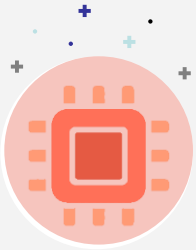
In The Name of God

Faculty of Electrical Engineering

Microprocessor Laboratory

V1 (@2025)



Microprocessor LAB

Lecture 8:

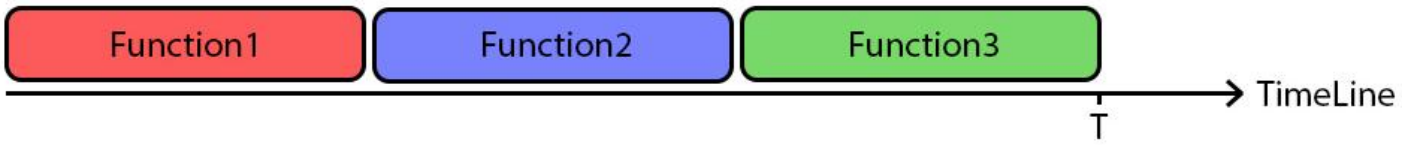
Timer/Counter in STM32 – Part 1

By: M.Razeghizadeh

Start now!

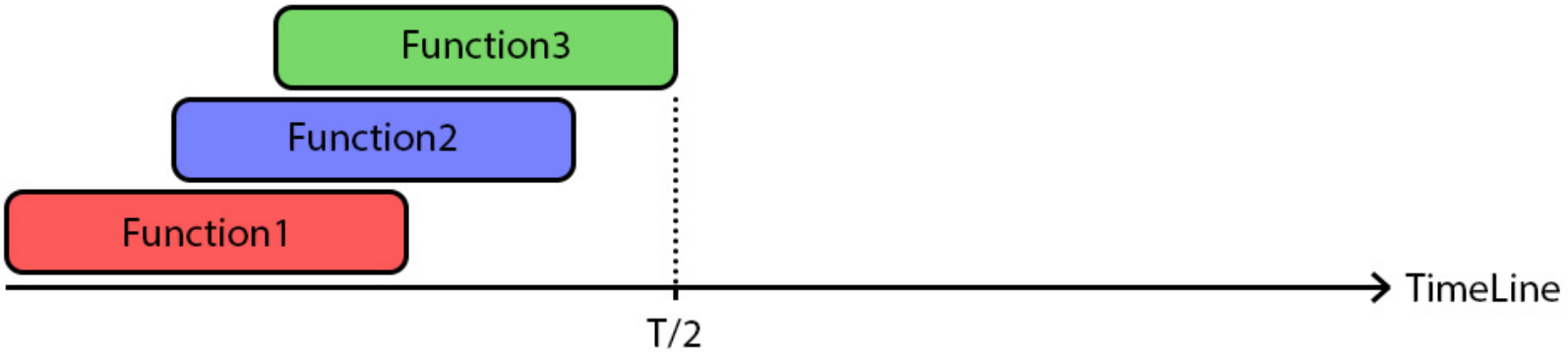
مشکل استفاده از Delay در میکروکنترلرها

- توابع Delay برنامه را وارد یک حلقه تکراری می‌کنند و تا پایان مدت زمان مشخص، هیچ عملی انجام نمی‌شود.
- در برنامه‌های ساده مشکلی ایجاد نمی‌کند. در برنامه‌های پردازشی و حجیم استفاده از Delay غیرکارآمد است.
- نتیجه: انجام چند عمل مختلف زمان زیادی می‌برد و کارایی کاهش می‌یابد.



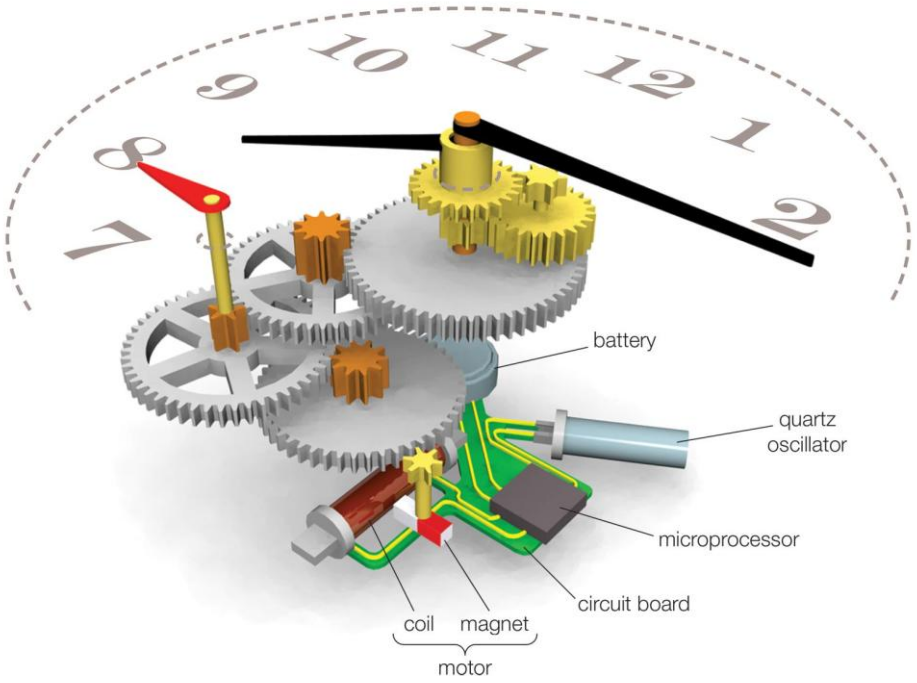
مشکل استفاده از Delay در میکروکنترلرها

- تایمرها ابزاری مستقل از CPU برای سنجش زمان هستند.
- می‌توانند به‌صورت موازی با CPU کار کنند.
- دقت بالاتر نسبت به توابع Delay دارند.
- در برنامه‌های پردازشی و حجیم، تایمرها جایگزین بهتری برای Delay هستند.
- امکان اجرای چند عملیات به‌صورت همزمان وجود دارد.
- نتیجه: زمان انجام عملیات کاهش می‌یابد و کارایی افزایش پیدا می‌کند.

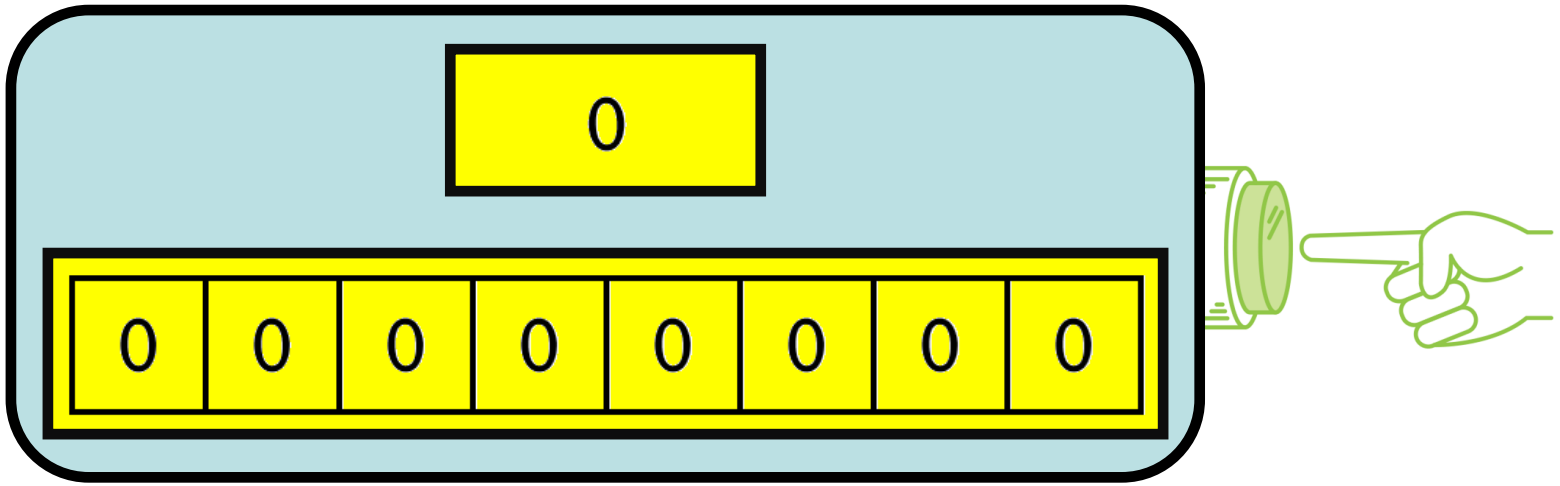


مفهوم و اهمیت واحد زمان‌سنج

- یکی از گسترده‌ترین و پرکاربردترین مباحث در میکروکنترلرها، به ویژه STM32، است.
- وظیفه اصلی Timer سنجش زمان است.
- ساختار Timer در الکترونیک دیجیتال مبتنی بر شمارنده (Counter) است.
- محاسبه زمان از طریق ضرب تعداد شمارش شده در مدت زمان هر شمارش به دست می‌آید.



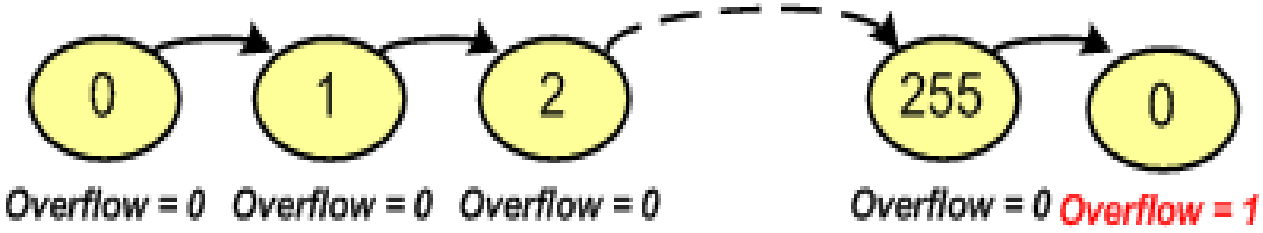
شمارنده (counter)



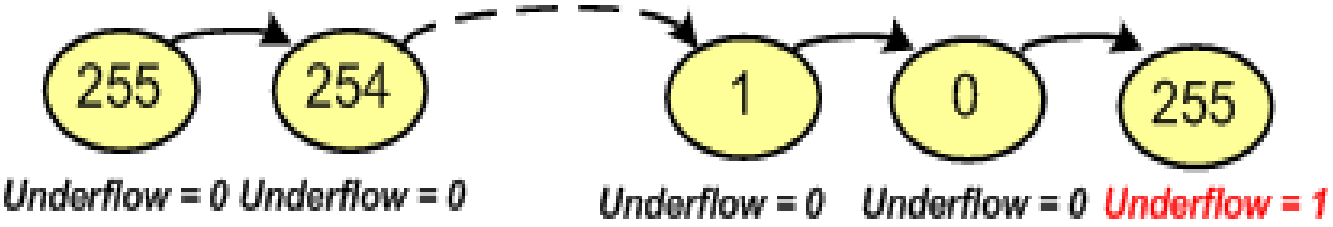
- اساس عملکرد یک Timerدیجیتال بر پایه‌ی Counter (شمارنده) است.
- شمارنده وظیفه‌ی افزایش عدد را برعهده دارد و زمان با استفاده از خروجی شمارنده محاسبه می‌شود.
- اگر مدت‌زمان هر شمارش مشخص باشد: زمان = تعداد شمارش × زمان هر شمارش
- مثال ساده
- یک شمارنده‌ی ۴ بیتی را در نظر بگیرید > شمارش از ۰ تا ۱۵
- منبع کلاک با فرکانس ۱ kHz تغذیه می‌شود.
- هر پالس کلاک برابر است با ۱ ms
- در نتیجه این شمارنده می‌تواند زمان را با پایه‌ی زمانی ۱ ms اندازه‌گیری کند.

