Лабораторна робота №7

ВИКОРИСТАННЯ АНАЛОГО-ЦИФРОВОГО ПЕРЕТВОРЮВАЧА В МІКРОКОНТРОЛЕРАХ STM32

Мета: отримати базові навички роботи з модулем АЦП. Програмна взаємодія та виконання прикладних задач в мікроконтролерах STM32.

Теоретичні відомості

Аналого-цифровий перетворювач (скор. АЦП, англ. Analog-to-digital converter, скор. ADC) – це пристрій або модуль, який перетворює аналогові сигнали в цифрові.

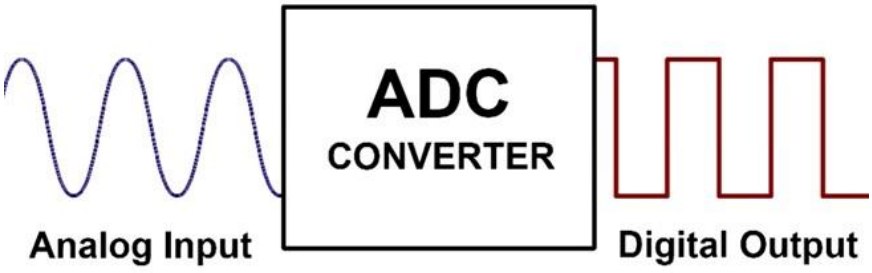


Рис.7.1 – Аналого-цифрове перетворення.

Аналогові сигнали це безперервні сигнали, амплітуда напруги яких плавно змінюється з часом. Цифрові сигнали — це сигнали з двома рівнями напруги (високий і низький), які можна представити в двійковому коді як 1 і 0. У цьому поданні сигнали можуть бути оброблені цифровими схемами для збору, відображення, зберігання та аналізу даних.

АЦП досить широко використовується: від вимірювання сигналів в лабораторному обладнанні до управління промисловими процесами. Аналого-цифрові перетворення є одним з основних елементів сучасних систем збору даних, моніторингу промислових процесів та моніторингу систем безпеки.

Модулі АЦП у зв'язці з іншими електронними модулями можуть використовуватися:

• В автоматичному керуванні технологічним обладнанням та транспортом, в автомобілях та літальних апаратах (у тому числі безпілотних).

• В електронних вимірювальних приладах.

• У бездротових системах для перетворення аналогових голосових або відеосигналів на цифрові, які можна надсилати через мережу.

• У ТВ-тюнерах та іншій відео техніці.

• У збройових системах управління та наведення.

Модулі АЦП входять до складу радіомодемів та інших пристроїв передачі даних. І нарешті вони входять до складу практично всіх мікроконтролерів.

У мікроконтролері **STM32** є **один АЦП — ADC1**, і він має **16 мультиплексованих каналів**. Які можна використовувати для вимірювання зовнішніх сигналів і два канали пов'язані з вбудованим датчиком температури та внутрішнім джерелом опорної напруги.

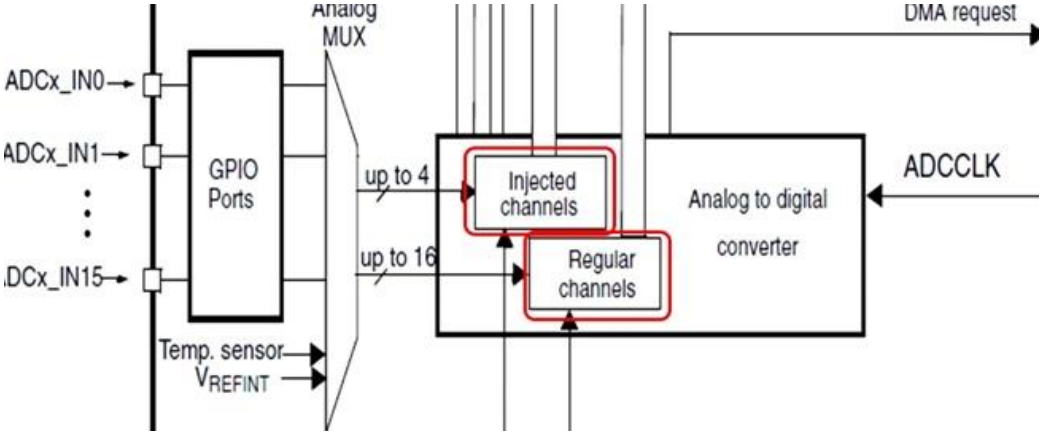
Аналого-цифрове перетворення різними каналами може виконуватися в одиночному, безперервному, скануючому або переривчастому режимах.

До інших можливостей АЦП належать:

- Самокалібрування.
- Генерація переривань наприкінці перетворення.

- Програмований час і періодичність вибірки.
- Запуск от зовнішнього сигналу або за командою від таймера.

Кожен АЦП підтримує два базові режими перетворення: **звичайний та інжектований**



У режимі **звичайних перетворень** визначається один канал або група каналів, які перетворюються по черзі. Можна встановити порядок перетворення каналів. Може бути задане багаторазове перетворення одного й того каналу.

