فصل هفتم : Timers

7

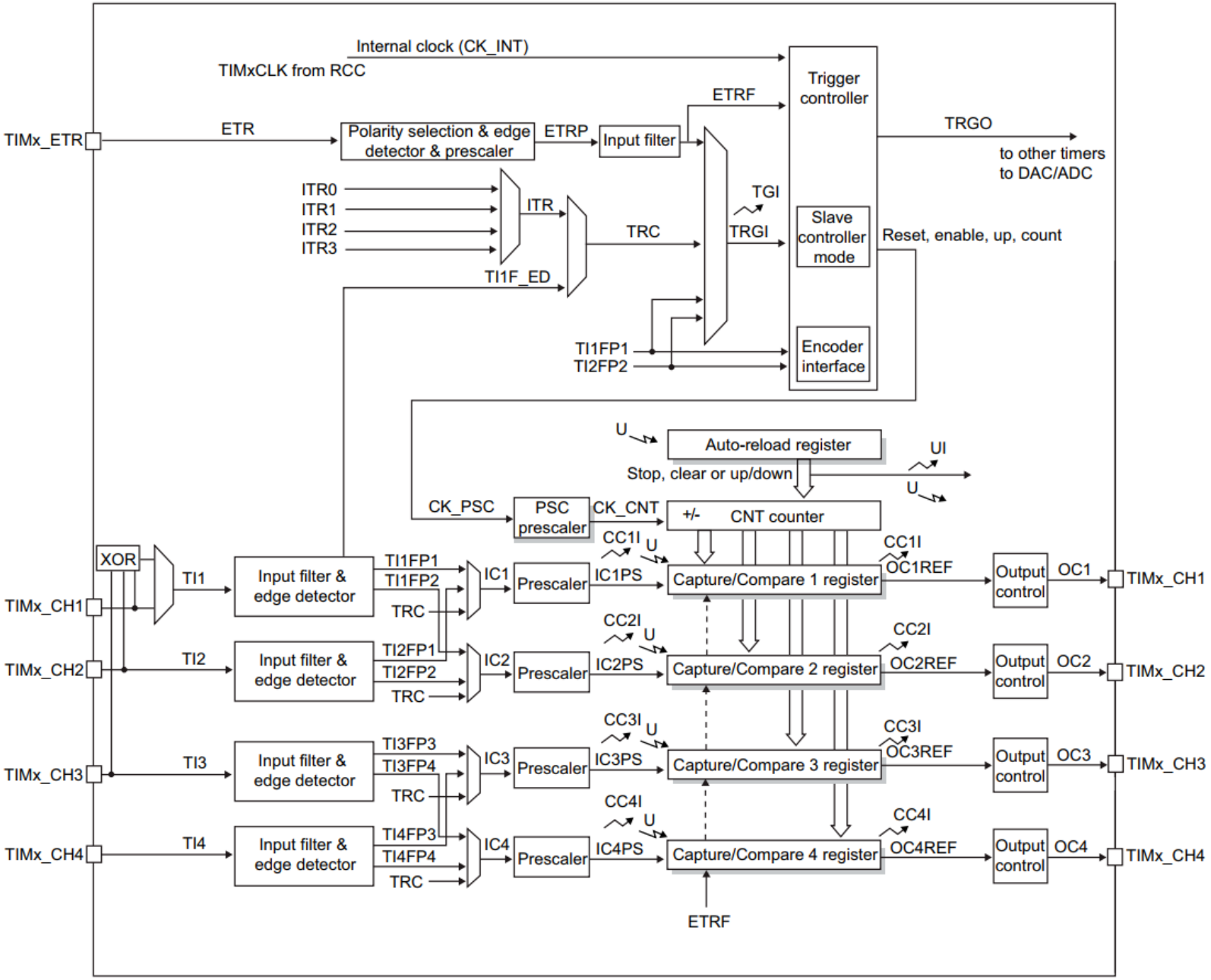
Timer Peripheral

- یکی از پرکاربردترین واحدهای موجود در هر میکروکنترولر، واحد تایمر یا شمارنده می‌باشد.
- در حالت ساده این واحد می‌تواند یک شمارنده باشد، که به ازای شرایط ورودی یا خروجی، می‌تواند تغییرات پارامتری را شمارش نماید. در صورتی که نسبت زمانی این شمارش‌ها نیز مشخص باشد، می‌توان زمان را نیز محاسبه نمود (تایمتر).
- در پردازنده‌های اولیه، قابلیت‌های واحد تایمر بسیار ابتدای و در حد شمارش و محاسبه زمان بوده است که بخش عمده پردازشی آن بر عهده کاربر بود (بایستی نرم‌افزاری مقادیر محاسبه می‌شدند).
- اما در پردازنده‌های امروزی (پیشرفته‌تر) قابلیت‌های سخت‌افزاری فراوانی به آن‌ها اضافه شده است تا بار محاسباتی پردازشگر کاهش یافته و بتواند سایر وظایف خود را مدیریت نماید.
- در ادامه با ویژگی‌های واحد تایمر در پردازنده‌های STM32F10x آشنا می‌شویم.

ویژگی‌های تایمر در STM32F1x

- شمارنده 16 بیتی با قابلیت شمارش رو به بالا، پایین و بالا/پایین (توانایی از سرگیری شمارش)
- قابلیت کاهش Clock، واحد شمارنده. (مقسم clock با ضریب 16 بیتی (1, 65536)).
- هر تایمر دارای 4 کانال مستقل می‌باشد :
  - Input capture
  - Output Compare
  - PWM generation
  - One-pulse mode Output
- مدار سنکرون‌ساز کنترل تایمر با سیگنال خارجی و یا اتصال چند تایمر به یکدیگر
- قابلیت تولید وقفه یا درخواست DMA در شرایط مختلف
- پشتیبانی از انکودرهای افزایشی و مدارات سنسور هال به منظور مکان‌سنجی
- قابلیت تحریک تایمر با Clock خارجی

Block diagram



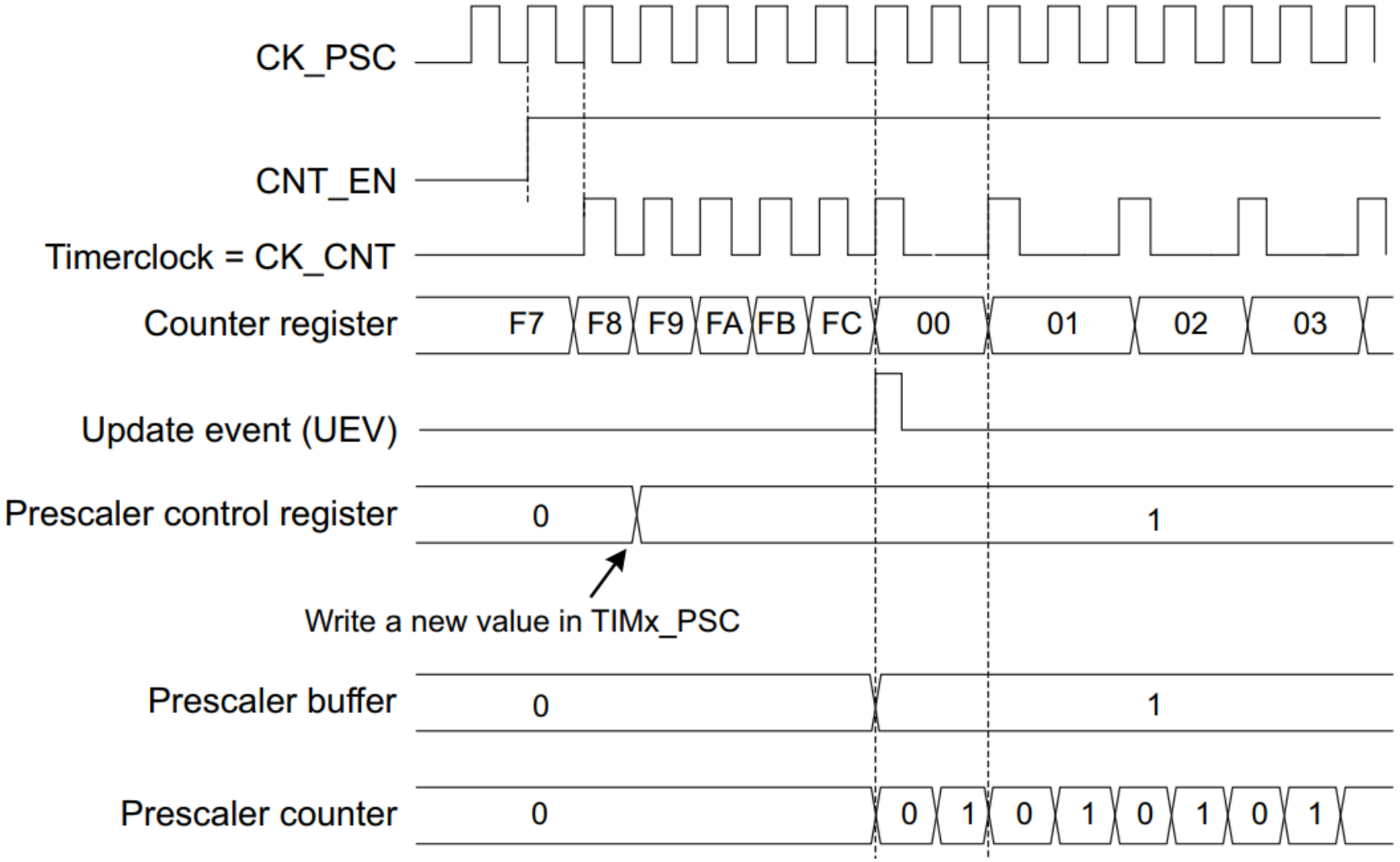
STM32f10x main registers

- در ادامه و به منظور تحلیل بهتر نمودارها، بایستی با این 3 رجیستر اصلی آشنا شویم :

  - Counter Register (TIMx_CNT) : این رجیستر مقدار شمارش شده توسط تایمر در هر لحظه را نشان می‌دهد.
  - Prescaler Register (TIMx_PSC) : این رجیستر مقدار مقسم Clock برای تایمر را نشان می‌دهد
  - Auto-Reload Register (TIMx_ARR) : این رجیستر حد شمارش شمارنده را مشخص می‌کند و پس از آن شمارنده مجدد مقدار اولیه خود را بدست می‌آورد.

