Chương 8

BỘ ĐỊNH THỜI

TIMER

8.1 GIỚI THIỆU TIMER

Họ STM32F1x có 3 loại timer là:

- Các timer điều khiển nâng cao (Advanced control timers – TIM1&TIM8)
- Các timer cơ bản (Basic timers – TIM6&TIM7)
- Các timer sử dụng cho mục đích thông thường(General purpose timers –timer còn lại)

Nội dung chương 8 chỉ tập trung trình bày về các timer điều khiển nâng cao vì đây là 2 timer có nhiều chức năng nhất. Điều khiển được 2 timer này ta cũng có thể điều khiển ở mức độ cơ bản các timer còn lại.

Timer điều khiển nâng cao (TIM1&TIM8) là các timer 16 bit, có khả năng tự động nạp lại và có bộ chia trước. Các timer này có thể được sử dụng cho nhiều mục đích khác nhau như là đo độ rộng xung hay tạo ra các dạng sóng điện áp ngõ ra với độ rộng xung có thể được điều biến từ vài micro giây đến vài mili giây bằng cách sử dụng các bộ chia trước. Timer điều khiển nâng cao và các timer sử dụng cho mục đích thông thường hoạt động hoàn toàn độc lập với nhau và không cùng chia sẻ bất cứ tài nguyên nào của chip.

8.2 NHỮNG ĐẶC ĐIỂM CHÍNH CỦA TIM1&TIM8

Các timer điều khiển nâng cao có khả năng tự động nạp lại và hỗ trợ nhiều kiểu đếm:

- Đếm lên
- Đếm xuống
- Đếm lên xuống

Bộ chia trước 16 bit cho phép chia tần số nguồn xung clock ngõ vào từ 1 đến 65535 lần.

Có tới 4 kênh độc lập

- Ngõ vào Capture(Input Capture)
- Ngõ ra Compare(Output Compare)
- Ngõ ra PWM(Output PWM)
- Ngõ ra của chế độ 1 xung(One-pulse mode output)

Mạch đồng bộ giúp điều khiển timer bằng tín hiệu bên ngoài và tạo kết nối bên trong giữa một vài timer với nhau.

Bộ đếm lặp cho phép tạo sự kiện cập nhật cập chỉ khi counter đếm đủ số chu kỳ đã được cài đặt trong bộ đếm lặp này.

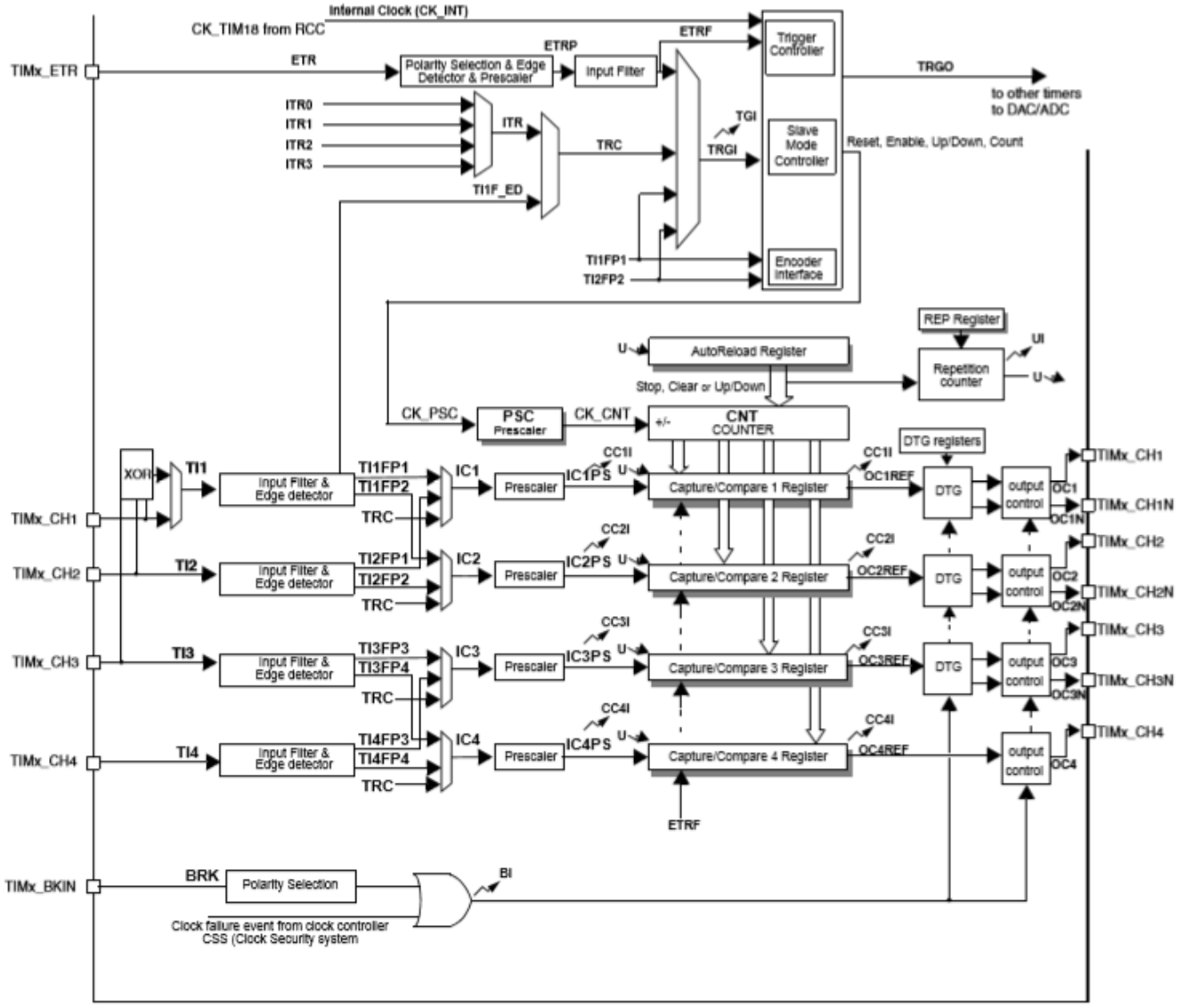
Ngõ vào Break cho phép điều khiển tín hiệu ngõ ra của timer về trạng thái mặc định.

Có thể tạo ra ngắt hoặc yêu cầu DMA khi một trong các sự kiện sau xảy ra:

- Cập nhật: Bộ đếm tràn, cài đặt bộ đếm.
- Sự kiện kích hoạt: bắt đầu đếm, dừng đếm, cài đặt hoặc đếm bằng nguồn kích hoạt bên trong, bên ngoài.
- Input Capture
- Output Compare
- Break Input

Hỗ trợ mạch giao tiếp với cảm biến hall và encoder

Có các ngõ vào nhận xung kích hoạt từ bên ngoài.



Hình 8.1 Sơ đồ khối của timer1 và timer8

8.3 CÁC CHẾ ĐỘ HOẠT ĐỘNG CHÍNH CỦA TIMER1&TIMER8

8.3.1 Các chế độ đếm

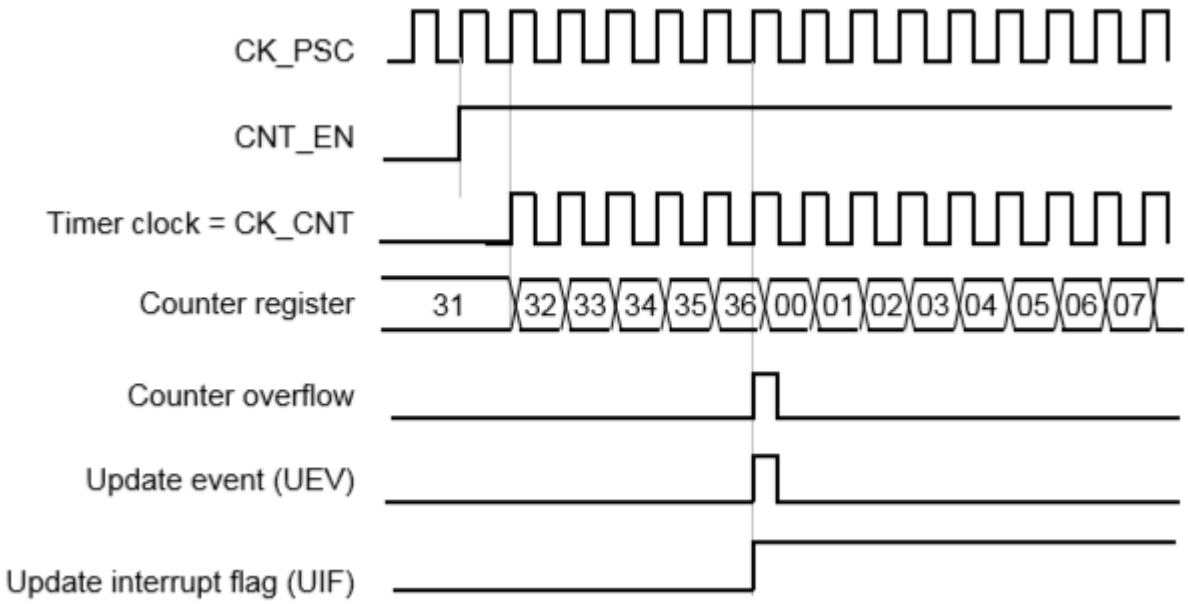
a. Chế độ đếm lên- upcounting mode

Bộ đếm sẽ đếm từ 0 đến giá trị được đăt trong thanh ghi TIMx_ARR, sau đó sẽ bắt đầu đếm lại từ 0 và tạo ra một sự kiện tràn timer. Nếu hoạt động ở chế độ đếm lặp(repetition counter) thì sự kiện cập nhật(UEV-update event) chỉ được tạo khi số lần tràn của bộ đếm lớn hơn giá trị đã được lập trình trong thanh ghi đếm lặp 1 đơn vị(TIMx_RCR+1). Tuy nhiên, nếu bộ đếm không hoạt động ở chế độ đếm lặp thì sự kiện cập nhật sẽ được tạo ra ngay khi bộ đếm tràn. Việc set bit UG trong thanh ghi TIMx_EGR(bằng phần mềm hoặc sử dụng chế độ tớ) cũng có thể tạo ra một sự kiện cập nhật. Sự kiện cập nhật UEV có thể được tắt bằng phần mềm bằng cách set bit UDIS trong thanh ghi TIMx_CR1. Việc tắt sự kiện cập nhật giúp bộ đếm không bị cập nhật giá trị cũ trong khi đang cài đặt giá trị mới.

Khi sự kiện cập nhật xảy ra thì:

- Cờ cập nhật được set
- Giá trị đếm lặp được nạp lại từ thanh ghi TIMx_RCR
- Giới hạn đếm được nạp lại từ thanh ghi TIMx_ARR.
- Giá trị của bộ chia được nạp lại từ thanh ghi TIMx_PSC

Sự kiện cập nhật có thể tạo ra yêu cầu ngắt hoặc DMA



Hình 8.2 Chế độ đếm lên với giá trị nạp lại là 0x36

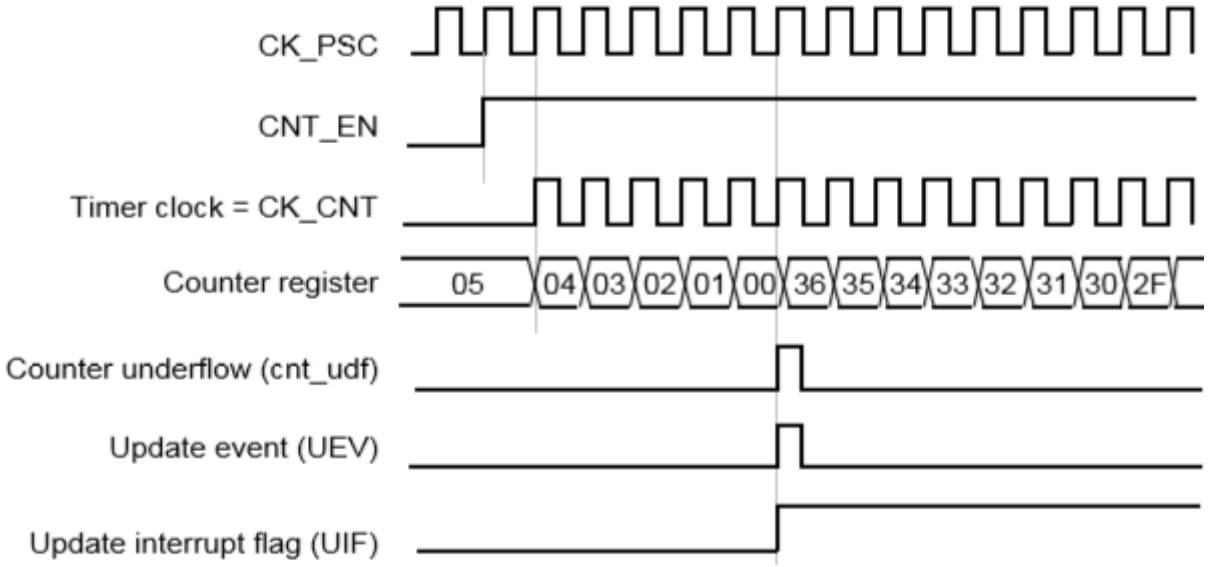
b. Chế độ đếm xuống – downcounting mode

Trong chế độ đếm xuống, bộ đếm sẽ đếm từ giá trị giá trị tự động nạp lại(chứa trong thanh ghi TIMx_ARR) xuống 0, sau đó tiếp tục đếm xuông từ giá trị tự động nạp lại và đồng thời tạo ra sự kiện tràn. Nếu hoạt động ở chế độ đếm lặp(repetition counter) thì sự kiện cập nhật(UEV) chỉ được tạo khi số lần tràn của bộ đếm lớn hơn giá trị đã được lập trình trong thanh ghi đếm lặp 1 đơn vị(TIMx_RCR+1). Tuy nhiên, nếu bộ đếm không hoạt động ở chế độ đếm lặp thì sự kiện cập nhật sẽ được tạo ra ngay khi bộ đếm tràn. Việc set bit UG trong thanh ghi TIMx_EGR(bằng phần mềm hoặc sử dụng chế độ tớ) cũng có thể tạo ra một sự kiện cập nhật. Sự kiện cập nhật UEV có thể được tắt bằng phần mềm bằng cách set bit UDIS trong thanh ghi TIMx_CR1. Việc tắt sự kiện cập nhật giúp bộ đếm không bị cập nhật giá trị cũ trong khi đang cài đặt giá trị mới.

