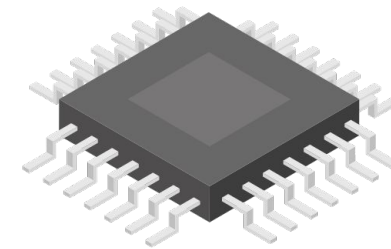






Microcontroladores



Prof.º: Pablo Jean Rozário



pablo.jean@padotec.com.br



/in/pablojeanrozario



<https://github.com/Pablo-Jean>

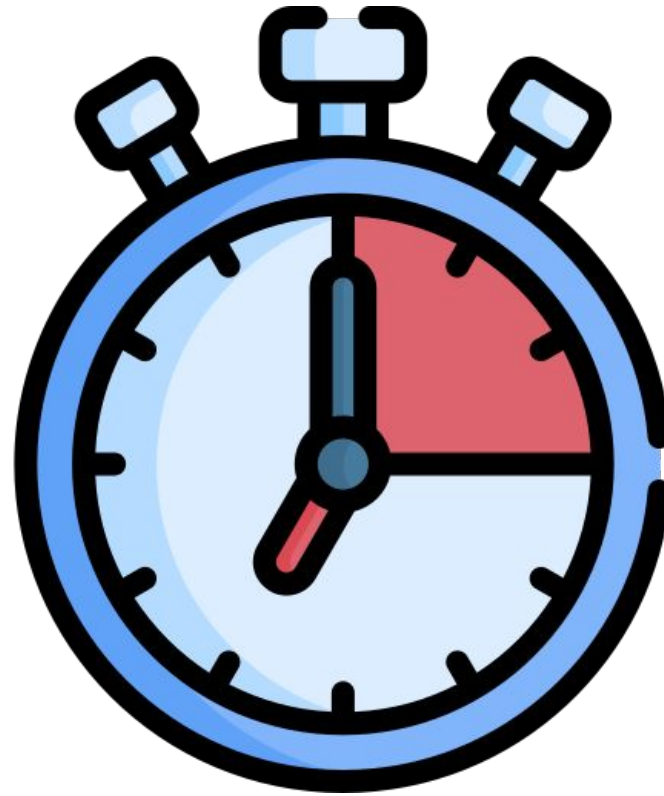
Timer e RTC

Índice da Aula #8



- Introdução
- Modo Temporizador
- Modo Contador
- Interrupções
- RTC
- Timers do STM32G0
- Modos de Operação
- Localização dos terminais
- Funções Utilizadas
- Lista de Exercícios #8

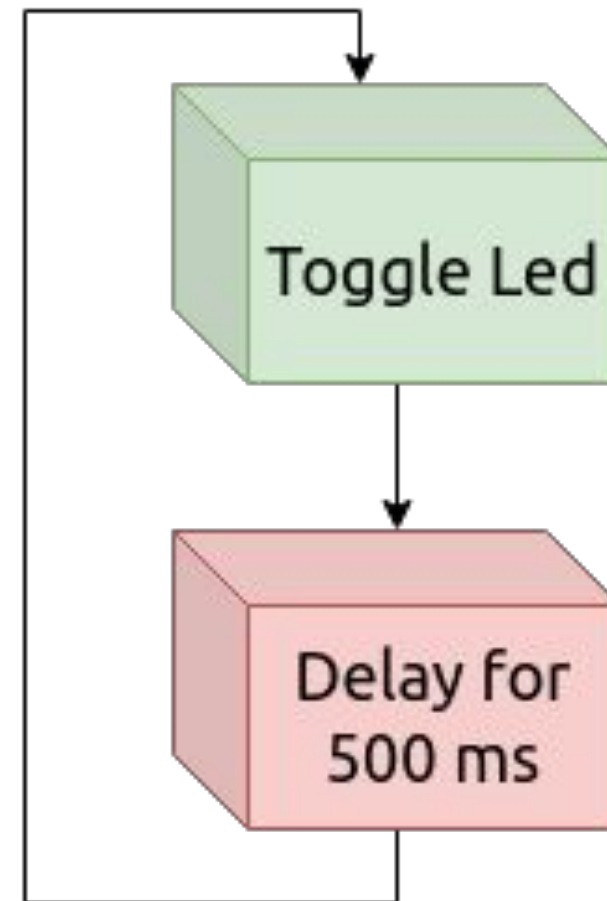
Timers



Problemática

Funções de *delay*, como `HAL_Delay()`, são muito utilizadas para gerar temporização em programas, como acender um LED.

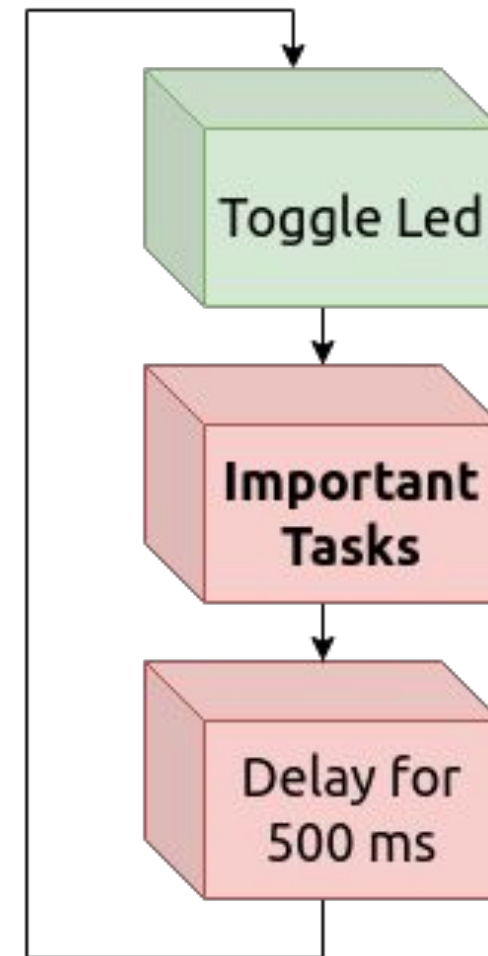
Mas note que o *delay* causa uma ociosidade no programa, onde se desperdiça ciclos de *clock*.



Problemática

Se o programa precisar executar outras atividades ou até mesmo entrar em *sleep* para economia de energia, o delay é inviável e traz problemas.

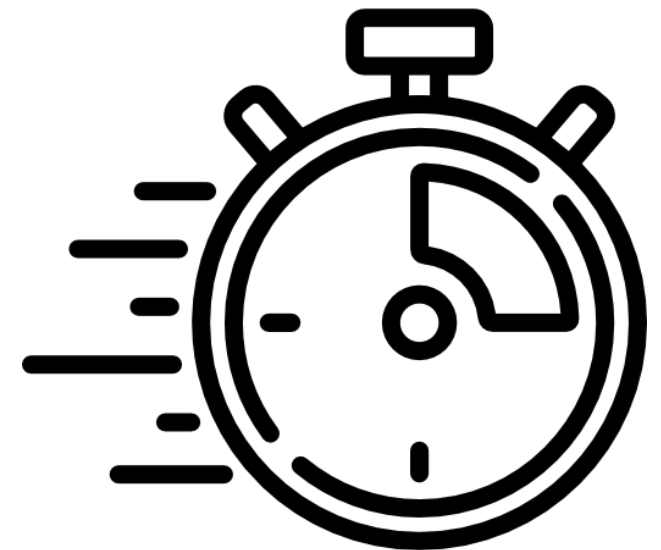
E por isto, estudaremos os *timers*, que resolvem este problema.



Introdução

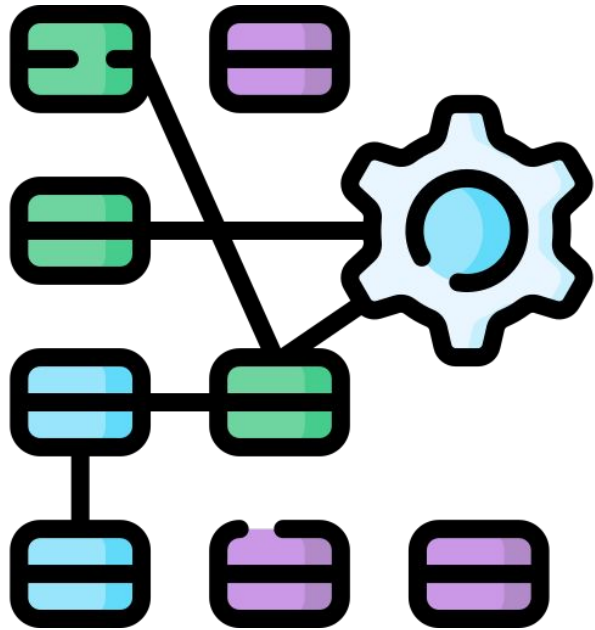
Um periférico de *timer* é a forma mais **básica** de realizar uma contagem de ciclos de *clock*. Podendo ser utilizado para gerar **temporização** ou **contagens**.

Possui também funcionalidades mais avançadas como o **PWM**, gerar *triggers* para periféricos como ADC e DAC, geração de *delays* precisos, etc.



Introdução

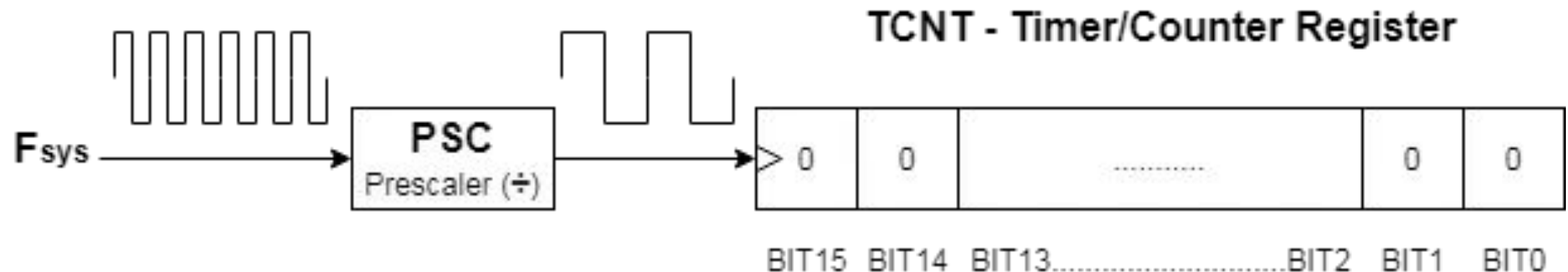
Em geral, os timers implementar um *prescaler* em seu circuito, que nada mais é que um divisor de frequência.



