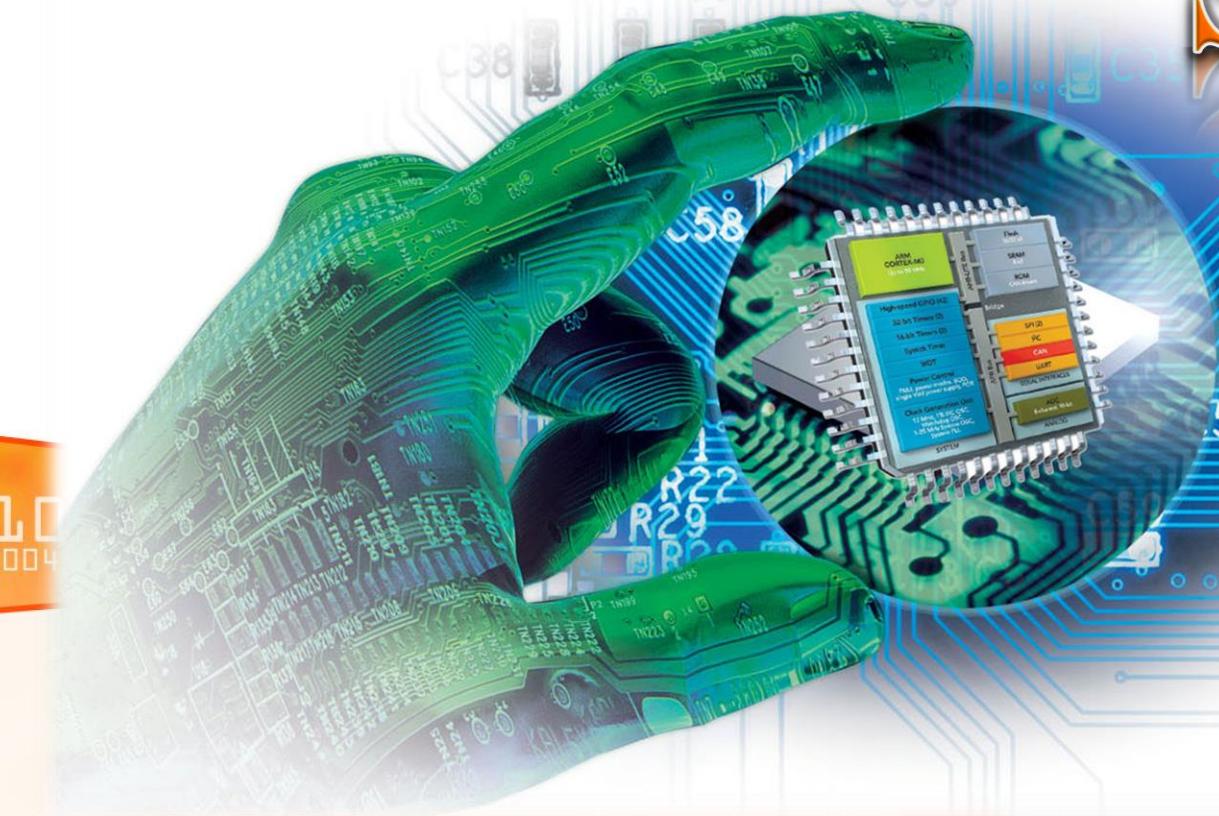


مُتَحَكِّمَات

STM32

4

5



موضو عات المعا ضر ظ :

- الأنواع المختلفة للمؤقتات
- مؤقتات الأغراض العامة
- أنماط العمل المختلفة للمؤقتات في متحكمات STM32
- ساعة الزمن الحقيقي (RTC)
- تطبيقات عملية

المؤقتات Timers

- يوجد في متحكمات STM32 العديد من المؤقتات المختلفة كل منها بإمكانها العمل بأنماط مختلفة
- المؤقت عبارة عن عداد يبدأ بالعد من الصفر ويزداد بمقدار عدة واحدة مع كل نبضة ساعة للمتحكم
- بإمكانه العد التصاعدي و التنازلي على حد سواء
- تسمح خاصية ال prescaler أو المقسم الترددی بتقسيم تردد الساعة على عدد معين يتم اختياره بين ال 0 و 65535

الأنواع المختلفة للمؤقتات

الأنواع المختلفة للمؤقتات

المؤقتات
عالية
الدقة

المؤقتات
المتقدمة

مؤقتات
الأغراض
العامة

مؤقتات
الطاقة
المنخفضة

المؤقتات
الأساسية

مؤقتات الأغراض العامة General Purpose Timers:

- المؤقتات في هذه المجموعة تكون إما 16 أو 32 بت (بناءً على عائلة STM32)
- يمتلك عداد تصاعدي / تنازلي بطول 16 بت قابل لإعادة التحميل **auto-reload counter**
- مقسم جهد بطول 16 بت يستخدم لتقسيم تردد الساعة للمتحكم على أي عدد يتراوح بين 1 و 65535

أنماط العمل المختلفة للمؤقتات في متحكمات STM32

Input
Capture

Output
Compare

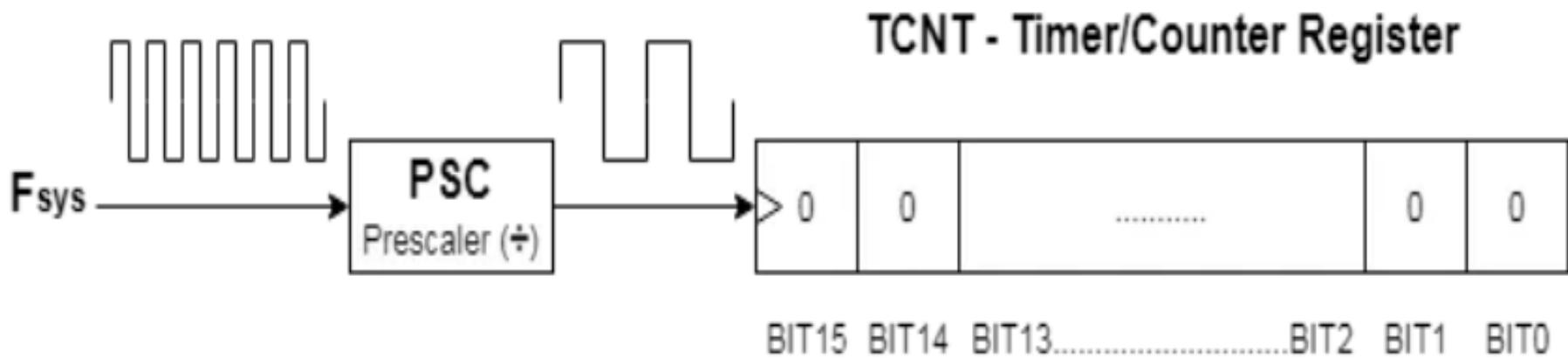
نط
 PWM

نط
 العداد

نط
 المؤقت

نُمْطُ الْمُؤْقَتِ Timer

- عند عمل المؤقت بنمط Timer، فإن المسجل TCNT تزداد قيمة بمقدار واحد مع كل نبضة ساعة للمؤقت
- يكون مصدر الساعة للمؤقت في هذه الحالة داخلي قادم من ساعة المتحكم



- حيث يقوم بالعد من الصفر إلى القيمة المحددة في حقل الـ **Period (preload)** أثناء تهيئة المؤقت ، وأكبر قيمة يمكن أن يصل إليها تحدد حسب طول المؤقت، حيث المؤقت 16 بت يمكنه العد إلى **0xffffffff** والموقت ذو 32 بت يمكنه العد إلى **0xffffffff ffffff**

نُمط المؤقت Timer

يعتمد تردد (سرعة العد) على المقسم الترددـي Prescaler حيث يتم تقسيم تردد ساعة المؤقت على واحدة من القيم المتاحة وهي من 1 حتى 65535 (حيث أن مسجل الـ Prescaler بطول 16 بت)

عندما يصل العداد إلى القيمة المحددة (preload) يحدث ما يسمى بالـ overflow أي يقوم بالتصغير والعد مرة أخرى من الصفر و يتم رفع العلم الخاص بالـ overflow Update Event(UEV)

نُمط المُؤقت Timer

ولحساب الزمن الذي سيطفح عنده المؤقت نستخدم المعادلة التالية: □

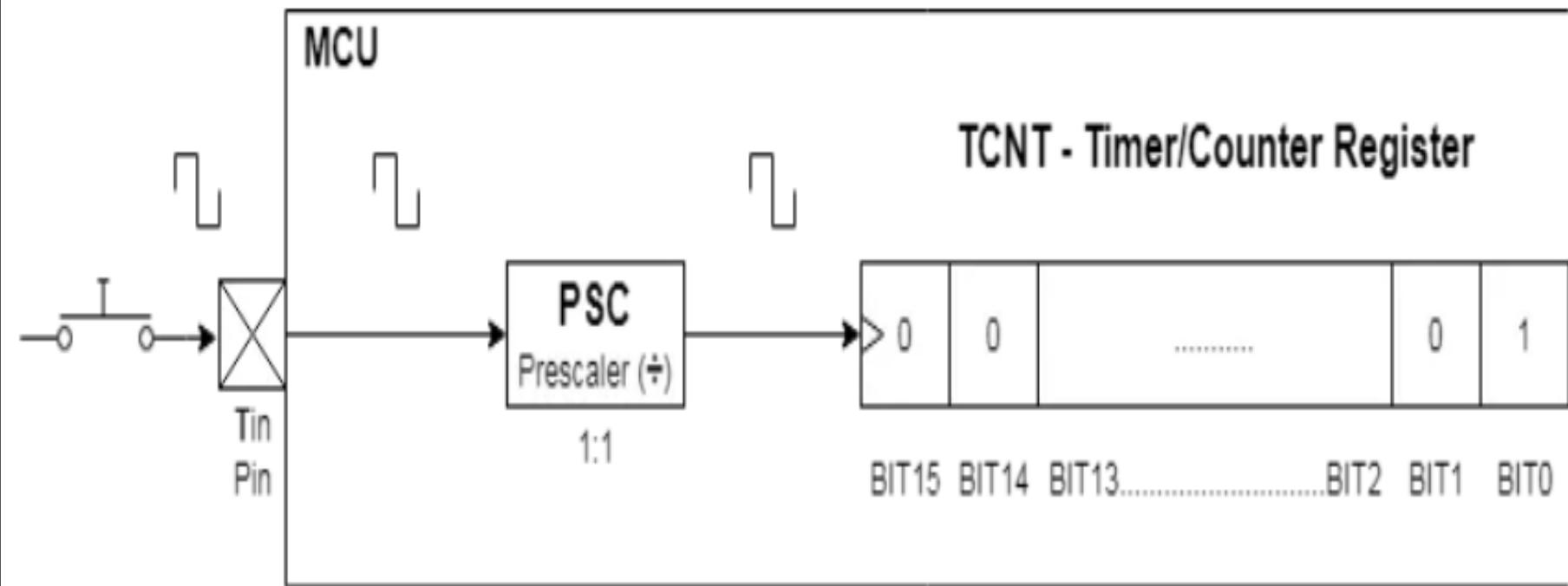
$$T_{out} = \frac{\text{Prescaler} \times \text{Preload}}{F_{CLK}}$$

