

مُتَحَكِّمَات

STM32

5



موضو عات المحاضرة:

- الأنواع المختلفة للمؤقتات
- مؤقتات الأغراض العامة
- أنماط العمل المختلفة للمؤقتات في متحكمات STM32
- 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 الترددية 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
- له أربع مداخل / مخارج منطقية قابلة للبرمجة بأحد الوظائف التالية:

Input Capture -
Output Compare -
- توليد نبضات PWM
One-Pulse mode Output -

أنماط العمل المختلفة للمؤقتات في متحكمات STM32

Input
Capture

Output
Compare

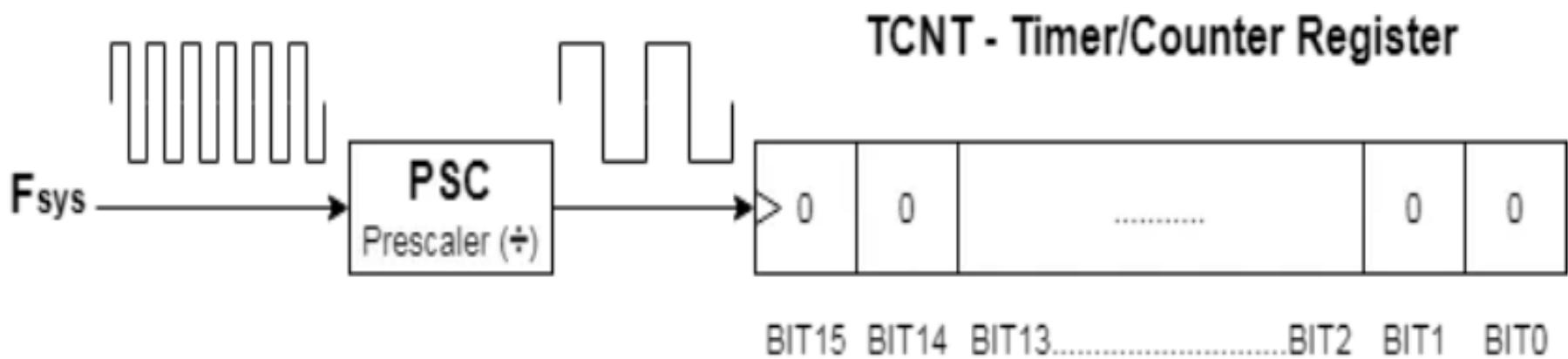
نط
 PWM

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 0xfffff fffff ذو 32 بت يمكنه العد إلى 7

نُمط المُؤقت 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: □

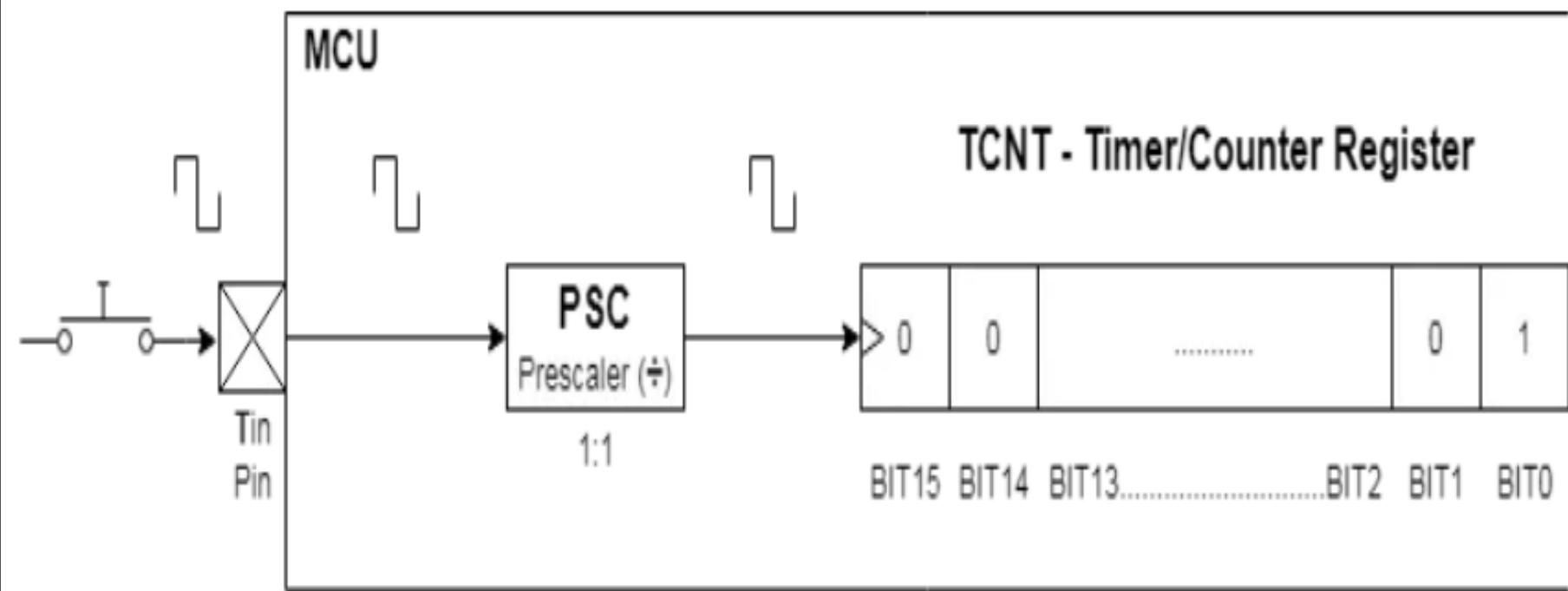
$$T_{out} = \frac{\text{Prescaler} \times \text{Preload}}{F_{CLK}}$$

على سبيل المثال ، لنفترض أن تردد الساعة للمتحكم مضبوط على MHz 48 وقيمة الـ Prescaler تساوي 48000 والـ period تساوي 500 سيفحدث 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



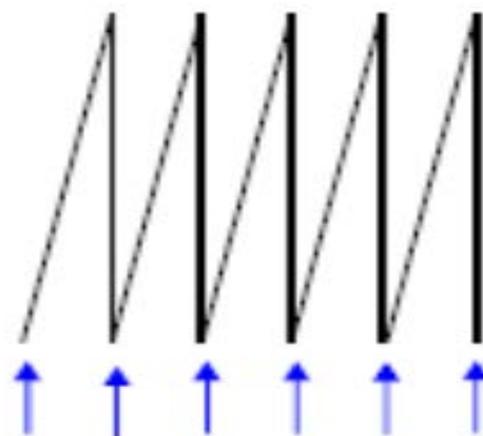
نُمطُ العَدَاد Counter

ويمكنه أن يعمل بثلاث أنماط مختلفة هي:

□ نُمطُ العَدِ التصاعدي Up-counting Mode

في هذا النُمط فإن العَدَاد يبدأ بالعَدِ من الصفر مع كل نبضة قادمة على قطب الدخُل ويستمر حتى يصل إلى القيمة المخزنة مسبقاً والموجودة في المسجل (TIMx_ARR)، ثم يعود للقيمة صفر ويولد حدث الطفاحان (Overflow) كلَّ مِنْتَهِيَةِ عَدِّيَّةِ العَدَادِ مع كل طفحان للعداد

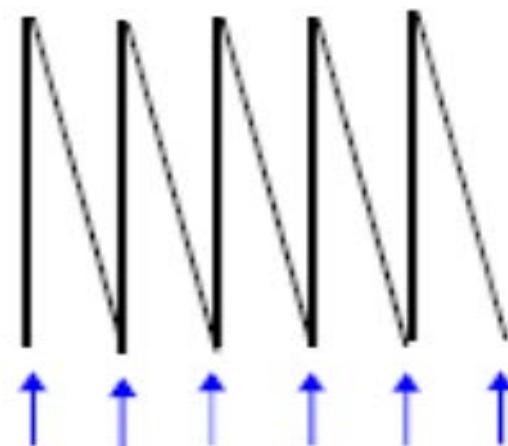
Up counting



نُمَطُ الْعَدِ التَّنَازُلِي Down-counting Mode

في هذا النُّمَط فإن العَدَاد يَبْدأ بِالْعُدُّ مِن القيمة المخزنة **auto-reload** في المسجل **TIMx-ARR** مع كل نبضة قادمة على قطب الدخول **value** ويستمر ليصل إلى الصفر ثم يعود ليبدأ من القيمة المخزنة سابقاً ويولد حدث **Underflow event** أيضاً يتم توليد حدث **Underflow** مع كل **Update**