O timer é um dos **principais** periféricos de um microcontrolador, pois é ele que permite que a complexidade de uma aplicação seja **escalada** inclusive a implementação de um **RTOS** (*Real Time Operational System*).

Modo Temporizador

Neste modo, o *timer* é conectado a uma fonte de *clock* (podendo ser interna ou externa), gerando um incremento ou decremento periódico do registrador de contagem.



Modo Temporizador - Equação



Como notamos na imagem anterior, o timer possui um prescaler que irá dividir a frequência de entrada. Cada bit do registrador será incrementado por um período dado pela equação:

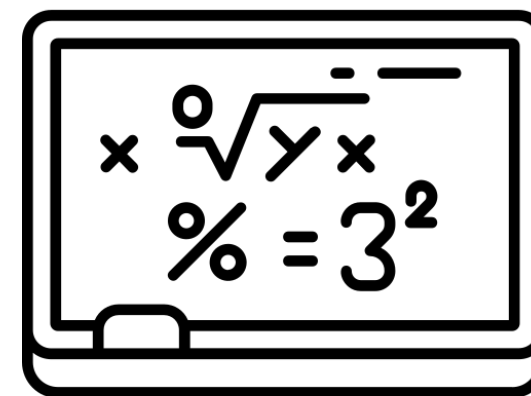
$$T_{psc} = \frac{PSC}{F_{sys}}$$

onde:

T_{psc} é o período de incremento do timer

PSC é o valor do prescaler

F_{sys} é a frequência de entrada do *prescaler*



Modo Temporizador - Equação



E para um timer atingir o valor máximo, iniciando de zero, temos de multiplicar pelo valor máximo e retornar a zero, utilizamos a equação:

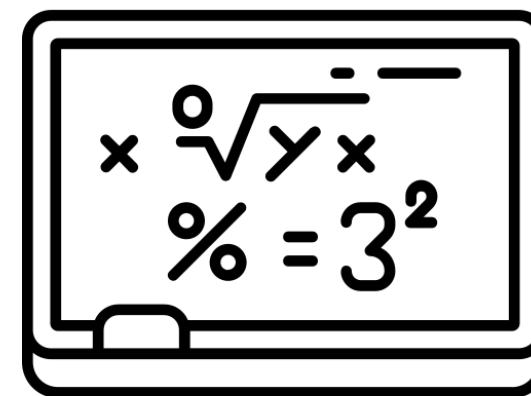
$$T_{timer} = T_{psc} * (2^N + 1)$$

onde:

Ttimer é o tempo para o registrador ir de 0 até o valor máximo e retornar a 0.

Tpsc é o período de incremento/decremento do *prescaler*

N é a quantidade de *bits* do *timer*

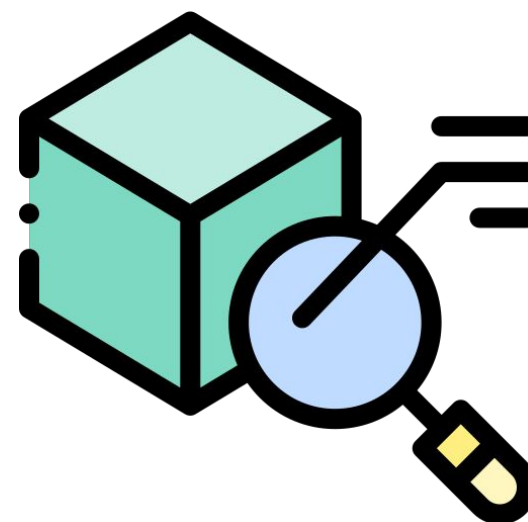


Modo Temporizador - Detalhes



A depender da arquitetura do *timer*, é possível programa-lo para gerar *overflow* em um valor determinado pelo desenvolvedor.

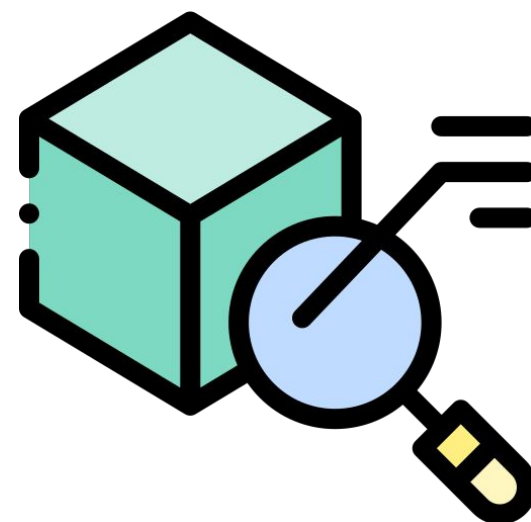
Outros *timer* requerem que você inicie-o com um valor para gerar o *overflow* no valor máximo, chamado de ***preload***.



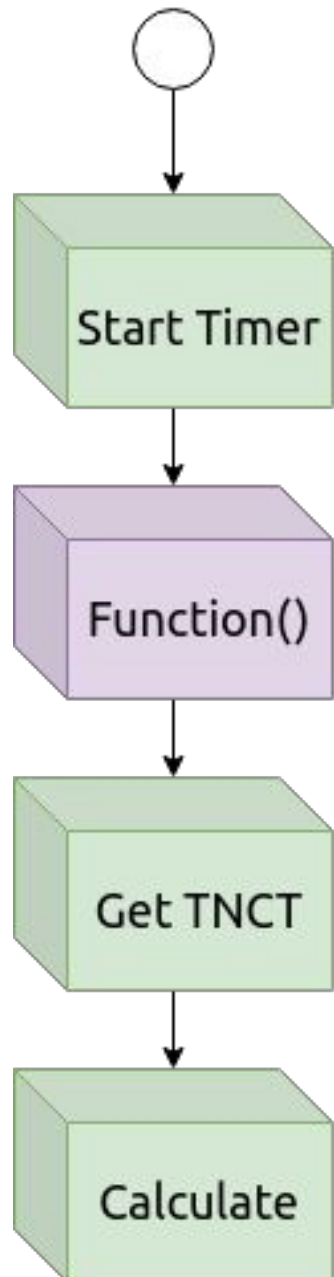
Modo Temporizador - Detalhes



Além disto, o registrador contador pode também ser lido pelo programa, com isto gerando várias outras aplicações, como *delays* precisos e, ainda, determinar qual o impacto de uma função, em termos de tempo, de uma função.



Modo Temporizador Detalhes



Zera o contador do timer e inicia a contagem

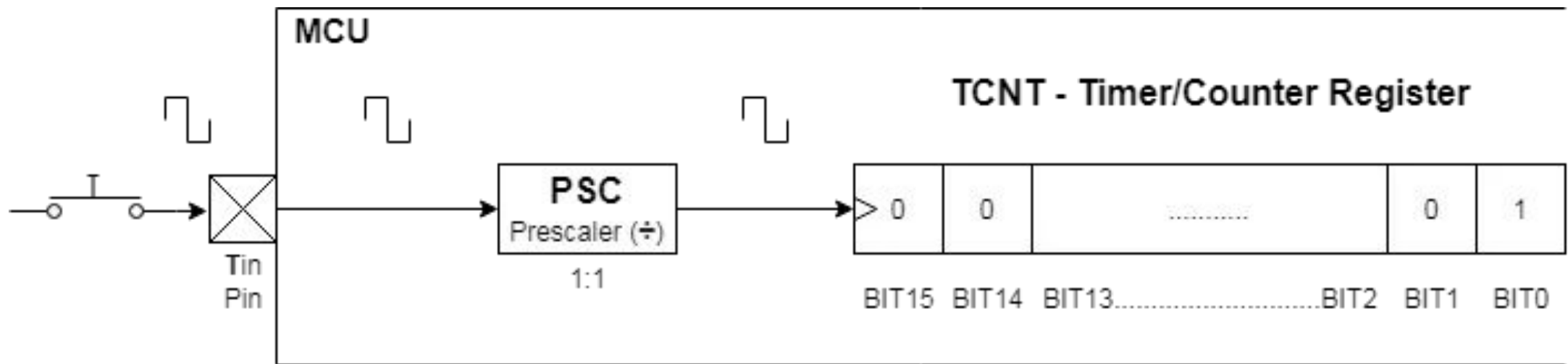
Chama a função que desejamos medir o tempo de execução

Realiza a leitura do registrador contador do *timer*

Se necessário, calcula o valor em termos de tempo

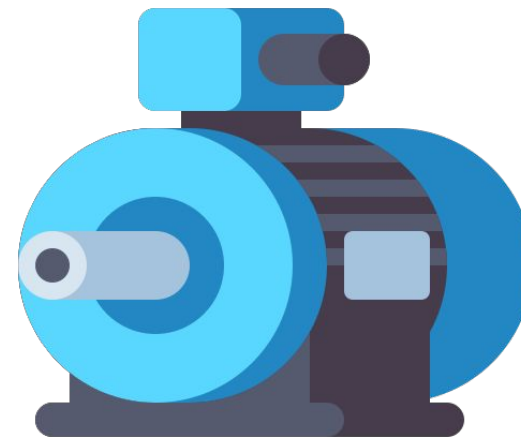
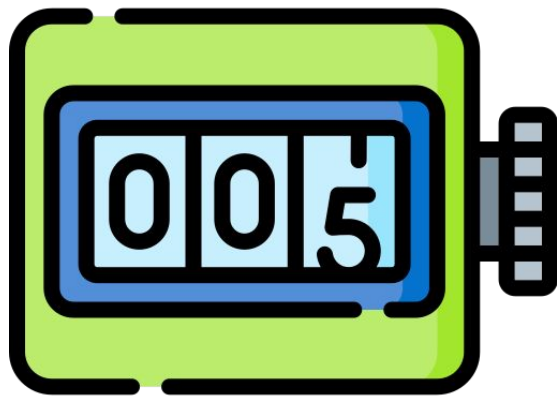
Modo Contador

Neste modo a entrada de *clock* do *timer* é ligada uma **GPIO** do microcontrolador, permitindo que pulsos externos sejam contados. Por exemplo, um botão.