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

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

نُمطُ العَدَاد Counter

يمكن للمؤقت أن يعمل بنمط Counter وفي هذه الحالة سيكون مصدراً لـ الساعة للمؤقت عبارة عن إشارة خارجية ممكن أن تكون قادمة من مفتاح لحظي، عندما ستزداد قيمة المؤقت مع كل جبهة صاعدة/هابطة عند ضغط المفتاح اللحظي وبالتالي سيعد المؤقت عدد المرات التي تم فيها ضغط المفتاح اللحظي



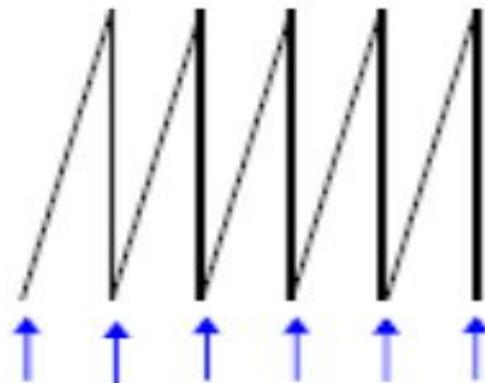
نُمطُ الْعَدَاد Counter

ويمكنه أن يعمل بثلاث أنماط مختلفة هي:

□ نُمطُ العد التصاعدي Up-counting Mode

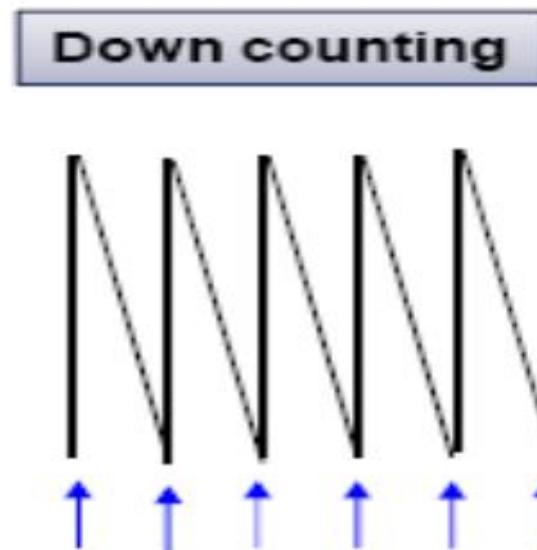
في هذا النمط فإن العداد يبدأ بالعد من الصفر مع كل نبضة قادمة على قطب الدخل ويستمر حتى يصل إلى القيمة المخزنة مسبقاً الموجودة في المسجل (TIMx_ARR)، ثم يعود للقيمة صفر ويولد حدث الطفhan كل يتم توليد حدث Update event مع كل طفحان للعداد

Up counting



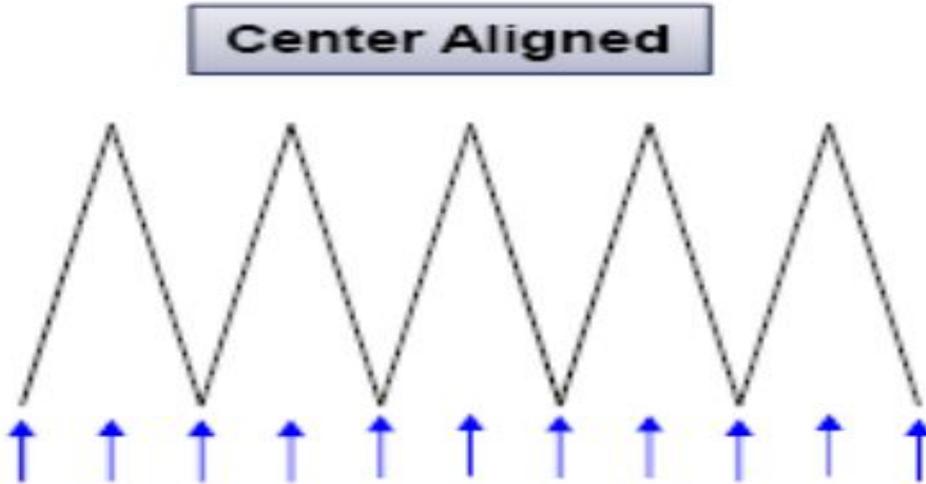
نُمطُ العَدِ التَّنَازُلِي Down-counting Mode

فِي هَذَا النُّمطْ فَإِنَّ الْعَدَادَ يَبْدأُ بِالْعُدُّ مِنْ القيمة المخزنة **auto-reload** في المسجل **TIMx-ARR** مع كل نبضة قادمة على قطب الدخل **value** وَيَسْتَمِرُ لِيَصْلُ إِلَى الصَّفَرِ ثُمَّ يَعُوْلِدُ مِنَ القيمة المخزنة سَابِقًاً وَيَوْلِدُ حَدَثًا **Underflow event** أَيْضًاً يَتَمُّ توليدُهُ مَعَ كُلِّ **Update** **Underflow**



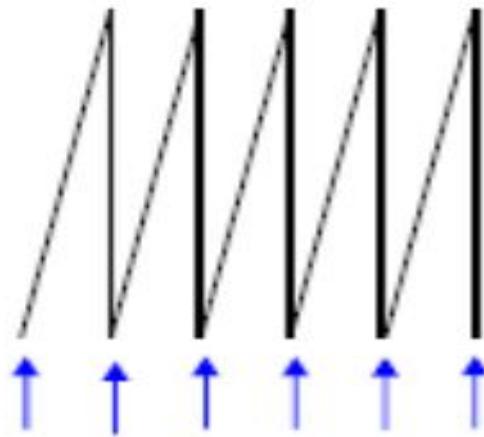
نُمطُ العَدَاد التصاعدي تنازلي Center-Aligned Mode:

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 الطفhan 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 Underflow ثم يعود للعد التصاعدي من الصفر وهكذا...،