Khi sự kiện cập nhật xảy ra thì:

- Cờ cập nhật được set
- Giá trị đếm lặp được nạp lại từ thanh ghi TIMx_RCR
- Giới hạn đếm được nạp lại từ thanh ghi TIMx_ARR

Sự kiện cập nhật có thể tạo ra yêu cầu ngắt hoặc DMA



Hình 8.3 Chế độ đếm xuống của với giá trị tự động nạp lại là 0x36

c. Chế độ canh giữa(đếm lên/xuống)-Center-aligned mode (up/down counting)

Trong chế độ canh giữa, bộ đếm sẽ đếm từ 0 đến giá trị bé hơn giá trị tự động nạp lại 1 đơn vị(TIMx_ARR -1) và tạo sự kiện tràn, tiếp đó bộ đếm sẽ đếm từ giá trị tự nạp lại trở về 1 rồi cũng tạo 1 sự kiện tràn. Sau đó, bộ đếm lại đếm lên lại từ 0.

Trong chế độ canh giữa, bit DIR trong thanh ghi TIMx_CR1 giúp ta xác định được chiều đếm hiện tại của bộ đếm và trạng thái bit này được cập nhật bằng phần cứng, người dùng không được phép ghi.

Sự kiện cập nhật có thể được tạo ra mỗi khi:

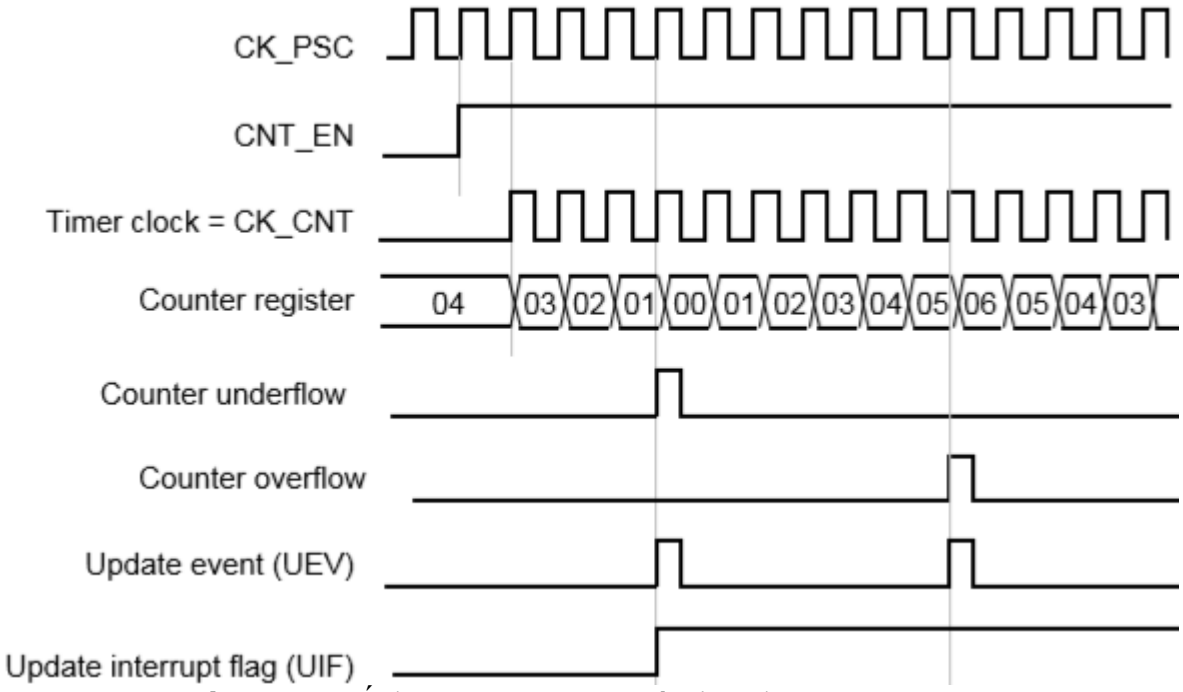
- Bộ đếm tràn
- Set bit UG trong thanh ghi TIMx_EGR bằng phần mềm hoặc sử dụng chế độ tớ. Trong trường hợp này bộ đếm sẽ đếm lại từ 0

Sự kiện cập nhật UEV có thể được tắt bằng phần mềm bằng cách set bit UDIS trong thanh ghi TIMx_CR1. Việc tắt sự kiện cập nhật giúp bộ đếm không bị cập nhật giá trị cũ trong khi đang cài đặt giá trị mới.

Khi sự kiện cập nhật xảy ra thì:

- Cờ cập nhật được set
- Giá trị đếm lặp được nạp lại từ thanh ghi TIMx_RCR
- Giá trị của bộ chia trước được nạp lại từ thanh ghi TIMx_PSC

Sự kiện cập nhật có thể tạo ra yêu cầu ngắt hoặc DMA



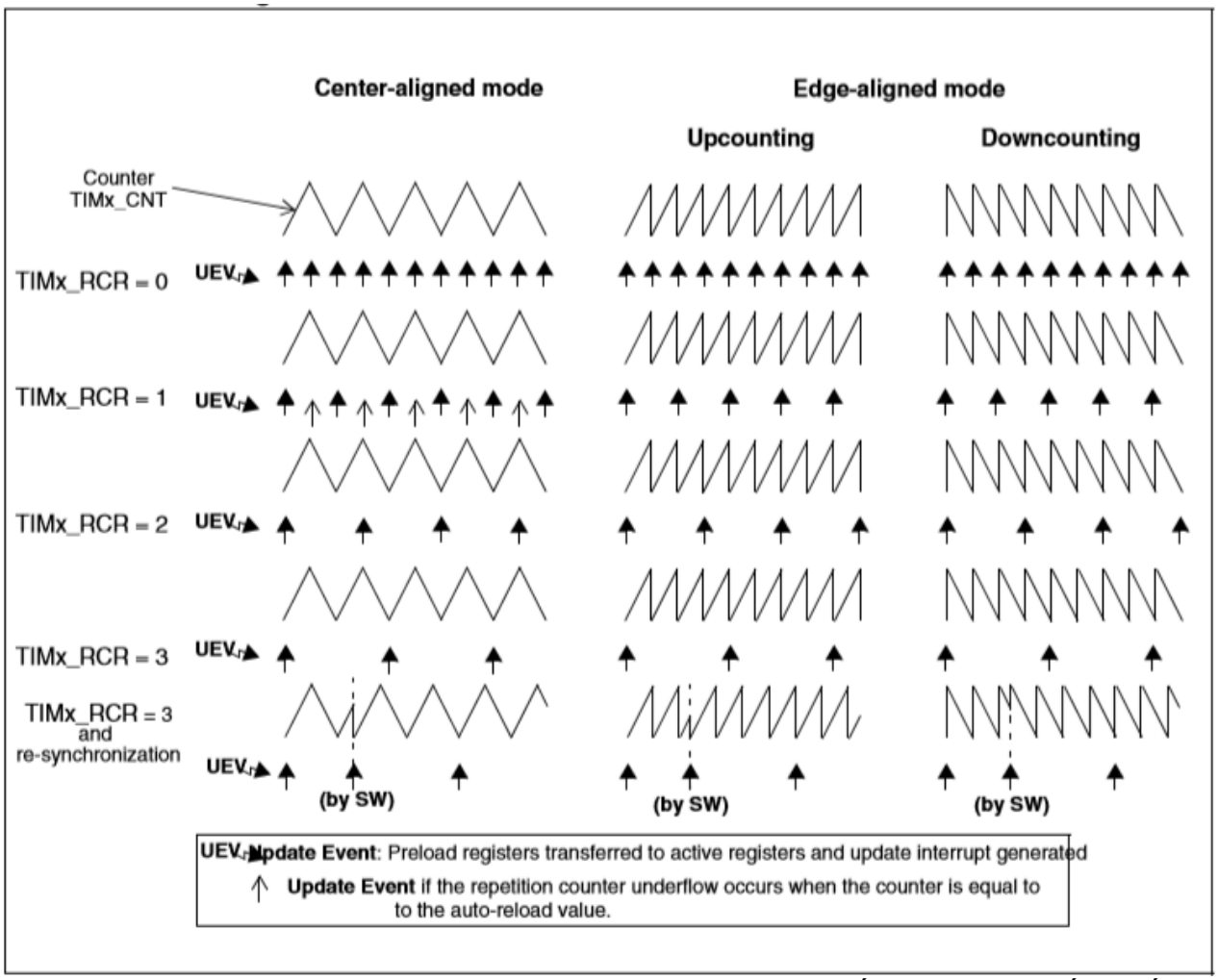
Hình 8.4 Chế độ canh giữa với giá trị tự động nạp lại la 0x06

8.3.2 Chế độ đếm lặp

Sự kiện cập nhật UEV chỉ xảy ra khi giá trị của bộ đếm lặp bằng 0. Giá trị đếm lặp được cài đặt bởi người lập trình và giá trị này chỉ giảm khi:

- Mỗi lần bộ đếm tràn trên ở chế độ đếm lên
- Mỗi lần bộ đếm tràn dưới ở chế độ đếm xuống
- Mỗi lần bộ đếm tràn trên và tràn dưới ở chế độ đếm lên xuống

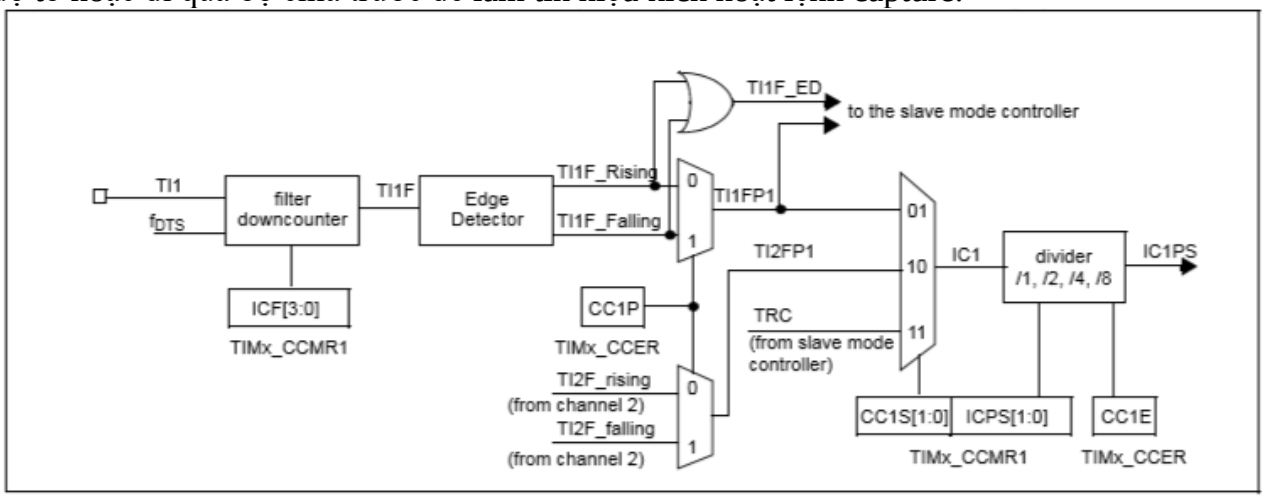
Khi tạo ra sự kiện cập nhật bằng phần mềm(set bit UG trong thanh ghi TIMx_EGR) hoặc bằng phần cứng thông qua chế độ tớ thì quá trình cập nhật xảy ra ngay lập tức bất chấp giá trị đếm lặp hiện tại là bao nhiêu và lúc này giá trị đếm lặp được nạp lại thông qua thanh ghi TIMx_RCR.



Hình 8.5 Quan hệ giữa sự kiện cập nhật với giá trị đếm lặp của 3 chế độ đếm

8.3.3 Chế độ input capture

Tín hiệu cần capture thông qua ngõ vào TIx qua bộ lọc trở thành tín hiệu TIxF, sau đó tín hiệu này được đưa qua mạch phát hiện cạnh với với cạnh tích cực đã được lập trình sẵn để trở thành tín hiệu TIxFPx. Tín hiệu TIxFPx có thể được dùng làm ngõ vào của timer khác trong chế độ tớ hoặc đi qua bộ chia trước để làm tín hiệu kích hoạt lệnh capture.



Hình 8.6 Mạch ngõ vào của tín hiệu capture

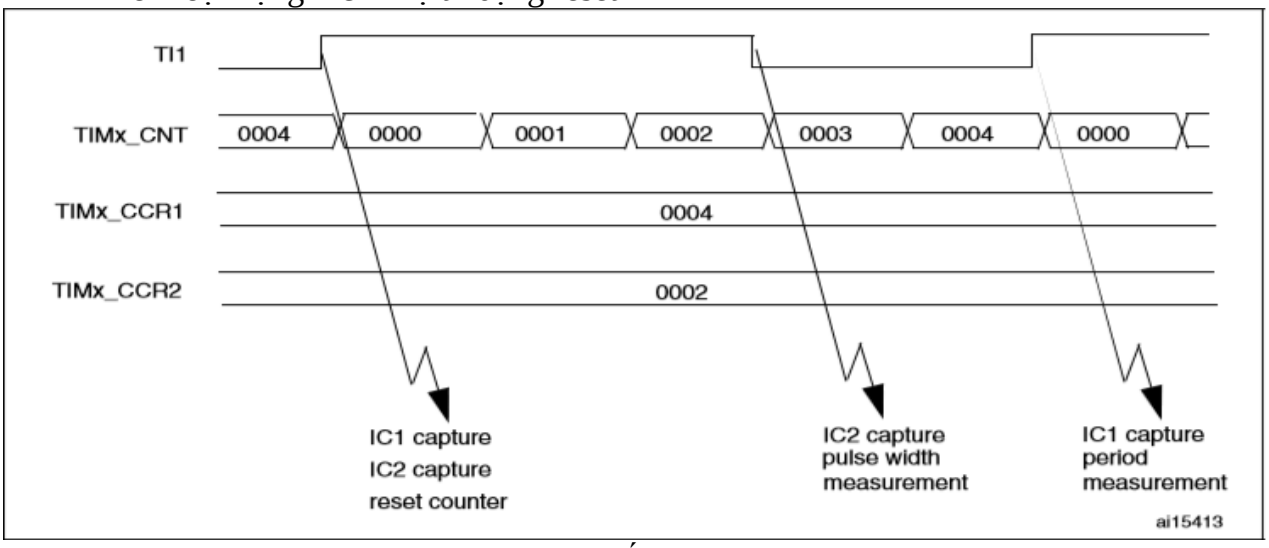
Trong chế độ input capture, thanh ghi compare/capture(TIMx_CCRx) được sử dụng để lưu giá trị đếm hiện tại của bộ đếm ngay sau khi nhận được tín hiệu yêu cầu capture ICx. Khi quá trình capture xảy ra cờ tương ứng CCxIF được set và có thể phát sinh yêu cầu ngắt hoặc DMA

nếu trước đó đã cho phép chúng. Cờ CCxIF có thể được xóa bằng phần mềm thông qua việc ghi bit ‘0’ vào nó hoặc tự động được xóa bằng phần cứng khi người dùng đọc giá trị capture được lưu trong thanh ghi TIMx_CCRx.