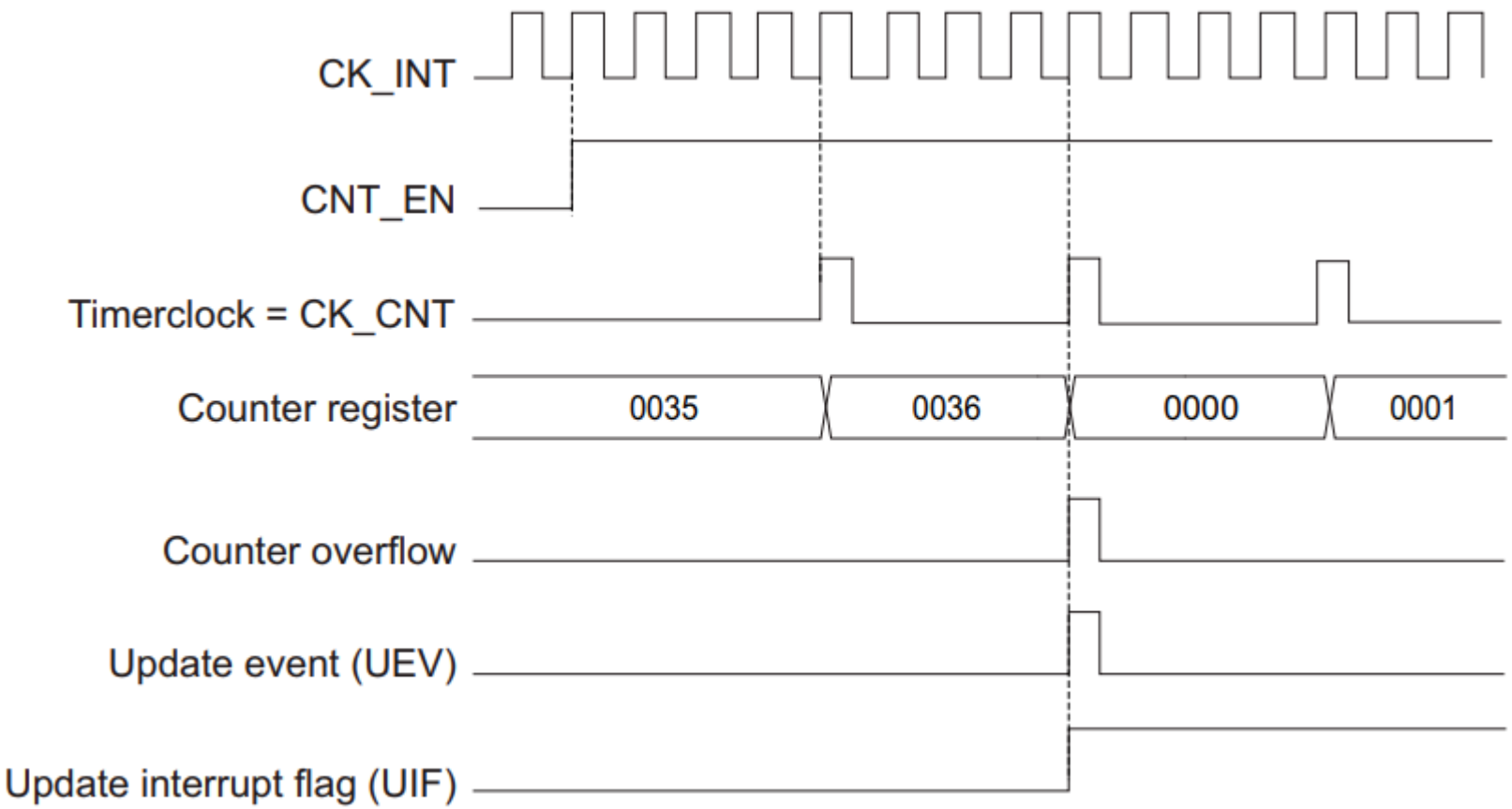
- سایر رجیسترها باتوجه به کاربرد آن‌ها و بر روی نمودار توضیح داده خواهند شد.

Prescaler


CK_PSC
CNT_EN
Timerclock = CK_CNT
Counter register
F7
F8
F9
FA
FB
FC
00
01
02
03
Update event (UEV)
Prescaler control register
0
1
Write a new value in TIMx_PSC
Prescaler buffer
0
1
Prescaler counter
0
0
1
0
1
0
1
0
1

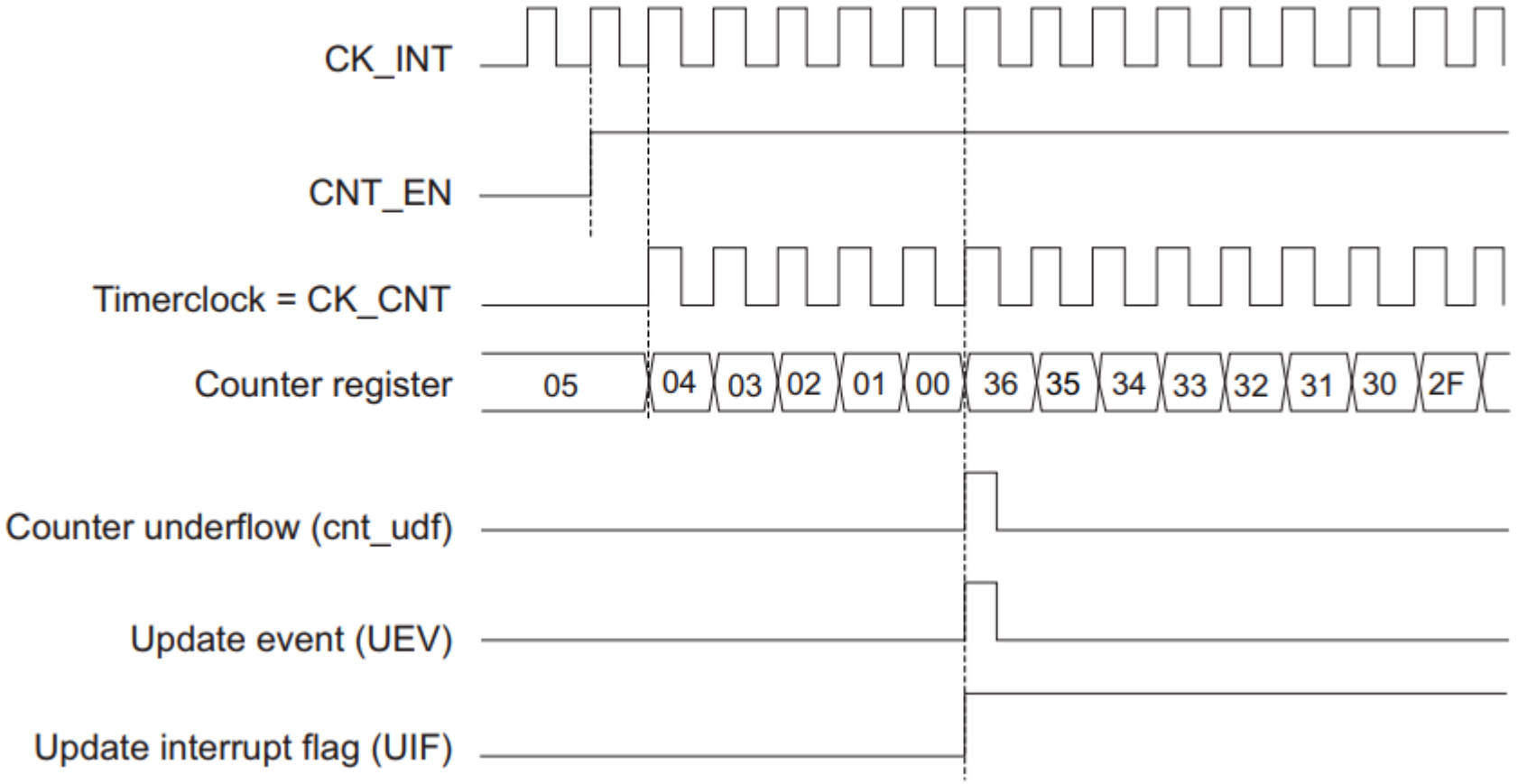
Upcounting

- Prescaler = 3
- پس از هر بار شمارش تا انتها سیگنال overflow ایجاد می‌شود.