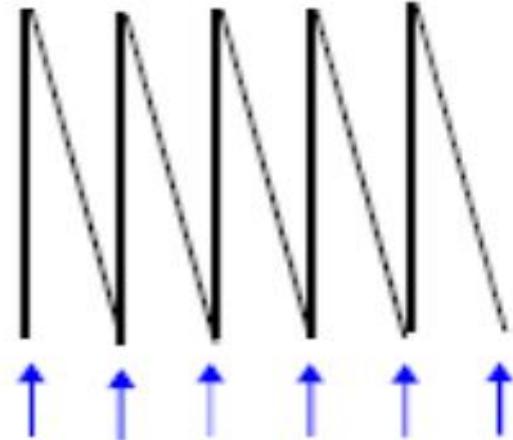


نُمْطُ الْعَدَاد Counter

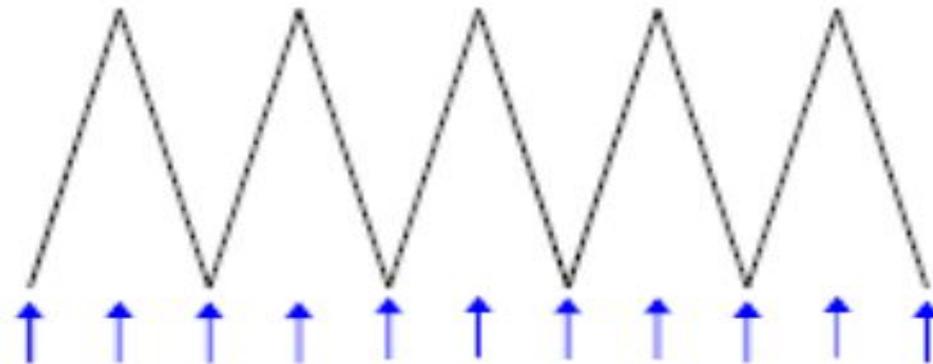
Up counting



Down counting

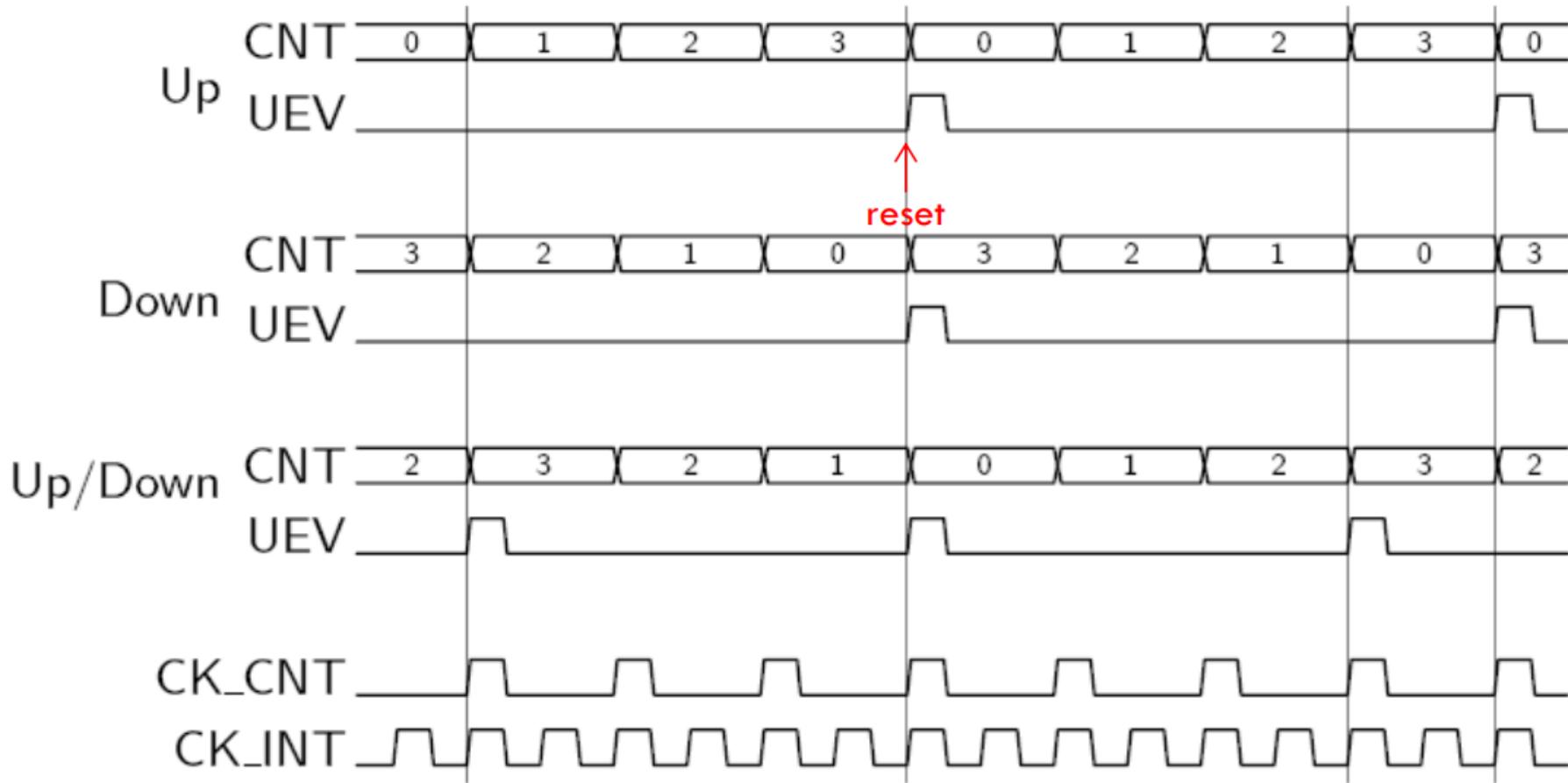


Center Aligned



نُمْطُ الْعَدَاد Counter

ويكون المخطط الزمني لأنماط العد الثلاث :

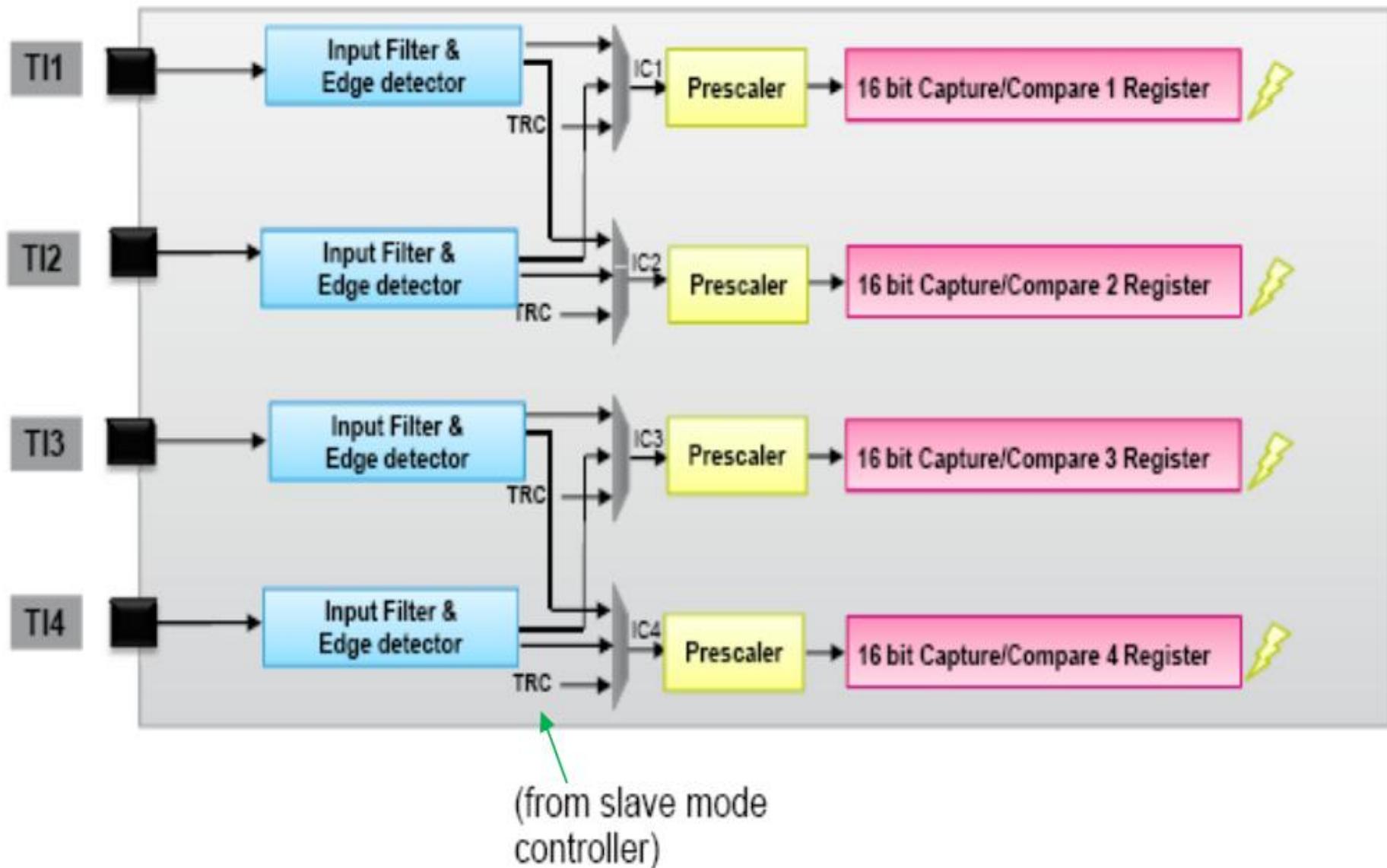


Counter Modes (ARR=3, PSC=1)

نط^ط Input Capture mode

- في نط^ط Input Capture يستقبل المؤقت نبضات الساعة الخاصة به من مصدر داخلي (ساعة المتحكم بعد استخدام المقسم التردد^y)
- يستمر بالعد إلى أن يحدث حدث معين (جبهة صاعدة/ جبهة هابطة) على قطب المتحكم الخاص بقناة الـ Input Capture عند^{ها} يتم حفظ القيمة التي وصل إليها المؤقت إلى مسجل input capture register
- لكل مؤقت في متحكمات STM32 عدة قنوات (input capture/compare output) channels مرتبطة بأقطاب المتحكم يمكن معرفتها من خلال الـ datasheet الخاصة بالمتحكم

Input Capture mode نمط



نُمط PWM mode

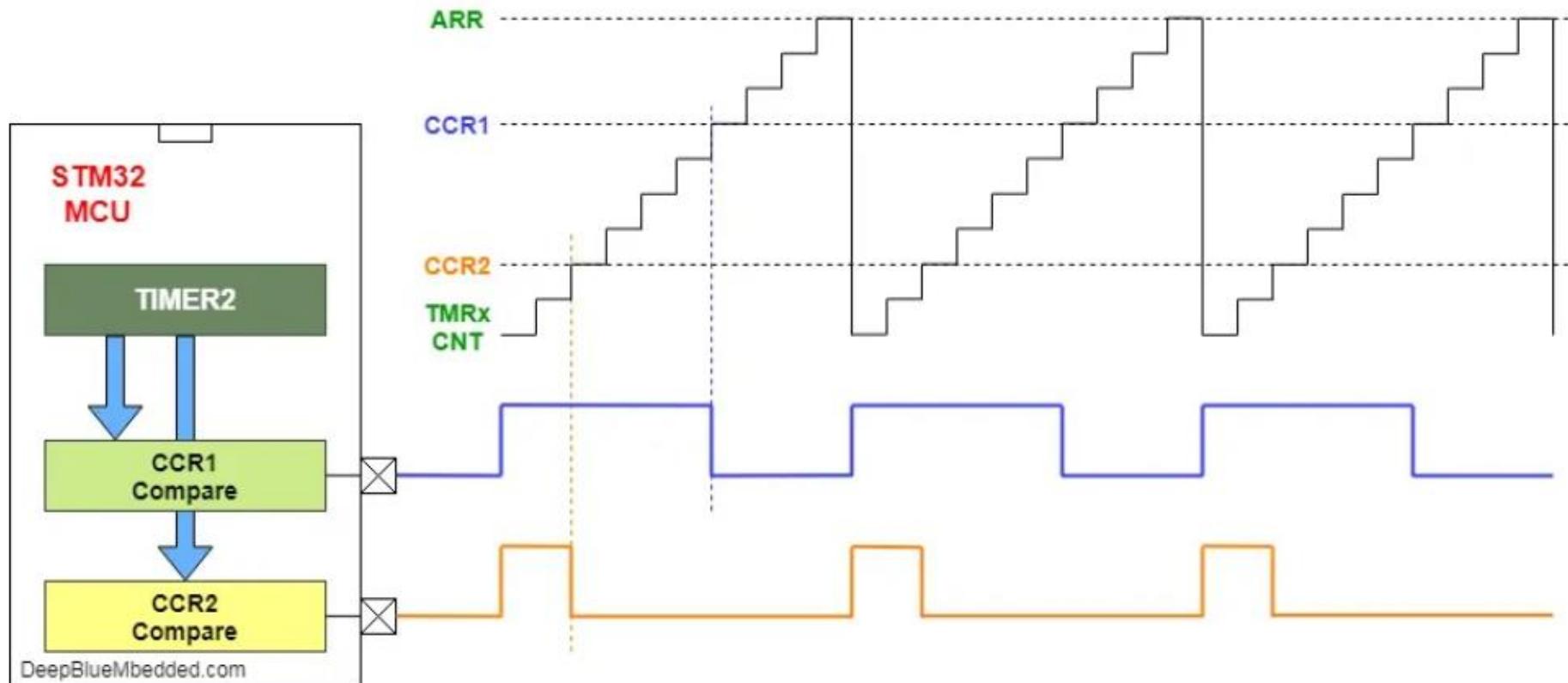
□ في نُمط PWM mode يستقبل المؤقت نبضات الساعة الخاصة به من الساعة الداخلية للمتحكم حيث يبدأ بالعد من الصفر ويزداد مع كل نبضة ساعة للمتحكم (طبعاً مع مراعاة إعدادات المقسم الترددية للمؤقت)

□ يتم وضع قطب الخرج الخاص بالـ PWM في وضع HIGH ويبقى كذلك إلى أن يصل العداد إلى القيمة المخزنة في المسجل CCRx، عدّها يصبح قطب الخرج في وضع LOW إلى أن يصل العداد إلى القيمة المخزنة في المسجل ARRx، وهذا

□ يدعى شكل الإشارة الناتجة بالـ Pulse Width Modulation (Modulation)، حيث يتم التحكم بالتردد من خلال تردد الساعة الداخلية للنظام، والمقسم الترددية Prescaler بالإضافة إلى قيمة المسجل (Auto Reload register) ARRx، كما يتم تحديد قيمة دورة التشغيل الـ duty cycle من خلال قيمة المسجل الـ CCR1