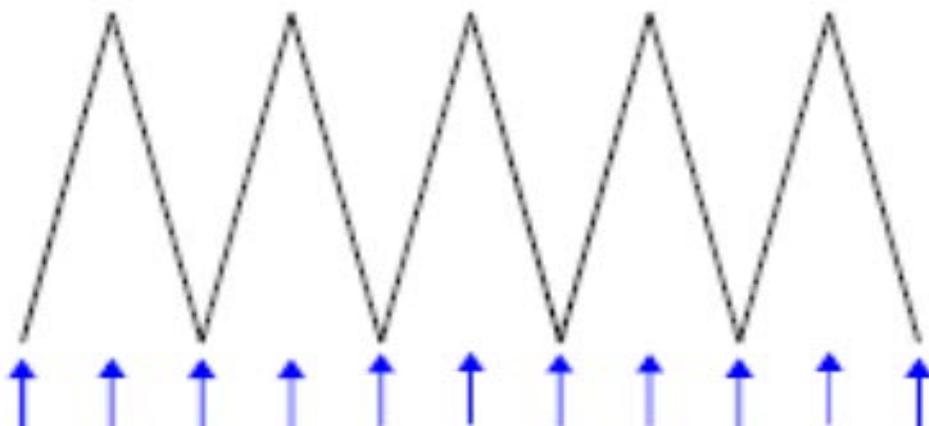
Down counting



نُمطُ العَدَاد التصاعدي تنازلي Center-Aligned Mode:

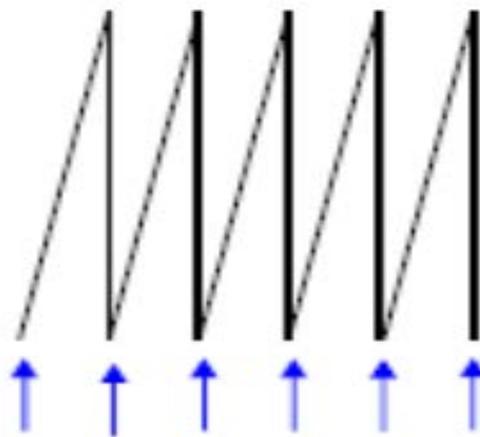
في هذا النمط فإن العداد يبدأ بالعد التصاعدي من الصفر ويستمر بالعد مع كل نبضة قادمة على قطب الدخل حتى يصل إلى القيمة المخزنة سابقاً-
auto-reload value في المسجل TIMx-ARR ثُم يبدأ بالعد التنازلي من القيمة المخزنة حدث الطفhan Overflow event ثُم يبدأ بالعد التنازلي من القيمة المخزنة سابقاً auto-reload value مع كل نبضة قادمة على قطب الدخل و حتى يصل إلى الصفر عندها يتم توليد حدث Underflow ثُم يعود للعد التصاعدي من الصفر وهكذا...،