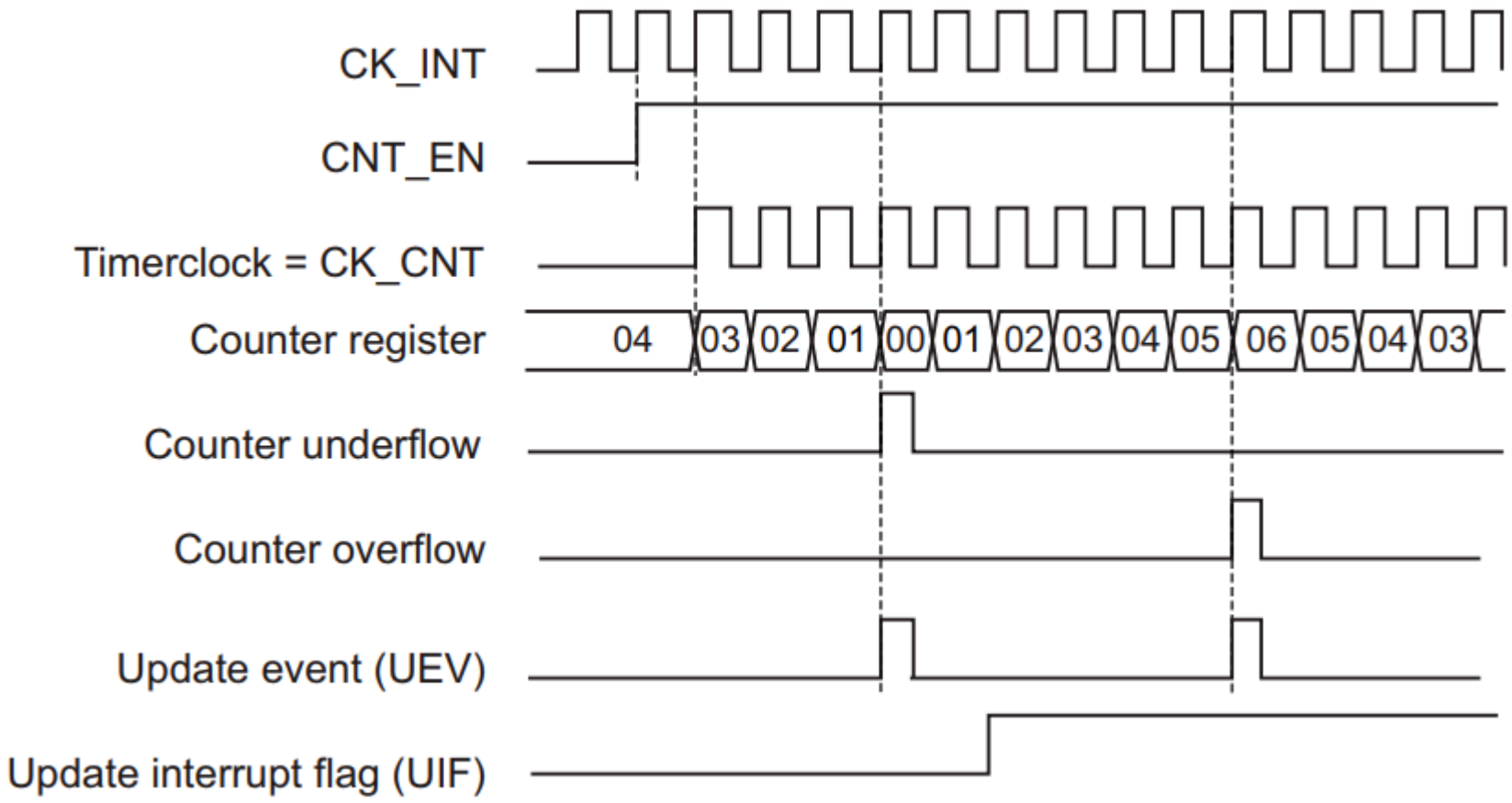


Downcounting

- Prescaler = 0
- پس از هر بار شمارش و رسیدن به عدد 0، سیگنال underflow ایجاد می‌شود.

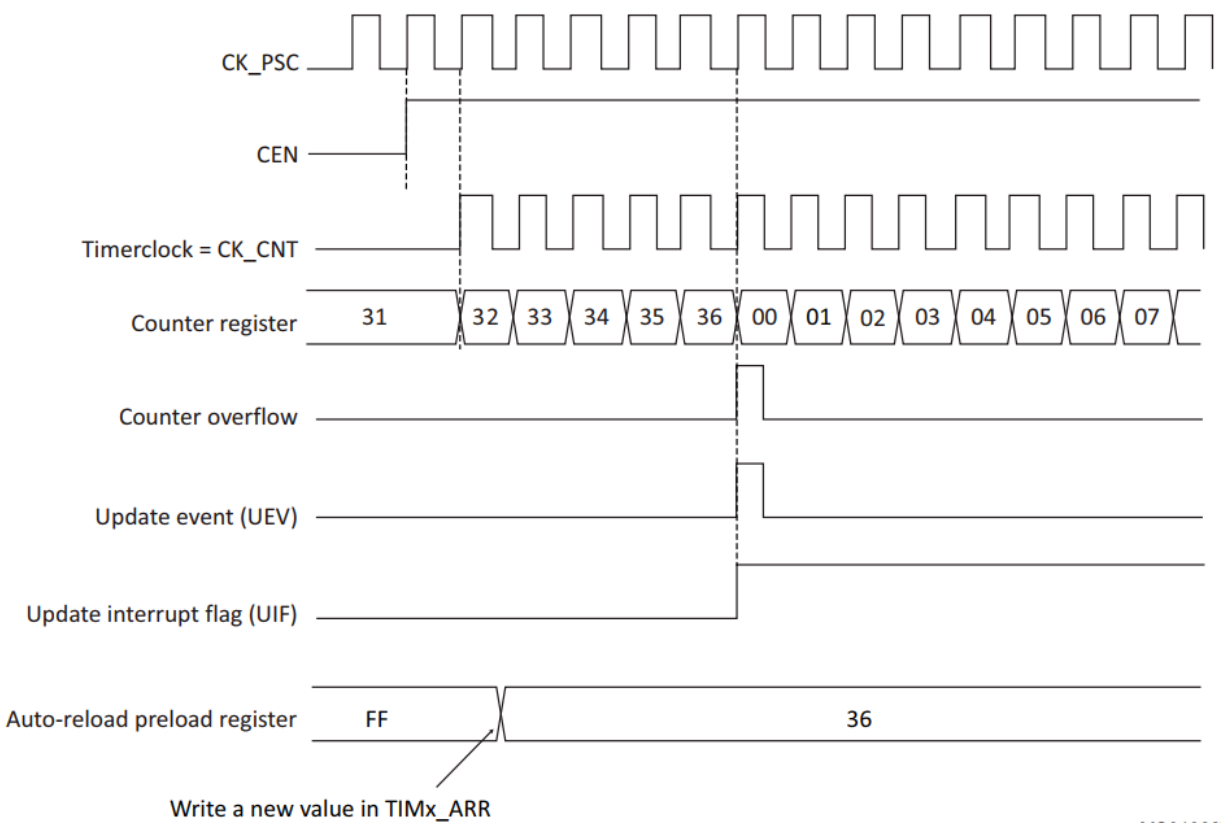


Center aligned mode (up/down counting)