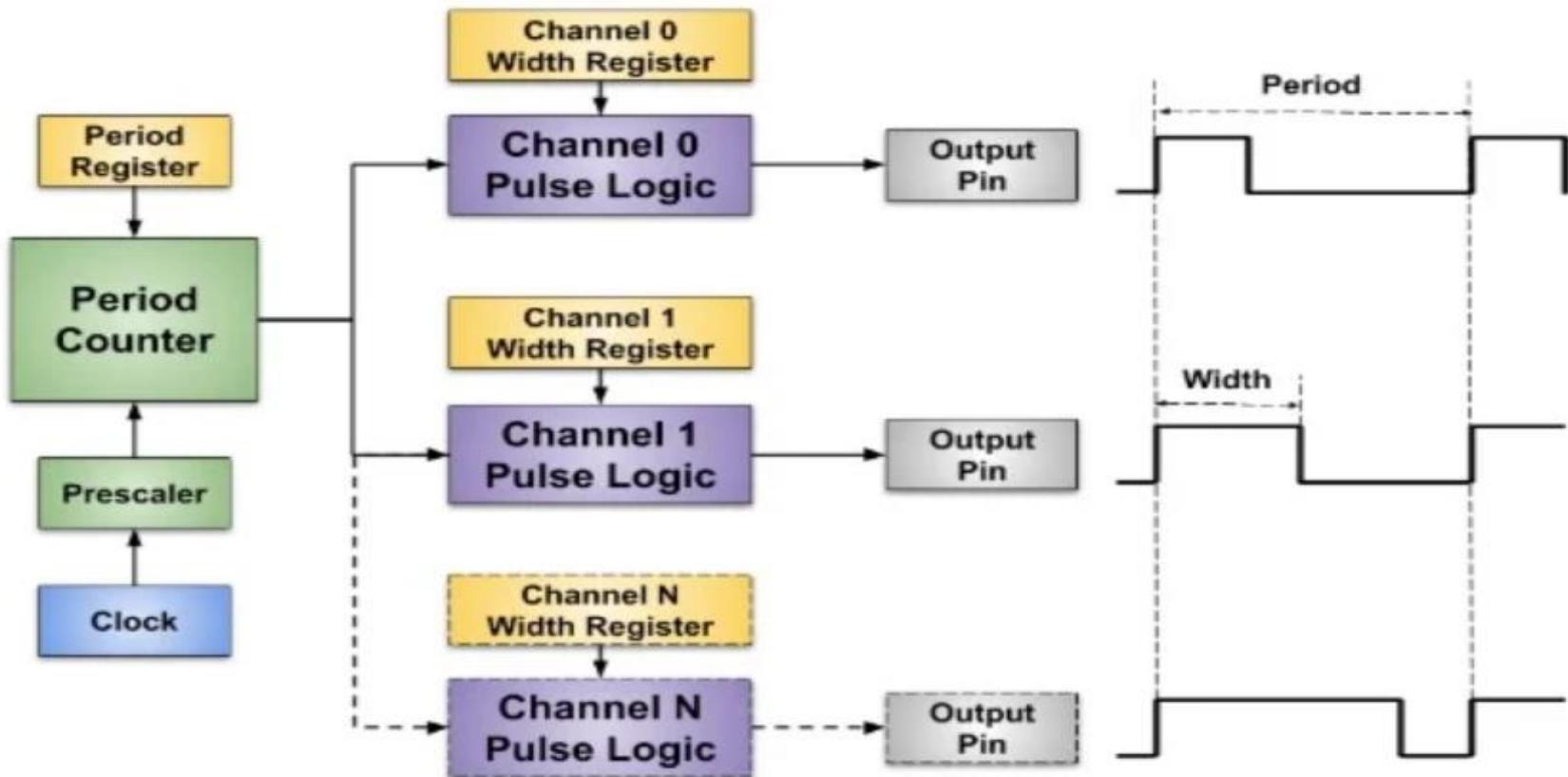
نُمَطْ PWM mode

□ يوضح المخطط التالي كيفية تأثير قيمة المسجل ARR في دور (تردد) إشارة الـ PWM، وكيف تؤثر قيمة المسجل CCR1 في قيمة دورة التشغيل duty cycle



نُمَطْ PWM mode

لكل مؤقت من مؤقتات المتحكم STM32 عدة قنوات ، لذا فإن كل مؤقت بإمكانه توليد عدة إشارات PWM لكل منها دورة تشغيل مختلفة ولكن لها نفس التردد وتعمل بالتزامن مع بعضها



نُمط PWM mode

تردد إشارة الـ :PWM

من PWM (1/FPWM) خلال

يتم التحكم بدور إشارة الـ البارامتراط التالية:

قيمة المسجل ARR

قيمة المقسم التردددي Prescaler

تردد الساعة الداخلية internal clock

عدد مرات التكرار

وذلك من خلال العلاقة التالية:

$$F_{PWM} = \frac{F_{CLK}}{(ARR + 1) \times (PSC + 1) \times (RCR + 1)}$$

نُمَط PWM mode

مثال:

• ARR=65535 • Prescaler=1 • MHZ F_{CLK} 72= 72
PWM:، احسب تردد نبضات الـ RCR =0

$$F_{PWM} = \frac{72 \times (10^6)}{(65535 + 1) \times (1 + 1) \times 1} = 549.3 HZ$$

Duty Cycle: دورة التشغيل

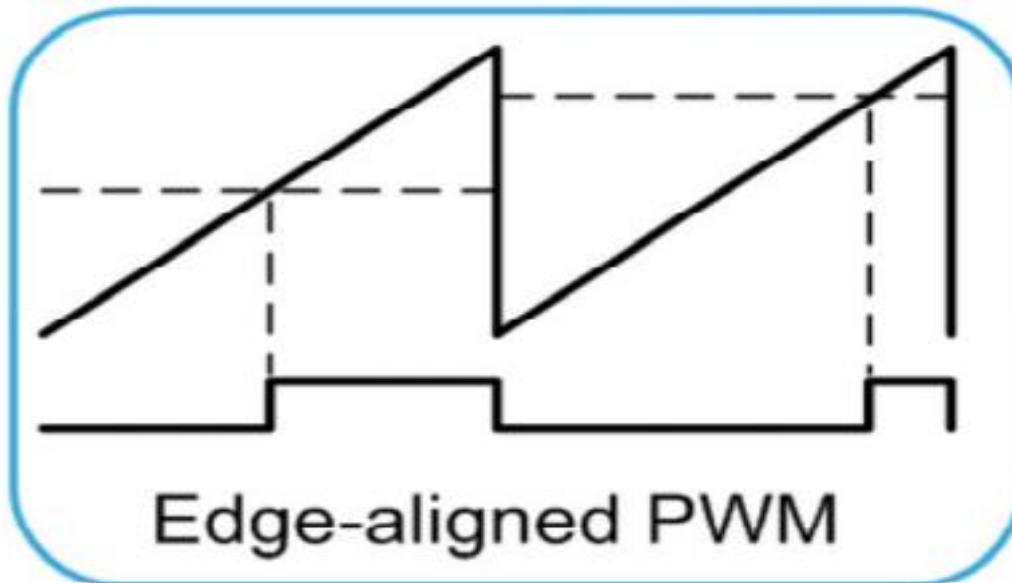
عند عمل المؤقت بنمط PWM وتوليد النبضات في وضع الـ edge-aligned mode up-counting، فإن دورة التشغيل يتم حسابها من خلال العلاقة التالية:

$$DutyCycle_{PWM}[\%] = \frac{CCRx}{ARRx} [\%]$$

نُمَط PWM mode

أنماط الـ PWM المختلفة:

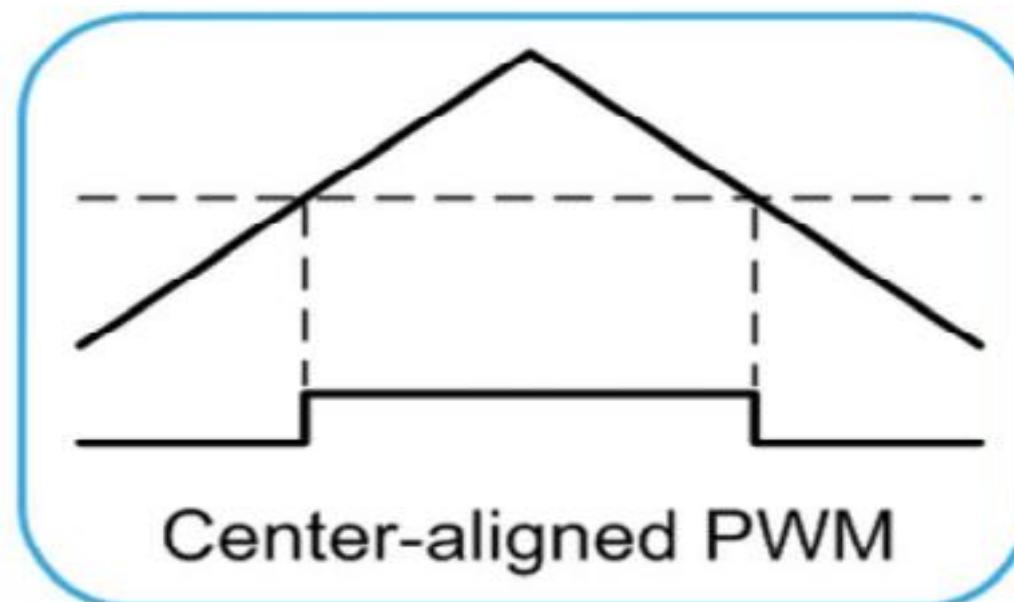
□ **Edge-aligned mode**: في هذا النمط فإن العداد يقوم بالعد بشكل تصاعدي فقط أو تنازلي فقط، وبإمكان المؤقت الواحد أن يولد حتى الـ 6 إشارات PWM لها نفس التردد ولكن بدورات تشغيل مختلفة، وهذه الإشارات جميعها متزامنة باعتبار أن الجبهة الهاابطة هي نفسها لجميع الإشارات.