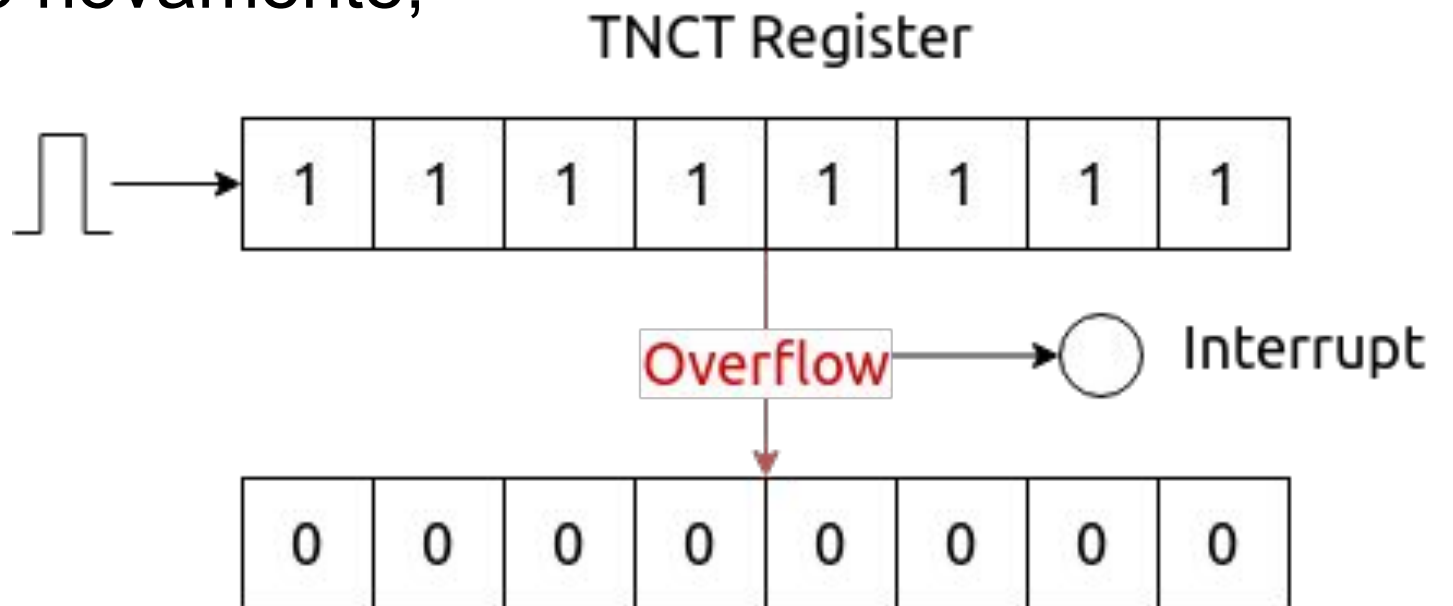
Modo Contador

Com isto, podemos ter aplicações como Contar quantas vezes um botão foi pressionado, implementar um tacômetro (medidor de rotação de um motor). etc.



Interrupções

Em geral, um *timer* sempre gera uma interrupção em um evento de *Overflow*, em alguns casos mais específicos de *Underflow*. O *overflow* ocorre quando o timer chega ao seu valor máximo e o mesmo é incrementado novamente, retornando a zero e acionando uma *flag*.



Interrupções

Algumas plataformas permitem também que, além da interrupção, sejam gerados **eventos** para acionar outros periféricos. Como por exemplo, iniciar uma conversão ADC com uma frequência fixa.



RTC

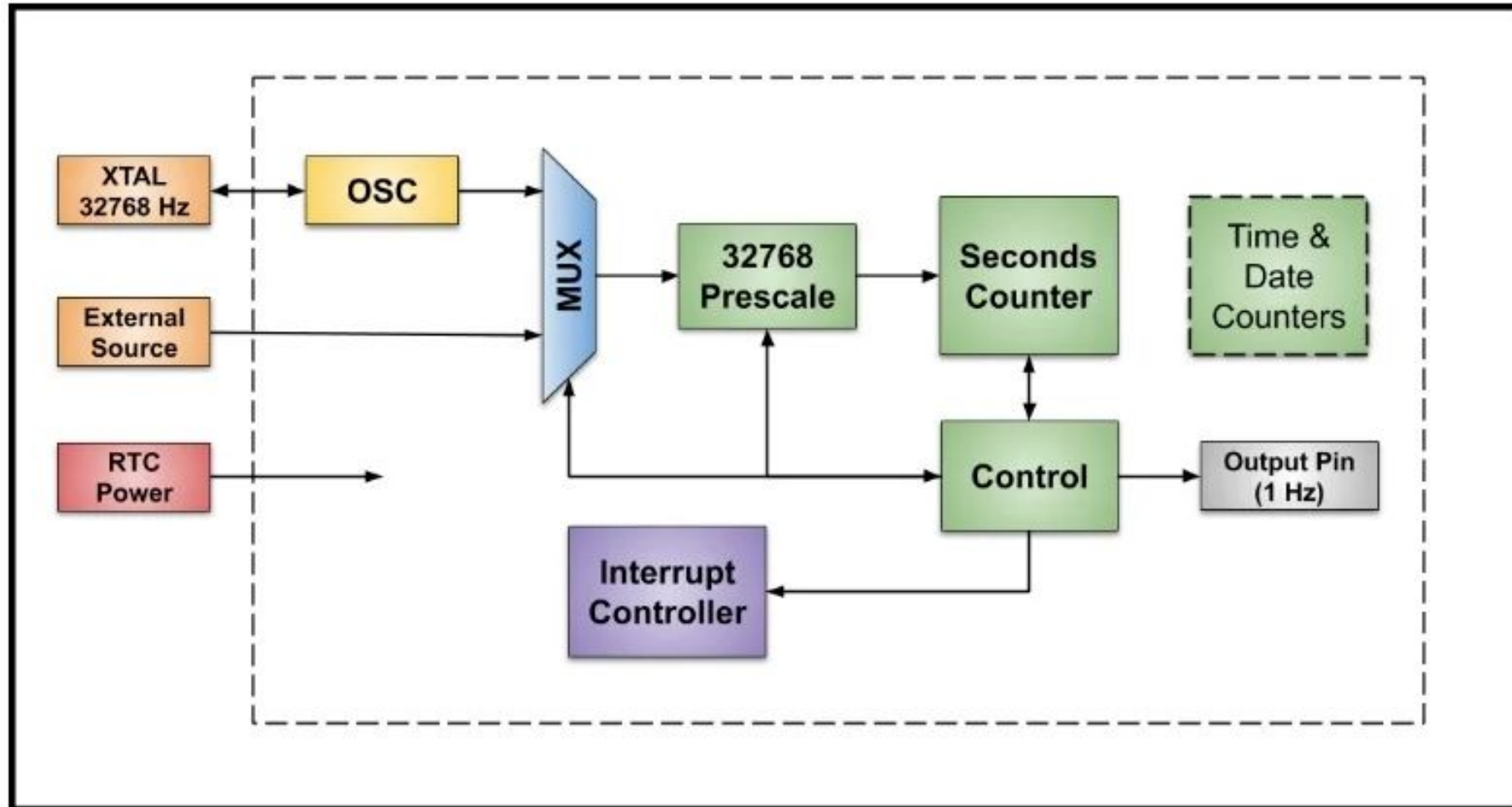


O RTC, ou *Real-time Clocks*, são um tipo especial de *timer* dedicado a manter uma base de tempo precisa de 1 segundo, ou ainda, um calendário.



Muito empregados quando há necessidade de implementar relógios e alarmes.

RTC



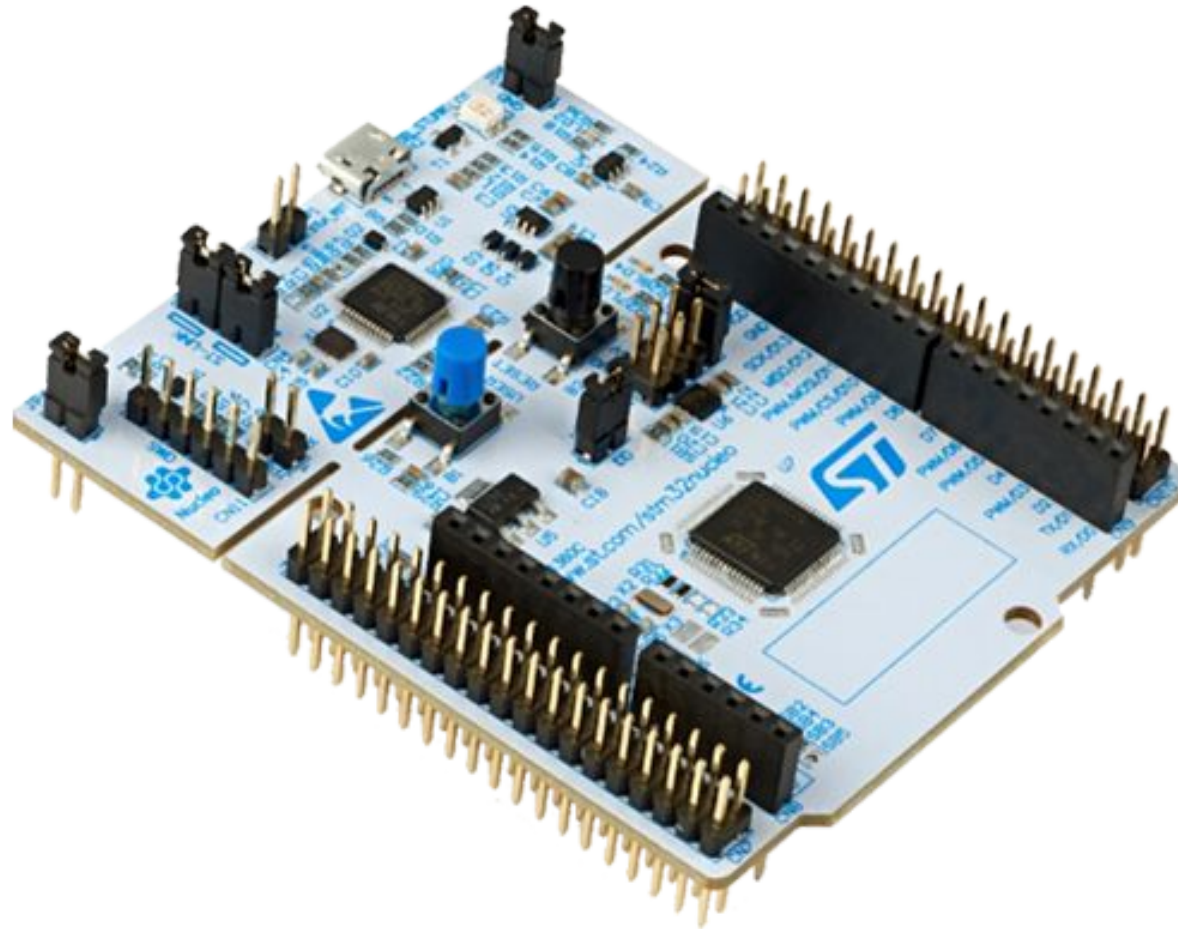
RTC - 32.768 kHz



As fontes de *clock* de um RTC operam, quase sempre, com cristal de baixa frequência de **32.768 kHz**, mas por quê? Porque uma baixa frequência gera um menor **consumo** energético além de mais **estáveis**. Além de que um circuito que gera pulsos a cada 1s com esta frequência são simples, baratos e consomem pouco.

