Timers in STM32

(1)

محتويات الجلسة:

1- مقدمة
2- المؤقتات Timers في متحكمات STM32
3- الأنواع المختلفة للمؤقتات
4- أنماط العمل المختلفة للمؤقتات في متحكمات STM32
5- Time-Base Unit
6- المقاطعات المتوفرة في المؤقتات في متحكمات STM32
7- ضبط الإعدادات في المؤقتات في متحكمات STM32
8- أوضاع الاستخدام المختلفة للمؤقتات في متحكمات STM32
9- التطبيق العملي الأول
10-التطبيق العملي الثاني

الأدوات اللازمة للجلسة:

- لوحة Nucleo-64bit
- كبل Type-A to Mini-B
- ليدات
- مفاتيح لحظية

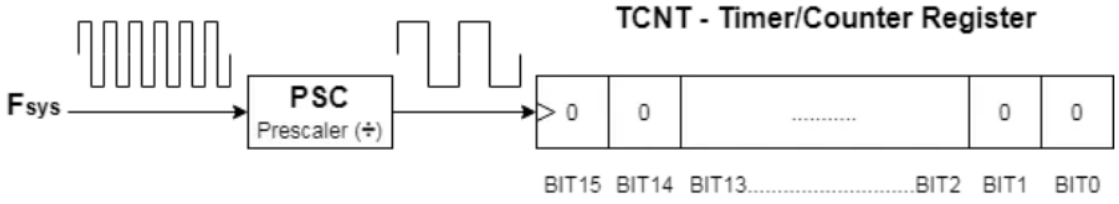
1- مقدمة :

في الأنظمة المدمجة يوجد بعض التطبيقات التي تتم بناءً على وقت ثابت أي تتم بطريقة متكررة كل فترة زمنية ثابتة، حيث عادةً يتم ا ستخدام دالة التأخير الزمني من مكتبة HAL وهي ()delay ، لكن هذه الدالة تت سبب في توقف المعالج عند هذا الـ سطر إلى أن ينتهي التأخير الزمني ثم يتم ا ستكمال تنفيذ التعليمات الأخرى الموجودة في الكود، ويعد هذا إهدار لسرعة المعالجة وتوقف معالجة بقية السطور من الكود وفقط التوقف عند تنفيذ هذا السطر التي توجد به ()delay، فلحل هذه المـ شكلة توفر المتحكمات المـ صغرة المؤقتات Timers ، وهي عبارة عن مجموعة من الوحدات الطرفية التي توفر الأزمنة المناسبة بالإضافة إلى العديد من المزايا الإضافية الأخرى.

2- المؤقتات Timers:

يوجد في متحكمات STM32 العديد من المؤقتات المختلفة كل منها بإمكانها العمل بأنماط مختلفة وتأدية الوظائف المطلوبة منها، حيث يعتبر المؤقت في أبـ سط أ شكاله وعند عمله بنمط Timer عبارة عن دارة منطقية تبدأ بالعد من الـ صفر وتزداد بمقدار عدة واحدة مع كل نبـ ضة ساعة للمتحكم، ولقد تمت إ ضافة العديد من المزايا للمؤقت التي تمكنه من العد التصاعدي و التنازلي على حدٍ سواء، أيضاً الـ prescaler - أو المقسم الترددي والذي يقوم بتقسيم تردد الساعة على عدد معين يتم اختياره، أيضاً دارة توليد نبضات الـ PWM - و دارة خاصة بـ Input - Capture والتي تستخدم عادةً لقياس تردد إشارة معينة وغيرها العديد من المزايا التي سنتطرق لها.
بفرض لدينا المؤقت الموضح بالشكل(1)، وهو عبارة عن مؤقت بطول 16بت بإمكانه العد من الصفر حتى يطفح المؤقت عند 65535 عدة، حيث تزداد القيمة الحالية للمؤقت مع كل نبـ ضة ساعة للمؤقت، لكن كما تلاحظ فإن

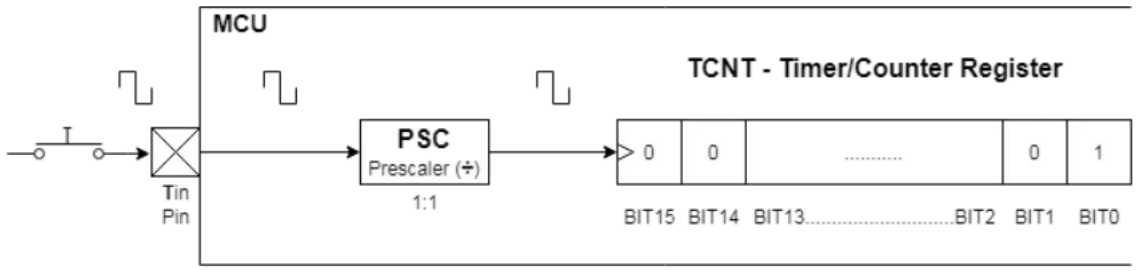
تردد الـ ساعة للمؤقت هي عبارة عن تردد ساعة النظام (المتحكم) Fsys بعد تق سيمها على المق سم الترددي Prescaler .



الشكل(1): المؤقت بنمط Timer

فعند عمل المؤقت بنمط Timer ، فإن المسجل TCNT تزداد قيمته بمقدار واحد مع كل نبضة ساعة للمؤقت حيث كما ذكرنا فإن تردد الـ ساعة للمؤقت هو (Fsys/PSC)، أي تردد ساعة المتحكم مق سومة على المق سم الترددي ، فإذا كان تردد الساعة Fsys هو 80MHZ والمقسم الترددي PSC هو 1:1024، فإن قيمة المسجل TCNT ستزداد بمقدار واحد كل 12.8usec، فإذا بدء المؤقت العد من الصفر عندها سيطفح عندما يصل للـ 65535 عدة أي بعد 0.839sec وسيتم توليد مقاطعة الطفحان.

أيضاً يمكن للمؤقت أن يعمل بنمط Counter وفي هذه الحالة سيكون مصدر الساعة للمؤقت عبارة عن إشارة خارجية ممكن أن تكون قادمة من مفتاح لحظي، عندها ستزداد قيمة المؤقت مع كل جبهة صاعدة/هابطة عند ضغط المفتاح اللحظي بالتالي سيعد المؤقت عدد المرات التي تم فيها ضغط المفتاح اللحظي كما هو مو ضح بالشكل التالي:



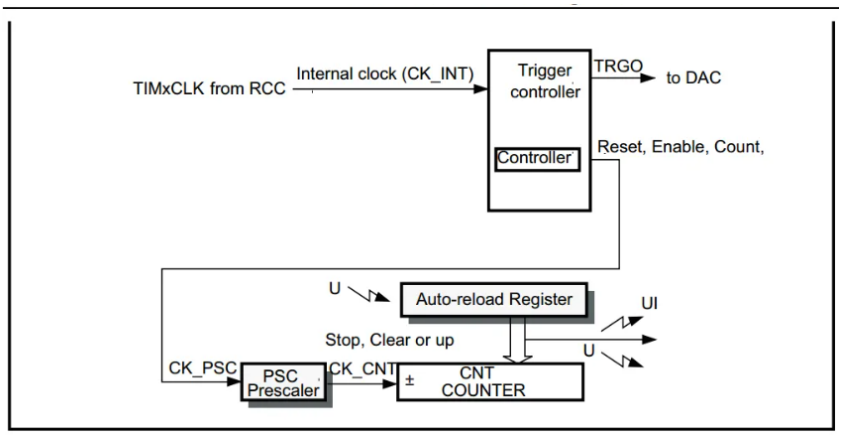
الشكل(2): المؤقت بنمط Counter

3- الأنواع المختلفة للمؤقتات:

يوجد في متحكمات STM32 العديد من المؤقتات المختلفة كل منها بإمكانها العمل بأنماط مختلفة وتأدية الوظائف والمهام المطلوبة منها، سنذكر منها الأنواع التالية:

- **المؤقتات الأساسية: Basic Timers**

وهو أ سهل تـ صنيف للمؤقتات في متحكمات STM32 ، وهو عبارة عن مؤقتات بطول 16-bit وتـ ستخدم كمولد للوقت Time base generator ولا يتـ صل بها أي طرف من أطراف المتحكم input/output pins ، كما يمكن أن تـ ستخدم لتغذية وحدة المبدل الرقمي تـ شابهي DAC حيث تتـ صل مخارج المؤقت داخلياً مع مداخل القدح للمبدل ، أيضاً يمكن أن تعمل هذه المؤقتات كـ master - لبقية المؤقتات (حيث يتم توليد نبضات الساعة منها والاعتماد عليها في باقي المؤقتات).



الشكل(3): المخطط الصندوقي للمؤقتات الأساسية Basic Timers

للمؤقت الأساسي عدة خصائص تتضمن:

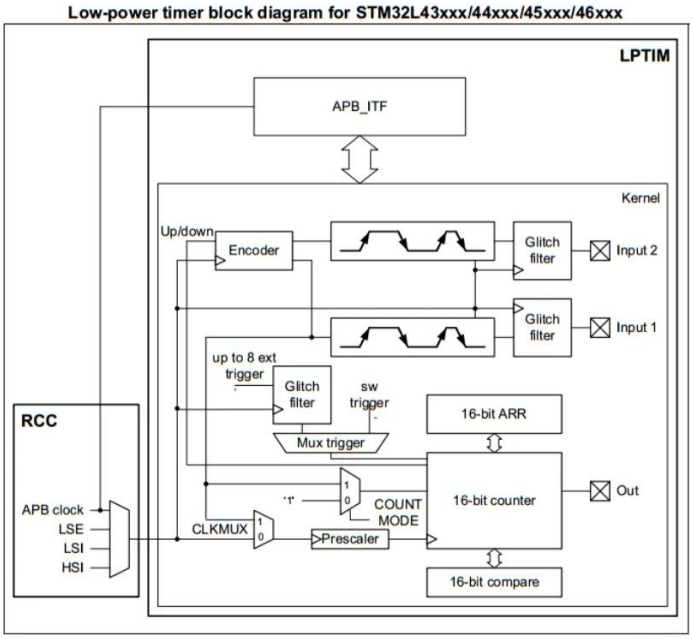
- عداد تصاعدي بطول 16بت
- مقسم ترددي بطول 16 بت يمكن تحميله بأي قيمة من 1 حتى 65536
- دارة مزامنة لقدح المبدل الرقمي التشابهي DAC
- توليد مقاطعة عند طفحان العداد counter overflow

للمؤقت الأساسي عدة مسجلات جميعها قابلة للكتابة/القراءة من خلال الكود، منها:

- مسجل العداد (مسجل القيمة الحالية للمؤقت) TIMx_CNT
- مسجل المقسم الترددي (TIMx_PSC)
- مسجل Auto-Reload (TIMx_ARR)

