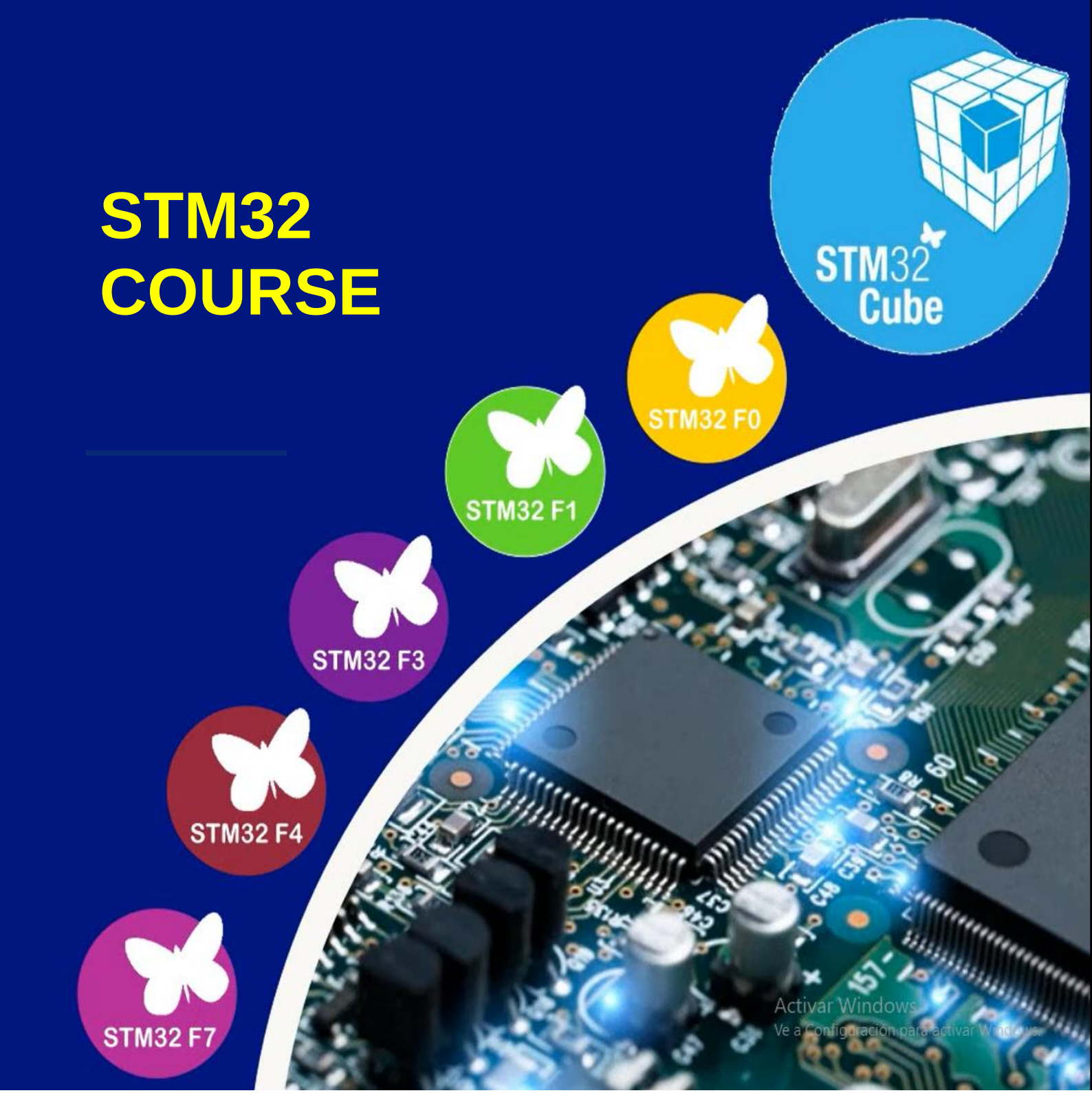
STM32 COURSE



FEBRUARY 23, 2021

006_Timer_Interruption

Creado por: Ing. Christian Salazar

SECTION 6

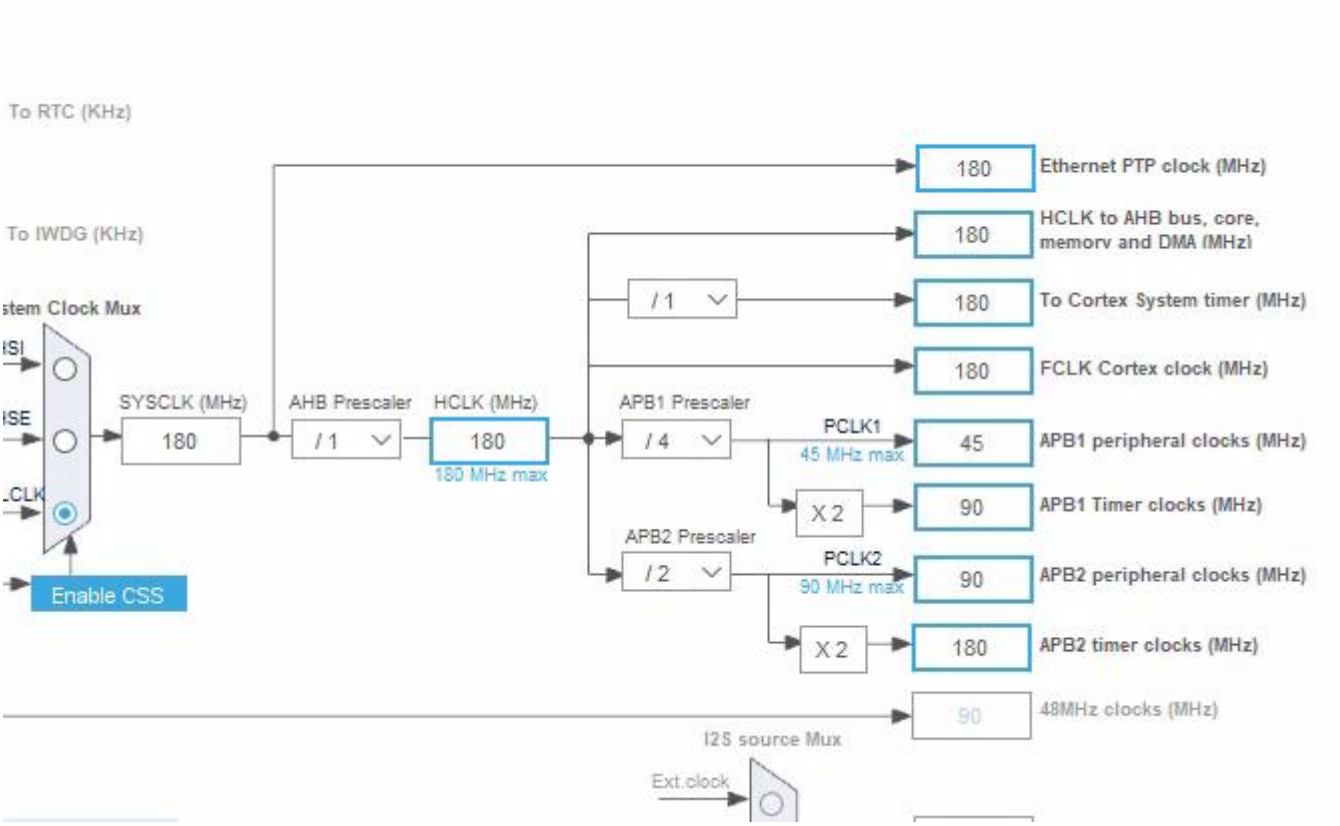
006_Timer_Interruption

What will we learn?

In this video we will calculate the values of the period and prescaler so that a timer generates an interruption, we will test when the option is enabled that the timer generates the interruption periodically or only once, we will use the microcontroller datasheet to determine the frequency at which the selected timer is oscillating.

"We will use HAL Drivers, which will help us greatly to port and recycle code routines from one processor in one Family to another in another Family."