شمارنده (counter)

- Up-counter

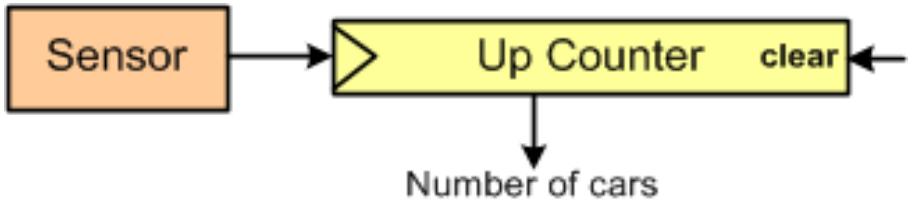


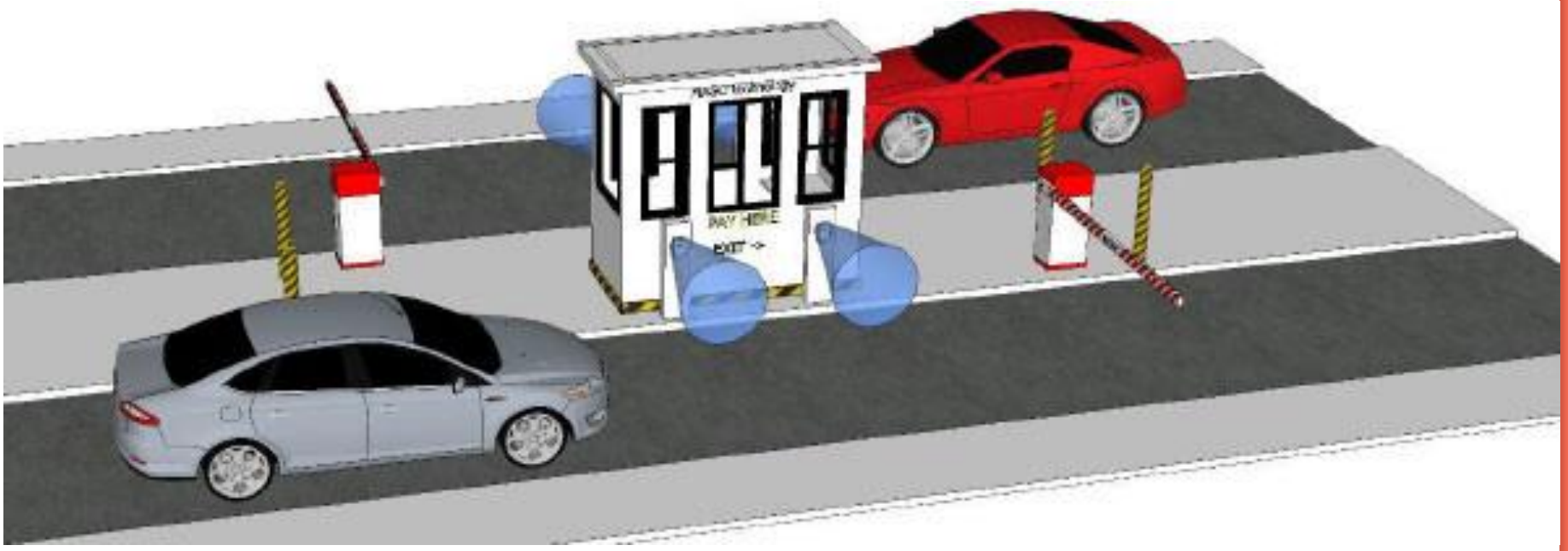
- Down counter



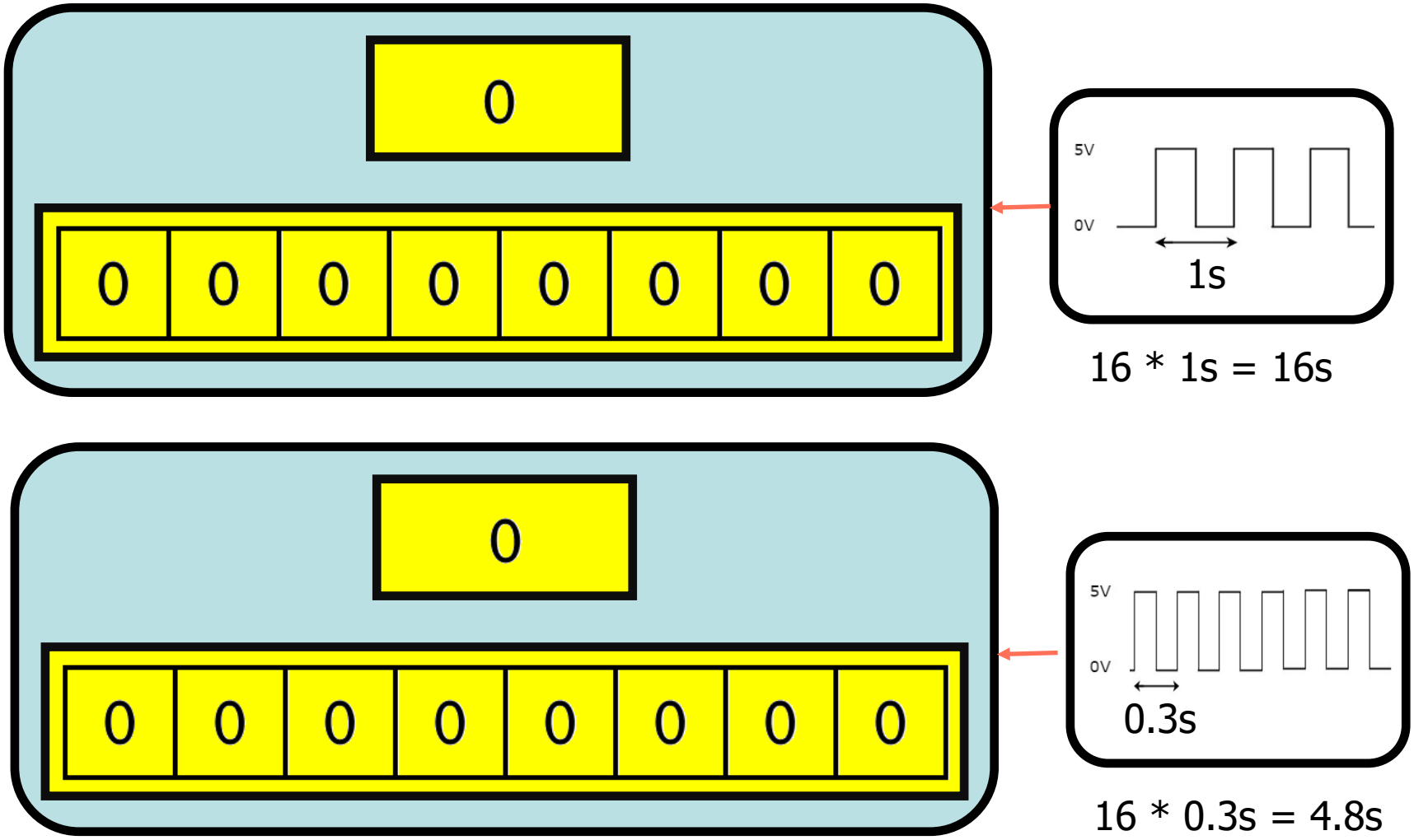
شمارنده (counter)

Event Counter

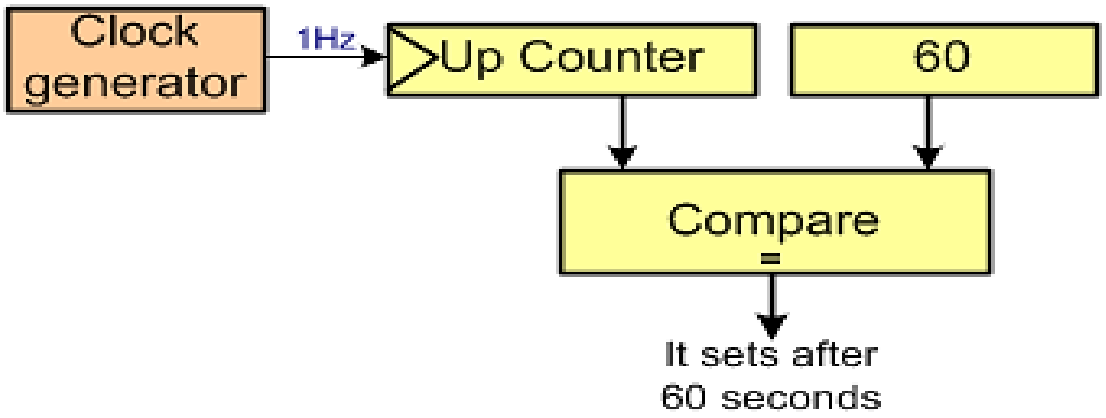




تایمر



تایمر


Clock generator
1Hz
Up Counter
60
Compare
=
It sets after
60 seconds

تایمرها در میکروکنترلرهاي STM32

- هر میکروکنترلر STM32 دارای چندین Timerبا قابلیت‌ها و نوع‌های مختلف است. انواع این تایمرها معمولاً به سه دسته تقسیم می‌شوند:
  - ✓ Advanced-control
  - ✓ General-purpose
  - ✓ Basic
- تعداد و نوع دقیق Timerها بسته به مدل میکروکنترلر متفاوت است.
- تایمرهاي ۱ و ۸ از نـوع Advanced-control هستند.
- تایمرهـاي ۲ تـا ۵ و ۹ تـا ۱٤ از نـوع General-purpose هستند.
- تایمرهاي ٦ و ۷ از نوع Basic هستند.

تایمرها در میکروکنترلرهای STM32

Timer Type		STM32F0	STM32F1	STM32F3	STM32G0	STM32G4	STM32F2	STM32F4	STM32F7	STM32H7	STM32L0	STM32L1	STM32L4	STM32L4+	STM32L5	STM32U5
Advanced		TIM1	TIM1	TIM1*	TIM1	TIM1	TIM1	TIM1	TIM1	TIM1			TIM1	TIM1	TIM1	TIM1
			TIM8*	TIM8*		TIM8	TIM8	TIM8*	TIM8	TIM8			TIM8*	TIM8*	TIM8	TIM8
						TIM20*										
General purpose	16-bit	TIM3	TIM2	TIM3	TIM3	TIM3	TIM3	TIM3*	TIM3	TIM3	TIM2	TIM2	TIM3*	TIM3*	TIM3	TIM3
			TIM3	TIM4		TIM4	TIM4	TIM4*	TIM4	TIM4		TIM3	TIM4*	TIM4*	TIM4	TIM4
			TIM4*									TIM4				
			TIM5*	TIM19*												
	32-bit	TIM2		TIM2	TIM2	TIM2	TIM2	TIM2*	TIM2	TIM2			TIM2	TIM2	TIM2	TIM2
				TIM5*		TIM5	TIM5	TIM5	TIM5	TIM5		TIM5	TIM5*	TIM5*	TIM5	TIM5
Basic		TIM6*	TIM6*	TIM6	TIM6	TIM6	TIM6	TIM6*	TIM6	TIM6	TIM6	TIM6	TIM6	TIM6	TIM6	TIM6
		TIM7*	TIM7*	TIM7	TIM7	TIM7	TIM7		TIM7	TIM7	TIM7	TIM7	TIM7*		TIM7	TIM7
				TIM18*												
1-channel			TIM10*				TIM10	TIM10*	TIM10			TIM10				
			TIM11*				TIM11	TIM11	TIM11			TIM11				
			TIM13*	TIM13*			TIM13		TIM13	TIM13						
		TIM14	TIM14*	TIM14*	TIM14	TIM14	TIM14		TIM14	TIM14						
2-channels			TIM9				TIM9	TIM9	TIM9		TIM21	TIM9				
			TIM12	TIM12*			TIM12		TIM12	TIM12	TIM22*					
2-channel with one complementary output		TIM15*	TIM15*	TIM15	TIM15	TIM15				TIM15			TIM15	TIM15	TIM15	TIM15
1-channel with one complementary output		TIM16	TIM16*	TIM16	TIM16	TIM16				TIM16			TIM16	TIM16	TIM16	TIM16
		TIM17	*TIM17	TIM17	TIM17	TIM17				TIM17			TIM17*		TIM17	TIM17
High-resolution				HRTIM*		HRTIM1*										
Low-power					LPTIM1	LPTIM1		LPTIM1	LPTIM1	LPTIM1	LPTIM1		LPTIM1	LPTIM1	LPTIM1	LPTIM1
					LPTIM2	LPTIM2				LPTIM2			LPTIM2	LPTIM2	LPTIM2	LPTIM2
										LPTIM3					LPTIM3	LPTIM3
										LPTIM4						

تایمرها در میکروکنترلرهای STM32

۱. Basic Timers

- فقط شمارش می‌کنند و توانایی ایجاد وقفه (Interrupt) دارند.
- برای تولید **تایمینگ ساده** یا **تاخیر** استفاده می‌شوند.
- مثال: **TIM6** و **TIM7** در اکثر STM32ها.
- ویژگی‌ها:

  شمارش صعودی یا نزولی.

  تولید **Update Event** برای وقفه یا DMA.

۲. General-purpose Timers

- **انعطاف‌پذیر و برای کاربردهای زمان‌بندی و PWM استفاده می‌شوند.**
- **مثال: TIM2، TIM3، TIM4، TIM5.**
- **ویژگی‌ها:**

  قابلیت شمارش صعودی/نزولی.

