope of the temperature vs. VSENSE curve (given in mV/°C or µV/°C)
  Refer to the datasheet's electrical characteristics section for the actual values of V25 and Avg_Slope.

Code to check just high and Low

25 June 2025 10:47

NOTE : Code has not been tested. And written in Keil, to check the output we can add the ADC_VAL variable to watch window.

```
#include "stm32f429xx.h"


void gpio_init()
{
	//Enable the clocks, for gpio port A
	RCC->AHB1ENR |=RCC_AHB1ENR_GPIOAEN;

	//Enable the clk for ADC1
	RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;

	//set GPIO mode to analog
	GPIOA->MODER |= ( (3 << 1*2) | (3 << 0));

}
void adc_init()
{
	//set prescaler to 4
	ADC->CCR &= ~(ADC_CCR_ADCPRE);
	ADC->CCR |= (1<<16);//moving 01 to 16th bit

	//Set the scan mode since we are using 2 channels
	ADC1->CR1 |= ADC_CR1_SCAN;
	//Set res bit is 12
	ADC1->CR1 |=ADC_CR1_RES;


	//Enable the ADC on.
	ADC1->CR2 |= ADC_CR2_ADON;
	//Continous Conversion bit is set
	ADC1->CR2 |= ADC_CR2_CONT;
	//Data alignment is set to right
	ADC1->CR2 &=~(1<<11);
	//set EOC after each conversion
	ADC1->CR2 |= ADC_CR2_EOCS;

	//clear L[3:0]
	ADC1->SQR1 &= ~(0xF<<20);
	//Set sequence length as 2(01), 2 channels
	ADC1->SQR1 |= 1<<20;


	//sampling rate as 3 for both channels.
	ADC1->SMPR2 &= ~((3<<0)| (3<<3));


}
```

```
uint16_t get_val(int channel)
{
        ADC1->SQR3 = 0;
    ADC1->SQR3 |= (channel << 0);//setting sequence
    ADC1->CR2 |= ADC_CR2_SWSTART; // Start conversion

    //reset the data register before conversion
    ADC1->SR=0;

    //Wait until conversion is complete
    while(!(ADC1->CR2 & ADC_SR_EOC));

    //return the value stored on the data register
    return ADC1->DR;

}
uint16_t ADC_VAL[2]={0,0};
int main()
{
    gpio_init();
    adc_init();

    while(1)
    {
        ADC_VAL[0]=get_val(0);
        ADC_VAL[1]=get_val(1);
    }


}
```

Code for inbuilt temp sensor

25 June 2025 10:48

NOTE : This sensor tells the temp around the chip. Code has not been tested.

```
#include "stm32f429xx.h"

void init_config()
{
	//Enable the clock for ADC1
	RCC->APB2ENR |= RCC_APB2ENR_ADC1EN;

	//clear the bit before setting it.
	ADC->CCR &= ~(ADC_CCR_TSVREFE);
	ADC->CCR |= (ADC_CCR_TSVREFE);

	//set alignment to right
	ADC1->CR2 &= ~(1<<11);

	//setting sample rate to 480 cycles.
	//(mentioned in reference manual to set higher sample rate)
	ADC1->SMPR1 |= (7<<24);

	//setting sequence length to 1
	ADC1->SQR1 &= ~(0xF << 20);

	//set first conversion as channel 18
	ADC1->SQR3 = 18;

	//enable the ADC
	ADC1->CR2 |= ADC_CR2_ADON;

}
uint16_t read_temp()
{
	//start conversion
	ADC1->CR2 |= ADC_CR2_SWSTART;

	while(!(ADC1->SR & ADC_SR_EOC));
	return ADC1->DR;
}
float cal_temp(uint16_t adc_val)
{
	float vsense = (adc_val * 3.3)/4095;
	//The adc_val from 0-4095 is converted to vtg value, 3.3 is vref

	float v25=0.76;//from the datasheet
	float avg=0.0025;

	float temp = ((vsense-v25)/avg)+25;

	return temp;
}
```

```
int main()
{
	init_config();
	while(1)
	{
		uint16_t adc_temp=read_temp();
		float res = cal_temp(adc_temp);
	}

}
```