TIMER FREQUENCY



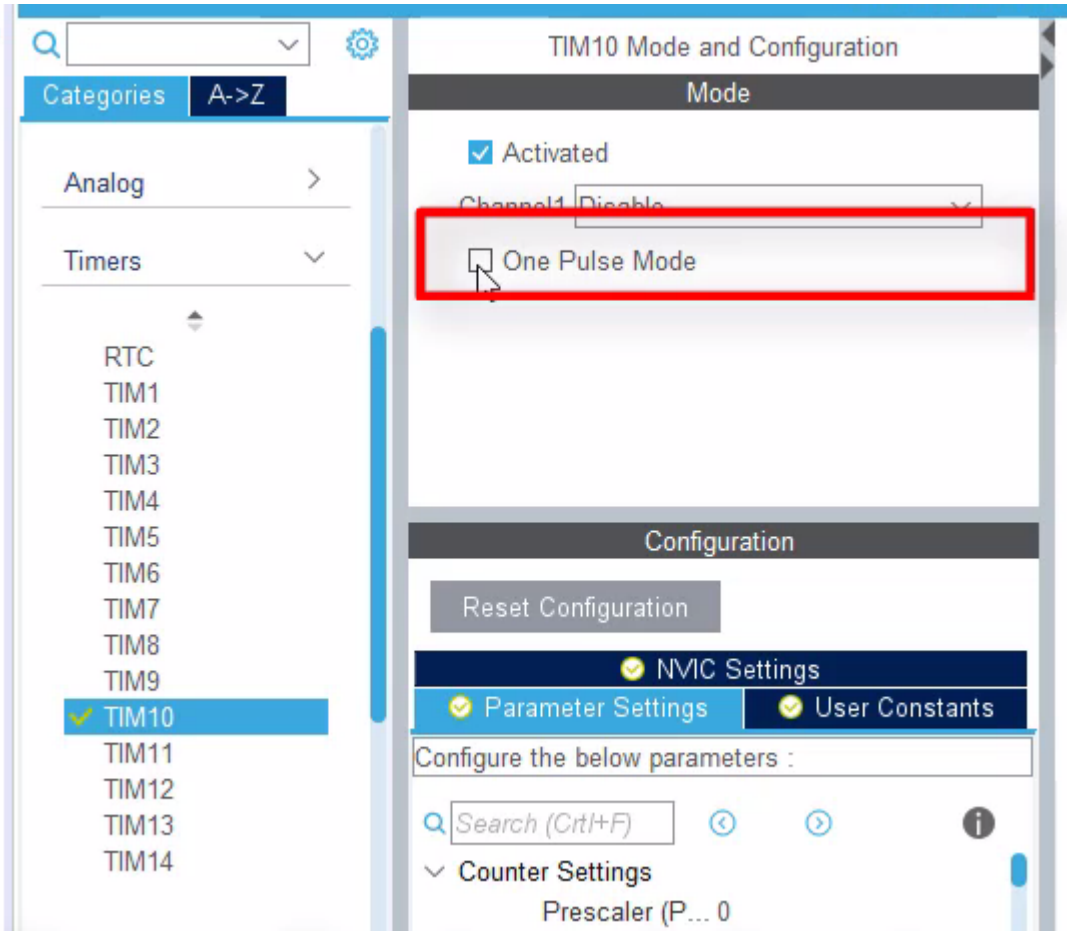
As we can see in this case we have 2 frequencies at which the timers work (APB1 Timer Clocks and APB2 timer clocks) at 90 and 180 MHz respectively, so if for example, as in the video, we select the TIMER10 we must search the microcontroller datasheet for which Bus (APB1 or APB2) said timer is connected.