  تولید PWM (Pulse Width Modulation).

  شمارش Encoder برای اندازه‌گیری موقعیت و سرعت موتور.

  امکان اتصال به DMA و وقفه‌ها.

۳. Advanced-control Timers

برای **کنترل موتورهای** DC/BLDC و کاربردهای قدرتی استفاده می‌شوند.

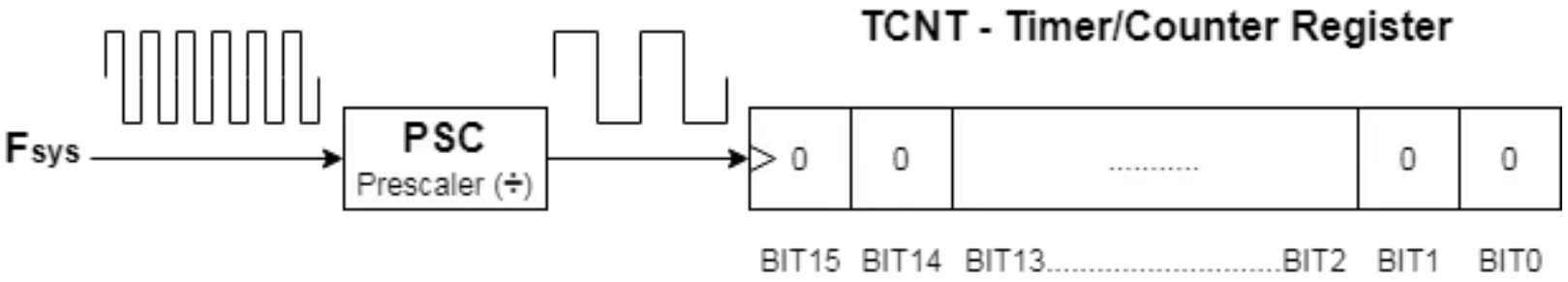
مثال: **TIM1** و **TIM8**.

ویژگی‌ها:

- تمام قابلیت‌های تایمر عمومی.
- قابلیت **Complementary PWM** (برای درایورهای پل H).
- قابلیت **Dead-time insertion**.
- Event و Break برای حفاظت مدار قدرت.

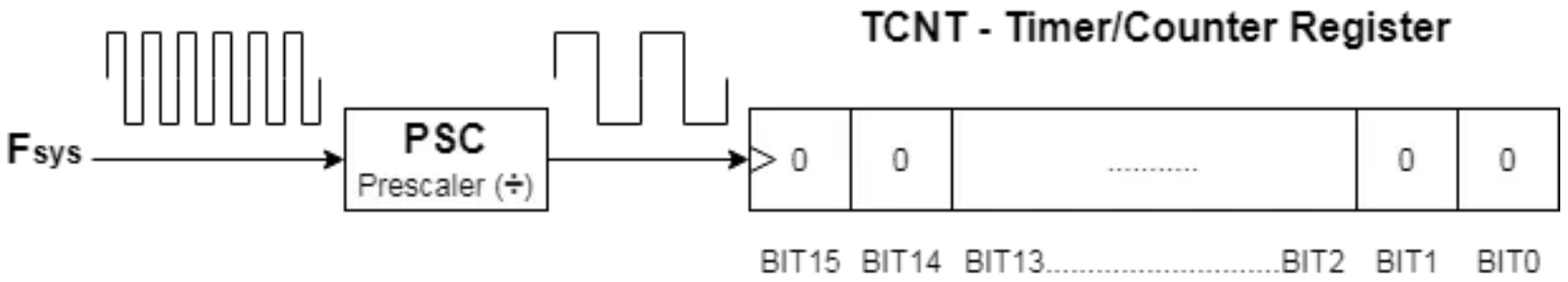
تايمرها در ميكروكنترلرهاي STM32

- تايمر در ميكروكنترلرهاي STM 32 داراي يك شمارنده ١٦ بيتي است (در سري هاي متفاوت اين عدد تا ٣٢ بيت هم مي رسد)
- اين شمارنده ميتواند به صورت بالا شمار، پـايين شـمار و بـالا-پـايين شمار، شمارش كند.
- نكته مهم اين است كه اين شـمارنده ١٦ بيتـي بـا چـه فركانسـي ميتواند شمارش كند.
- حداكثر فركانس واحد همه ي تايمرها در ميكروكنترلر مدنظر ما، 48 مگاهرتز است كه از طريق باسهاي مربوطه تأمين مي شود.



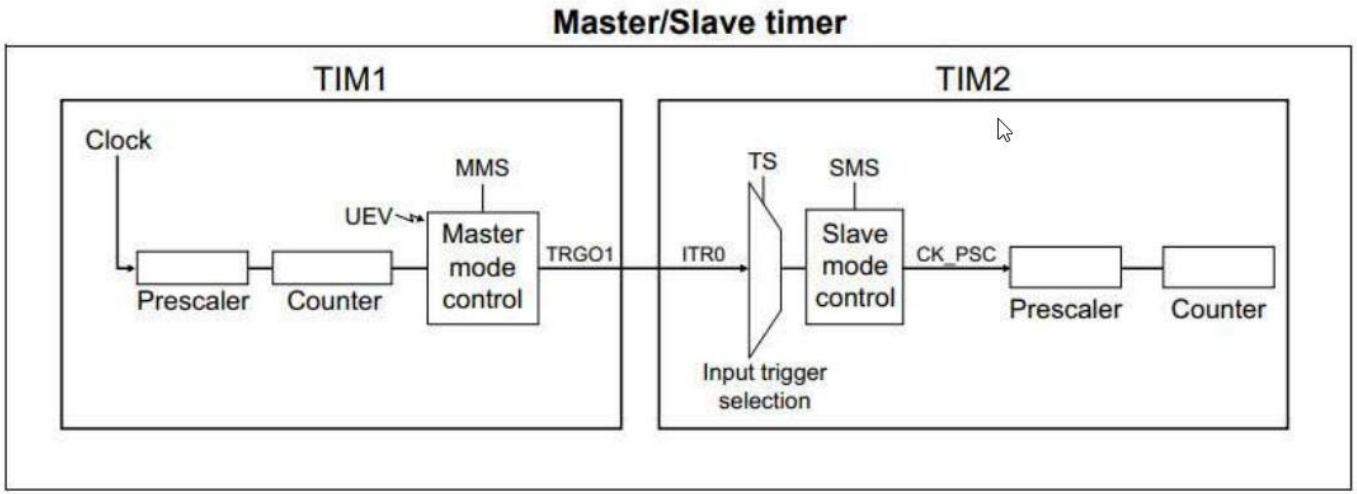
تايمرها در ميكروكنترلرهاي STM32

- حال مي توان مستقيما اين فركانس 48 MHz را مستقيماً به شمارنده ١٦ بيتي داد.
- يا اين فركـانس را بـا استفاده از Prescaler به فركانسهاي كوچكتري تبديل كرده و سپس آن را به شـمارنده بـدهيم.
- تـايمر در ميكروكنترلرهاي STM 32 داراي يك Prescaler با طول ١٦ بيت است كه فركانس ورودي واحد تايمر را به عددي بين ١ تا ٦٥٥٣٦ تقسيم مي كند.
- علاوه بر اينكه با استفاده از Prescalerها و ضرب كننده هاي فركانسي كه قبل از واحـد تـايمر قـرار دارند، ميتوان فركانس ورودي واحد تايمر را تعيين نمود، با استفاده از Prescaler كه در خود واحد تـايمر قرار دارد مي‌توان فركانس را تا حد بسيار زيادي، و تقريباً به هر عدد دلخواه تغيير داد.

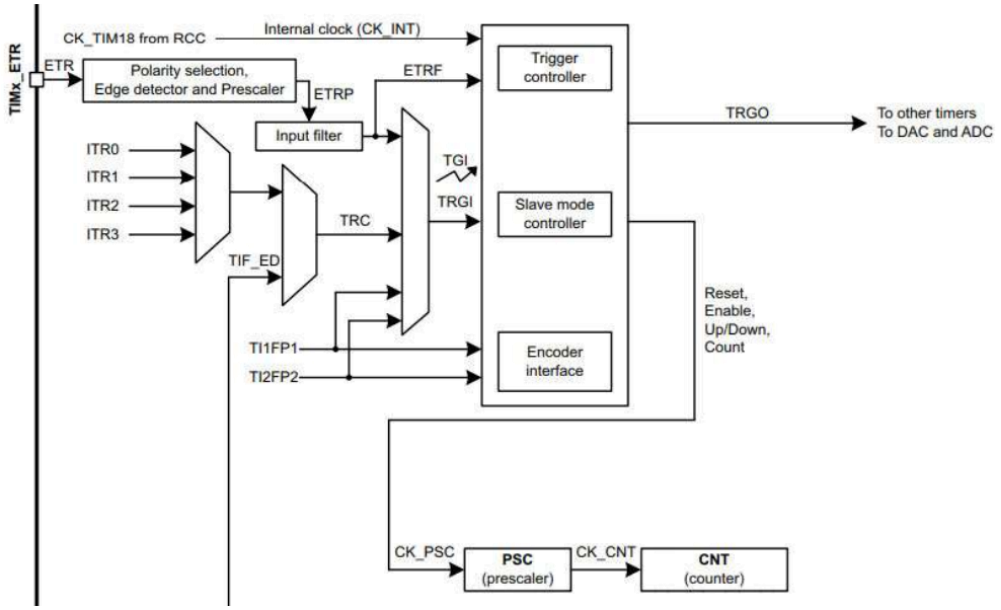


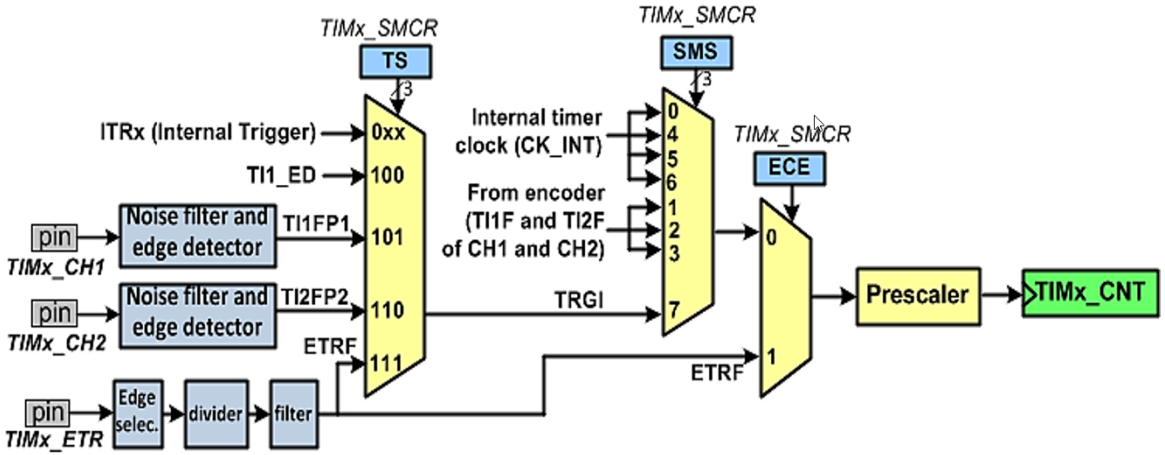
منابع کلاک تایمرها در میکروکنترلرهاي STM32

- کلاك ورودي بـــه واحد Timer در میکروکنترلرهاي STM 32 مي‌تواند از منابع مختلفي تامین شود:
  - ✓ کلاك داخلي میکروکنترلر،
  - ✓ اسیلاتورهای خارجي
  - ✓ از طریـق یـك تـایمر دیگـر
  - ✓ در صورتی که کلاك تایمر از طریق یك تایمر دیگر تـأمین شـود، در ایـن حالـت یـك تـایمر به عنوان Master و تایمر دیگر به عنوان Slave در نظر گرفته مي شود.