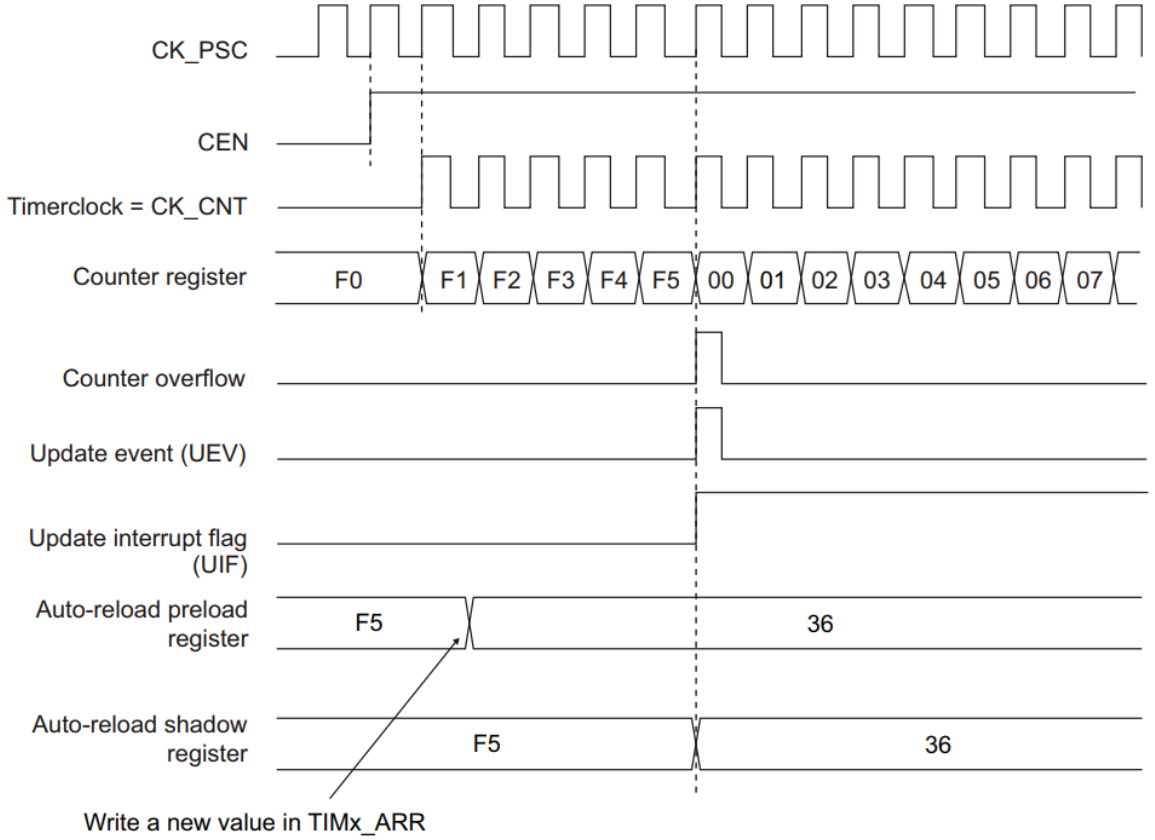


- Prescaler = 0
- ARR = 6
- در هر سیکل با شروع 0 هنگامی که به ARR - 1 می‌رسد، سیگنال Overflow رخ می‌دهد.
- در هر سیکل با شروع از ARR و رسیدن به 1، سیگنال Underflow رخ می‌دهد و مجدد از 0 شروع می‌کند.
- به ازای هر Overflow و Underflow، یک UEV رخ می‌دهد.

Auto Reload Preload Enable (ARPE)