Center Aligned

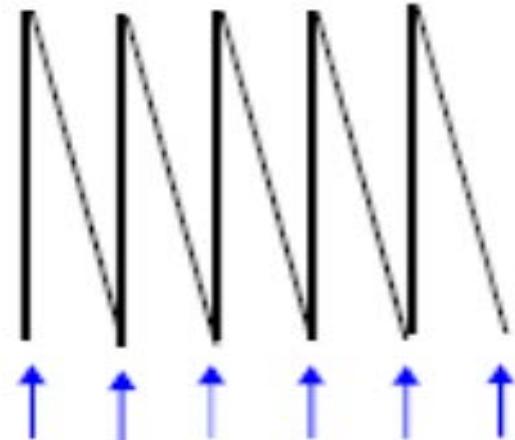


نُمْطُ الْعَدَاد Counter

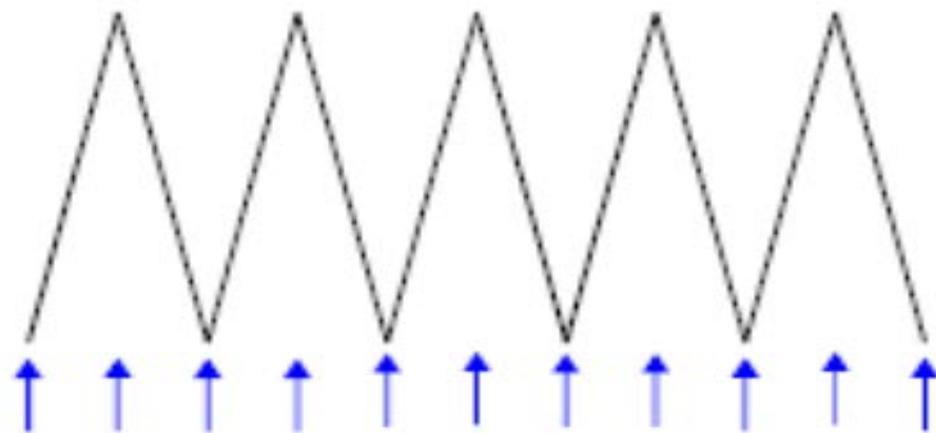
Up counting



Down counting

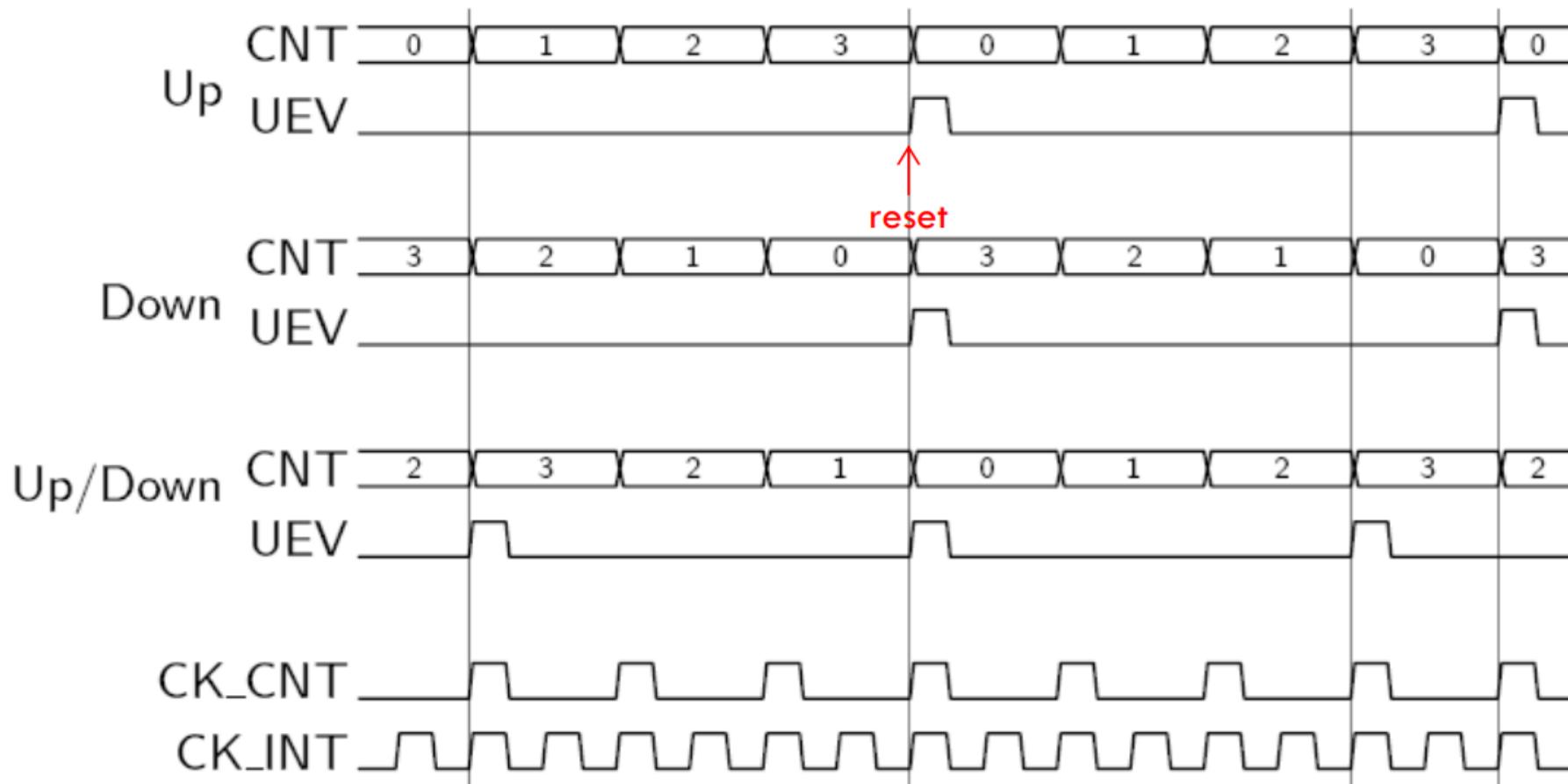


Center Aligned



نُمْطُ الْعَدَاد Counter

وَيَكُونُ الْمُخْطَطُ الزَّمْنِيُّ لِأَنْمَاطِ الْعَدِ الْثَّلَاثَ :

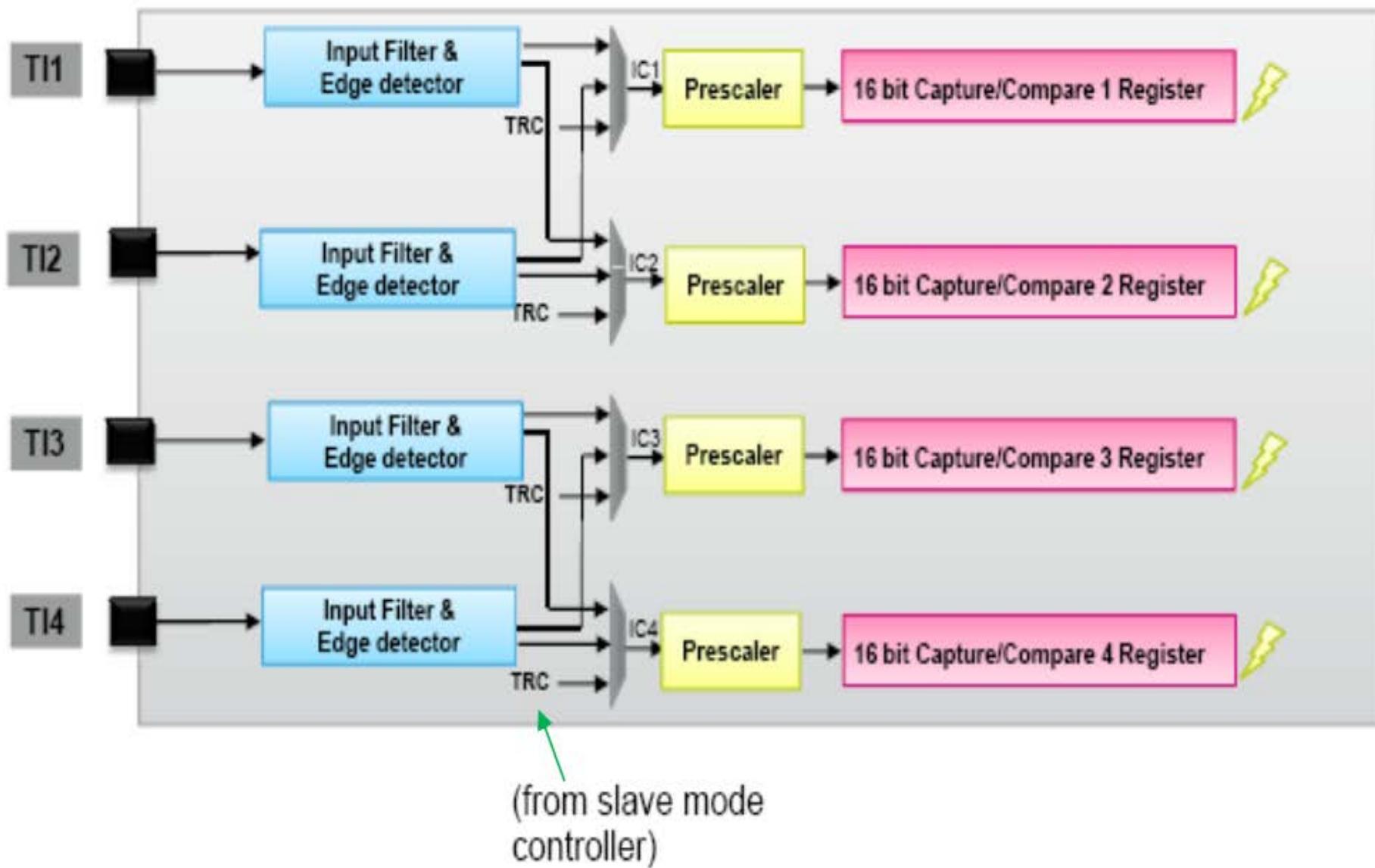


Counter Modes (ARR=3, PSC=1)

نمط Input Capture mode

- في نمط Input Capture يستقبل المؤقت نبضات الساعة الخاصة به من مصدر داخلي (ساعة المتحكم بعد استخدام المقسم التردددي)
- يستمر بالعد إلى أن يحدث حدث معين (جبهة صاعدة/ جبهة هابطة) على قطب المتحكم الخاص بقناة الـ Input Capture عندها يتم حفظ القيمة التي وصل إليها المؤقت إلى Channel 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

