Chương 7

BỘ CHUYỂN ĐỔI TÍN HIỆU TƯƠNG TỰ SANG SỐ

ANALOG TO DIGITAL CONVERTER-ADC

7.1 GIỚI THIỆU ADC

ADC của họ ARM STM32 F1 có độ phân giải là 12 bit và có tất cả 18 kênh tương tự trong đó 16 kênh nhận tín hiệu bên ngoài và 2 kênh nhận tín hiệu bên trong để chuyển đổi.

Mỗi kênh ADC có thể được cấu hình độc lập để chuyển đổi tín hiệu tương tự sang số theo nhiều chế độ khác nhau như là chuyển đổi đơn(single), liên tục(continuous), quét(scan) hoặc không liên tục(discontinuous).

Đối với kết quả chuyển đổi ADC người dùng có thể cài đặt canh phải hoặc canh trái khi lưu vào thanh ghi dữ liệu ADC.

Bộ giám sát tín hiệu tương tự(analog watchdog) cho phép phát hiện việc điện áp vượt khỏi ngưỡng cao hoặc thấp do người dùng cài đặt.

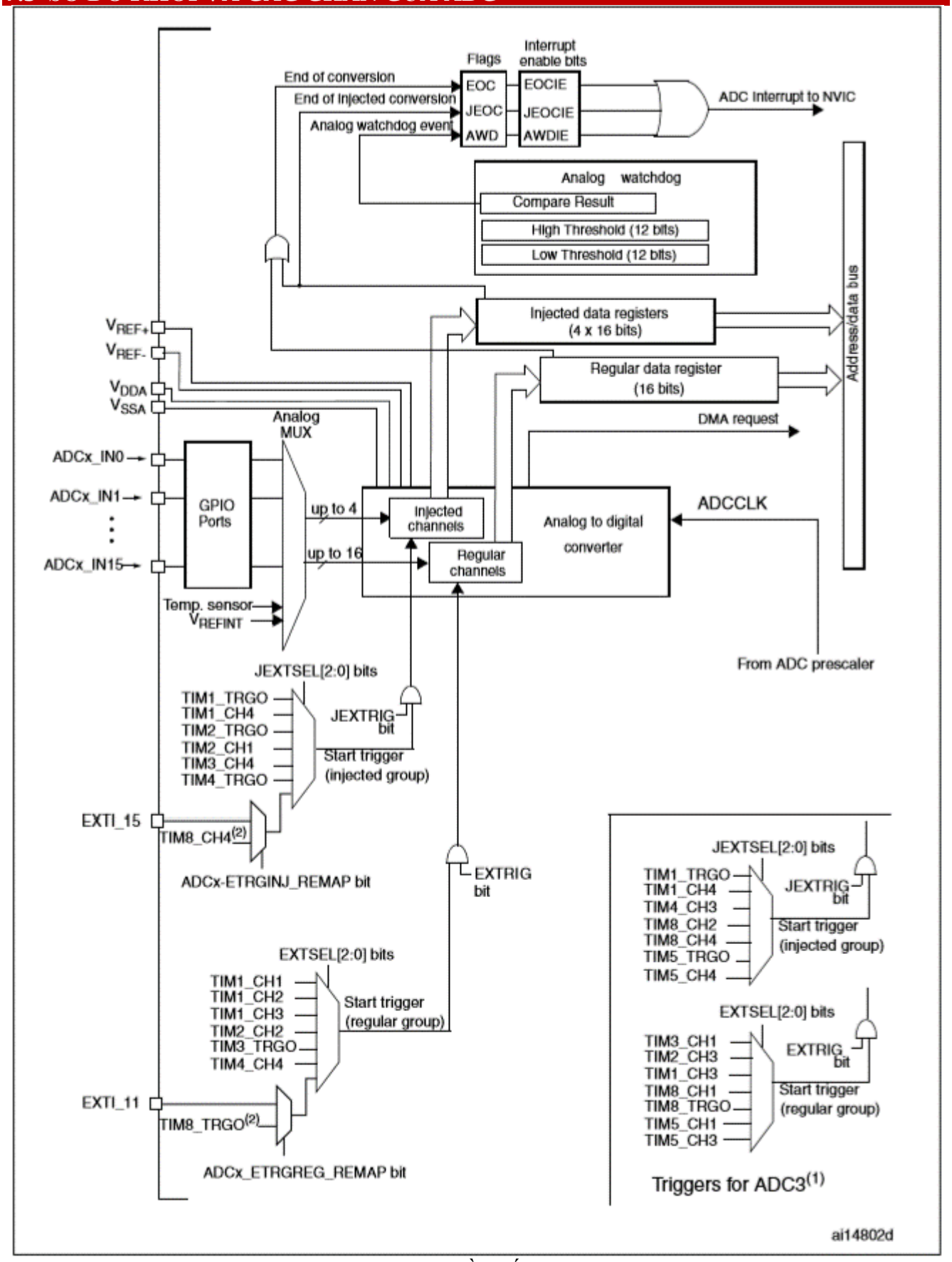
Xung clock cấp cho ADC được tạo ra bởi PLCK2 và đi qua một bộ chia sao cho khi cấp cho ADC tần số xung này không vượt quá 14 Mhz.

7.2 NHỮNG ĐẶC ĐIỂM CHÍNH CỦA ADC

- Độ phân giải 12 bit.
- Có khả năng tạo ra ngắt khi chuyển đổi hoàn tất, và khi bộ giám sát tín hiệu tương tự(Anglog Watchdog) phát hiện điện áp vào vượt quá ngưỡng cài đặt.
- Có thể hoạt động được ở chế độ chuyển đổi 1 lần(Single) hoặc liên tục(Continuous)
- Hỗ trợ chế độ quét(Scan) cho phép tự động chuyển đổi từ kênh 0 đến kênh "n".
- Có bộ hiệu chỉnh thông số cho phép kết quả chuyển đổi được chính xác.
- Dữ liệu sau khi chuyển đổi cho phép canh phải hoặc canh trái.
- Dữ liệu sau khi chuyển đổi cho phép canh phải hoặc canh trái
- Mỗi kênh có thể được lập trình thời gian lấy mẫu riêng
- Hỗ trợ tín hiệu yêu cầu chuyển đổi từ bên ngoài cho cả hai nhóm regular và injected
- Hỗ trợ chế độ không liên tục(Discontinuous mode)
- Có thể hoạt động ở chế độ song song(chỉ có trên các dòng ARM có từ 2 ADC trở lên)
- Thời gian chuyển đổi:
  - STM32F103xx: 1 µs khi hoạt động ở 56 MHz (1.17 µs tại 72 MHz)
  - STM32F101xx: 1 µs khi hoạt động ở 28 MHz (1.55 µs tại 36 MHz)
  - STM32F102xx: 1.2 µs khi hoạt động ở 48 MHz
  - STM32F105xx và STM32F107xx: 1 µs khi hoạt động ở 56 MHz (1.17 µs tại 72 MHz)
- Điện áp cấp cho ADC từ 2.4V đến 3.6V
- Điện áp tưng tự vào ADC: VREF- ≤ VIN ≤ VREF+
- Tạo tín hiệu yêu cầu DMA trong suốt quá trình chuyển đổi của kênh regular

Chú ý: Nếu dòng ARM đang sử dụng có chân VREF- thì phải nối chân này với VSSA

7.3 SƠ ĐỒ KHỐI VÀ CÁC CHÂN CỦA ADC



Hình 7.1 Sơ đồ khối của ADC

Bảng 7.1 Chức năng các chân của ADC

Tên chân	Chức năng	Chú ý
V_{REF+}, V_{REF-}	Ngõ vào cấp điện áp tham chiếu cho ADC	2.4 V ≤ V_{REF+} ≤ VDDA V_{REF-} = VSS
V_{DDA}, V_{SSA}	Nguồn cấp cho	2.4 V ≤ VDDA ≤ 3.6 V VSSA = VSS
ADCx_IN[15:0]	Ngõ vào tín hiệu tương tự	

16 kênh tương tự ngõ vào được chia ra làm 2 nhóm là regular và injected.

- Nhóm regular có thể được cài đặt tối đa 16 yêu cầu chuyển đổi.
- Nhóm injected có thể được cài đặt tối đa 4 yêu cầu chuyển đổi.

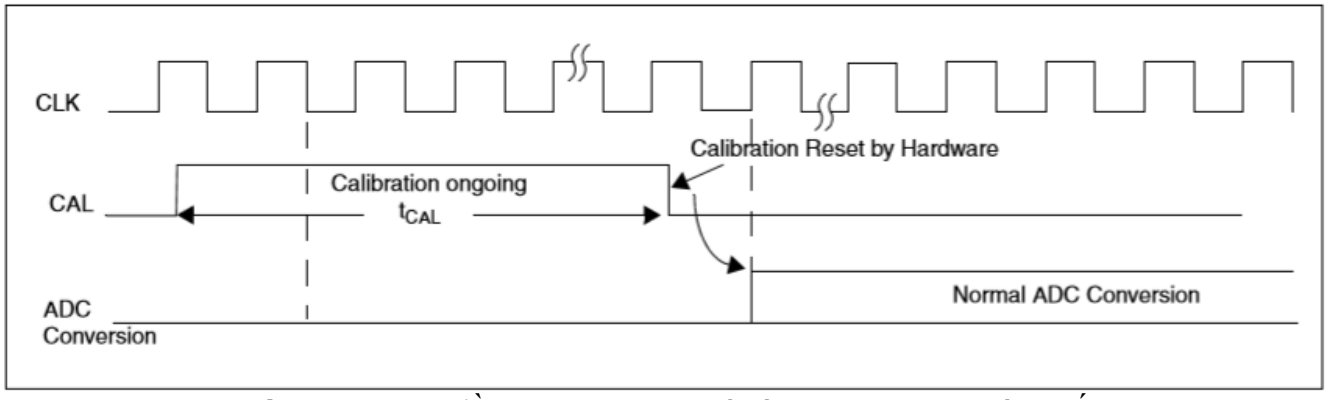
Số kênh và thứ tự chuyển đổi của các kênh trong 2 nhóm regular và injected được cấu hình bằng phần mềm. Nếu những cấu hình này bị thay đổi trong khi ADC đang chuyển đổi thì công việc chuyển đổi hiện tại sẽ bị reset và đến khi có tín hiệu yêu cầu chuyển đổi tiếp theo thì ADC sẽ chuyển đổi theo cấu hình mới.

Cảm biến đo nhiệt độ bên trong vi điều khiển được nối với ngõ vào tương tự thứ 16 (ADCx_IN16). Điện áp tham chiếu nội được nối với kênh 17(ADCx_IN17). Cả 2 kênh này có thể cấu hình hoạt động theo nhóm injected hay regular đều được. Khi sử dụng 2 kênh này ta cần chú ý là phải sử dụng ADC1 vì chỉ ADC1 mới kết nối với chúng.

7.4 HIỆU CHỈNH THÔNG SỐ(CALIBRATION)

ADC được xây dựng 1 chế độ hiệu chỉnh thông số. Hiệu chỉnh thông số là công việc đặc biệt quan trọng để giảm sai số. Khi quá trình hiệu chỉnh thông số xảy ra thì bit CAL trong thanh thi ADC_CR2 lên ‘1’ và khi quá trình này kết thúc thì phần cứng sẽ tự động reset bit CAL và ADC trở về chế độ hoạt động bình thường.

Chú ý: Hiệu chỉnh thông số là một bước quan trọng và cần được thực hiện lại sau mỗi lần cấp nguồn. ADC phải được bật(bit ADON = ‘1’)tối thiểu 2 chu kỳ ADCCLK trước khi hiệu chỉnh.