Група **інжектованих перетворень** відрізняється підвищеним рівнем пріоритету. Після запуску цієї групи, зупиняється оцифрування у групі звичайних перетворень, виконується послідовність інжектованих перетворень, а потім відновлюється черговість перетворень у звичайній групі.

У STM32 існує три різні способи програмування перетворень.

1) **Метод опитування** – це самий простий метод, коли процесор мікроконтролера чекає, поки АЦП завершить всі перетворення, щоб він міг відновити обробку основного коду.

2) **Метод переривання** –це коли процесор не блокується і може продовжувати виконання основної процедури коду, а запуск АЦП та обробка аналогових сигналів здійснювався у фоновому режимі. І тільки після закінчення АЦП процесор може перейти на обробку результатів перетворення.

3) **Метод використання прямого доступу до пам'яті** (англ. DMA) при якому передача результатів роботи АЦП в пам'ять виконується без втручання процесора мікроконтролера і це є найбільш ефективним способом перетворення декількох каналів АЦП з дуже високою швидкістю.

Основний блок модуля АЦП – це **регістр послідовного наближення** (англ. скор. SAR).

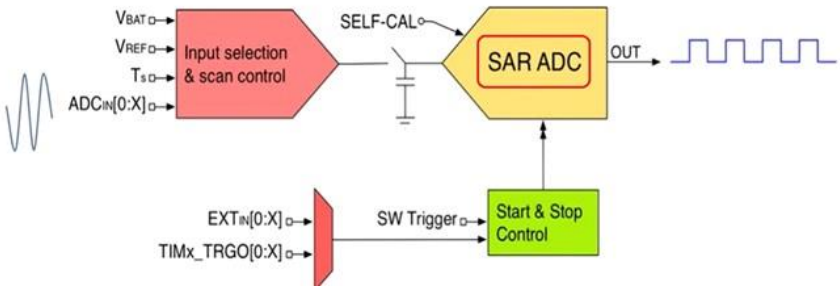


Рис.7.2 – Регістр послідовного наближення.

Тут виконуються такі процеси. Напруга вхідного сигналу обчислюється шляхом його порівняння з внутрішньою опорною напругою мікроконтролера за кілька етапів. На кожному наступному етапі рівень опорної напруги змінюється. Якщо вхідна напруга більша за поточний рівень опорної напруги, то опорна напруга на наступному кроці збільшується, а

якщо вхідна напруга менша – опорна напруга зменшується. Кількість кроків дорівнює кількості розрядів АЦП. Кожен крок видає на вихід один біт двійкового коду результату АЦП.

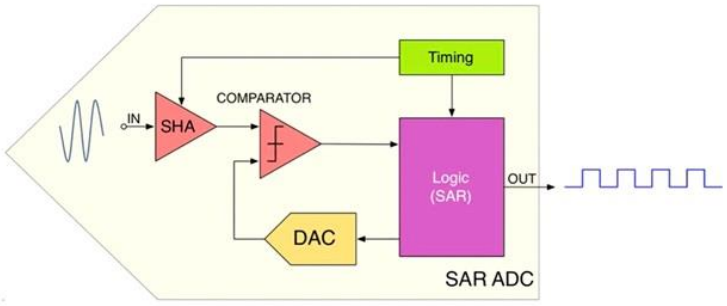
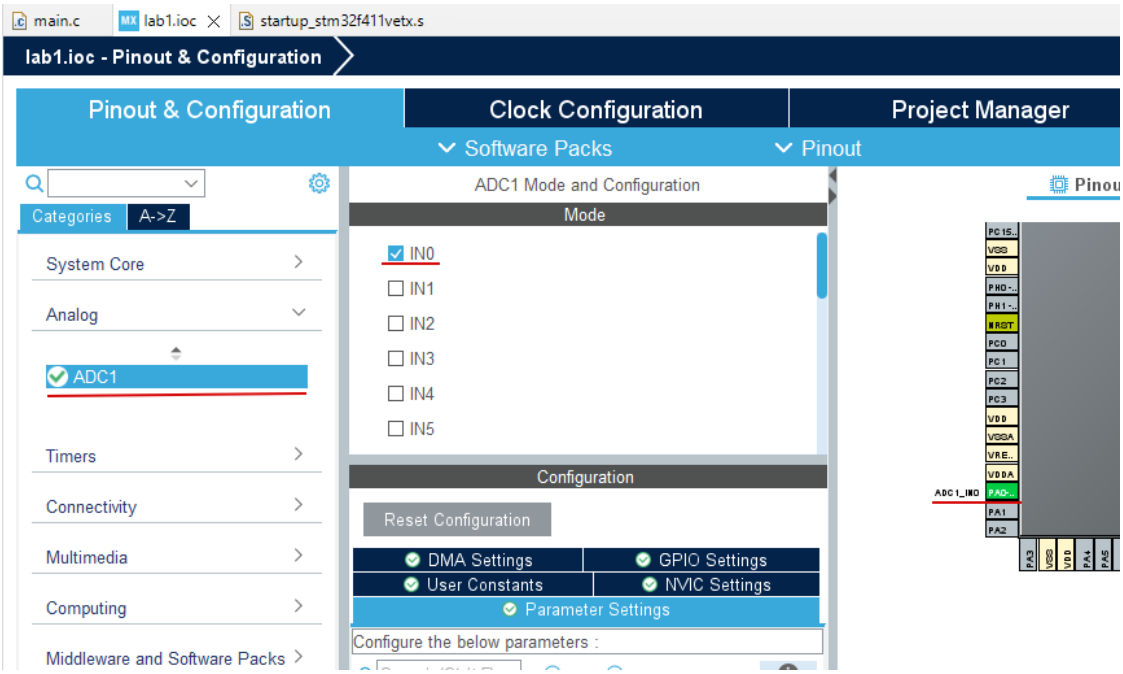