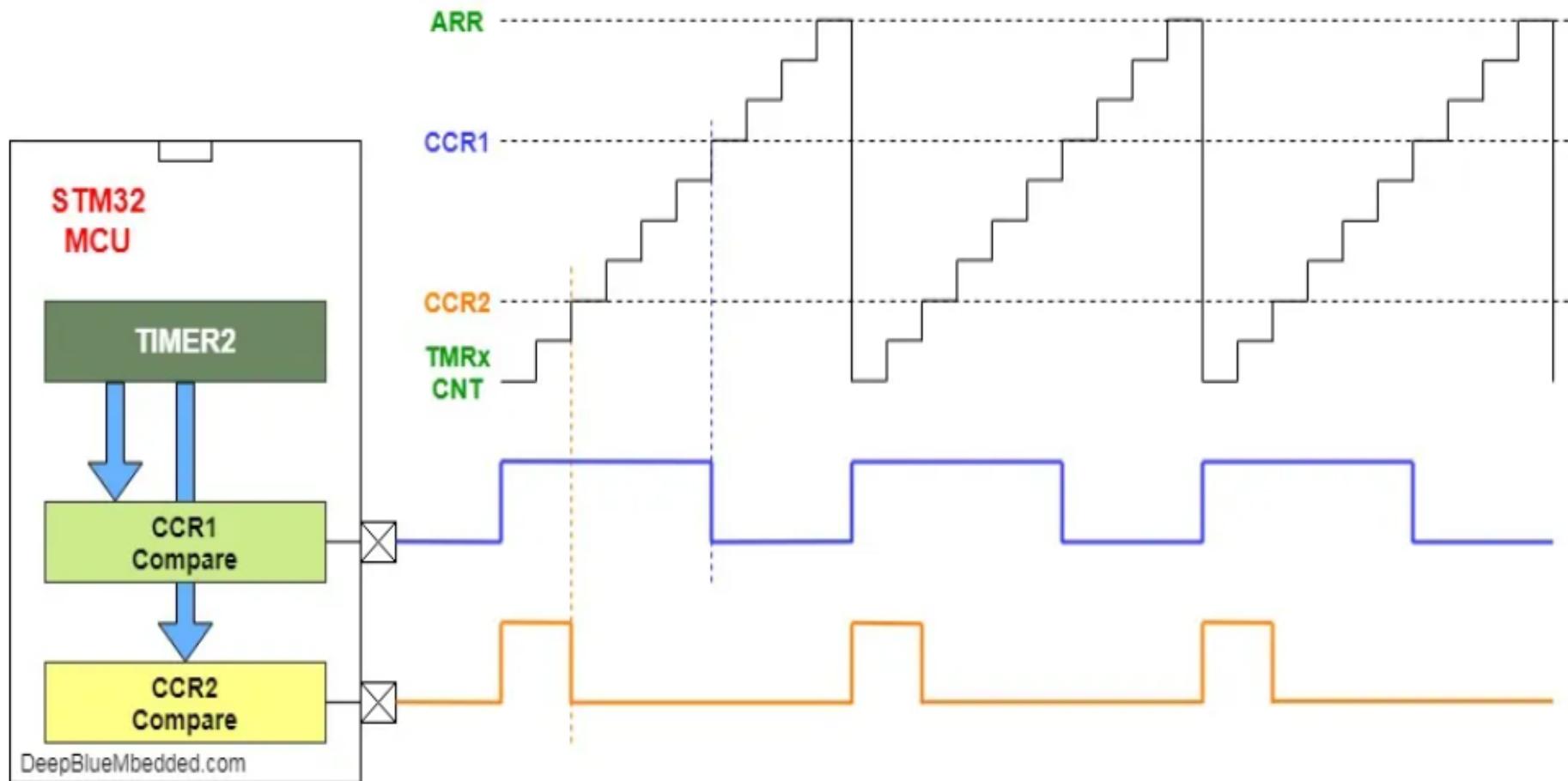


نُمط PWM mode

- 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

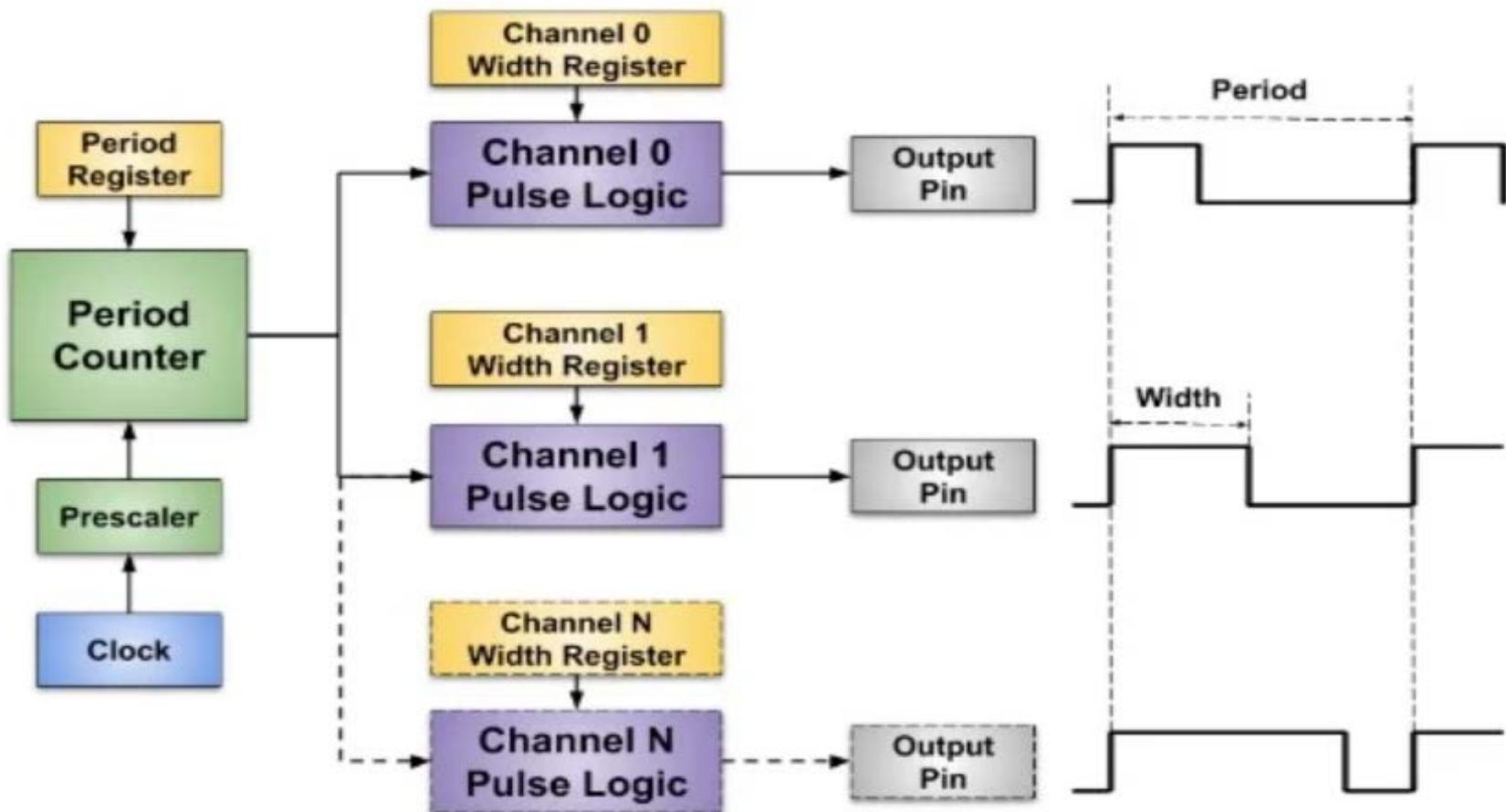
نُمط PWM mode

□ 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



نُمط 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:

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

نُمط PWM mode

مثال:

• ARR=65535 • Prescaler=1 • MHZ F_{CLK} 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

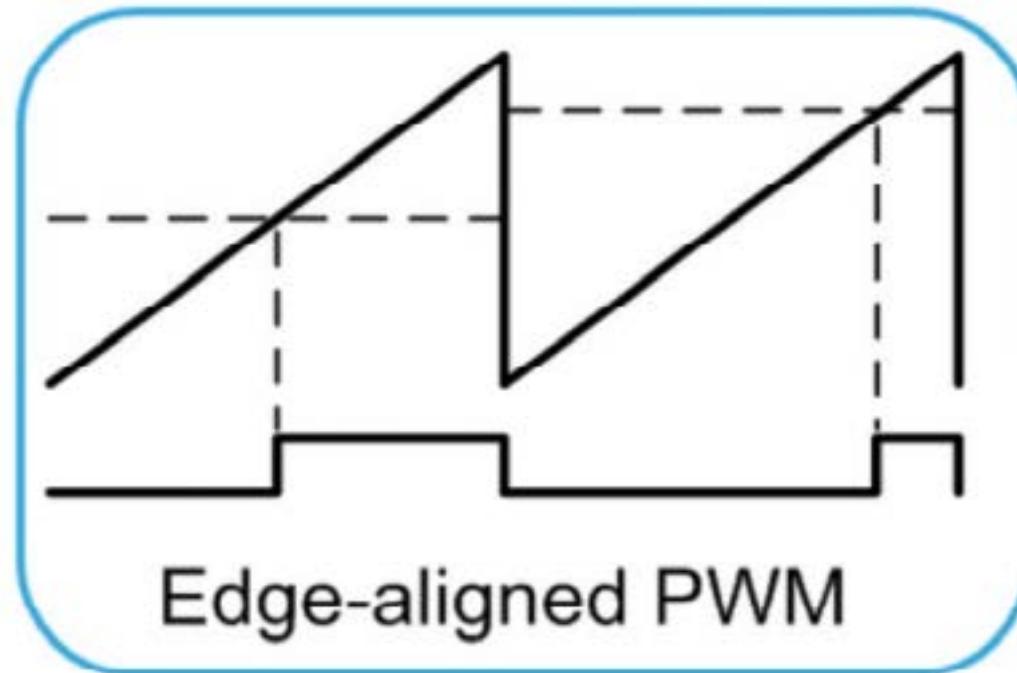
عند عمل المؤقت بنمط PWM وتوليد النبضات في وضع الـ edge-aligned mode up-counting، فإن دورة التشغيل يتم حسابها من خلال العلاقة التالية:

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

نُمَط PWM mode

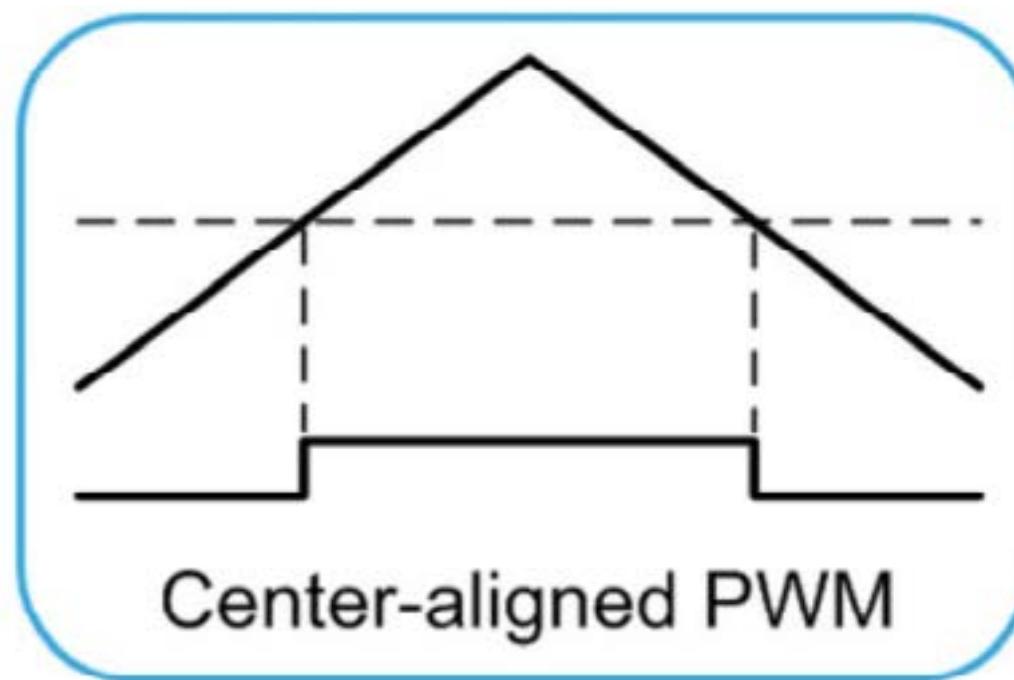
أنماط الـ PWM المختلفة:

□ **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.

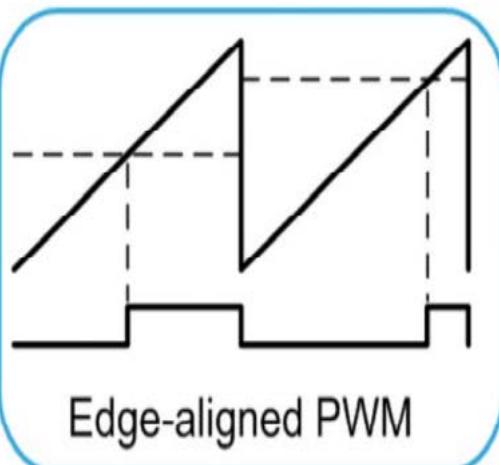


أنماط الـ PWM المختلفة:

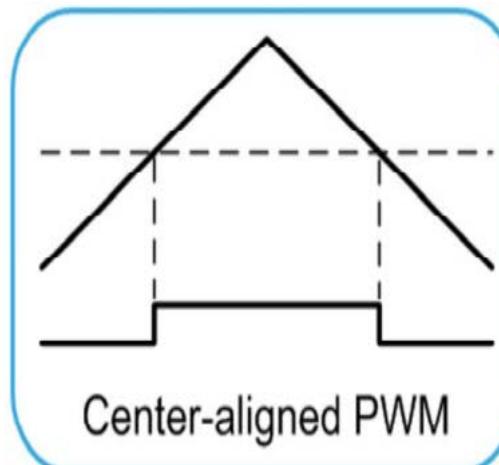
□ **Center-aligned mode** في هذا النمط فإن العداد يقوم بالعد بشكل تصاعدي/تنازلي، وتكون إشارات الـ PWM الناتجة من مؤقت واحد غير متزامنة لأن الجبهة الهاابطة لكل منها مختلفة، لذا فإن أزمنة التبديل لكل إشارة PWM تكون مختلفة عن الإشارة الأخرى