منابع کلاک تایمرها در میکروکنترلرهاي STM32



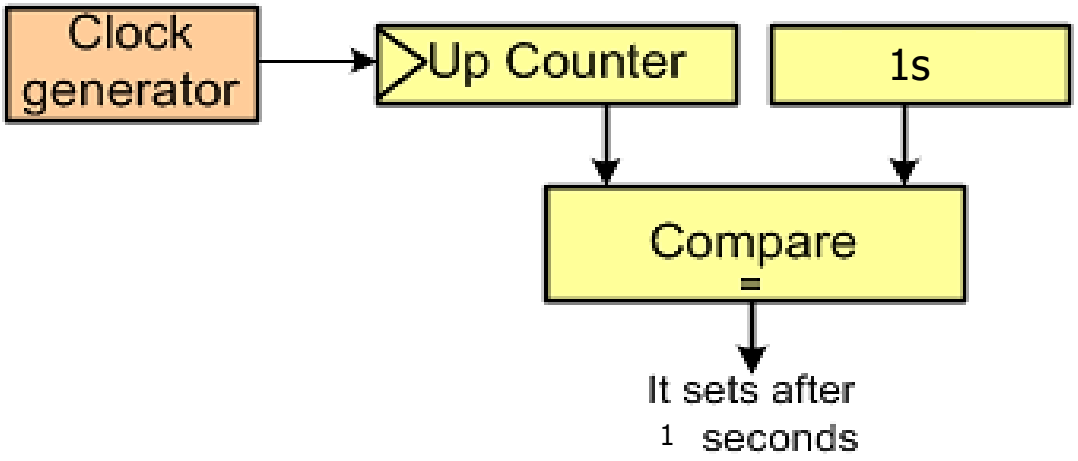


کاربرد تایمرها در میکروکنترلرهای STM32

- Timer در میکروکنترلرهای STM32 قابلیت‌های زیادی دارد، اما سه کاربرد مهم و پایه‌ای آن که در این درس به آن‌ها پرداخته می‌شود شامل موارد زیر است:
  - ✓ **ایجاد پایه زمانی (Time Base)**
  - ✓ **اندازه‌گیری سیگنال ورودی با استفاده از Input Capture**
  - ✓ **تولید سیگنال PWM**
- بسیاری از Timer های STM32 دارای ۴ کانال هستند.
- برخی مدل‌ها ۲ کانال دارند و Timer های Basic بدون کانال هستند.
- کانال یک بخش سخت‌افزاری در تایمر است که هرکدام می‌تواند یکی از عملیات‌های زیر را انجام دهد:
  - ✓ **PWM Output** - تولید موج PWM روی یک پایه (مثل PWM برای کنترل موتور، LED، شارژ، بالانس و ...)
  - ✓ **Output Compare (OC)** - مقایسه مقدار شمارنده با یک مقدار مشخص و تولید خروجی
  - ✓ **Input Capture (IC)** - اندازه‌گیری زمان ورود یک پالس - (مثل سنجش فرکانس، اندازه‌گیری فاصله، انکدر)
  - ✓ **One-Pulse Mode** - ایجاد یک پالس دقیق
  - ✓ **Encoder Interface** - استفاده از دو کانال برای خواندن انکدر مکانیکی
- هر کانال مستقل از بقیه کانال‌ها کار می‌کند، اما همه آن‌ها توسط یک **تایمر واحد** مدیریت می‌شوند.

مد Time Base

- در ادامه قصد داریم با استفاده از یکی از Timerها، **پایه زمانی** ایجاد کنیم و با تنظیمات مناسب، مدت‌زمان ۱ ثانیه را اندازه‌گیری کنیم.
- در این مثال تایمر را در حالت بالا شمار راه اندازي كرده و پس ازاينكه اعداد شمارش شده توسط شمارنده معادل ۱ ثانیه شد، وقفه واحد تايمر را فعال كرده تا متوجه سپري شدن زمان ۱ ثانیه بشویم و متناسب با آن عملیات موردنظر را انجام دهیم.



مد Time Base

- چگونه تشخیص دهیم شمارنده معادل ۱ ثانیه شمرده است؟

برای کنترل سرعت شمارش تایمر، از دو پارامتر استفاده می‌کنیم:

۱) منبع کلاک Timer

در اینجا منبع کلاک داخلی میکرو را با فرکانس ۸ MHz انتخاب می‌کنیم.

۲) Prescaler (PSC)

- ✓ Prescaler فرکانس کلاک را تقسیم می‌کند تا سرعت شمارش Counter تنظیم شود.

اگر Prescaler را ۷۹۹۹ قرار دهیم: از آنجا که شمارش از 0 شروع می شود، Prescaler مقدار ۸۰۰۰ را اعمال می‌کند

- ✓ فرکانس ورودی Counter برابر می‌شود با:

$$f_{CNT} = \frac{8MHz}{8000} = 1kHz$$

هر Counter یک 1ms واحد افزایش پیدا می‌کند.

مد Time Base

- برای دستیابی به ۱ ثانیه:
- اگر هر شمارش = ۱ ms باشد، نیاز داریم تایمر ۱۰۰۰ بار بشمارد
- اینجاست که از رجیستر ARR (Auto-Reload Register) استفاده می‌کنیم.

- نقش رجیستر ARR:
- ✓ مقدار ARR تعیین می‌کند شمارنده تا چه عددی بشمارد.
- ✓ وقتی Counter به مقدار ARR رسید: Counter صفر می‌شود (در حالت Upcount)
- ✓ وقفه‌ی Update Interrupt تولید می‌شود

در این مثال:

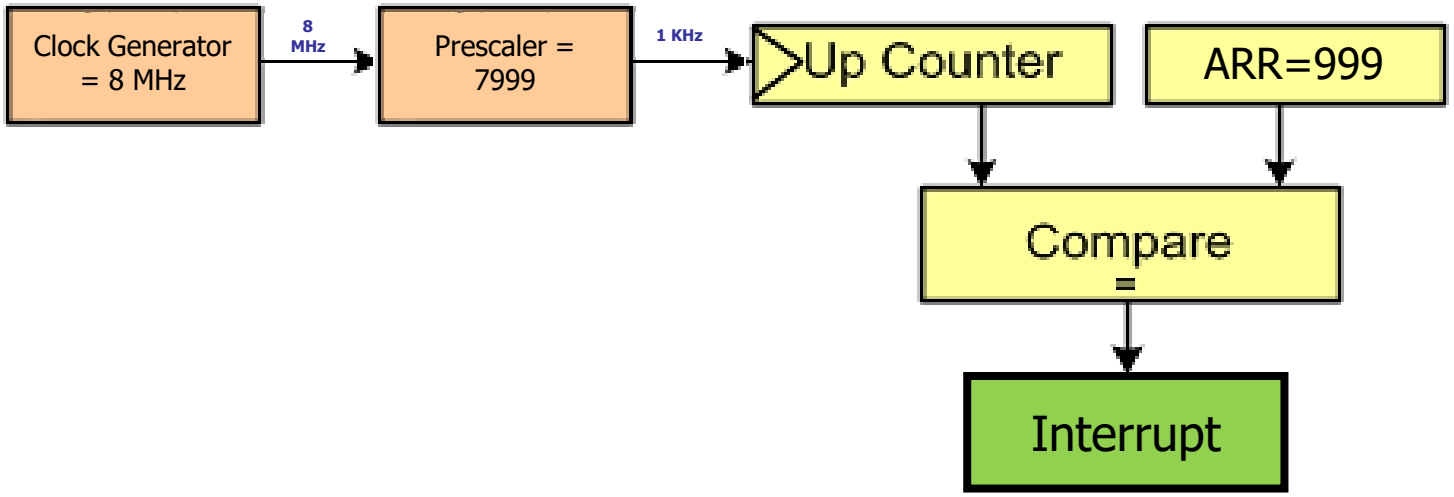
$$ARR = 999$$

(چون از صفر شروع می‌شود > مجموع ۱۰۰۰ شمارش = ۱ ثانیه)

مد Time Base

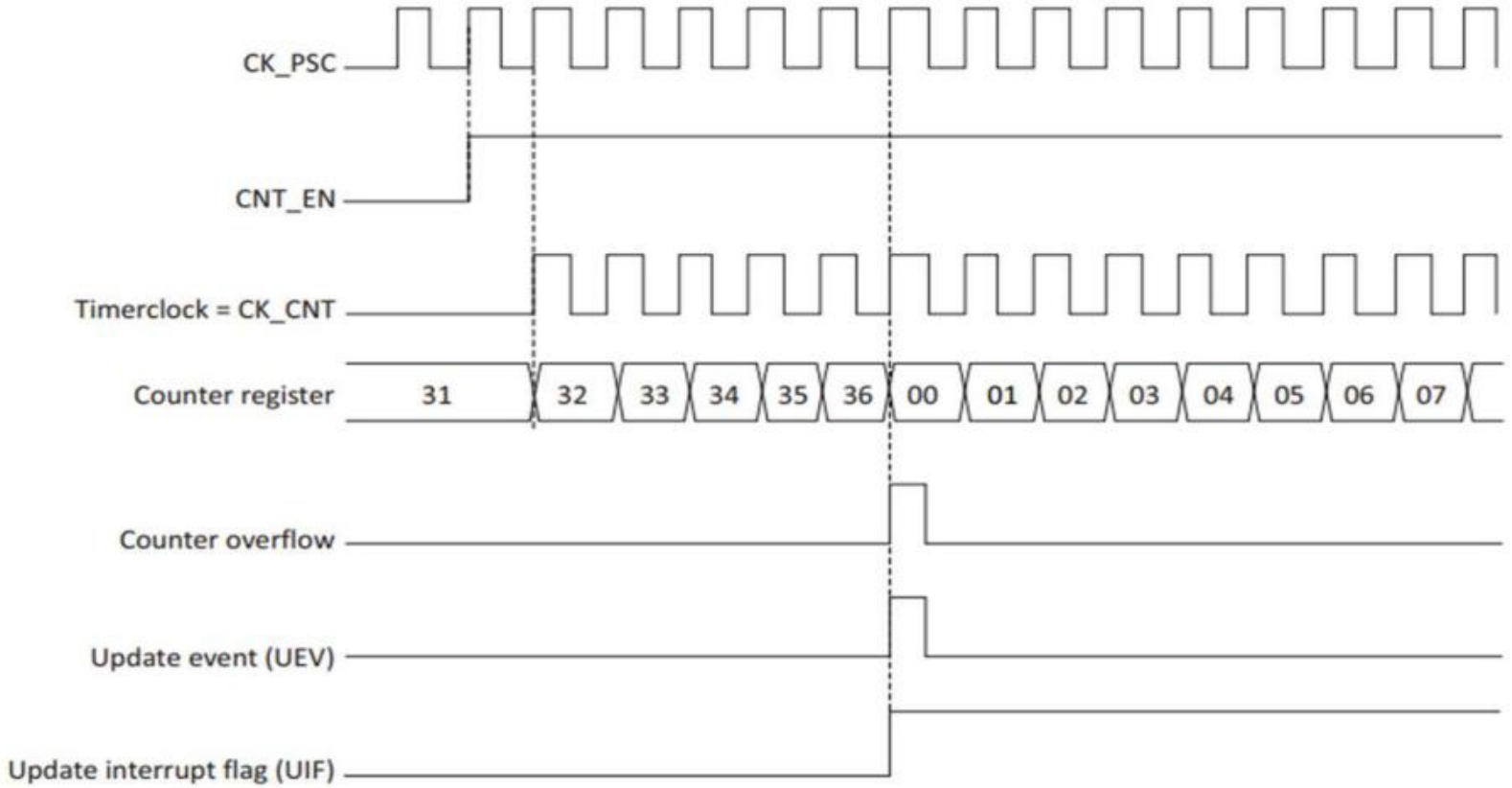
با تنظیمات زیر:

- ✓ کلاک تایمر = ۸ MHz
- ✓ Prescaler = 7999
- ✓ فرکانس شمارنده = 1 kHz
- ✓ ARR = 999 > مدت شمارش = 1000ms = 1s
- ✓ هر ۱ ثانیه تایمر یک **وقفه** می‌دهد و در روتین وقفه می‌توانیم عملیات موردنظر را انجام دهیم.



