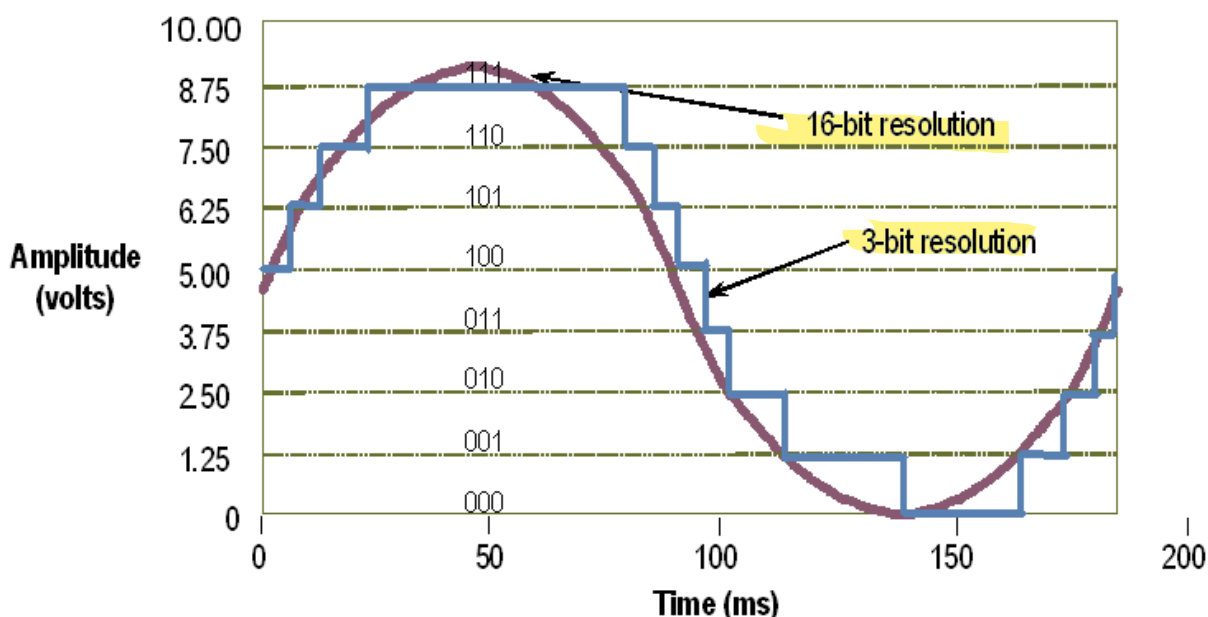


Chức năng ADC sử dụng vi điều khiển STM32F103C8T6

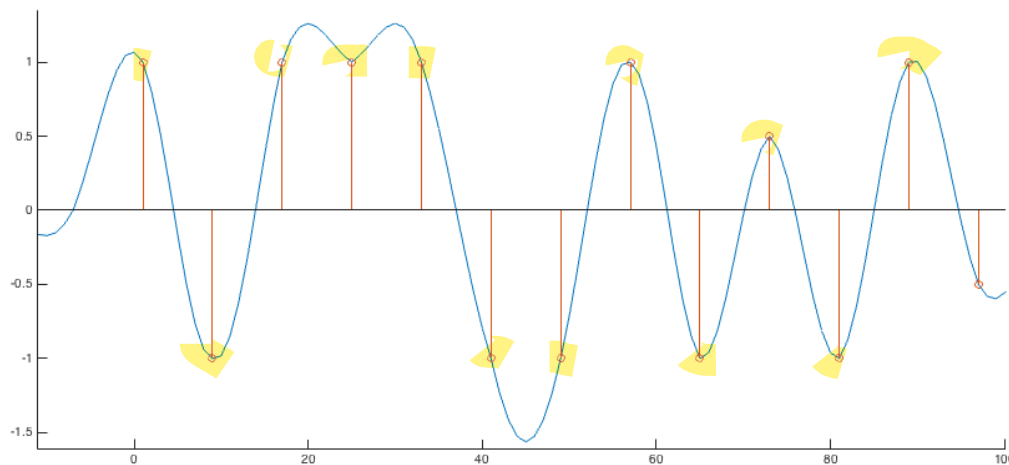
Trong các ứng dụng vi điều khiển – hệ thống nhúng, bộ chuyển đổi tương tự-số (ADC) là 1 thành phần rất quan trọng để có thể chuyển đổi các dữ liệu dạng analog từ môi trường (nhiệt độ, độ ẩm, độ sáng,...) sang dạng digital để vi điều khiển có thể xử lý được. STM32F103C8 có tích hợp sẵn các bộ chuyển đổi ADC với độ phân giải 12bit. Có 12 kênh cho phép đo tín hiệu từ 10 nguồn bên ngoài và 2 nguồn nội bên trong. Trong bài này, chúng ta sẽ cùng tìm hiểu về chế độ đơn kênh với STM32, sử dụng Interrupt để báo quá trình chuyển đổi hoàn tất.

Trong bộ chuyển đổi ADC, có 2 thuật ngữ mà chúng ta cần chú ý đến, đó là độ phân giải (resolution) và thời gian lấy mẫu (sampling time)



– Độ phân giải (resolution): dùng để chỉ số bit cần thiết để chứa hết các mức giá trị số (digital) sau quá trình chuyển đổi ở ngõ ra. Như trên biểu đồ:

- Màu xanh tương ứng thể hiện độ phân giải của bộ chuyển đổi này là 3 bit, tương ứng với 8 sự thay đổi ở đầu ra số ($2^3=8$). Khi đưa vào điện áp tương tự, bộ chuyển đổi sẽ thực hiện một công đoạn lượng tử hóa để đưa các kết quả tương ứng từ điện áp tương tự về số ở ngõ ra.
- Màu tím tương ứng với độ phân giải của bộ chuyển đổi 16 bit. Dễ dàng nhận thấy với một bộ chuyển đổi có độ phân giải càng thấp, quá trình chuyển đổi sẽ cho ra kết quả là một điện áp càng biến dạng ở ngõ ra so với ngõ vào và ngược lại. Bộ chuyển đổi ADC của STM32F103 có độ phân giải mặc định là 12 bit, tức là có thể chuyển đổi ra $2^{12} = 4096$ giá trị ở ngõ ra số.



- Thời gian lấy mẫu (sampling time) là khái niệm được dùng để chỉ thời gian giữa 2 lần số hóa của bộ chuyển đổi. Như ở hình trên, sau khi thực hiện lấy mẫu, các điểm tròn chính là giá trị đưa ra tại ngõ ra số. Dễ nhận thấy nếu thời gian lấy mẫu quá lớn thì sẽ làm cho quá trình chuyển đổi càng bị mất tín hiệu ở những khoảng thời gian không nằm tại thời điểm lấy mẫu. Thời gian lấy mẫu càng nhỏ sẽ làm cho việc tái thiết tín hiệu trở nên tin cậy hơn.

Các chức năng chính của ADC trong STM32

Độ phân giải 12Bit

Sinh ra ngắt tại các sự kiện End of convert, End of Injected, Analog Watchdog

Chế độ Single hoặc Continuous

Chế độ Scan tự động quét từ Kênh 0 đến Kênh n (mỗi bộ có 10 kênh tối đa)

Có cơ chế cân chỉnh tay

Data Alignment (Căn chỉnh Data) căn trái hoặc căn phải

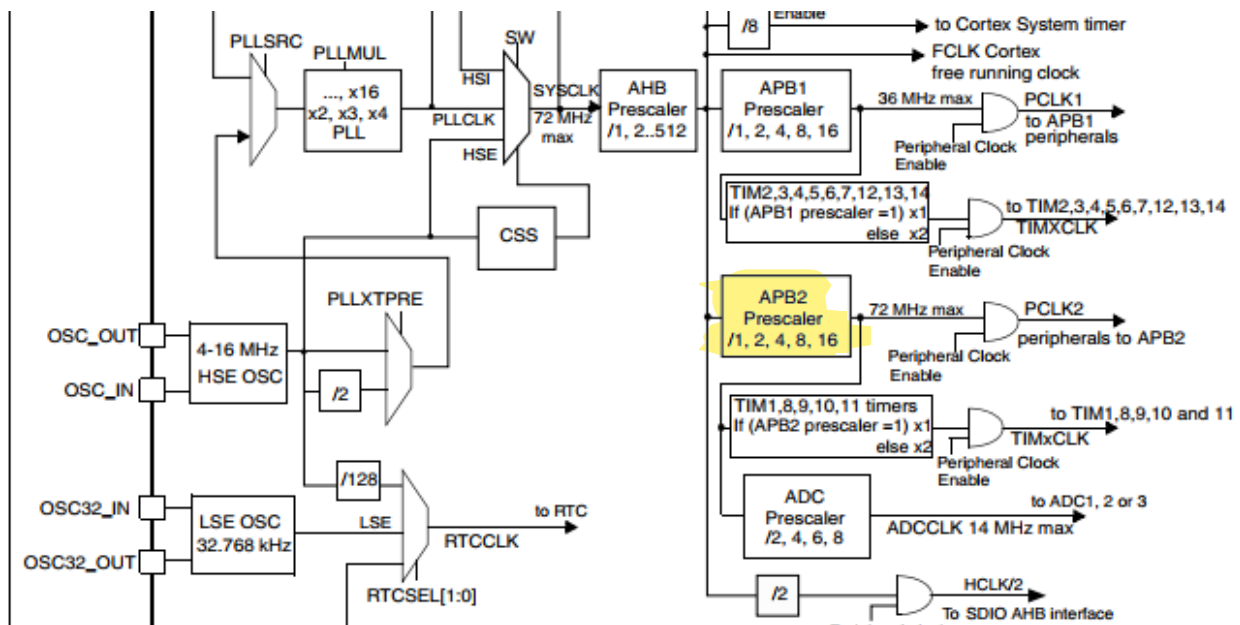
Cài đặt thời gian chuyển đổi đến từng Kênh

Có thể kích hoạt bằng xung bên ngoài

Chế độ Dual mode sử dụng cùng lúc 2 hoặc nhiều bộ ADC

Hỗ trợ DMA

Tần số chuyển đổi ADC được lấy từ bộ ABP2 thông qua ADC prescaler và phải nhỏ hơn 14mhz



ADC một kênh chế độ Single và Continuous

Với chế độ **Single** bộ ADC chỉ chuyển đổi 1 lần rồi dừng, một sự kiện ngắt được sinh ra nếu bit EOCIR được set lên 1

Với chế độ **Continuous** bộ ADC sẽ chuyển đổi liên tục, một sự kiện ngắt được sinh ra nếu bit EOCIR được set lên 1

DMA sẽ sinh ra nếu bit DMA dc set lên 1 (DMA chỉ có trên ADC1 và ADC3)