8.3.4 Chế độ PWM input

Chế độ này cũng tương tự như chế độ input capture ngoại trừ:

- 2 tín hiệu ICx được tạo ra từ cùng một ngõ vào Tix
- 2 tín hiệu ICx này tích cực bằng 2 cạnh đối lập nhau
- 1 trong 2 tín hiệu TIxFP được chọn làm ngõ vào kích hoạt
- Timer hoạt động ở chế độ tớ dạng reset



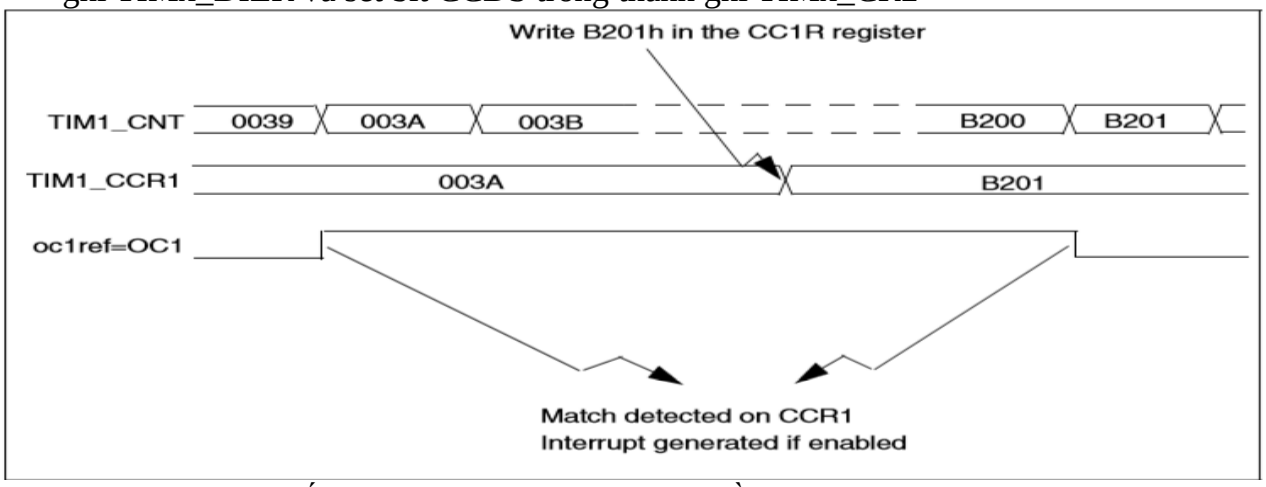
Hình 8.7 ví dụ chế độ PWM input

Chú ý: chỉ có kênh 1 và kênh 2 của timer mới sử dụng được chức năng PWM input.

8.3.5 Chế độ output compare

Chế độ này được dùng để điều khiển dạng sóng điện áp ngõ ra hoặc để biểu thị việc kết thúc chu kỳ đếm. Khi giá trị đếm bằng với gia trị trong thanh ghi capture/compare thì bộ so sánh ngõ ra sẽ:

- Gán trạng thái chân được sử dụng làm output compare bằng một trạng thái đã được lập trình trước đó(tích cực, không tích cực hoặc đảo trạng thái).
- Set cờ ngắt trong thanh ghi trạng thái ngắt(TIMx_SR) và tạo yêu cầu ngắt nếu trước đó đã cho phép bằng cách set bit CCXIE trong thanh ghi TIMx_DIER.
- Tạo yêu cầu DMA nếu trước đó đã cho phép DMA bằng cách set bit CCxDE trong thanh ghi TIMx_DIER và set bit CCDS trong thanh ghi TIMx_CR2



Hình 8.8 chế độ output compare tích cực bằng việc đảo trạng thái chân

8.3.6 Chế độ PWM

Chế độ PWM(điều biến độ rộng xung) cho phép người dùng có thể tạo ra tín hiệu xung vuông có tần số được định bằng giá trị trong thanh thi TIMx_ARR và duty định bằng giá trị trong thanh ghi TIMx_CCRx.

Mỗi kênh của timer có thể cấu hình được chế độ PWM độc lập với nhau.

Mỗi ngõ ra OCx có thể cấu hình đươc mức tích cực cao hoặc thấp bằng phần mềm.

Trong chế độ PWM(1 hoặc 2) giá trị của thanh ghi TIMx_CNT và thanh ghi TIMx_CCRx luôn được so sánh với nhau để để xác định IMx_CCRx ≤ TIMx_CNT hoặc TIMx_CNT ≤ TIMx_CCRx (phụ thuộc vào chiều đếm của bộ đếm).

Bộ định thời có thể tạo PWM theo 2 chế độ là canh theo cạnh(edge-aligned mode) hoặc canh giữa(center-aligned mode)

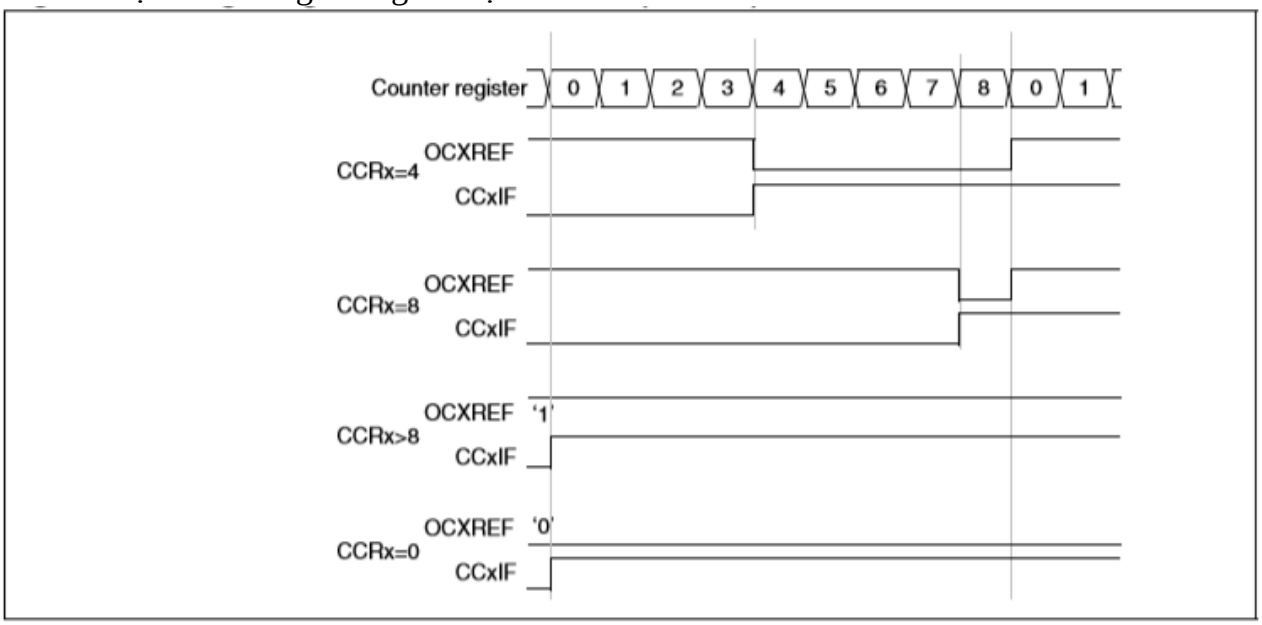
a. Chế độ PWM canh theo cạnh (PWM edge-aligned mode)

Nếu bộ đếm được cấu hình ở chế độ đếm lên:

- Tín hiệu tham chiếu PWM(OCxREF) ở mức cao khi TIMx_CNT < TIMx_CCRx và ngược lại thì tín hiệu này ở mức thấp.
- Nếu giá trị của thanh ghi TIMx_CCRx lớn hơn giá trị tự động nạp lại của thanh ghi TIMx_ARR thì tín hiệu OCxREF luôn ở mức cao.
- Nếu giá trị so sánh bằng 0 thì tín hiệu OCxREF luôn ở mức thấp.

Nếu bộ đếm được cấu hình ở chế độ đếm xuống

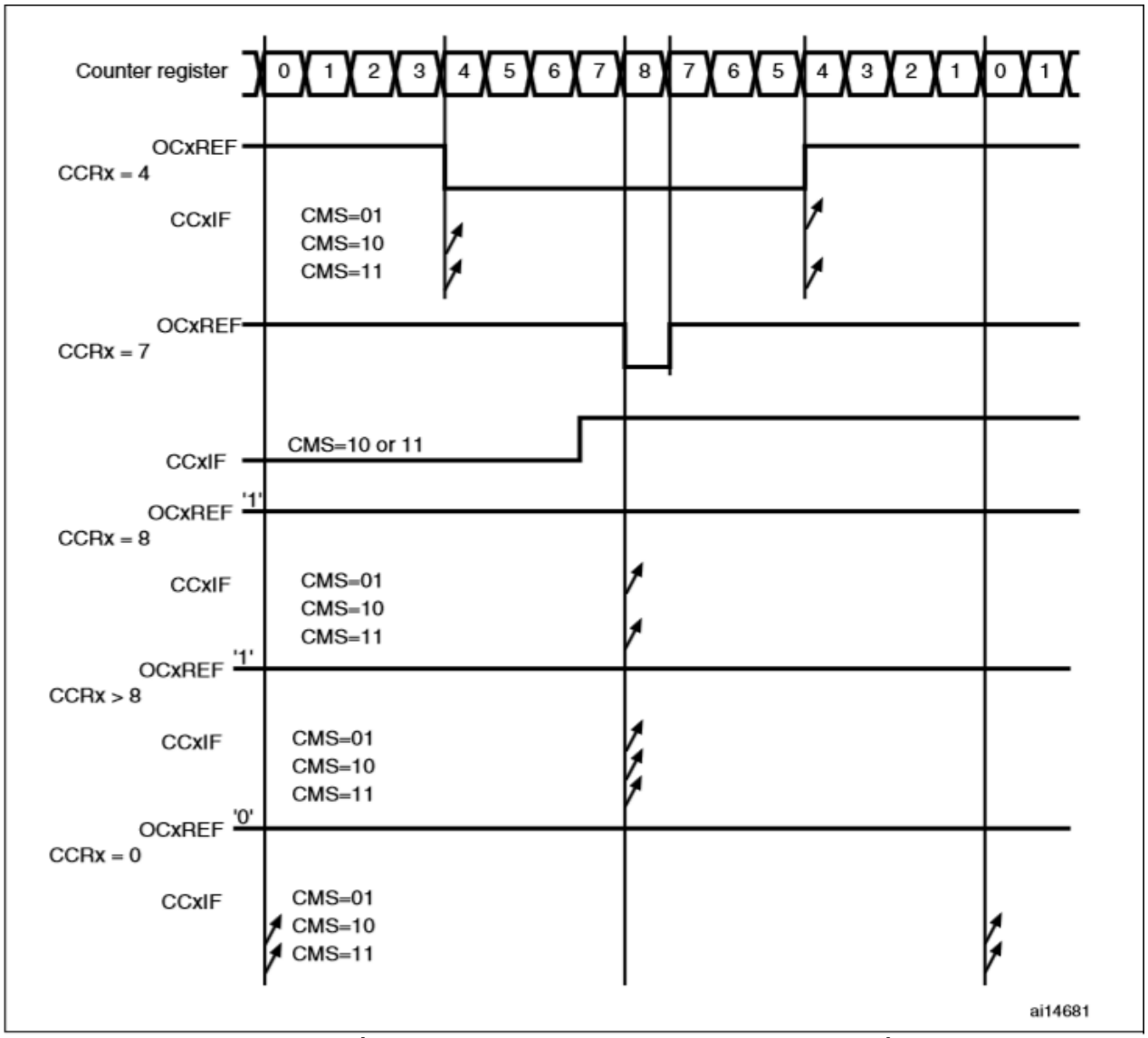
- Tín hiệu OCxREF ở mức thấp khi TIMx_CNT > TIMx_CCRx và ngược lại thì mức cao
- Nếu giá trị so sánh lưu trong thanh ghi TIMx_CCRx lớn hơn giá trị tự động nạp lại của thanh ghi TIMx_ARR thì tín hiệu OCxREF luôn ở mức 1.
- Chế độ đếm xuống không hỗ trợ PWM 0%.



Hình 8.9 chế độ PWM canh theo cạnh với giá trị nạp lại bằng 8

b. Chế độ PWM canh giữa (PWM center -aligned mode)

Chế độ canh giữa được bật bằng các bit CMS trong thanh ghi TIMx_CR1 khi chúng được cài đặt khác "00"**. Hình 8.10** là ví về chế độ PWM chế độ 1 canh giữa với TIMx_ARR = 8.