Рис.7.3 – Будова регістру послідовного наближення.

Вбудований АЦП в мікроконролері STM32 має 12 біт. Тому точність розрахунку буде вижчою. Напруга 3,3 вольта відповідає десятковому числу 4095 одиниць АЦП. Це означає, що 1 одиниця 12 бітного АЦП приблизно відповідає: 3,3/4095 = 0,0008 вольта.

Для налаштування роботи АЦП потрібно:

1.1.В розділі Analog обрати ADC1 та будь-який доступний канал.



1.2.Зберегти конфігурацію та згенерувати файл *main.c*.

1.3.Підключити бібліотеку для роботи з рядками та бібліотеку вводу/виводу:

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "string.h"
#include "stdio.h"
/* Private includes ----------------------------------------------------------*/
```

1.4.Оголосити глобальну змінну abc з початковим занчення 0:

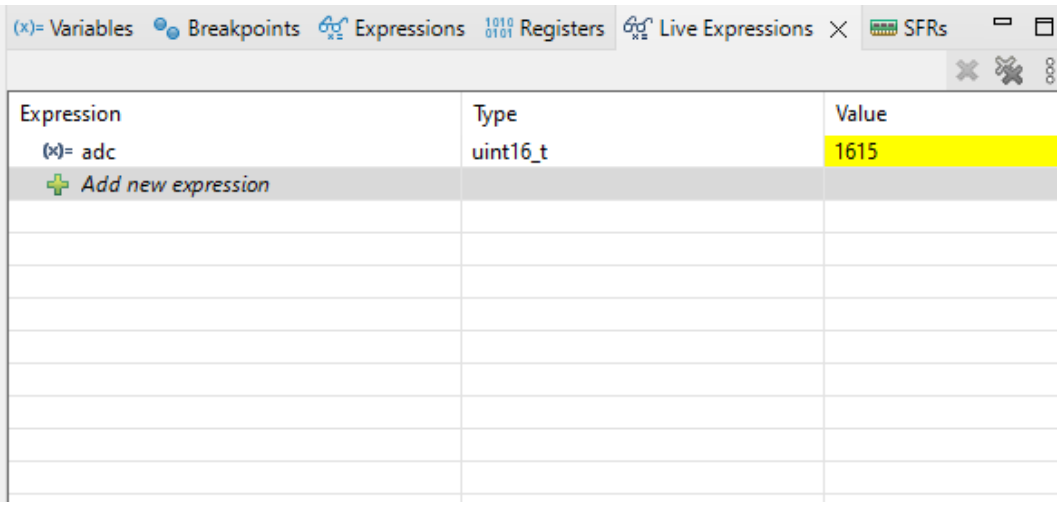
```
uint16_t adc = 0;
```

1.5.В головному циклі коду запустити процес зчитування з аналогового каналу інформації та запис її в змінну abc:

```
while (1)
```

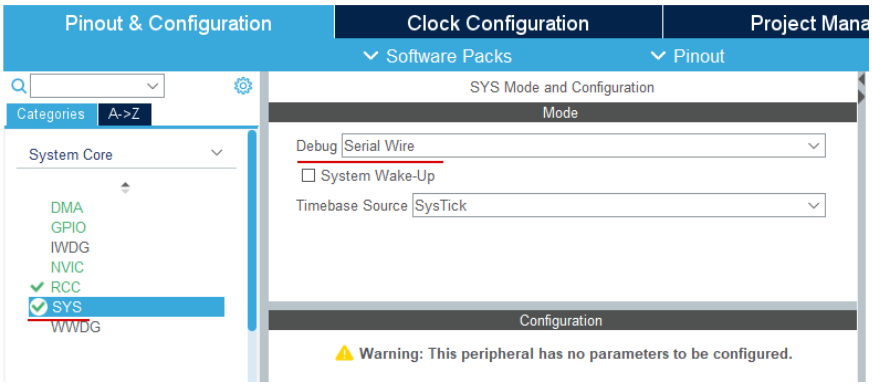
```
{
  /* USER CODE END WHILE */
        HAL_ADC_Start(&hadc1);
        adc = HAL_ADC_GetValue(&hadc1);
  /* USER CODE BEGIN 3 */
}
```

1.6. Підключити будь-який аналоговий пристрій для перевірки роботи АЦП та запустити в режимі дебагу. При правильному налаштуванні значення буде змінюватись під час взаємодії з аналоговим пристроєм:

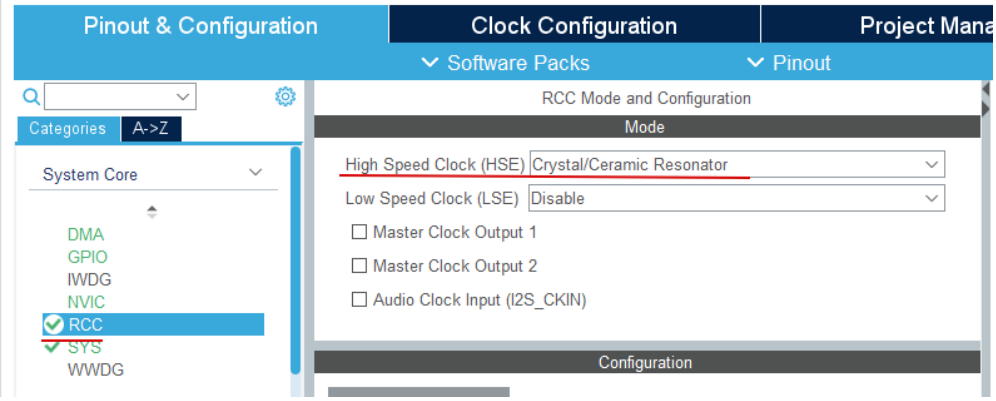


2 Отримання інформації про температуру з вбудованого внутрішнього термометра(Internal Temperature Sensor).

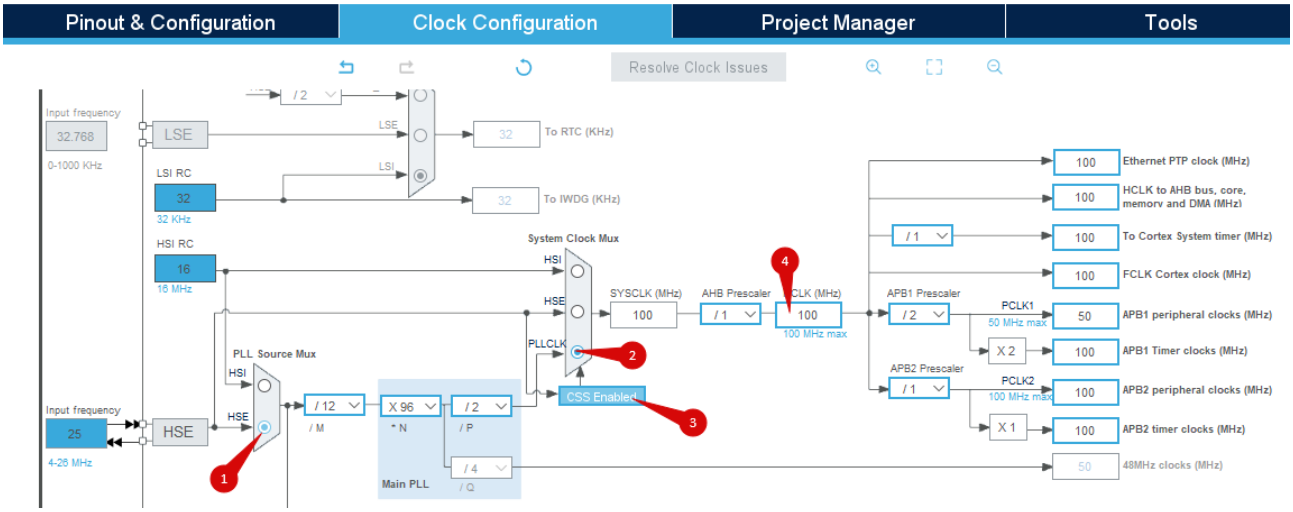