- **مؤقتات الطاقة المنخفضة Low Power Timers(LPTIM):**

المؤقتات في هذه المجموعة متعلقة بتطبيقات الا ستهلاك المنخفض للطاقة (Low power consumption)، حيث تحافظ على عملها في كل أو ضاع الطاقة المختلفة ماعد و ضع الا ستعداد Standby Mode ، مع ميزة العمل حتى بدون مصدر ساعة داخلي ، يمكن أن تستخدم هذه المؤقتات ك - pulse counter ، أيضاً لها القدرة على إيقاظ wake-up المتحكم في أوضاع توفير الطاقة، ولها المخطط الصندوقي التالي:



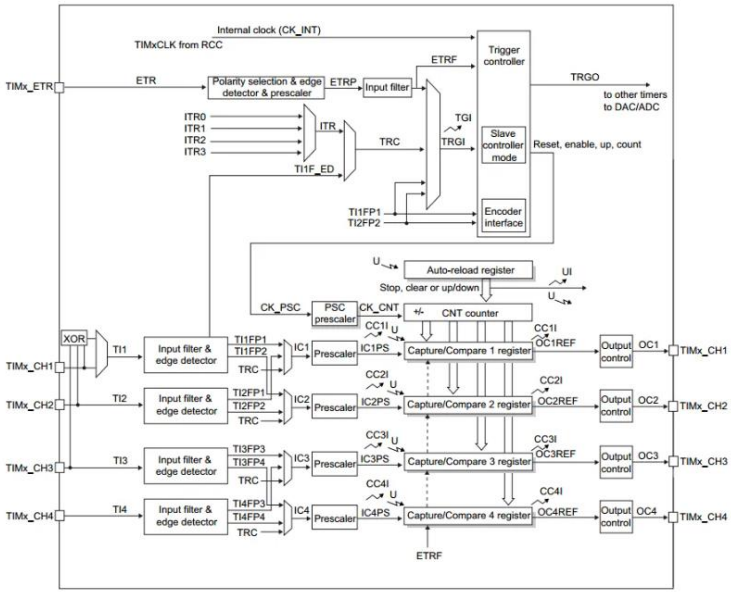
الشكل(4):المخطط الصندوقي للمؤقتات من نوع LPTIM

للمؤقتات من نوع LPTIM المواصفات التالية:

- عداد بطول 16 بت
- مقسم ترددي بطول 3-بت حيث يمكن الاختيار بين القيم التالية للتقسيم (1,2,4,8,16,32,64,128)
- يمكن اختيار مصدر الساعة للمؤقت :
  الساعة الداخلية APB, HSI16 ,LSE, LSI
  الساعة الخارجية من خلال مدخل المؤقت LPTIM تستخدم من أجل تطبيقات Pulse Counter
- مسجل ARR auto-reload register بطول 16 بت
- مسجل مقارنة بطول 16 بت
- يمكن التحكم بشكل إشارة الخرج إما Pulse أو PWM
- يمكن أن يعمل بنمط Encoder mode

- **مؤقتات الأغراض العامة General Purpose Timers:**

المؤقتات في هذه المجموعة تكون إما 16 أو 32 بت(بناءً على عائلة STM32)، ويتم من خلاله تنفيذ المهام الكلاسيكية التي يتم تنفيذها من أي مؤقت في متحكمات الأنظمة المدمجة Modern Embedded Microcontrollers وهي تستخدم كـ Output compare (أي إخراج إشارة عند بلوغ زمن معين)، أيضاً يمكن أن تعمل كـ OnePulse mode و Input Capture (لقياس تردد إشارة خارجية)، أيضاً يستخدم للتعامل مع الحساسات المختلفة (Encoder, HALL sensor)، كما يمكن لمؤقت الأغراض العامة أن يعمل كمؤقت أساسي Basic Timer ، وله المخطط الصندوقي التالي:



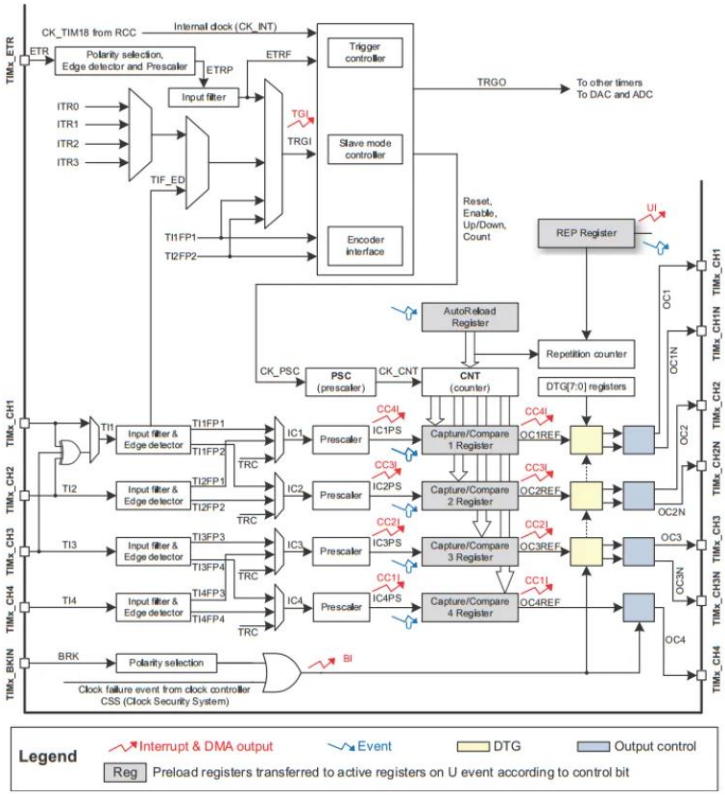
الشكل(5):المخطط الصندوقي لمؤقتات الأغراض العامة General Purpose Timers

لمؤقتات الأغراض العامة المزايا التالية:

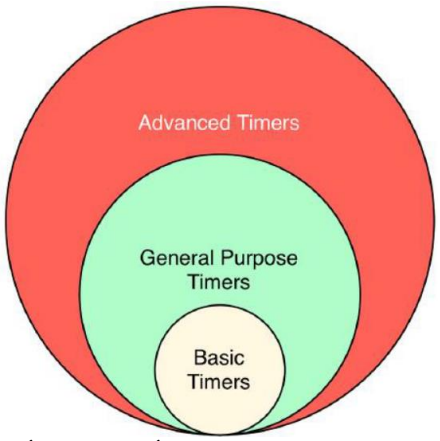
- عداد تصاعدي/ تنازلي بطول 16 بت قابل لإعادة التحميل تلقائياً auto-reload counter
- مقسم جهد بطول 16 بت يستخدم لتقسيم تردد الساعة للمتحكم على أي عدد يتراوح بين 1 و 65535
- له أربع مداخل/ مخارج منطقية قابلة للبرمجة بأحد الوظائف التالية:
  - Input Capture
  - Output Compare
  - توليد نبضات PWM بنمطي Edge و Center-aligned
  - One-Pulse mode Output
- دارة مزامنة للتحكم بالمؤقت من خلال إشارات خارجية ولربط عدة مؤقتات ببعضها
- توليد interrupt/DMA عند الأحداث التالية:
  - Update: وتشمل حدوث Overflow/Underflow للعداد، تهيئة العداد من خلال الـ software أو من خلال قدح داخلي أو خارجي.
  - Trigger event: بدء العداد، توقف العداد، تهيئة العداد أو العد من خلال قدح داخلي/خارجي
  - Input Capture
  - Output Compare
- يدعم المشفر التصاعدي incremental quadrate encoder وحساس hall sensor لأغراض تحديد الموضع positionality purpose
- له مدخل خاص يستخدم لقدح المؤقت من خلال ساعة خارجية

- **المؤقتات المتقدمة Advanced Timers:**

المؤقتات في هذه المجموعة هي الأكثر اكتمالاً من بين متحكمات STM32 ، حيث لها نفس المزايا والمواصفات الموجودة في مؤقتات الأغراض العامة بالإضافة إلى العديد من الميزات الأخرى المستخدمة في تطبيقات التحكم بالمحركات Motor control وتطبيقات تحويل الطاقة Power Conversion Application ، أي ضاً لها ثلاثة من الإشارات التكميلية Complimentary signals مع خاصية الـ dead time - ووجود مدخل لوضع التوقف للطوارئ Emergency shut-down input، ولها المخطط الصندوقي التالي:



الشكل(6):المخطط الصندوقي للمؤقتات المتقدمة



الشكل(7): العلاقة بين أشهر ثلاث أنواع للمؤقتات

- **المؤقتات عالية الدقة High Resolution Timers (HRTIM):**

المؤقتات في هذه المجموعة هي مؤقتات مخصصة توجد في عائلات معينة من متحكمات STM32 كمتحكمات STM32F3 ،فهذه المؤقتات لديها القدرة على توليد إ شارات رقمية بدقة عالية جداً مثل PWM أو Phase-shift pulse وهي تتكون من 6 مؤقتات فرعية واحد منها Master و5 الآخرين Slaves بمجمل 10 مخارج بدقة عالية High-Output Resolution، هذه المؤقتات مصممة بشكل أساسي لقيادة أنظمة تبديل الطاقة على سبيل المثال التبديل بين منابع التغذية switch-mode power supplies أو أنظمة الإ ضاءة lighting systems ، حيث تتجاوز دقة هذا المؤقت 217pSec، للمؤقت عالي الدقة المزايا والمواصفات تتضمن:

1. دقة تصل إلى 217 ps ثابتة حتى عند تغير الجهد والحرارة

2. الدقة العالية محققة عند جميع مخارج المؤقت مع إمكانية ضبط دورة التشغيل duty cycle والتردد عند العمل بنمط one-pulse mode
3. 6 وحدات توقيت(مؤقتات) بطول 16 بت لكل منها عداد منفصل و4 مقارنات
4. 10 مخارج يمكن التحكم بها من خلال أي وحدة توقيت من وحدات التوقيت الستة لمؤقت عالي الدقة
5. عدة وصلات تربط المؤقت عالي الدقة بطرفيات المبدلات الموجودة في المتحكم منها:
   - 4Triggers للمبدل التشابهي الرقمي ADC
   - 3 Triggers للمبدل الرقيم التشابهي DAC
   - 3 مقارنات لملاءمة الإشارة التشابهية
6. 7 أشعة مقاطعة لكل منها ما يصل إلى 14 مصدر مختلف
7. 6 DMA requests ، مع ما يصل إلى 14 مصدر .