Hình 8.10 chế độ PWM1 canh giữa với giá trị nạp lại bằng 8

Chú ý:

CMS[1:0] là các bit lựa chọn chế độ canh giữa hoặc canh theo cạnh:

- "00": Canh theo cạnh - đếm lên và xuống độc lập với nhau và phục thuộc vào bit hướng(DIR).
- "01": Chế độ canh giữa 1- đếm lên và xuống luân phiên. Cờ ngắt out compare được set khi đếm xuống
- "01": Chế độ canh giữa 2- đếm lên và xuống luân phiên. Cờ ngắt out compare được set khi đếm lên
- "11": Chế độ canh giữa 3- đếm lên và xuống luân phiên. Cờ ngắt out compare được set khi đếm lên và xuống.

Không được chuyển từ chế độ canh theo cạnh sang chế độ canh giữa khi bộ đếm đang bật.

Không nên cài đặt giá trị vào bộ đếm khi nó đang hoạt động ở chế độ canh giữa.

8.3.7 Ngõ ra bổ sung và khả năng chèn vào thời gian trễ

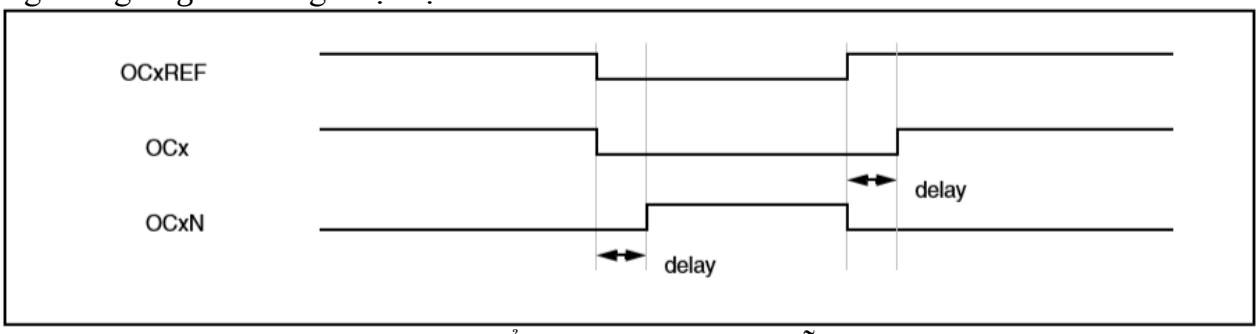
Ngõ ra của timer điều khiển nâng cao(TIM1&TIM8) có thể xuất ra 2 dạng tín hiệu bổ sung cho nhau là OCx và OCxN. Các ngõ ra này có thể được lập trình thời gian trễ để xuất tín hiệu ra sao cho phù hợp với đặc điểm của các thiết bị mà vi điều khiển đang giao tiếp.

Người dùng có thể chọn cạnh hoặc mức tích cực cho ngõ ra chính OCx và ngõ ra bổ sung OcxN một cách độc lập.

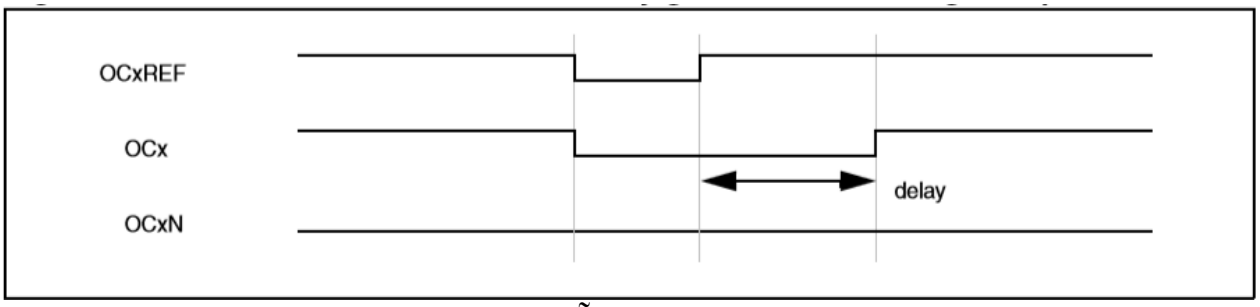
Mỗi kênh có thể được lập trình cấu hình thời gian trễ so với tín hiệu tham chiếu OcxREF bằng một giá trị 10 bit. Nếu 2 ngõ ra OCx và OcxN cấu hình tích cực mức cao thì:

- Ngõ ra tín hiệu OCx giống với tín hiệu tham chiếu ngoại trừ việc cạnh lên bị trễ bằng với lượng thời gian đã được cấu hình so với cạnh lên của tín hiệu tham chiếu.
- Ngõ ra tín hiệu OCxN bị đảo so với tín hiệu tham chiếu và cạnh lên bị trễ bằng với lượng thời gian trễ đã được cấu hình so với cạnh xuống của tín hiệu tham chiếu.

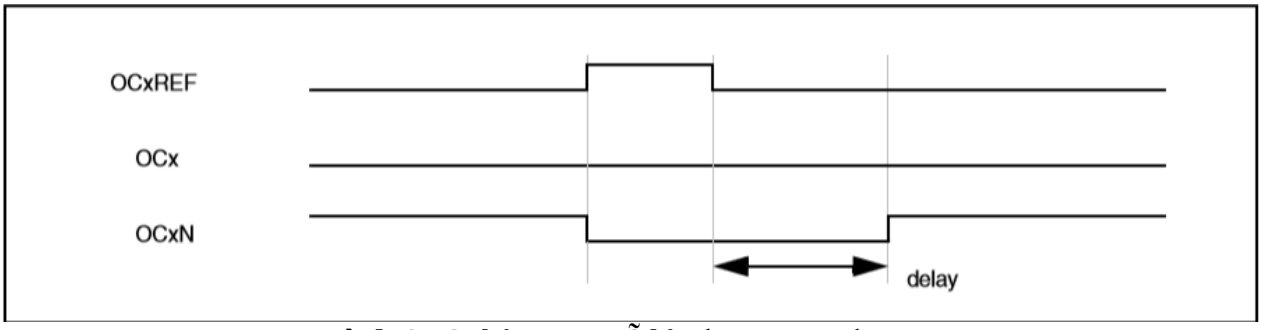
Nếu thời gian trễ được cấu hình lớn hơn bề rộng tích cực của các ngõ ra OCx và OcxN thì xung tương ứng sẽ không được tạo ra.



Hình 8.11 ngõ ra bổ sung và thời gian trễ được chèn vào



Hình 8.12 thời gian trễ lớn hơn xung âm



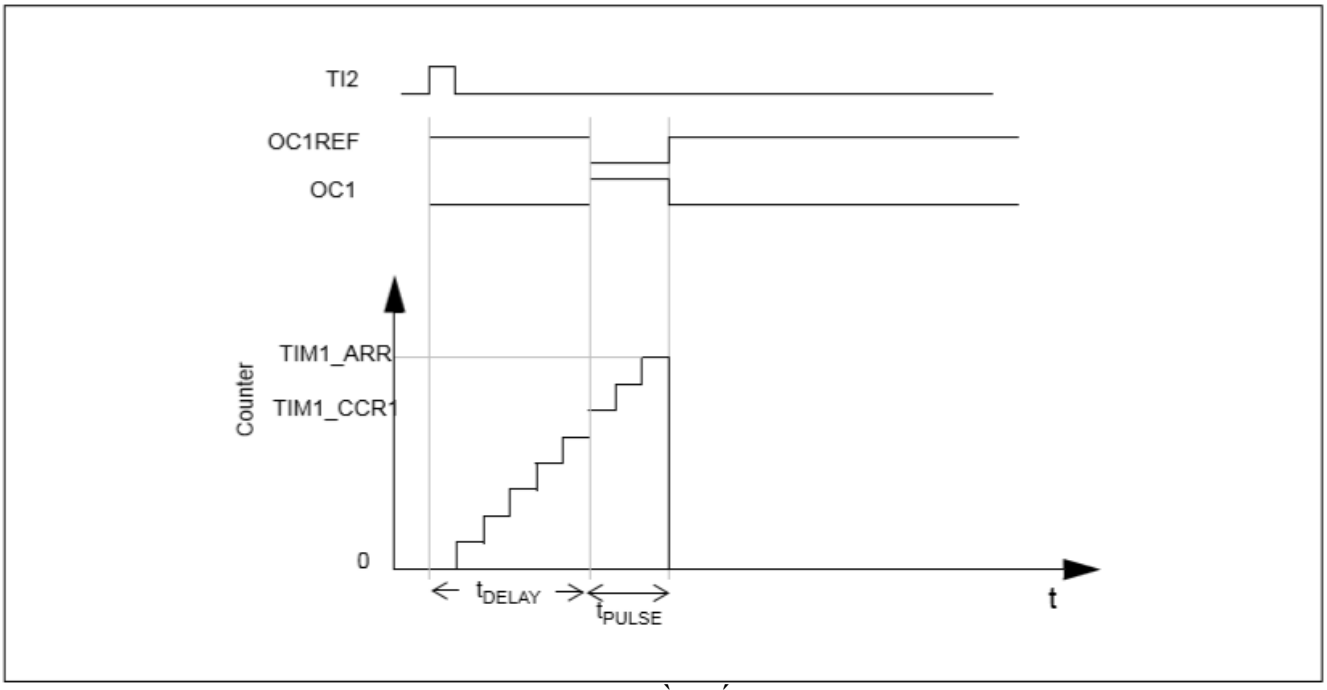
Hình 8.13 thời gian trễ lớn hơn xung dương

8.3.8 Chế độ one-pulse

Chế độ one pulse cho phép người dùng có thể lập trình tạo ra một xung có độ rộng xung và thời gian trễ trước khi xuất xung thay đổi được.

Để tạo ra xung chính xác ta phải cài đặt giá trị đếm và giá trị so sánh khác và phụ thuộc vào chế độ đếm:

- Nếu sử dụng chế độ đếm lên thì : CNT < CCRx ≤ ARR (thường thì hay chọn 0 < CCRx)
- Nếu sử dụng chế độ đếm xuống thì: CNT > CCRx



Hình 8.14 ví dụ về chế độ one pulse

Thông số thời gian trễ t_{DELAY} được định bởi giá trị so sánh trong thanh ghi TIMx_CCR1

Thông số độ rộng xung t_{PULSE} được định bởi hiệu số của giá trị tự động nạp lại với giá trị so sánh(TIMx_ARR - TIMx_CCR1).

Đối với chế độ one pulse tín hiệu cạnh từ ngõ vào TIx sẽ đóng vai trò là tín hiệu cho phép timer đếm, giá trị đếm này sẽ được so sánh với các giá trị đã được cài đặt sẵn trong TIMx_ARR và TIMx_CCR1 để điều khiển đảo ngõ ra.

8.3.9 Chế độ giao tiếp với encoder

Chế độ này cho phép người dùng giao tiếp với encoder theo 3 dạng sau:

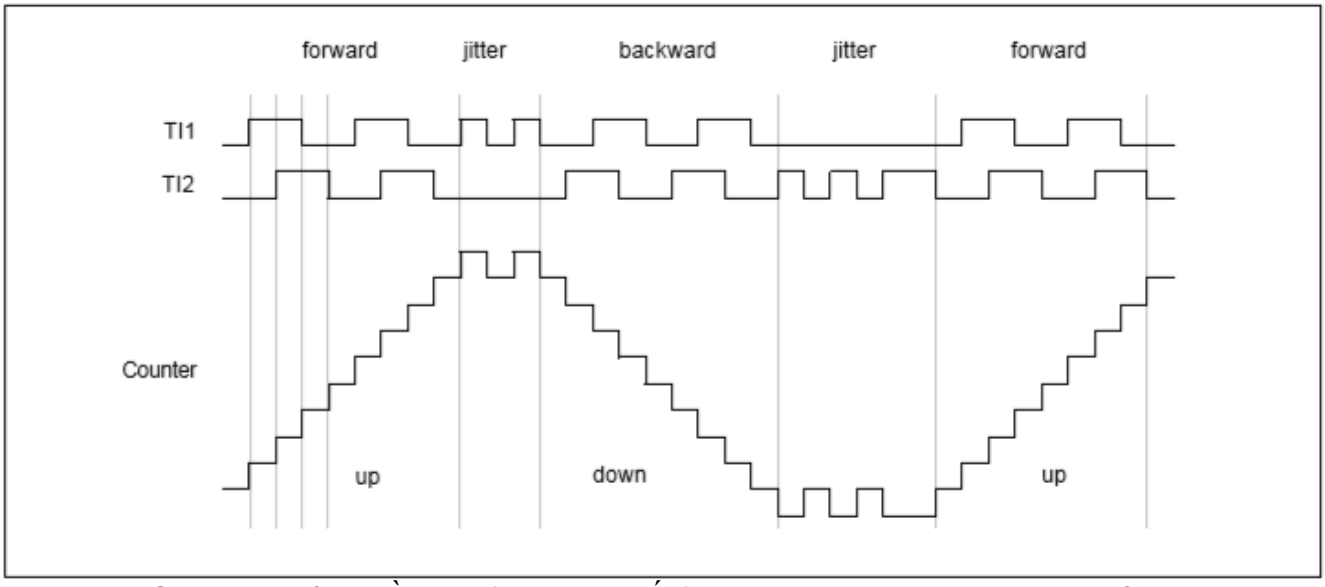
- Kết nối ngõ ra của encoder với TI1
- Kết nối ngõ ra của encoder với TI2
- Kết nối ngõ ra của encoder với cả TI1 và TI2

Chế độ giao tiếp với encoder và chế độ đếm xung clock ngoại thứ 2(External clock mode 2) không tương thích với nhau do đó không được cấu hình sử dụng cả 2 chế độ này cùng lúc.