PWM mode نمط

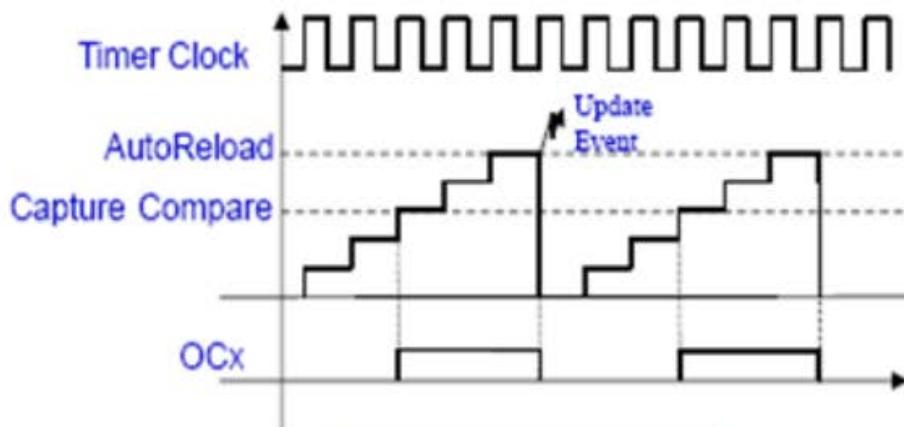


Edge-aligned PWM



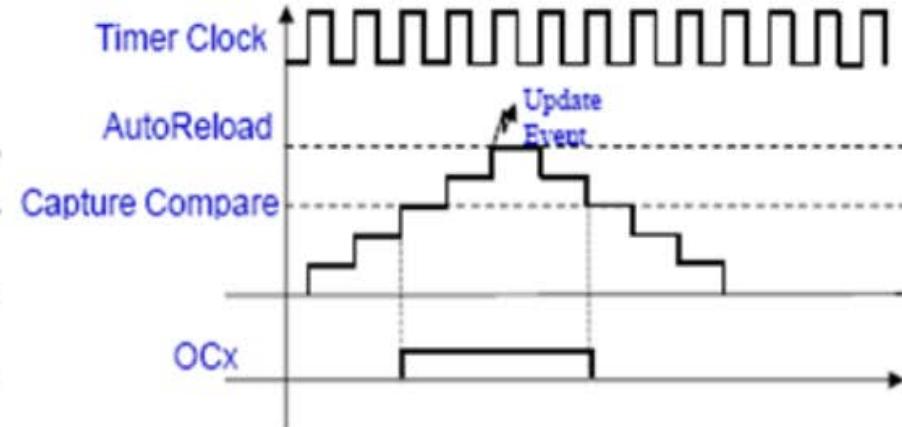
Center-aligned PWM

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();`

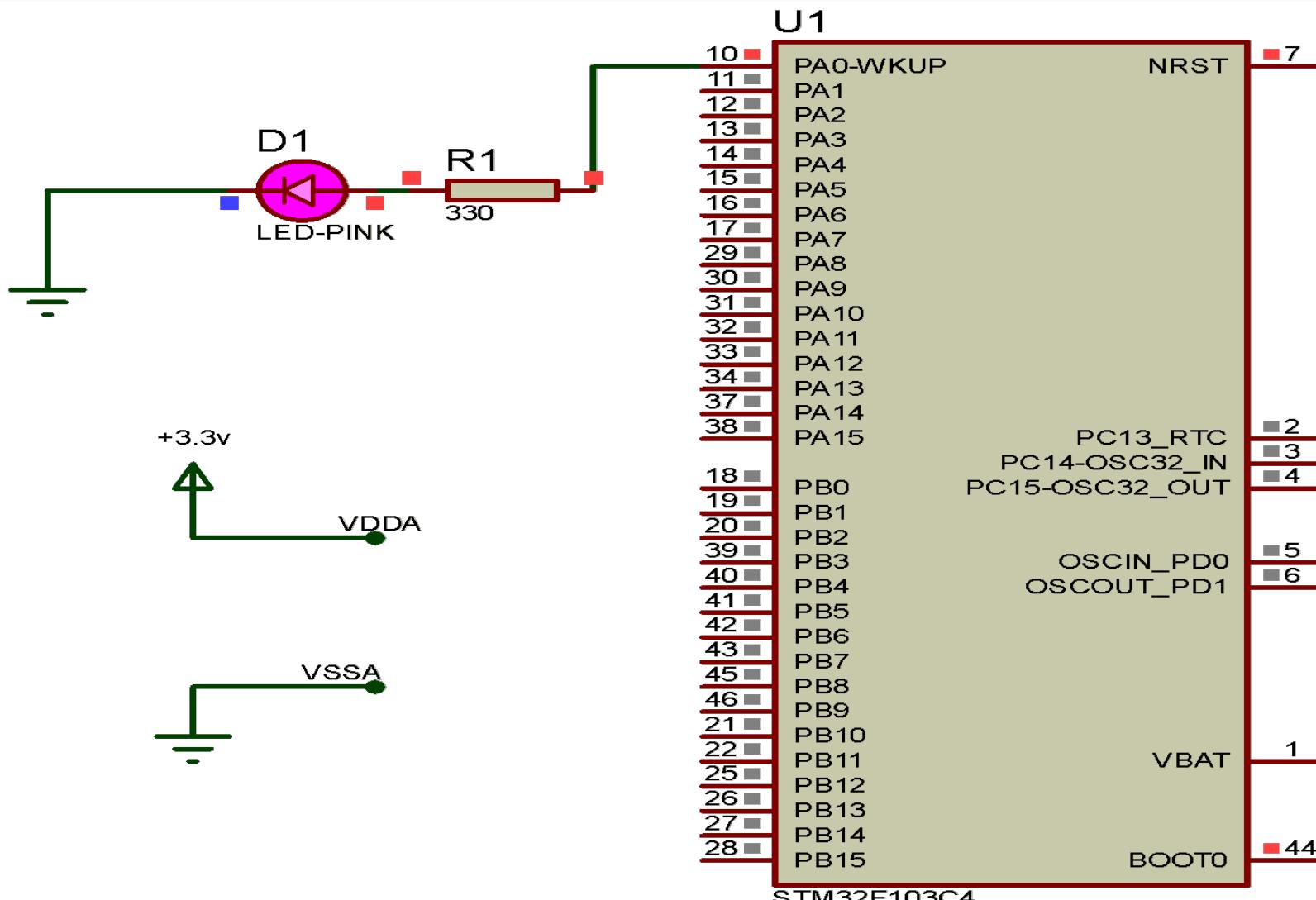
وضع Interrupt □: في هذا الوضع عند الوصول إلى Overflow/underflow أو أي من أحداث المقاطة سيتم التوجّه آلياً لتنفيذ برنامج خدمة المقاطة، وهذا الوضع الذي سستخدمه في جميع التطبيقات القادمة.

توفر مكتبة HAL الدالة التالية لبدء المؤقت في وضع المقاطة:

`HAL_TIM_Base_Start();`

وضع DMA □

التطبيق العملي : استخدام المؤقت في نمط Timer mode وبوضع المقاطة لتوليد زمن بدلًا من استخدام دالة delay() واستخدامه في عملToggle للب الموصول على القطب PA5



التطبيق العملي : استخدام المؤقت في نمط Timer mode وبوضع المقاطة لـ توليد زمن بدلًا من استخدام دالة delay() واستخدامه في عملToggle لـ توليد الموصول على القطب PA5

ختار القطب PA5 لضبطه كقطب خرج

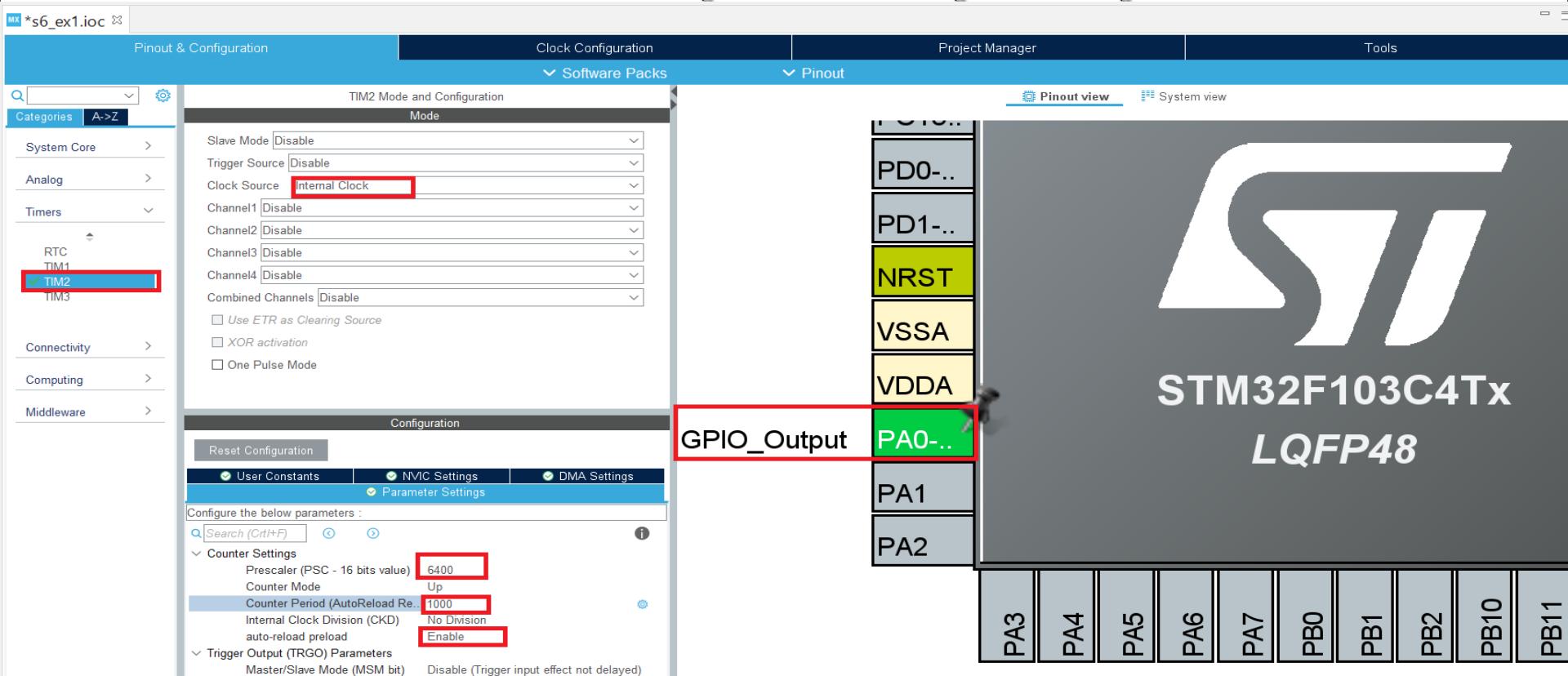
نقوم بضبط إعدادات المؤقت كي نحصل على زمن 100 msec
لعكس حالة اليد الموصول على القطب رقم 5 من المنفذ A، من المعادلة السابقة سنفترض أن تردد ساعة المتحكم هي 8 MHz
والمقسم التردد 8000 بقي فقط حساب (Preload) Period
بتغيير القيم في المعادلة :