Kết quả convert được lưu vào thanh ghi DR

Bắt đầu convert bằng cách set bit ADON lên 1

Cơ chế để tạo ADC như sau:

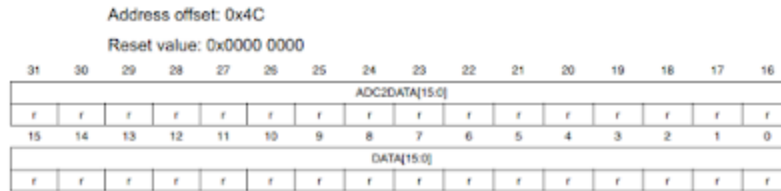
1. Enable bộ clock cho ADC, ghi hệ số chia cho ADC prescaler
2. Chọn các kênh cần chuyển đổi
3. Chọn chế độ chuyển đổi Single, Continuous, Scan, Discontinuous
4. Chọn thời gian lấy mẫu (Sampling Time)
5. Chọn Ngắt hoặc DMA

6. Start bộ chuyển đổi

7. Kiểm tra cờ EOC hoặc trong ngắt đọc dữ liệu từ thanh ghi DR về

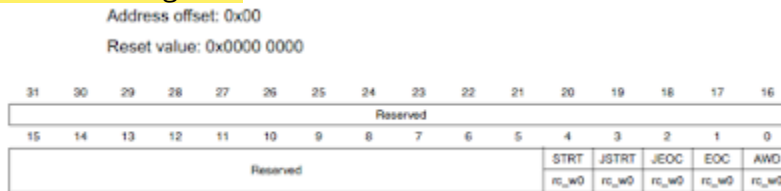
Một số thanh ghi quan trọng.

1. **ADC_DR** – ADC regular data register.



Thanh ghi này chứa giá trị ADC đọc về, nó là thanh ghi 32 bit với 16 bit data của bộ ADC1 và 16 bit data của bộ ADC2.

2. **ADC_SR** – ADC status register.



Thanh ghi này chứa các cờ báo trạng thái như:

1. **STRT** : báo channel đã bắt đầu chuyển đổi giá trị ADC hay chưa.
 2. **JSTRT** : báo channel đã bắt đầu chuyển đổi khi có tín hiệu bên ngoài điều khiển hay chưa.
 3. **JEOC**: báo kết thúc quá trình chuyển đổi khi có tín hiệu bên ngoài điều khiển hay chưa.
 4. **EOC**: báo kết thúc quá trình chuyển đổi ADC.
 5. **AWD**: báo có sự kiện Analog Watchdog có xảy ra hay không.
3. **ADC_CR2** – ADC Control register 2.



Thanh ghi này điều khiển các quá trình chuyển đổi ADC như:

1. **TSVREFE**: bật hay tắt cảm biến nhiệt độ và Vrefint.
2. **SWSTART** : bật hay reset trạng thái bộ chuyển đổi liên tục.
3. **JSWSTART**: bật hay reset trạng thái bộ chuyển đổi liên tục được điều khiển từ bên ngoài bộ ADC.

4. EXTTRIG: cho phép hay không cho phép bắt đầu bộ chuyển đổi liên tục từ xung trigger bên ngoài.
 5. EXTSEL[2:0] : bit chọn lựa xung trigger bên ngoài từ nguồn nào.
 6. ALIGN : sắp xếp thanh ghi data theo chiều từ lớn đến bé hoặc ngược lại.
 7. DMA: có sử dụng bộ DMA hay không.
 8. RSTCAL: reset lại thanh ghi calib hay không.
 9. CAL: cho phép hay báo là đã calib xong.
 10. CONT: lựa chọn mode chuyển đổi liên tục hay chuyển đổi đơn.
 11. ADON: bật hay tắt bộ chuyển đổi ADC.
4. ADC_SMPR2 – ADC sample time register.

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved		SMP9[2:0]				SMP8[2:0]				SMP7[2:0]				SMP6[2:0]		SMP5[2:1]
Rst.		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
SMP5_0		SMP4[2:0]				SMP3[2:0]				SMP2[2:0]				SMP1[2:0]		SMP0[2:0]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Thanh ghi này thiết lập thời gian lấy mẫu nhanh hay chậm và được cài đặt bởi lập trình. SMPx[2:0] tương ứng giá trị nhị phân từ 0->7 sẽ tương ứng với thời gian lấy mẫu là: 1.5 - 7.5 - 13.5 - 28.5 - 41.5 - 55.5 - 71.5 - 239.5 cycles. Cách tính thời gian dựa theo hình sau:

ADC samples the input voltage for a number of ADC_CLK cycles which can be modified using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. Each channel can be sampled with a different sample time.

The total conversion time is calculated as follows:

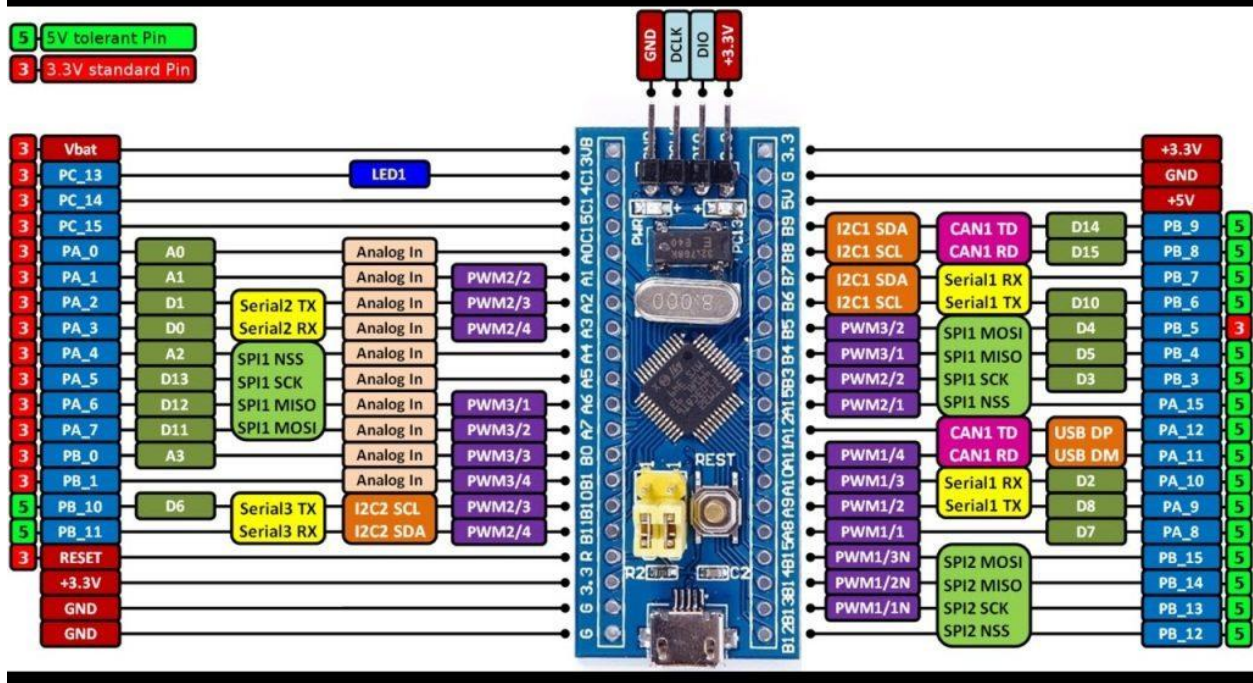
$$T_{\text{conv}} = \text{Sampling time} + 12.5 \text{ cycles}$$

Example:

With an ADCCLK = 14 MHz and a sampling time of 1.5 cycles:

$$T_{\text{conv}} = 1.5 + 12.5 = 14 \text{ cycles} = 1 \mu\text{s}$$

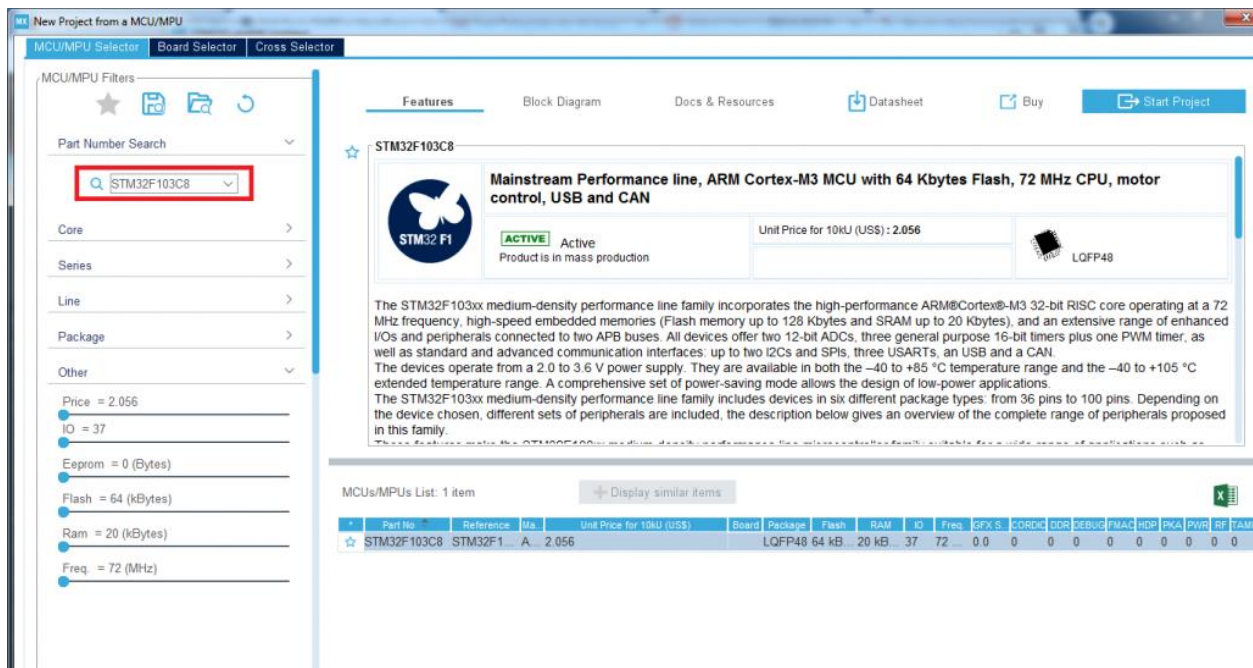
Trong bài này chúng ta sẽ thông qua công cụ CubeMX để có thể cấu hình thời gian lấy mẫu tại module ADC1



Ở trên kit gồm có 10 channel ADC, chúng ta sẽ sử dụng channel 1 để đọc điện áp từ một biến trở đưa vào