Isto acontece pois 32.768 é múltiplo de 2, sendo barato um prescaler para esta frequência.

Timers no STM32G0

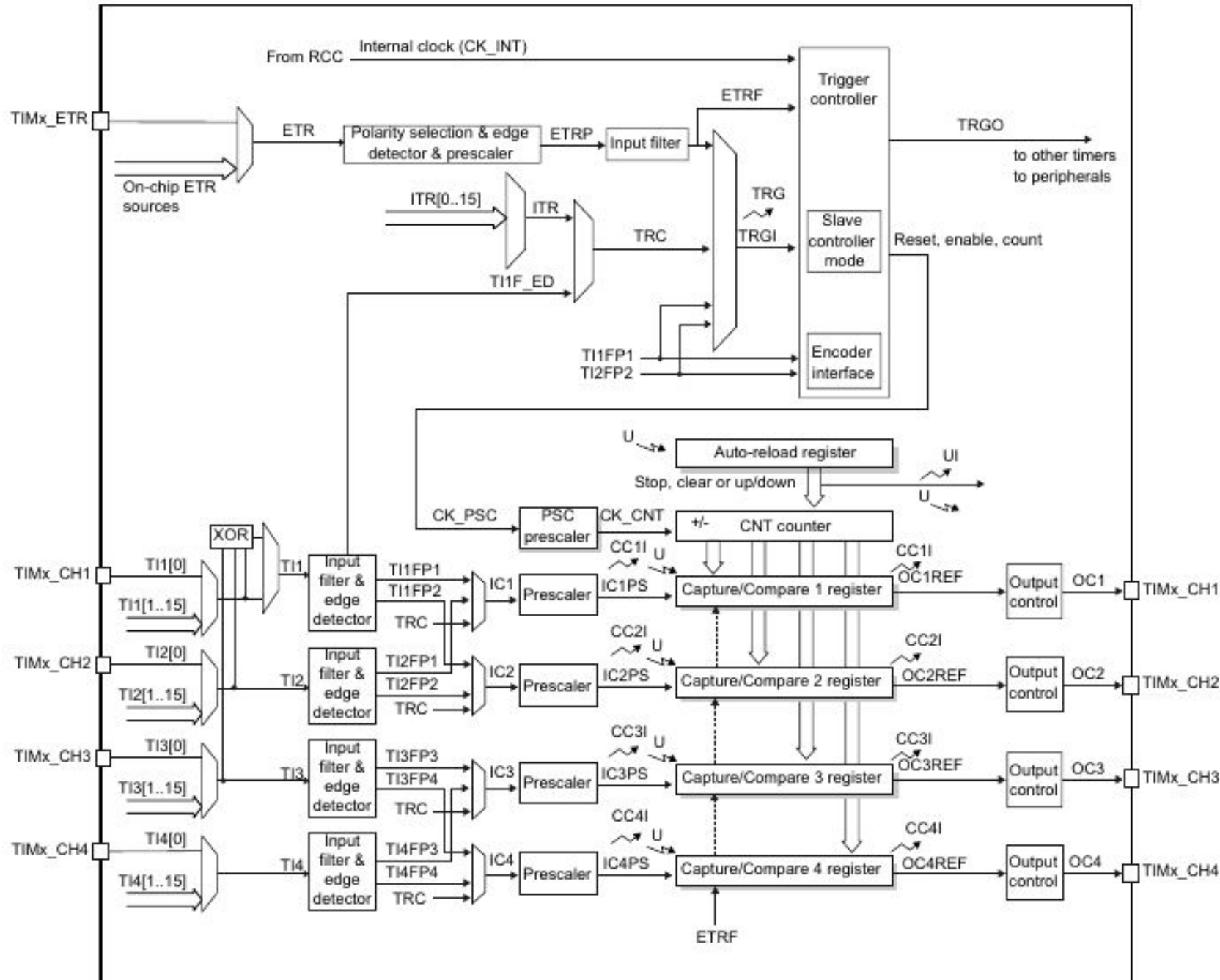


STM32 - Características do Timer

Um timer genérico do STM32G0 possui as seguintes características



- Contador de *16bits* ou *32bits* (TIM2)
- Opera em *up*, *down* ou *up/down* com registrador de *auto-reload*
- até 4 canais de Input/Output Capture, PWM e One-pulse
- **Entre outras opções como DMA, encoders, sensores hall, etc.**



Tipos de Timers

Os STM32s possuem diferentes *timers*, sendo para nosso **STM32G0B1RE** (capítulos referem-se ao **RM0444**):

Advanced-control Timers (**TIM1**) (Cap 21)

General-purpose Timers (**TIM2/TIM3/TIM4**) (Cap 22)

Basic Timers (**TIM6/TIM7**) (Cap 23)

General-purpose Timer (**TIM14**) (Cap 24)

General-purpose Timers (**TIM15/TIM16/TIM17**) (Cap 25)

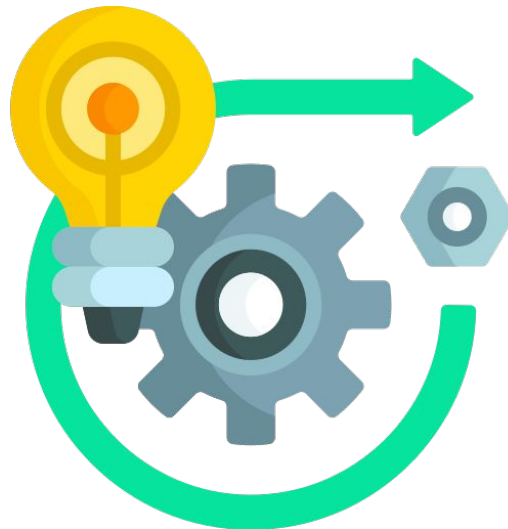
Low-power Timers (**LPTIM**) (Cap 26)

Real-time Clock (**RTC**) (Cap 30)



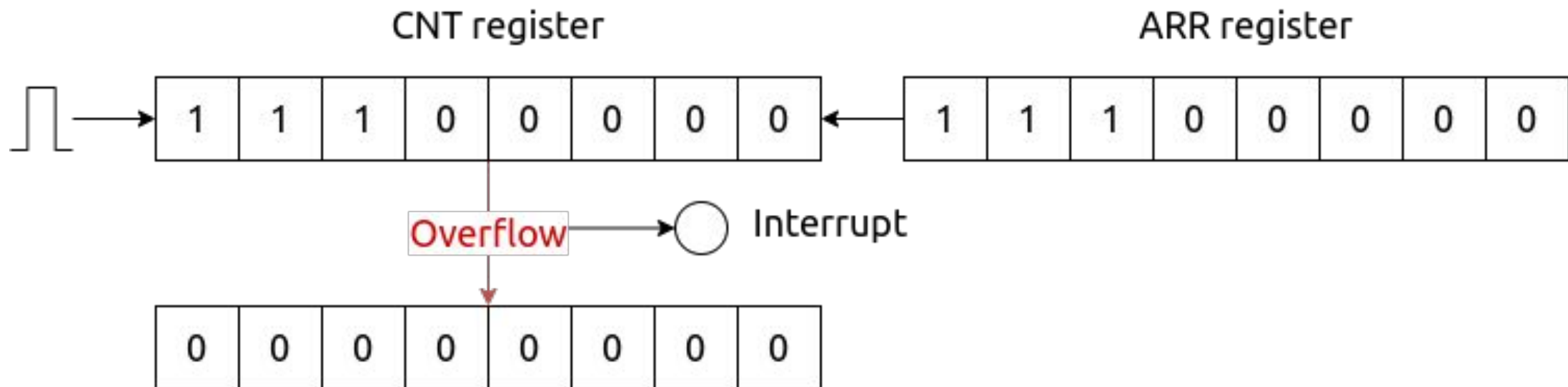
Modos de Operação

Os *timers* do STM32G0 podem operar em diversos modos além do Temporizador e do Contador, a depender também de qual tipo que iremos utilizar, estes modos são:



Timer Mode

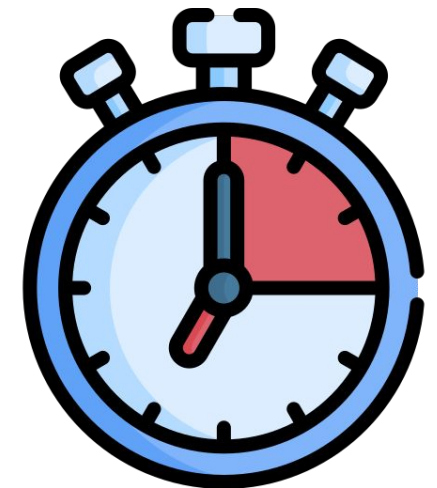
Utilizado para gerar **eventos/interrupções periódicos**. Aqui, quando em modo de contagem para cima (*up*), o *overflow* ocorre quando o contador atinge o valor do **ARR** (*Auto-reload register*), retornando para zero e gerando a interrupção.