Table 1. STM32F4xx register boundary addresses (continued)

Boundary address	Peripheral	Bus	Register map
0x4001 4800 - 0x4001 4BFF	TIM11	APB2	Section 19.5.12: TIM10/11/13/14 register map on page 694
0x4001 4400 - 0x4001 47FF	TIM10		
0x4001 4000 - 0x4001 43FF	TIM9		Section 19.4.13: TIM9/12 register map on page 684
0x4001 3C00 - 0x4001 3FFF	EXTI		Section 12.3.7: EXTI register map on page 387
0x4001 3800 - 0x4001 3BFF	SYSCFG		Section 9.2.8: SYSCFG register maps for STM32F405xx/07xx and STM32F415xx/17xx on page 294 and Section 9.3.8: SYSCFG register maps for STM32F42xxx and STM32F43xxx on page 301
0x4001 3400 - 0x4001 37FF	SPI4	APB2	Section 28.5.10: SPI register map on page 925
0x4001 3000 - 0x4001 33FF	SPI1	APB2	Section 28.5.10: SPI register map on page 925
0x4001 2C00 - 0x4001 2FFF	SDIO		Section 31.9.16: SDIO register map on page 1074
0x4001 2000 - 0x4001 23FF	ADC1 - ADC2 - ADC3		Section 13.13.18: ADC register map on page 430
0x4001 1400 - 0x4001 17FF	USART6		Section 30.6.8: USART register map on page 1018
0x4001 1000 - 0x4001 13FF	USART1		
0x4001 0400 - 0x4001 07FF	TIM8		Section 17.4.21: TIM1 and TIM8 register map on page 587
0x4001 0000 - 0x4001 03FF	TIM1		
0x4000 7C00 - 0x4000 7FFF	UART8	APB1	Section 30.6.8: USART register map on page 1018
0x4000 7800 - 0x4000 7BFF	UART7		

We can see in the PDF RM0090 of the STM32F429xx that the TIM10 is in the APB2 domain, that is, it is running at 180 MHz, this frequency is important to be able to calculate a certain time required.

ONE-TIME TIMER COUNT



One Pulse Mode is the option that if we enable it, it will make the timer count at the calculated time ONLY ONE TIME, it will do it again if we order it again and so on, it will count only once and it will generate the time interruption only when we order it to start. the count with the proper function.

TIMER SELF-LOADING (PERIODIC)

Likewise, with the previous case, if we do not mark the One Pulse Mode option, we are saying that our timer will generate a periodic interruption of the time established with the calculation of the period and prescaler when we indicate it with the timer start function, in both cases. We have the option to put the timer count in STOP and

manipulate the current value of the count to 0 to restart when we wish (TIM10-> CNT = 0).

CALCULATE PERIOD AND PRESCALER TO DETERMINE A TIME

To calculate and generate an interrupt at 1 millisecond:

1. Calculate the frequency of the time required.
For 1 mS, we know that F (Frequency) = 1 / T (Time), which gives us a frequency of 1 Khz.

2. Know the frequency of the Timer (Datasheet), as we saw previously, the TIM10 is at 180 Mhz.

3. We eliminate the required frequency by dividing.
FreqReq = 180,000,000 Hz / 1,000 Hz (1 Khz is 1,000 Hz)
FreqReq = 180,000.

4. Calculate Period and Prescaler
Period and Prescaler are 2 factors that are multiplied to know the value up to which the timer must count.

These two Factors are unsigned 16-bit registers which means that they can have a value from 0 to 65535.
With the formula FreqReq = Period * Prescaler, with the previously calculated we have:
180,000 = Period * Prescaler
You understand what we must do is assume one of the values in such a way that the assumed number is between 0 and 65535 and that the

result of the clearing of the other variable in the previous formula with the assumed number is in the same way between 0 and 65535.
We will assume for the Period = 100
Solving for we have that the Prescaler = 1800
As we can see, both fulfill that they are from 0 to 65535.
To configure the timer and put these values, it is subtracted by 1 that is:
Period = 100 - 1 = 99
Prescaler = 1800 - 1 = 1799
These are the values that we really must assign in the configuration, these values are set since the timer starts counting from 0 and there is a cycle in the jump from 0 to 1.
It is like a Vector of a variable:
int MyVariable [1800];
Set the last space of the vector to 0, we would have
MyVariable [1799] = 0;
We see that position 0 has its own value.