ARPE = 0 → not buffered

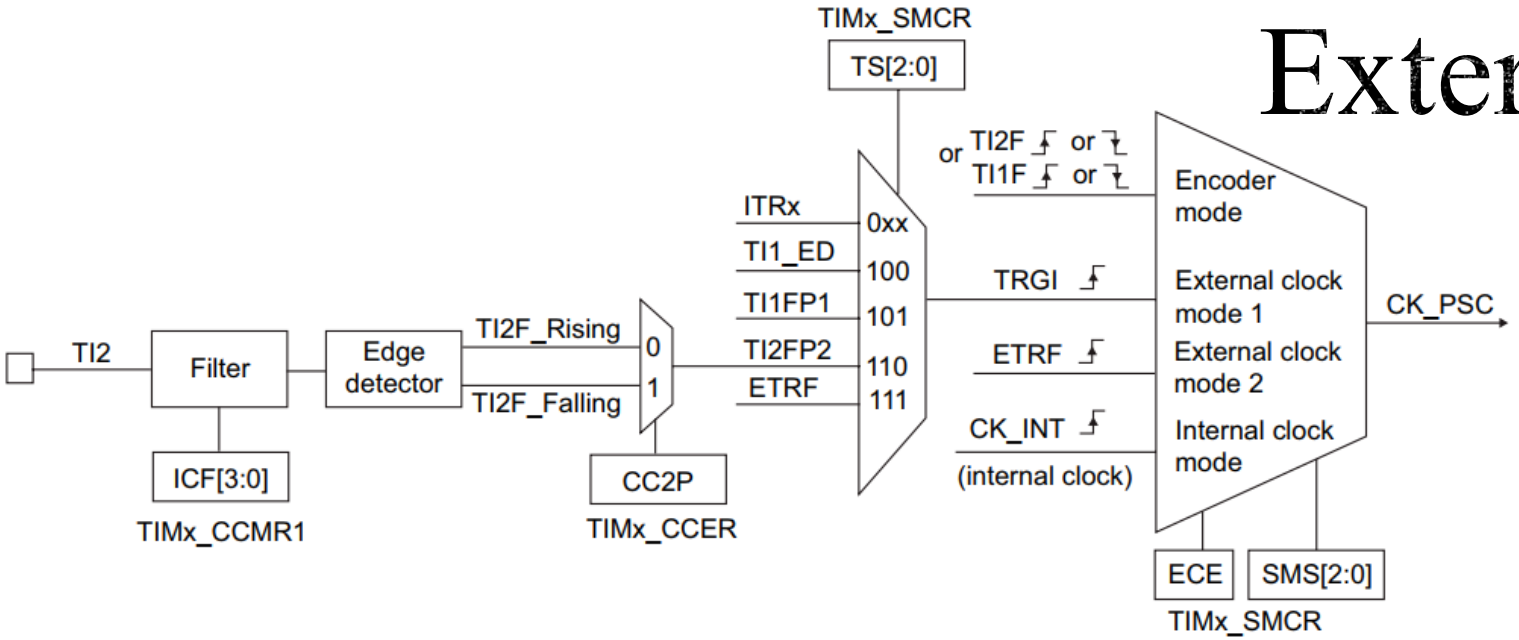


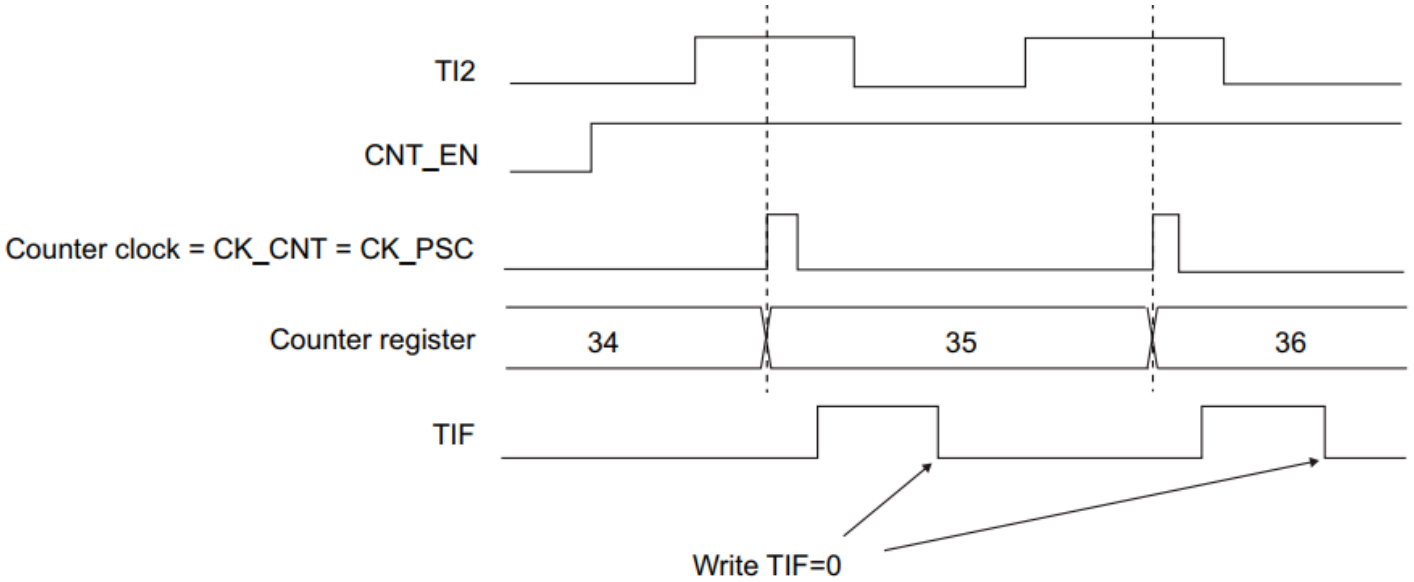
ARPE = 1 → buffered

انتخاب Clock

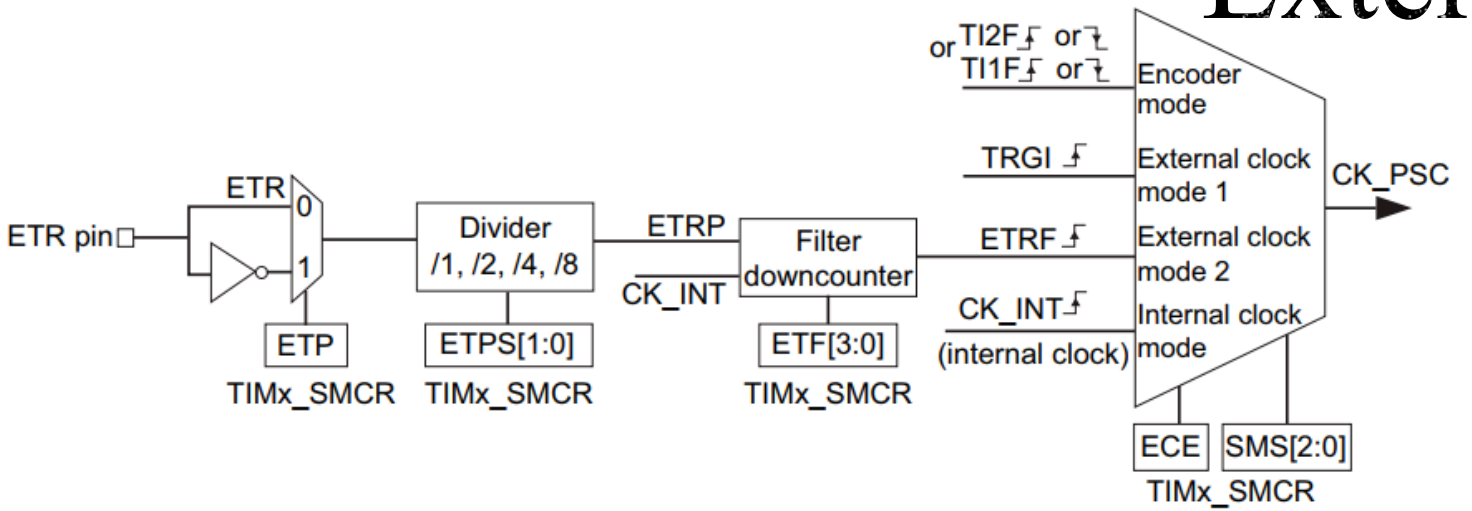
- منابع زیر به عنوان Clock واحد Timer می‌توانند انتخاب شوند
  - Internal clock (CK_INT)
  - External clock mode 1 : external input pin (Tix)
  - External clock mode 2 : external trigger input (ETR)
  - Internal trigger inputs (ITRx) : using one timer as prescaler for another

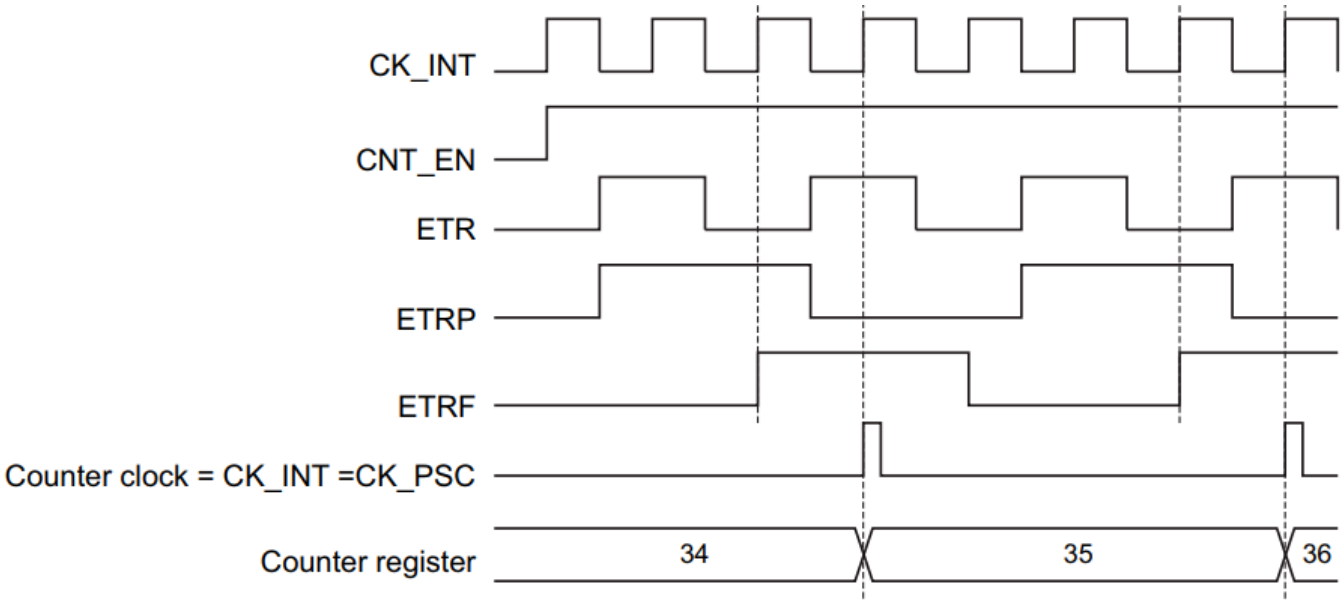
External clock mode 1





External clock mode 2





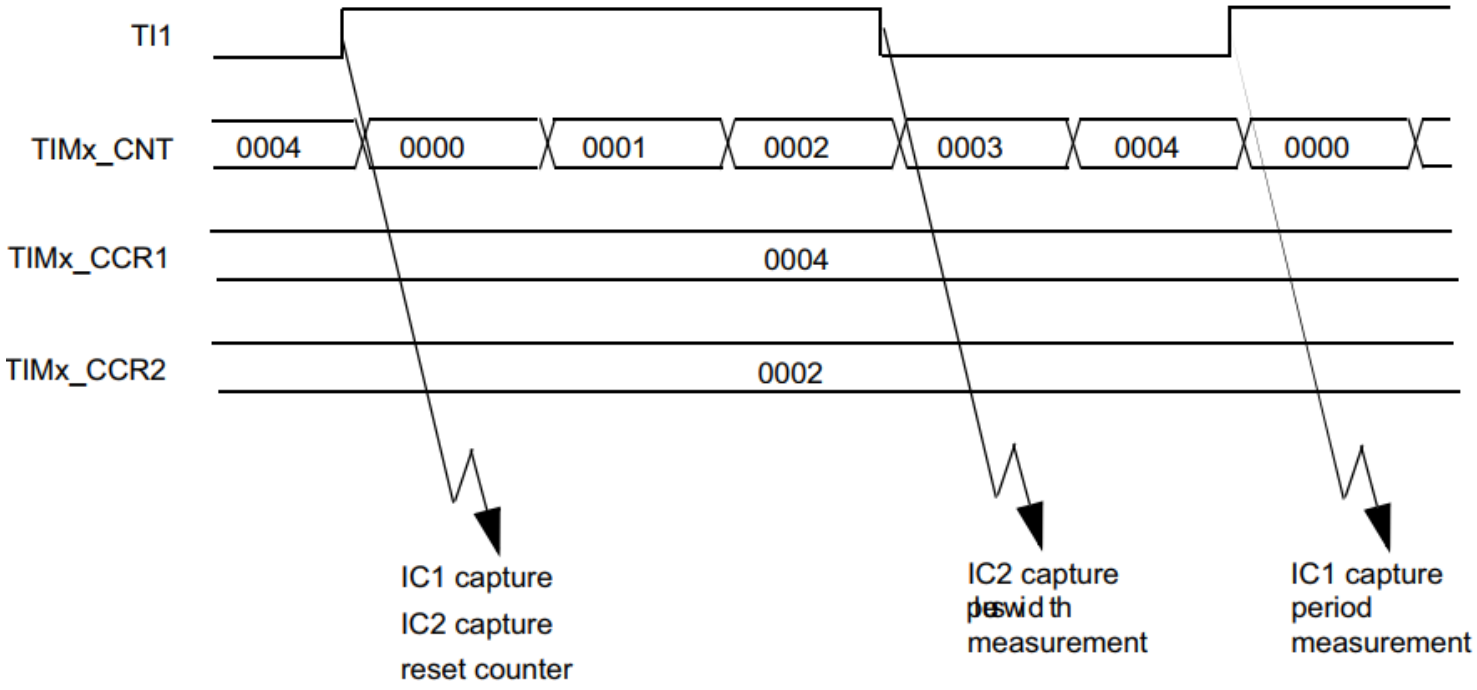
Capture input

- در این ویژگی، هنگامی که ورودی تغییر وضعیت دهد، مقدار شمارنده در رجیستر Capture/Compare Register (TIMx_CCR) ثبت می‌گردد. همچنین یک بیت CCxIF را 1 می‌کند و می‌تواند وقفه یا DMA را فعال نماید.
- در صورتی که تغییر وضعیت بعدی رخ دهد و تغییر وضعیت پیشین توسط کاربر خوانده نشده باشد، سیگنال CCxOF (over-capture) 1 می‌شود تا کاربر مطلع شود.
- مثال : فرض کنید قصد دارید زمان تغییر وضعیت یک سیگنال را محاسبه نمایید، با اتصال آن به پایه TIx و فعال‌سازی تایمر در حالت Capture input، می‌توانید مقدار آن را بدون آن‌که پردازنده درگیر شود بدست آورید.

- سوال : اگر از این قابلیت استفاده نکنیم، چه راه‌حل هایی وجود دارد؟!