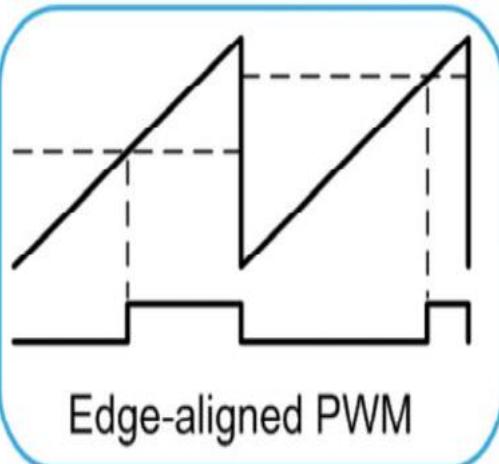
نُمط PWM mode

أنماط الـ PWM المختلفة:

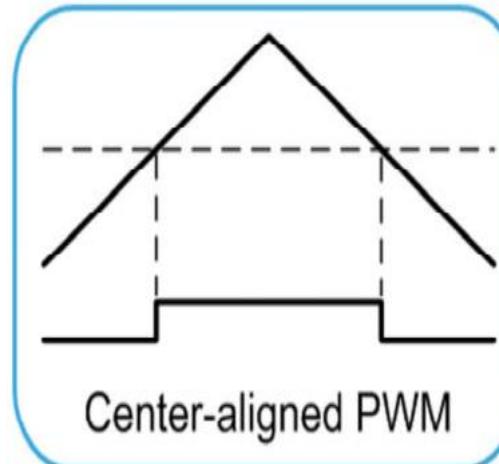
□ **Center-aligned mode** في هذا النمط فإن العداد يقوم بالعد بشكل تصاعدي/تنازلي، وتكون إشارات الـ PWM الناتجة من مؤقت واحد غير متزامنة لأن الجبهة الهاابطة لكل منها مختلفة، لذا فإن أزمنة التبديل لكل إشارة PWM تكون مختلفة عن الإشارة الأخرى



PWM mode نمط

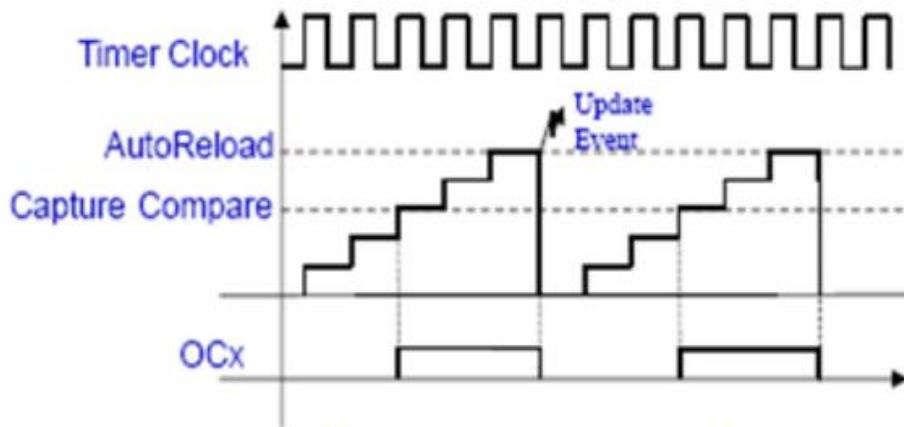


Edge-aligned PWM



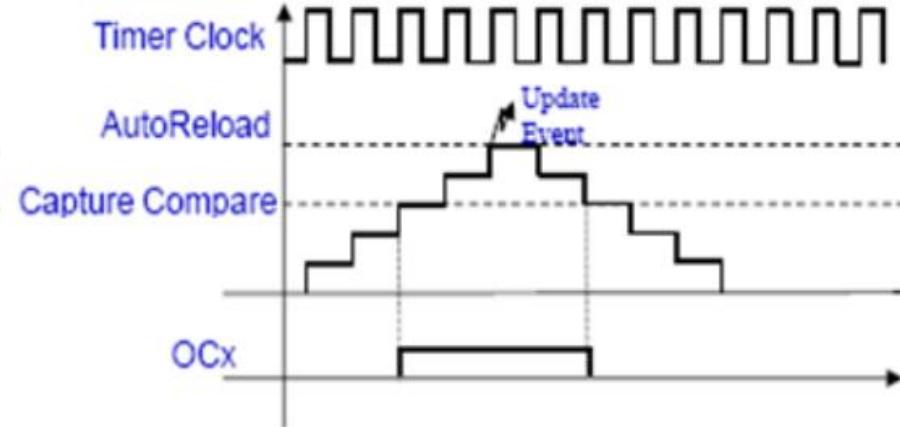
Center-aligned PWM

Edge-aligned Mode



PWM mode 2

Center-aligned Mode



هناك ثلات أوضاع مختلفة لاستخدام المؤقتات هي:

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 تسبب 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();

أوضاع الاستخدام المختلفة للمؤقتات في متحكمات STM32

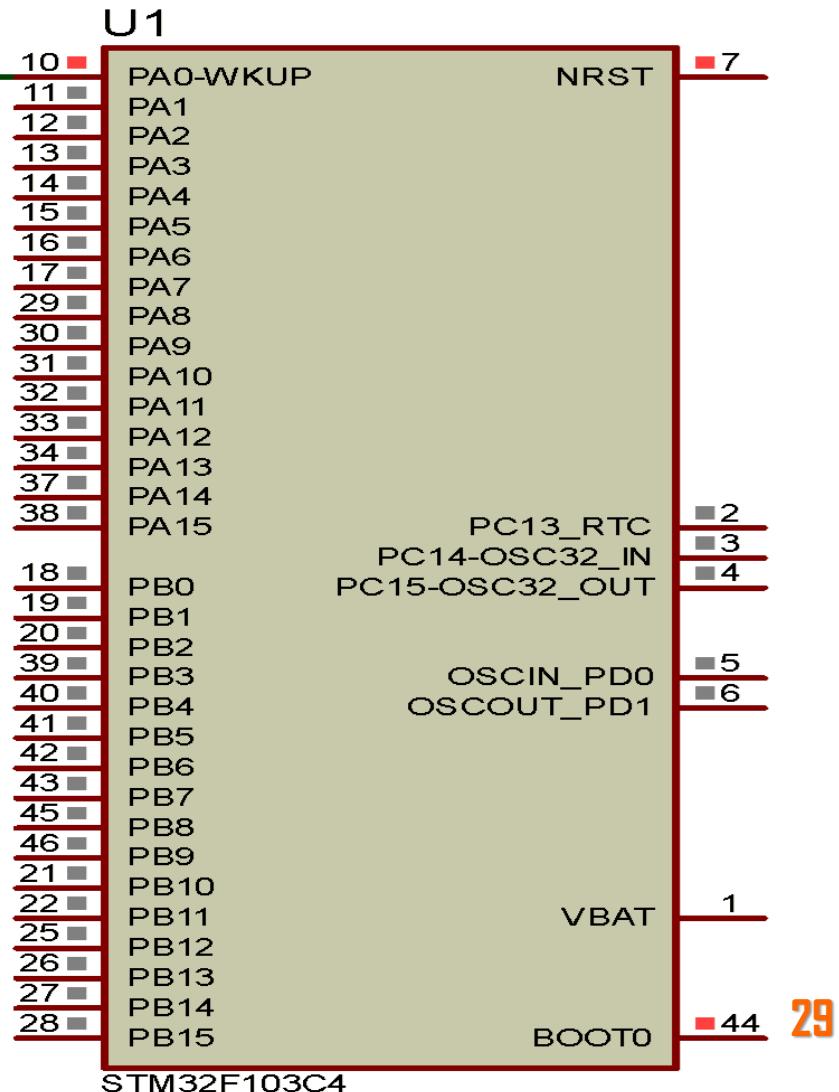
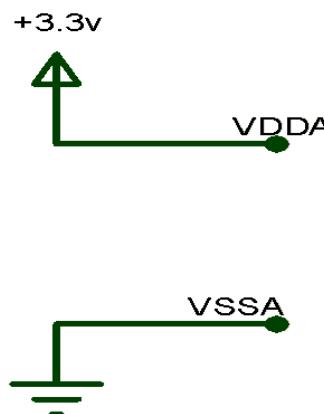
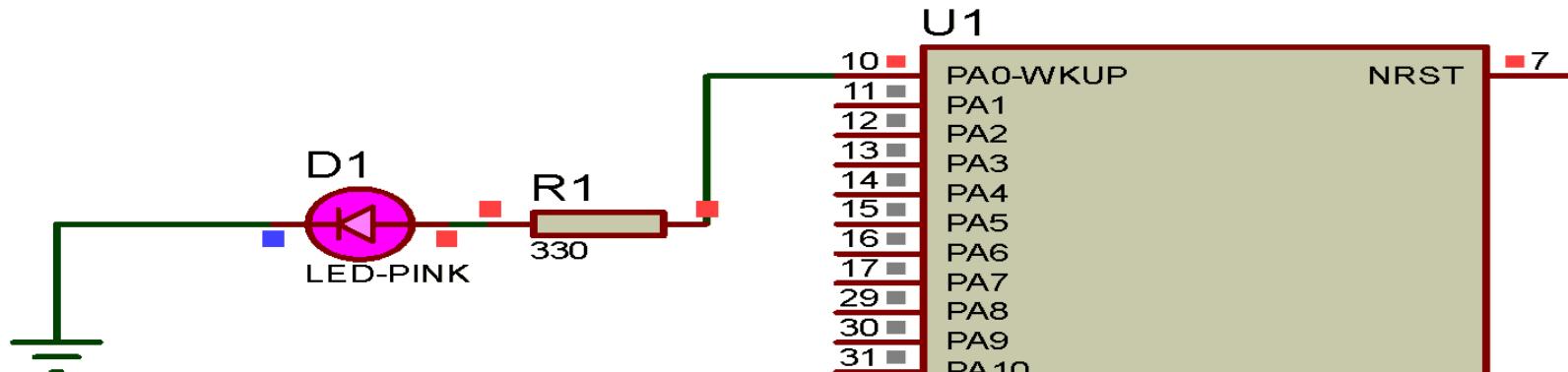
وضع Interrupt  : في هذا الوضع عند الوصول إلى Overflow/underflow أو أي من أحداث المقاطة سيتم التوجّه آلياً لتنفيذ برنامج خدمة المقاطة، وهذا الوضع الذي سستخدمه في جميع التطبيقات القادمة.