Timer Type	Counter resolution	Counter type	DMA	Channels	Complimentary channels	Synchronization	
						Master	Slave
Advanced	16-bit	up, down and center aligned	Yes	4	3	Yes	Yes
General purpose	16/32-bit	up, down and center aligned	Yes	4	0	Yes	Yes
Basic	16-bit	up	Yes	0	0	Yes	No
1-channel	16-bit	up	No	1	0	Yes (OC signal)	No
2-channels	16-bit	up	No	2	0	Yes	Yes
1-channel with one complementary output	16-bit	up	Yes	1	1	Yes (OC signal)	No
2-channel with one complementary output	16-bit	up	Yes	2	1	Yes	Yes
High-resolution	16-bit	up	Yes	10	10	Yes	Yes
Low-power	16-bit	up	No	1	0	No	No

جدول (1): يوضح الخصائص المميزة لكل مجموعة من مجموعات المؤقتات

يوضح الجدول التالي أرقام وأنواع المؤقتات المتوفرة في متحكمات STM32G0 :

Feature		TIM1 (Advanced Control)	TIM2 (General-Purpose)	TIM3 (General-Purpose)	TIM6 (Basic)	TIM7 (Basic)	TIM14 (General-Purpose)	TIM15 (General-Purpose)	TIM16 (General-Purpose)	TIM17 (General-Purpose)
Clock source		CK_INT External input pin External trigger input ETR	CK_INT External input pin External trigger input ETR Internal trigger inputs		CK_INT		CK_INT	CK_INT External input pin Internal trigger inputs	CK_INT External input pin	
Resolution		16-bit	**32-bit**	16-bit	16-bit		16-bit	16-bit		
Prescaler		16-bit								
Counter direction		Up, Down, Up&Down	Up, Down, Up&Down		Up		Up	Up		
Repetition counter		✓	-		-		-	✓		
Synchronization	Master	✓	✓		✓		-	✓		
	Slave	✓	✓		-		-	✓	-	
Number of channels		6: ➢ CH1/CH1N ➢ CH2/CH2N ➢ CH3/CH3N ➢ CH4 ➢ CH5 and CH6 output only, not available externally	4: ➢ CH1 ➢ CH2 ➢ CH3 ➢ CH4		0		1: ➢ CH1	2: ➢ CH1/CH1N ➢ CH2	1: ➢ CH1/CH1N	
Trigger input		✓		✓						

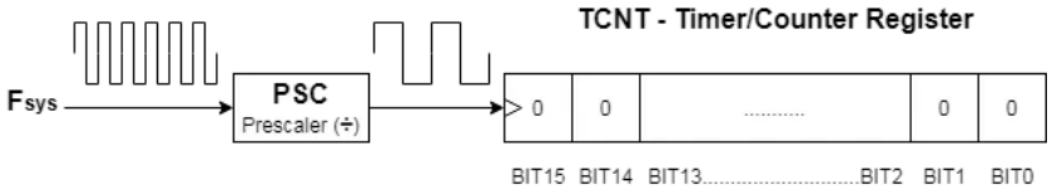
الجدول(2):أرقام وأنواع المؤقتات المتوفرة في متحكمات STM32G0

4- أنماط العمل المختلفة للمؤقتات في متحكمات STM32:

للمؤقتات في متحكمات STM32 أنماط عمل مختلفة سنذكر منها عدة أنماط، ولكن جدير بالذكر أن ليس كل أنواع المؤقتات بإمكانها العمل بجميع هذه الأنماط:

1. نمط المؤقت Timer Mode:

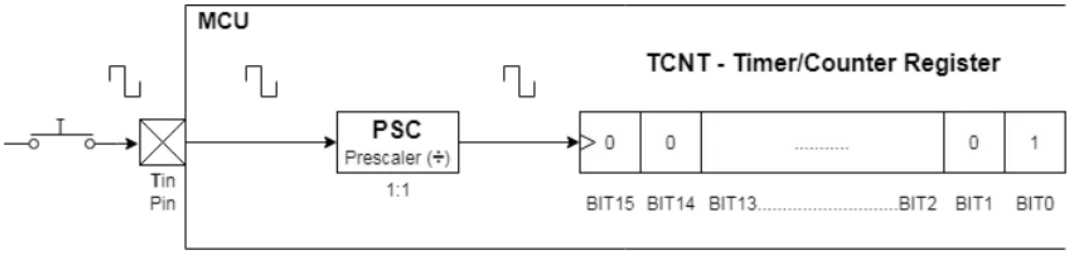
في هذا النمط من العمل فإن المؤقت يحصل على نبضات الساعة من ساعة المتحكم وباعتبار أن تردد ساعة المتحكم معروف بالتالي يمكن حساب زمن طفحان المؤقت كما يمكن التحكم بزمن الطفحان من خلال مسجل preload register من أجل الحصول على أي زمن مراد، وعند حدوث الطفحان تحدث مقاطعة الطفحان، هذا النمط من العمل يستخدم عادةً من أجل جدولة ومزامنة الأعمال والمهام المطلوبة من المتحكم خلال أزمنة معينة لكل مهمة من المهام ، كما يمكن استخدامه لاستبدال دالة التأخير الزمني الـ Delay() - مما يؤدي إلى رفع مستوى الأداء للمعالج.



الشكل(8):Timer Mode

2. نمط العداد Counter Mode:

في هذا النمط من العمل فإن نبضات الـ Timer - تأتي من مصدر خارجي (قطب دخل المؤقت timer input pin)، لذا فإن المؤقت يقوم بالعد التصاعدي أو التنازلي مع كل جبهة صاعدة/هابطة لإشارة الدخل، هذا النمط من العمل مفيد عندما تحتاج إلى وجود عداد رقمي دون الحاجة للقراءة الدورية لحالة اقطاب الدخل GPIO للمتحكم أو حتى حدوث مقاطعة في كل مرة في حال استخدام أحد أقطاب المقاطعة الخارجية EXTI

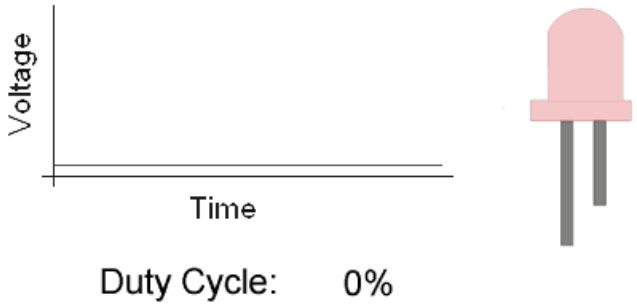


الشكل(9):Counter mode

3. نمط تعديل عرض النبضة PWM Mode:

في هذا النمط من العمل فإن نبضات الـ ساعة للـ Timer - تأتي داخلياً من ساعة المتحكم ، حيث يقوم الـ Timer بتوليد إشارة رقمية PWM على قطب الخرج للمؤقت من خلال استخدام مسجلات المقارنة OCR، حيث تتم مقارنة القيمة الحالية للمؤقت مع القيمة الموجودة في مسجل المقارنة OCR وعندما تتساوى القيم يتم عكس الحالة المنطقية لقطب الخرج حتى نهاية الدور ثم تعاد العملية مرة أخرى، حيث يمكن التحكم بتردد

نبضات الـ PWM وأيضاً دورة التشغيل duty cycle برمجياً من خلال المسجلات المناسبة، وتتعلق دقة الـ PWM بتردد الإشارة أي F_{PWM} وعوامل أخرى سنتكلم عنها بالتفصيل.



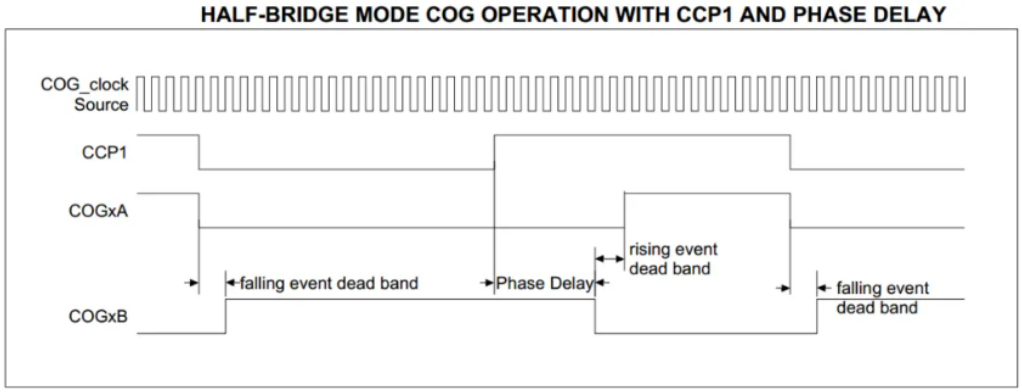
الشكل(10): PWM Mode

4. **نمط تعديل عرض النبضة المتقدم Advanced PWM Mode:**

هذا النمط من العمل يتيح التحكم بعدد أكبر من البارامترات وأي ضاً بتـ صميم الدارة المـ سؤولة عن توليد النبضات لإضافة مزايا متقدمة لإشارة الـ PWM وتتضمن:

- القدرة على توليد إشارات PWM مكملة وهي نفس إشارة الـ PWM المتولدة ولكن معكوسة منطقياً
- القدرة على إضافة زمن ميت dead-time لإشارة الـ PWM - لتطبيقات قيادة المحرك لتجنب ارتفاع التيار الكبير كنتيجة لـ PWM signals overlapping.
- القدرة على عمل إيقاف آلي auto-shutdown لإشارة الـ PWM والتي تعتبر ميزة مهمة جداً في تطبيقات safety-critical.

في الـ شكل التالي مثال لإ شارة PWM مع الإ شارة المكملة لها ومع حقن زمن ميت dead-band وأي ضاً phase delay:

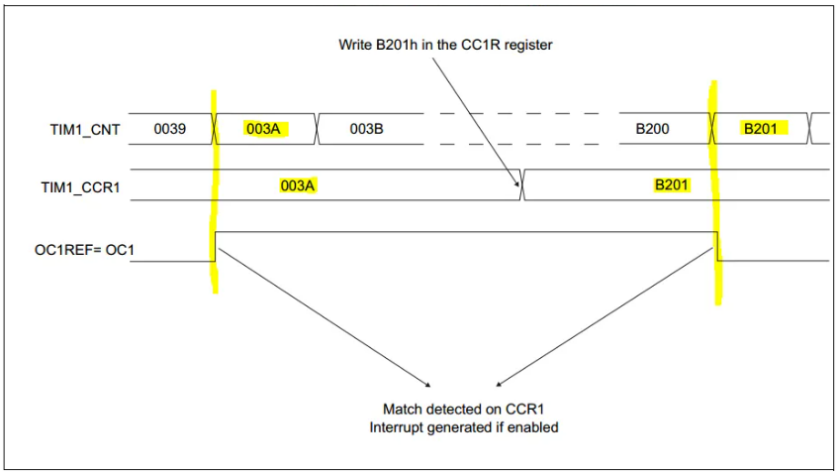


الشكل(11):Advanced PWM Mode

5. **نمط Output Compare:**

هذا النمط من العمل يسمح بالتحكم بخرج معين خلال زمن معين ، فعند تتساوى القيمة الحالية للعداد مع القيمة المخزنة في مسجل المقارنة OCR يتم تغيير حالة الخرج الرقمي وفقاً لما تمت برمجته خلال الكود فمثلاً قد يصبح وضع HIGH أو LOW أو toggles أو حتى يبقى بنفس وضعه دون تغيير.