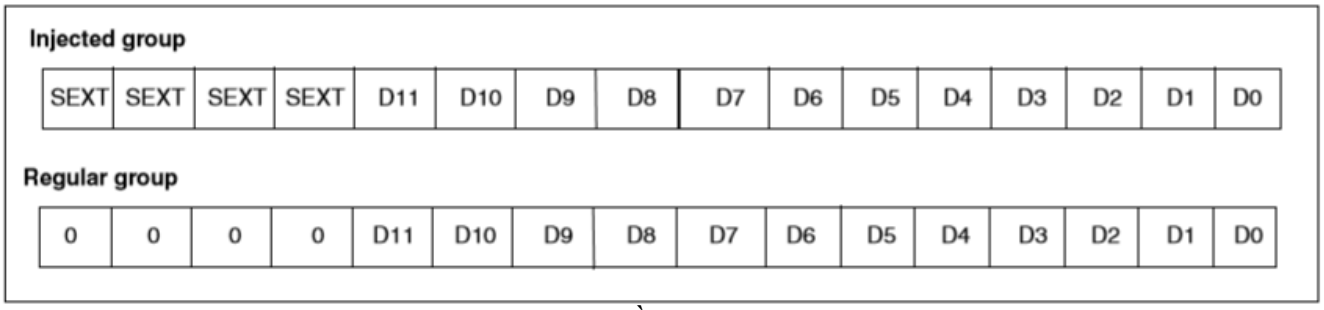
Hình 7.2 Giản đồ thời gian của quá trình hiệu chỉnh thông số

7.5 CANH LỀ KẾT QUẢ CHUYỂN ĐỔI ADC(DATA ALIGNMENT)

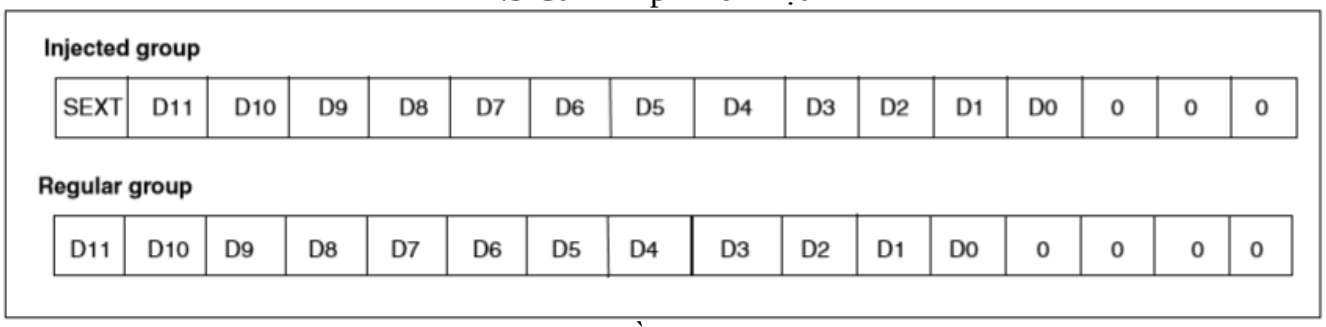
Dữ liệu sau khi chuyển đổi có thể được canh lề trái hoặc phải dựa vào bit ALIGN trong thanh ghi ADC_CR2.

Đối với các kênh nhóm injected dữ liệu sau chuyển đổi có thể được trừ đi một lượng do người lập trình cài đặt vào thanh ghi ADC_JOFRx vì vậy kết quả chuyển đổi có thể là số âm và bit SEXT đóng vai trò là bit dấu.

Đối với các kênh thuộc nhóm regular do kết quả chuyển đổi không bị trừ như nhóm injected nên luôn là số dương và chỉ chứa trong 12 bit.



Hình 7.3 Canh lề phải dữ liệu



Hình 7.4 Canh lề trái dữ liệu

7.6 LẬP TRÌNH THỜI GIAN LẤY MẪU CHO TỪNG KÊNH

Thời gian lấy mẫu điện áp đưa vào kênh tương tự của ADC có thể được cấu hình theo bội số của chu kỳ ADCCLK và việc cấu hình này được thực hiện bởi các bit SMP[2:0] trong thanh ghi ADC_SMPR1 và ADC_SMPR2. Mỗi kênh có thể được cấu hình thời gian lấy mẫu khác nhau.

Ta có công thức tính tổng thời gian chuyển đổi(Tconv) như sau:

Tconv = thời gian lấy mẫu + 12.5 chu kỳ ADCCLK

Ví dụ: Chọn ADCCLK =14 Mhz, chọn tốc độ lấy mẫu là 1.5 chu kỳ ADCCLK

Tconv = 1.5 + 12.5 =14 chu kỳ ADCCLK = 1 µs

7.7 KÍCH HOẠT CHUYỂN ĐỔI BẰNG TÍN HIỆU TỪ BÊN NGOÀI

Việc chuyển đổi có thể được kích hoạt bằng một sự kiện bên ngoài. Nếu bit điều khiển EXT- TRIG được set thì khi phát sinh các sự kiện từ bên ngoài có thể làm cho quá trình chuyển đổi bắt đầu. Các bit EXT- SEL[2:0] và JEXTSEL[2:0] cho phép chọn nguồn sự kiện kích hoạt cho nhóm regular và injected.

Chú ý: Khi cấu hình ADC regular hay injected chuyển đổi theo nguồn tín hiệu kích hoạt bên ngoài thì chỉ có cạnh lên của tín hiệu này mới có thể cho phép bắt đầu chuyển đổi.

7.8 YÊU CẦU DMA

Do dữ liệu chuyển đổi của nhiều kênh regular chỉ được lưu trữ trong 1 thanh ghi duy nhất là ADC_DR nên ta cần phải sử dụng DMA để chuyển các kết quả chuyển đổi này vào nơi lưu trữ khác được định bởi người dùng để tránh mất dữ liệu.

Chỉ khi quá trình chuyển đổi kết thúc thì tín hiệu yêu cầu DMA mới được tạo ra.

Chú ý: Chỉ có ADC1 và ADC3 mới có thể sử dụng DMA còn ADC2 thì chỉ có thể chuyển dữ liệu thông qua đường DMA của ADC chủ(ADC1) khi hoạt động ở chế độ song song.

7.9 CẢM BIẾN ĐO NHIỆT ĐỘ NỘI

Cảm biến đo nhiệt độ được sử dụng để đo nhiệt độ CPU.

Ngõ ra của cảm biến đo nhiệt độ đã được kết nối sẵn bên trong với kênh số 16 của ADC(ADCx_IN16) giúp chuyển điện áp tương tự trả về từ cảm biến thành giá trị số.

Đối với cảm biến đo nhiệt độ ta nên chọn thời gian lấy mẫu là 17.1 µs.

Mặc định cảm biến đo nhiệt độ được tắt đi để tiết kiệm năng lượng do đó trước khi sử dụng ta phải bật lên.

Chý ý:

- Bit TSVREFE phải được set để cho phép các kênh tương tự nội ADCx_IN16 (cảm biến đo nhiệt độ) và ADCx_IN17 (VREFINT) chuyển đổi.
- Ngõ ra của cảm biến đo nhiệt độ là giá trị điện áp tuyến tính với nhiệt độ đo được.

7.9.1 Các bước đo nhiệt độ

- Chọn kênh số 16
- Chọn thời gian lấy mẫu là 17.1 µs
- Set bit TSVREFE trong thanh ghi ADC_CR2 để cho phép cấp nguồn cho cảm biến đo nhiệt độ.
- Cho phép chuyển đổi bằng cách set bit ADON hoặc sử dụng sự kiện kích hoạt bên ngoài.
- Đọc kết quả chuyển đổi được trong thanh ghi dữ liệu ADC
- Tính toán giá trị nhiệt độ thông qua công thức 1

Nhiệt độ(°C) = {(V25 - VSENSE) / Avg_Slope} + 25	(1)

Trong đó:

V25: Giá trị điện áp trả về của cảm biến tại 25 °C
(chọn V25 = 1.43 V khi điện áp tham chiếu là 3.3 V)

VSENSE: Giá trị điện áp trả về của cảm biến ở nhiệt độ hiện tại

Avg_Slope: độ dốc trung bình của đồ thị quan hệ gữa nhiệt độ và giá trị chuyển đổi (đơn vị mV/° C hoặc µV/ °C)
(chọn Avg_Slope = 0.0043 V/° C khi điện áp tham chiếu là 3.3 V)

Vậy ta có:

Nhiệt độ(°C) = {(1.43 - VSENSE) / 0.0043} + 25	(2)

7.10 Các chế độ hoạt động cơ bản của ADC

7.10.1 Chế độ chuyển đổi đơn(Single Conversion Mode)

Trong chế độ chuyển đổi đơn ADC chỉ thực hiện 1 chuyển đổi và khi hoàn tất thì:

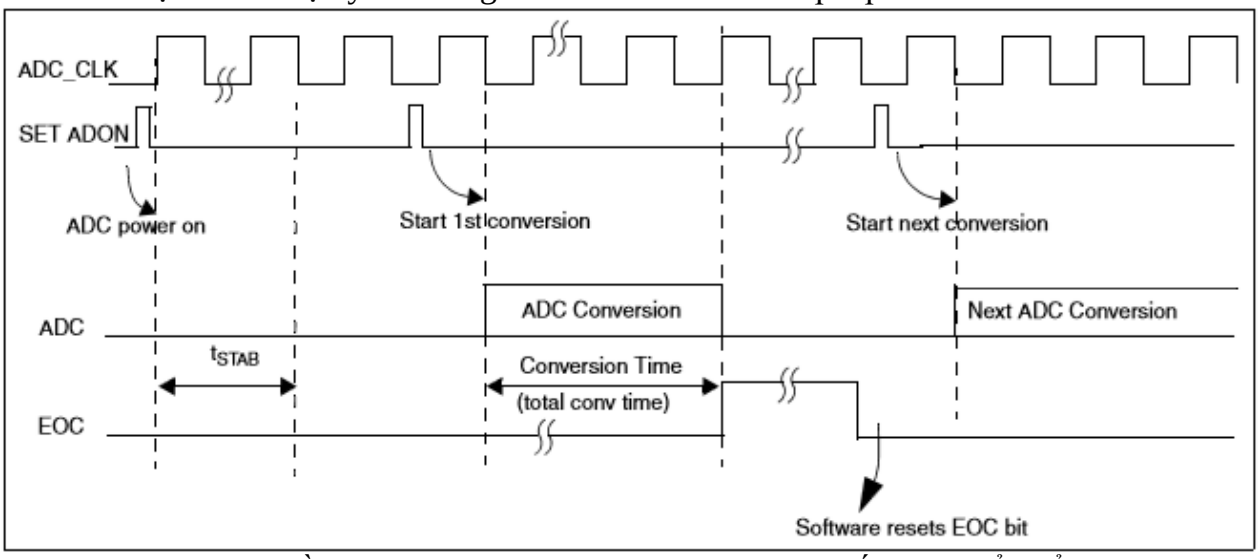
- Nếu đang sử dụng nhóm regular thì kết quả chuyển đổi sẽ được lưu trong thanh ghi 16 bit ADC_DR, cờ EOC(End Of Conversion) được set lên ‘1’ và tạo ra tín hiệu yêu cầu ngắt nếu trước đó đã cho phép báo ngắt khi chuyển đổi hoàn tất.
- Nếu đang sử dụng nhóm injected thì kết quả chuyển đổi sẽ đươc lưu trong các thanh ghi 16 bit ADC_DRJ**x**, cờ JEOC(End Of Conversion Injected) được set lên ‘1’ và đồng thời tạo ra tín hiệu yêu cầu ngắt nếu trước đó đã cho phép báo ngắt khi chuyển đổi hoàn tất
- Sau khi thực hiện xong các công việc trên ADC sẽ dừng hoạt động.

7.10.2 Chế độ liên tục(Continuous Mode)

Trong chế độ liên tục ADC sẽ bắt đầu chuyển đổi mới ngay khi vừa hoàn tất chuyển đổi trước đó. Sau mỗi lần chuyển đổi xong thì:

- Nếu đang sử dụng nhóm regular thì kết quả chuyển đổi sẽ được lưu trong thanh ghi 16 bit ADC_DR, cờ EOC(End Of Conversion) được set lên ‘1’ và tạo ra tín hiệu yêu cầu ngắt nếu trước đó đã cho phép báo ngắt khi chuyển đổi hoàn tất.

- Nếu đang sử dụng nhóm injected thì kết quả chuyển đổi sẽ được lưu trong các thanh ghi 16 bit ADC_DRJ**x**, cờ JEOC(End Of Conversion Injected) được set lên '1' và đồng thời tạo ra tín hiệu yêu cầu ngắt nếu trước đó đã cho phép.