مد Time Base

مثال:



مد Input Capture

مفهوم Input Capture

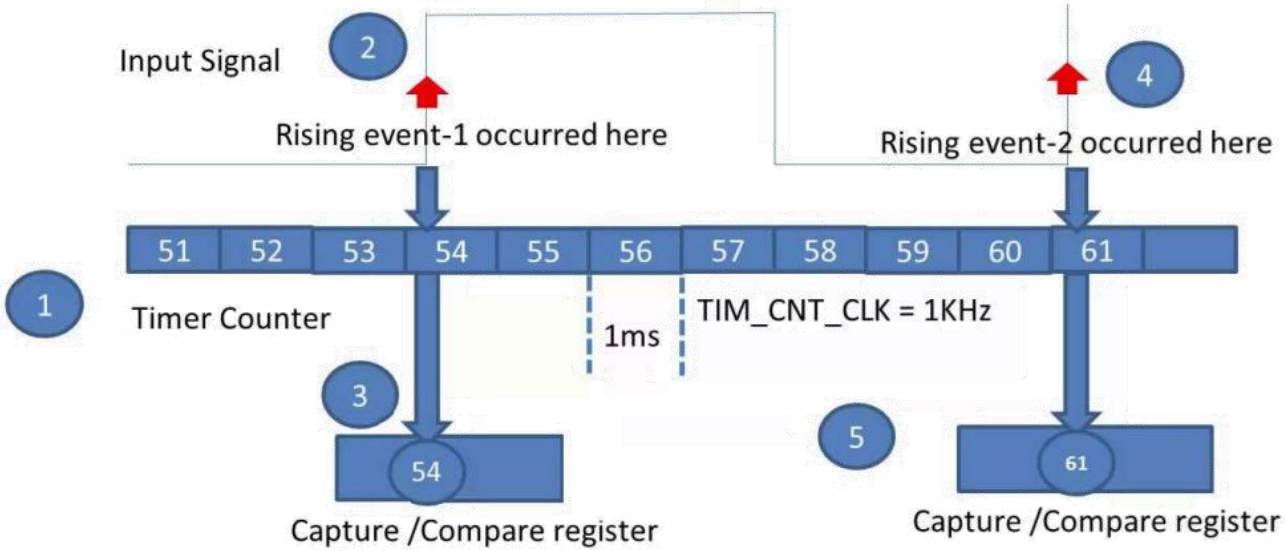
- ✓ Input Capture به‌معنی «ثبت مقدار ورودی» است.
- ✓ در این حالت، تایمر در هر رویداد ورودی (مثلاً لبه‌ی بالارونده یا پایین‌رونده) مقدار لحظه‌ای شمارنده (CNT) را در رجیستر Capture ذخیره می‌کند.
- ✓ با این کار می‌توان **فرکانس**، **دوره تناوب** یا **پهنای پالس سیگنال ورودی** را اندازه گرفت.

فرآیند اندازه‌گیری فرکانس در مد Input Capture

۱) فعال‌کردن تایمر (مرحله ۱)

- ✓ ابتدا تایمر را با یک کلاک مشخص (مثلاً 1 kHz) در حالت بالاشمار راه‌اندازی می‌کنیم.
- ✓ شمارنده از ۰ شروع می‌شود و با هر تحریک کلاک، یک واحد افزایش پیدا می‌کند.

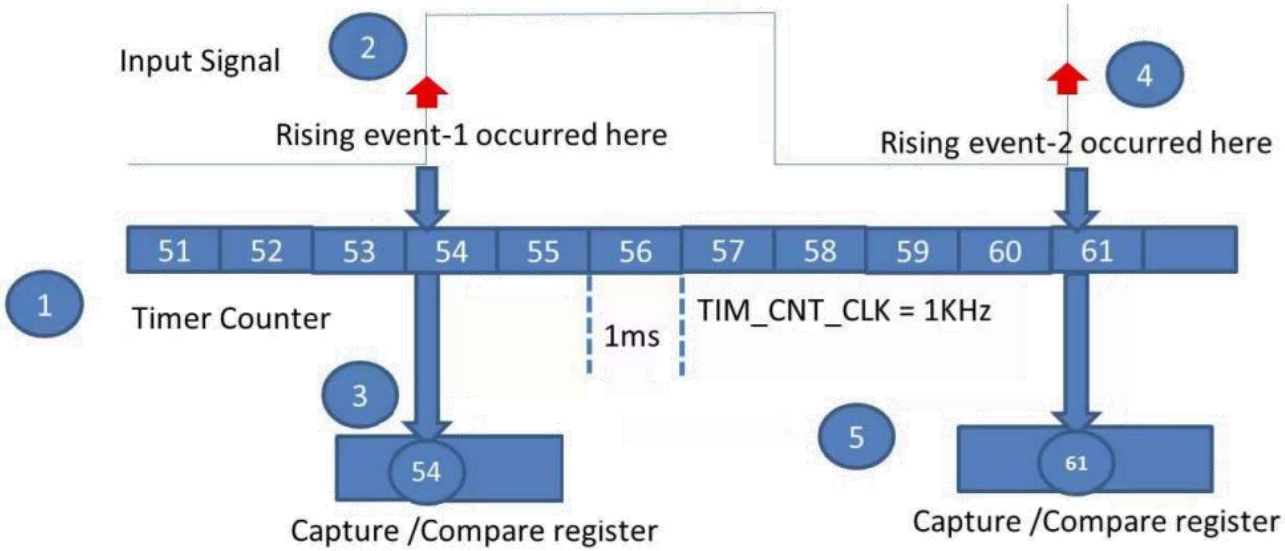
مد Input Capture



2) اعمال سیگنال ورودی (مرحله 2)

- ✓ سیگنال ورودی را به پین یکی از کانال‌های تایمر وصل می‌کنیم.
- ✓ کانال را حساس به لبه‌ی بالارونده تنظیم می‌کنیم.
- ✓ هر بار که لبه رخ دهد:یک وقفه‌ی Capture ایجاد می‌شود.
- ✓ مقدار CNTدر همان لحظه ذخیره می‌شود.

مد Input Capture

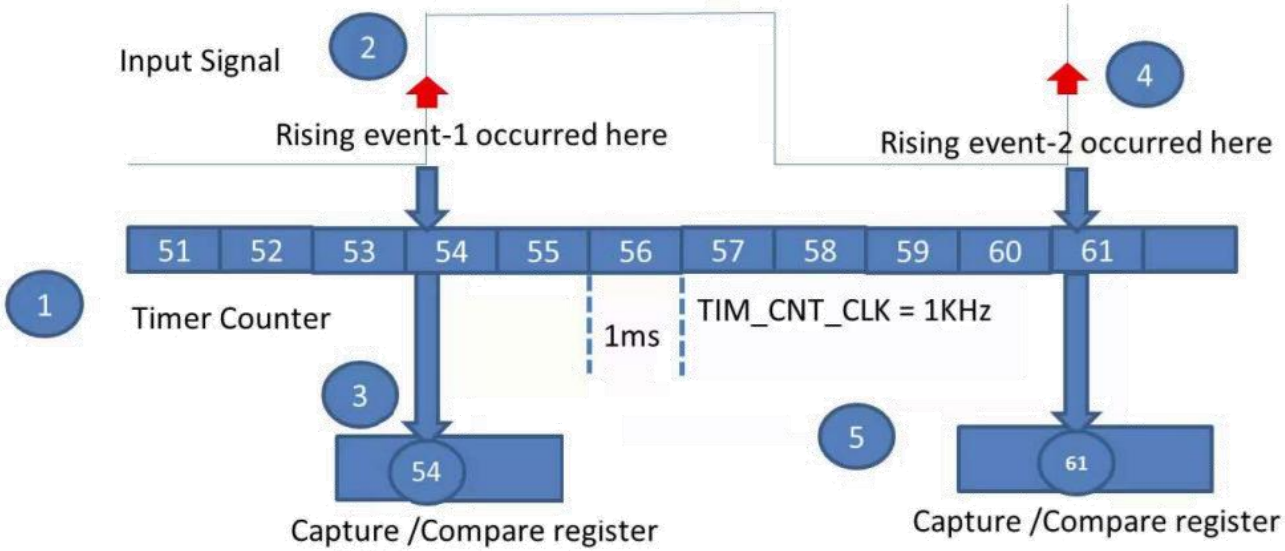


۳) اولین رویداد Capture (مرحله 3)

در اولین لبه‌ی بالارونده:

- ✓ وقفه رخ می‌دهد
- ✓ مقدار CNTکه مثلاً ۵۴ بوده، در رجیستر Capture ذخیره می‌شود
- ✓ شمارنده ادامه می‌دهد و متوقف نمی‌شود

مد Input Capture



۴) دومین رویداد Capture (مرحله 4 و 5)

در دومین لبه‌ی بالارونده:

- ✓ دوباره وقفه‌ی Capture فعال می‌شود
- ✓ مقدار جدید CNT که حالا ۶۱ است، در رجیستر Capture ذخیره می‌شود

۵) محاسبه فرکانس سیگنال ورودی

اکنون سه داده داریم:

۱) مقدار Capture اول = 54

۲) مقدار Capture دوم = 61

۳) فرکانس کلاک تایمر = 1 kHz (یعنی هر شمارش = 1 ms)

اختلاف دو مقدار:

$$\Delta = 61 - 54 = 7$$

این یعنی **فاصله دو لبه ۷ میلی‌ثانیه** بوده است.

بنابراین دوره تناوب سیگنال ورودی:

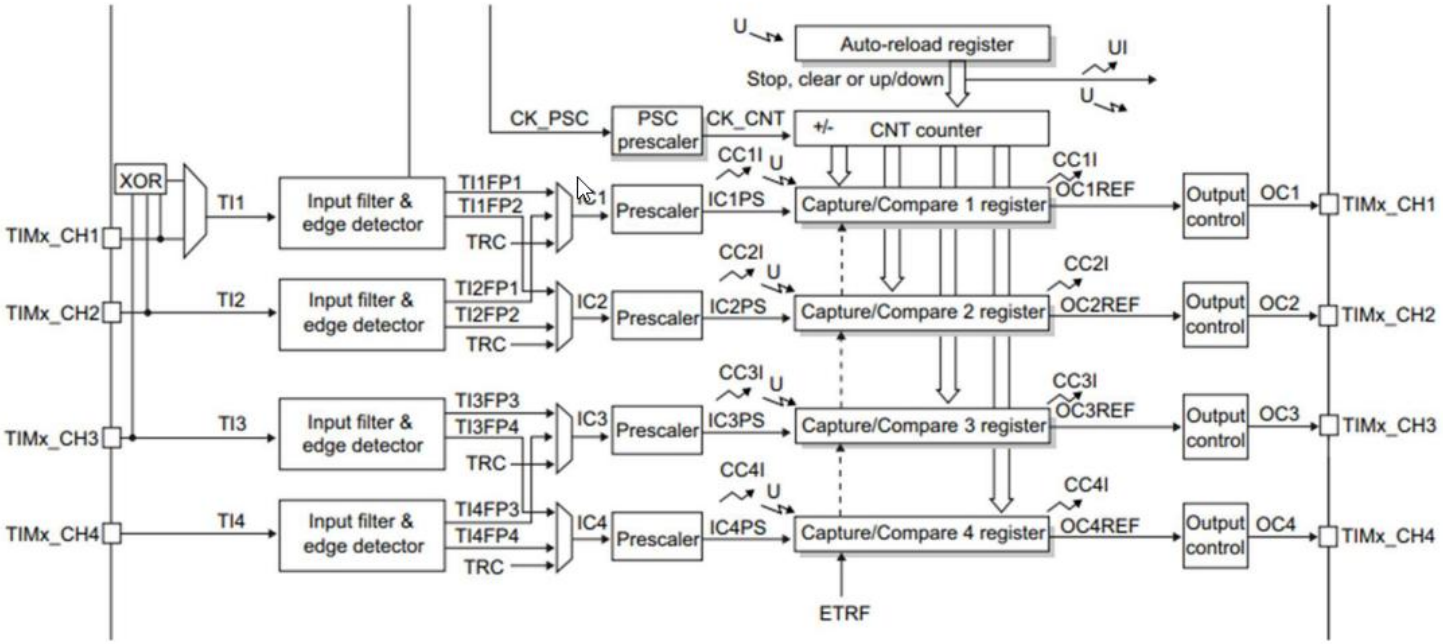
$$T = 7\ ms$$

و فرکانس:

$$f = \frac{1}{T} = \frac{1}{0.007} \approx 142.8\ Hz$$

بدست می آید.