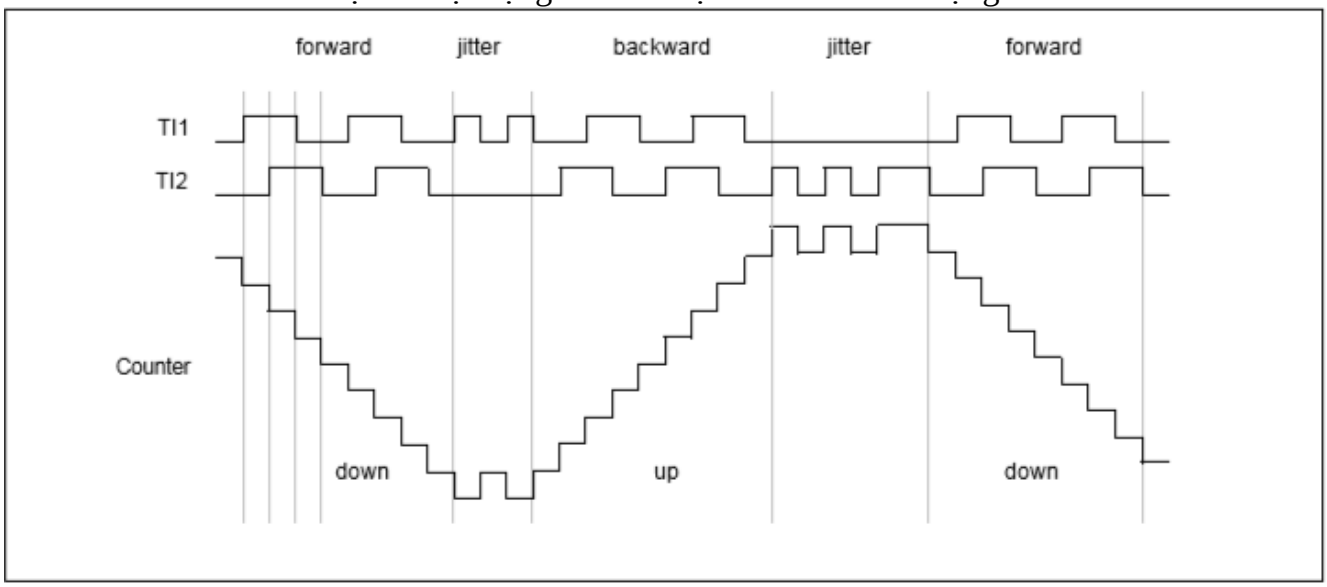
Trong chế độ này bộ đếm được điều chỉnh một cách tự động phụ thuộc vào tốc độ và hướng của encoder vì vậy thông qua giá trị đếm được ta có thể xác định được vị trí và tốc độ quay của động cơ. **Bảng 8.1** mô tả mối liên hệ giữa hướng đếm với các chế độ sử dụng.

Bảng 8.1 mối liên hệ giữa hướng đếm với các chế độ sử dụng

Cạnh tích cực	Mức logic của tín hiệu còn lại giữa TI1FP1 và TI2FP2	Tín hiệu TI1FP1		Tín hiệu TI2FP2	
		Cạnh lên	Cạnh xuống	Cạnh lên	Cạnh xuống
Đếm xung ở TI1	Cao	Đếm xuống	Đếm lên	Không đếm	Không đếm
	Thấp	Đếm lên	Đếm xuống	Không đếm	Không đếm
Đếm xung ở TI2	Cao	Không đếm	Không đếm	Đếm lên	Đếm xuống
	Thấp	Không đếm	Không đếm	Đếm xuống	Đếm lên
Đếm xung trên cả TI1 và TI2	Cao	Đếm xuống	Đếm lên	Đếm lên	Đếm xuống
	Thấp	Đếm lên	Đếm xuống	Đếm xuống	Đếm lên



Hình 8.15 Ví dụ về hoạt động của chế độ encoder khi sử dụng cả TI1 và TI2



Hình 8.16 Ví dụ về hoạt động của chế độ encoder với tín hiệu TI1FP1 nghịch đảo

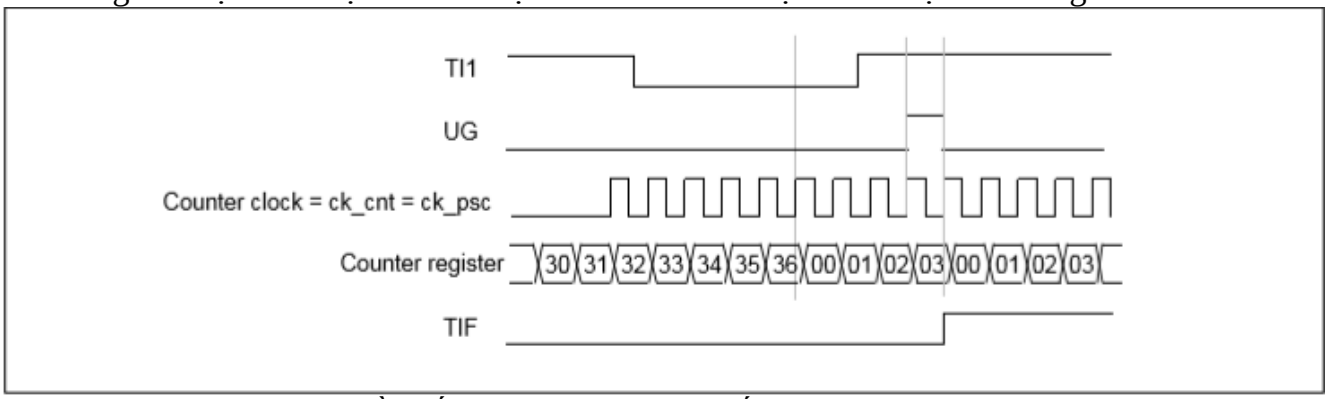
8.3.10 Chế độ tớ(slave mode)

Timer có thể được đồng bộ với tín hiệu kích hoạt từ bên ngoài thông qua các chế độ tớ

- Chế độ Reset
- Chế độ Gate
- Chế độ Trigger

a. Chế độ Reset

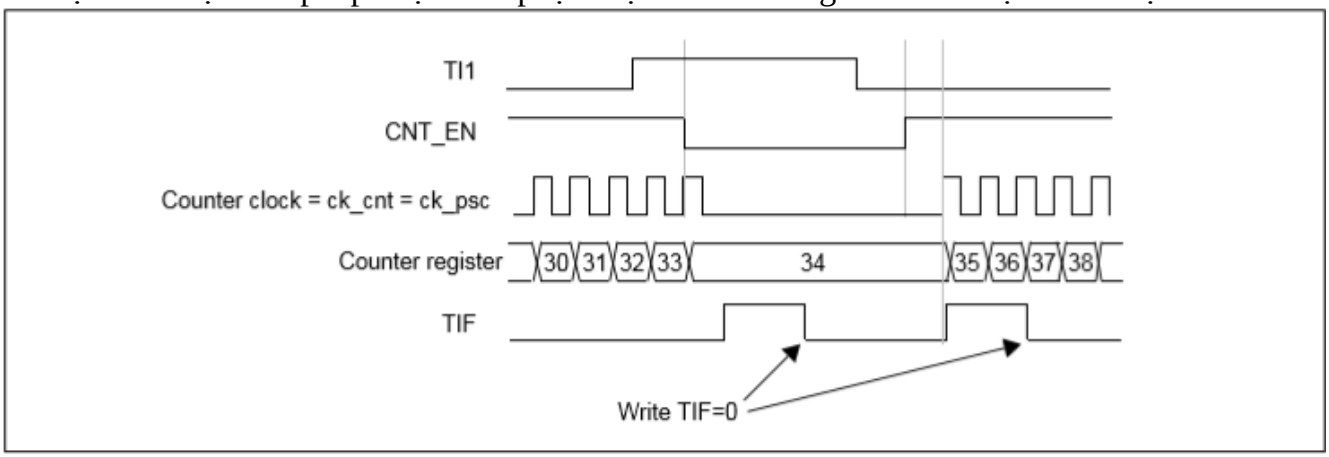
Trong chế độ Reset bộ đếm sẽ được xóa khi có tín hiệu kích hoạt từ bên ngoài.



Hình 8.17 Ví dụ về chế độ Reset khi timer đếm lên với giá trị tự nạp lại là 0x36

b. Chế độ Gate

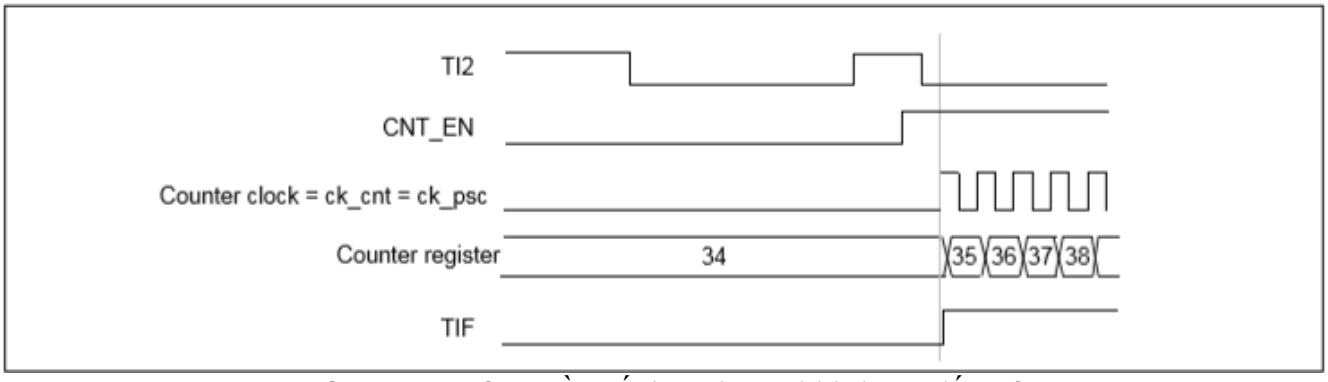
Bộ đếm được cho phép hoặc cấm phụ thuộc vào mức logic của tín hiệu kích hoạt.



Hình 8.18 Ví dụ về chế độ Gate khi timer đếm lên với giá trị tự nạp lại là 0x36

c. Chế độ Trigger

Khi nhận được tín hiệu kích hoạt thì bộ đếm mới bắt đầu đếm



Hình 8.19 Ví dụ về chế độ Trigger khi timer đếm lên

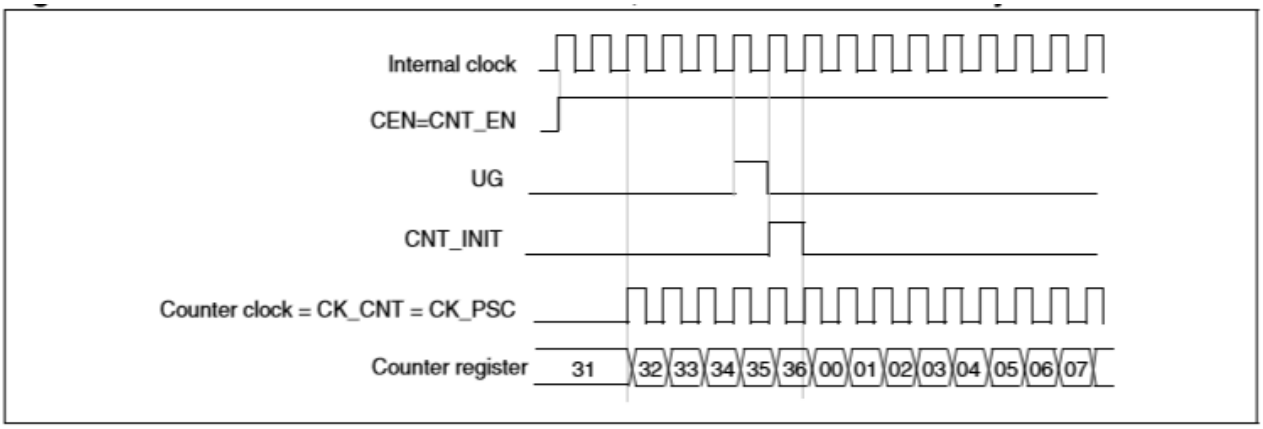
8.4. LỰA CHỌN NGUỒN XUNG CLOCK CHO TIMER

Bộ đếm có thể được cung cấp xung clock từ các nguồn sau đây:

- Xung clock nội(CK_INT)
- Xung clock ngoại chế độ 1(External clock mode1): xung clock từ ngõ vào bên ngoài
- Xung clock ngoại chế độ 2(External clock mode2): xung clock đưa vào chân nhận kích hoạt từ bên ngoài(chân ETR)
- Ngõ nhận xung kích hoạt nội(ITRx): sử dụng 1 timer làm bộ chia trước cho timer khác. Ví dụ sử dụng Timer 1 làm bộ chia trước cho Timer 2

8.4.1 Xung clock nội (Internal clock source - CK_INT)

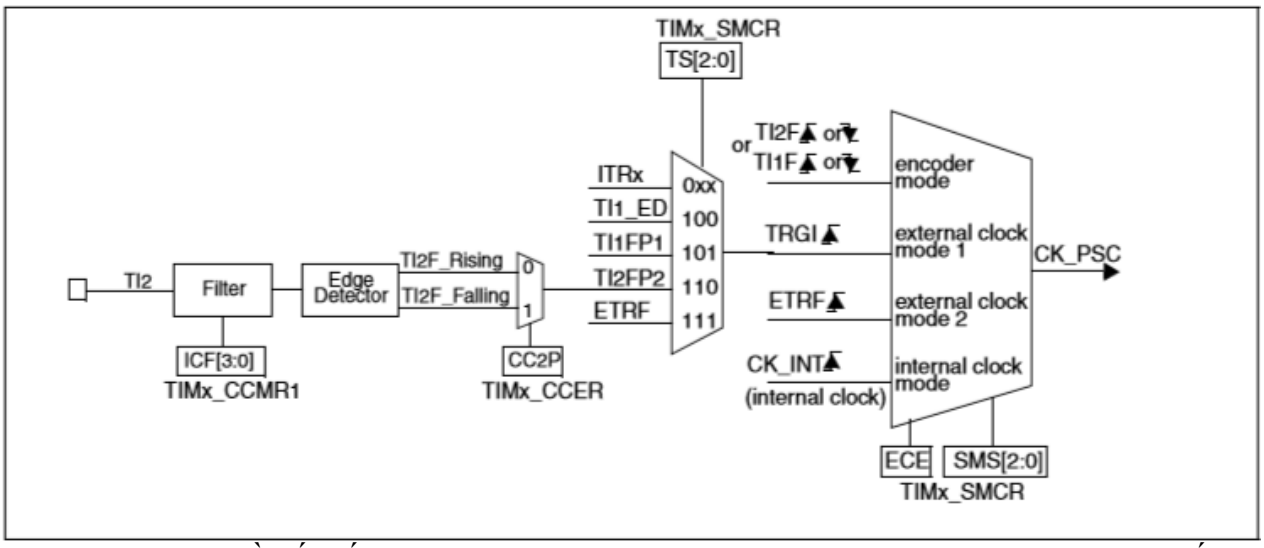
Nếu tắt chế độ tớ(SMS=000) thì các bit CEN, DIR trong thanh ghi TIMx_CR1 và bit UG trong thanh ghi TIMx_EGR được sử dụng như là những bit điều khiển thật sự và chỉ có thể được thay đổi bằng phần mềm(ngoại trừ bit UG được tự động xóa). Ngay khi set bit CEN thì bộ chia trước được cấp xung clock bởi nguồn clock nội CK_INT.