توفر مكتبة HAL الدالة التالية لبدء المؤقت في وضع المقاطة:

`HAL_TIM_Base_Start();`

وضع DMA 

التطبيق العملي الأول : استخدام المؤقت في نمط Timer mode وبوضع المقاطعة لتوليد زمن بدلاً من استخدام دالة delay() واستخدامه في عملToggle لليد الموصول على القطب PA5



التطبيق العملي : استخدام المؤقت في نمط Timer mode وبوضع المقاطعة لتوليد زمن بدلًا من استخدام دالة delay() واستخدامه في عمل Toggle لـ ليد الموصل على القطب PA5

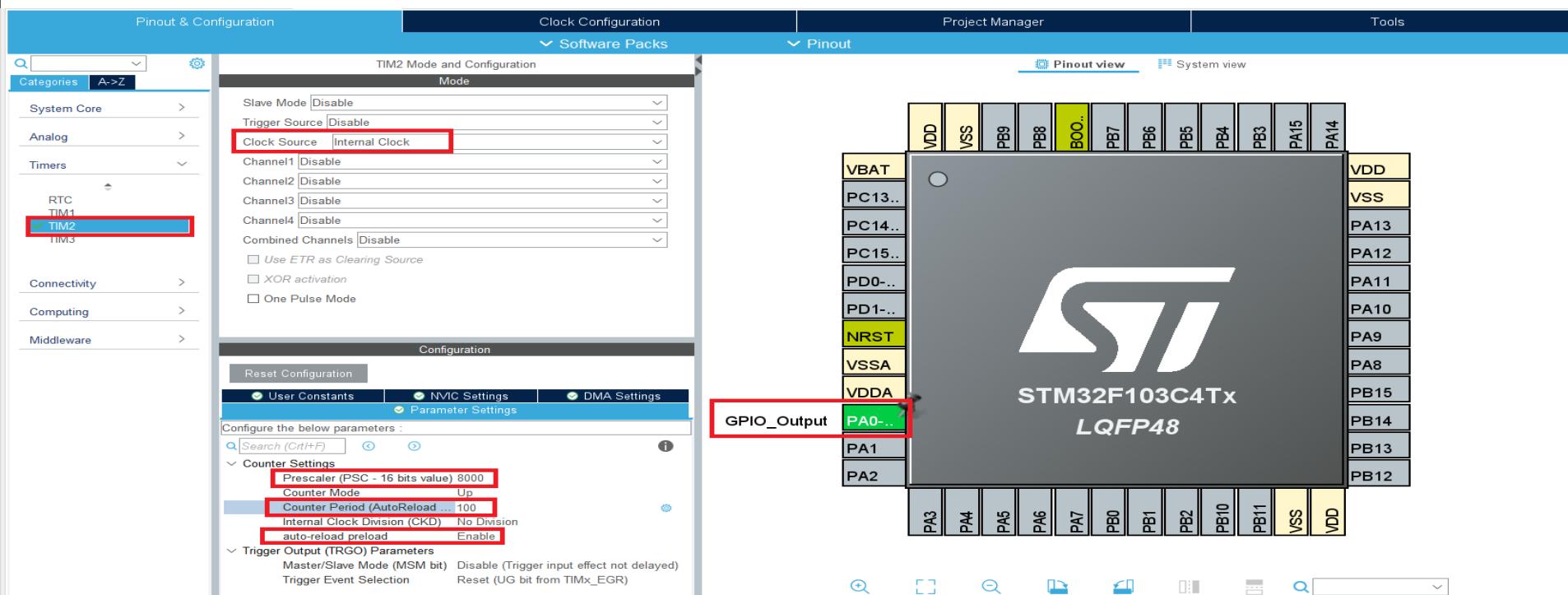
- ختار القطب PA5 لضبطه كقطب خرج
- نقوم بضبط إعدادات المؤقت كي نحصل على زمن 100 msec
لعكس حالة الـ لـيد الموصل على القطب رقم 5 من المنفذ A، من
المعادلة السابقة سنفترض أن تردد ساعة المتحكم هي 8 MHz
والمقسم التردد 8000 بقى فقط حساب (Period) Preload
بتغيير القيم في المعادلة :

$$T_{out} = \frac{Prescaler \times Preload}{F_{CLK}} = \frac{8000 \times Preload}{8000000}$$

$$Preload = 100$$

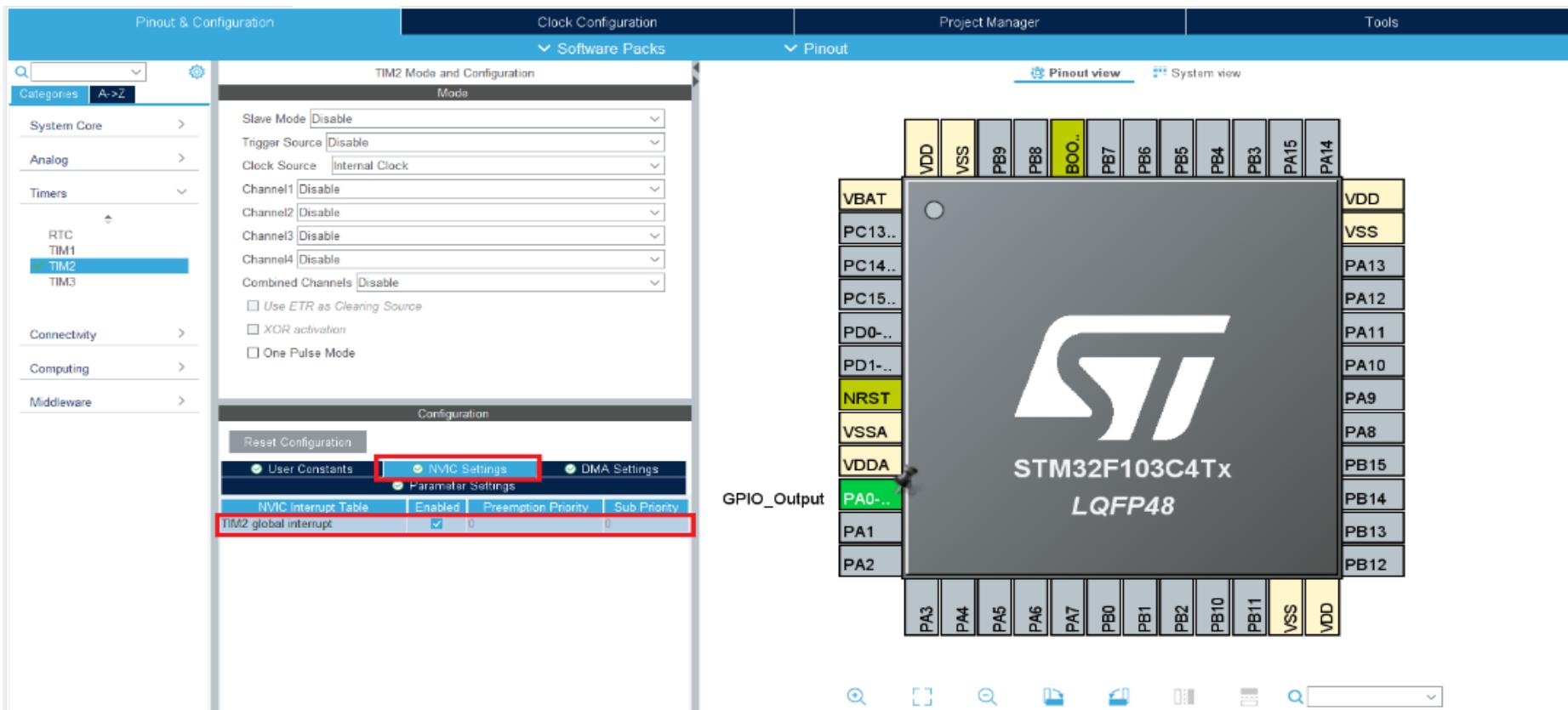
التطبيق العملي : استخدام المؤقت في نمط Timer mode وبوضع المقاطعة لتوليد زمن بدلاً من استخدام دالة delay() واستخدامه في عمل Toggle لليد الموصول على القطب PA5

سنقوم باختيار مصدر الساعة للمؤقت داخلي، المقسم الترددية 8000 ، الـ $\text{Preload} = 100$ ، أيضاً سنقوم بتفعيل إعادة التحميل التلقائي، كما في الشكل التالي:

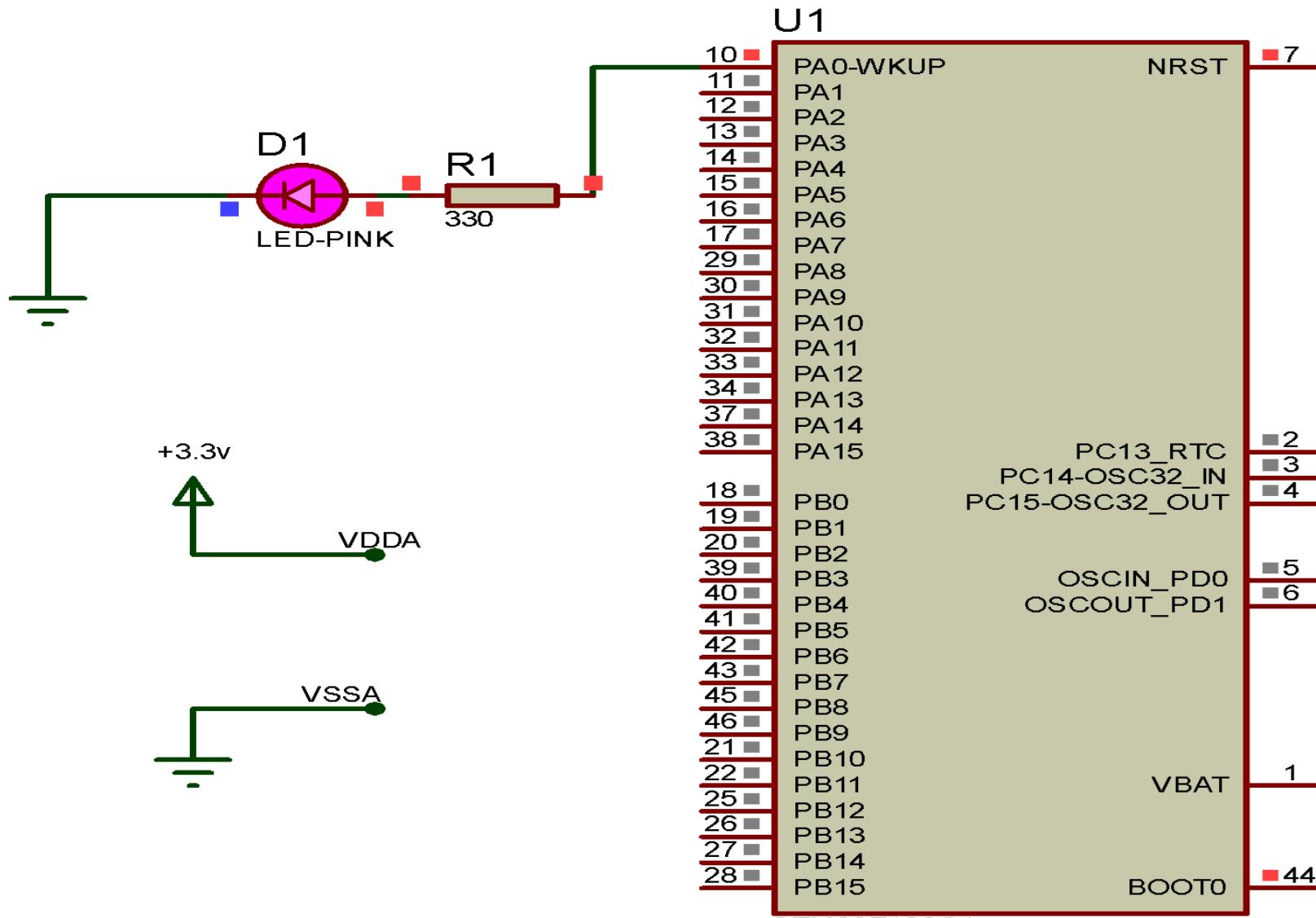


التطبيق العملي : استخدام المؤقت في نمط Timer mode وبوضع المقاطةة لتوليد زمن بدلاً من استخدام دالة delay() واستخدامه في عملToggle لـ ليد الموصل على القطب PA5