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

$$Preload = 100$$

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

نقوم بتفعيل مقاطعة المؤقت من شريط الـ NVIC tab

System Core >

Analog >

Timers > ▾

- LPTIM1
- LPTIM2
- RTC
- TIM1
- TIM2**
- TIM3
- TIM6
- TIM7
- TIM14
- TIM15
- TIM16
- TIM17

Connectivity >

Multimedia >

Computing >

Slave Mode

Trigger Source

Clock Source

Channel1

Channel2

Channel3

Channel4

Combined Channels

Use ETR as Clearing Source

XOR activation

One Pulse Mode

Configuration

User Constants
 NVIC Settings
 DMA Settings

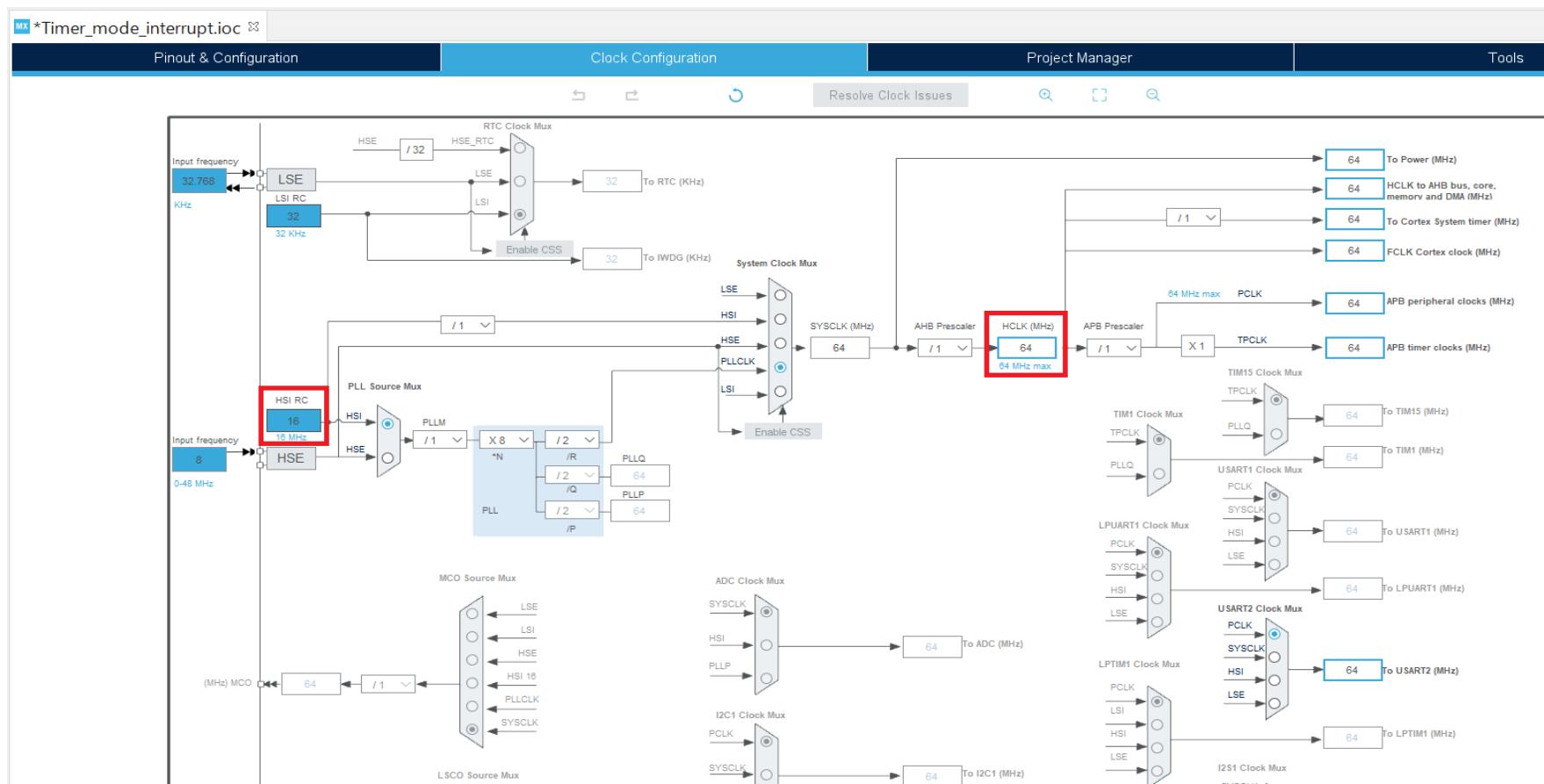
Parameter Settings

NVIC Interrupt Table	Enabled	Preemption Priority
TIM2 global interrupt	<input checked="" type="checkbox"/>	0

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

نختار Simulation

نقوم بضبط تردد ساعة المتحكم: أثناء الـ Toggling تردد الساعة Mhz8



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

□ 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)  
{
```

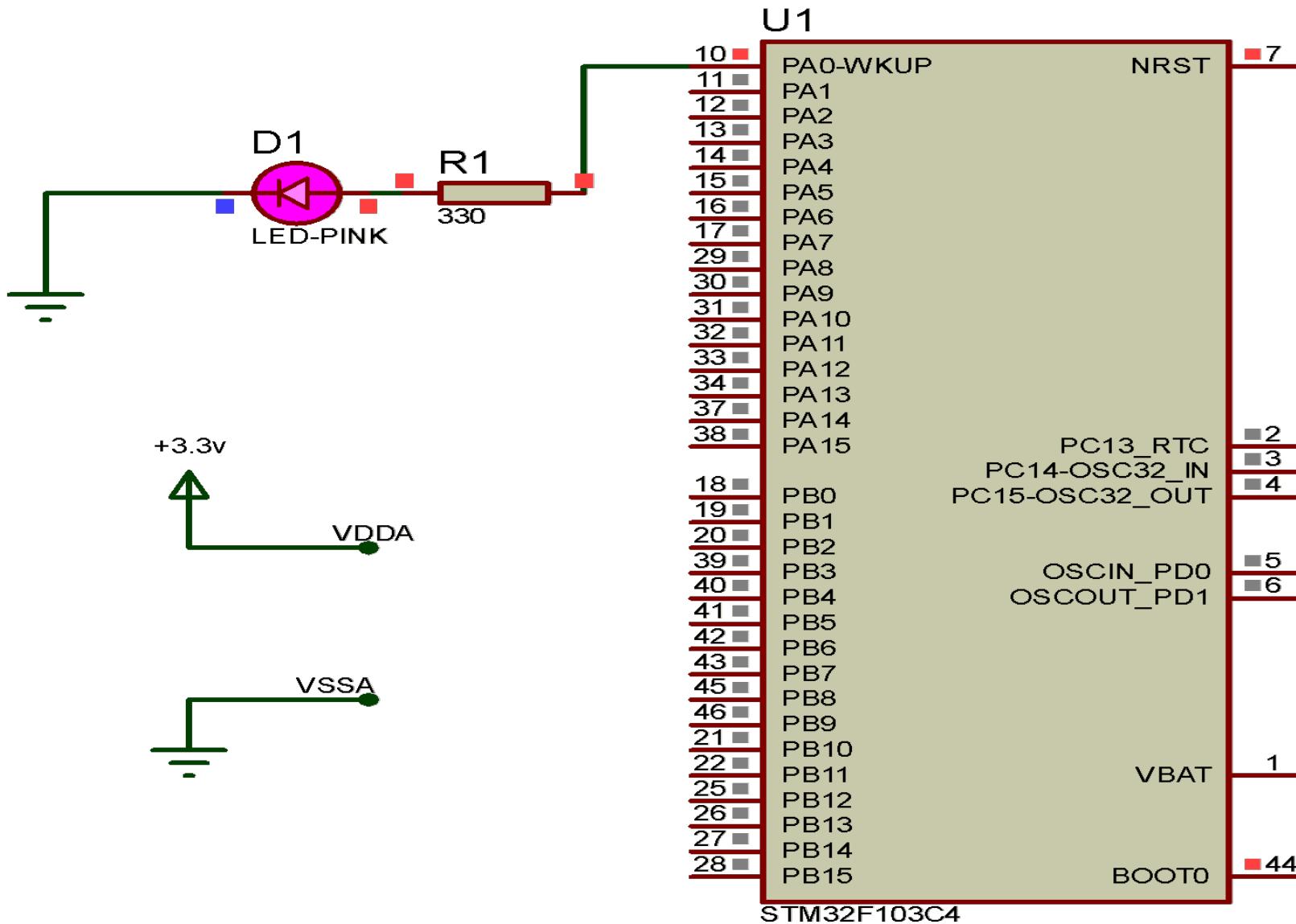
التطبيق العملي : استخدام المؤقت في نمط Timer mode وبوضع المقاطعة لتوليد زمن بدلًا من استخدام دالة delay() واستخدامه في عمل Toggle لـ لـيد الموصول على القطب PA5