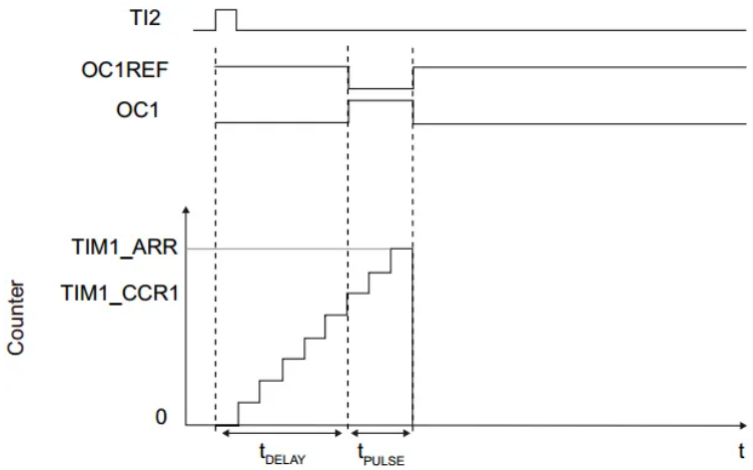
يوضح الشكل التالي مثال على نمط Output compare حيث المطلوب عمل toggle لقطب الخرج عند زمن معين:



الشكل(12):مثال على نمط Output compare

6. نمط One Pulse Mode:

يعتبر هذا النمط من العمل حالة خاصة من الأنماط السابقة، حيث يسمح للعداد بالاستجابة لحدث معين وتوليد نبضة بعرض وتأخير زمني معين يتم تحديدهم من خلال الكود، توليد النبضة يتم بنمط PWM أو Output compare ،كما يتم توليدها فقط عندما تكون قيمة المقارنة مختلفة عن القيمة الابتدائية للعداد، حيث يجب عند ضبط الإعدادات أن تكون $CNT<CCRx \leq ARR$ وبالأخص يجب أن تكون $0<CCRx$، على سبيل المثال إذا أردنا توليد نبضة على الخرج OC1 وبعرض t_{PULSE} وبتأخير زمني t_{DELAY} عند الجبهة الصاعدة للإشارة على القطب TI2 :



الشكل(13):One Pulse Mode

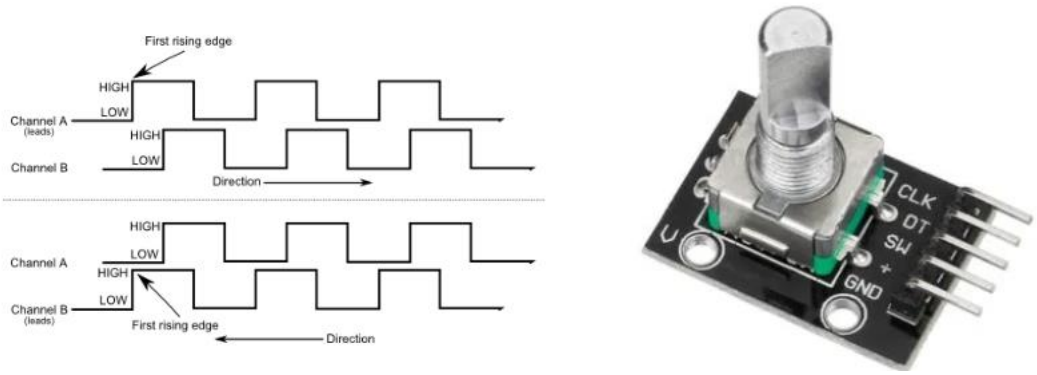
7. نمط Input Capture Mode:

في هذا النمط تستخدم المسجلات (TIMx_CCRx) لمسك القيمة التي وصل إليها العداد بعد الانتهاء من قياس الإشارة القادمة ICx ، فعند حدوث عملية الـ Capture - يتم رفع العلم CCXIF والموجود ضمن المسجل TIMx_SR ويتم طلب مقاطعة أو DMA في حال كانت إحداهما مفعلة.

يعتبر هذا النمط مهم بشكلٍ خاص عند الحاجة لقياس إشارة خارجية أو عند الحاجة لحساب زمن حدث خارجي ما ، وفي تطبيقات القياس بشكلٍ عام ، مثال عن استخدام هذا النمط مع حساس الـ Ultrasonic - والذي يقيس المسافة ويرسل المعلومات على شكل نبضة للمتحكم ومن خلال حساب زمن (عرض)النبضة يمكن حساب المسافة.

8. نمط Encoder Mode:

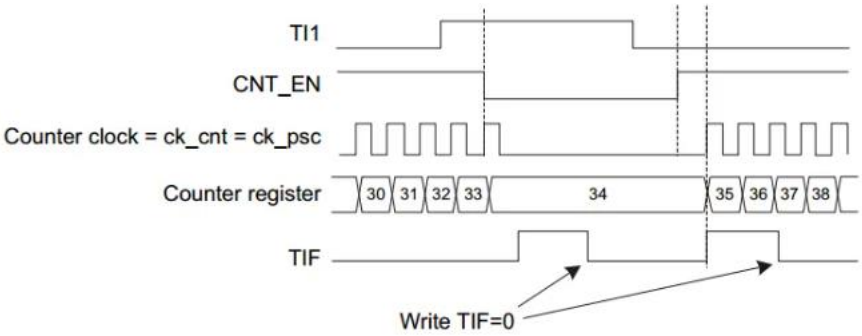
في هذا النمط من العمل، يعمل المؤقت كعداد رقمي لمدخلين، حيث من خلال تتابع الإشارات على هذين المدخلين يتم تحديد اتجاه العد تصاعدي/تنازلي بالإضافة إلى عد النبضات القادمة على هذين الدخلين، حيث لا يمكن عد النبضات بشكل منفصل على أحد المدخلين، أيضاً يمكن للمستخدم الحصول على معلومات ديناميكية كالسرعة والتسارع من خلال استخدام مؤقت في نمط Encoder ومؤقت آخر في نمط Capture mode



الشكل (14): المؤقت في نمط Encoder

9. نمط Timer Gate Mode (Slave Mode):

يدعى هذا النمط أيضاً Slave Mode، حيث يبدأ العداد في المؤقت بعد النبضات الداخلية للساعة ويستمر بالعد طالما قطب الدخل TI1 للمؤقت بحالة LOW ويتوقف عن العد عندما يصبح قطب الدخل بحالة HIGH، وفي هذه الحالة فإن قطب الدخل للمؤقت هو الذي يسمح للعداد بالعد أي يكون بمثابة Timer Gate، وحيث يتم رفع العلم TIF في المسجل TIMx_SR عند بدأ العداد بالعد وعند توقفه، حيث يتعلق التأخير الزمني بين الجبهة الصاعدة و التوقف الفعلي للعداد بدارة resynchronization على قطب الدخل TI1، هذا النمط له مجال واسع من التطبيقات وقياس الإشارات بالأخص الإشارات ذات النبضات القصيرة نسبياً وبدقة عالية جداً، كما يمكن جعل هذا المؤقت يبدأ بالعد تبعاً لأحداث معينة خارجية قادمة من حساسات أو من متحكمات أخرى.



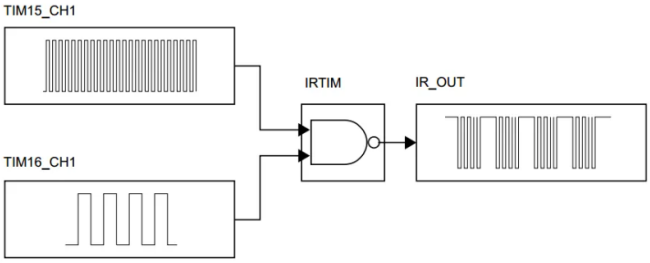
الشكل(15): دارة التحكم في نمط Gate mode

10.نمط Timer DMA Burst Mode:

ليـ ست كل المؤقتات في متحكمات STM32 لها القدرة على توليد عدة طلبات DMA بناءً على حدث وحيد، فالغاية الرئيـ سية من توليد عدة طلبات DMA هو القدرة على إعادة برمجة المؤقت عدة مرات دون تدخل الكود في ذلك.

11.نمط IRTIM Infrared Mode:

يمكن ا ستخدام هذا النمط في تطبيقات التحكم عن بعد remote control حيث يمكن ا ستخدامه مع مر سل الأ شعة تحت الحمراء، حيث يتم الا ستفادة من الاتـ صال الداخلي بين المؤقتين TIM16 , TIM15 وتوليد إشارة التحكم على القطب IR_TIM كما هو موضح بالشكل التالي



الشكل(16):نمط IRTIM

يوضح الجدول التالي أنماط المؤقتات المتوفرة في متحكمات STM32G0:

Feature	TIM1 (Advanced Control)	TIM2	TIM3	TIM6	TIM7	TIM14	TIM15	TIM16	TIM17
Input capture mode	✓	✓		-		✓		✓	
PWM input mode	✓	✓		-			✓	-	
Forced output mode	✓	✓		-		✓		✓	
Output compare mode	✓	✓		-		✓		✓	
PWM	Standard Asymmetric Combined Combined 3-phase 6-step PWM	Standard Asymmetric Combined		-		Standard	Standard Asymmetric Combined	Standard	
Programmable dead-time	✓ (CH1-3)	-		-		-	✓ (CH1)	-	
Break inputs	2 bidirectional	0		0		0	1 bidirectional		
One-Pulse Mode	✓	✓		-		✓		✓	
Retriggerable one pulse mode	✓	✓		-		-	✓	-	
Encoder interface mode	✓	✓		-		-		-	
Timer input XOR function	✓	-		-		-	✓	-	
DMA	✓	✓		✓		-		✓	

الجدول(3): أنماط المؤقتات المتوفرة في متحكمات STM32G0

5- Time-Base Unit :

يتألف الـ Timer في متحكمات STM32 بشكل أساسي من عداد 16 بت (TIMx_CNT) ومسجل إعادة التحميل التلقائي (TIMx_ARR)، حيث ممكن للعداد أن يعد بشكل تصاعدي أو تنازلي أو الاثنان معاً، بالإضافة إلى مسجل المقسم الترددي Prescaler (TIMx_PSC) وهذه المسجلات يمكن قراءتها من خلال الكود أو حتى أثناء عمل العداد.