Hình 8.20 Ví dụ về sử dụng xung clock nội với bộ chia trước là 1

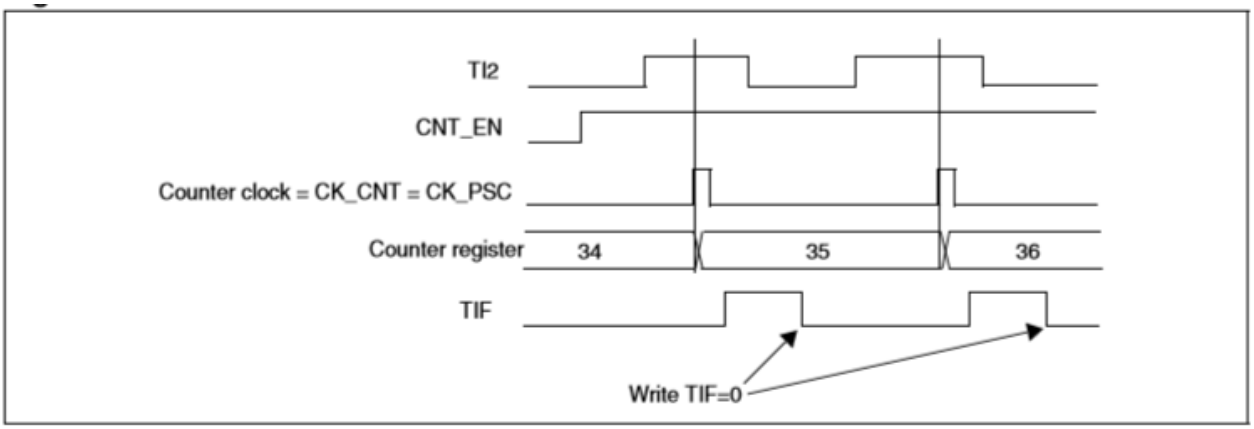
8.4.2 Xung clock ngoại chế độ 1(External clock source mode 1)

Bộ đếm sẽ hoạt động ở chế độ này khi ta cấu hình các bit SMS trong thanh ghi TIMx_SMCR bằng "111". Bộ đếm sẽ đếm khi có xung cạnh xuống hoặc cạnh lên đưa vào ngõ vào đã được chọn trước đó.



Hình 8.21 Sơ đồ kết nối ngõ vào TI2 trước khi được dùng làm xung clock cho bộ đếm

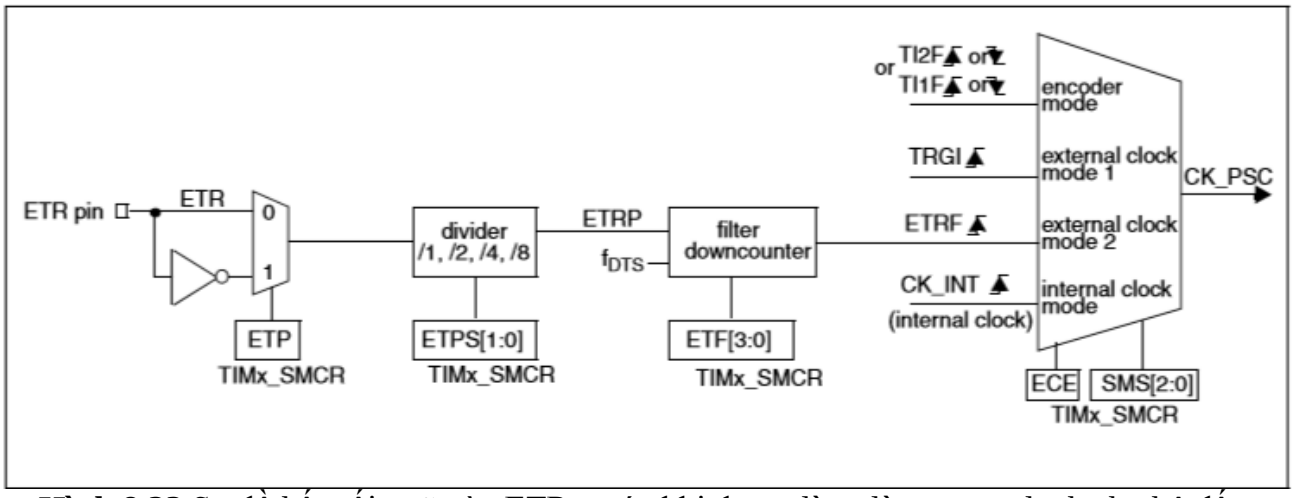
Khi có tín hiệu cạnh lên ở chân TI2 bộ đếm sẽ đếm lên 1 đơn vị và và cờ TIF được set. Thời gian trễ từ khi xuất hiện xung cạnh lên ở TI2 đến khi bộ đếm đếm phụ thuộc vào mạch đồng bộ ở ngõ vào TI2.



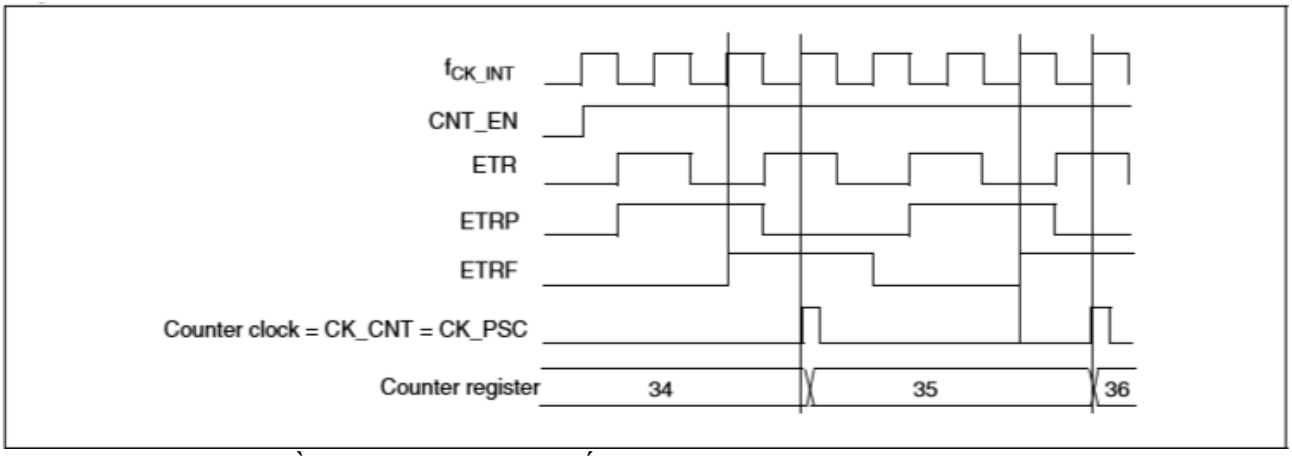
Hình 8.22 Ví dụ về hoạt động của bộ đếm khi sử dụng xung clock từ TI2

8.4.3 Xung clock ngoại chế độ 2(External clock source mode 2)

Để cấu hình bộ đếm hoạt động ở chế độ này ta set bit ECE trong thanh ghi TIMx_SMCR. Bộ đếm sẽ đếm mỗi khi có xung cạnh lên hoặc cạnh xuống tác động vào chân ETR.



Hình 8.23 Sơ đồ kết nối ngõ vào ETR trước khi được dùng làm xung clock cho bộ đếm



Hình 8.24 Ví dụ về hoạt động của bộ đếm khi sử dụng xung clock từ ETR qua bộ chia 2

8.5. CÁC LỆNH THƯỜNG DÙNG LIÊN QUAN ĐẾN TIMER

Bảng 8.2 Các lệnh thông dụng dùng để cấu hình timer base

SỬ DỤNG THƯ VIỆN "stm32f10x_tim"	
Lệnh	
Thông số hay dùng	**Giải thích**
TIM_TimeBaseInitTypeDef A; (Khai báo biến A thuộc kiểu TIM_TimeBaseInitTypeDef)	
A.TIM_Period = B; (Lệnh chọn chu kỳ đếm cho timer- giá trị tự động nạp lại)	
B: [0-65535]	**B:** Giá trị cài đặt Giới hạn từ 0 đến 65535
A. TIM_Prescaler = C; (Lệnh chọn bộ chia(xung) trước khi vào bộ đếm)	
C: [0-65535]	**C:** Giá trị cài đặt Giới hạn từ 0 đến 65535

A. TIM_ClockDivision= D; (Lệnh chọn bộ chia xung clock cho timer)	
D: TIM_CKD_DIV1 TIM_CKD_DIV2 TIM_CKD_DIV4	**D:** Bộ chia Chia 1 Chia 2 Chia 4
A.TIM_RepetitionCounter= E; (Lệnh chọn số lần đếm lặp cho timer)	
E: [0-255]	**E:** Số lần đếm lặp Giá trị từ 0 đến 255
A. TIM_CounterMode= F; (Lệnh chọn chế độ hoạt động cho timer)	
F: TIM_CounterMode_Up TIM_CounterMode_Down TIM_CounterMode_CenterAligned1 TIM_CounterMode_CenterAligned2 TIM_CounterMode_CenterAligned3	**F:** Chế độ hoạt động Đếm lên Đếm xuống Đếm lên xuống 1 Đếm lên xuống 2 Đếm lên xuống 3
TIM_TimeBaseInit(G, &A); (Lệnh cài đặt các thông số đã cấu hình cho Timer)	
G: TIM1 … TIMn	**G:** Timer cần cấu hình Timer 1 … Timer n

Bảng 8.3 Các lệnh thông dụng dùng để cấu hình input capture

SỬ DỤNG THƯ VIỆN "stm32f10x_tim"	
Lệnh	
Thông số hay dùng	**Giải thích**
TIM_ICInitTypeDef A; (Khai báo biến A thuộc kiểu TIM_ICInitTypeDef)	
A. TIM_Channel =B; (Lệnh chọn kênh Input Capture)	
B: TIM_Channel_1 TIM_Channel_2 TIM_Channel_3 TIM_Channel_4	**B:** Kênh được chọn Kênh 1 Kênh 2 Kênh 3 Kênh 4
A. TIM_ICPolarity = C; (Lệnh chọn cạnh tích cực cho tín hiệu Input Capture)	
C: TIM_ICPolarity_Rising TIM_ICPolarity_Falling TIM_ICPolarity_BothEdge	**C:** Cạnh tích cực Tích cực cạnh lên Tích cực cạnh xuống Tích cực cả cạnh lên lẫn cạnh xuống

A. TIM_ICSelection= D; (Lệnh cấu hình cách kết nối các tín hiệu Input Capture)	
D: TIM_ICSelection_DirectTI TIM_ICSelection_IndirectTI TIM_ICSelection_TRC	**D:** Cách kết nối Timer Input 1, 2, 3, 4 được kết nối tương ứng với Input Capture 1, 2, 3, và 4 Timer Input 1, 2, 3, 4 được kết nối tương ứng với Input Capture 2, 1, 4, 3 Các ngõ Input Capture kết nối với TRC
A. TIM_ICPrescaler = E; (Lệnh chọn bộ chia trước cho tín hiệu Input Captuer)	
E: TIM_ICPSC_DIV1 TIM_ICPSC_DIV2 TIM_ICPSC_DIV4 TIM_ICPSC_DIV8	**E:** Bộ chia trước Chia 1 Chia 2 Chia 4 Chia 8
A. TIM_ICFilter = F; (Lệnh chọn hệ số lọc cho tín hiệu Input Capture)	
F: [0-15]	**F:** Hệ số lọc Giá trị từ 0 đến 15
TIM_ICInit (G,&A); (Lệnh cài đặt các thông số đã chọn để Timer hoạt động ở chế độ Capture bình thường)	
G: TIM1 … TIMn	**G:** Timer cần cấu hình Timer 1 … Timer n
TIM_PWMIConfig(G, &A); (Lệnh cài đặt các thông số đã chọn để timer hoạt động ở chế độ Captuer xung PWM)	
G: TIM1 … TIMn	**G:** Timer cần cấu hình Timer 1 … Timer n
TIM_SelectInputTrigger(G, H) ; (Lệnh chọn tín hiệu kích hoạt từ bên ngoài)	
G: TIM1 … TIMn **H:** TIM_TS_ITR0 TIM_TS_ITR1 TIM_TS_ITR2 TIM_TS_ITR3 TIM_TS_TI1F_ED TIM_TS_TI1FP1 TIM_TS_TI2FP2 TIM_TS_ETRF	**G:** Timer cần cấu hình Timer 1 … Timer n **H:** Tín hiệu kích hoạt được chọn ITR0 (từ timer 1 trigger out) ITR1 (từ timer 2 trigger out) ITR2 (từ timer 3 trigger out) ITR3 (từ timer 4 trigger out) TI1(CH1) qua mạch phát hiện cạnh TI1(CH1) qua mạch lọc TI2(CH2) qua mạch lọc ETR qua mạch lọc