Hình 7.5 Giản đồ thời gian của ADC khi hoạt động ở chế độ chuyển đổi liên tục

7.10.3 Chế độ quét(Scan mode)

Chế độ quét đươc sử dụng để quét 1 nhóm kênh tương tự. Trong chế độ này, ADC sẽ quét tất cả các kênh đã chọn và cho phép từng kênh chuyển đổi, ngay sau khi mỗi kênh chuyển đổi xong kênh kế tiếp sẽ tự động bắt đầu chuyển đổi. Nếu chế độ chuyển đổi liên tục được chọn thì sau khi chuyển đổi xong kênh cuối cùng của nhóm ADC sẽ quay về chuyển đổi lại từ kênh đầu.

Khi ADC hoạt động ở chế độ quét ta cần sử dụng kết hợp với DMA để cho phép chuyển nhanh kết quả chuyển đổi được lưu trong thanh ghi ADC_DR(nếu sử dụng kênh regular) tới SRAM ngay khi thanh ghi này cập nhật giá trị mới. Đối với các kênh injected thì kết quả chuyển đổi luôn được lưu trong các thanh ghi ADC_JDRx.

7.10.4 Chế độ không liên tục(Discontinuous mode)

a. Đối với nhóm regular:

Chế độ này có thể được dùng để thực hiện một chuỗi **n** chuyển đổi ngắn (**n** <=8). Giá trị của **n** được cấu hình bằng phần mềm thông qua các bit DISCNUM[2:0] của thanh ghi ADC_CR1. Mỗi khi phát sinh một tín hiệu yêu cầu chuyển đổi từ bên ngoài ADC sẽ bắt đầu thực hiện **n** chuyển đổi mới và khi **n** chuyển đổi này hoàn tất thì bit EOC được set lên '1' để báo hiệu đã chuyển đổi xong.

Nếu ADC đã chuyển đổi xong hết tất cả các kênh được chọn mà vẫn tiếp tục nhận được xung kích hoạt thì ADC sẽ quay trở về thực hiện lại **n** kênh đầu tiên. Tổng số kênh cần chuyển đổi được cấu hình bởi các bit L[3:0] trong thanh ghi ADC_SQR1.

Ví dụ: n=3, tổng số kênh cần chuyển đổi là 8 và các kênh cần chuyển đổi là: 0, 1, 2,3, 6, 7, 9, 10.

Khi nhận được tín hiệu yêu cầu chuyên đổi đầu tiên thì ADC sẽ chuyển đổi các kênh 0, 1, 2 (do n=3 nên chuyển đổi 3 kênh). Ở tín hiệu yêu cầu thứ 2 thì ADC sẽ chuyển đổi các kênh 3, 6, 7 và ở tín hiệu yêu cầu lần 3 thì các kênh đươc chuyển đổi là 9,10. Như vậy sau 3 tín hiệu yêu cầu ADC đã chuyển đổi xong hết tất cả các kênh được chọn lúc này nếu xảy ra tín hiệu yêu cầu chuyển đổi thứ 4 thì ADC sẽ trở về chuyển đổi lại các kênh 0, 1, 2.

b. Đối với nhóm injected:

Chế độ này được dùng để chuyển đổi từng kênh một trong nhóm injected sau mỗi lần nhận được tín hiệu yêu cầu chuyển đổi từ bên ngoài.

Mỗi khi nhận được tín hiệu kích hoạt từ bên ngoài ADC sẽ tiến hành chuyển đổi kênh tiếp theo trong nhóm kênh injected được lựa chọn bởi thanh ghi ADC_JSQR. Sau khi chuyển đổi xong tất cả các kênh đã chọn thì cờ JEOC được set lên '1' và lúc này nếu ADC nhận được xung kích hoạt tiếp theo thì ADC sẽ quay về chuyển đổi kênh đầu tiên trong nhóm kênh đã chọn. Tổng số kênh cần chuyển đổi được cấu hình bằng bit JL[1:0] trong thanh ghi ADC_JSQR

Ví dụ: tổng số kênh cần chyển đổi là 3 và các kênh đó là: 1, 2, 3. Xung kích hoạt đầu tiên ADC sẽ chuyển đổi kênh 1, xung thứ hai là kênh 2 và xung thứ ba là kênh 3. Đến lúc này ADC đã chuyển đổi xong hết các kênh đã chọn nên cờ JEOC được set lên '1'. Khi có tín hiệu yêu cầu chuyển đổi thứ tư ADC sẽ chuyển đổi lại kênh 1.

Chú ý: Không nên cài đặt chế độ không liên tục cho cả nhóm regular và injected cùng lúc.

7.10.5 Chế độ kích hoạt chuyển đổi các kênh thuộc nhóm injected(Triggered injection)

Để có thể sử dụng được chế độ này thì thi bit JAUTO phải được xóa đồng thời bit SCAN trong thanh ghi ADC_CR1 phải được set ở mức '1'.

Khi có tín hiệu yêu cầu chuyển đổi thì các kênh thuộc nhóm regular sẽ được chuyển đổi.

Khi có tín hiệu từ bên ngoài yêu cầu chuyển đổi các kênh thuộc nhóm injected thì chuyển đổi hiện tại sẽ bị reset và các kênh thuộc nhóm injected được chuyển đổi theo chế độ quét.

Sau khi tất cả các kênh thuộc nhóm injected được chuyển đổi xong thì các kênh thuộc nhóm regular mới được phép bắt đầu được chuyển đổi tiếp tục tại vị trí bị gián đoạn.

Chú ý: Thời gian giữa 2 lần gởi tín hiệu từ bên ngoài yêu cầu các kênh thuộc nhóm injected chuyển đổi phải lớn hơn thời gian của toàn bộ chu trình chuyển đổi nhóm kênh này.

7.10.6 Chế độ tự động chuyển đổi các kênh thuộc nhóm injected(Auto-injection)

Để có thể sử dụng chế độ này ta cần set bit JAUTO ở mức '1'

Chế độ này cho phép các kênh thuộc nhóm injected tự động chuyển đổi ngay khi các kênh thuộc nhóm regular chuyển đổi xong.

Chế độ này cho phép mở rộng chu trình lên đến tối đa 20 chuyển đổi.

Tín hiệu yêu cầu các kênh thuộc nhóm injected chuyển đổi phải được tắt khi hoạt động ở chế độ này

Chú ý: Chế độ này không được phép sử dụng đồng thời với chế độ không liên tục.

7.10.7 Chế độ giám sát điện áp tương tự(Analog watchdog)

Bit trạng thái AWD của bộ giám sát điện áp tương tự được set ở mức '1' nếu kết quả chuyển đổi của ADC thấp hơn ngưỡng dưới hoặc cao hơn ngưỡng trên đã được lập trình trước đó trong 2 thanh ghi 16 bit ADC_LTR và ADC_HTR.

Nếu trước đó đã cho phép ngắt bằng cách set bit AWDIE trong thanh thi ADC_CR1 thì yêu cầu ngắt sẽ được tạo ra khi bộ giám sát điện áp tương tự phát hiện điện áp vượt ngưỡng cài đặt.

Giá trị ngưỡng cài đặt không phụ thuộc vào chế độc canh phải hoặc canh trái vì ADC đã so sánh kết quả chuyển đổi trước khi thực hiện canh lề.

Bộ giám sát điện áp tương tự có thể được bật trên 1 hoặc nhiều kênh bằng cách cấu hình theo **bảng 7.2**:

Bảng 7.2 Các chế độ hoạt động của bộ giám sát điện áp tương tự

Các kênh được giám sát bởi AWD	Các bit trong thanh ghi ADC_CR1 (x = bất chấp)		
	Bit AWDGSL	Bit AWDEN	Bit JAWDEN
Không có kênh nào	x	0	0
Tất cả các kênh thuộc nhóm injected	0	0	1

Tất cả các kênh thuộc nhóm regular	0	1	0
Tất cả các kênh injected và reguarl	0	1	1
Một kênh injected	1	0	1
Một kênh regular	1	1	0
Một kênh injected hoặc regular	1	1	1

Chú ý: Khi chọn chế độ giám sát 1 kênh ta cần phải xác định kênh đó qua các bit AWDCH[4:0]

7.10.8 Chế độ hoạt động song song 2 ADC(Dual ADC mode)

(chỉ hoạt động đối với các dòng ARM có từ 2 ADC trở lên)

Trong chế độ song song 2 ADC tín hiệu yêu cầu chuyển đổi được kích hoạt luân phiên hoặc đồng thời bởi ADC chủ(ADC1) tới ADC tớ(ADC2) phụ thuộc vào chế độ hoạt động được chọn bởi bit DUALMOD[2:0] trong thanh ghi ADC1_CR1.

Chú ý:

- Ở chế độ hoạt động song song khi cấu hình tín hiệu yêu cầu chuyển đổi từ bên ngoài người dùng chỉ cài đặt tín hiệu này cho ADC chủ còn ADC tớ thì cấu hình tín hiệu yêu cầu bằng phần mềm để ngăn chặn việc ADC tớ chuyển đổi ngoài ý muốn.
- Phải cho phép cả 2 ADC chủ và tớ được nhận tín hiệu yêu cầu chuyển đổi từ bên ngoài.

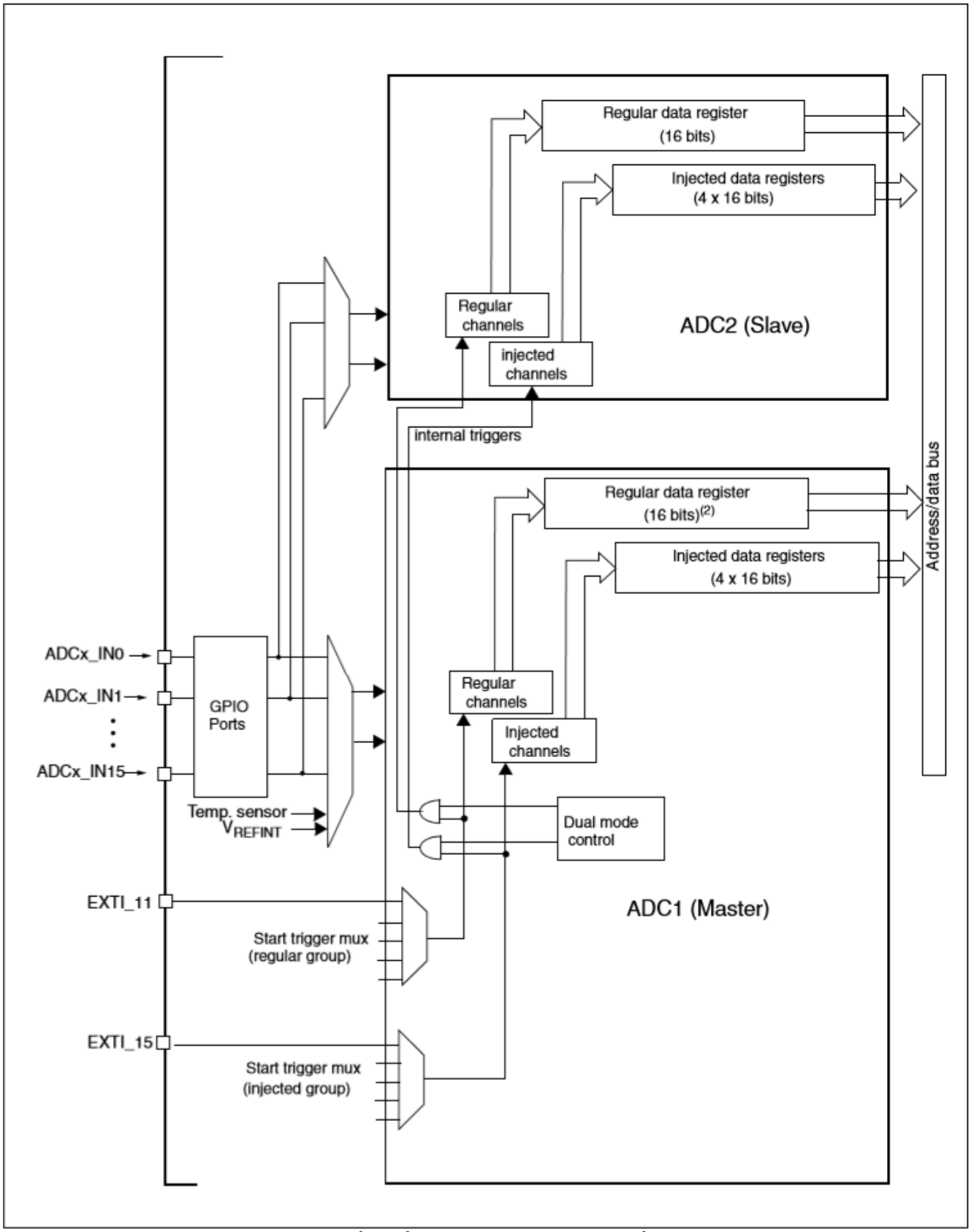
Có 6 chế độ có thể cấu hình cho ADC hoạt động ở chế độ song song:

- Injected đồng thời (Injected simultaneous mode)
- Regular đồng thời(Regular simultaneous mode)
- Xen kẽ nhanh(Fast interleaved mode)
- Xen kẽ chậm(Low interleaved mode)
- Kích hoạt chuyển đổi luân phiên(Alternate trigger mode)
- Độc lập giữa 2 ADC(Independent mode)

Có thể kết hợp 6 chế độ trên để tạo thành các chế độ khác như:

- Injected đồng thời + Regular đồng thời
  (Injected simultaneous mode + Regular simultaneous mode)
- Regular đồng thời + kích hoạt chuyển đổi luân phiên
  (Regular simultaneous mode + Alternate trigger mode)
- Injected đồng thời + xen kẽ
  (Injected simultaneous mode + Interleaved mode)

Chú ý: trong chế độ song song để đọc được dữ liệu chuyển đổi của ADC tớ trong thanh ghi dữ liệu của ADC chủ thì cần phải cho phép DMA ngay cả khi nó không được sử dụng để vận chuyển kết quả chuyển đổi của nhóm regular.