2.1 Активувати Debug в режим Serial Wire:



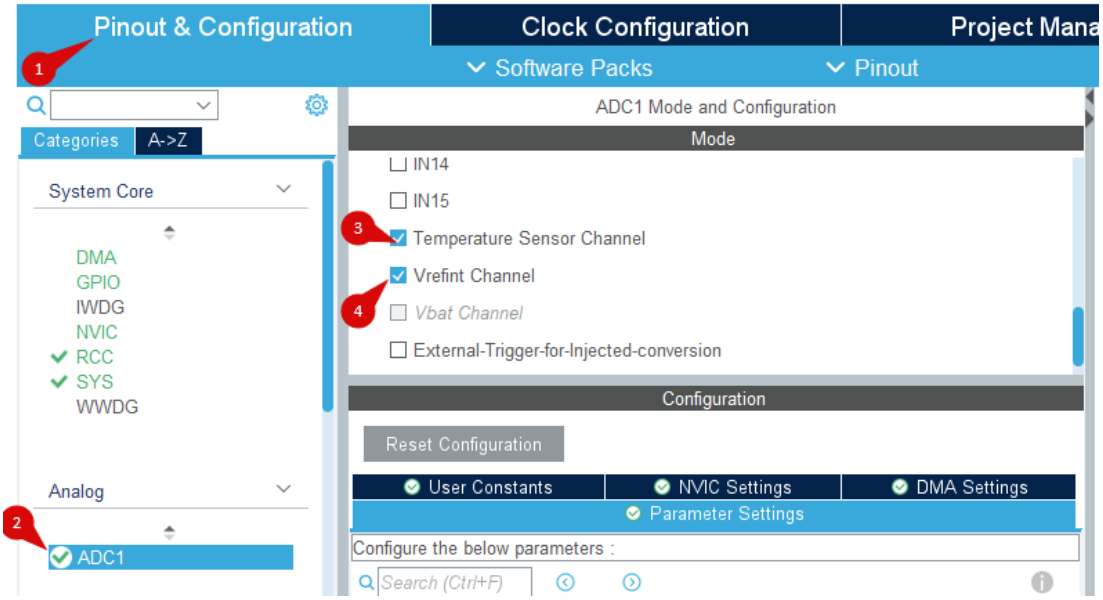
2.2 Увімкнути HSE(High Speed Clock) обравши Crystal/Ceramic Resonator:



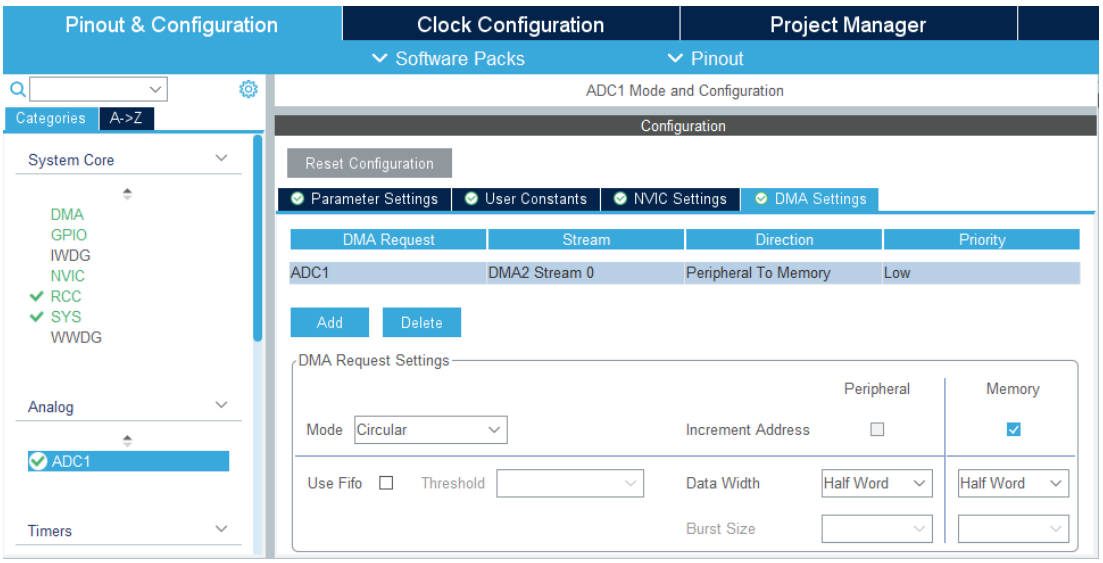
2.3 У вкладці Clock Configuration обрати HSE, тактовий сигнал PLL, увімкнути CSS та встановити максимальну тактову частоту (100MHz).



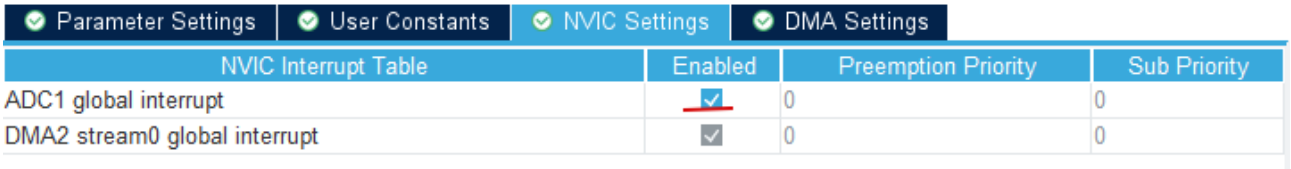
2.4 Активувати датчик температуриTemperature Sensor Channel та Vrefint Channel в вкладці Pinout & Configuration в розділі Analog>ADC1



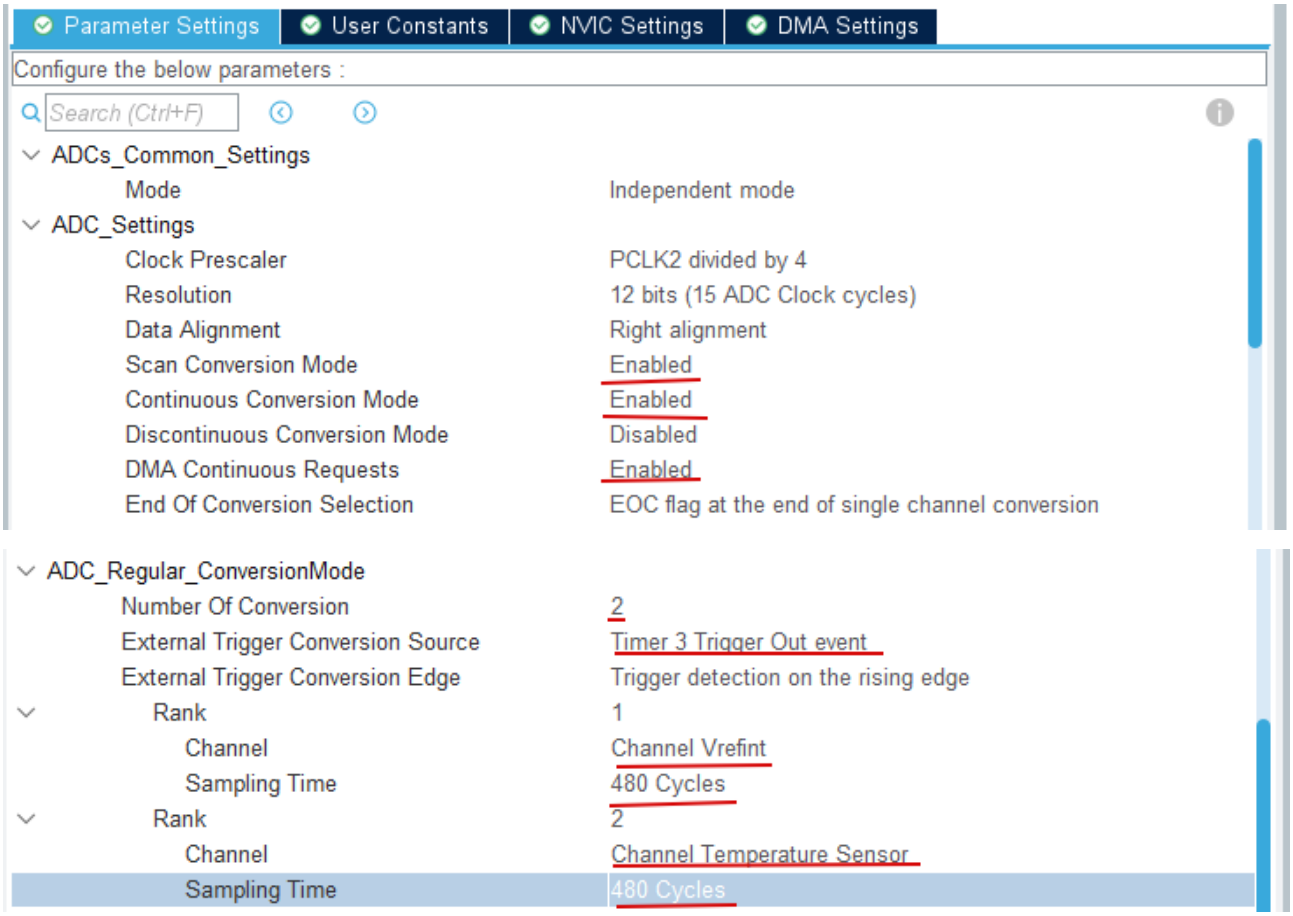
2.5 Налаштувати АЦП для цього натиснути DMA Settings>Add та обрати ADC1. В налаштуваннях обрати Mode: Circular