Timer Mode

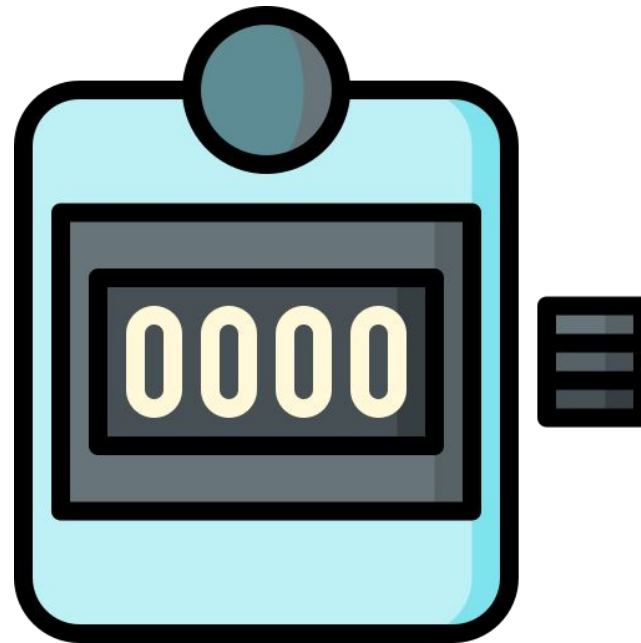
Em modo de contagem da para baixo (*down*), a interrupção ocorre no *Underflow*. Independente destes modos, para calcular a frequência de interrupção do timer, utilizamos a seguinte equação:

$$f_{tim} = \frac{f_{sys}}{(ARR + 1) * (PSC + 1)}$$



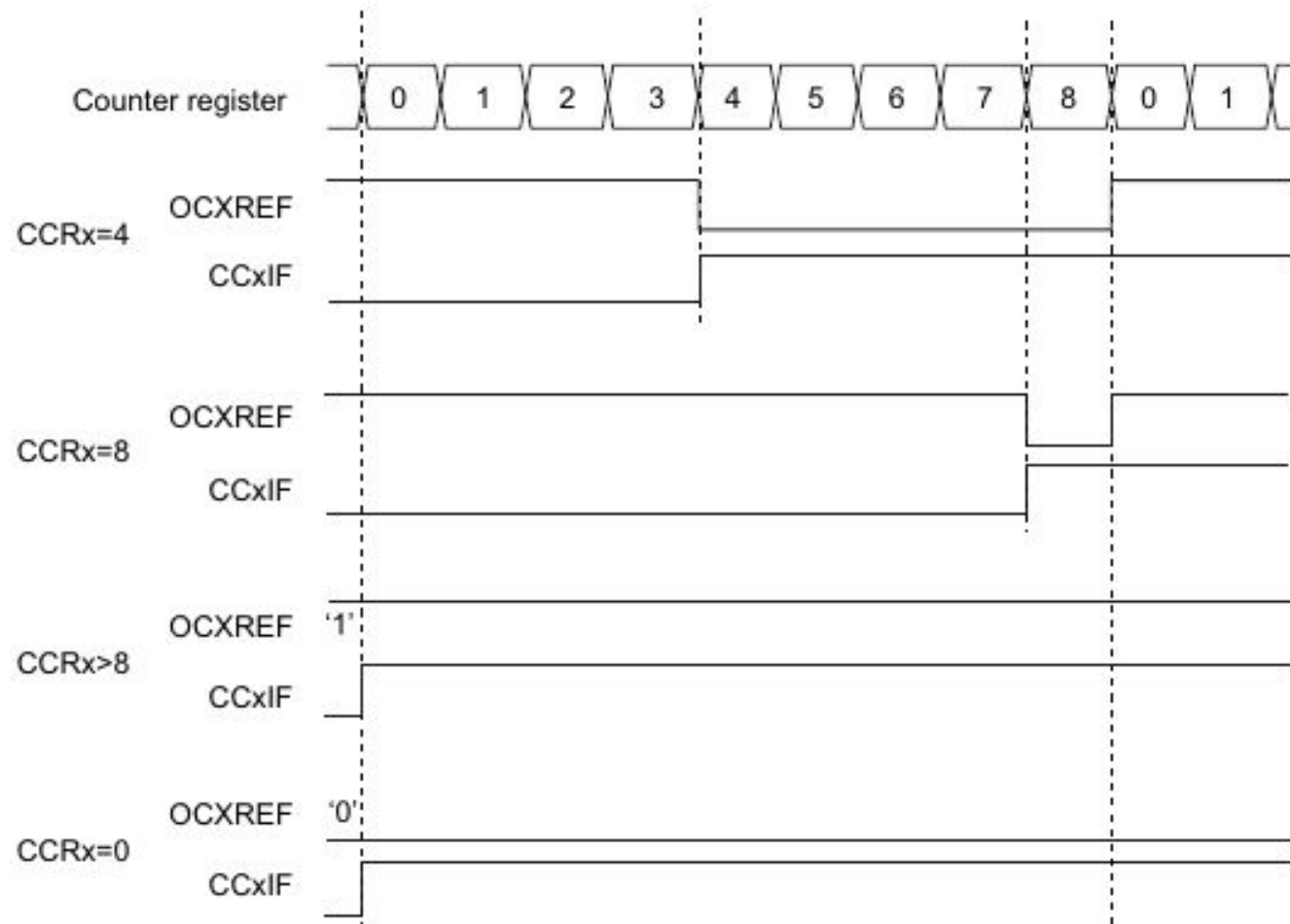
Counter Mode

A fonte de clock do timer advém de uma **fonte externa** através de um GPIO, pode ser por exemplo, um botão, um sensor de interrupção, etc.



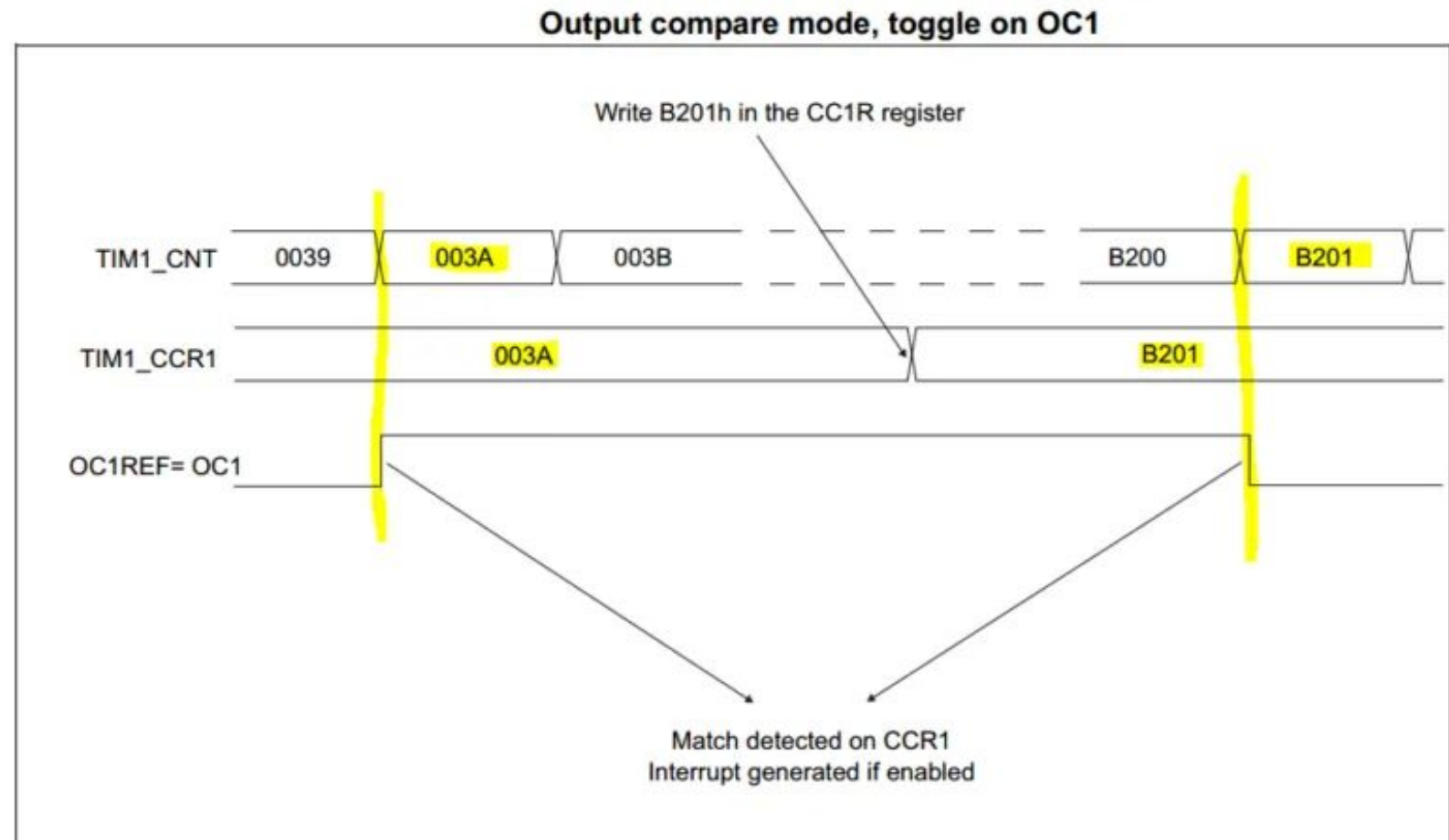
PWM Mode

Modo para controle de **potência** através de *Pulse Width Modulation* (PWM), variando o *Duty Cycle*.