Bảng 8.4 Các lệnh thông dụng dùng để cấu hình ouput compare

SỬ DỤNG THƯ VIỆN "stm32f10x_tim"	
Lệnh	
Thông số hay dùng	**Giải thích**
TIM_OCInitTypeDef A; (Khai báo biến A thuộc kiểu TIM_OCInitTypeDef)	
A.TIM_OCMode = B; (Lệnh chọn chế độ hoạt động cho ouput compare)	
B: TIM_OCMode_Timing TIM_OCMode_Active TIM_OCMode_Inactive TIM_OCMode_Toggle TIM_OCMode_PWM1 TIM_OCMode_PWM2	**B:** chế độ hoạt động Chế độ định thời Chế độ tích cực Chế độ không tích cực Chế độ nghịch đảo ngõ ra Chế độ tạo xung PWM ở ngõ ra loại 1 Chế độ tạo xung PWM ở ngõ ra loại 2 (PWM loại 1 và loại 2 nghịch đảo nhau)
A.TIM_OutputState = C; (Lệnh cho phép hoặc cấm Output Compare xuất tín hiệu ra chân CHx của Timer)	
C: TIM_OutputState_Disable TIM_OutputState_Enable	**C:** Cho phép hoặc cấm Không cho phép Cho phép
A.TIM_OutputNState = D; (Lệnh cho phép hoặc cấm Output Compare xuất tín hiệu ra các chân CHxN) (Chỉ dùng cho TIMER 1 và TIMER 8)	
D: TIM_OutputNState_Disable TIM_OutputNState_Enable	**D:** Cho phép hoặc cấm Không cho phép Cho phép
A.TIM_OCPolarity = E; (Lệnh cấu hình mức tích cực của Output Compare đối với các chân CHx)	
E: TIM_OCPolarity_High TIM_OCPolarity_Low	**E:** Mức tích cực Tích cực mức cao Tích cực mức thấp
A.TIM_OCNPolarity = F; (Lệnh cấu hình mức tích cực của Output Compare đối với các chân CHxN) (Chỉ dùng cho TIMER 1 và TIMER 8)	
F: TIM_OCNPolarity_High TIM_OCNPolarity_Low	**F:** Mức tích cực Tích cực mức cao Tích cực mức thấp
A.TIM_Pulse = G; (Lệnh cấu hình giá trị so sánh cho Output Compare)	
G: **[0-65535]**	**G:** Giá trị so sánh Giới hạn từ 0 đến 65535
TIM_OC1Init(H, &A); **TIM_OC2Init(H, &A);** **TIM_OC3Init(H, &A);** **TIM_OC4Init(H, &A);** (Các lệnh cấu hình những thông số được lưu trong biến A các bộ output compare)	

H: TIM1 … TIMn	**H:** Timer cần cấu hình Timer 1 … Timer n
TIM_SelectOutputTrigger (H, I); (Lệnh cấu hình tín hiệu kích hoạt xuất ra bên ngoài- TRGO)	
H: TIM1 … TIMn **I:** TIM_TRGOSource_Reset TIM_TRGOSource_Enable TIM_TRGOSource_Update TIM_TRGOSource_OC1 TIM_TRGOSource_OC1Ref TIM_TRGOSource_OC2Ref TIM_TRGOSource_OC3Ref TIM_TRGOSource_OC4Ref	**H:** timer cần cấu hình TIM1 … TIMn **I:** Nguồn xung kích hoạt Xuất xung kích hoạt khi timer reset Xuất xung kích hoạt khi cho phép timer Xuất xung kích hoạt khi timer cập nhật Xung kích hoạt là tín hiệu OC1 Xung kích hoạt là tín hiệu OC1Ref Xung kích hoạt là tín hiệu OC2Ref Xung kích hoạt là tín hiệu OC3Ref Xung kích hoạt là tín hiệu OC4Ref
TIM_CtrlPWMOutputs(H, J) (Lệnh cho phép hoặc cấm xuất xung PWM ở ngõ ra)	
H: TIM1 … TIMn **J:** ENABLE DISABLE	**H:** Timer cần cấu hình Timer 1 … Timer n **J:** Cho phép hoặc cấm Cho phép Cấm

Bảng 8.5 Các lệnh thông dụng để chọn lựa xung clock cấp cho timer

SỬ DỤNG THƯ VIỆN "stm32f10x_tim"	
Lệnh	
Thông số hay dùng	**Giải thích**
TIM_ITRxExternalClockConfig(A, B) (Lệnh cấu hình các tín hiệu ITRx làm xung clock cho timer)	
A: TIM1 … TIMn **B:** TIM_TS_ITR0 TIM_TS_ITR1 TIM_TS_ITR2 TIM_TS_ITR3	**A:** timer cần cấu hình Timer 1 … Timer n **B:** Nguồn xung clock được chọn ITR0 (từ timer 1 trigger out) ITR1 (từ timer 2 trigger out) ITR2 (từ timer 3 trigger out) ITR3 (từ timer 4 trigger out)

TIM_TIxExternalClockConfig(A, B, C, D); (Lệnh cấu hình Timer đếm xung đưa vào TI)	
A:	**A:** timer cần cấu hình
TIM1	Timer 1
…	…
TIMn	Timer n
B:	**B:** Chọn xung ngõ vào cho bộ đếm
TIM_TIxExternalCLK1Source_TI1ED	TI1(CH1) qua bộ phát hiện cạnh
TIM_TIxExternalCLK1Source_TI1	TI1(CH1) qua mạch lọc
TIM_TIxExternalCLK1Source_TI2	TI2(CH2) qua mạch lọc
C:	**C:** Chọn cạnh tích cực
TIM_ICPolarity_Rising	Tích cực cạnh lên
TIM_ICPolarity_Falling	Tích cực cạnh xuống
D:	**D:** Chọn thông số bộ lọc
[0-15]	Giá trị từ 0 đến 15
TIM_ETRClockMode1Config(A, B, C, D); (Lệnh cấu hình clock ngoại chế độ 1 cho timer- Sử dụng ngõ TRGI cấp xung cho timer)	
A:	**A:** timer cần cấu hình
TIM1	Timer 1
…	…
TIMn	Timer n
B:	**B:** Chọn bộ chia xung
TIM_ExtTRGPSC_OFF	Không dùng bộ chia
TIM_ExtTRGPSC_DIV2	Chọn bộ chia 2
TIM_ExtTRGPSC_DIV4	Chọn bộ chia 4
TIM_ExtTRGPSC_DIV8	Chọn bộ chia 8
C:	**C:** Chọn cạnh hoặc mức tích cực
TIM_ExtTRGPolarity_Inverted	Tích cực cạnh xuống hoặc mức thấp
TIM_ExtTRGPolarity_NonInverted	Tích cực cạnh lên hoặc mức cao
D:	**D:** Chọn thông số bộ lọc
[0-15]	Giá trị từ 0 đến 15
TIM_ETRClockMode2Config(A, B, C, D); (Lệnh cấu hình clock ngoại chế độ 2 cho timer- Sử dụng ngõ ETRF cấp xung cho timer)	
A:	**A:** timer cần cấu hình
TIM1	Timer 1
…	…
TIMn	Timer n
B:	**B:** Chọn bộ chia xung
TIM_ExtTRGPSC_OFF	Không dùng bộ chia
TIM_ExtTRGPSC_DIV2	Chọn bộ chia 2
TIM_ExtTRGPSC_DIV4	Chọn bộ chia 4
TIM_ExtTRGPSC_DIV8	Chọn bộ chia 8
C:	**C:** Chọn cạnh hoặc mức tích cực
TIM_ExtTRGPolarity_Inverted	Tích cực cạnh xuống hoặc mức thấp
TIM_ExtTRGPolarity_NonInverted	Tích cực cạnh lên hoặc mức cao
D:	**D:** Chọn thông số bộ lọc
[0-15]	Giá trị từ 0 đến 15

Bảng 8.6 Các lệnh thông dụng liên quan đến ngắt timer

SỬ DỤNG THƯ VIỆN "stm32f10x_tim"	
Lệnh	
Thông số hay dùng	**Giải thích**
TIM_ITConfig (A, B,C); (Lệnh cấu hình ngắt cho timer)	
A:	**A:** Timer cần cho phép ngắt
TIM1	Timer 1
…	…
TIMn	Timer n
B:	**B:** Chọn nguồn ngắt
TIM_IT_Update	Ngắt khi timer cập nhật
TIM_IT_CC1	Ngắt của bộ Capture Compare 1
TIM_IT_CC2	Ngắt của bộ Capture Compare 2
TIM_IT_CC3	Ngắt của bộ Capture Compare 3
TIM_IT_CC4	Ngắt của bộ Capture Compare 4
TIM_IT_COM	Ngắt có sự kiện Comunication
TIM_IT_Trigger	Ngắt khi có sự kiện Trigger
TIM_IT_Break	Ngắt khi có sự kiện Break Input
C:	**C:** Cho phép hoặc không cho phép
ENABLE	Cho phép
DISABLE	Không cho phép
TIM_GetITStatus(A, B, C); (Lệnh đọc cờ ngắt của timer)	
A:	**A:** Timer cần đọc
TIM1	Timer 1
…	…
TIMn	Timer n
B:	**B:** Cờ ngắt cần đọc
TIM_IT_Update	Ngắt khi timer cập nhật
TIM_IT_CC1	Ngắt của bộ Capture Compare 1
TIM_IT_CC2	Ngắt của bộ Capture Compare 2
TIM_IT_CC3	Ngắt của bộ Capture Compare 3
TIM_IT_CC4	Ngắt của bộ Capture Compare 4
TIM_IT_COM	Ngắt có sự kiện Comunication
TIM_IT_Trigger	Ngắt khi có sự kiện Trigger
TIM_IT_Break	Ngắt khi có sự kiện Break Input
TIM_ClearITPendingBit(A, B); (Lệnh xóa cờ ngắt của Timer)	
A:	**A:** Timer cần xóa
TIM1	Timer 1
…	…
TIMn	Timer n
B:	**B:** Cờ ngắt cần xóa
TIM_IT_Update	Ngắt khi timer cập nhật
…	…
	giống **B** của TIM_GetITStatus(A, **B**, C);

Bảng 8.7 Các lệnh thông dụng liên quan đến chế độ slave

SỬ DỤNG THƯ VIỆN "stm32f10x_tim"	
Lệnh	
Thông số hay dùng	**Giải thích**
TIM_SelectMasterSlaveMode(A, B); (Lệnh cho phép hoặc cấm chế độ chủ tớ)	
A: TIM1 … TIMn **B:** TIM_MasterSlaveMode_Enable TIM_MasterSlaveMode_Disable	**A:** timer cần cấu hình Timer 1 … Timer n **B:** Cho phép hoặc cấm Cho phép Cấm
TIM_SelectSlaveMode(A, B); (Lệnh chọn chế độ hoạt động của Timer tớ)	
A: TIM1 … TIMn **B:** TIM_SlaveMode_Reset TIM_SlaveMode_Gated TIM_SlaveMode_Trigger TIM_SlaveMode_External1	**A:** Timer cần cấu hình Timer 1 … Timer n **B:** Chế độ hoạt động Chế độ Reset Chế độ Gate Chế độ Trigger Chế độ External1(Tín hiệu cạnh lên của TRGI là nguồn xung clock cấp cho bộ đếm)

Bảng 8.8 Các lệnh thông dụng khác của timer

SỬ DỤNG THƯ VIỆN "stm32f10x_tim"	
Lệnh	
Thông số hay dùng	**Giải thích**
TIM_EncoderInterfaceConfig(A, B, C, D); (Lệnh cấu hình ngõ vào Encoder)	
A: TIM1 … TIMn **B:** TIM_EncoderMode_TI1 TIM_EncoderMode_TI2 TIM_EncoderMode_TI12 **C:** TIM_ICPolarity_Falling TIM_ICPolarity_Rising **D:** TIM_ICPolarity_Falling TIM_ICPolarity_Rising	**A:** timer cần điều khiển Timer 1 … Timer n **B:** Ngõ vào của xung encoder TI1 (CH1) TI2 (CH2) TI12(CH1 và CH2) **C:** Chọn cạnh tích cực cho TI1 Cạnh xuống Cạnh lên **D:** Chọn cạnh tích cực cho TI2 Cạnh xuống Cạnh lên

TIM_ARRPreloadConfig(A, B) (Lệnh cho phép hoặc hoặc cấm chế độ tự động nạp lại)	
A: TIM1 … TIMn **B:** ENABLE DISABLE	**A:** timer cần điều khiển Timer 1 … Timer n **B:** Cho phép hoặc cấm Cho phép Cấm
TIM_Cmd (A, B); (Lệnh cho phép hoặc cấm timer)	
A: TIM1 … TIMn **B:** ENABLE DISABLE	**A:** timer cần điều khiển Timer 1 … Timer n **B:** Cho phép hoặc cấm Cho phép Cấm
unsigned short B; **B=TIM_GetCounter(A);** (Lệnh đọc giá trị đếm của timer)	
A: TIM1 … TIMn	**A:** timer cần đọc Timer 1 … Timer n
TIM_SetCounter(A, B); (Lệnh cài đặt giá trị đếm cho timer)	
A: TIM1 … TIMn **B:** 0-65535	**A:** timer cần cài đặt Timer 1 … Timer n **B:** Giá trị cần cài đặt Từ 0 đến 65535
TIM_GetCaptureX(A) ; (Lệnh đọc giá trị input capture **X** của timer)	
A: TIM1 … TIMn **X:** 1 … 4	**A:** timer cần đọc Timer 1 … Timer n **X:** input capture cần đọc Input capture 1 … Input capture 4

TIM_SetCompareX(A, B); (Lệnh cài đặt giá trị cho kênh so sánh **X** của timer)	
A: TIM1 … TIMn **B:** 0-65535 **X:** 1 … 4	**A:** timer cần đọc Timer 1 … Timer n **B:** giá trị cài đặt 0 đến 65535 **X:** kênh so sánh cần cài đặt Kênh 1 … Kênh 4

8.6 CÁC VÍ DỤ LIÊN QUAN ĐẾN TIMER

8.6.1 Ví dụ về định thời sử dụng Timer

Ví dụ 8.1 Viết chương trình đếm giây từ 00-59 sử dụng ngắt timer 2 để cập nhật giá trị giây

a. Tính toán thời gian

- Do sử dụng lệnh SystemInit() nên APB1 = 36Mhz và nguồn xung này trước khi cấp cho Timer 4 được nhân 2 nên thành 72Mhz

⇨ Timer phải đếm 72 triệu số thì được 1 giây

- Mà giá trị đếm tối đa của timer chỉ được 65535 nên ta phải sử dụng bộ chia trước