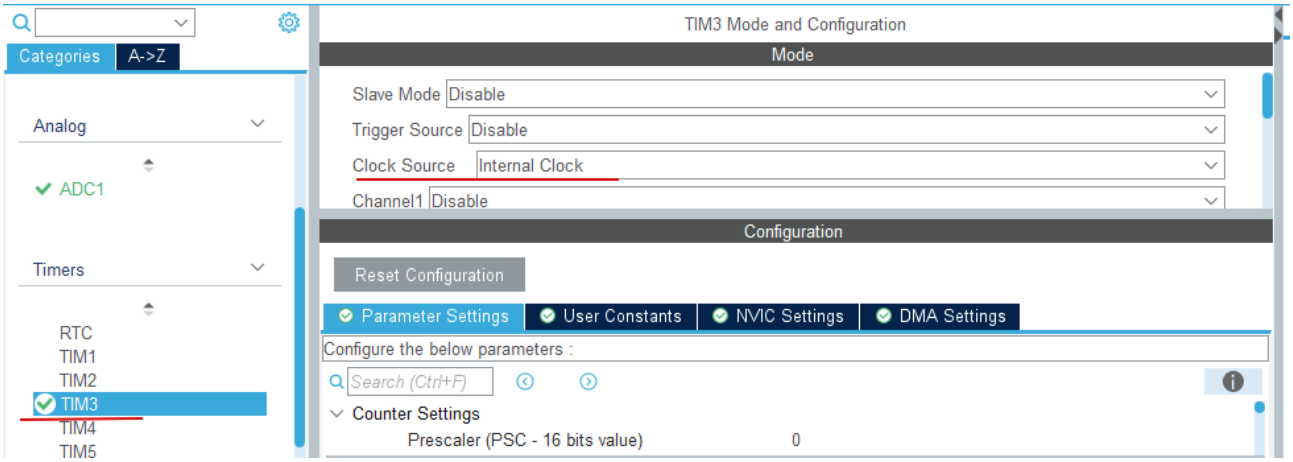
2.5 В NVIC Settings обрати глобальне переривання, щоб дізнатись про завершення перетворення.

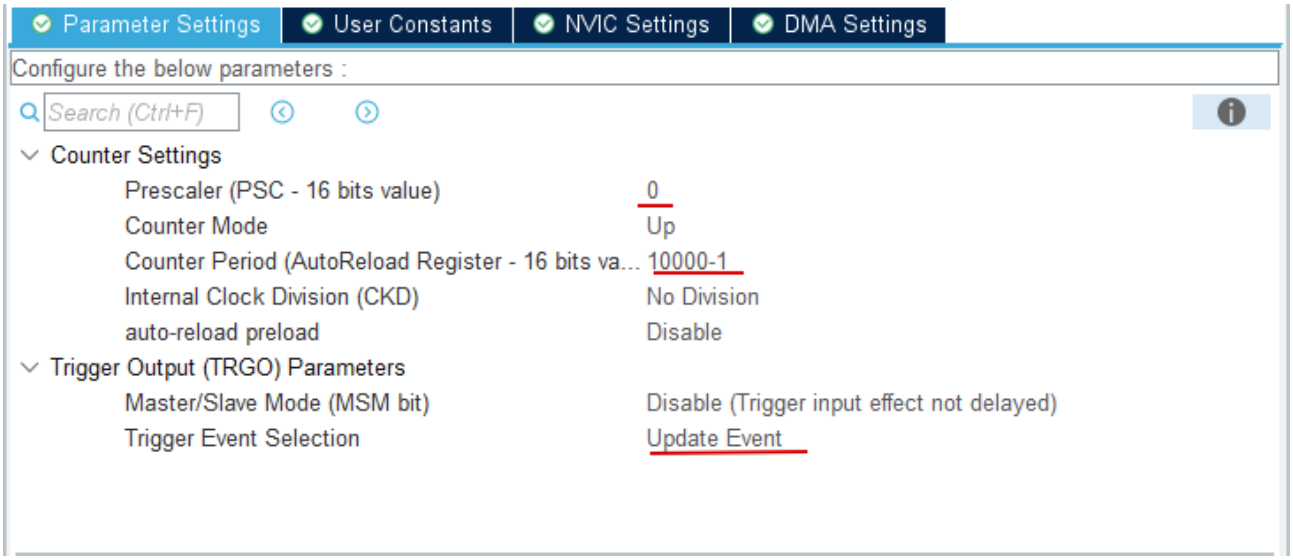


2.6 В вкладці Parameter Settings обрати наступні налаштування:



2.7 Останнім кроком налаштування конфігурації є активація таймера переривання:





2.8 Створити 2 файли tmpsensor.h та tmpsensor.c та вставити код з додатку 1 та додатку 2. В цих файлах прописана основна логіка роботи з вбудованим АЦП для визначення температури.

2.9 В код файлу main.c додати наступний код:

```
/* USER CODE BEGIN PV */

typedef struct AdcValues{

	uint16_t Raw[2]; /* Raw values from ADC */
	double IntSensTmp; /* Temperature */

}adcval_t;

adcval_t Adc;

typedef struct Flags
{
	uint8_t ADCCMPLT;

}flag_t;

flag_t Flg = {0, };

/* USER CODE END PV */
```

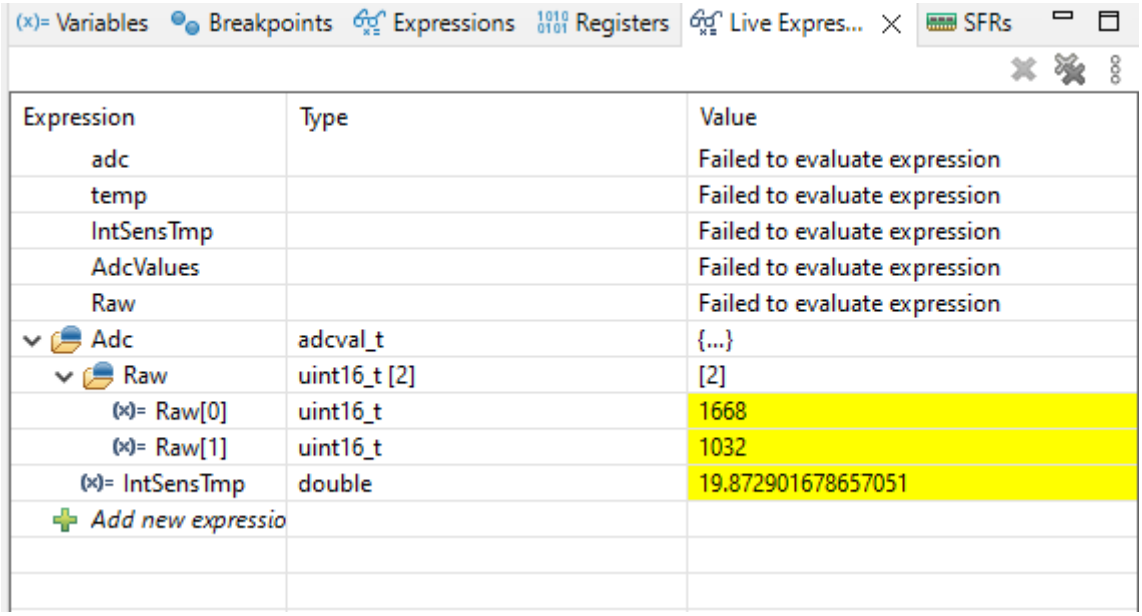
2.10 Додати код ініціалізації АЦП та запуск таймера який тригерить початок конвертації АЦП.