نقوم بتفعيل مقاطعة المؤقت من شريط الـ NVIC tab



التطبيق العملي الثاني : استخدام المؤقت في نمط PWM mode للتحكم في شدة إضاءة اليد



التطبيق العملي الثاني : استخدام المؤقت في نمط PWM mode للتحكم في شدة إضاءة اليد

سنتبع في هذا التطبيق الخطوات التالية للتحكم بشدة إضاءة اليد:

- ضبط باراترات المؤقت TIM2 ليعمل في نمط الـ PWM وباستخدام الساعة الداخلية للمتحكم internal clock، ثم تفعيل القناة الأولى CH1 لاستخدامها كقناة الخرج لإشارة الـ PWM
- ضبط قيمة المسجل ARR على القيمة العظمى وهي 65535 ، والقسم الترددی prescaler على 1، فيصبح التردد 61HZ خلال العلاقة:

$$F_{\text{PWM}} = (8 \times (10^6)) / ((65535 + 1) \times (1 + 1) \times 1) = 61 \text{HZ}$$

- التحكم بدورة التشغيل duty cycle من خلال كتابة القيمة المناسبة على المسجل CCMR1

جعل دورة التشغيل تتغير من 0% حتى 100% وتعيد الكرة باستمرار

التطبيق العملي الثاني : استخدام المؤقت في نمط PWM mode للتحكم في شدة إضاءة اليد

□ ضبط إعدادات المؤقت ليعمل في نمط PWM يقوم بضبط مصدر الساعة للمؤقت على الساعة الداخلية للنظام internal clock، يقوم بتشغيل القناة CH1 لتكون القناة التي سيتم إخراج إشارة الـ PWM عليها، ضبط قيمة المسجل ARR على القيمة العظمى وهي 65535 ، والمقسم الترددى prescaler على 1، فيصبح التردد 61HZ ، نفعل خاصية Auto Reload preload PWM ونختار نمط إشارة الـ

التطبيق العملي الثاني : استخدام المؤقت في نمط PWM mode للتحكم في شدة إضاءة الـ LED

Pinout & Configuration Clock Configuration Project Manager Tools

Categories A-Z

System Core

- Analog
- Timers
 - RTC
 - TIM1
 - TIM2**
 - TIM3
- Connectivity
- Computing
- Middleware

Clock Configuration

Software Packs

Pinout

Pinout view **System view**

VDD	VSS	PB9	PB8	BO0	PB7	PB6	PB5	PB4	PB3	PA15	PA14	VDD	VSS	PA13	PA12	PA11	PA10	PA9	PA8	PB15	PB14	PB13	PB12
VBAT																							
PC13..																							
PC14..																							
PC15..																							
PD0-..																							
PD1-..																							
NRST																							
VSSA																							
VDDA																							
TIM2_CH1	PA0...																						
PA1																							
PA2																							
PA3																							
PA4																							
PA5																							
PA6																							
PA7																							
PB0																							
PB1																							
PB2																							
PB10																							
PB11																							
VSS																							
VDD																							

Search (Ctrl+F) **+** **□** **Q** **File** **Print** **Help**

ساعة الزمن الحقيقي Real Time Clock (RTC)

- ساعة الزمن الحقيقي عبارة عن أداة لحفظ الوقت، تستخدم مع التطبيقات التي يتم تنفيذها عند أزمنة محددة، كساعة التوقيت الموجودة ضمن الغسالات الآلية ، تطبيق إعطاء الأدوية للمرضى بأزمنة محددة وغيرها...
- فهي عبارة عن عدد زمن لكنها تعطي دقة أكبر من المؤقتات الموجودة في المتحكم، فالمؤقتات مناسبة لتوليد الأزمنة المختلفة و إشارات الـ PWM على سبيل المثال..
- معظم متحكمات 8bit لا تحتوي على RTC داخلية وإنما يتم استخدام إحدى شرائح الـ RTC الخارجية كـ DS1302 أو PCF8563 بالإضافة إلى بعض العناصر الالكترونية الازمة كي تعمل بشكل أمثل كما تحتاج إلى مساحة إضافية على الدارة المطبوعة

ساعة الزمن الحقيقي Real Time Clock (RTC)

- تحتوي متحكمات stm32 على موديول RTC مدمج بداخل المتحكم وهي لا تحتاج لأية عناصر إضافية أو دارات ملائمة كي تعمل لوحة الـ RTC مصدرى ساعة هما:
 - RTC Timer/counter: ويستخدم كمصدر ساعة للـ RTCCCLK
 - APB clock: ويستخدم كمصدر ساعة للـ RTC register من أجل عمليات القراءة والكتابة على المسجلات

ساعة الزمن الحقيقي Real Time Clock (RTC)

نبضات الساعة لوحدة الـ RTC (RTCCLK) يمكن أن تكون قادمة من:

(HSE) High Speed External clock

مقسمة على 32

(LSE) Low Speed External clock

(LSI) Low Speed Internal Clock

لكن عندما يكون المتحكم يعمل في نمط VBAT أو يكون في حالة إيقاف

تشغيل shutdown ، في هذه الحالة يجب أن يكون RTC clock هي

LSI أو LSE

ساعة الزمن الحقيقي RTC

- يتم تقسيم تردد clock RTC باستخدام مقسم تردد قابل للضبط وبطول 7bit ، ومن أجل تخفيض استهلاك الطاقة ينصح باستخدام نسبة تقسيم مرتفعة حيث القيمة الافتراضية هي 128
- ثم يتم تقسيم التردد الناتج عن المقسم باستخدام مقسم تردد آخر قابل للضبط وبطول 15bit ، ويجب أن يكون التردد الناتج عنه 1HZ كي يتم تحديث الزمن والتاريخ في كل 1sec كل BCD registers