```
HAL_Init();
SystemClock_Config();
MX_GPIO_Init();
MX_TIM2_Init();
HAL_TIM_Base_Start_IT(&htim2);
while (1)
{
}
```

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

```
void HAL_TIM_PeriodElapsedCallback(  
TIM_HandleTypeDef* htim)  
{  
HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_5);  
}
```

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 ، فيصبح التردد HZ 488.25
- التحكم بدورة التشغيل 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، نضبط القيمة العظمى للمسجل على القيمة 65535 Auto لتصبح تردد إشارة PWM هو 488.25 HZ، نفعل خاصية Reload preload ونختار نمط إشارة الـ PWM

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

The screenshot shows the STM32CubeMX software interface for configuring the STM32F103C4Tx microcontroller. The project is named "s6_ex2.ioc".

Pinout & Configuration tab (selected):

- Mode** section:
 - Slave Mode: Disable
 - Trigger Source: Disable
 - Clock Source: Internal Clock (highlighted with a red box)
 - Channel1: PWM Generation CH1 (highlighted with a red box)
 - Channel2: Disable
 - Channel3: Disable
 - Channel4: Disable
 - Combined Channels: Disable
 - Use ETR as Clearing Source
 - XOR activation
 - One Pulse Mode
- Configuration** section:
 - Reset Configuration
 - Checkboxes: NVIC Settings, DMA Settings, GPIO Settings, Parameter Settings, User Constants
 - Configure the below parameters:
 - Prescaler (PSC - 16 bits value): 0
 - Counter Mode: Up
 - Counter Period (AutoReload R.): 65535 (highlighted with a red box)
 - Internal Clock Division (CKD): No Division
 - auto-reload preload: Enable (highlighted with a red box)
 - Trigger Output (TRGO) Parameters:
 - Master/Slave Mode (MSM bit): Disable (Trigger input effect not delayed)
 - Trigger Event Selection: Reset (UG bit from TIMx_EGR)
 - PWM Generation Channel 1

Clock Configuration tab: Not selected.

Project Manager tab: Not selected.

Tools tab: Not selected.

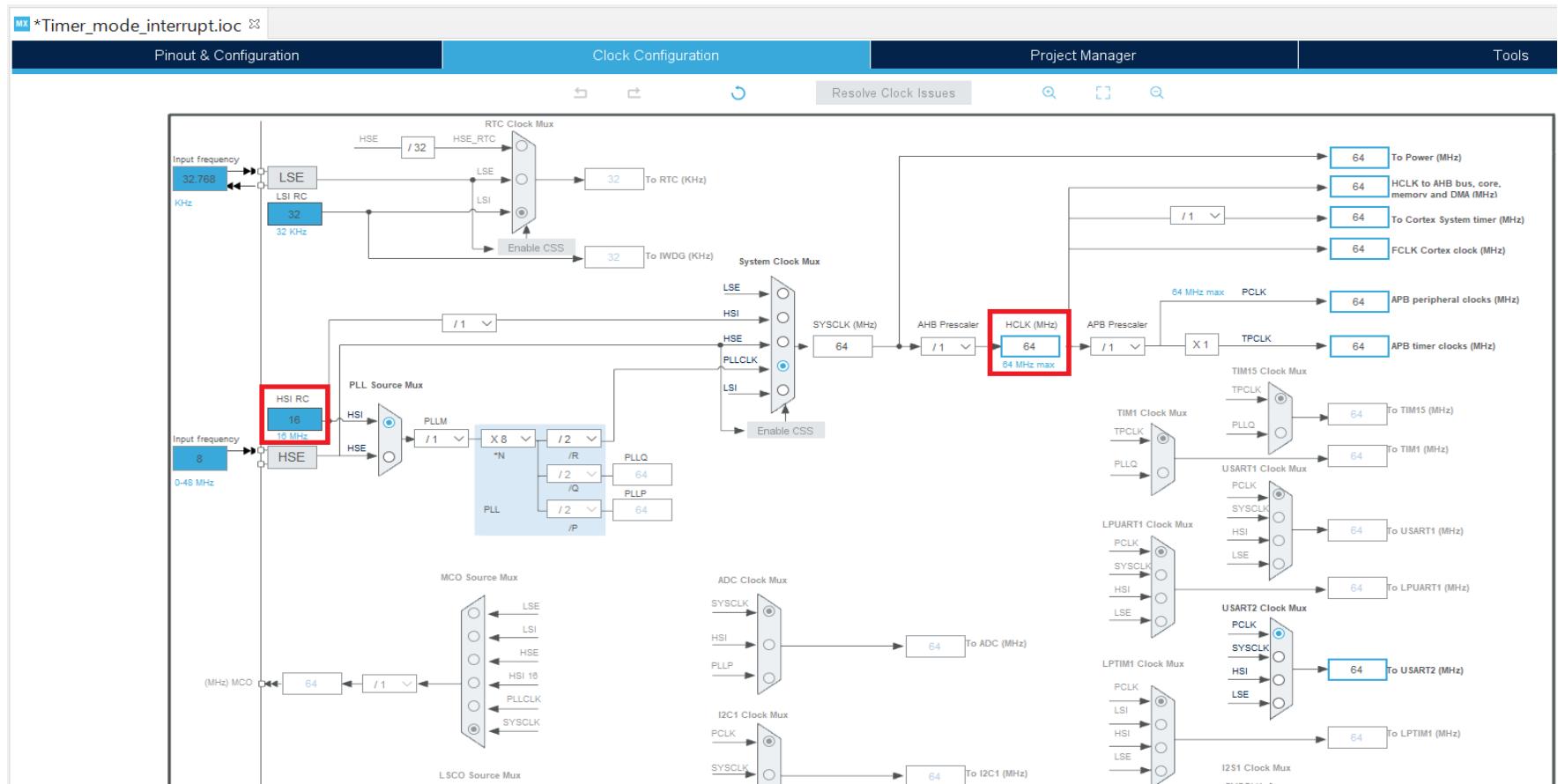
Pinout tab (selected): Shows the pinout for the STM32F103C4Tx LQFP48 package. The pins are grouped by function:

- VBAT
- PC13..
- PC14..
- PC15..
- PD0..
- PD1..
- NRST** (highlighted with a yellow box)
- VSSA
- VDDA
- PA0..** (highlighted with a green box)
- PA1
- PA2
- PA3
- PA4
- PA5
- PA6
- PA7
- PA8
- PA9
- PA10
- PA11
- PA12
- PA13
- PA14
- PA15
- PB1
- PB2
- PB10
- PB11
- VSS
- VDD

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

نختار Simulation

نقوم بضبط تردد ساعة المتحكم: أثناء الـ **64MHZ**



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

□ 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)
{
```

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

```
while (1)
```

```
{
```

```
    while(CH1_DC < 65535)
```

```
{
```

```
        TIM2->CCMR1 = CH1_DC;
```

```
        CH1_DC += 70;
```

```
        HAL_Delay(1);
```

```
}
```

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

```
while(CH1_DC > 0)
{
    TIM2->CCMR1 = CH1_DC;
    CH1_DC -= 70;
    HAL_Delay(1);
}
{
}
}
```

Thank you for listening