مد Input Capture



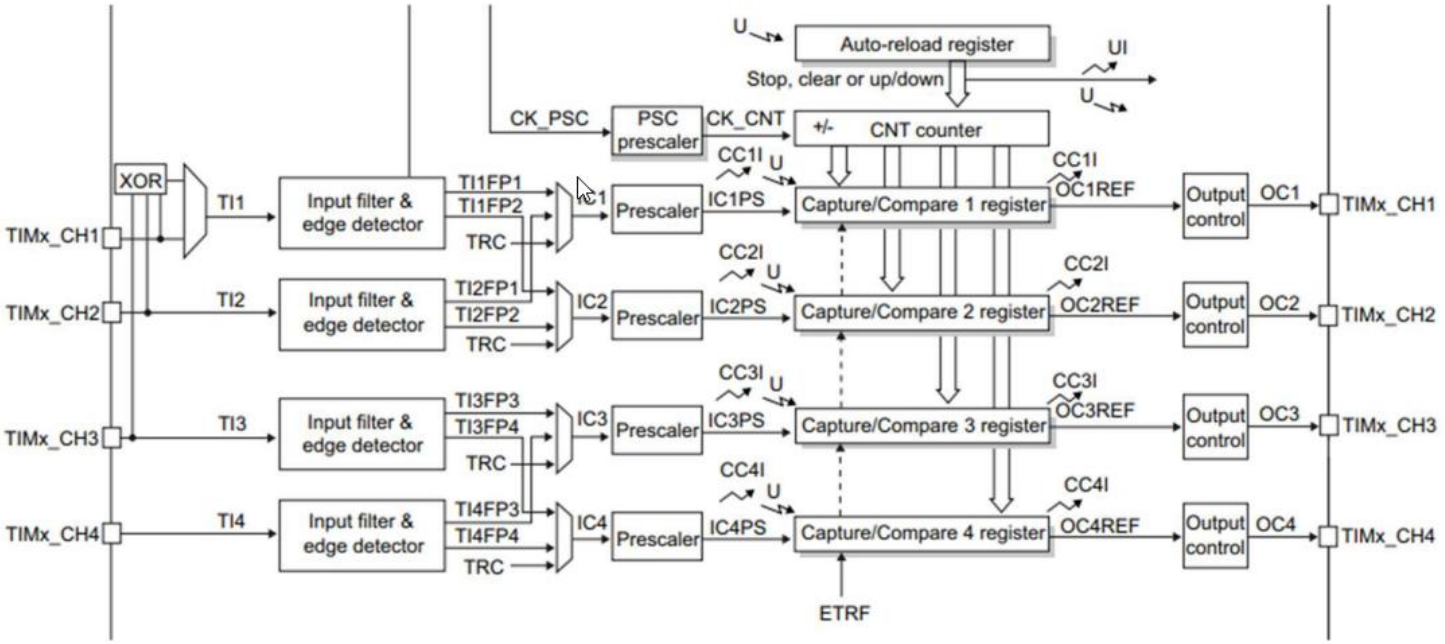
برای تفهیم بهتر مد capture Input در میکروکنترلرهای STM 32 به تصویر بالا دقت کنید:

1) مسیر ورودی سیگنال

برای استفاده از حالت Input Capture جهت اندازه‌گیری فرکانس یک سیگنال، ابتدا باید سیگنال را به یکی از کانال‌های ورودی تایمر (TIMx_CH1..4) وصل کرد.

همان‌طور که در تصویر مشاهده می شود، هر کانال قبل از رسیدن به واحد Capture از سه بلوک عبور می‌کند.

مد Input Capture



1. فیلتر دیجیتال (Input Filter):

جهت حذف نویز، کاهش خطا در لبه‌ها (در این پروژه برای سادگی غیرفعال می‌شود)

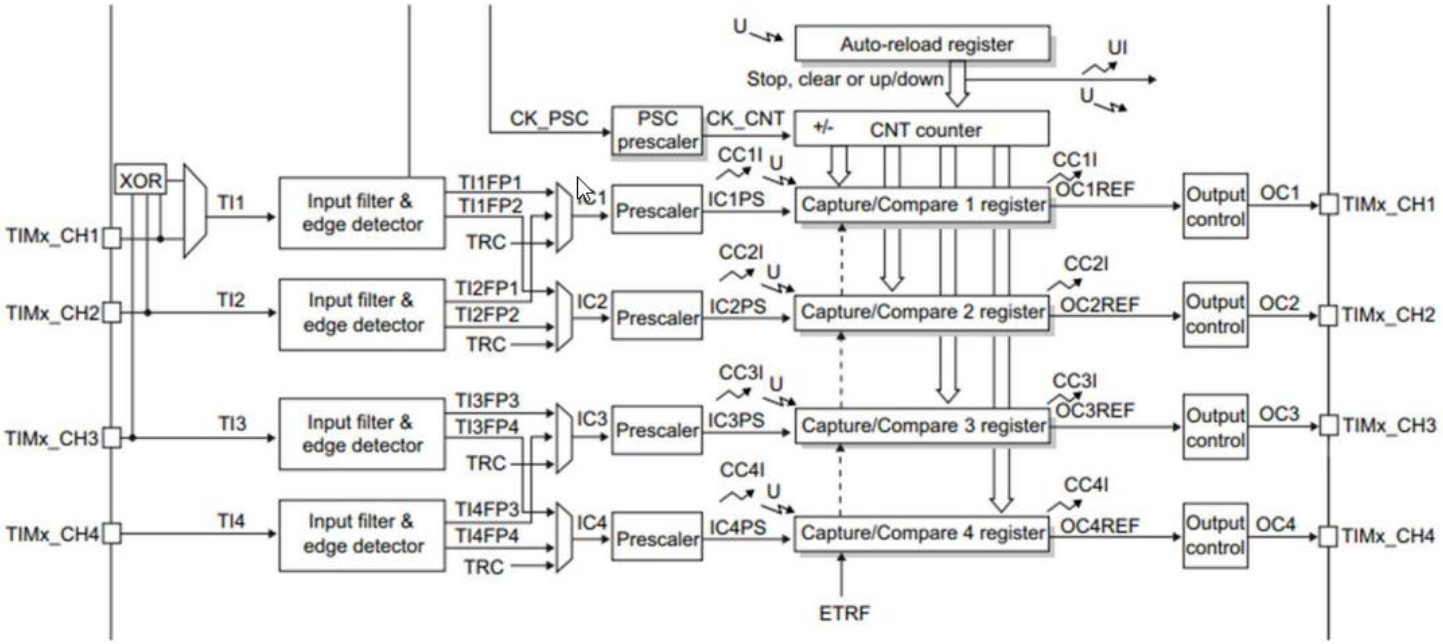
2. تشخیص لبه (Edge Detector):

قابل تنظیم روی لبه بالا/پایین (آن را روی لبه بالارونده قرار می‌دهیم تا با هر لبه یک وقفه تولید شود)

3. پرسکیلر ورودی (Input Prescaler):

می‌تواند هر 1، 2، 4 یا 8 رویداد را یکی کند.(در اینجا استفاده نمی‌کنیم)

مد Input Capture



۲) انتقال داده به رجیستر Capture

پس از اعمال تنظیمات، با وقوع هر لبه بالارونده:

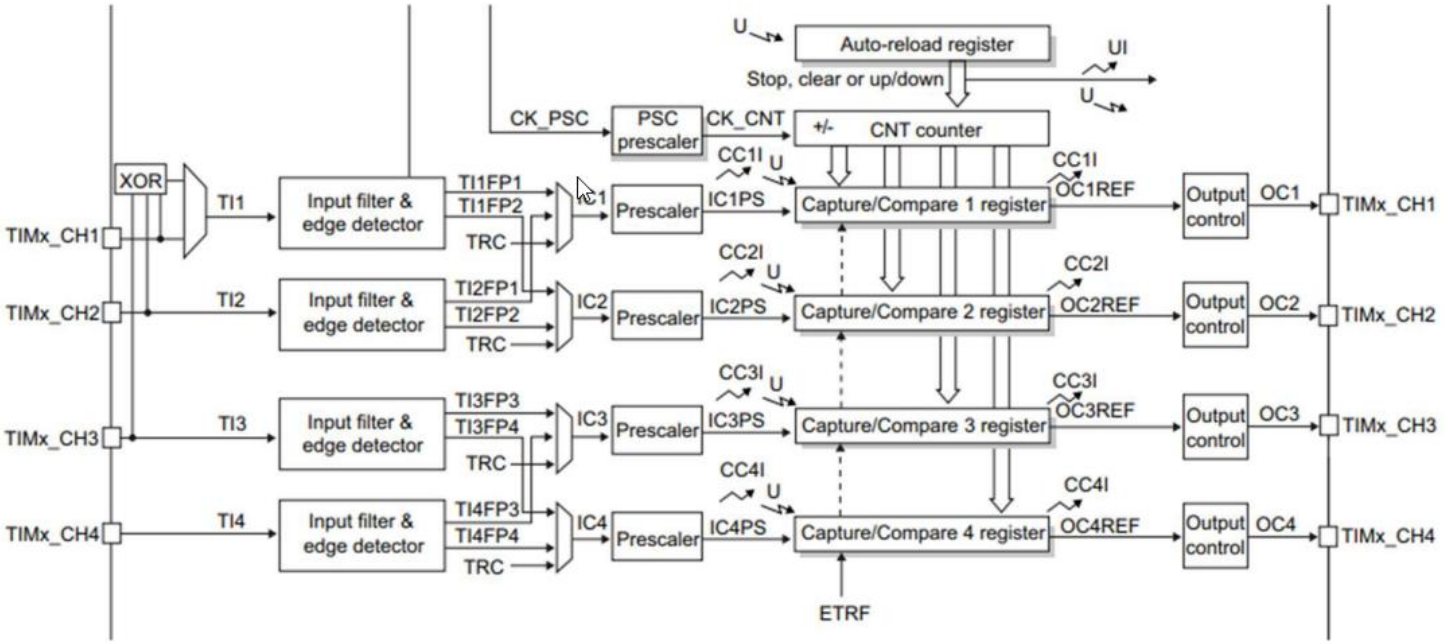
Edge Detector یک رخداد Capture تولید می‌کند.

مقدار لحظه‌ای CNT (شمارنده تایمر) به رجیستر Capture مربوطه منتقل می‌شود.

(مثلاً CCR1 برای کانال 1)

این همان مقدار کلیدی است که برای محاسبه دوره تناوب و فرکانس استفاده می‌شود.

مد Input Capture



۳) اهمیت کلیدِی دو مؤلفه:

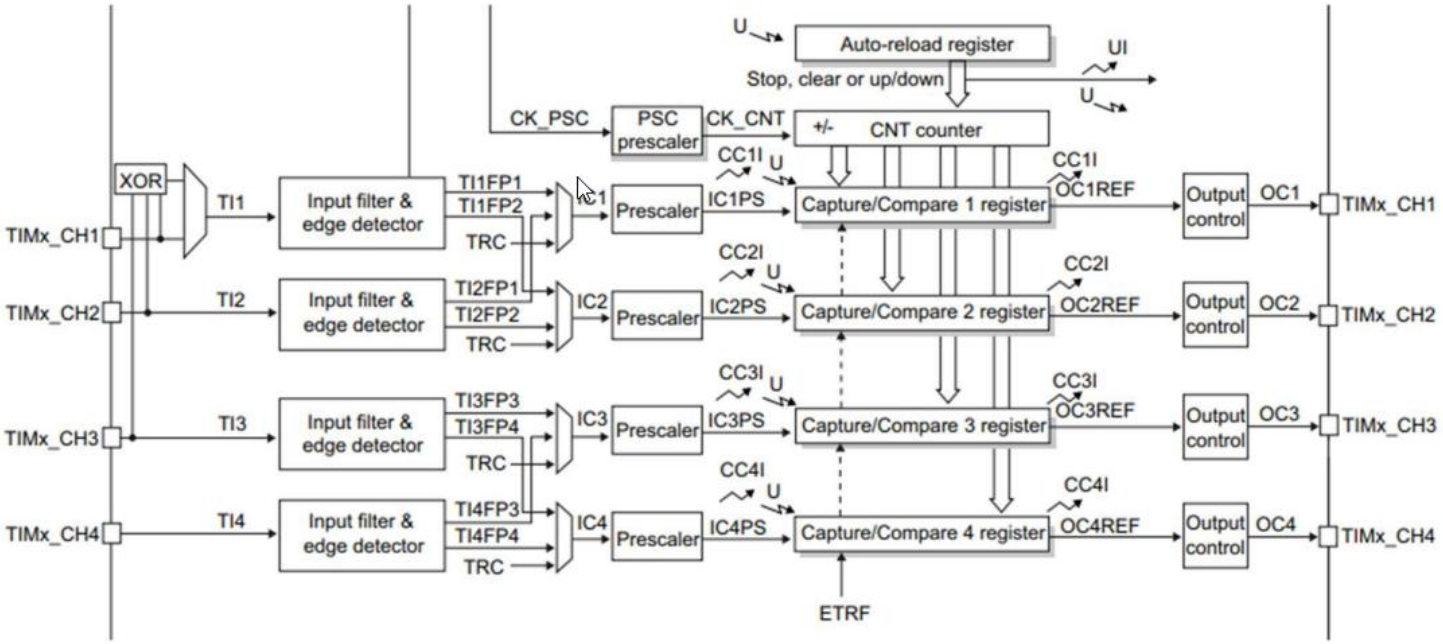
- فرکانس کلاک شمارنده (TIMx_CLK / PSC)

این مقدار تعیین می‌کند که هر پله از CNT چند نانو/میکروثانیه است.

- مقدار Auto-Reload Register (ARR)

تعیین‌کننده‌ی مقدار بیشینه‌ی شمارش (مثلاً 65535 برای تایمر 16 بیتی).