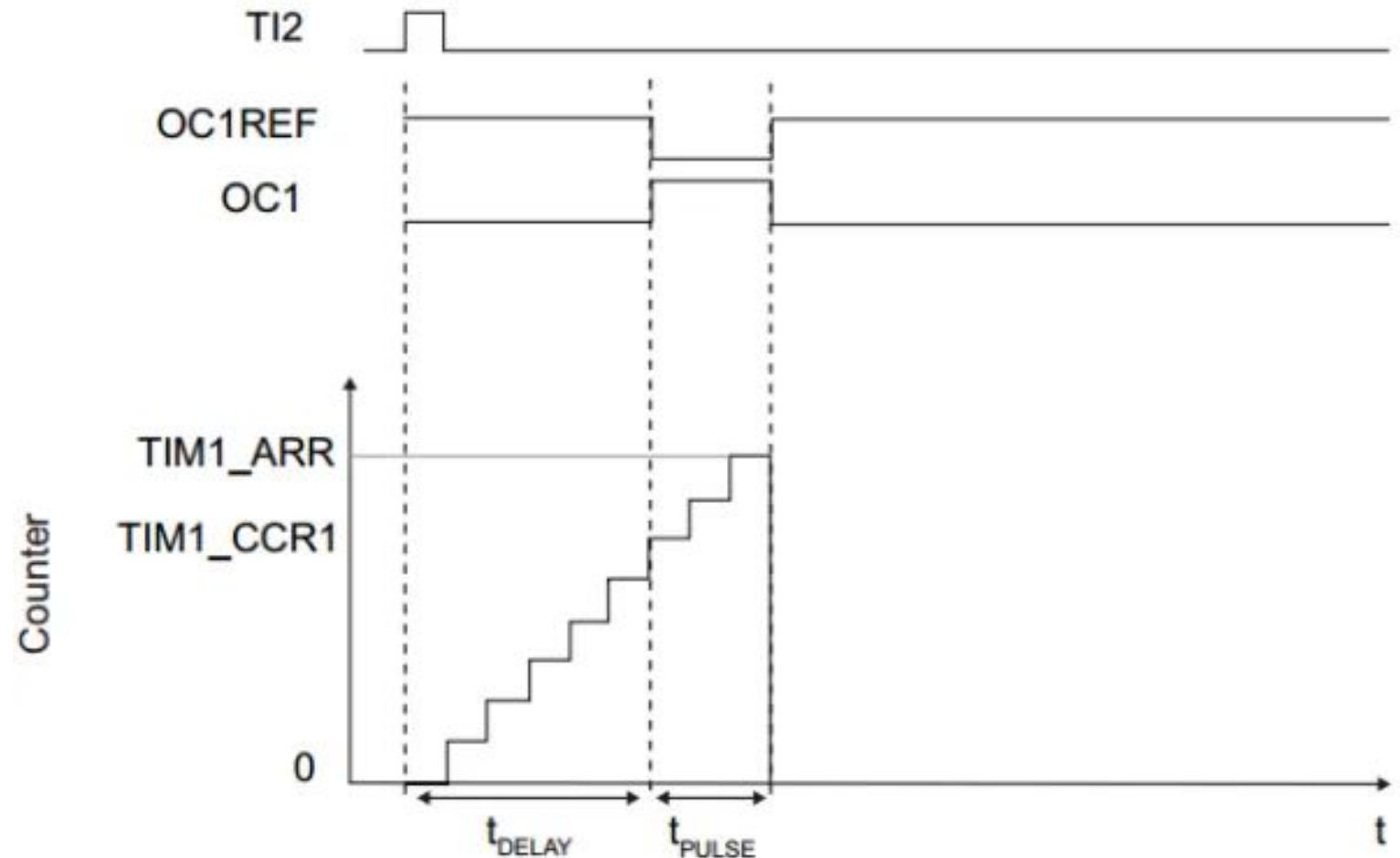
Output Compare Mode

Pode realizar diferentes funções de *output* quando há um *match* do valor do comparador com o contador.



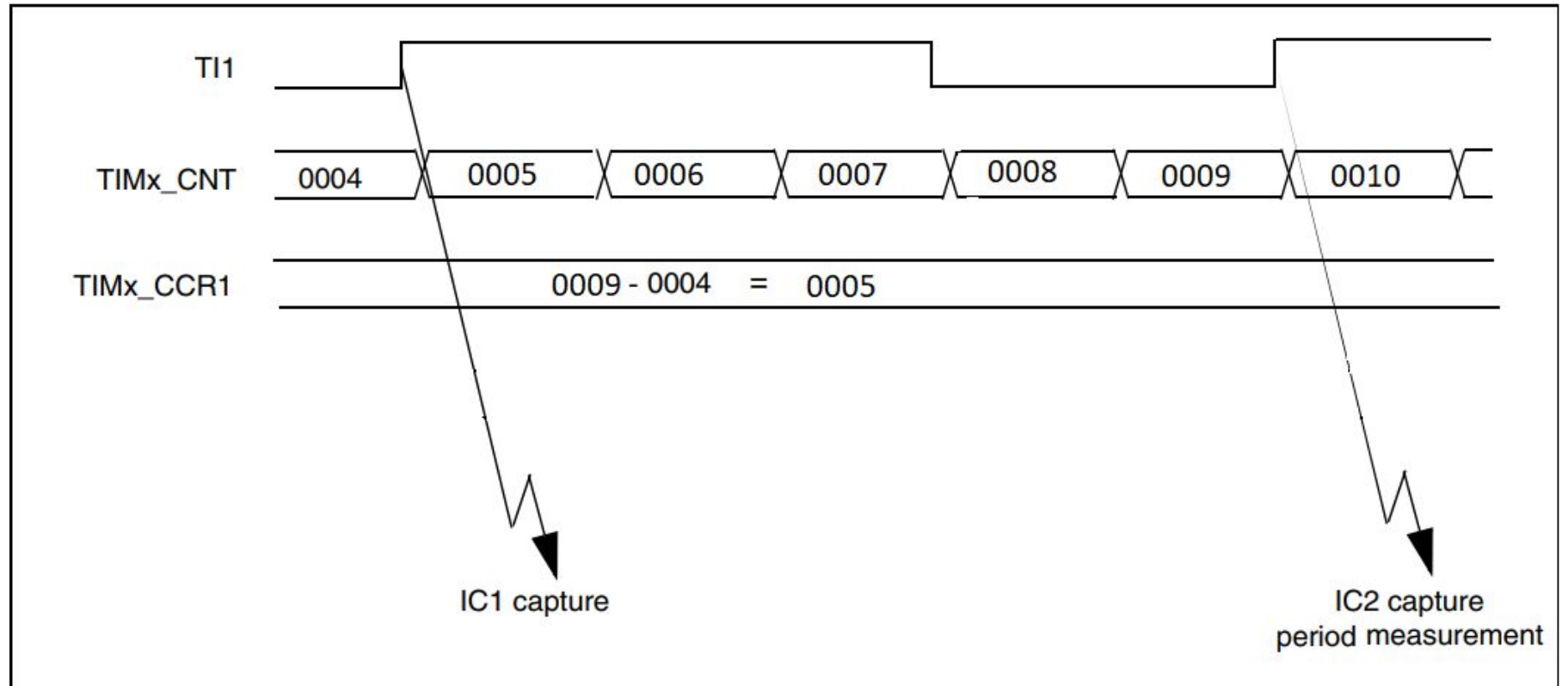
One-pulse Mode

Permite que seja gerado um pulso unitário com tempo e atraso definidos.



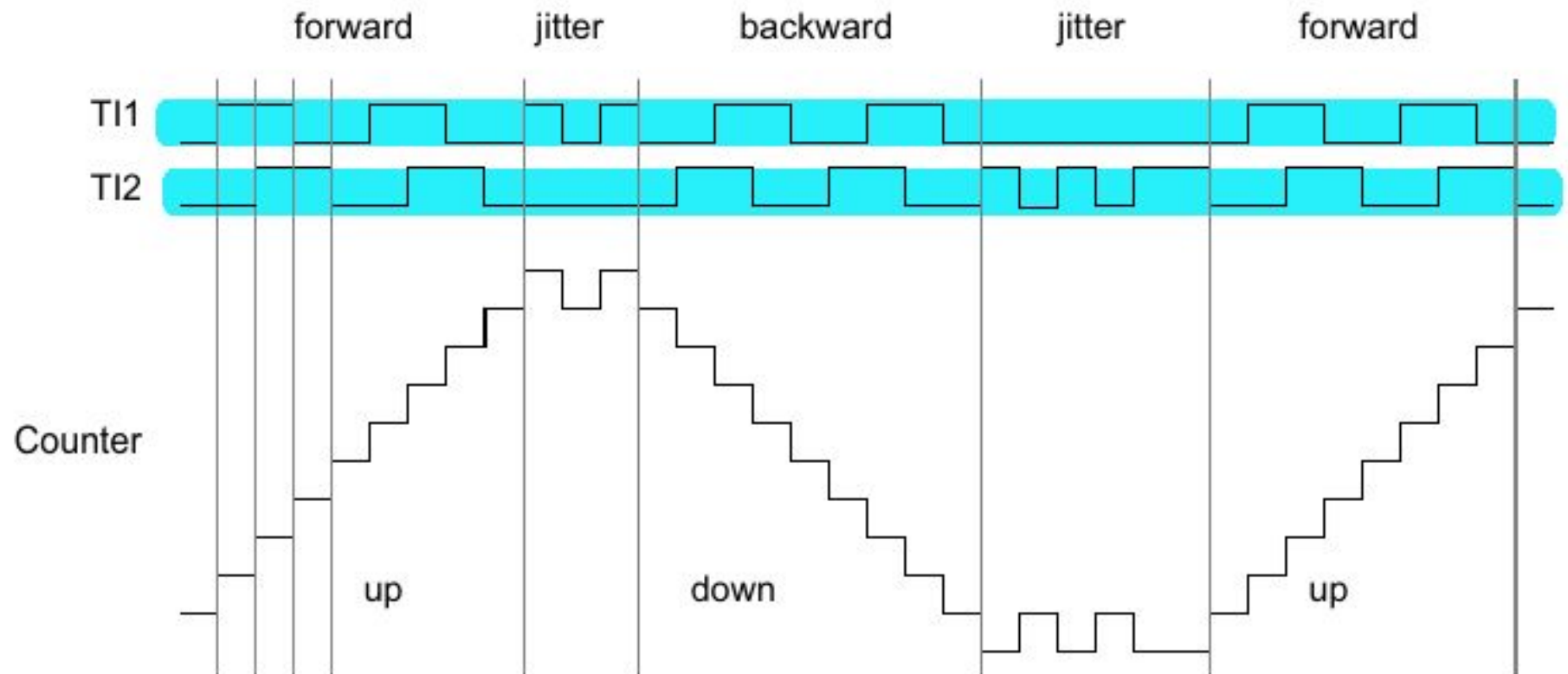
Input Compare Mode

Podemos realizar a contagem de tempo, a partir de um primeiro evento de *capture* até um segundo evento de *capture*. Útil para sensores de ultrassom.