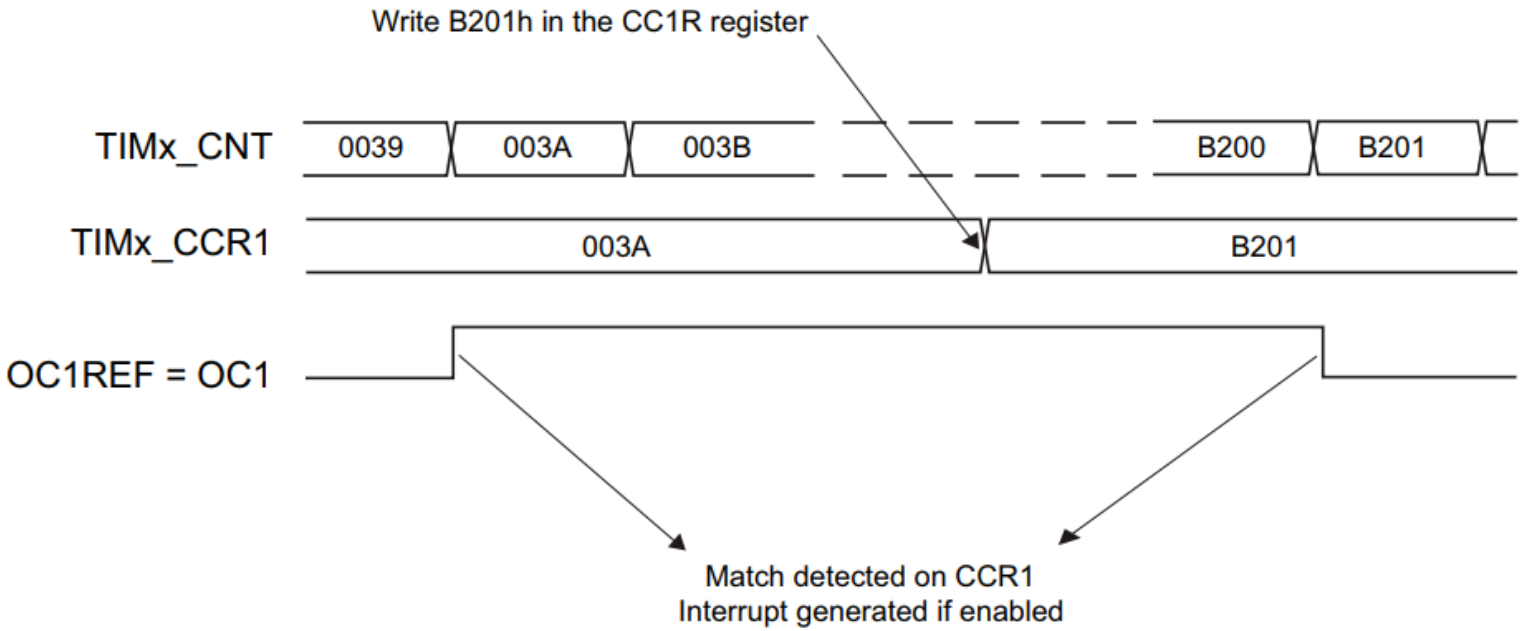
PWM input mode

- این قابلیت به منظور محاسبه دوره و Duty Cycle یک سیگنال PWM استفاده می‌شود.
- در این حالت دو ICx را به یک TIx متصل می‌کنیم.
- دو ICx را با لبه‌های متفاوت به منظور تحریک، تنظیم می‌کنیم.



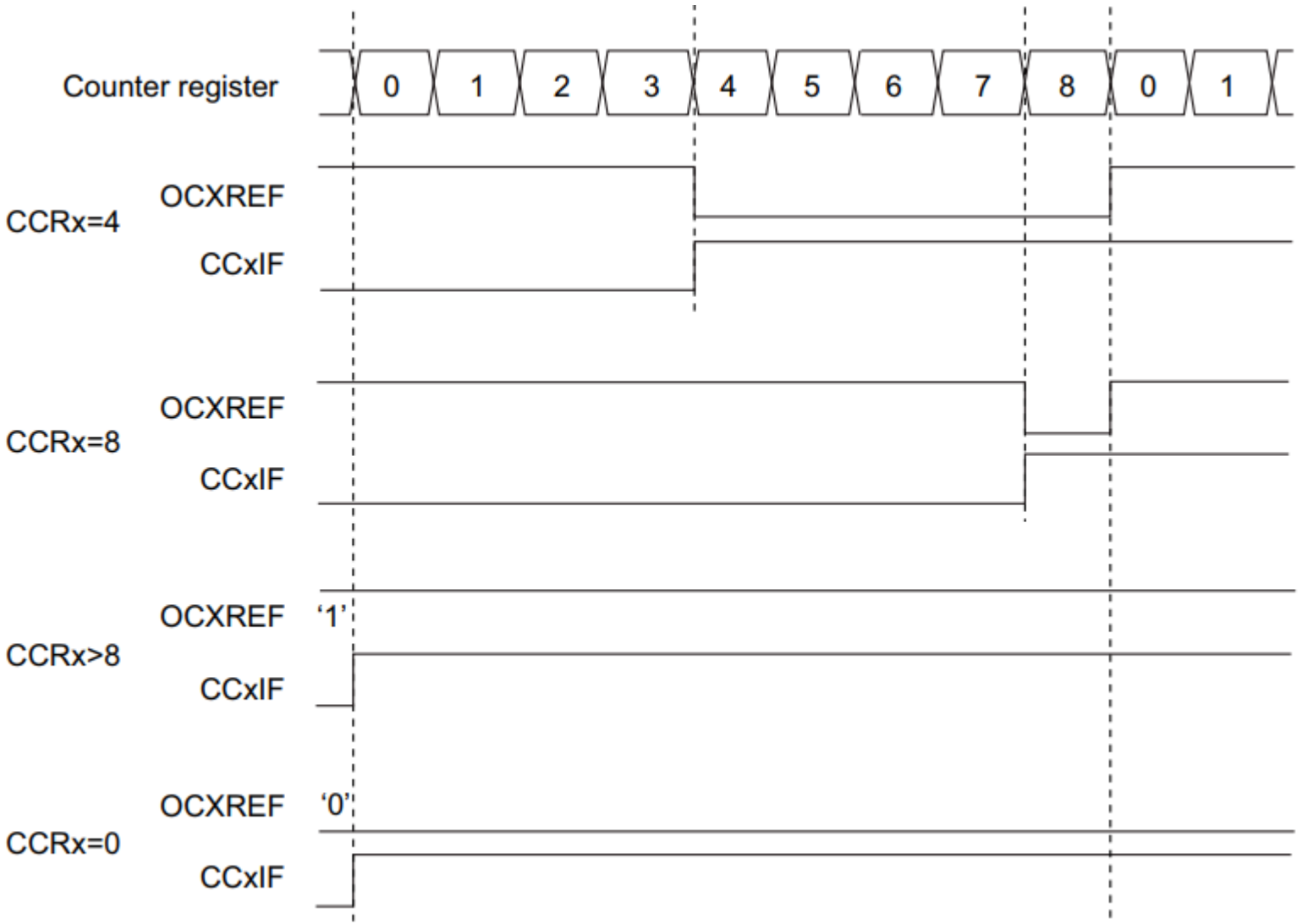
Output compare mode

- این قابلیت به منظور ارسال فرمان پس از گذشت بازه‌ای از زمان و یا ساخت یک شکل موج خاص استفاده می‌شود.
- در این حالت امکان تعریف سناریو در هنگام برابر شدن دو عدد TIMx_CNT و TIMx_CCR1 وجود دارد
  - وضعیت سیگنال خروجی OC1 (صفر یا یک)
  - تغییر وضعیت سیگنال خروجی (Toggle)
  - عدم تغییر سیگنال خروجی!؟
- قابلیت تولید وقفه و DMA
- قابلیت تنظیم عدد CCR به صورت لحظه‌ای