مد Input Capture



۴) رفتار شمارنده و اهمیت سرریز (Overflow)

شمارنده از 0 شمارش را شروع می‌کند و تا مقدار ARR بالا می‌رود.

وقتی به ARR رسید > Overflow رخ می‌دهد > شمارنده دوباره از 0 شروع می‌کند

در نتیجه هنگام اندازه‌گیری فاصله بین دو لبه ورودی، سه حالت ممکن است رخ دهد

مد Input Capture

حالت اول: بدون سرریز

هر دو رخداد قبل از رسیدن به ARR اتفاق می‌افتند.

مثال:

Capture اول: 500

Capture دوم: 1000

اختلاف: $\Delta = 1000 - 500 = 500$

هیچ مشکلی وجود ندارد؛ مقدار کاملاً صحیح است.

مد Input Capture

حالت دوم: یک بار سرریز رخ داده است

Capture اول قبل از Overflow

Capture دوم بعد از Overflow

مثال:

Capture اول: 40000

Capture دوم: 2000

چون شمارنده یک دور کامل زده است:

$$\Delta=(ARR-40000+1)+2000$$

این حالت ۱۰۰٪ قابل جبران است و هیچ خطایی ندارد.

زیرحالت‌ها

در حالت دوم ممکن است مقدار Capture دوم:

کمتر از اول باشد (مثال بالا > ۲۰۰۰)

یا حتی بیشتر از اول باشد (مثلاً ۵۰۰۰۰)

در هر دو حالت فرمول یکسان است چون overflow فقط یک بار رخ داده.

مد Input Capture

حالت سوم: بیش از یک بار سرریز رخ داده است

این حالت مشکل‌ساز است.

مثال:

Capture اول: 6000

Capture دوم: 3000

با دو بار یا بیشتر سرریز بین این دو رخداد

در این حالت دیگر نمی‌توانیم تعیین کنیم که شمارنده بین دو Capture چند بار overflow شده است و اختلاف CNT نامشخص می‌شود.

راه‌حل چیست؟

یک راهکار ساده:

زمان بین دو لبه را محدود کنیم (یعنی بازه بین دو Capture نباید از دامنه‌ی شمارنده بزرگ‌تر شود)

یا با اندازه‌گیری چند پالس به‌صورت تجمیعی مقدار اشتباه را تصحیح کنیم

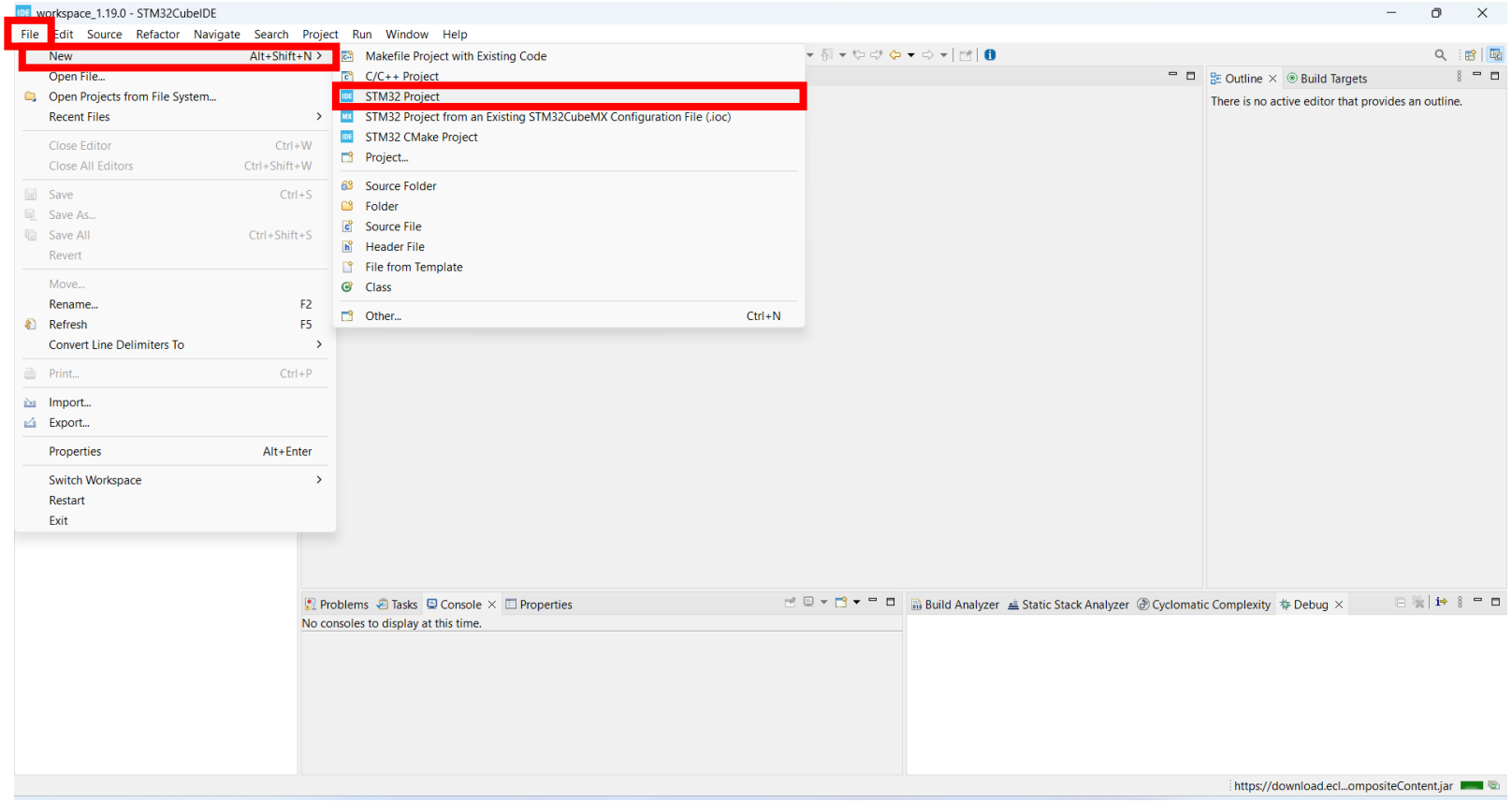
(مثلاً اندازه‌گیری 16 پریود پشت هم)

این روش‌ها در بخش کدنویسی قابل پیاده‌سازی است.

پیکربندی اولیه مسئله Time Base در نرم افزار STM32CubeIDE

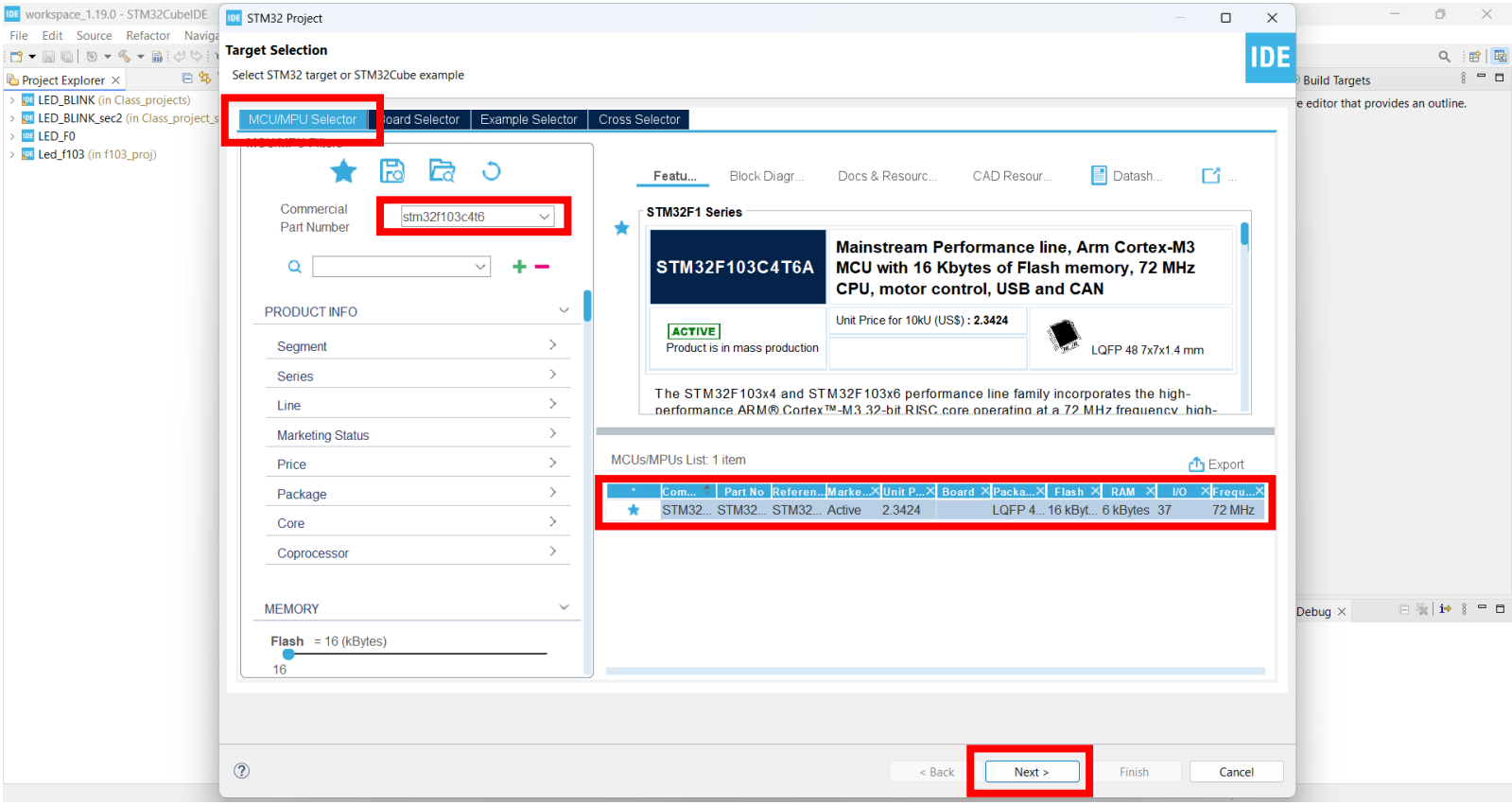
پیکربندی اولیه مسئله در نرم افزار STM32CubeIDE

ابتدا در نرم افزار STM32CubeIDE از منو File ، New و سپس گزینه New Project را انتخاب کنید.



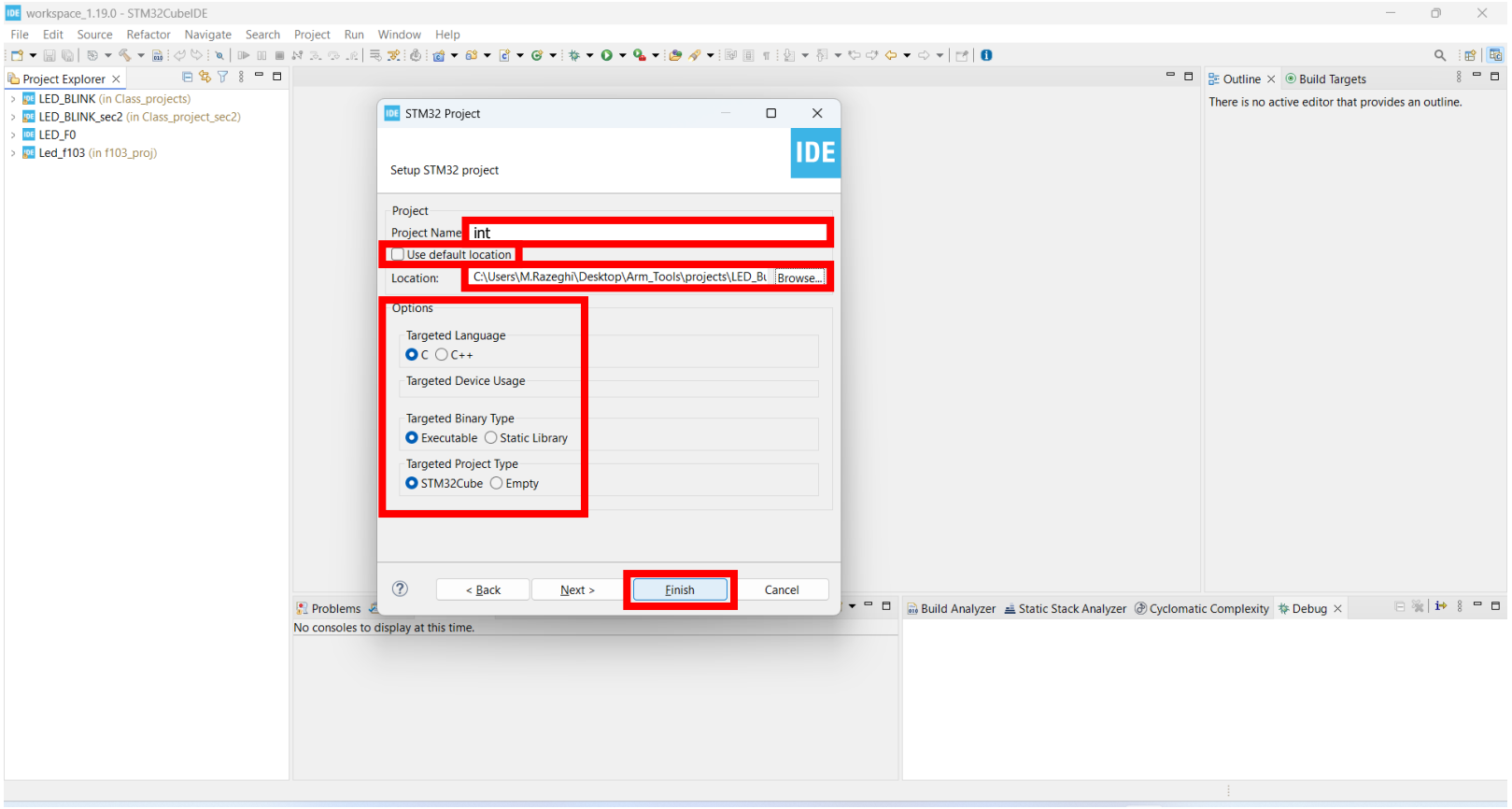
پیکربندی اولیه مسئله در نرم افزار STM32CubeIDE

سپس در پنجره باز شده در قسمت Part Number Search نام میکروکنترلر مورد نظر خود را انتخاب کنید (برای مثال STM32F103C4) و بر روی Next کلیک کنید.