Hình 7.6 Sơ đồ khối ADC hoạt động ở chế độ song song

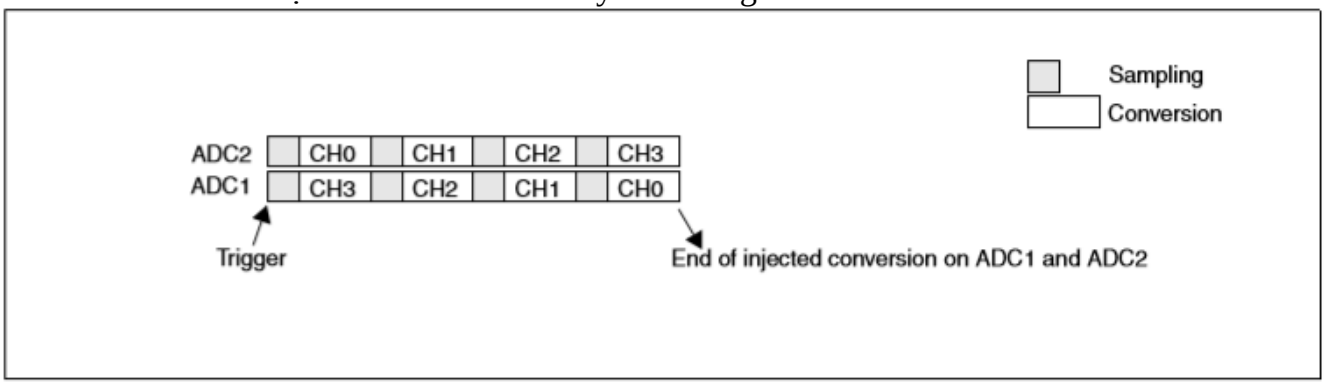
a. Chế độ injected đồng thời

Chế độ này cho phép cả 2 ADC chuyển đổi một nhóm các kênh injected cùng một lúc nhưng 2 ADC này không được chuyển đổi cùng một kênh tại cùng một thời điểm. Sau khi chuyển đổi xong kết quả chuyển đổi sẽ được lưu trong thanh ghi ADC_JDRx của mỗi ADC và tín hiệu báo

ngắt JEOC được tạo ra nếu trước đó đã cho phép ngắt ở 1 trong 2 ADC khi tất cả các kênh thuộc nhóm injected được chọn đã chuyển đổi xong.

Nếu trước đó đã cho phép ngắt ở 1 trong 2 ADC thì sau khi tất cả các kênh thuộc nhóm injected chuyển đổi xong sẽ tạo ra yêu cầu ngắt.

Chú ý: trong chế độ injected đồng thời người dùng cần cấu hình chính xác thời gian lấy mẫu cho 2 kênh mà sẽ được ADC1 và ADC2 lấy mẫu cùng lúc.



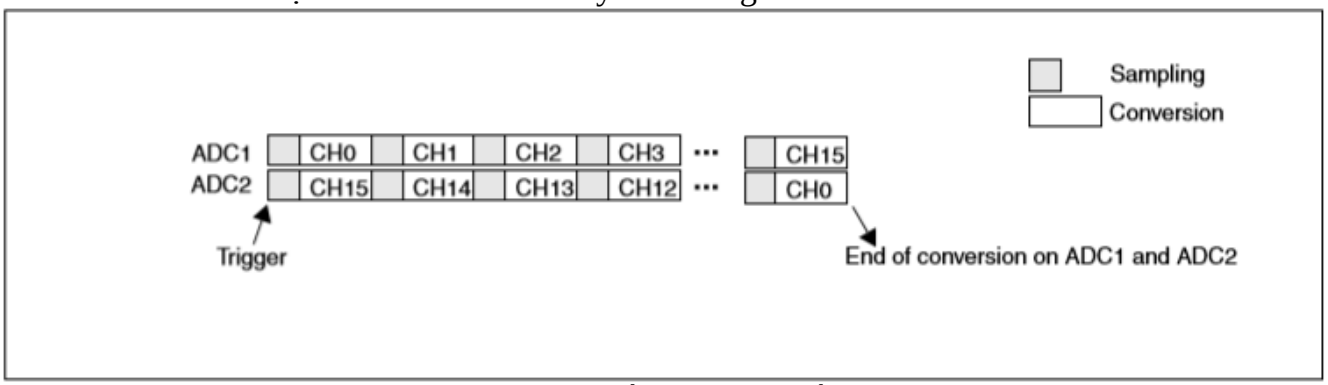
Hình 7.7 Chế độ injected đồng thời

b. Chế độ regular đồng thời

Chế độ này cho phép cả 2 ADC chuyển đổi một nhóm kênh regular cùng một lúc nhưng không được chuyển đổi cùng một kênh tại cùng một thời điểm. Sau mỗi lần chuyển đổi xong ADC sẽ tạo ra yêu cầu DMA 32 bit(nếu đã cho phép DMA) để chuyển dữ liệu từ thanh ghi 32 bit ADC1_DR(16 bit cao chứa kết quả chuyển đổi của ADC2 và 16 bit thấp chứa kết quả chuyển đổi của ADC1) tới SRAM.

Khi tất cả các kênh regular được chọn của ADC1, ADC2 đã chuyển đổi xong thì tín hiệu ngắt EOC được tạo ra nếu trước đó đã cho phép ngắt ở 1 trong 2 ADC .

Chú ý: trong chế độ regular đồng thời người dùng cần cấu hình chính xác thời gian lấy mẫu cho 2 kênh mà sẽ được ADC1 và ADC2 lấy mẫu cùng lúc.



Hình 7.8 Chế độ regular đồng thời

c. Chế độ xen kẽ nhanh

Chế độ này có thể hoạt động trên 1 kênh thuộc nhóm regular. Khi có tín hiệu yêu cầu chuyển đổi từ bên ngoài thì:

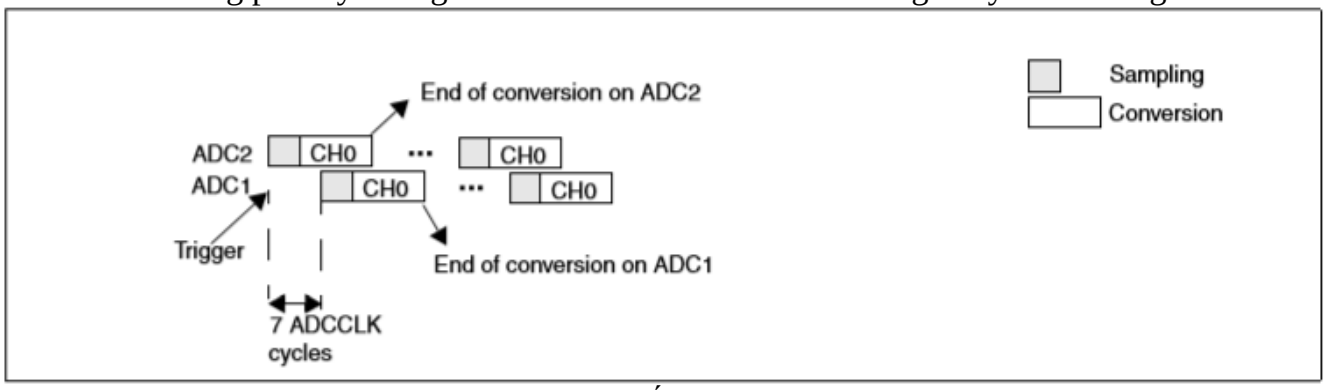
- ADC2 sẽ chuyển đổi ngay lập tức
- ADC1 chuyển đổi ngay sau đó 7 chu kỳ xung ADC

Nếu bit CONT được set trên cả 2 ADC thì những kênh đã được chọn trên các ADC này sẽ hoạt động ở chế độ chuyển đổi liên tục.

Sau khi tín hiệu báo ngắt EOC được tạo ra bởi ADC1(nếu trước đó đã cho phép ngắt) thì tín hiệu yêu cầu DMA 32 bit cũng được tạo ra(nếu trước đó đã cho phép DMA). DMA sẽ

chuyển dữ liệu 32 bit từ thanh ghi ADC1_DR(trong đó 16 bit cao là giá trị chuyển đổi của ADC2 và 16 bit thấp là của ADC1) vào SRAM.

Chú ý: Giá trị lớn nhất của thời gian lấy mẫu phải bé hơn 7 chu kỳ xung ADC(ADCCLK) để tránh trùng pha lấy mẫu giữa ADC1 và ADC2 khi mà chúng chuyển đổi cùng 1 kênh.



Hình 7.9 Chế độ xen kẽ nhanh

d. Chế độ xen kẽ chậm

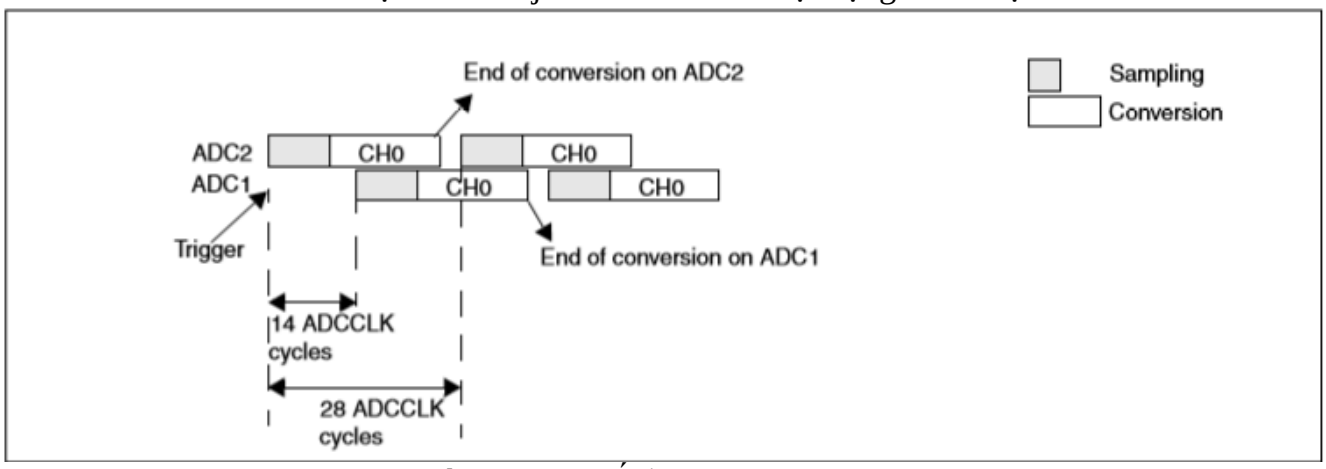
Chế độ này chỉ hoạt động trên 1 kênh thuộc nhóm regular. Khi có tín hiệu yêu cầu chuyển đổi từ bên ngoài thì:

- ADC2 sẽ chuyển đổi ngay lập tức.
- ADC1 chuyển đổi ngay sau đó 14 chu kỳ xung ADC.
- Sau 14 chu kỳ xung ADC kế tiếp ADC2 lại bắt đầu chuyển đổi, và cứ tiếp tục như thế(có nghĩa là chu kỳ chuyển đổi của mỗi ADC là 28 xung ADC)

Sau khi tín hiệu báo ngắt EOC được tạo ra bởi ADC1(nếu trước đó đã cho phép ngắt) thì tín hiệu yêu cầu DMA 32 bit cũng được tạo ra(nếu trước đó đã cho phép DMA). DMA sẽ chuyển dữ liệu 32 bit từ thanh ghi ADC1_DR vào SRAM.