```
/* USER CODE BEGIN 2 */
 HAL_ADC_Start_DMA(&hadc1, (uint32_t*)Adc.Raw, 2);
 HAL_TIM_Base_Start(&htim3); /* This timer starts ADC conversion */
 /* USER CODE END 2 */
```

2.11 Почати зчитування даних температури:

```
while (1)
  {
	Adc.IntSensTmp = TMPSENSOR_getTemperature(Adc.Raw[1], Adc.Raw[0]);
  }
```

2.12 Запустити код в режимі дебагу та протестувати роботу. При правильному налагодженні має змінюватись значення.



Expression	Type	Value
adc		Failed to evaluate expression
temp		Failed to evaluate expression
IntSensTmp		Failed to evaluate expression
AdcValues		Failed to evaluate expression
Raw		Failed to evaluate expression
Adc	adcval_t	{...}
Raw	uint16_t [2]	[2]
Raw[0]	uint16_t	1668
Raw[1]	uint16_t	1032
IntSensTmp	double	19.872901678657051
Add new expressio		

Індивідуальне завдання

1. Продемонструвати роботу вбудованого внутрішнього термометра.
2. Під’єднати фоторезистор до контролера, зчитати данні та вивести на LCD дисплей;
3. Написати код керування освітленням за допомогою фоторезистора. При недостатньому освітлені буде загорятись світлодіод, а при достатньому затухати;

Зміст звіту

1. Тема та мета лабораторної роботи.
2. Скріншот роботи вбудованого термометра;
3. Скріншот конфігурації мікропроцесора (файл .ioc) для п.2 індивідуального плану;
4. Код файлу main.c функції main для кожного пункту індивідуального плану;
5. Висновки.

Контрольні питання

1. Які режими роботи має АЦП в STM32?
2. Як зменшити похибку вимірювання температури або освітлення?
3. Яка одиниця виміру використовується для цифрового значення з АЦП?
4. Для чого використовується функція HAL_ADC_Start_DMA()?
5. Як часто слід оновлювати значення температури для побудови точного графіка?

.

Додаток 1 - "tmpsensor.h"

```
#ifndef INC_TMPSENSOR_H_
#define INC_TMPSENSOR_H_

#include "main.h"

/* Configurations BEGIN */
#define TMPSENSOR_USE_INTREF 1 /* 1 - Use Internal Reference Voltage; 0 - Not use; */
/* Configurations END */

/* Constant values BEGIN */
#define TMPSENSOR_AVGSLOPE 2.5 /* mV/°C */
#define TMPSENSOR_V25  0.76 /* V (at 25 °C)  */

#define TMPSENSOR_ADCMAX 4095.0 /* 12-bit ADC maximum value (12^2)-1)  */
#define TMPSENSOR_ADCREFVOL  3.3 /* Typical reference voltage, V  */
#define TMPSENSOR_ADCVREFINT  1.21 /* Internal reference voltage, V  */
/* Constant values END */


/* PFP BEGIN */
double TMPSENSOR_getTemperature(uint16_t adc_sensor, uint16_t adc_intref);
/* PFP END */

#endif /* INC_TMPSENSOR_H_ */
```

Додаток 2 - "tmpsensor.c"

```
#include "tmpsensor.h"


double TMPSENSOR_getTemperature(uint16_t adc_sensor, uint16_t adc_intref){

#if(TMPSENSOR_USE_INTREF)

	double intref_vol =
(TMPSENSOR_ADCMAX*TMPSENSOR_ADCVREFINT)/adc_intref;

#else
	double intref_vol = TMPSENSOR_ADCREFVOL;
#endif

	double sensor_vol = adc_sensor * intref_vol/TMPSENSOR_ADCMAX;

	double sensor_tmp = (sensor_vol - TMPSENSOR_V25)
*1000.0/TMPSENSOR_AVGSLOPE + 25.0;

	return sensor_tmp;
}
```