6- المقاطعات المتوفرة في المؤقتات في متحكمات STM32 :

يمكن للمؤقتات ذات الأغراض العامة General-Purpose توليد مقاطعات/DMA عند الأحداث التالية:

- Update: عند حالتي counter overflow/underflow أو عند تهيئة العداد counter initialization سواء من قبل الكود أو من خلال القدح الداخلي أو الخارجي للمؤقت.
- Trigger event: عند بدء أو توقف أو تهيئة العداد أو العد من خلال قدح داخلي أو خارجي
- Input Capture
- Output compare

7- ضبط الإعدادات في المؤقتات في متحكمات STM32 :

كما ذكرنا سابقاً فإن المؤقت الأ سا سي Basic Timer هو عبارة عن عداد بإمكانه العد بـ شكل تـ صاعدي فقط، حيث يقوم بالعد من الصفر إلى القيمة المحددة في حقل الـ Period(*Preload*) أثناء تهيئة المؤقت ، وأكبر قيمة يمكن أن يـ صل إليها تحدد حـ سب طول المؤقت، حيث المؤقت 16 بت يمكنه العد إلى 0xffff والمؤقت ذو 32 بت يمكنه العد إلى 0xffff ffff ، حيث يعتمد تردد (سرعة العد) على سرعة الناقل المتصل به المؤقت بالإضافة إلى المقسم الترددي Prescaler حيث يتم تقـ سيم تردد ساعة المؤقت على واحدة من القيم المتاحة وهي من 1 حتى 65535 (حيث أن مسجل الـ Prescaler بطول 16بت)، وعندما يصل العداد إلى القيمة المحددة Period(*Preload*) يحدث ما يسمى بالطفحان overflow أي يقوم بالتـ صفير والعد مرة أخرى من الـ صفر و يتم رفع العلم الخاص بالطفحان Update Event(UEV).

أي أن المسجلات الخاصة بالـ Period ($Preload$) و Prescaler هي التي تحدد تردد المؤقت أي الزمن الذي سيستغرقه المؤقت حتى يحدث حدث الطفحان Overflow وعندها يتم رفع العلم UEV ، ويتم اختيار القيم المناسبة لهذه المسجلات بناءً على هذه المعادلة:

$$T_{out} = \frac{Prescaler \times Preload}{F_{CLK}}$$

على سبيل المثال ، لنفترض أن تردد الساعة للمتحكم مضبوط على 48MHz وقيمة الـ Prescaler تساوي 48000 والـ period تساوي 500 سيحدث overflow للمؤقت كل:

$$T_{out} = \frac{48000 \times 500}{48000000} = 0.5sec$$

يمكن أيضاً استخدام المعادلة التالية لحساب تردد عمل المؤقت:

$$UpdateEvent = \frac{Timer_{clock}}{(Prescaler + 1)(Period + 1)}$$

هذا بالنسبة للمؤقتات الأساسية، أما بالنسبة للمؤقتات المتقدمة Advanced Timer فهناك حقل إضافي في المعادلة السابقة يدعى RepetitionCounter يعطي عدد مرات الـ Overflow/Underflow ، وهو يستخدم لزيادة الزمن الممكن الحصول عليه من المؤقت، حيث تصبح المعادلة بالشكل التالي:

$$UpdateEvent = \frac{Timer_{clock}}{(Prescaler + 1)(Period + 1)(\text{RepetitionCounter } + 1)}$$

فإذا اخترنا عدد RepetitionCounter 5 عندها سيصبح الزمن الممكن الحصول عليه من المؤقت من أجل نفس المعطيات السابقة:

$$UpdateEvent = \frac{48.000.000}{(47999 + 1)(499 + 1)(5 + 1)} = 0.33Hz = 3s$$

هذه الخاصية موجودة فقط في المؤقتات المتقدمة ، وفي حال تم ترك القيمة الافتراضية للـ repetionCounter صفر ، سيعمل المؤقت المتقدم كمؤقت أساسي.

نستعرض في هذا الجدول أهم المسجلات الخاصة بالمؤقتات:

Description	Name
Control Register 1	TIMx_CR1
Control Register 2	TIMx_CR2
DMA/Interrupt Enable Register	TIMx_DIER
Status Register	TIMx_SR
Event Generation Register	TIMx_EGR
Capture/Compare Mode Register 1	TIMx_CCMR1
Capture/Compare Mode Register 2	TIMx_CCMR2
Capture/Compare Enable Register	TIMx_CCER
Counter	TIMx_CNT
Prescaler	TIMx_PSC
Auto-Reload Register	TIMx_ARR
Capture/Compare Register 1	TIMx_CCR1
Capture/Compare Register 2	TIMx_CCR2
Capture/Compare Register 3	TIMx_CCR3
Capture/Compare Register 4	TIMx_CCR4

الجدول(4):

نستعرض في هذا الجدول أهم البتات الخاصة بالمؤقتات

Reg	Bits	Name	Description
TIMx_CR1	11	UIFREMAP	UIF status bit remapping
	7	ARPE	Auto-reload preload enable
	3	OPM	One-pulse mode
	2	URS	Update request source
	1	UDIS	Update disable
	0	**CEN**	**Counter enable**
TIMx_CR2	6:4	MMS	Master mode selection
TIMx_DIER	8	UDE	Update DMA request enable
	4	CC4IE	Capture/Compare 4 interrupt enable
	3	CC3IE	Capture/Compare 4 interrupt enable
	2	CC2iE	Capture/Compare 4 interrupt enable
	1	CC1IE	Capture/Compare 4 interrupt enable
	0	**UIE**	**Update interrupt enable**
TIMx_SR	**0**	**UIF**	**Update interrupt flag**
TIMx_EGR	0	UG	Update generation

الجدول(5):

8- أوضاع الاستخدام المختلفة للمؤقتات في متحكمات STM32 :

هناك ثلاث أوضاع مختلفة لاستخدام المؤقتات هي:

1. وضع Polling: أي استخدام المؤقت بدون مقاطعة وفي هذه الحالة يجب فحص القيمة التي وصل إليها العداد بشكل يدوي داخل الكود بشكل مستمر أو يمكن بدلاً من ذلك فحص حالة العلم Flag أيضاً بشكل مستمر داخل الكود مما يؤدي إلى تعطيل العديد من وظائف المتحكم أو قد تتسبب في عدم الوصول إلى القيمة المحددة بالضبط، لذا فإننا لن نستخدم هذا الوضع ضمن تطبيقاتنا.
توفر مكتبة HAL الدالة التالية لبدء المؤقت:

```
HAL_TIM_Base_Start();
```

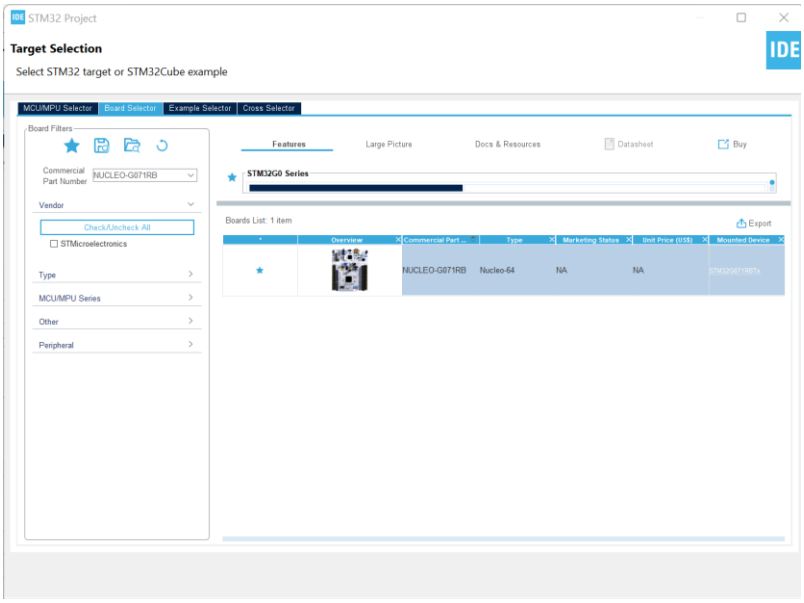
2. وضع الـ Interrupt:في هذا الوضع عند الوصول إلى Overflow/underflow أو أي من أحداث المقاطعة سيتم التوجه آلياً لتنفيذ برنامج خدمة المقاطعة، وهذا الوضع الذي سنستخدمه في جميع التطبيقات القادمة.
توفر مكتبة HAL الدالة التالية لبدء المؤقت في وضع المقاطعة:

```
HAL_TIM_Base_Start_IT();
```

3. وضع الـ DMA: سيكون لها بحث منفصل.

9- التطبيق العملي الأول: استخدام المؤقت في نمط Timer mode وبوضع المقاطعة لتوليد زمن بدلاً من استخدام دالة ()delay واستخدامه في عمل Toggle لليد الموصول على القطب PA5(الليد الموجود على اللوحة) :

سنقوم بضبط الإعدادات من خلال CubeMx المدمجة داخل بيئة STM32CubeIDE وفقاً للخطوات التالية:
الخطوة الأولى: قم بفتح برنامج STM32CubeIDE ومن ثم قم بإنشاء مشروع جديد من نافذة File ثم New ثم STM32Project ثم قم باختيار المتحكم المصغر أو من خلال اختيار اسم اللوحة المستخدمة وهي في حالتنا Nucleo-G071RB كما في الشكل التالي:



الشكل(17) :بدء مشروع جديد في بيئة STM32CubeIDE

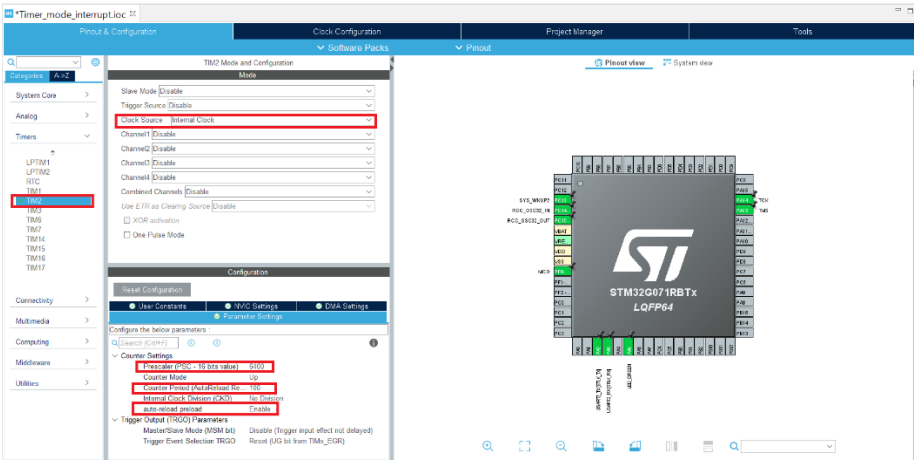
الخطوة الثانية: ضبط إعدادات المؤقت

كي نحصل على زمن 100msec لعكس حالة الليد الموصول على القطب رقم 5 من المنفذ A ، من المعادلة السابقة سنفترض أن تردد ساعة المتحكم هي 64MHz والمقسم الترددي 6400 بقي فقط حساب Period (Preload)، بتعويض القيم في المعادلة :

$$T_{out} = \frac{Prescaler \times Preload}{F_{CLK}} = \frac{6400 \times Preload}{64000000}$$