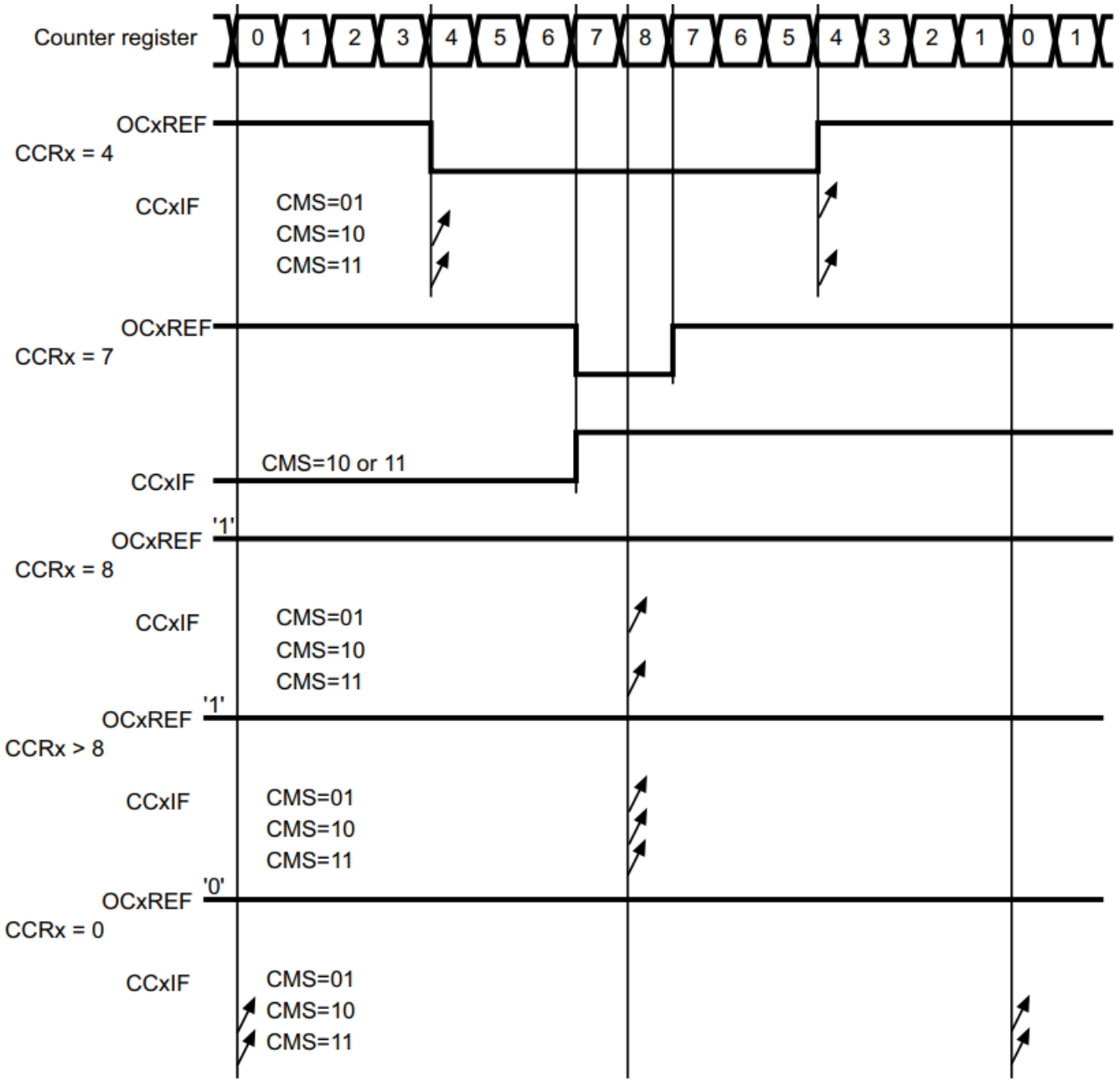
PWM mode

- PWM edge aligned mode

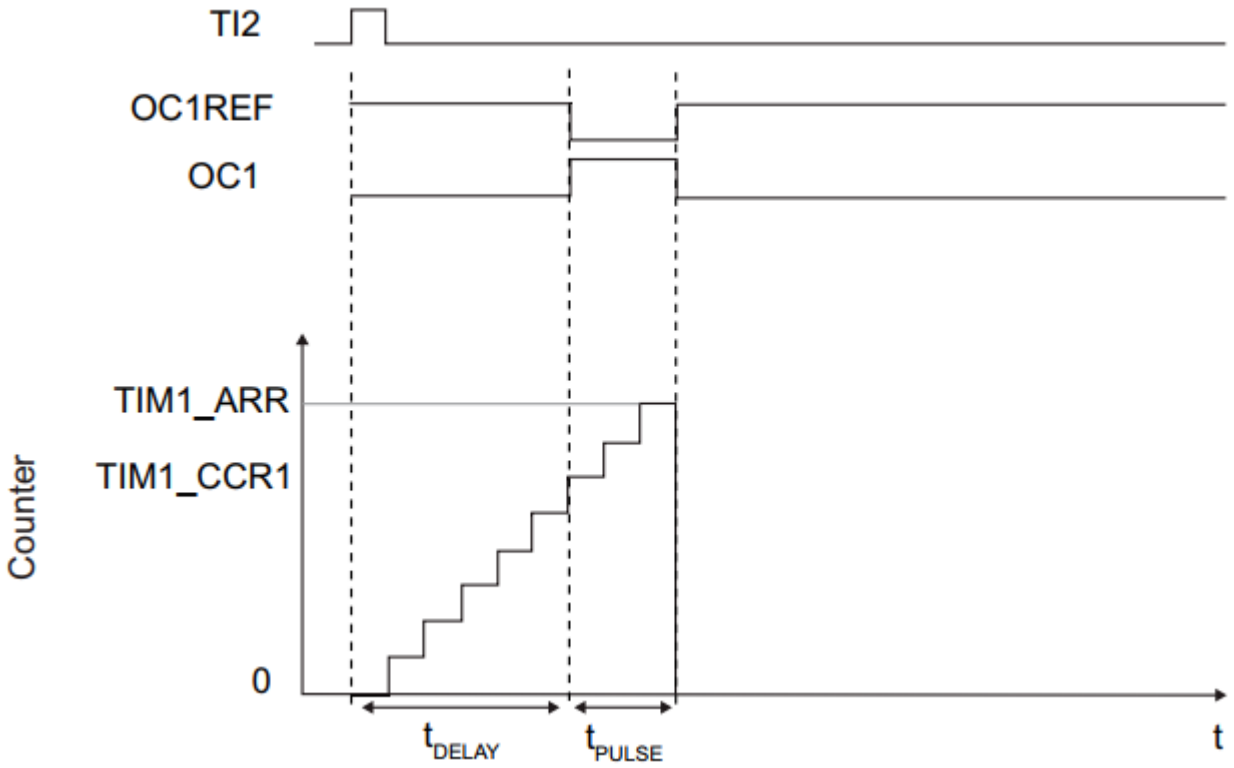


PWM mode

- PWM center aligned mode



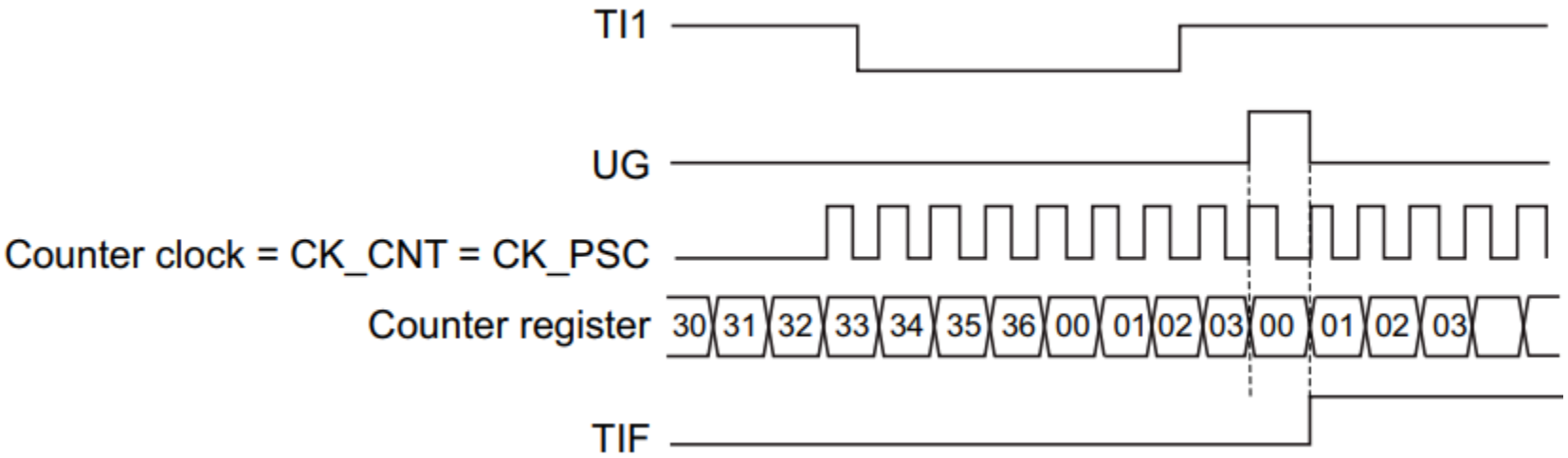
One-pulse mode



- مشابه با PWM می‌باشد با این تفاوت که پس از یک سیکل، تایمر متوقف می‌شود و منتظر سیگنال تحریک بعدی می‌باشد.
- در اکثر اوقات سیگنال تحریک از طریق پایه‌های ورودی TIx اعمال می‌گردد.
- مثال : فرض کنید قصد دارید تا پس از هر بار دریافت یک سیگنال، پس از بازه زمانی معینی سیگنال کنترلی ارسال نمایید.