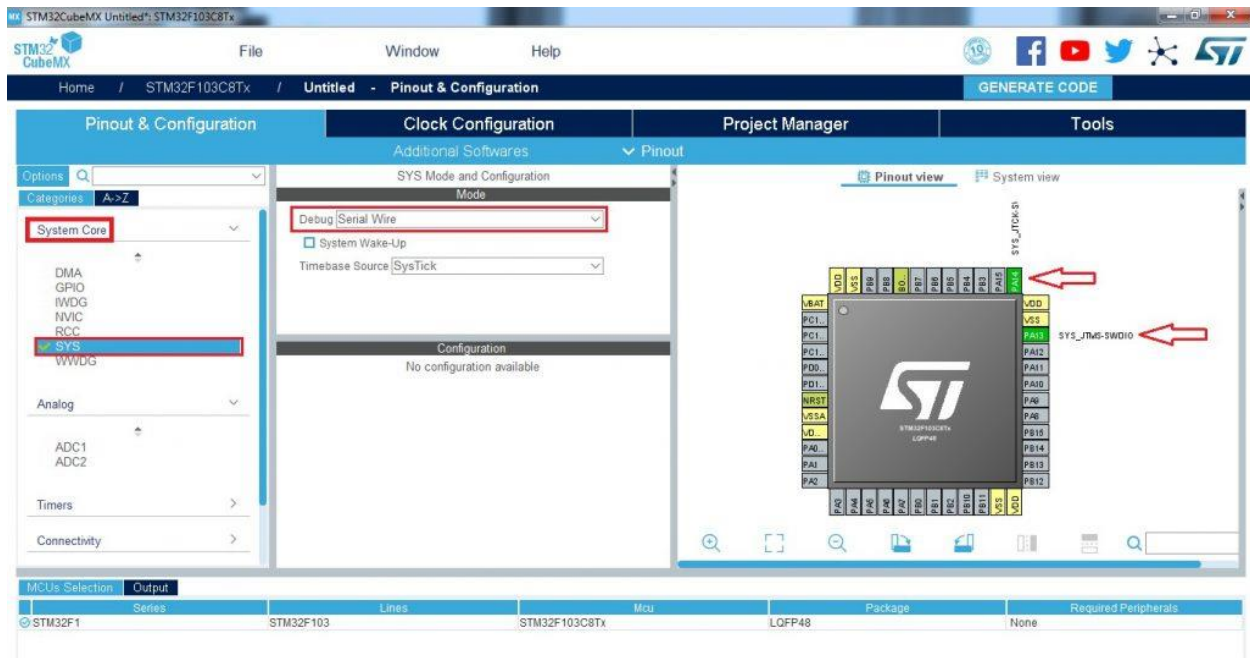
Bước 1:

- Khởi động CubeMX
- Chọn chip
- Bắt đầu khởi tạo project mới



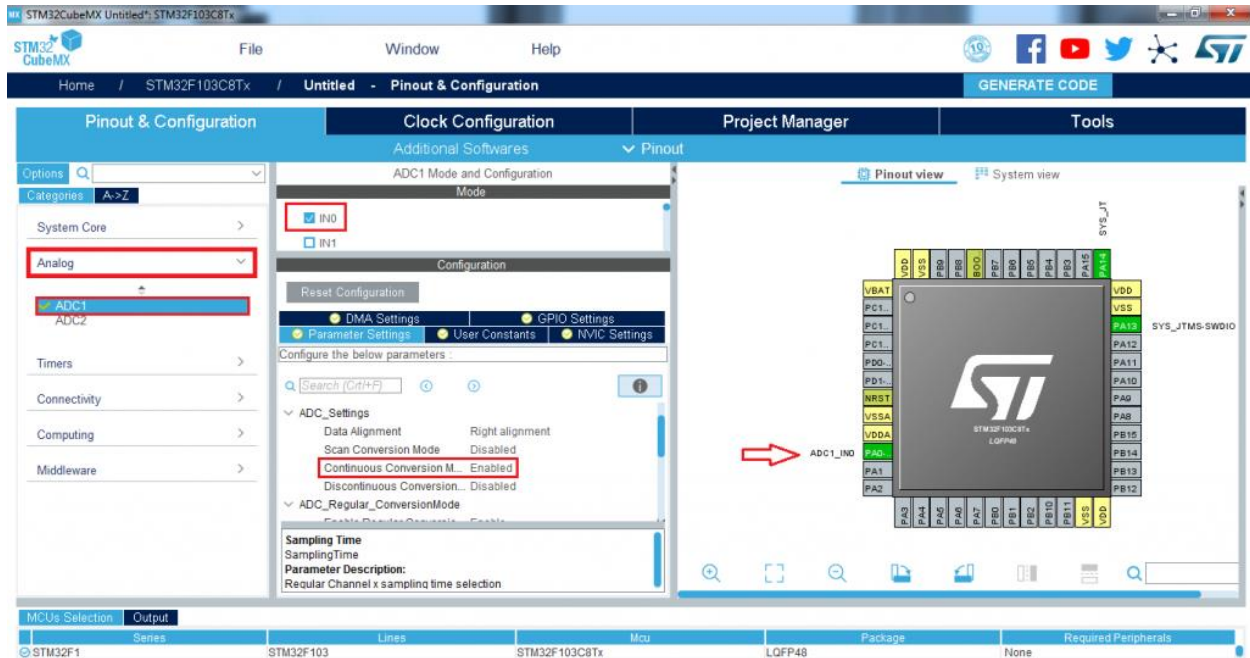
Bước 2:

Cấu hình việc nạp code ở module SYS sang Serial Wire



Bước 3:

Chọn channel 0 của ADC1 (như ta thấy, chân PA0 đã được cấu hình) (nếu như muốn chọn các channel khác, các chân tại port A tương ứng sẽ được cấu hình theo, ví dụ như channel 2 sẽ là PA2, channel 3 sẽ là PA3,...)

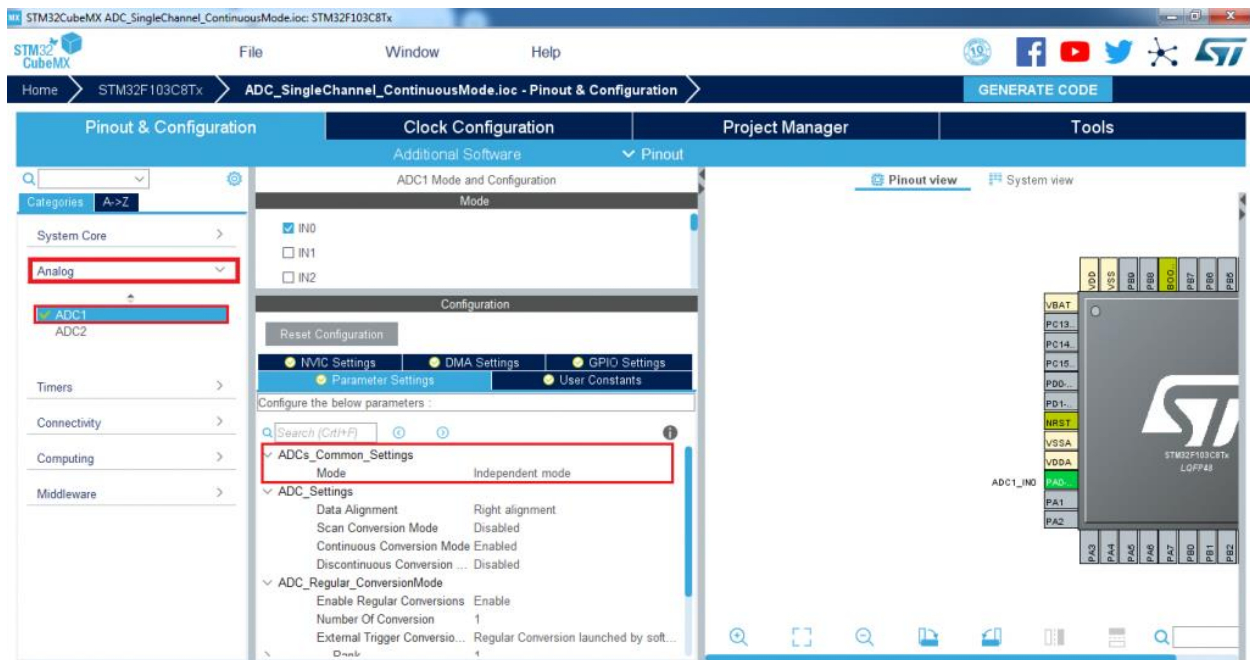


Bước 4:

Chuyển sang mục Analog -> ADC1 -> Configuration để thay đổi các thông số của module ADC1

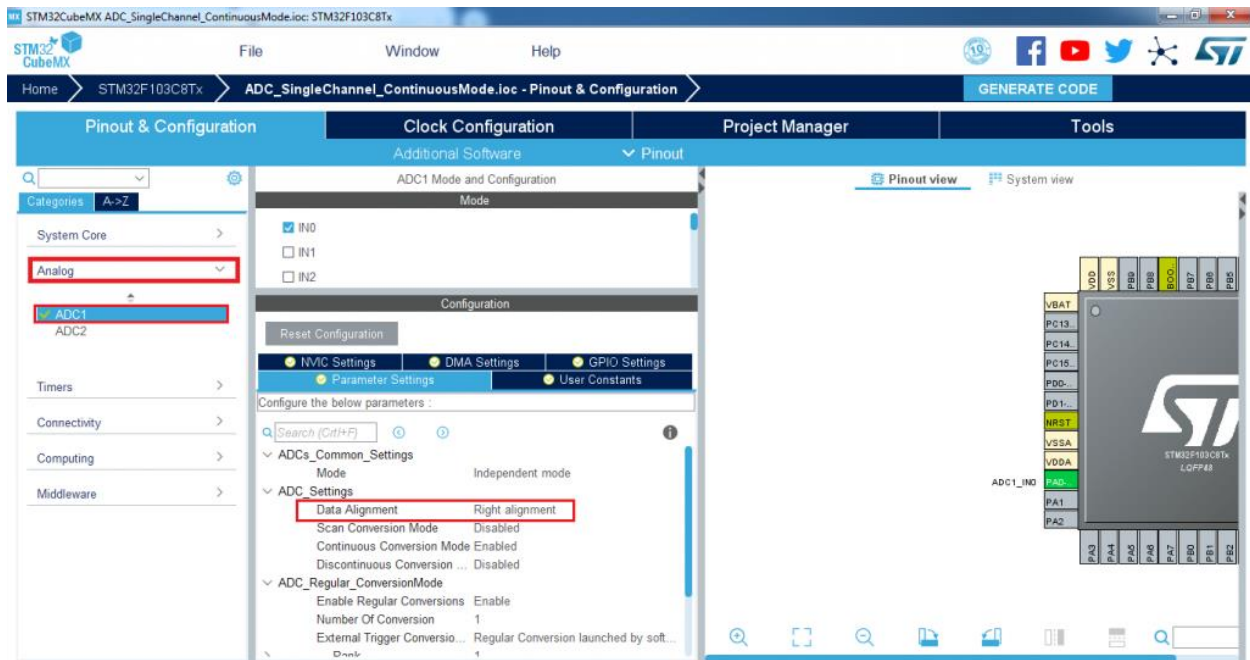
Bước 5:

Ở mục Mode, chỉ có 1 chế độ *Independent Mode* nên chúng ta sẽ giữ nguyên



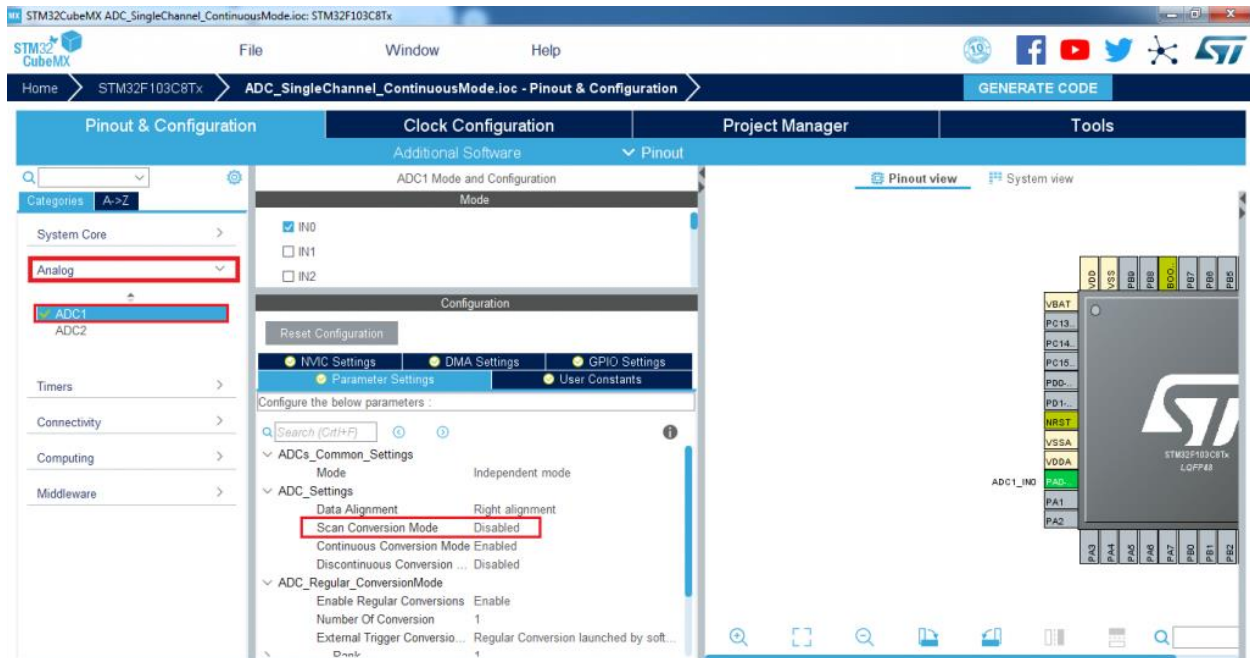
Bước 6:

Bộ ADC của STM32F103 có độ phân giải là 12bit mà ta sẽ phải cần lưu trữ vào một thanh ghi 32 bit, do đó sẽ còn thừa 20 bit. Chúng ta sẽ cấu hình việc căn lề cho 12 bit này nằm bên phải hay bên trái trong thanh ghi 32 bit đó tại mục *Data Alignment*



Bước 7:

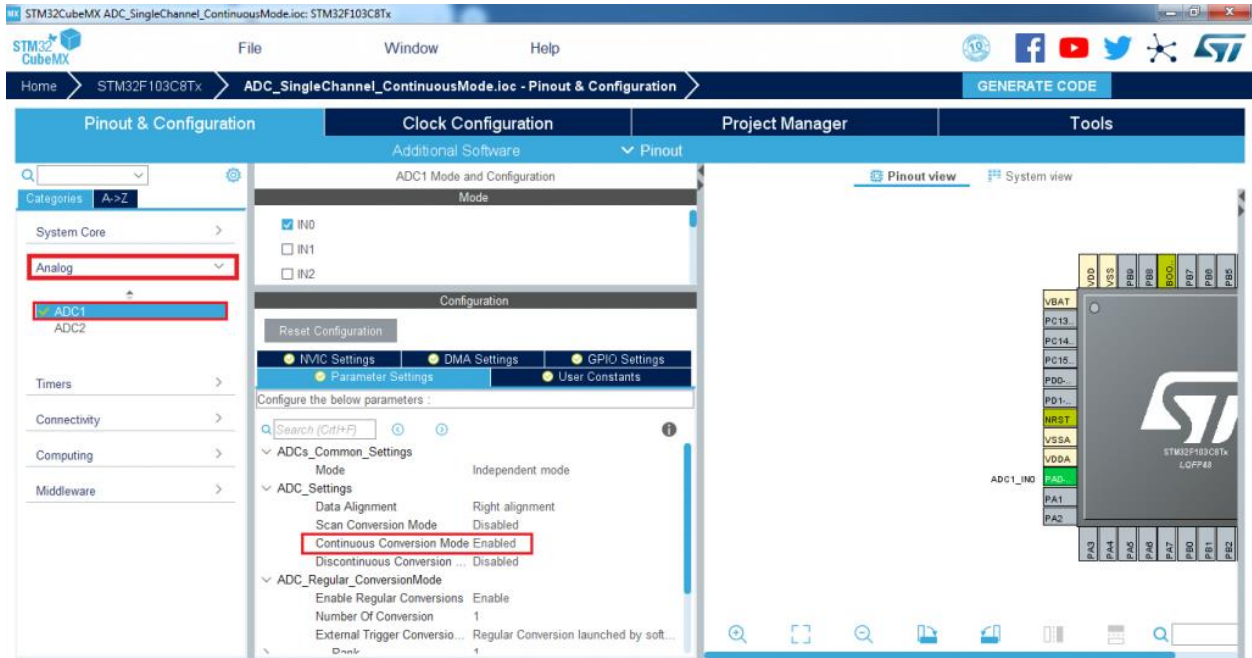
Mục *Scan Conversion Mode* sẽ được sử dụng để “quét” qua lần lượt các kênh ADC trong quá trình đọc dữ liệu, vì ta đang sử dụng chế độ đơn kênh nên chế độ này sẽ không có tác dụng, ta giữ nguyên



Bước 8:

- Mục *Continuous Conversion Mode* sẽ quyết định cho bộ ADC của chúng ta có sử dụng chế độ chuyển đổi liên tục hay không.
- Chọn *Enabled*

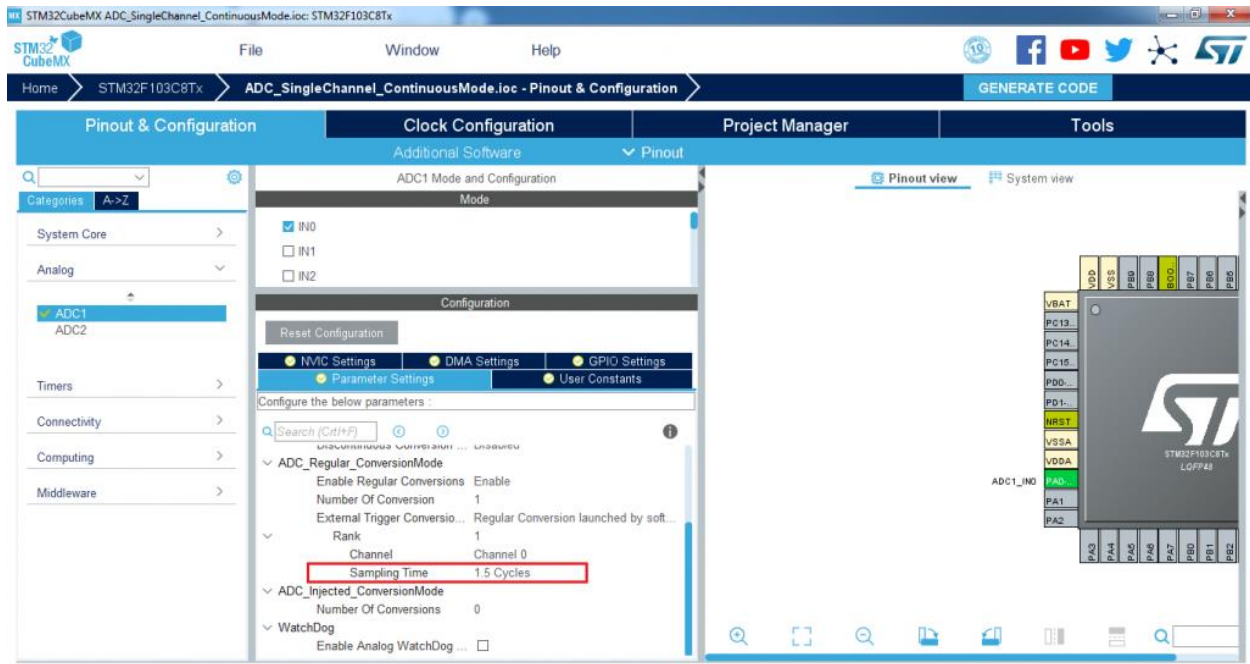
Nếu chúng ta không *enable* mode này, sau mỗi lần chuyển đổi, ta sẽ phải gọi lại lệnh đọc giá trị ADC để bắt đầu quá trình chuyển đổi mới



Bước 9:

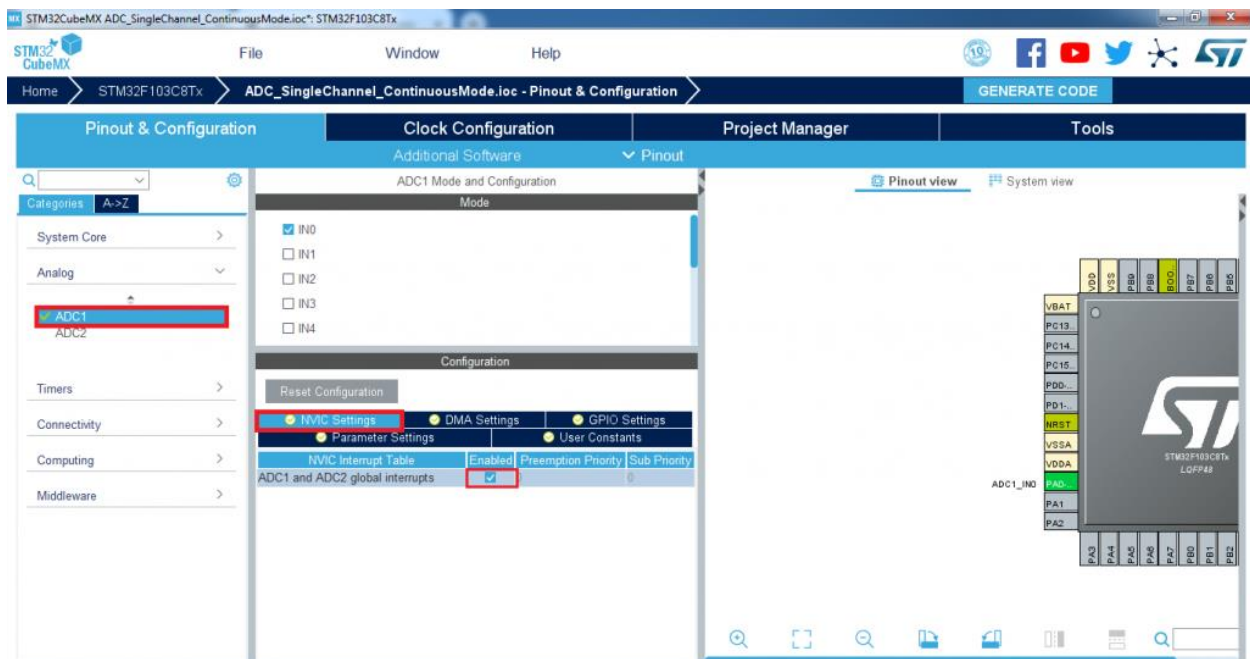
Tại mục *Sampling Time*, chúng ta sẽ chọn thời gian lấy mẫu trong quá trình số hóa. Tùy vào ứng dụng mà chúng ta có thể chọn thời gian lấy mẫu cho phù hợp.

****Lưu ý:** thời gian lấy mẫu càng ngắn, việc tái thiết tín hiệu càng chính xác nhưng năng lượng tiêu tốn sẽ càng cao



Bước 10:

Chuyển sang tab *NVIC Settings* , sử dụng chế độ Interrupt để báo hiệu việc chuyển đổi đã hoàn tất nên ta sẽ *enable* interrupt ADC



Bước 11:

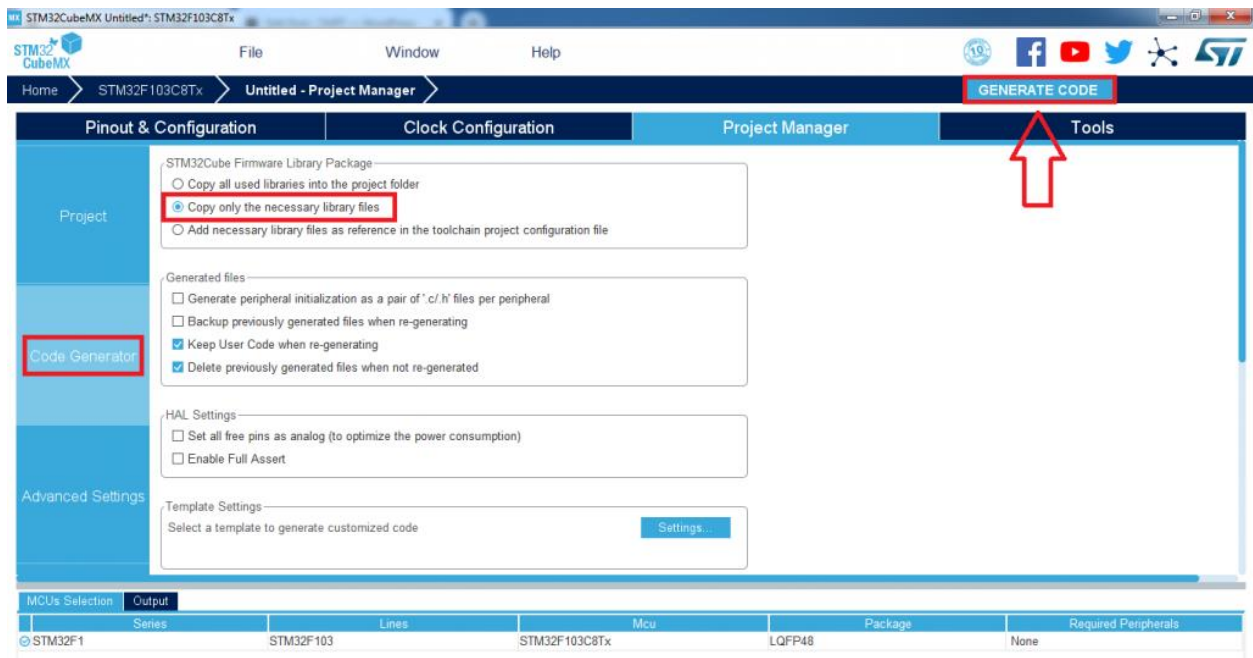
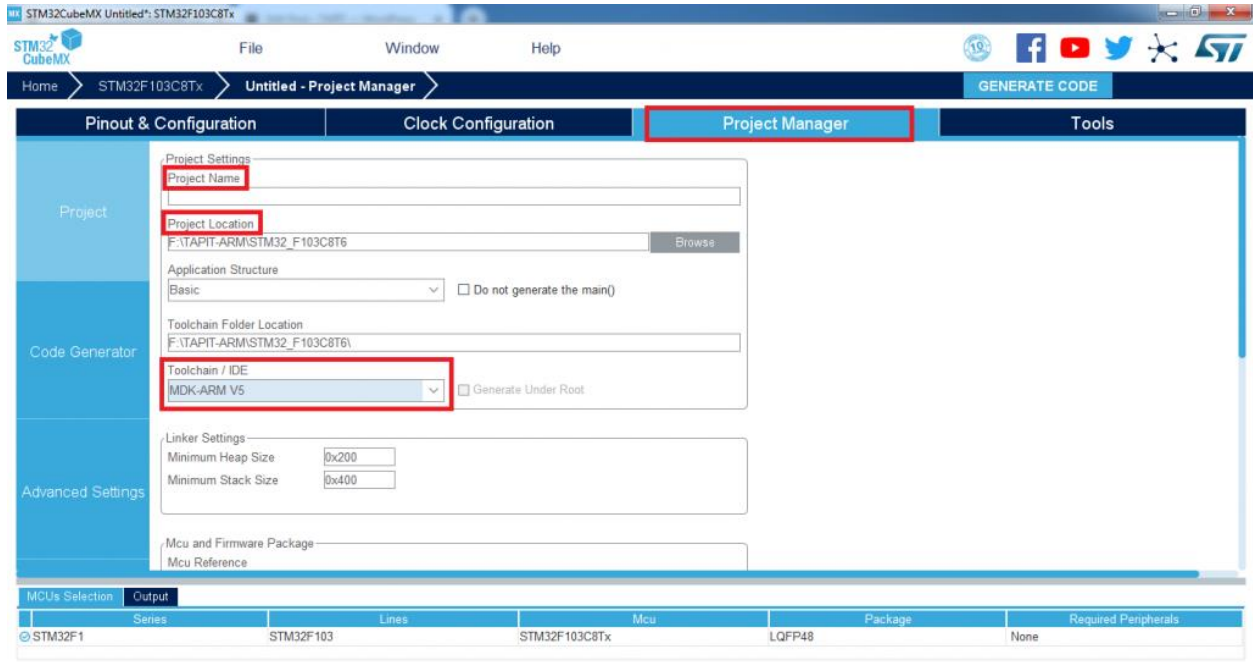
Sau khi hoàn thành các bước trên:

- Chọn OK để xác nhận và đóng hộp thoại cấu hình

- Chuyển sang mục Setting để cài đặt việc tạo code cấu hình từ CubeMX

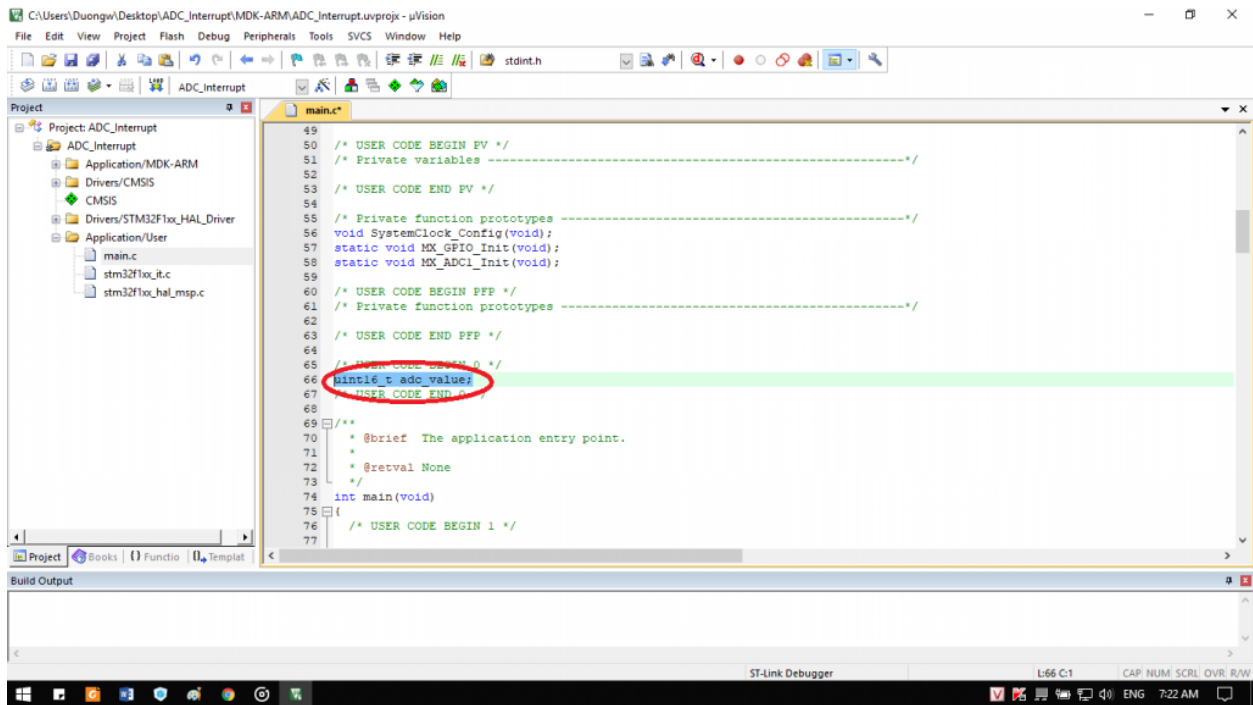
Bước 12:

- Chọn toolchain/IDE là MDK-ARM V5
- Đặt tên project
- Chọn ok để xác nhận và sinh code



Bước 13:

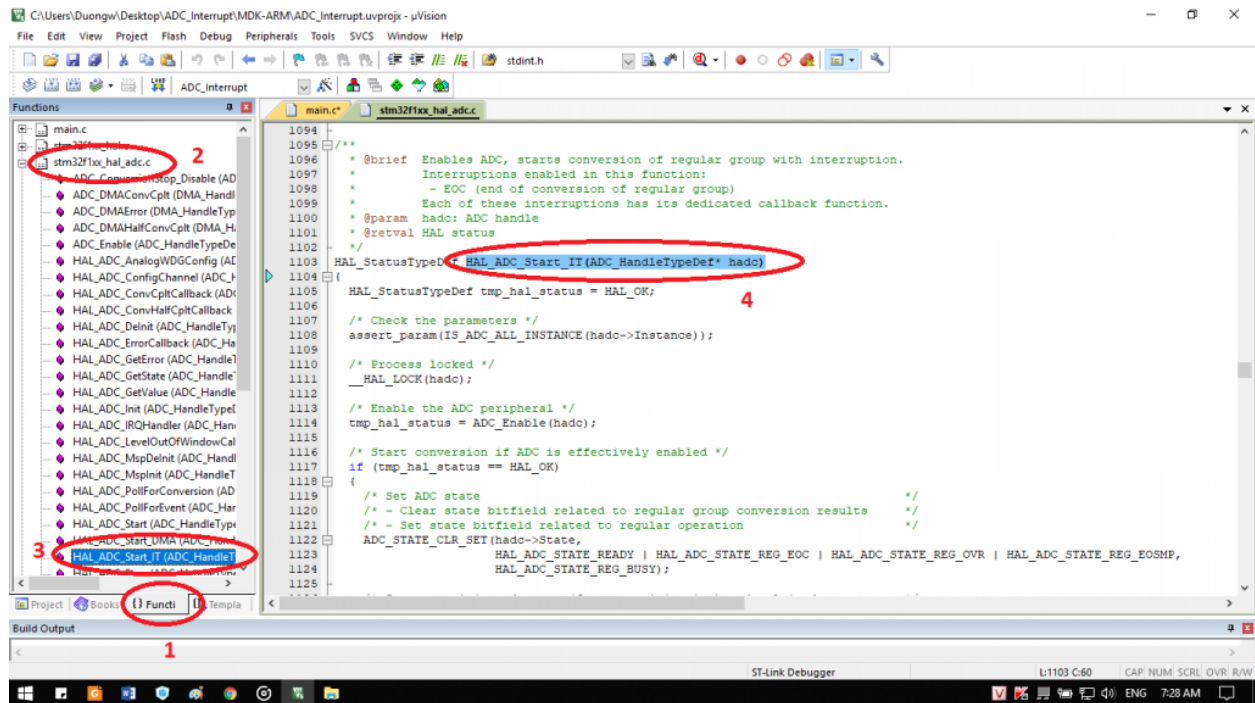
Tạo trước một biến *int* 16 bit để lưu giá trị của điện áp đọc vào từ biến trở trong file *main.c*



Bước 14:

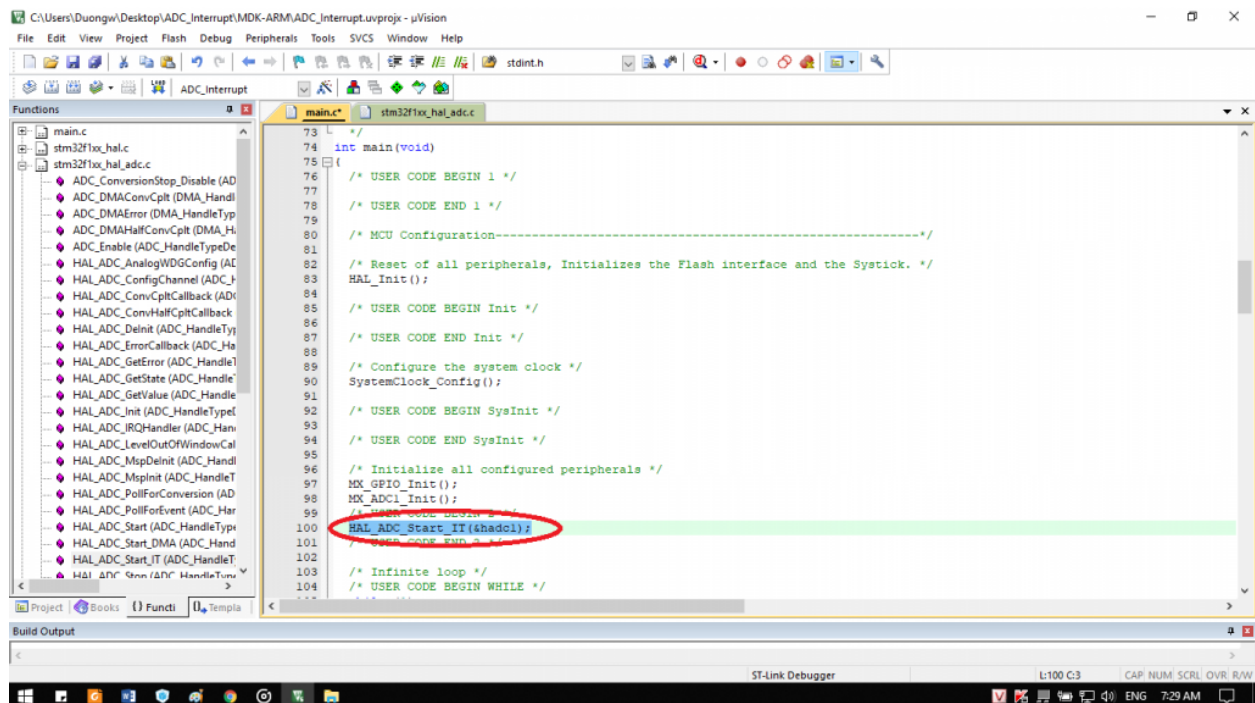
Để sử dụng module ADC1 ở chế độ Interrupt, chúng ta cần phải khởi động nó.

- Mở tab Function, mở file stm32f1xx_hal_adc.c lên
- Tìm đến hàm HAL_ADC_Start_IT⁽¹⁾, và chọn copy hàm đó



Bước 15:

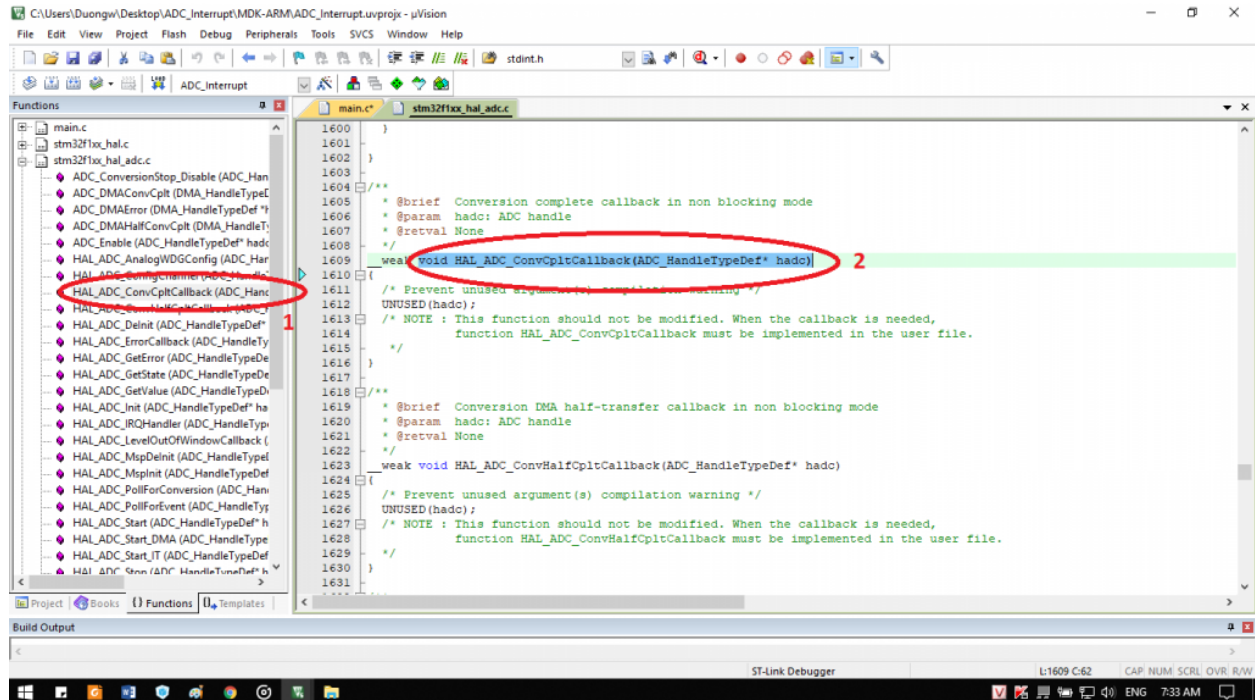
- Paste phần đó vào trong hàm main tại file main.c
- Sửa đổi tham số truyền vào thành &adc1 (vì chúng ta đang cần khởi động module ADC1)



Bước 16:

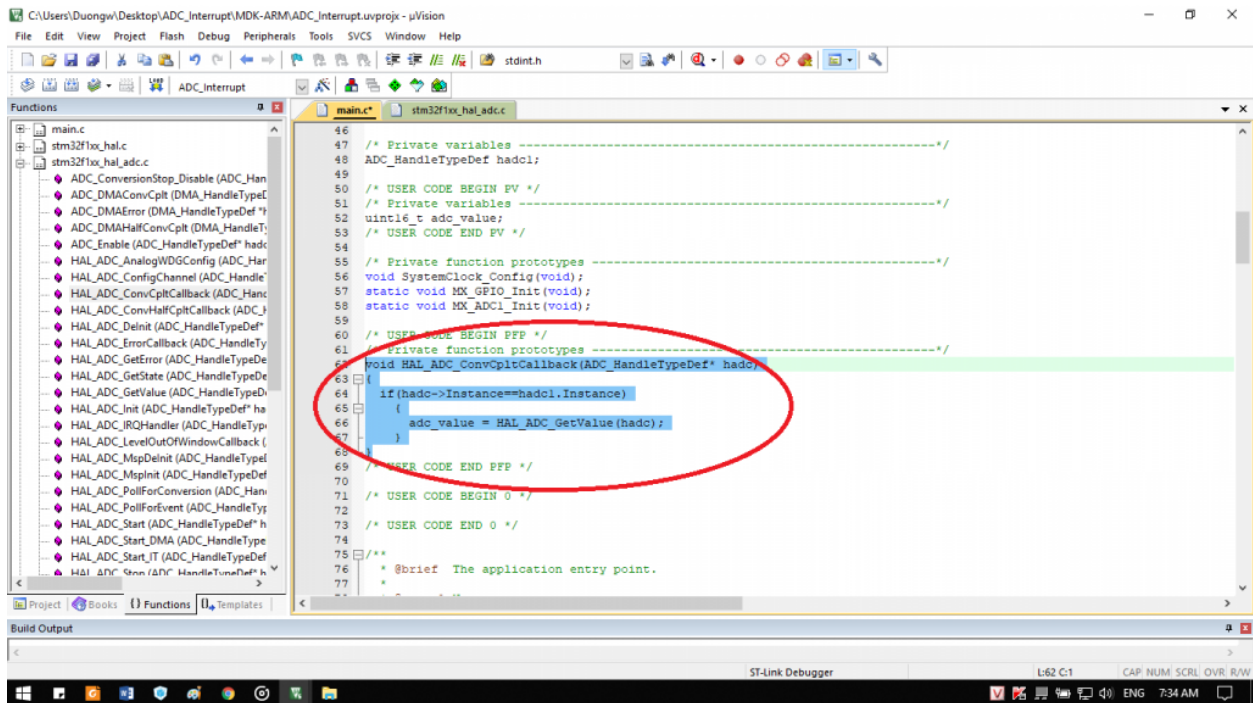
Sau khi quá trình chuyển đổi số tại 1 thời điểm lấy mẫu thành công, chúng ta sẽ cần có một thông báo để bắt đầu lưu dữ liệu vào một biến, trong trường hợp này, chúng ta sẽ sử dụng một ngắt để báo hiệu.

- Tìm đến hàm HAL_ADC_ConvCpltCallback⁽²⁾
- Copy hàm đó



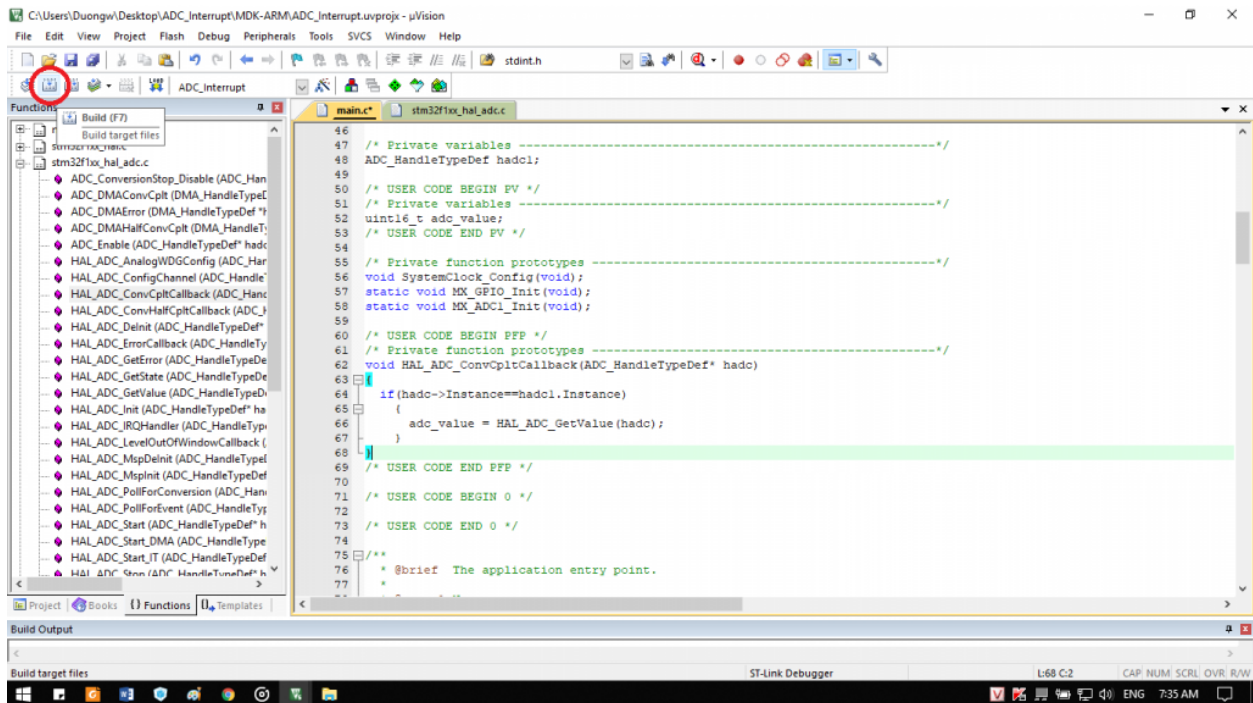
Bước 17:

- Paste hàm này vào file main.c và bên ngoài hàm main (vì đây là một IRQ Handle), thêm một lệnh if như dưới vào để kiểm tra xem ngắt báo hiệu chuyển đổi thành công đến từ ADC1 hay ADC2.
- Sau khi hàm này được gọi, hàm HAL_ADC_GetValue⁽³⁾ sẽ được gọi để đọc giá trị số hóa và lưu vào biến adc_value



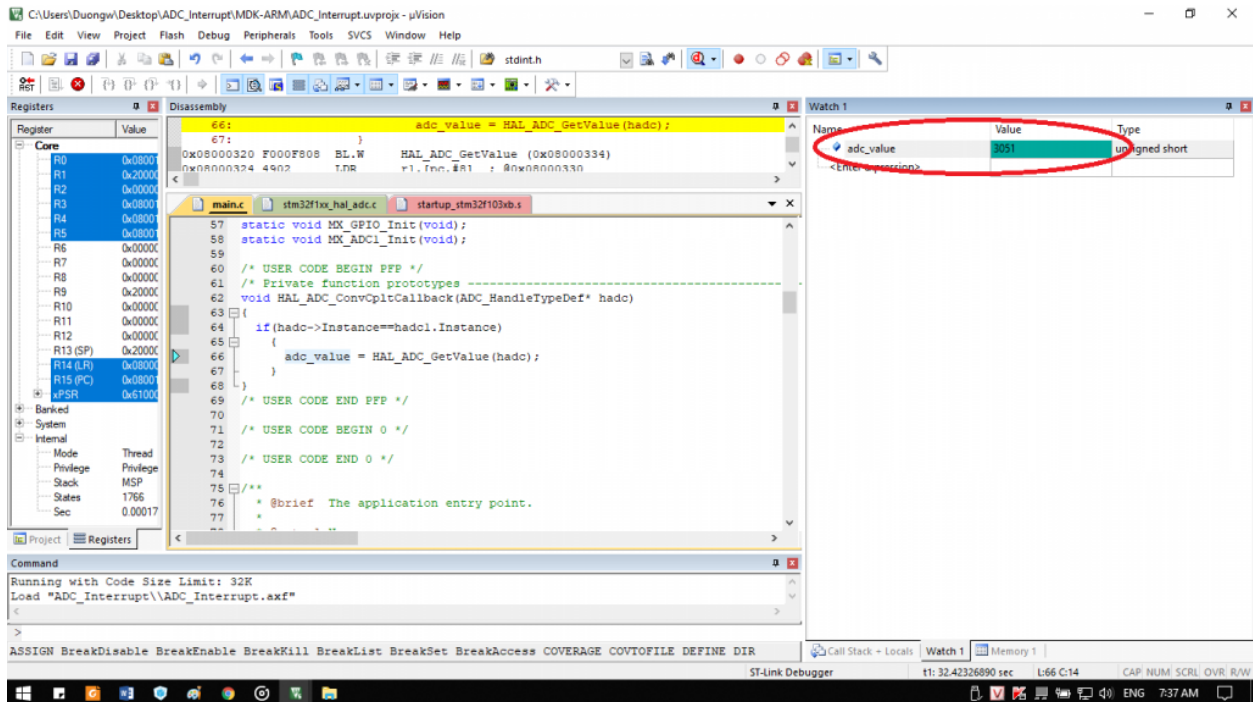
Bước 18:

Thực hiện Build Project



Bước 19:

Thực hiện chức năng Debug để quan sát sự thay đổi của biến `adc_value`



****Chú thích các hàm cơ bản:**

(1) `HAL_ADC_Start_IT`:

-Chức năng: Khởi động module ADC với ngắt báo hiệu chuyển đổi thành công (ngắt ở đây là End Of Conversion (EOC))

-Tham số truyền vào: Địa chỉ của module ADC cần được khởi động

(2) `HAL_ADC_ConvCpltCallback`:

-Chức năng: IRQ Handle, được gọi khi có quá trình chuyển đổi thành công từ một module ADC bất kỳ

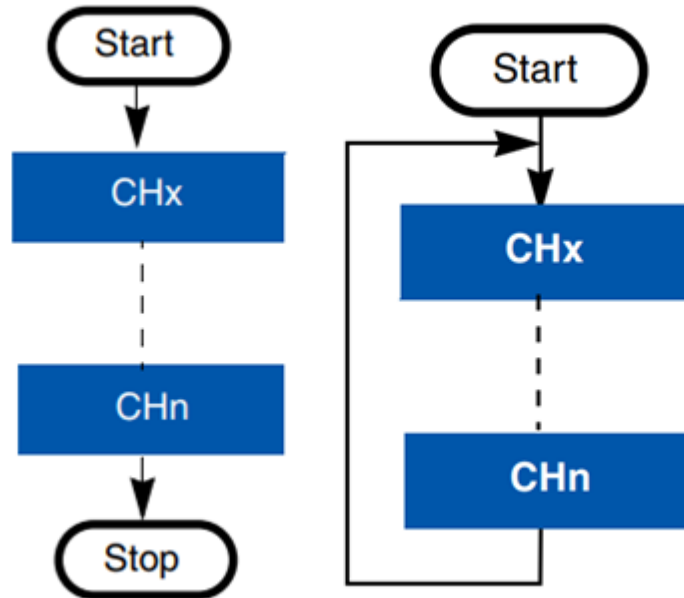
(3) `HAL_ADC_GetValue`:

-Chức năng: Đọc giá trị số hóa sau khi đã hoàn thành chuyển đổi

-Tham số truyền vào: Địa chỉ của module ADC cần lấy dữ liệu

-Trả về: Giá trị sau khi đã hoàn thành chuyển đổi

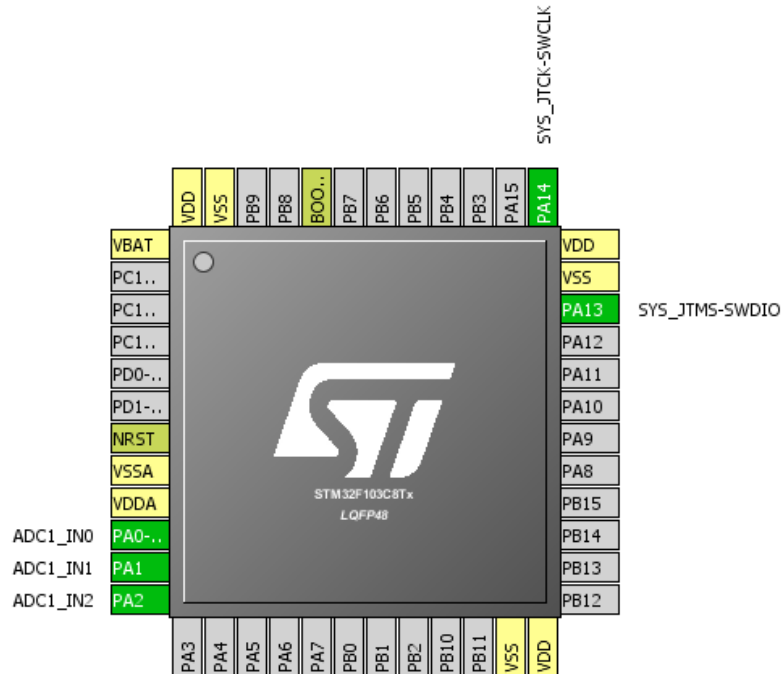
Đọc ADC nhiều kênh trên SMT32 sử dụng DMA



Multichannel (scan), single conversion mode là mode được sử dụng để đọc nhiều kênh sau mỗi lần bắt đầu, còn Multichannel (scan) continuous, conversion mode cũng là mode được sử dụng để đọc nhiều kênh ADC, tuy nhiên dữ liệu ADC sẽ được đọc liên tục sau khi bắt đầu.

Đầu tiên, các bạn cần cấu hình ADC và DMA của vi điều khiển STM32F103C8T6 trên phần mềm **STM32CubeMX** và sau đó sinh code qua những bước sau:

Bước 1: Tạo project, tại thẻ Pinout, các bạn cấu hình chọn các chân PA0, PA1, PA2 thành các kênh input ADC IN0, IN1, IN2. Và các bạn đừng quên chọn SYS -> Serial Wire để cấu hình 2 chân nạp code thông qua mạch ST-Link V2.



Bước 2: Tại thẻ Configuration, các bạn chọn ADC1 để cấu hình cho chức năng ADC như hình dưới, trong ví dụ này mình **disable** Continuous Conversion Mode, sau đó trong code bên Keil C mình sẽ cho cập nhật ADC 0.5s một lần.

ADC1 Configuration
✕

✔ Parameter Settings
✔ User Constants
✔ NVIC Settings
✔ DMA Settings
✔ GPIO Settings

Configure the below parameters :

Search : ↕ ↕

[-] ADCs_Common_Settings	
Mode	Independent mode
[-] ADC_Settings	
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
[-] ADC_Regular_ConversionMode	
Enable Regular Conversions	Enable
Number Of Conversion	3
External Trigger Conversion Source	Regular Conversion launched by software
[+] Rank	1
[+] Rank	2
[+] Rank	3

Scan Conversion Mode

ScanConvMode

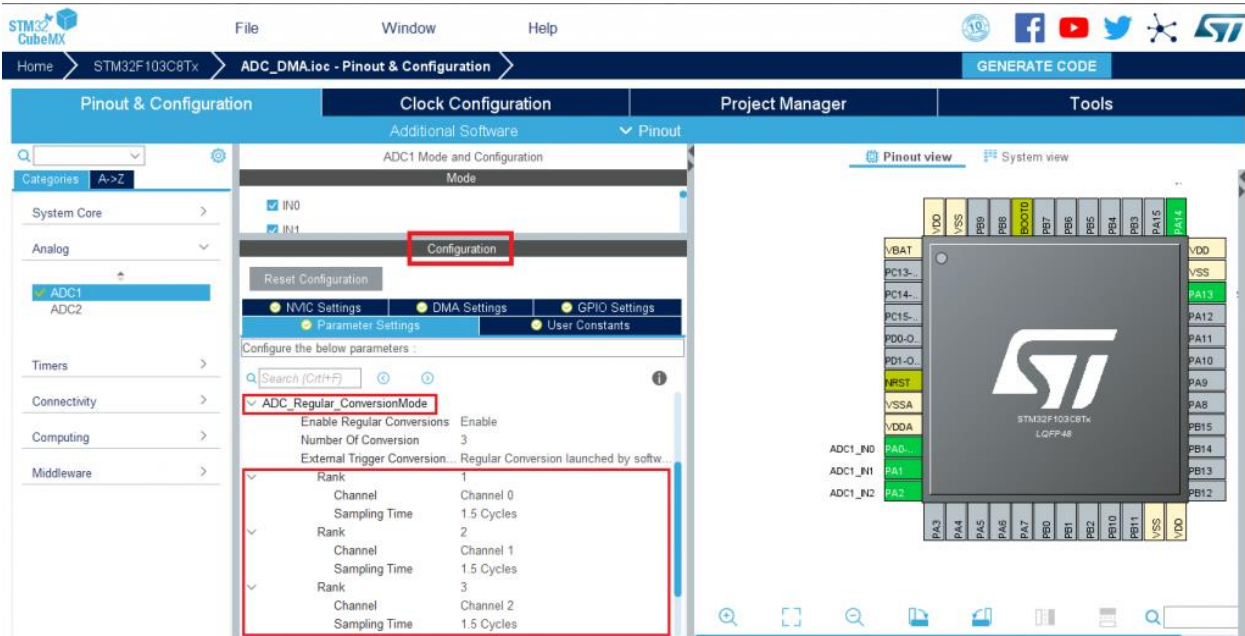
Parameter Description:

Enable or Disable the Scan Conversion Mode

Restore Default
Apply
Ok
Cancel

– Tại mục Rank, các bạn chọn thứ tự để đọc giá trị các kênh analog cho các channel mà bạn muốn, channel nào có rank 1 thì sẽ được đọc đầu tiên. Ở ví dụ này mình dùng 3 channel nên sẽ có 3 rank, channel PA0 có vị trí rank là 1, PA1 là rank 2 và PA3 là rank 3, tốc độ lấy mẫu của cả 3 kênh là 1.5 cycles. Lưu ý với tốc độ lấy mẫu là 1.5 cycles thì những câu lệnh trong hàm while(1) sẽ không được thực hiện vì thời gian lấy mẫu quá nhanh nên MCU sẽ thực hiện lấy mẫu ADC liên tục, vì vậy đối với các project

không đòi hỏi cập nhật giá trị ADC liên tục thì các bạn nên tăng thời gian lấy mẫu lên.



Bước 3: Trong thẻ Configuration, các bạn tiến hành thêm DMA cho ADC1 với cấu hình như bên dưới, lưu ý độ rộng dữ liệu là Half Word.

DMA Request	Channel	Direction	Priority
ADC1	DMA1 Channel 1	Peripheral To Memory	Low

Add Delete

DMA Request Settings

Mode: Circular

Increment Address: ☐

Data Width: Half Word

Peripheral: ☐ Memory: ☒

Half Word Half Word

Apply Ok Cancel

Sau khi cấu hình xong, các bạn sinh code và chọn Toolchain/IDE tương ứng, ở đây mình code bằng Keil C nên chọn **MDK ARM V5**. Sau khi sinh code xong, các bạn mở project Keil C lên và tiến hành thêm các bước sau:

Bước 1: Các bạn khai báo một biến mảng để nhận dữ liệu DMA từ ADC, có số lượng phần tử bằng số lượng kênh ADC mà các bạn muốn scan. Kiểu dữ liệu của biến này là uint16_t khớp với cấu hình độ rộng dữ liệu DMA half word đã cấu hình trên phần mềm CubeMX.

```

/* Private variables -----
ADC_HandleTypeDef hadc1;
DMA_HandleTypeDef hdma_adc1;

/* USER CODE BEGIN PV */
/* Private variables -----
uint16_t a[3];
//uint32_t adc_value0,adc_value1,adc_value2;
/* USER CODE END PV */

```

Bước 2: Tại vòng lặp vô hạn trong hàm main(), các bạn gọi bắt đầu chuyển đổi ADC_DMA và truyền vào các tham số như hình. Để hiểu rõ ý nghĩa của từng tham số, các bạn compile code sau đó nhấn chuột phải vào hàm và chọn Go to definition of “...” để xem.

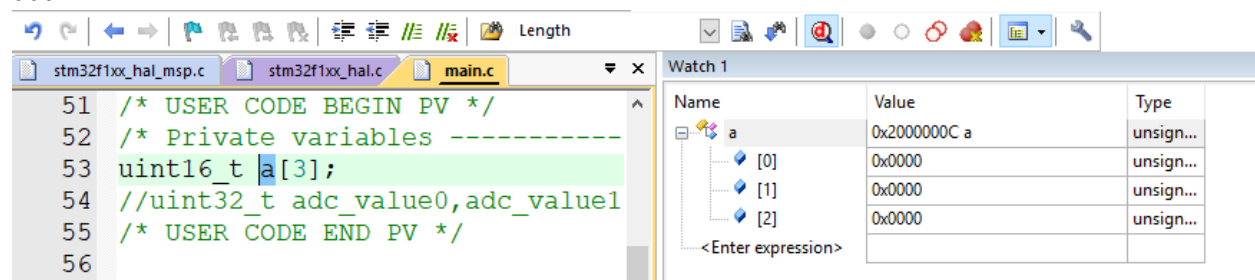
Các bạn thêm HAL_Delay(500); để có thời điểm dừng cho chúng ta quan sát kết quả cập nhật trên giao diện debug ở bước tiếp theo.

```
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    HAL_ADC_Start_DMA(&hadc1, (uint32_t*)a, 3);
    HAL_Delay(500);
}
/* USER CODE END 3 */
}
```

Cuối cùng, các bạn compile code và bật Debug sau đó bấm run để xem thành quả.

Các bạn kiểm tra bằng cách lần lượt cắm các chân PA0, PA1, PA2 lên nguồn trong khi 2 chân còn lại nối đất.



The screenshot shows an IDE with the following components:

- Code Editor:** Displays the `main.c` file with the following code:

```
51 /* USER CODE BEGIN PV */
52 /* Private variables -----
53 uint16_t a[3];
54 //uint32_t adc_value0,adc_value1
55 /* USER CODE END PV */
56
```
- Watch Window:** Titled "Watch 1", it contains a table with the following data:

Name	Value	Type
a	0x2000000C a	unsign...
[0]	0x0000	unsign...
[1]	0x0000	unsign...
[2]	0x0000	unsign...
<Enter expression>		