Actual registers

Date

Day

Month

Date

Year

Time

HH

MM

SS

SSR

Synchronization

DR

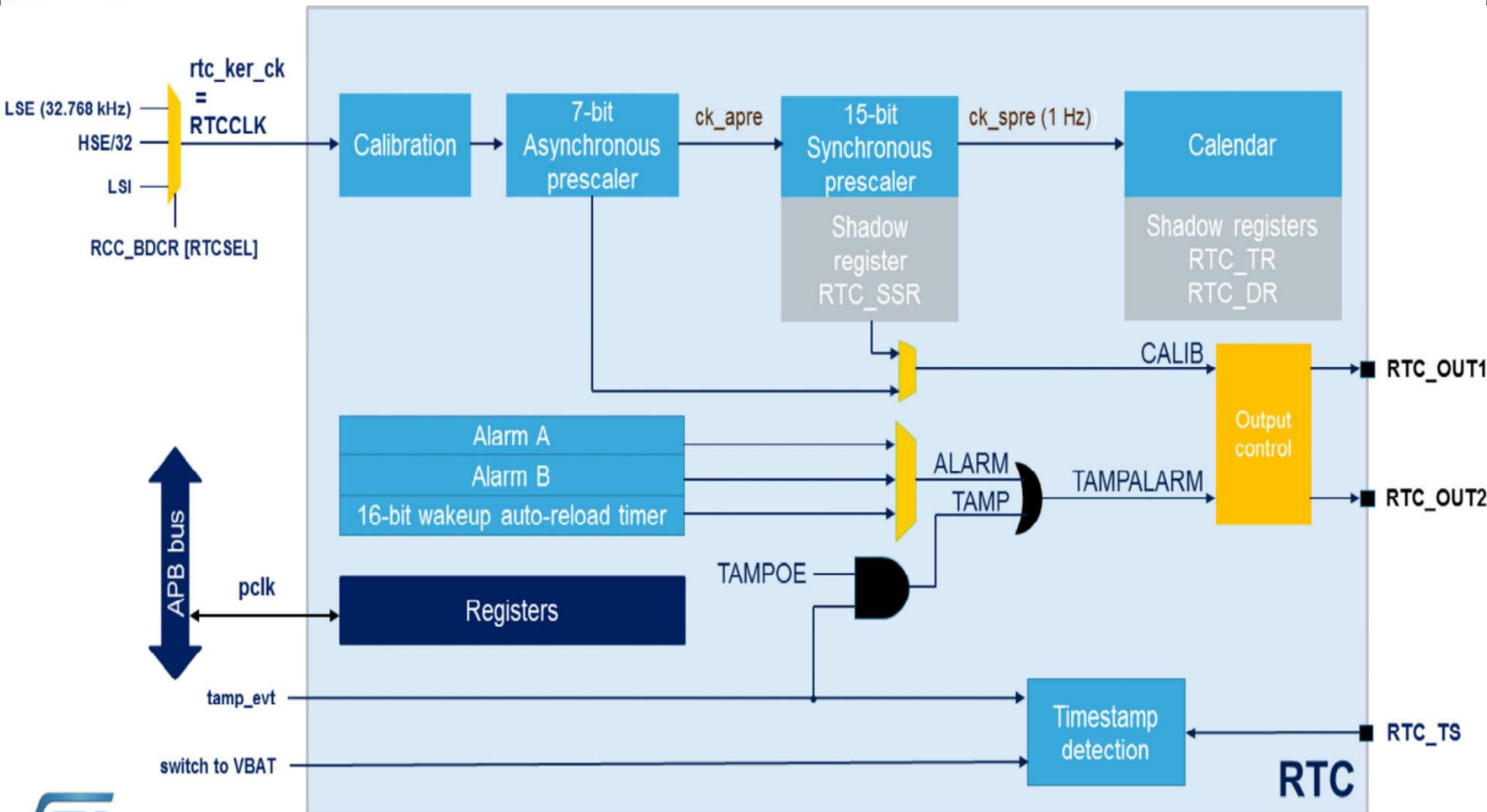
TR

SSR

Shadow registers

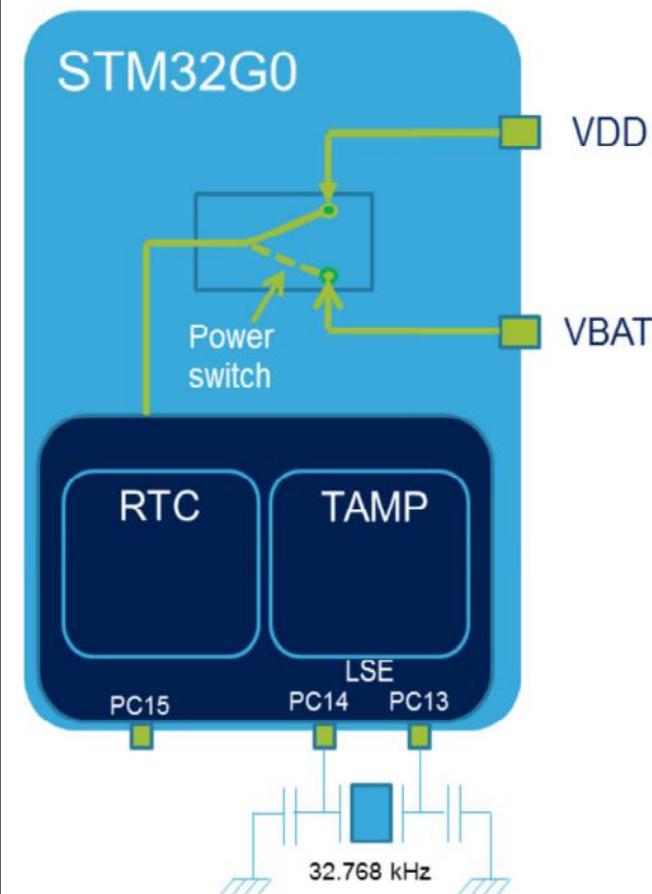
المخطط الصندوقي لساعة الزمن الحقيقي (RTC)

Real Time Clock (RTC)



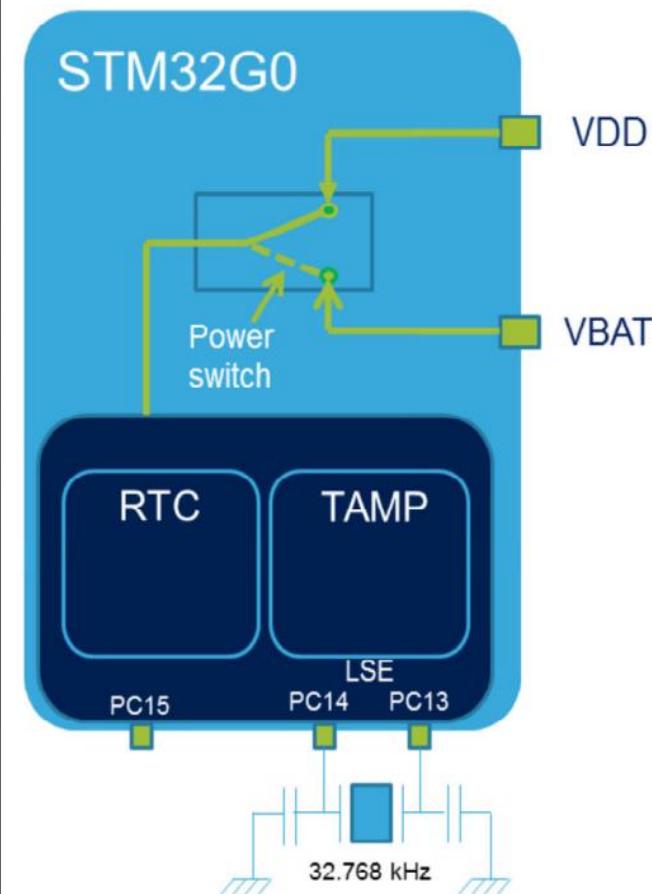
ساعة الزمن الحقيقي RTC

- يمكن لوحدة RTC أن تعمل في جميع أنماط الطاقة المنخفضة للمتحكم
- فعندما يتم تزويدها بنبضات الساعة من خلال Low speed external oscillator (LSE) بتردد 32.768KHz ، عندها مستمرة وحدة RTC بالعمل حتى عند فصل التغذية الأساسية عن المتحكم ، عندما يكون قطب VBAT موصول بطارية احتياطية



ساعة الزمن الحقيقي RTC

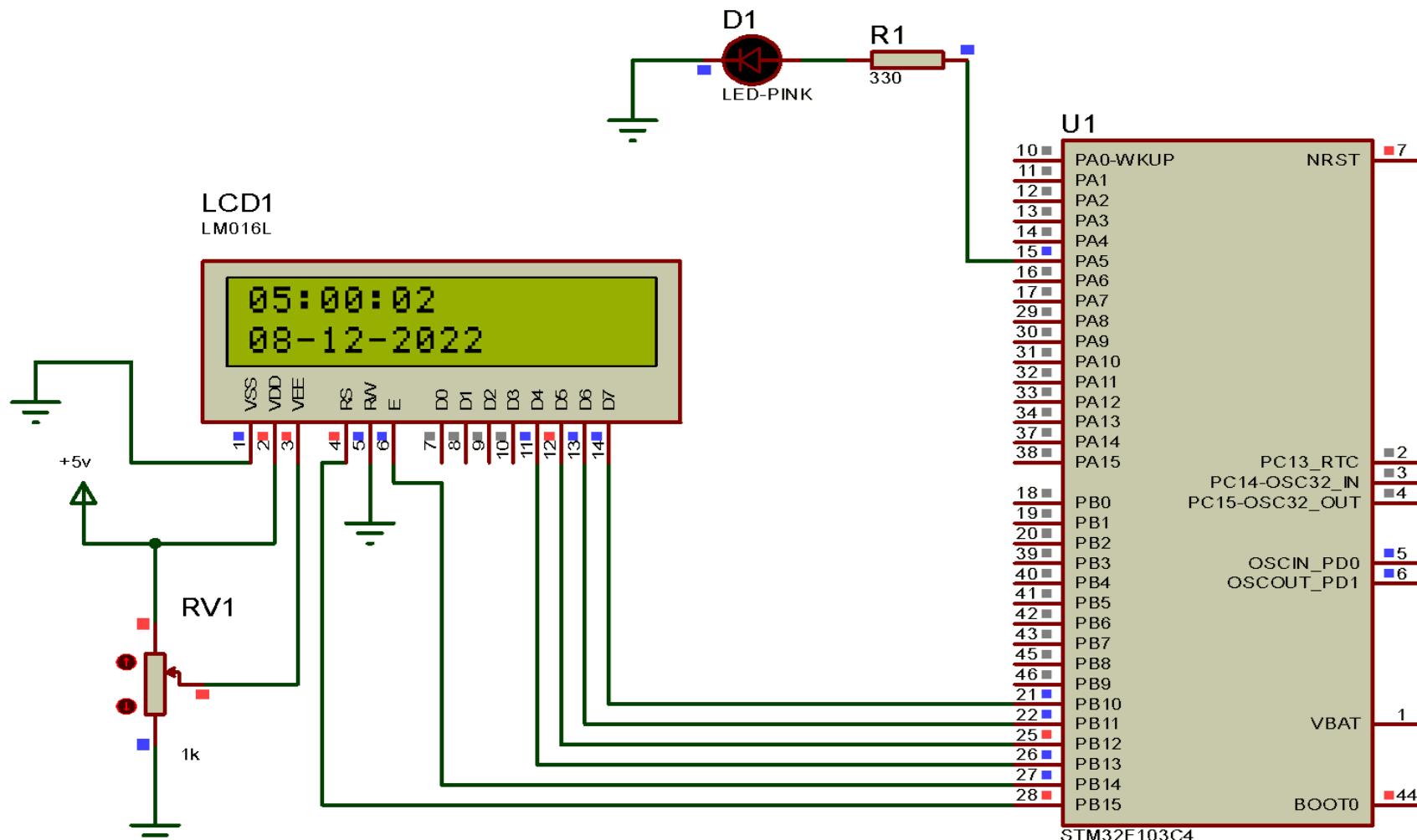
- استهلاك وحدة RTC للتيار فقط 300nA عند تغذيتها بجهد 1.8v
- وهذا التيار يتضمن استهلاك LSI للتيار
- يكون التقويم الناتج عن وحدة RTC مشفر بشفرة BCD لتقليل التعقيد في الكود البرمجي، بما في ذلك الثواني ، الدقائق ، الساعات، الأيام ، الشهور والسنين، بينما تكون أجزاء الثانية مشفرة بالشفرة الثنائية يمكن بسهولة إضافة أو إنفصال ساعة من التقويم لإدارة التوقيت الصيفي



ساعة الزمن الحقيقي Real Time Clock (RTC)

- لوحدة RTC مخرجان لهما القدرة على توليد تببيهان قابلان للبرمجة ولهم القدرة على إيقاظ المعالج من كافة أنماط توفير الطاقة
- تحتوي وحدة RTC على مؤقت مدمج قابل للضبط وإعادة تحميل القيمة آلياً والذي يستخدم لتوليد مقاطعات دورية لها القدرة على إيقاظ المعالج ، كما يمكن ضبط دقة هذا المؤقت

التطبيق العملي الثالث : استخدام وحدة الـ RTC لإظهار التاريخ و الوقت على شاشة lcd او ضبط المنبه على وقت محدد لتشغيل ليد



ضبط إعدادات وحدة RTC وتفعيل المقاطعة الخاصة بها

حيث سنقوم بضبط التاريخ والساعة والتزبيه من خلال الكود



The screenshot shows the STM32CubeMX software interface. On the left, the project navigation bar includes files like *rtc_simulation.ioc, main.c, lcd_txt.h, and stm32f1xx_hal_rtc.c. The main window has tabs for Pinout & Configuration, Clock Configuration, Project Manager, and Tools. The Pinout & Configuration tab is active, displaying the STM32F103C4Tx LQFP48 pinout. The pinout diagram shows pins PA14.., PC15.., PD0-.., PD1-.., NRST, VSSA, VDDA, PA0-.., PA1, PA2, PA3, PA4, PA5, PA6, PA7, PB0, PB1, PB2, PB10, PB11, VSS, and VDD. Pins PA5, PA10, PB11, and PB12 are highlighted in green and labeled as GPIO_Output. The RTC configuration section on the left shows 'Activate Clock Source' and 'Activate Calendar' checkboxes checked. The NVIC Settings table lists two entries:

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
RTC global interrupt	<input checked="" type="checkbox"/>	0	0
RTC alarm interrupt through EXTI line 17	<input checked="" type="checkbox"/>	0	0

Thank you for listening