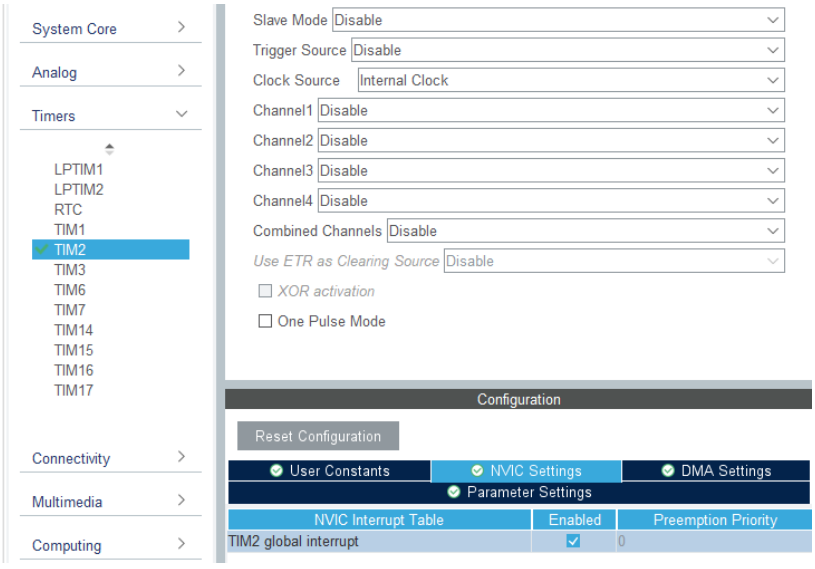
$$Preload = 1000$$

سنقوم باختيار مصدر الساعة للمؤقت داخلي، المقسم الترددي 6400، الـ Preload=1000، أيضاً سنقوم بتفعيل إعادة التحميل التلقائي، كما في الشكل التالي:



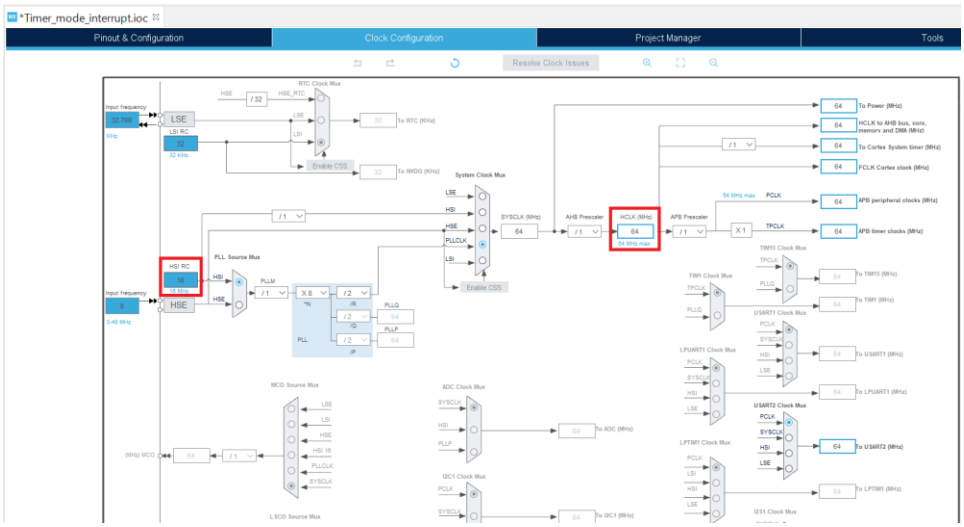
الشكل(18):ضبط إعدادات المؤقت

الخطوة الثالثة: تفعيل مقاطعة المؤقت من شريط الـ NVIC tab



الشكل(19):تفعيل مقاطعة المؤقت

الخطوة الرابعة: ضبط تردد ساعة المتحكم



الشكل(20):ضبط تردد ساعة المتحكم

الخطوة الخامسة: توليد الكود بناءً على الإعدادات التي تم ضبطها

```c
#include "main.h"

TIM_HandleTypeDef htim2;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM2_Init(void);

int main(void)
{

  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_TIM2_Init();

  while (1)
  {

  }

}
```

الشكل(21):توليد الكود بناءً على الإعدادات التي تم ضبطها

الخطوة السادسة: بدء المؤقت

على الرغم من ضبط إعدادات المؤقت فإنه سيبقى في حالة IDLE أي خمول ولن يبدأ بالعد حتى تقوم باستدعاء الدالة الخاصة ببدأ عمل المؤقت وهي :

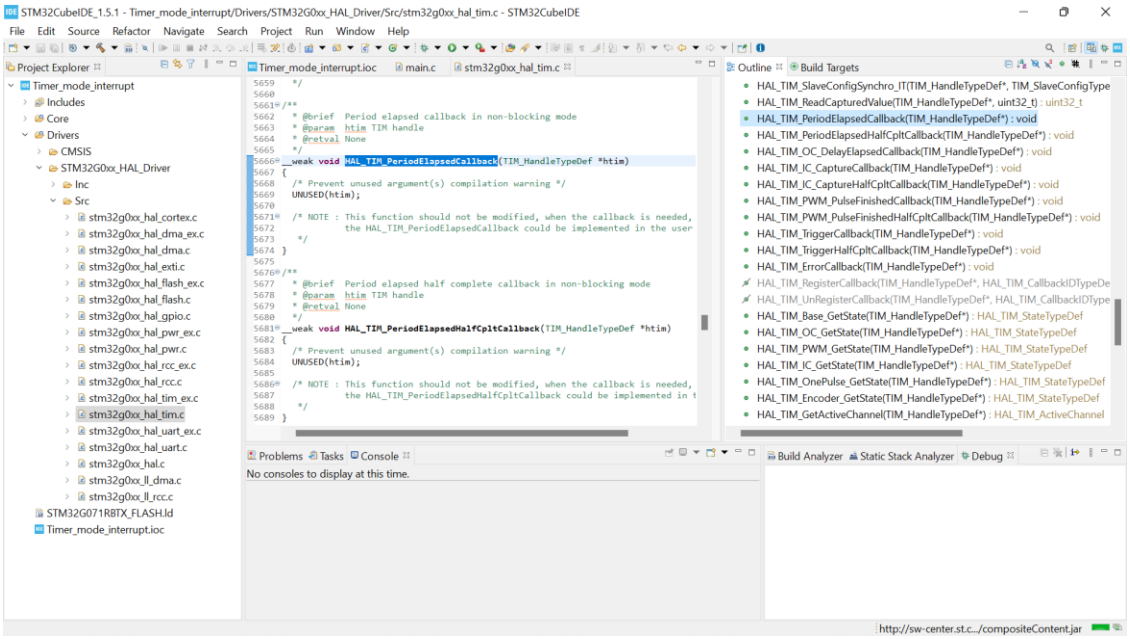
HAL_TIM_Base_Start_IT()

سنستدعي هذه الدالة في بداية الكود.

الخطوة السابعة: إضافة دالة خدمة مقاطعة الطفحان

عند حدوث طفحان للمؤقت بو صوله للقيمة التي تم ضبطها في counter Period ، يذهب المعالج تلقائياً لبرنامج خدمة المقاطعة الخاص بالطفحان والذي يكون معرف بـ شكل افترا ضي ك ـ _weak ضمن ملف الـ stm32g0xx_hal_tim.c الموجود ضمن مجلد الـ ـ Drivers ثم مجلد الـ ـ Src و يدعى

HAL_TIM_PeriodElapsedCallback()، حيث نقوم بنسخ اسمه وإضافته للبرنامج الرئيسي ثم نقوم ضمنه بعكس حالة الليد.



الشكل(22): نسخ دالة خدمة المقاطعة

يصبح الكود بالشكل التالي:

```
#include "main.h"

TIM_HandleTypeDef htim2;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM2_Init(void);
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_TIM2_Init();
  HAL_TIM_Base_Start_IT(&htim2);
  while (1)
  {

  }

}
void HAL_TIM_PeriodElapsedCallback( TIM_HandleTypeDef* htim)
{

        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);
}
```

الخطوة الثامنة: ترجمة الكود ورفعه للمتحكم ومراقبته من خلال فتح جلسة Debug

قم بالضغط على زر الـ Debug لترجمة الكود ورفعه للمتحكم من خلال الـDebug ، كما يمكنك بعد رفع الكود للمتحكم إغلاق جلسة الـ Debug وعمل Reset للمتحكم لبدء تنفيذ الكود الذي تم تحميله.

10- التطبيق العملي الثاني: استخدام المؤقت في نمط Counter mode وبوضع المقاطعة ليعمل كعداد رقمي بسيط :

كما ذكرنا سابقاً فإن المؤقت يمكن أن يعمل كعداد رقمي أي يستقبل نبضات الساعة الخاصة به من مصدر خارجي (قطب دخل) ويقوم بعدِها، ويمكنه أن يعمل بثلاث أنماط مختلفة هي:

1. **نمط العد التصاعدي Up-counting Mode**

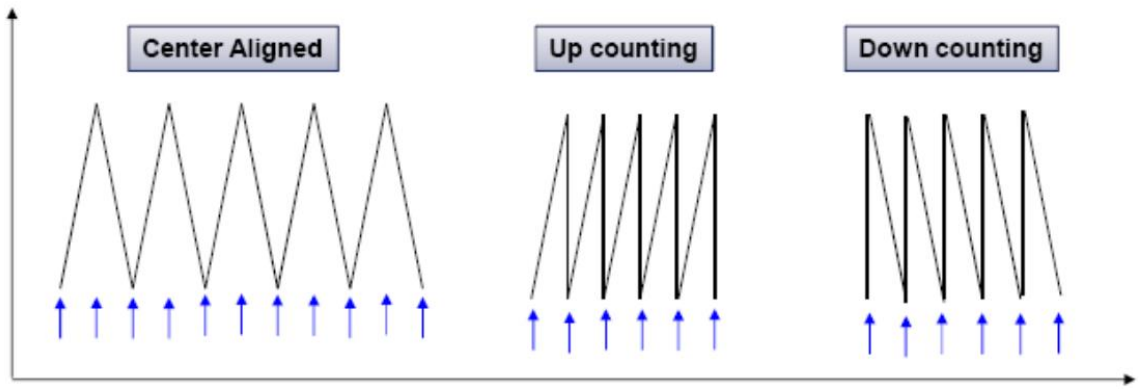
في هذا النمط فإن العداد يبدأ بالعد من الصفر مع كل نبضة قادمة على قطب الدخل ويستمر حتى يصل إلى القيمة المخزنة مسبقاً والموجودة في المسجل (TIMx_ARR) ، ثم يعود للقيمة صفر ويولد حدث الطفحان Overflow كمل يتم توليد حدث Update event مع كل طفحان للعداد أو من خلال وضع واحد منطقي في البت UG الموجود في المسجل EGR من خلال الكود أو من خلال استخدام نمط Slave Mode controller.