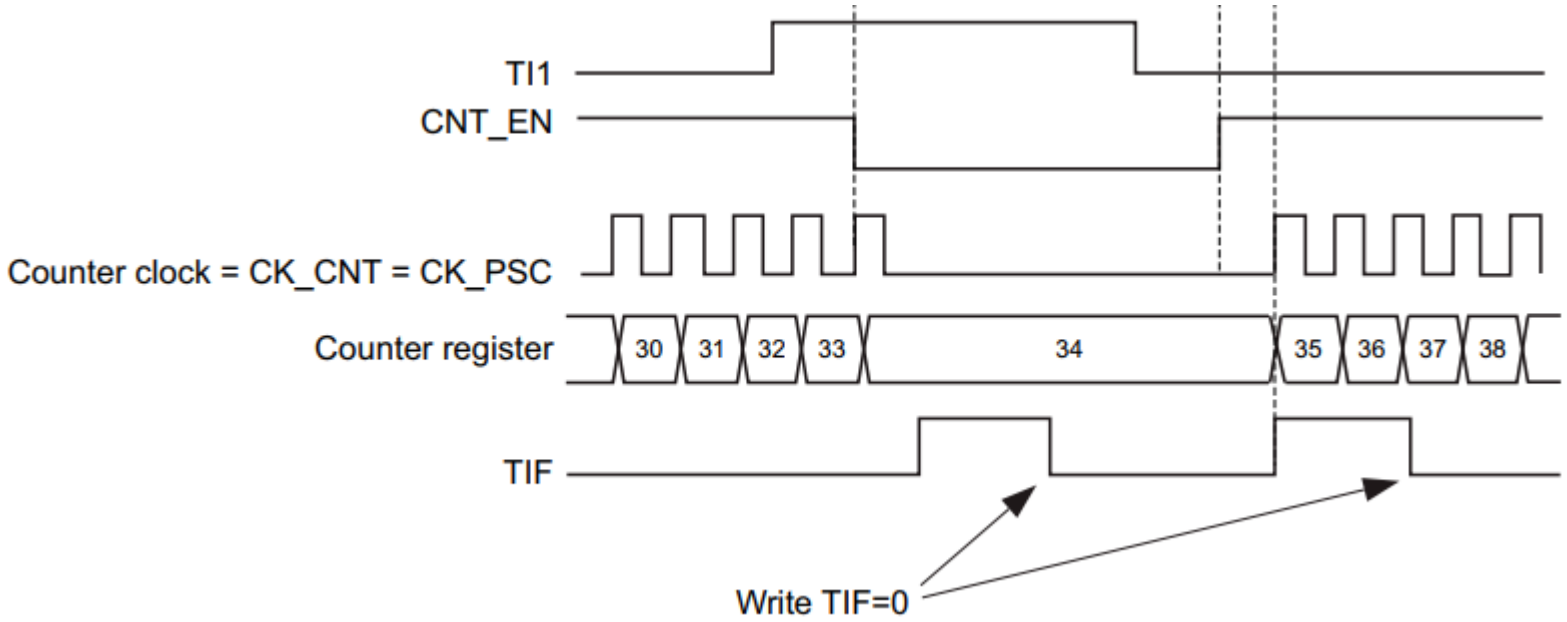
Timers and external trigger synchronization (Slave mode : Reset mode)

- در این حالت تایمر در حال شمارش بر اساس Clock مشخص شده (در اینجا clock داخلی) می‌باشد. این قابلیت وجود دارد که با لبه‌ی بالاروند از یک سیگنال ورودی (TIx)، مقدار آن ریست شود و مجدد شروع به شمارش نماید.
- همچنین قابلیت ارسال وقفه را نیز دارا می‌باشد
- همچنین سیگنال UG (update generation) نیز تولید می‌شود!؟ (وظیفه آن!؟)
- تاخیر بین لبه‌ی بالارونده TI1 و اعمال ریست در مدار ناشی از تاخیر مدار واسط می‌باشد.



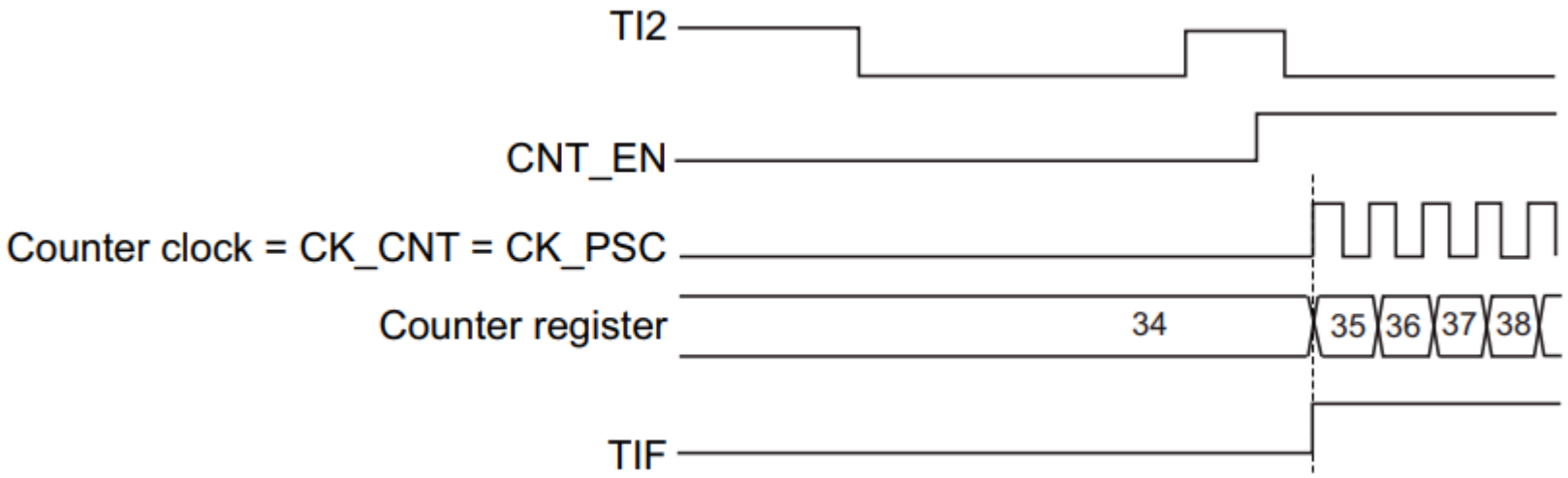
Timers and external trigger synchronization (Slave mode : Gated mode)

- در این حالت تایمر در حال شمارش بر اساس Clock مشخص شده (در اینجا clock داخلی) می‌باشد. این قابلیت وجود دارد که با سطح یک سیگنال ورودی (TIx)، عملیات شمارش متوقف یا از سر گرفته شود.
- همچنین قابلیت ارسال وقفه را نیز دارا می‌باشد
- تاخیر بین تغییر وضعیت TI1 و اعمال توقف یا از سرگیری در مدار ناشی از تاخیر مدار واسط می‌باشد.



Timers and external trigger synchronization (Slave mode : Gated mode)

- در این حالت تایمر هنگامی شروع به شمارش می‌کند که لبه‌ی بالا رونده یک سیگنال تحریک شود. (TI2)
- همچنین قابلیت ارسال وقفه را نیز دارا می‌باشد
- تاخیر بین تغییر وضعیت TI2 و سرگیری در مدار ناشی از تاخیر مدار واسط می‌باشد.



توابع HAL کاربردی در Timers
Base functions

HAL_StatusTypeDef HAL_TIM_Base_Start (TIM_HandleTypeDef * htim)

HAL_StatusTypeDef HAL_TIM_Base_Stop (TIM_HandleTypeDef * htim)

HAL_StatusTypeDef HAL_TIM_Base_Start_IT (TIM_HandleTypeDef * htim)

HAL_StatusTypeDef HAL_TIM_Base_Stop_IT (TIM_HandleTypeDef * htim)

توابع HAL کاربردی در Timers
Output compare

HAL_StatusTypeDef HAL_TIM_OC_Start (TIM_HandleTypeDef * htim, uint32_t Channel)

HAL_StatusTypeDef HAL_TIM_OC_Stop (TIM_HandleTypeDef * htim, uint32_t Channel)

HAL_StatusTypeDef HAL_TIM_OC_Start_IT (TIM_HandleTypeDef * htim, uint32_t Channel)

HAL_StatusTypeDef HAL_TIM_OC_Stop_IT (TIM_HandleTypeDef * htim, uint32_t Channel)

HAL_StatusTypeDef HAL_TIM_PWM_Start (TIM_HandleTypeDef * htim, uint32_t Channel)

HAL_StatusTypeDef HAL_TIM_PWM_Stop (TIM_HandleTypeDef * htim, uint32_t Channel)

توابع HAL کاربردی در Timers
Input compare

HAL_StatusTypeDef HAL_TIM_IC_Start (TIM_HandleTypeDef * htim, uint32_t Channel)

HAL_StatusTypeDef HAL_TIM_IC_Stop (TIM_HandleTypeDef * htim, uint32_t Channel)

HAL_StatusTypeDef HAL_TIM_IC_Start_IT (TIM_HandleTypeDef * htim, uint32_t Channel)

HAL_StatusTypeDef HAL_TIM_IC_Stop_IT (TIM_HandleTypeDef * htim, uint32_t Channel)