Chú ý:

- Thời gian lấy mẫu phải bé hơn 14 ADCCLK để tránh trùng lấp với chuyển đổi kế.
- Người lập trình cần bảo đảm là sẽ không xảy ra yêu cầu chuyển đổi từ bên ngoài cho các kênh thuộc nhóm injected khi ADC hoạt động ở chế độ xen kẽ.



Hình 7.10 Chế độ xen kẽ chậm

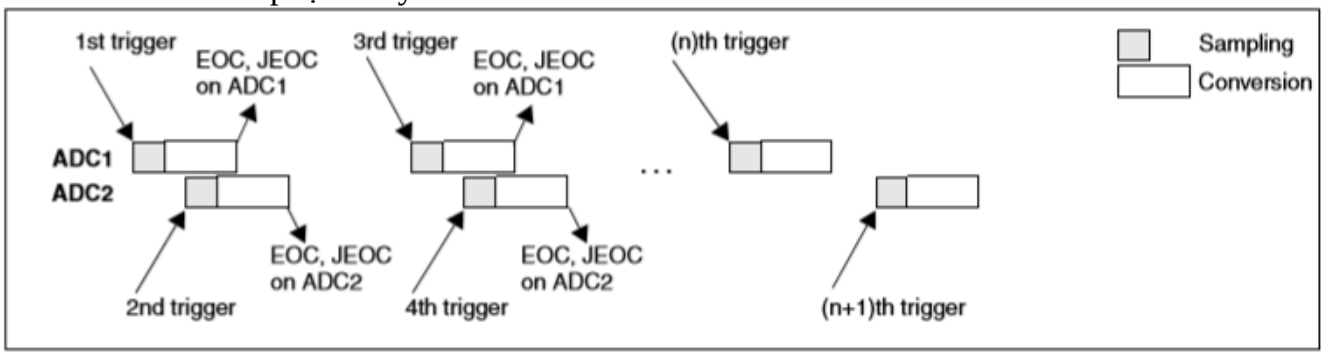
e. Chế độ kích hoạt chuyển đổi luân phiên

Chế độ này chỉ hoạt động trên nhóm injected.

Khi nhận được yêu cầu chuyển đổi thứ nhất từ bên ngoài thì tất cả các kênh thuộc nhóm injected trong ADC1 chuyển đổi.

Khi nhận được yêu cầu chuyển đổi thứ hai từ bên ngoài thì tất cả các kênh thuộc nhóm injected trong ADC2 chuyển đổi và cứ tiếp tục như thế

Tín hiệu yêu cầu ngắt JEOC được tạo ra(nếu trước đó đã cho phép) khi tất cả các kênh injected của ADC2 được chuyển đổi xong. Nếu vẫn tiếp tục nhận được yêu cầu chuyển đổi khi mà tất cả các kênh injected được chọn đã thực hiện xong thì chế độ kích hoạt luân phiên được restart và ADC1 tiếp tục chuyển đổi.

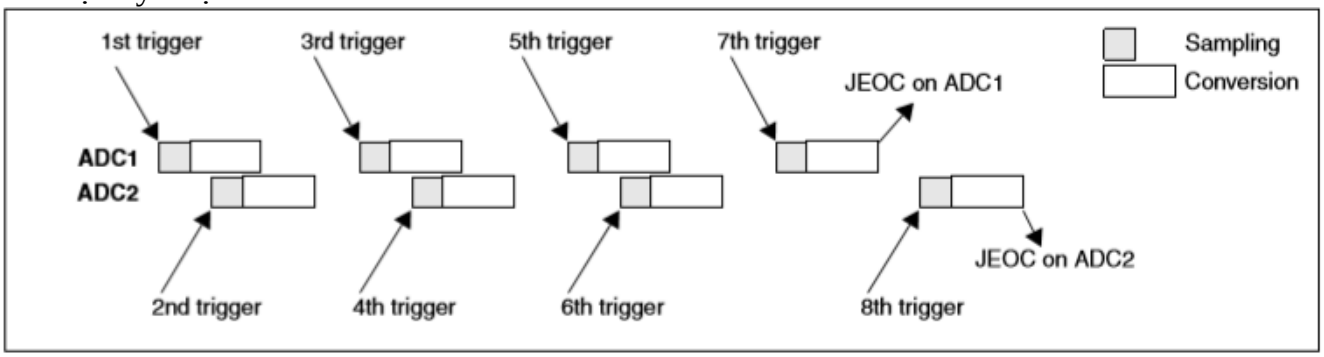


Hình 7.11 Chế độ kích hoạt chuyển đổi luân phiên

Nếu cả 2 ADC đều được cấu hình hoạt động ở chế độ injected không liên tục thì:

- Khi xuất hiện tín hiệu yêu cầu chuyển đổi thứ nhất ADC1 sẽ chuyển đổi kênh đầu tiên thuộc nhóm injected.
- Khi có tín hiệu yêu cầu chuyển đổi thứ hai thì ADC2 sẽ sẽ chuyển đổi kênh đầu tiên thuộc nhóm injected và cứ tiếp tục như thế.

Tín hiệu yêu cầu ngắt JEOC được tạo ra(nếu trước đó đã cho phép) khi tất cả các kênh thuộc nhóm injected của ADC1 hoặc ADC2 được chuyển đổi xong. Nếu vẫn tiếp tục nhận được yêu cầu chuyển đổi khi mà tất cả các kênh thuộc nhóm injected được chọn đã chuyển đổi xong thì chế độ này được restart.



Hình 7.12 Chế độ kích hoạt chuyển đổi luân phiên không liên tục

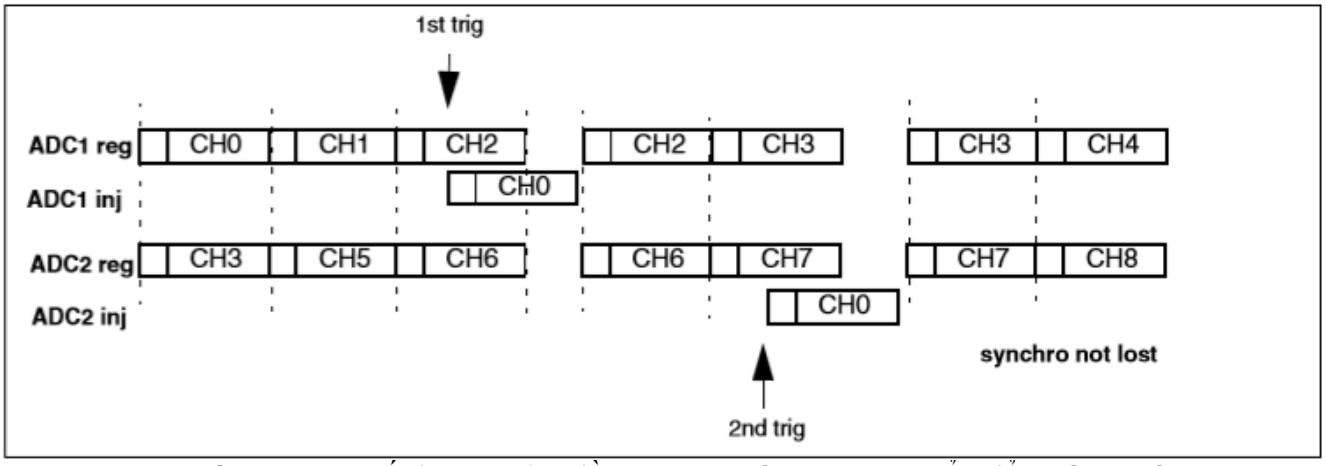
f. Chế độ độc lập

Trong chế độ này mỗi ADC làm việc độc lập với nhau.

g. Regular đồng thời + kích hoạt chuyển đổi luân phiên

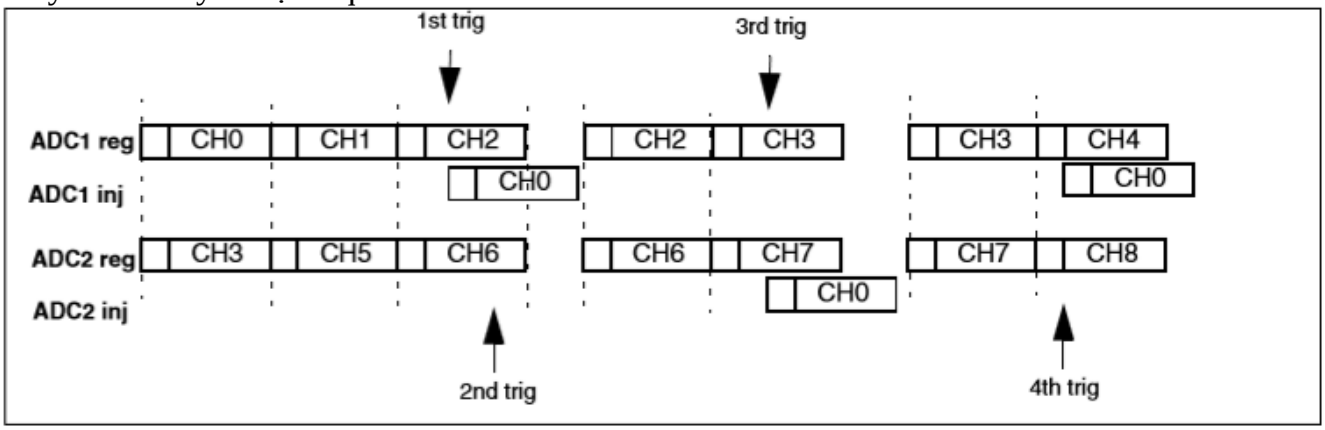
Có thể tạm dừng chuyển đổi đồng thời của nhóm regular để bắt đầu kích hoạt chuyển đổi luân phiên của nhóm injected. Chế độ kích hoạt chuyển đổi luân phiên injected được bắt đầu ngay sau khi nhận được sự kiện injected. Nếu nhóm regular đang hoạt động thì để đảm bảo tính đồng bộ sau khi injected chuyển đổi xong, các chuyển đổi regular của cả hai ADC chủ và tớ đều bị dừng và kích hoạt lại đồng thời khi injected chuyển đổi xong.

Chú ý: Trong chế đọ regular đồng thời + kích hoạt chuyển đổi luân phiên thì người dùng cần cấu hình chính xác thời gian lấy mẫu cho 2 kênh mà sẽ được ADC1 và ADC2 lấy mẫu cùng lúc.



Hình 7.13 Chế độ Regular đồng thời + kích hoạt chuyển đổi luân phiên

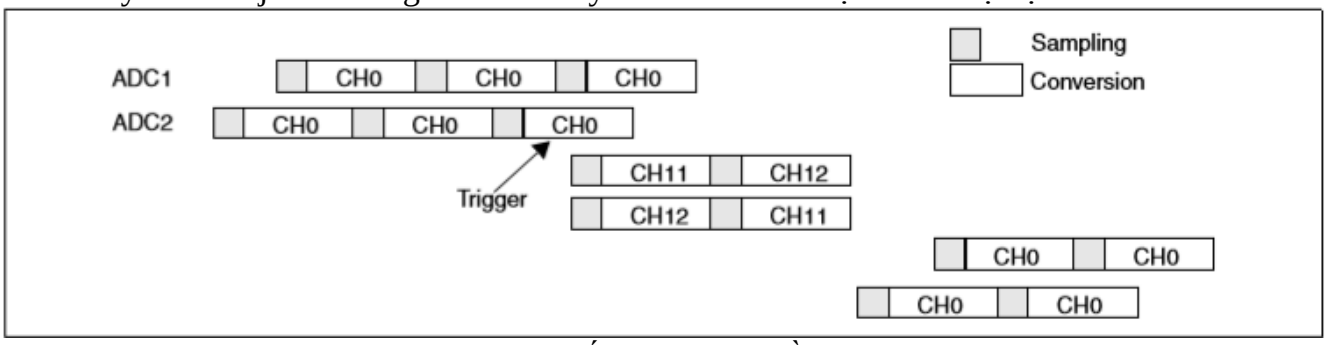
Nếu một tín hiệu yêu cầu chuyển đổi injected phát sinh khi chuyển đổi regular đang bị ngắt thì yêu cầu này sẽ bị bỏ qua.