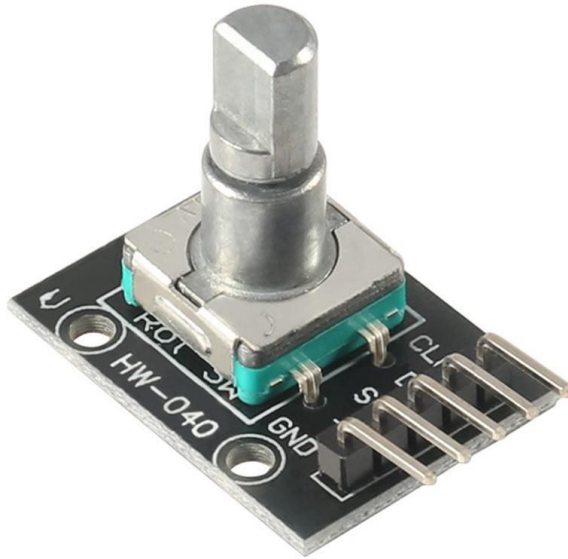
Encoder Mode

Neste modo o timer opera como um contador digital de duas entradas. A depender da sequência de pulsos ele incrementa ou decrementa



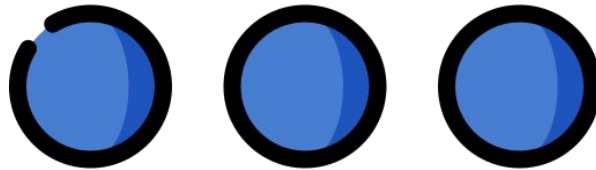
Encoder Mode

Este modo permite que possamos ler *Rotary Encoders*, por *hardware*, estes dispositivos são utilizados para interfaces de usuários e tacômetros.



Outros Modos

Existem ainda mais modos de operação dos timers, onde cada tipo tem suas limitações e aplicações. Para mais informações, consulte a documentação **RM0444**, nos capítulos referente aos *timers*.



Localização dos Terminais



Acesse e **STM32G0B1xB/xC/xE**, e consulte a **tabela 13 à 20**.

PA0	I2S2_CK	USART2_CTS	TIM2_CH1_ETR	-	USART4_TX	LPTIM1_OUT	UCPD2_FRSTX	COMP1_OUT
PA1	SPI1_SCK/ I2S1_CK	USART2_RTS _DE_CK	TIM2_CH2	-	USART4_RX	TIM15_CH1N	I2C1_SMBA	EVENTOUT
PA2	SPI1_MOSI/ I2S1_SD	USART2_TX	TIM2_CH3	-	UCPD1_FRSTX	TIM15_CH1	LPUART1_TX	COMP2_OUT
PA3	SPI2_MISO/ I2S2_MCK	USART2_RX	TIM2_CH4	-	UCPD2_FRSTX	TIM15_CH2	LPUART1_RX	EVENTOUT
PA4	SPI1_NSS/ I2S1_WS	SPI2_MOSI/ I2S2_SD	USB_NOE	USART6_TX	TIM14_CH1	LPTIM2_OUT	UCPD2_FRSTX	EVENTOUT
PA5	SPI1_SCK/ I2S1_CK	CEC	TIM2_CH1_ETR	USART6_RX	USART3_TX	LPTIM2_ETR	UCPD1_FRSTX	EVENTOUT
PA6	SPI1_MISO/ I2S1_MCK	TIM3_CH1	TIM1_BKIN	USART6_CTS	USART3_CTS	TIM16_CH1	LPUART1_CTS	COMP1_OUT
PA7	SPI1_MOSI/ I2S1_SD	TIM3_CH2	TIM1_CH1N	USART6_RTS _DE_CK	TIM14_CH1	TIM17_CH1	UCPD1_FRSTX	COMP2_OUT
PA8	MCO	SPI2_NSS/ I2S2_WS	TIM1_CH1	-	CRS1_SYNC	LPTIM2_OUT	-	EVENTOUT
PA9	MCO	USART1_TX	TIM1_CH2	-	SPI2_MISO/ I2S2_MCK	TIM15_BKIN	I2C1_SCL	EVENTOUT
PA10	SPI2_MOSI/ I2S2_SD	USART1_RX	TIM1_CH3	MCO2	-	TIM17_BKIN	I2C1_SDA	EVENTOUT

STM32G0 - Funções Utilizadas



Funções utilizadas para iniciar e parar o contador de um timer, sem habilitar as interrupções

```
1  /*
2   * Funcao de start onde o timer inicializa mas sem as funcionalidades
   de interrupcao ,
3   * onde htim e o handle do timer
4   */
5  HAL_StatusTypeDef HAL_TIM_Base_Start(TIM_HandleTypeDef *htim);
6
7  /*
8   * Funcao de para o contador do timer
9   */
10 HAL_StatusTypeDef HAL_TIM_Base_Stop(TIM_HandleTypeDef *htim);
```



STM32G0 - Funções Utilizadas



Se for necessário habilitar as interrupções, utilizamos as funções:

```
13      * Funcao de start onde o timer inicializa com as funcionalidades de
      interrupcao ,
14      * onde htim e o handle do timer
15      */
16      HAL_StatusTypeDef HAL_TIM_Base_Start_IT(TIM_HandleTypeDef *htim);
17
18      /*
19      * Funcao de para o contador do timer, quando inicializado em modo de
20      * interrupcao
21      */
22      HAL_StatusTypeDef HAL_TIM_Base_Stop_IT(TIM_HandleTypeDef *htim);
```

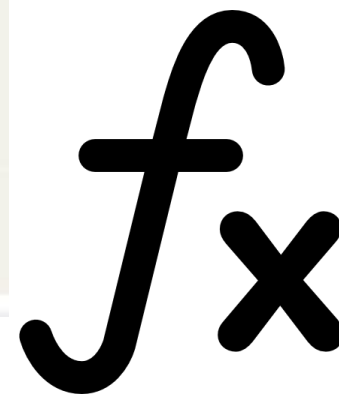
f_x

STM32G0 - Funções Utilizadas



Podemos também ler e escrever no registrador de contador, utilizando as funções definidas por macros:

```
1  /*
2  * Funcao que retorna o valor do contador do timer , onde
3  * __HANDLE__ refere-se ao handle do timer
4  */
5  __HAL_TIM_GET_COUNTER(__HANDLE__);
6
7  /*
8  * Funcao que define o valor do contador do timer , onde
9  * __HANDLE__ refere-se ao handle do timer e __COUNTER__ ao valor
10 * que deseja-se definir
11 */
12 __HAL_TIM_SET_COUNTER(__HANDLE__, __COUNTER__);
```

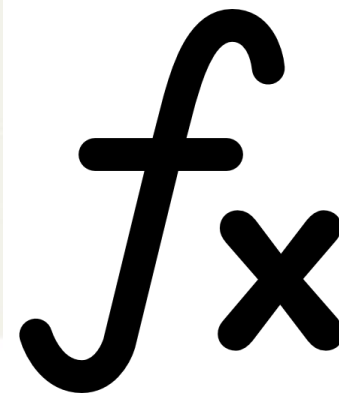


STM32G0 - Funções Utilizadas



Podemos também ler e escrever no registrador de *Auto-reload*, utilizando as funções definidas por macros:

```
1  /*
2   * Funcao que retorna o valor do auto-reload do timer, onde
3   * __HANDLE__ refere-se ao handle do timer
4   */
5  __HAL_TIM_GET_AUTORELOAD(__HANDLE__);
6
7  /*
8   * Funcao que define o valor do auto-reload do timer, onde
9   * __HANDLE__ refere-se ao handle do timer e __COUNTER__ ao valor
10  * que deseja-se definir
11  */
12  __HAL_TIM_SET_AUTORELOAD(__HANDLE__, __COUNTER__);
```



STM32G0 - Funções de Callback



```
1 // callback chamado quando ocorre um overflow/underflow
2 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim);
3
4 // callback chamado em modo output compare
5 void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim);
6
7 // callback chamado em modo input compare
8 void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim);
```



STM32G0 - Funções de Callback



```
10 // callback chamado quando um pulso de pwm e finalizado
11 void HAL_TIM_PWM_PulseFinishedCallback(TIM_HandleTypeDef *htim);
12
13 // callback referente a configuracao de trigger
14 void HAL_TIM_TriggerCallback(TIM_HandleTypeDef *htim);
15
16 // callback chamado quando ocorre algum erro no periferico
17 void HAL_TIM_ErrorCallback(TIM_HandleTypeDef *htim);
```



STM32G0 - Mais Funções

Para consultar mais funções, como para utilizar os outros modos do timer, utilize a documentação **UM2319:Description of STM32G0 HAL and low-layer drivers**, nos capítulos:

- 48 HAL TIM Generic Driver
- 49 HAL TIM Extension Driver



Dúvidas ??



Referências



COLLEY, Stephen. **Real Time Clocks (RTCs) in Microcontroller Timers**. 2020.

<https://www.allaboutcircuits.com/technical-articles/introduction-to-microcontroller-timers-real-time-clock-s/>. Acesso em 5 de Março de 2022.

GUNASEKARAN, Balaji. **Microcontroller Timers, Their Ty-pes and Applications**. 2022.

<https://embeddedinventor.com/embedded-timers-their-types-and-applications/>. Acesso em 5 de Março de 2022.

MAGDY, Khaled. **STM32 Timers Tutorial | Hardware Timers Explained**. 2020.

<https://deepbluembedded.com/stm32-timers-tutorial-hardware-timers-explained/>. Acesso em 1 de Março de 2022.

ST MICROELECTRONICS. **RM0444 - Reference Manual**. 5. ed. [S.l.], 2020. STM32G0x1 advanced Arm ® -based 32-bit MCUs.

__. **STM32G0B1xB/xC/xE**. 2. ed. [S.l.], 2021. Arm ® Cortex ® -M0+ 32-bit MCU, up to 512KB Flash, 144KB RAM, 6x USART, timers, ADC, DAC, comm. I/Fs, 1.7-3.6V.