2. **نمط العد التنازلي Down-counting Mode**

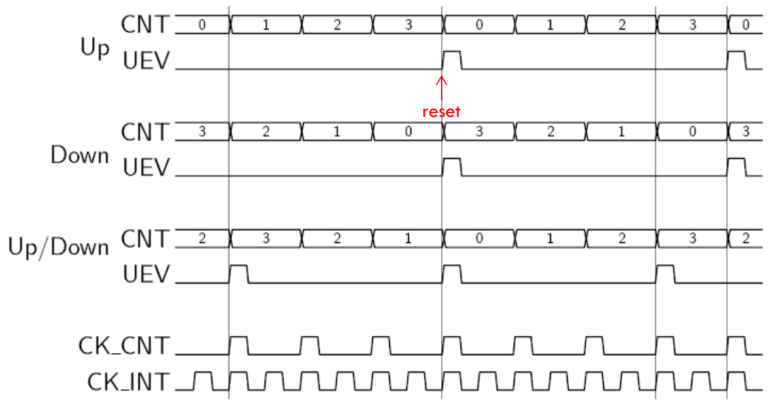
في هذا النمط فإن العداد يبدأ بالعد من القيمة المخزنة auto-reload value في المسجل TIMx-ARR مع كل نبضة قادمة على قطب الدخل ويستمر ليصل إلى الصفر ثم يعو ليبدأ من القيمة المخزنة سابقاً ويولد حدث الـ Underflow event أيضاً يتم توليد حدث الـ Update مع كل Underflow أو من خلال وضع واحد منطقي في البت UG الموجود في المسجل EGR من خلال الكود أو من خلال استخدام نمط Slave Mode controller.

3. **نمط العد التصاعدي تنازلي Center-Aligned Mode:**

في هذا النمط فإن العداد يبدأ بالعد التصاعدي من الصفر ويستمر بالعد مع كل نبضة قادمة على قطب الدخل حتى يصل إلى القيمة المخزنة سابقاً auto-reload value في المسجل TIMx-ARR ناقص واحد ، ثم يتم توليد حدث الطفحان Overflow event ثم يبدأ بالعد التنازلي من القيمة المخزنة سابقاً auto-reload value مع كل نبضة قادمة على قطب الدخل وحتى يصل إلى الصفر عندها يتم توليد حدث الـ Underflow ثم يعود للعد التصاعدي من الصفر وهكذا...، في هذا النمط فإن بت الاتجاه DIR في المسجل TIMx_CR1 غير قابل للكتابة أو تغيير قيمته فقط يتم تحديث قيمته من خلال الـ Hardware ويعبر عن الاتجاه الحالي للعد.



الشكل(23): أنماط العد المختلفة



Counter Modes (ARR=3, PSC=1)

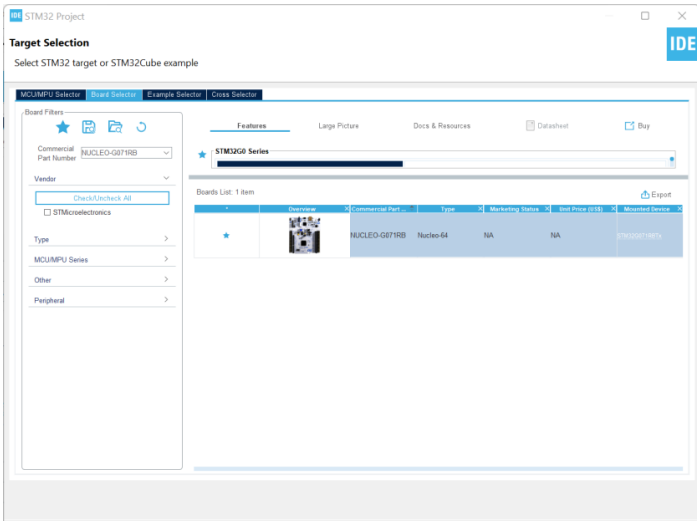
الشكل(24):المخطط الزمني لأنماط العد المختلفة

الخطوات التي سنقوم بها في تطبيقنا هي:

- ضبط إعدادات المؤقت TIM2 ليعمل في نمط العداد
- ضبط القيمة المخزنة سابقاً Preload value (Period) على القيمة 20 ، لذا فإن العداد سيطفح ويولد مقاطعة الطفحان بعد أن يعد 20 نبضة.
- ضبط قطب دخل المؤقت GPIO pin
- ضبط إعدادات المنفذ التسلسلي USART2 لطباعة النبضات التي يقوم العداد بعدها، لمراقبة عمل العداد.
- طباعة رسالة على المنفذ التسلسلي ضمن برنامج خدمة المقاطعة (خدمة مقاطعة الطفحان) بهدف التأكد من سير العملية بشكل صحيح.

سنقوم بضبط الإعدادات من خلال أداة CubeMx المدمجة داخل بيئة STM32CubeIDE وفقاً للخطوات التالية:

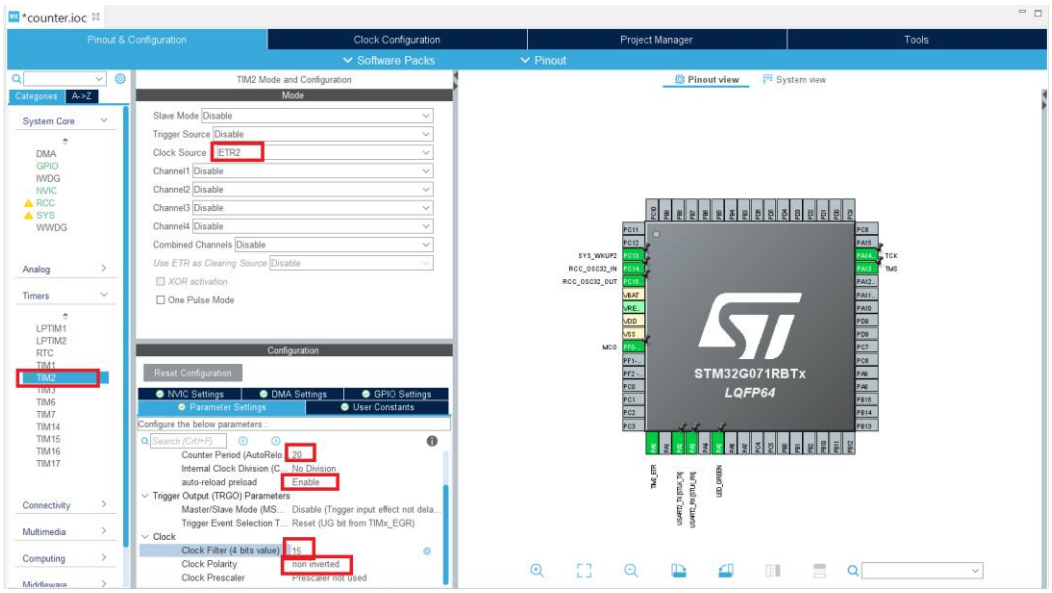
الخطوة الأولى: قم بفتح برنامج STM32CubeIDE ومن ثم قم بإنشاء مشروع جديد من نافذة File ثم New ثم STM32Project ثم قم باختيار المتحكم المصغر أو من خلال اختيار اسم اللوحة المستخدمة وهي في حالتنا Nucleo-G071RB كما في الشكل التالي:



الشكل(25) :بدء مشروع جديد في بيئة STM32CubeIDE

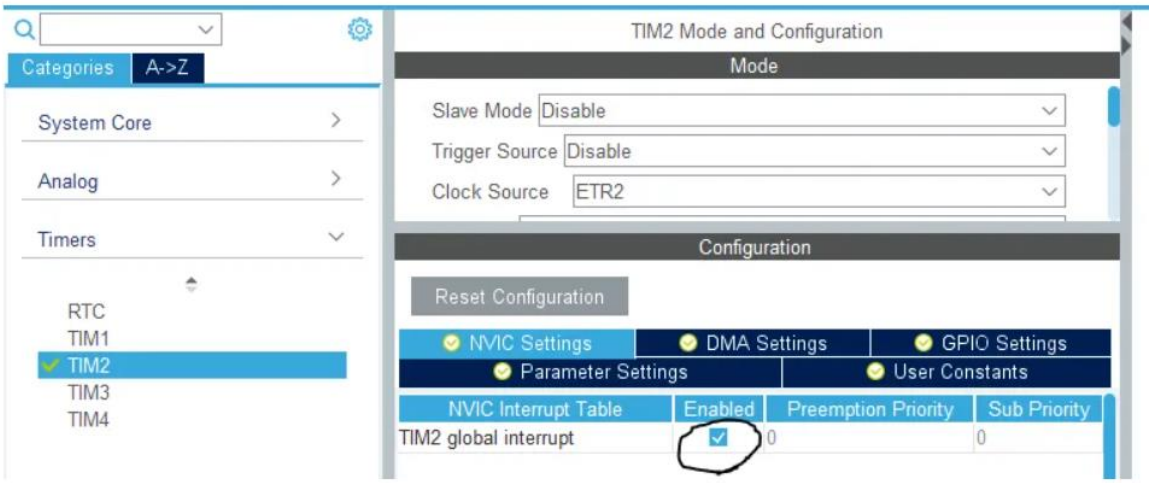
الخطوة الثانية: ضبط إعدادات المؤقت ليعمل كعداد رقمي

ضبط مـ صدر الـ ساعة للمؤقت كي يعمل كعداد رقمي (من خلال قطب الدخل للمؤقت) على المـ صدر الخارجي external pin وهو ETR2 والذي يوافق القطب PA0، سنـ ضبط القيمة المخزنة مـ سبقاً للعداد counter period على القيمة 20 ، عندها عندما يـصل العداد إلى القيمة 20 ستحدث مقاطعة الطفحان، سنقوم أيضاً بـضبط فلتر رقمي لقطب المؤقت لتجنب الـ ضجيج الناتج عن اهتزاز المفتاح Switch bouncing ، حيث يتراوح مجال القيم للفلتر بين 0 والـ 15، أخيراً يمكن ضبط الجبهة التي سيعد عندها العداد صاعدة أو هابطة، سنختار الجبهة الصاعدة .