Hình 7.14 Tín hiệu yêu cầu chuyển đổi bị bỏ qua trong khi injected đang chuyển đổi

h. Injected đồng thời + xen kẽ

Có thể tạm dừng một chuyển đổi xen kẽ bằng mộ sự kiện injected. Khi xảy ra sự kiện injected chuyển đổi xen kẽ được tạm dừng và chuyển đổi injected đồng thời được bắt đầu và đến khi kết thúc chuyển đổi injected đồng thời thì chuyển đổi xen kẽ được kích hoạt lại.



Hình 7.15 Chế độ injected đồng thời + xen kẽ

7.11 CÁC LỆNH THÔNG DỤNG LIÊN QUAN ĐẾN ADC

Bảng 7.3 Các lệnh thông dụng để cấu hình cơ bản cho ADC

SỬ DỤNG THƯ VIỆN "stm32f10x_adc"	
Lệnh	
Thông số hay dùng	**Giải thích**
ADC_InitTypeDef A; (Khai báo biến A thuộc kiểu ADC_InitTypeDef)	
A.ADC_Mode = B; (Lệnh chọn chế độ hoạt động cho ADC)	
B: ADC_Mode_Independent ADC_Mode_RegInjecSimult ADC_Mode_RegSimult_AlterTrig ADC_Mode_InjecSimult_FastInterl ADC_Mode_InjecSimult_SlowInterl ADC_Mode_InjecSimult ADC_Mode_RegSimult ADC_Mode_FastInterl ADC_Mode_SlowInterl ADC_Mode_AlterTrig	**B:** Chế độ hoạt động Các ADC hoạt động độc lập với nhau Regular, Injected đồng thời Regular đồng thời + chuyển đổi luân phiên Injected đồng thời + xen kẽ nhanh Injected đồng thời + xen kẽ chậm Injected đồng thời Regular đồng thời Xen kẽ nhanh Xen kẽ chậm Kích hoạt chuyển đổi luân phiên
A.ADC_ScanConvMode = C; (Lệnh cho phép hoặc cấm chế độ Scan)	
C: ENABLE DISABLE	**C:** Cho phép hoặc cấm Cho phép Cấm
A.ADC_ContinuousConvMode= D; (Lệnh cho phép hoặc cấm chuyển đổi liên tục)	
D: ENABLE DISABLE	**D:** Cho phép hoặc cấm Cho phép Cấm
A.ADC_DataAlign = E; (Lệnh canh lề dữ liệu)	
E: ADC_DataAlign_Right ADC_DataAlign_Left	**E:** Canh lề phải hoặc trái Canh lề phải Canh lề trái
A.ADC_NbrOfChannel = F; (Lệnh khai báo số kênh regular được sử dụng)	
F: 1 … 16	**F:** Số kênh 1 kênh regular … 16 kênh regular
A.ADC_ExternalTrigConv = G; (Lệnh chọn nguồn tín hiệu yêu cầu chuyển đổi từ bên ngoài cho nhóm regular)	

G:	**G:** Nguồn tín hiệu yêu cầu chuyển đổi
ADC_ExternalTrigConv_T1_CC1	Capture Compare 1 của Timer 1
ADC_ExternalTrigConv_T1_CC2	Capture Compare 2 của Timer 1
ADC_ExternalTrigConv_T2_CC2	Capture Compare 2 của Timer 2
ADC_ExternalTrigConv_T3_TRGO	Trigger Out của Timer 3
ADC_ExternalTrigConv_T4_CC4	Capture Compare 4 của Timer 4
ADC_ExternalTrigConv_Ext_IT11_TIM8_TRGO	EXIT line 11 và Trigger Out của Timer 8
ADC_ExternalTrigConv_T1_CC3	Capture Compare 3 của Timer 1
ADC_ExternalTrigConv_None	Không sài yêu cầu chuyển đổi bên ngoài
ADC_ExternalTrigConv_T3_CC1	Capture Compare 1 của Timer 3
ADC_ExternalTrigConv_T2_CC3	Capture Compare 3 của Timer 2
ADC_ExternalTrigConv_T8_CC1	Capture Compare 1 của Timer 8
ADC_ExternalTrigConv_T8_TRGO	Trigger Out của Timer 8
ADC_ExternalTrigConv_T5_CC1	Capture Compare 1 của Timer 5
ADC_ExternalTrigConv_T5_CC3	Capture Compare 3 của Timer 5
ADC_Init(H, &A); (Lệnh cài đặt các thông số đã cấu hình trong biến A cho ADC)	
H:	**H:** ADC cần cấu hình
ADC1	Cấu hình cho ADC1
ADC2	Cấu hình cho ADC2
ADC3	Cấu hình cho ADC3

Bảng 7.4 Các lệnh thông dụng liên quan đến nhóm regular

SỬ DỤNG THƯ VIỆN "stm32f10x_adc"	
Lệnh	
Thông số hay dùng	**Giải thích**
ADC_RegularChannelConfig (A, B, C, D); (Lệnh cấu hình nhóm regular cho ADC)	
A:	**A:** ADC cần cấu hình
ADC1	Cấu hình cho ADC1
ADC2	Cấu hình cho ADC2
ADC3	Cấu hình cho ADC3
B:	**B:** Kênh cần cấu hình
ADC_Channel_0	Kênh tương tự 0
…	…
ADC_Channel_17	Kênh tương tự 17
C:	**C:** Hạng của kênh trong nhóm regular
1 đến 16	Hạng từ 1 đến 16
D:	**D:** Thời gian lấy mẫu
ADC_SampleTime_1Cycles5	1.5 chu kỳ ADCCLK
ADC_SampleTime_7Cycles5	7.5 chu kỳ ADCCLK
ADC_SampleTime_13Cycles5	13.5 chu kỳ ADCCLK
ADC_SampleTime_28Cycles5	28.5 chu kỳ ADCCLK
ADC_SampleTime_41Cycles5	41.5 chu kỳ ADCCLK
ADC_SampleTime_55Cycles5	55.5 chu kỳ ADCCLK
ADC_SampleTime_71Cycles5	71.5 chu kỳ ADCCLK
ADC_SampleTime_239Cycles5	239.5 chu kỳ ADCCLK

ADC_ExternalTrigConvCmd(A, B); (Lệnh cho phép hoặc cấm yêu cầu chuyển đổi từ bên ngoài đối với nhóm regular)	
A: ADC1 ADC2 ADC3 **B:** ENABLE DISABLE	**A:** ADC cần cấu hình Cấu hình cho ADC1 Cấu hình cho ADC2 Cấu hình cho ADC3 **B:** Cho phép hoặc cấm Cho phép Cấm
ADC_DMACmd(A, B); (Lệnh cho phép hoặc cấm ADC sử dụng DMA)	
A: ADC1 ADC2 ADC3 **B:** ENABLE DISABLE	**A:** ADC cần cấu hình Cấu hình cho ADC1 Cấu hình cho ADC2 Cấu hình cho ADC3 **B:** cho phép hoặc cấm Cho phép Cấm ngắt
ADC_Cmd(A, B); (Lệnh cho phép hoặc cấm ADC hoạt động)	
A: ADC1 ADC2 ADC3 **B:** ENABLE DISABLE	**A:** ADC cần cấu hình Cấu hình cho ADC1 Cấu hình cho ADC2 Cấu hình cho ADC3 **B:** cho phép hoặc cấm Cho phép Cấm ngắt
ADC_ResetCalibration(A); **while(ADC_GetResetCalibrationStatus(A));** **ADC_StartCalibration(A);** **while(ADC_GetCalibrationStatus(A));** (Các lệnh reset và hiệu chỉnh các thông số của ADC)	
A: ADC1 ADC2 ADC3	**A:** ADC cần cấu hình Cấu hình cho ADC1 Cấu hình cho ADC2 Cấu hình cho ADC3
ADC_SoftwareStartConvCmd(A, B); (Lệnh cho phép hoặc cấm bắt đầu chuyển đổi bằng phần mềm - sử dụng khi không chọn yêu cầu chuyển đổi từ bên ngoài)	
A: ADC1 ADC2 ADC3 **B:** ENABLE DISABLE	**A:** ADC cần cấu hình Cấu hình cho ADC1 Cấu hình cho ADC2 Cấu hình cho ADC3 **B:** cho phép hoặc cấm Cho phép Cấm ngắt
ADC_GetFlagStatus(A, B) ; (Lệnh đọc cờ trạng thái của ADC)	

A:	**A:** ADC cần đọc
ADC1	Đọc ADC1
ADC2	Đọc ADC2
ADC3	Đọc ADC3
B:	**B:** Cờ cần đọc
ADC_FLAG_AWD	Cờ vượt ngưỡng của bộ giám sát tương tự
ADC_FLAG_EOC	Cờ báo chuyển đổi xong của nhóm regular
ADC_FLAG_JEOC	Cờ báo chuyển đổi xong của nhóm injected
ADC_FLAG_JSTRT	Cờ báo bắt đầu chuyển đổi nhóm injected
ADC_FLAG_STRT	Cờ báo bắt đầu chuyển đổi nhóm regular
ADC_ClearFlag(A, B) ; (Lệnh xóa cờ trạng thái của ADC)	
A:	**A:** ADC cần xóa
ADC1	ADC1
ADC2	ADC2
ADC3	ADC3
B:	**B:** Cờ cần xóa
ADC_FLAG_AWD	Cờ vượt ngưỡng của bộ giám sát tương tự
ADC_FLAG_EOC	Cờ báo chuyển đổi xong của nhóm regular
ADC_FLAG_JEOC	Cờ báo chuyển đổi xong của nhóm injected
ADC_FLAG_JSTRT	Cờ báo bắt đầu chuyển đổi nhóm injected
ADC_FLAG_STRT	Cờ báo bắt đầu chuyển đổi nhóm regular
unsinged short A; **A = ADC_GetConversionValue(B);** (Lệnh đọc giá trị chuyển đổi được của kênh regular)	
B:	**B:** ADC cần đọc
ADC1	Đọc ADC1
ADC2	Đọc ADC2
ADC3	Đọc ADC3
unsinged long A; **A = ADC_GetDualModeConversionValue();** (Lệnh đọc giá trị chuyển đổi được ADC1 và ADC2 ở chế độ song song chủ tớ)	

Bảng 7.5 Các lệnh thông dụng liên quan đến nhóm injected

SỬ DỤNG THƯ VIỆN "stm32f10x_adc"	
Lệnh	
Thông số hay dùng	**Giải thích**
ADC_InjectedChannelConfig(A, B, C, D); (Lệnh cấu hình cho nhóm Injected)	
A:	**A:** ADC cần cấu hình
ADC1	Cấu hình cho ADC1
ADC2	Cấu hình cho ADC2
ADC3	Cấu hình cho ADC3
B:	**B:** Kênh cần cấu hình
ADC_Channel_0	Kênh tương tự 0
…	…
ADC_Channel_17	Kênh tương tự 17
C:	**C:** Hạng của kênh trong nhóm injected