⇨ Chọn bộ chi trước là 7200

⇨ Vậy lúc này timer phải đếm 72M/7200 = 10 000 xung thì được 1 giây

⇨ Cài đặt cho giá trị tự động nạp lại là 10 000 – 1 = 9 999

b. Chương trình

```
#include<stm32f10x.h>
char giay=0;
void cauhinhTIMER()
{
        TIM_TimeBaseInitTypeDef   TM;
        RCC_APB1PeriphClockCmd (RCC_APB1Periph_TIM4, ENABLE );
        TM.TIM_ClockDivision =0;
        // Không sử dụng bộ chia xung
        TM.TIM_CounterMode = TIM_CounterMode_Up ;
        // Sử dụng chế độ đếm lên
        TM.TIM_Period = 9999;
        // Giá trị tự động nạp lại là 9 999
        TM.TIM_Prescaler = 7199;
        // Sử dụng bộ chia trước 7 200
        TM.TIM_RepetitionCounter =0;
        // Mỗi lần tràn đều cho phép báo ngắt
        TIM_TimeBaseInit (TIM4,&TM);
        TIM_ITConfig(TIM4,TIM_IT_Update,ENABLE );
        TIM_Cmd(TIM4,ENABLE );
}

void cauhinhNVIC()
{
```

```
        NVIC_InitTypeDef    NV;
        NVIC_PriorityGroupConfig ( NVIC_PriorityGroup_0 );
        NV.NVIC_IRQChannel = TIM4_IRQn;
        NV.NVIC_IRQChannelSubPriority =1;
        NV.NVIC_IRQChannelCmd = ENABLE ;
        NVIC_Init (&NV);
}
void TIM4_IRQHandler()
{
        if(TIM_GetITStatus (TIM4,TIM_IT_Update ))
         {
              TIM_ClearITPendingBit (TIM4,TIM_IT_Update );
              giay++;
              if(giay==60)giay=0;
         }
}
int main()
{
        SystemInit ();
        cauhinhTIMER();
        cauhinhNVIC();
        while(1)
        {
              // Hiển thị giá trị "giay" để quan sát
        }
}
```

8.6.2 Ví dụ về đếm xung ngoại sử dụng Timer

Ví dụ 8.2 Viết chương trình đếm xung ngoại từ 1 đến 10 đưa vào chân A0 (timer 2 channel 1)

Chương trình

```
#include<stm32f10x.h>
void cauhinhGPIO() // Câu hình chan A0 nhận xung clock
{
       GPIO_InitTypeDef     GPIO;
       RCC_APB2PeriphClockCmd (RCC_APB2Periph_GPIOA,ENABLE );
       GPIO.GPIO_Pin=GPIO_Pin_0;
       GPIO.GPIO_Mode=GPIO_Mode_IN_FLOATING ;
       GPIO_Init(GPIOA, &GPIO);
}
void cauhinhCounter()
{
       TIM_TimeBaseInitTypeDef   TM;
       RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
       TM.TIM_ClockDivision =0;
       TM.TIM_CounterMode = TIM_CounterMode_Up ;
       TM.TIM_Period = 10;
        //Đếm tối đa là 10
        TM.TIM_Prescaler = 0;
```

```
        TM.TIM_RepetitionCounter =0;
        TIM_TimeBaseInit (TIM2,&TM);
        TIM_TIxExternalClockConfig(TIM2,
                                   TIM_TIxExternalCLK1Source_TI1ED,
                                   TIM_ICPolarity_Falling, 15);
        // Đếm xung ngoại từ timer 2 channel 1
        TIM_Cmd(TIM2,ENABLE);
}
int main()
{   unsigned int dem;
    SystemInit();
    cauhinhGPIO();
    cauhinhCounter();
    while(1)
     {
          dem=TIM_GetCounter(TIM2);
          // Đọc giá trị đếm
          // Hiển thị giá trị đếm để quan sát
     }
 }
```

8.6.3 Ví dụ về sử PWM INPUT và PWM OUTPUT

Ví dụ 8.3 Viết chương trình phát xung vuông ở chân A2 (timer channel 3) với tần số là 2Khz và duty là 70%. Đồng thời viết chương trình cho timer 5 channel 2 (chân A1) đo tín hiệu xung vuông phát ra từ chân A2(nối tắt chân A2 với A1) để kiểm tra.

a. Tính toán thời gian

- Tính toán như **ví dụ 8.1** để tạo được xung có tần số 2Khz ta có

⇨ Bộ chia trước là 36 (giá trị cài đặt là: 36-1=35)

⇨ Giá trị tự động nạp lại là 1000(giá trị cài đặt là : 1000-1= 999)

⇨ Với chu kỳ là 1000 nên duty 70% = 7 00

b. Chương trình

```
#include<stm32f10x.h>
unsigned long Frequency = 0,DutyCycle = 0,IC2Value = 0;
unsigned short  chuky=999,bochia=35,duty=700;
void cauhinhGPIO()
{
          GPIO_InitTypeDef                  PWM;
          RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
          // Cấu hình chân A2 là ngõ ra PWM của timer 2 kênh 3
          PWM.GPIO_Pin = GPIO_Pin_2;
          PWM.GPIO_Mode = GPIO_Mode_AF_PP;
          PWM.GPIO_Speed = GPIO_Speed_50MHz ;
          GPIO_Init(GPIOA,&PWM);
          // Cấu hình chân A1 là ngõ vào PWM của timer 5 kênh 2
          PWM.GPIO_Pin = GPIO_Pin_1;
          PWM.GPIO_Mode = GPIO_Mode_IN_FLOATING ;
              GPIO_Init(GPIOA,&PWM);
}
```

```
void cauhinhPWMOUT()
{
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
    TIM_OCInitTypeDef        TIM_OCInitStructure;
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
    TIM_TimeBaseStructure.TIM_Period = chuky;
    TIM_TimeBaseStructure.TIM_Prescaler = bochia;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
    // Cấu hình timer 2 CH3 phát xung PWM với duty = 7000/10000
    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
    TIM_OCInitStructure.TIM_OutputState =
                                        TIM_OutputState_Enable;
    TIM_OCInitStructure.TIM_Pulse =duty;
    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
    TIM_OC3Init(TIM2, &TIM_OCInitStructure);
    TIM_CtrlPWMOutputs(TIM2, ENABLE);
    TIM_Cmd(TIM2, ENABLE);
}

void cauhinhPWMIN()
{
    TIM_ICInitTypeDef  TIM_ICInitStructure;
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM5, ENABLE);
    // Cấu hình timer 5 base, nếu tràn thì tăng bộ chia trước
    TIM_TimeBaseStructure.TIM_Period = 0xffff;
    TIM_TimeBaseStructure.TIM_Prescaler = 0;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseInit(TIM5, &TIM_TimeBaseStructure);
    // cấu hình timer 5 CH2 làm ngõ PWM IN
    TIM_ICInitStructure.TIM_Channel = TIM_Channel_2;
    TIM_ICInitStructure.TIM_ICPolarity = TIM_ICPolarity_Rising;
    TIM_ICInitStructure.TIM_ICSelection =
                                      TIM_ICSelection_DirectTI;
    TIM_ICInitStructure.TIM_ICPrescaler = TIM_ICPSC_DIV1;
    TIM_ICInitStructure.TIM_ICFilter =15;
    TIM_PWMIConfig(TIM5, &TIM_ICInitStructure);
    TIM_ITConfig(TIM5, TIM_IT_CC2, ENABLE);
    TIM_SelectInputTrigger(TIM5, TIM_TS_TI2FP2);
    TIM_SelectMasterSlaveMode(TIM5,TIM_MasterSlaveMode_Enable);
    TIM_SelectSlaveMode(TIM5, TIM_SlaveMode_Reset);
    // Sử dụng chế độ slave reset - xem lý thuyết PWM INPUT
    TIM_Cmd(TIM5, ENABLE);
}

void NVIC_Configuration(void)
 {
```

```
    NVIC_InitTypeDef NVIC_InitStructure;
    NVIC_InitStructure.NVIC_IRQChannel = TIM5_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
 }
void TIM5_IRQHandler(void)
 {
   if( TIM_GetITStatus(TIM5,TIM_IT_CC2))
     {
        TIM_ClearITPendingBit(TIM5, TIM_IT_CC2);
        IC2Value = TIM_GetCapture2(TIM5);
        if (IC2Value != 0)
           {
            DutyCycle = TIM_GetCapture1(TIM5)*100/IC2Value;
            // tính duty theo đơn vị %
            Frequency = 72000000/ IC2Value;
            // tính tần số
           }
        else
           {
            DutyCycle = 0;
            Frequency = 0;
           }
      }
 }

int main()
{
    SystemInit();
    cauhinhGPIO();
    cauhinhPWMOUT();
    cauhinhPWMIN();
    NVIC_Configuration();
   while(1)
     {
       // hiển thị "DutyCycle" và "Frequency" để quan sát
     }
 }
```

8.6.4 Ví dụ về sử dụng trigger out của timer để kích hoạt ADC chuyển đổi

Ví dụ 8.4 Viết chương trình dùng timer kích hoạt ADC1 chuyển đổi tín hiệu tương tự đi vào kênh số 15 cứ mỗi 50ms một lần. Sau mỗi giây sẽ tính trung bình kết quả của 20 lần chuyển đổi để quan sát.

a. Tính toán thời gian

- Tra lệnh "ADC_ExternalTrigConv" trong **bảng 7.3** ta chọn nguồn xung kích hoạt ngoại cho ADC chuyển đổi là Timer 3 TRGO.
- Tính toán như **ví dụ 8.1** để định thời được 50 ms ta có:

⇨ Bộ chia trước là 7200(giá trị cài đặt là : 7200-1= 7199)
⇨ Giá trị tự động nạp lại là 500(giá trị cài đặt là : 500-1= 499)

b. Chương trình

```
#include<stm32f10x.h>
char n;
unsigned short kq[20];
unsigned long kqt=0;
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          CHADC;
     CHADC.ADC_Mode = ADC_Mode_Independent ;
     CHADC.ADC_ScanConvMode = DISABLE;
     CHADC.ADC_ContinuousConvMode = DISABLE ;
     CHADC.ADC_ExternalTrigConv =ADC_ExternalTrigConv_T3_TRGO ;
     CHADC.ADC_DataAlign = ADC_DataAlign_Right ;
     CHADC.ADC_NbrOfChannel = 1;
     ADC_Init(ADC1,&CHADC);
     ADC_RegularChannelConfig(ADC1,ADC_Channel_15,1,
                                   ADC_SampleTime_13Cycles5 );
     ADC_DMACmd(ADC1 ,ENABLE );
     ADC_Cmd(ADC1,ENABLE);
     ADC_ResetCalibration(ADC1);
     while(ADC_GetResetCalibrationStatus(ADC1));
     ADC_StartCalibration(ADC1);
     while(ADC_GetCalibrationStatus (ADC1));
     ADC_ExternalTrigConvCmd (ADC1,ENABLE );
}
void cauhinhT3()
{    TIM_TimeBaseInitTypeDef   TIM_TimeBaseStructure;
     RCC_APB1PeriphClockCmd (RCC_APB1Periph_TIM3 ,ENABLE );
     TIM_TimeBaseStructure.TIM_Period = 499;
     TIM_TimeBaseStructure.TIM_Prescaler = 7199;
     TIM_TimeBaseStructure.TIM_ClockDivision = 0;
     TIM_TimeBaseStructure.TIM_CounterMode =TIM_CounterMode_Up;
     TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
     TIM_SelectOutputTrigger (TIM3,TIM_TRGOSource_Update);
     TIM_Cmd (TIM3,ENABLE );
}
void cauhinhDMA()
{
   DMA_InitTypeDef          DM;
```

```
    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1,ENABLE );
    DMA_DeInit(DMA1_Channel1);
    DM.DMA_PeripheralBaseAddr = (unsigned long)&ADC1->DR;
    DM.DMA_MemoryBaseAddr = (unsigned long)&kq;
    DM.DMA_DIR= DMA_DIR_PeripheralSRC ;
    DM.DMA_BufferSize = 20;
    DM.DMA_PeripheralInc= DMA_PeripheralInc_Disable;
    DM.DMA_MemoryInc = DMA_MemoryInc_Enable;
    DM.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;
    DM.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord ;
    DM.DMA_Priority = DMA_Priority_High ;
    DM.DMA_Mode= DMA_Mode_Circular ;
    DM.DMA_M2M = DMA_M2M_Disable;
    DMA_Init(DMA1_Channel1,&DM);
    DMA_ITConfig(DMA1_Channel1,DMA_IT_TC,ENABLE );
    DMA_Cmd(DMA1_Channel1,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 = DMA1_Channel1_IRQn;
    NV.NVIC_IRQChannelSubPriority = 1;
    NV.NVIC_IRQChannelCmd = ENABLE ;
    NVIC_Init(&NV);
}
// Được 20 lần đo thì DMA xảy ra ngắt lúc này tính trung bình
void DMA1_Channel1_IRQHandler()
{
    if(DMA_GetITStatus(DMA_IT_TC ) ==1)
      {
         DMA_ClearITPendingBit(DMA_IT_TC );
         kqt=0;
         for(n=0;n<20;n++)kqt = kqt+ kq[n];
         kqt /=20;
         // Hiển thị biến "kqt" ra để quan sát
       }
}
int main()
{
          SystemInit();
          cauhinhGPIO();
          cauhinhADC();
          cauhinhDMA();
          cauhinhNVIC();
```

```
        cauhinhT3();
        while(1);
}
```

8.6.5 Ví dụ về giao tiếp encoder

Ví dụ 8.5 Viết chương trình dùng timer 3 đếm xung từ encoder đưa về qua kênh 1 và 2

Chương trình

```
#include<stm32f10x.h>
void cauhinhGPIO()
{   GPIO_InitTypeDef                ECD;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    ECD.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7;
    ECD.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA,&ECD);
}
void cauhinhEncorder()
{
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
    TIM_TimeBaseStructure.TIM_Prescaler = 0;
    TIM_TimeBaseStructure.TIM_Period = 65535;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
    TIM_EncoderInterfaceConfig(TIM3, TIM_EncoderMode_TI12,
                  TIM_ICPolarity_Rising, TIM_ICPolarity_Rising);
    //Cấu hình encoder giao tiếp 2 kênh tích cực cạnh xuống
    TIM_Cmd(TIM3, ENABLE);
}
int main()
{   unsigned short kq;
    SystemInit();
    cauhinhGPIO();
    cauhinhEncorder();
    TIM_SetCounter(TIM3,0);
   while(1)
     {
        kq= TIM_GetCounter (TIM3);
        // Hiển thị biến "kq" ra để quan sát
     }
 }
```