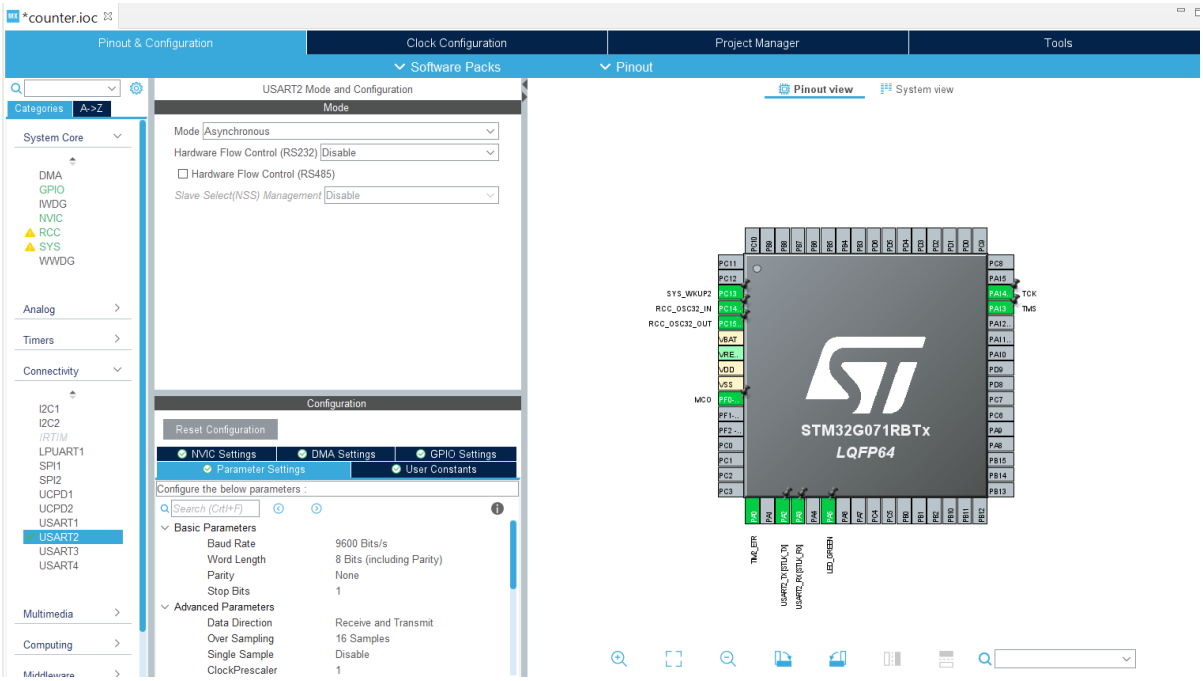
الشكل(26) ضبط إعدادات المؤقت T2

الخطوة الثالثة: تفعيل مقاطعة المؤقت من خلال قائمة NVIC



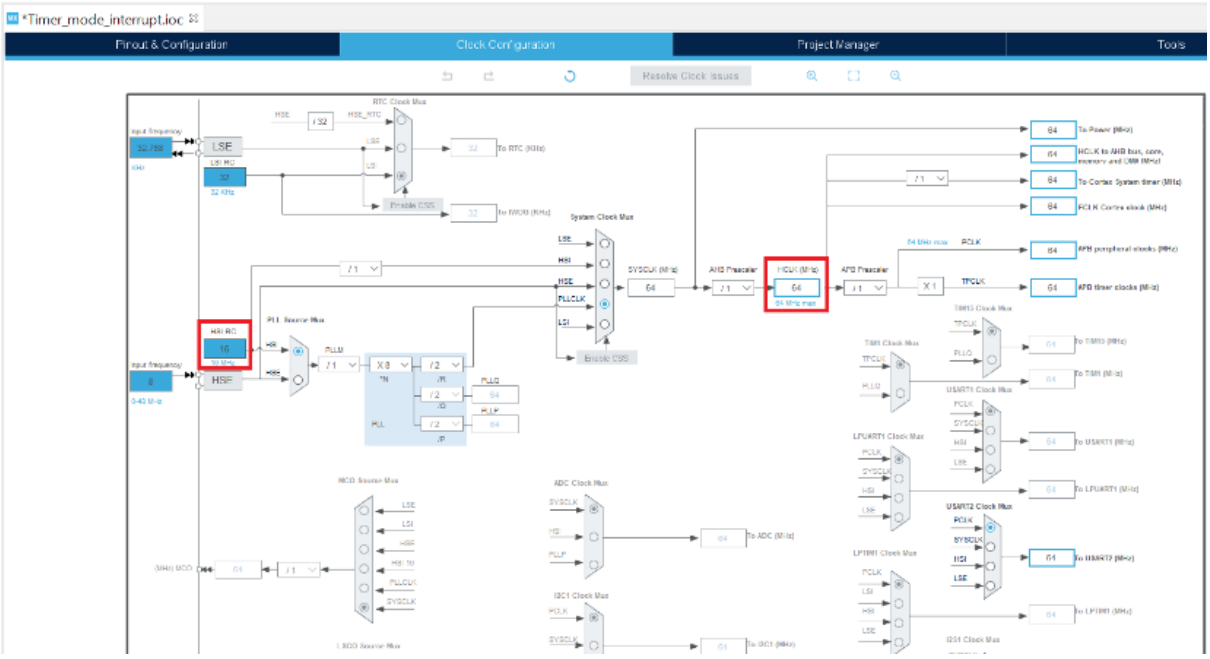
الشكل(27):تفعيل مقاطعة المؤقت

الخطوة الرابعة: ضبط إعدادات المنفذ التسلسلي الثاني USART2 للعمل في نمط Async وبمعدل نقل بيانات 9600bps



الشكل(28): ضبط إعدادات المنفذ التسلسلي الثاني

الخطوة الخامسة: ضبط تردد ساعة المتحكم



الشكل(29):ضبط تردد ساعة المتحكم

الخطوة السادسة: توليد الكود بناءً على الإعدادات التي تم ضبطها من خلال ctrl+s

```
#include "main.h"
TIM_HandleTypeDef htim2;
UART_HandleTypeDef huart2;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_TIM2_Init(void);
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_TIM2_Init();
   while (1)
  {
     }}
```

الخطوة السابعة: بدء المؤقت

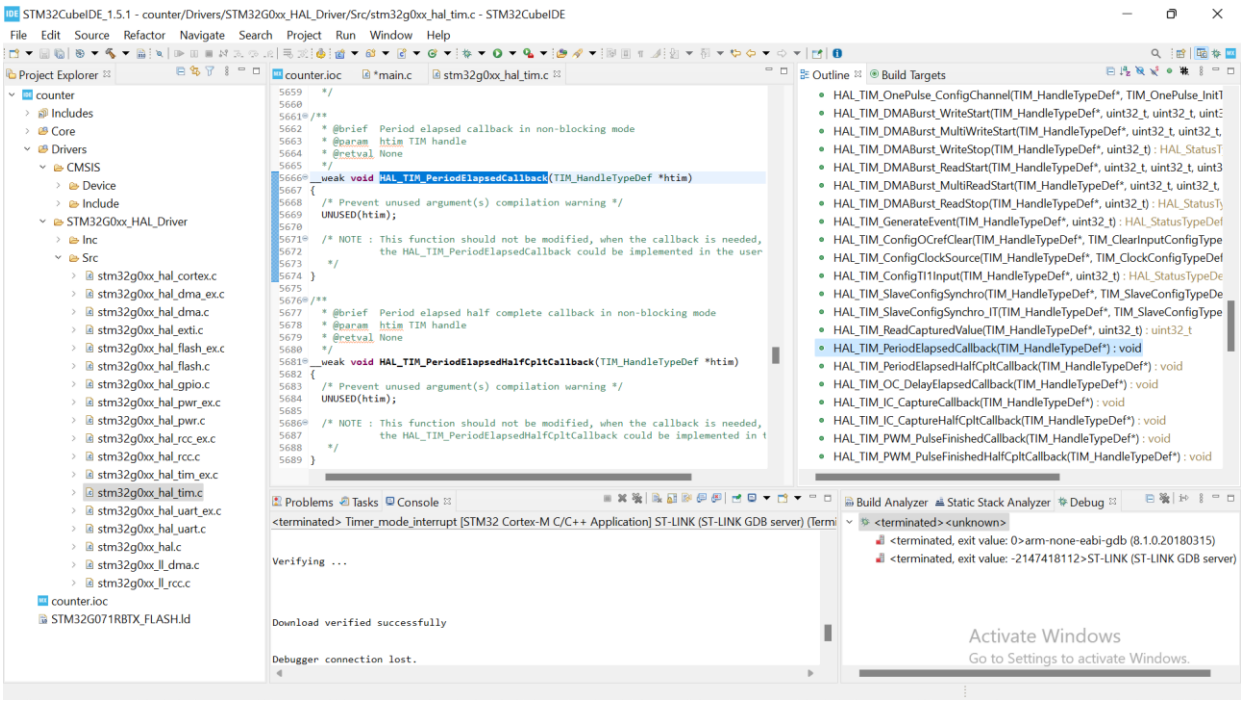
على الرغم من ضبط إعدادات المؤقت فإنه سيبقى في حالة IDLE أي خمول ولن يبدأ بالعد حتى تقوم باستدعاء الدالة الخاصة ببدأ عمل المؤقت وهي :

```
HAL_TIM_Base_Start_IT(&htim2);
```

سنستدعي هذه الدالة في بداية الكود.

الخطوة الثامنة: إضافة دالة خدمة مقاطعة الطفحان

عند حدوث طفحان للمؤقت بوصوله للقيمة التي تم ضبطها في counter Period ، يذهب المعالج تلقائياً لبرنامج خدمة المقاطعة الخاص بالطفحان والذي يكون معرف بشكل افتراضي كـ weak_ ضمن ملف الـ stm32g0xx_hal_tim.c الموجود ضمن مجلد الـ Drivers ثم مجلد الـ Src ويدعى HAL_TIM_PeriodElapsedCallback()، حيث نقوم بنسخ اسمه وإضافته للبرنامج الرئيسي ثم نقوم ضمنه بعكس حالة الليد.



الشكل(30): نسخ دالة خدمة المقاطعة

يصبح الكود بالشكل التالي:

```
#include "main.h"
uint8_t END_MSG[35] = "Overflow Reached! Counter Reset!\n\r";
TIM_HandleTypeDef htim2;
UART_HandleTypeDef huart2;
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_TIM2_Init(void);
int main(void)
{

  uint8_t MSG[20] = {'\0'};
  uint16_t CounterTicks = 0;

  HAL_Init();
  SystemClock_Config();
  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_TIM2_Init();
  HAL_TIM_Base_Start_IT(&htim2);

   while (1)
```

```
  {
        // Read The Counter Ticks Register
         CounterTicks = TIM2->CNT;
        // Print The Ticks Count Via UART1
         sprintf(MSG, "Ticks = %d\n\r", CounterTicks);
          HAL_UART_Transmit(&huart2, MSG, sizeof(MSG), 100);
          HAL_Delay(100);
  }

}
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
     HAL_UART_Transmit(&huart2, END_MSG, sizeof(END_MSG), 100);
                                                                    }
```

الخطوة التاسعة: ترجمة الكود ورفعه للمتحكم ومراقبته من خلال فتح جلسة Debug

قم بالضغط على زر الـ Debug لترجمة الكود ورفعه للمتحكم من خلال الـDebug ، كما يمكنك بعد رفع الكود للمتحكم إغلاق جلسة الـ Debug وعمل Reset للمتحكم لبدء تنفيذ الكود الذي تم تحميله.