1	Hạng 1
…	...
4	Hạng 4
D:	**D:** Thời gian lấy mẫu
ADC_SampleTime_1Cycles5	1.5 chu kỳ ADCCLK
ADC_SampleTime_7Cycles5	7.5 chu kỳ ADCCLK
ADC_SampleTime_13Cycles5	13.5 chu kỳ ADCCLK
ADC_SampleTime_28Cycles5	28.5 chu kỳ ADCCLK
ADC_SampleTime_41Cycles5	41.5 chu kỳ ADCCLK
ADC_SampleTime_55Cycles5	55.5 chu kỳ ADCCLK
ADC_SampleTime_71Cycles5	71.5 chu kỳ ADCCLK
ADC_SampleTime_239Cycles5	239.5 chu kỳ ADCCLK
ADC_InjectedSequencerLengthConfig(A, B); (Lệnh khai báo số kênh thuộc nhóm Injected được sử dụng)	
A:	**A:** ADC cần cấu hình
ADC1	Cấu hình cho ADC1
ADC2	Cấu hình cho ADC2
ADC3	Cấu hình cho ADC3
B:	**B:** Số lượng kênh Injected được sử dụng
1	1 kênh
…	…
4	4 kênh
ADC_ExternalTrigInjectedConvConfig(A, B); (Lệnh chọn tín hiệu yêu cầu chuyển đổi từ bên ngoài cho nhóm Injected)	
A:	**A:** ADC được cấu hình
ADC1	Cấu hình cho ADC1
ADC2	Cấu hình cho ADC2
ADC3	Cấu hình cho ADC3
B:	**B:** Nguồn tín hiệu yêu cầu chuyển đổi
ADC_ExternalTrigInjecConv_T1_TRGO	Trigger Out của Timer 1
ADC_ExternalTrigInjecConv_T1_CC4	Capture Compare 2 của Timer 1
ADC_ExternalTrigInjecConv_T2_TRGO	Trigger Out của Timer 2
ADC_ExternalTrigInjecConv_T2_CC1	Capture Compare 1 của Timer 2
ADC_ExternalTrigInjecConv_T3_CC4	Capture Compare 4 của Timer 3
ADC_ExternalTrigInjecConv_T4_TRGO	Trigger Out của Timer 4
ADC_ExternalTrigInjecConv_Ext_IT15_TIM8_CC4	EXIT line 15 và Capture Compare 4 của Timer 8
ADC_ExternalTrigInjecConv_T4_CC3	Capture Compare 3 của Timer 4
ADC_ExternalTrigInjecConv_T8_CC2	Capture Compare 2 của Timer 8
ADC_ExternalTrigInjecConv_T8_CC4	Capture Compare 4 của Timer 8
ADC_ExternalTrigInjecConv_T5_TRGO	Trigger Out của Timer 5
ADC_ExternalTrigInjecConv_T5_CC4	Capture Compare 4 của Timer 5
ADC_ExternalTrigInjecConv_None	Không dùng yêu cầu chuyển đổi ngoại
ADC_ExternalTrigInjectedConvCmd(A, B); (Lệnh cho phép hoặc cấm yêu cầu chuyên đổi từ bên ngoài cho nhóm injected)	
A:	**A:** ADC cần cấu hình
ADC1	Cấu hình cho ADC1
ADC2	Cấu hình cho ADC2
ADC3	Cấu hình cho ADC3
B:	**B:** Cho phép hoặc cấm

ENABLE DISABLE	Cho phép Cấm
ADC_AutoInjectedConvCmd(A, B); (Lệnh cho phép hoặc cấm chế độ AutoInjected)	
A: ADC1 ADC2 ADC3 **B:** ENABLE DISABLE	**A:** ADC cần cấu hình Cấu hình cho ADC1 Cấu hình cho ADC2 Cấu hình cho ADC3 **B:** Cho phép hoặc cấm Cho phép Cấm
ADC_DMACmd(A, B); (Lệnh cho phép hoặc cấm ADC sử dụng DMA)	
A: ADC1 ADC2 ADC3 **B:** ENABLE DISABLE	**A:** ADC cần cấu hình Cấu hình cho ADC1 Cấu hình cho ADC2 Cấu hình cho ADC3 **B:** cho phép hoặc cấm Cho phép Cấm ngắt
ADC_Cmd(A, B); (Lệnh cho phép hoặc cấm ADC hoạt động)	
A: ADC1 ADC2 ADC3 **B:** ENABLE DISABLE	**A:** ADC cần cấu hình Cấu hình cho ADC1 Cấu hình cho ADC2 Cấu hình cho ADC3 **B:** cho phép hoặc cấm Cho phép Cấm ngắt
ADC_ResetCalibration(A); **while(ADC_GetResetCalibrationStatus(A));** **ADC_StartCalibration(A);** **while(ADC_GetCalibrationStatus(A));** (Các lệnh reset và hiệu chỉnh các thông số của ADC)	
A: ADC1 ADC2 ADC3	**A:** ADC cần cấu hình Cấu hình cho ADC1 Cấu hình cho ADC2 Cấu hình cho ADC3
unsinged short A; **A= ADC_GetInjectedConversionValue(B, C);** (Lệnh đọc giá trị chuyển đổi được của kênh injected)	
B: ADC1 ADC2 ADC3 **C:** ADC_InjectedChannel_1 … ADC_InjectedChannel_4	**B:** ADC cần đọc Đọc ADC1 Đọc ADC2 Đọc ADC3 **C:** Kênh injected cần đọc Kênh injected 1 … Kênh injected 4

Bảng 7.6 Các lệnh thông dụng liên quan đến bộ giám sát điện áp tương tự AWD

SỬ DỤNG THƯ VIỆN "stm32f10x_adc"	
Lệnh	
Thông số hay dùng	**Giải thích**
ADC_AnalogWatchdogThresholdsConfig(A, B, C); (Lệnh cài đặt ngưỡng trên và dưới cho analog watchdog)	
A: ADC1 ADC2 ADC3 **B:** [0 - 4095] **C:** [0 - 4095]	**A:** ADC cần cấu hình Cấu hình cho ADC1 Cấu hình cho ADC2 Cấu hình cho ADC3 **B:** Ngưỡng cao [0 - 4095] **C**: Ngưỡng thấp [0 - 4095]
ADC_AnalogWatchdogCmd(A, B); (Lệnh cấu hình chế độ giám sát cho analog watchdog)	
A: ADC1 ADC2 ADC3 **B:** ADC_AnalogWatchdog_SingleRegEnable ADC_AnalogWatchdog_SingleInjecEnable ADC_AnalogWatchdog_SingleRegOrInjecEnable ADC_AnalogWatchdog_AllRegEnable ADC_AnalogWatchdog_AllInjecEnable ADC_AnalogWatchdog_AllRegAllInjecEnable ADC_AnalogWatchdog_None	**A:** Chọn ADC Chọn ADC1 Chọn ADC2 Chọn ADC3 **B:** Chế độ hoạt giám sát Giám sát 1 kênh regular Giám sát 1 kênh injected Giám sát 1 kênh regular hoặc injected Giám sát tất cả các kênh regular Giám sát tất cả các kênh injected Giám sát tất cả các kênh regular, injected Không giám sát kênh nào
ADC_AnalogWatchdogSingleChannelConfig(A, B); (Lệnh chọn kênh cần được giám sát bởi analog watchdog ở chế độ giám sát 1 kênh)	
A: ADC1 ADC2 ADC3 **B:** ADC_Channel_0 … ADC_Channel_17	**A:** Chọn ADC Chọn ADC1 Chọn ADC2 Chọn ADC3 **B:** Kênh được theo dõi Kênh 0 … Kênh 17

Bảng 7.7 Các lệnh thông dụng liên quan đến chế độ không liên tục(Discontinuous mode)

SỬ DỤNG THƯ VIỆN "stm32f10x_adc"	
Lệnh	
Thông số hay dùng	**Giải thích**
ADC_DiscModeChannelCountConfig(A, B); (Lệnh chọn số lượng kênh được chuyển đổi bởi 1 lần yêu cầu ở chế độ không liên tục)	

A: ADC1 ADC2 ADC3 **B:** 1 … 8	**A:** ADC cần cấu hình Cấu hình cho ADC1 Cấu hình cho ADC2 Cấu hình cho ADC3 B: Số lượng kênh 1 kênh … 8 kênh
ADC_DiscModeCmd(A, B); (Lệnh cho phép hoặc cấm chế độ không liên tục)	
A: ADC1 ADC2 ADC3 **B:** ENABLE DISABLE	**A:** ADC cần cấu hình Cấu hình cho ADC1 Cấu hình cho ADC2 Cấu hình cho ADC3 **B:** Cho phép hoặc cấm Cho phép Cấm

Bảng 7.8 Các lệnh thông dụng liên quan đến ngắt ADC

SỬ DỤNG THƯ VIỆN "stm32f10x_adc"	
Lệnh	
Thông số hay dùng	**Giải thích**
ADC_ITConfig(A, B, C); (Lệnh cấu hình cho phép ngắt ADC)	
A: ADC1 ADC2 ADC3 **B:** ADC_IT_EOC ADC_IT_AWD ADC_IT_JEOC **C:** ENABLE DISABLE	**A:** ADC cần cấu hình Cấu hình cho ADC1 Cấu hình cho ADC2 Cấu hình cho ADC3 **B:** Loại ngắt Ngắt khi regular chuyển đổi hoàn tất Ngắt Analog Watchdog Ngắt khi injected chuyển đổi hoàn tất **C:** Cho phép hoặc cấm ngắt Cho phép Cấm ngắt
ADC_GetITStatus (A, B); (Lệnh đọc trạng thái ngắt của ADC)	
A: ADC1 ADC2 ADC3 **B:** ADC_IT_EOC ADC_IT_AWD ADC_IT_JEOC	**A:** ADC cần đọc Đọc ADC1 Đọc ADC2 Đọc ADC3 **B:** Trạng thái ngắt cần đọc Ngắt khi regular chuyển đổi xong Ngắt analog watchdog Ngắt kh injected chuyển đổi hoàn tất
ADC_ClearITPendingBit(A, B); (Lệnh xóa cờ ngắt của ADC)	

A:	**A:** ADC cần cấu hình
ADC1	Cấu hình cho ADC1
ADC2	Cấu hình cho ADC2
ADC3	Cấu hình cho ADC3
B:	**B:** Loại cờ ngắt
ADC_IT_EOC	Ngắt khi regular chuyển đổi hoàn tất
ADC_IT_AWD	Ngắt Analog Watchdog
ADC_IT_JEOC	Ngắt khi injected chuyển đổi hoàn tất

7.12 CÁC VÍ DỤ LIÊN QUAN ĐẾN ADC

7.12.1 Ví dụ về chế độ chuyển đổi đơn(Single Conversion Mode)

Ví dụ 7.1 Viết chương trình điều khiển ADC1 tự động chuyển đổi bằng phần mềm 1 lần duy nhất giá trị điện áp đưa vào kênh số 15 thành dữ liệu số.

Chương trình

```
#include<stm32f10x.h>
void cauhinhGPIO()
{
   GPIO_InitTypeDef    GPIO;
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC|
                          RCC_APB2Periph_ADC1|
                          RCC_APB2Periph_AFIO,ENABLE);
   GPIO.GPIO_Pin = GPIO_Pin_5;
   GPIO.GPIO_Mode = GPIO_Mode_AIN ;
   GPIO_Init(GPIOC,&GPIO);
}
void cauhinhADC()
{
    ADC_InitTypeDef ADC_InitStructure;
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;
    ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
    // Chọn chế độ Single Conversion
    ADC_InitStructure.ADC_ExternalTrigConv = \
                                ADC_ExternalTrigConv_None;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_NbrOfChannel = 1;
    ADC_Init(ADC1, &ADC_InitStructure);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_15, 1,\
                                ADC_SampleTime_239Cycles5);
    ADC_Cmd(ADC1, ENABLE);
    ADC_ResetCalibration(ADC1);
    while(ADC_GetResetCalibrationStatus(ADC1));
    ADC_StartCalibration(ADC1);
    while(ADC_GetCalibrationStatus(ADC1));
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);
}
int main()
```

```
{    unsigned short kqadc;
     SystemInit();
     cauhinhGPIO();
     cauhinhADC();
    while(1)
      {
        if(ADC_GetFlagStatus(ADC1,ADC_FLAG_EOC)==1)
         {
          ADC_ClearFlag(ADC1,ADC_FLAG_EOC);
          kqadc = ADC_GetConversionValue(ADC1);
          // Hiển thị biến kqadc để quan sát
         }
      }
 }
```

7.12.2 Ví dụ về chế độ chuyển đổi liên tục(Continuous Mode)

Ví dụ 7.2 Viết chương trình điều khiển ADC1 tự động chuyển đổi liên tục bằng phần mềm giá trị điện áp đưa vào kênh số 15 thành dữ liệu số.