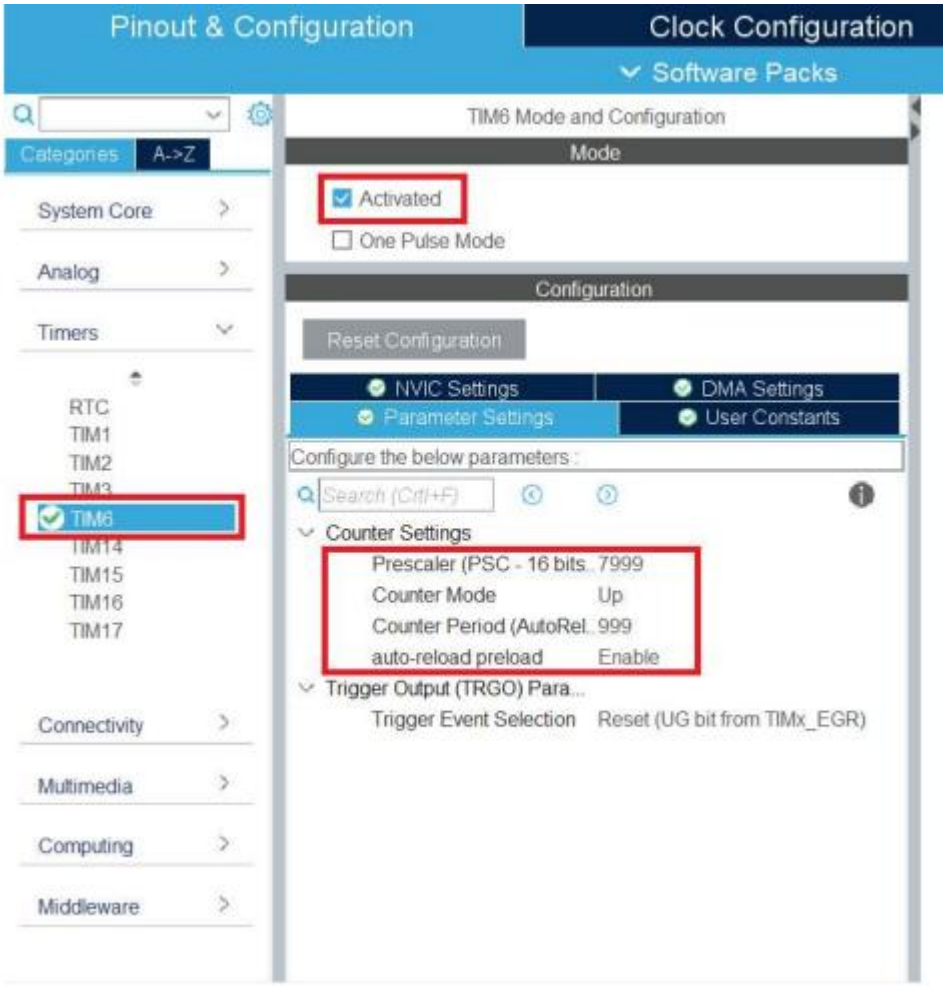


پیکربندی اولیه مسئله در نرم افزار STM32CubeIDE

سپس در پنجره باز شده در قسمت Project Name، نام پروژه و در صورت لزوم مکان ذخیره پروژه و تنظیمات مدنظر را مطابق تصویر را وارد کرده و نهایتا روی Finish کلیک کنید.



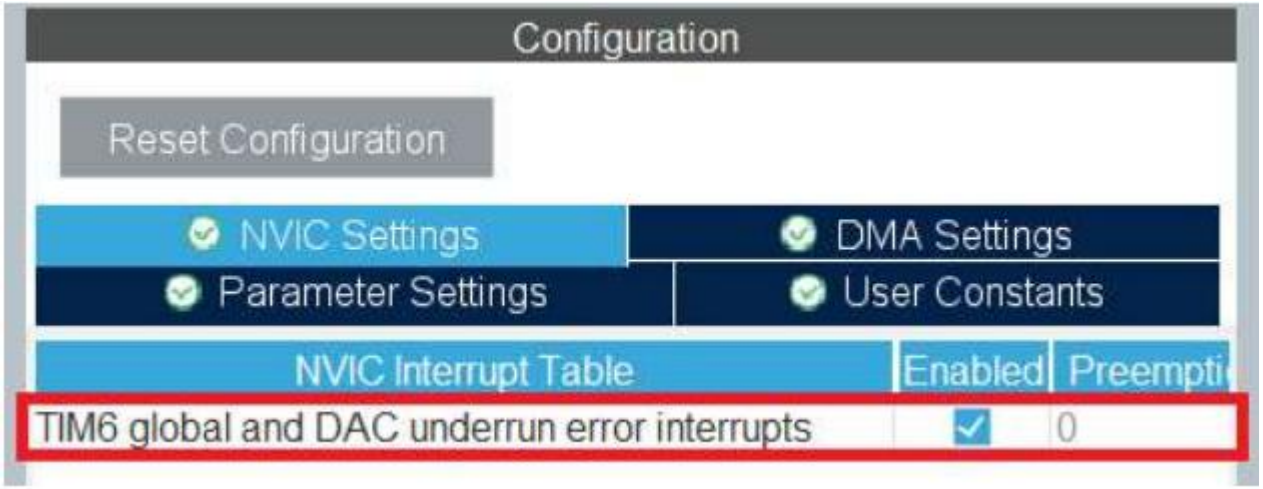
پیکربندی اولیه مسئله در نرم افزار STM32CubeIDE



حال مقادیر مناسب را مانند تصویر براي TIM 6 تنظيم كنيد، يك پين از ميكروكنترلر را هم بر روي حالت خروجي قرار دهيد تا هر ۱ ثانيه يك بار در روتين وقفه آن را Toggle كنيم (به طور پيش فرض می توان از ال ای دی c8 استفاده نمود):

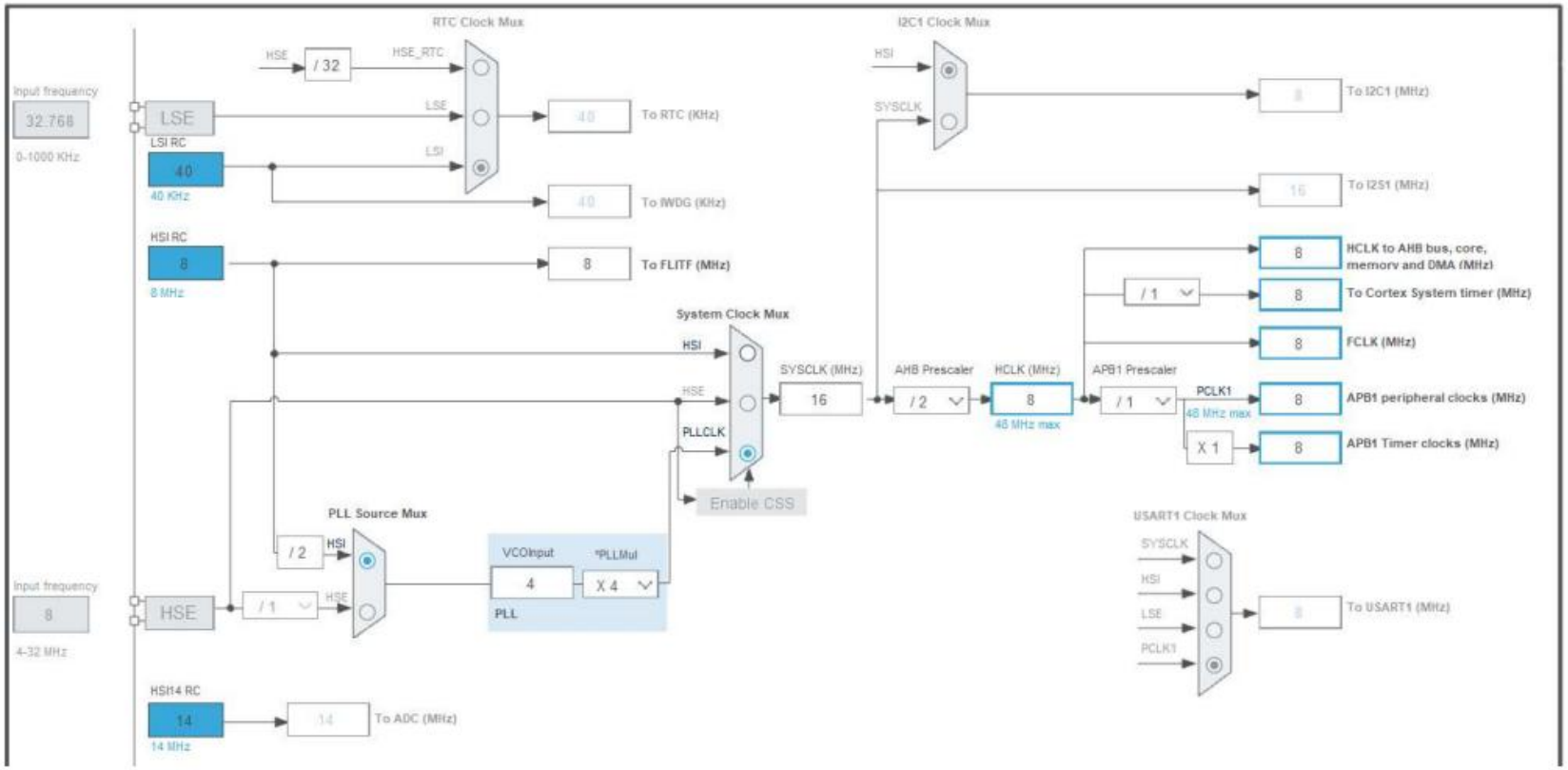
پیکربندی اولیه مسئله در نرم افزار STM32CubeIDE

در قسمت Setting NVIC وقفه مربوطه را نیز فعال کنید:



پیکربندی اولیه مسئله در نرم افزار STM32CubeIDE

تنظیمات Configuration Clock را به نحوي تنظيم كنيد كه كلاك ورودي تايمر 8MHz باشد:



کد نویسی مسئله Time Base در نرم افزار STM32CubeIDE

تـابع callBack در صورت رخداد وقفه تایمر:

```
/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
/* Prevent unused argument(s) compilation warning */
UNUSED(htim);
HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_8);
/* NOTE : This function should not be modified, when the
callback is needed,
the HAL_TIM_PeriodElapsedCallback could be implemented in the
user file
*/
}
/* USER CODE END 4 */
```

کد نویسی در نرم افزار STM32CubeIDE

دستور شروع به کار تایمر:

```
/* USER CODE BEGIN 2 */
 HAL_TIM_Base_Start_IT(&htim6);
/* USER CODE END 2 */
```

razeghizade@gmail.com

Razeghizade.pudica.ir

CREADITS: This presentation was created by M.Razeghizadeh
Please keep this slide for attribution