__. **UM2324 - User Manual**. 4. ed. [S.l.], 2021. STM32 Nucleo-64 boards (MB1360).

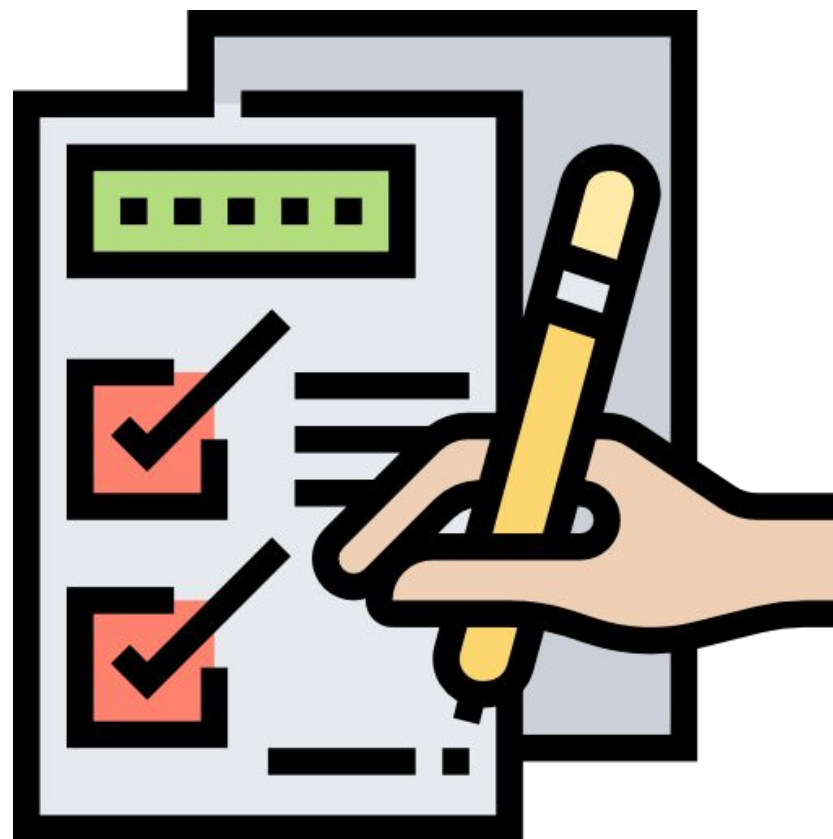
__. **UM2319: Description of STM32G0 HAL and low-layer drivers**. 2. ed. [S.l.], 2020.



Mão na Massa

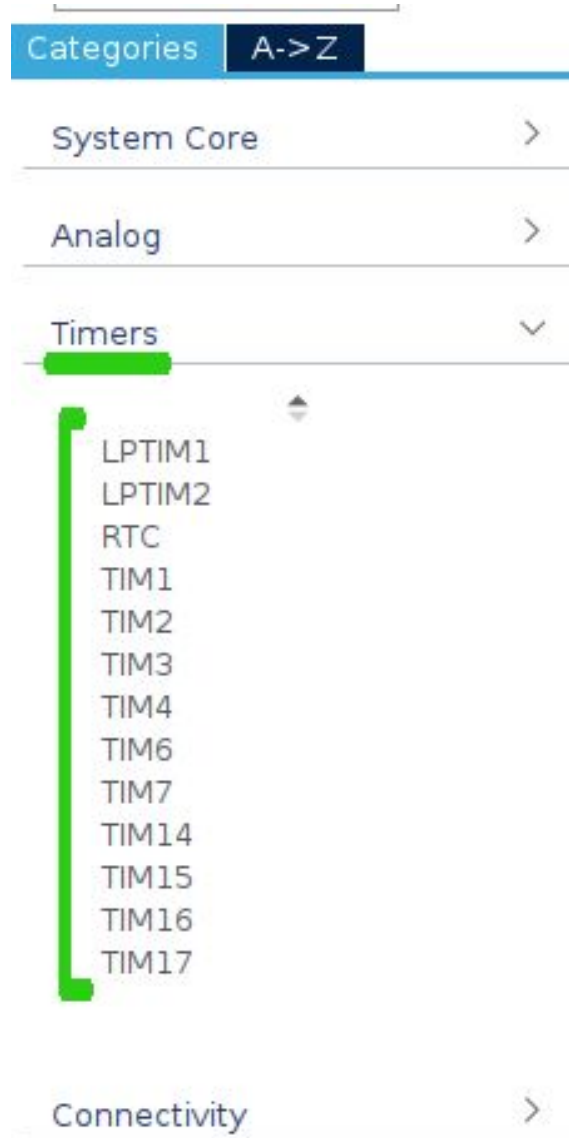


Lista de Exercícios #8



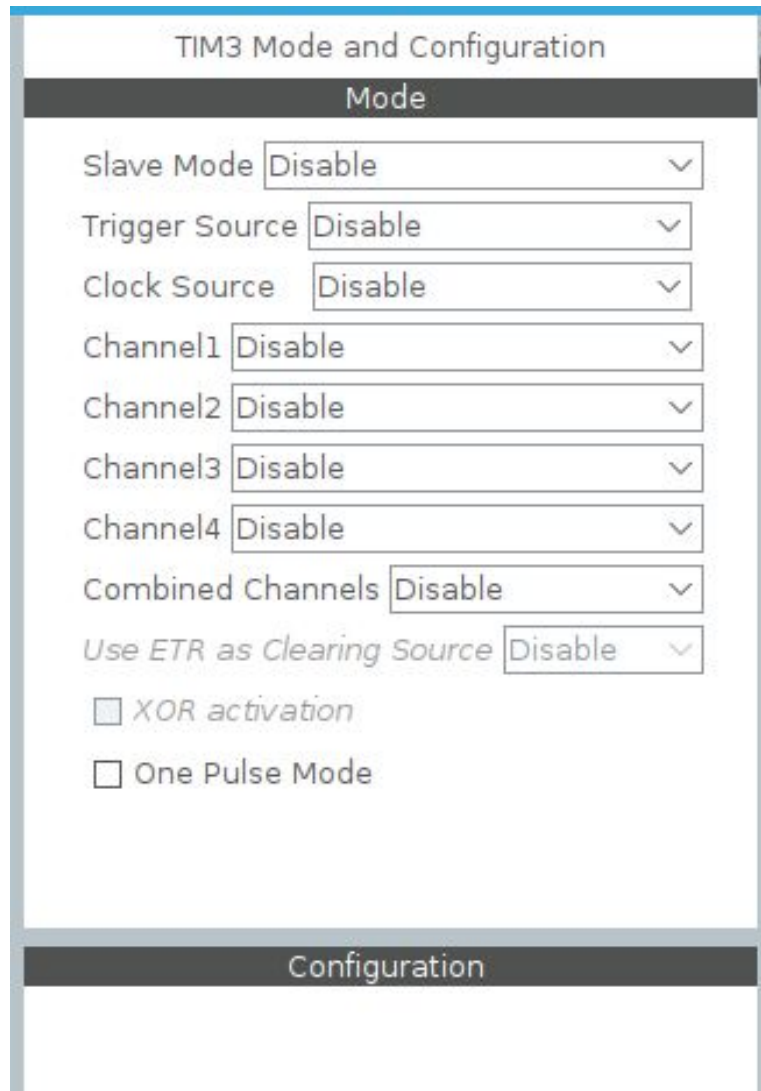
Timers e RTC

Timer - Configuração Inicial



Selecione o timer e/ou os timers desejados, de acordo com suas características.

Timer - Configuração Inicial

A screenshot of a software interface titled "TIM3 Mode and Configuration". The interface is divided into two main sections: "Mode" and "Configuration". The "Mode" section contains several dropdown menus, all currently set to "Disable": "Slave Mode", "Trigger Source", "Clock Source", "Channel1", "Channel2", "Channel3", "Channel4", "Combined Channels", and "Use ETR as Clearing Source". Below these are two checkboxes: "XOR activation" and "One Pulse Mode", both of which are unchecked. The "Configuration" section is visible at the bottom but its contents are not shown in this view.

TIM3 Mode and Configuration

Mode

Slave Mode

Trigger Source

Clock Source

Channel1

Channel2

Channel3

Channel4

Combined Channels

Use ETR as Clearing Source

☐ XOR activation

☐ One Pulse Mode

Configuration

Será então aberta a configuração dos Modos do timer. As opções variam de acordo com cada Timer.

Exemplificarei como configurar para modo Temporizador e Contador.

Timer - Configuração Inicial

Mode

Slave Mode Disable

Trigger Source Disable

Clock Source Disable

Channel1 Disable

Channel2 Disable

Channel3 Disable

Channel4 Disable

Combined Channels Disable

Use ETR as Clearing Source Disable

☐ XOR activation

☐ One Pulse Mode

OU

Mode

Slave Mode Disable

Trigger Source Disable

Clock Source Disable

Channel1 Disable

Channel2 Disable

Channel3 Disable

Channel4 Disable

Combined Channels Disable

Use ETR as Clearing Source Disable

☐ XOR activation

☐ One Pulse Mode

Para modo contador, selecione um *trigger source* (ETR1 para GPIO)

Escolha uma fonte de clock para o sinal de clock (*internal clock*).

Timer - Configuração Inicial



Configuration

Reset Configuration

✓ NVIC Settings ✓ DMA Settings

✓ Parameter Settings ✓ User Constants

Configure the below parameters :

Search (Ctrl+F) ⏪ ⏩ ⓘ

▼ Counter Settings

Prescaler (PSC - 16 bits value)	0
Counter Mode	Up
Counter Period (AutoReload R...)	65535
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

▼ Trigger Output (TRGO) Parameters

Master/Slave Mode (MSM bit)	Disable (Trigger input effect not...
Trigger Event Selection TRGO	Reset (UG bit from TIMx_EGR)

É liberado o menu de configurações. Em geral alteramos apenas o *prescaler* e o *Auto-reload*.

Timer - Configuração Interrupção



Configuration

Reset Configuration

✓ NVIC Settings	✓ DMA Settings
✓ Parameter Settings	✓ User Constants

NVIC Interrupt Table	Enabled	Preemption Priority
TIM3, TIM4 global Interrupt	<input checked="" type="checkbox"/>	0

Vamos até o *NVIC Settings*, e habilitamos a caixa *Enabled*.

Os timers podem ter mais de uma fonte de interrupção.



PADO
Labs