Chương trình

```
...
   ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
   // Chương trình giống với chế độ chuyển đổi đơn chỉ khác
   // dòng này ta đổi "DISABLE" thành "ENABLE"
...
```

7.12.3 Ví dụ về chế độ quét(Scan Mode)

Ví dụ 7.3 Viết chương trình điều khiển ADC1 tự động chuyển đổi liên tục bằng phần mềm giá trị điện áp đưa vào kênh số 15 và kênh số 16(cảm biến nhiệt độ) thành dữ liệu số.

Chương trình

```
#include<stm32f10x.h>
unsigned short DLCD[2]; // Mảng lưu giá trị chuyển đổi
void cauhinhGPIO()
{
   GPIO_InitTypeDef     GPIO;
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC|
                          RCC_APB2Periph_ADC1
                          RCC_APB2Periph_AFIO,ENABLE);
   GPIO.GPIO_Pin = GPIO_Pin_5;
   GPIO.GPIO_Mode = GPIO_Mode_AIN ;
   GPIO_Init(GPIOC,&GPIO);
}
void cauhinhDMA()
{
     DMA_InitTypeDef DMA_InitStructure;
     RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
     DMA_DeInit(DMA1_Channel1);
     DMA_InitStructure.DMA_PeripheralBaseAddr = (u32)&ADC1->DR;
```

```
    DMA_InitStructure.DMA_MemoryBaseAddr = (u32)&DLCD;
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
    DMA_InitStructure.DMA_BufferSize = 2;
    DMA_InitStructure.DMA_PeripheralInc =
                                        DMA_PeripheralInc_Disable;
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
    DMA_InitStructure.DMA_PeripheralDataSize =
                                    DMA_PeripheralDataSize_HalfWord;
    DMA_InitStructure.DMA_MemoryDataSize =
                                      DMA_MemoryDataSize_HalfWord;
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
    DMA_InitStructure.DMA_Priority = DMA_Priority_High;
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
    DMA_Init(DMA1_Channel1, &DMA_InitStructure);
}
void cauhinhADC()
{
    ADC_InitTypeDef ADC_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
    ADC_InitStructure.ADC_ScanConvMode = ENABLE;
    // Bật chế độ scan
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
    ADC_InitStructure.ADC_ExternalTrigConv =
                                        ADC_ExternalTrigConv_None;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_NbrOfChannel = 2;
    ADC_Init(ADC1, &ADC_InitStructure);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_15, 1,
                                        ADC_SampleTime_239Cycles5);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_16, 2,
                                        ADC_SampleTime_239Cycles5);
    ADC_Cmd(ADC1, ENABLE);
    ADC_DMACmd(ADC1, ENABLE);
    ADC_TempSensorVrefintCmd(ENABLE);// Bật cảm biến nhiệt độ
    ADC_ResetCalibration(ADC1);
    while(ADC_GetResetCalibrationStatus(ADC1));
    ADC_StartCalibration(ADC1);
    while(ADC_GetCalibrationStatus(ADC1));
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);
    DMA_Cmd(DMA1_Channel1, ENABLE);
}
int main()
{
    float nhietdo;
    SystemInit();
    cauhinhGPIO();
    cauhinhDMA();
    cauhinhADC();
    while(1)
```

```
    {
            if(DMA_GetFlagStatus(DMA1_FLAG_TC1)==1)
            {
              DMA_ClearFlag (DMA1_FLAG_TC1);
              nhietdo =( DLCD[1]*3.3)/4096;
              nhietdo=(1.43-nhietdo)/0.0043+25;
              // Hiên thị DLCD[0] để xem kết quả kênh 15
              // Hiển thị biến nhietdo để xem nhiệt độ
            }
    }
 }
```

7.12.4 Ví dụ về bộ giám sát điện áp tương tự(Analog watchdog)

Ví dụ 7.4 Viết chương trình điều khiển ADC1 tự động chuyển đổi liên tục bằng phần mềm giá trị điện áp đưa vào kênh số 15 đồng thời sử dụng analog watchdog để giám sát giá trị chuyển đổi được nếu bé hơn 1500 hoặc lớn hơn 3500 thì mở LED nối với chân D8 và ngược lại tắt LED.

Chương trình

```
#include<stm32f10x.h>
void cauhinhGPIO()
{
   GPIO_InitTypeDef     GPIO;
   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC|
                     RCC_APB2Periph_GPIOD|RCC_APB2Periph_ADC1
                     | RCC_APB2Periph_AFIO,ENABLE);
   GPIO.GPIO_Pin = GPIO_Pin_5;
   GPIO.GPIO_Mode = GPIO_Mode_AIN ;
   GPIO_Init(GPIOC,&GPIO);
   GPIO.GPIO_Pin = GPIO_Pin_8;
   GPIO.GPIO_Mode = GPIO_Mode_Out_PP;
   GPIO.GPIO_Speed= GPIO_Speed_50MHz;
   GPIO_Init(GPIOD,&GPIO);
}
void cauhinhADC()
{
    ADC_InitTypeDef ADC_InitStructure;
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
    ADC_InitStructure.ADC_ScanConvMode = DISABLE;
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE ;
    ADC_InitStructure.ADC_ExternalTrigConv = \
                                  ADC_ExternalTrigConv_None;
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    ADC_InitStructure.ADC_NbrOfChannel = 1;
   ADC_Init(ADC1, &ADC_InitStructure);
   ADC_RegularChannelConfig(ADC1, ADC_Channel_15, 1,\
                                ADC_SampleTime_239Cycles5);
   ADC_AnalogWatchdogThresholdsConfig (ADC1,3500,1500);
    // Cài đặt ngưỡng cao là 3500 và thấp là 1500
   ADC_AnalogWatchdogSingleChannelConfig(ADC1,ADC_Channel_15 );
```

```
    // Chọn giám sát kênh số 15 của ADC1
   ADC_AnalogWatchdogCmd
                      (ADC1,ADC_AnalogWatchdog_SingleRegEnable);
    // Cho phép chế độ giám sát 1 kênh regular
   ADC_ITConfig (ADC1,ADC_IT_AWD ,ENABLE );
    // Cho phép ngắt analog watchdog
   ADC_Cmd(ADC1, ENABLE);
   ADC_ResetCalibration(ADC1);
   while(ADC_GetResetCalibrationStatus(ADC1));
   ADC_StartCalibration(ADC1);
   while(ADC_GetCalibrationStatus(ADC1));
   ADC_SoftwareStartConvCmd(ADC1, ENABLE);
}
void cauhinhNVIC()
{
   NVIC_InitTypeDef       NV;
   #ifdef   VECT_TAB_RAM
   NVIC_SetVectorTable(NVIC_VectTab_RAM,0);
   #else
   NVIC_SetVectorTable(NVIC_VectTab_FLASH,0);
   #endif
   NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0 );
   NV.NVIC_IRQChannel = ADC1_2_IRQn;
   NV.NVIC_IRQChannelSubPriority = 1;
   NV.NVIC_IRQChannelCmd = ENABLE ;
   NVIC_Init(&NV);
}
void ADC1_2_IRQHandler()
{
   if( ADC_GetITStatus ( ADC1,ADC_IT_AWD )==1)
     {
        ADC_ClearITPendingBit( ADC1,ADC_IT_AWD );
        GPIOD->BSRR = GPIO_Pin_8;
     }
}
int main()
{   unsigned short kqadc;
    SystemInit();
    cauhinhGPIO();
    cauhinhADC();
    cauhinhNVIC();
    while(1)
     {
       if(ADC_GetFlagStatus(ADC1,ADC_FLAG_EOC)==1)
        {
         ADC_ClearFlag(ADC1,ADC_FLAG_EOC);
         kqadc = ADC_GetConversionValue(ADC1);
         if((kqadc >1500)&&(kqadc<3500))
         GPIOD->BRR = GPIO_Pin_8;
        }
```

```
    }
}
```

7.12.5 Ví dụ về chế độ song song 2 ADC- injected đồng thời(Injected simultaneous mode)

Ví dụ 7.5 Viết chương trình điều khiển ADC1 và ADC2 chuyển đổi liên tục 2 kênh 16 và 15 theo chế độ injected đồng thời.

Chương trình

```
#include<stm32f10x.h>
unsigend short kq1, kq2;
void cauhinhGPIO()
{
    GPIO_InitTypeDef    GPIO;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC
                    |RCC_APB2Periph_ADC1|RCC_APB2Periph_ADC2
                    |RCC_APB2Periph_AFIO,ENABLE);
    GPIO.GPIO_Pin = GPIO_Pin_5;
    GPIO.GPIO_Mode = GPIO_Mode_AIN ;
    GPIO_Init(GPIOC,&GPIO);
}
void cauhinhADC()
{
    ADC_InitTypeDef          CHADC;
    CHADC.ADC_Mode = ADC_Mode_InjecSimult ;
    CHADC.ADC_ScanConvMode = DISABLE;
    CHADC.ADC_ContinuousConvMode = ENABLE ;
    CHADC.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None ;
    CHADC.ADC_DataAlign = ADC_DataAlign_Right ;
    CHADC.ADC_NbrOfChannel = 1;
    ADC_Init(ADC1,&CHADC);
    ADC_Init(ADC2,&CHADC);
    ADC_InjectedChannelConfig(ADC1,ADC_Channel_16,
                              1,ADC_SampleTime_239Cycles5 );
    ADC_InjectedChannelConfig(ADC2,ADC_Channel_15
                              ,1,ADC_SampleTime_239Cycles5);
    ADC_InjectedSequencerLengthConfig(ADC1, 1);
    ADC_InjectedSequencerLengthConfig(ADC2, 1);
    ADC_ExternalTrigInjectedConvConfig(ADC1,
                              ADC_ExternalTrigInjecConv_None);
    ADC_ExternalTrigInjectedConvConfig(ADC2,
                              ADC_ExternalTrigInjecConv_None);
    ADC_AutoInjectedConvCmd(ADC1, ENABLE);
    ADC_AutoInjectedConvCmd(ADC2, ENABLE);
        ADC_ExternalTrigInjectedConvCmd(ADC2, ENABLE );
        ADC_ITConfig(ADC1,ADC_IT_JEOC,ENABLE );
        ADC_Cmd(ADC1,ENABLE);
        ADC_Cmd(ADC2,ENABLE);
        ADC_TempSensorVrefintCmd(ENABLE );
        ADC_ResetCalibration(ADC1);
```

```
        while(ADC_GetResetCalibrationStatus(ADC1));
        ADC_StartCalibration(ADC1);
        while(ADC_GetCalibrationStatus (ADC1));
        ADC_ResetCalibration(ADC2);
        while(ADC_GetResetCalibrationStatus(ADC2));
        ADC_StartCalibration(ADC2);
        while(ADC_GetCalibrationStatus (ADC2));
        ADC_SoftwareStartInjectedConvCmd (ADC1,ENABLE );
}
void cauhinhNVIC()
{
        NVIC_InitTypeDef      NV;
        #ifdef   VECT_TAB_RAM
        NVIC_SetVectorTable(NVIC_VectTab_RAM,0);
        #else
         NVIC_SetVectorTable(NVIC_VectTab_FLASH,0);
        #endif
        NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0 );
        NV.NVIC_IRQChannel = ADC1_2_IRQn;
        NV.NVIC_IRQChannelSubPriority = 1;
        NV.NVIC_IRQChannelCmd = ENABLE ;
        NVIC_Init(&NV);
}
void ADC1_2_IRQHandler()
{
        if( ADC_GetITStatus ( ADC1,ADC_IT_JEOC )==1)
         {
          ADC_ClearITPendingBit( ADC1,ADC_IT_JEOC );
          kq1 = ADC_GetInjectedConversionValue
                                  (ADC1,ADC_InjectedChannel_1);
          kq2 = ADC_GetInjectedConversionValue
                                  (ADC2,ADC_InjectedChannel_1);
         }
}
int main()
{
       char ht[10];
       SystemInit();
       cauhinhGPIO();
       cauhinhADC();
       cauhinhNVIC();
       while(1){ // hiển thị kq1 và kq2 để quan sát}

}
```