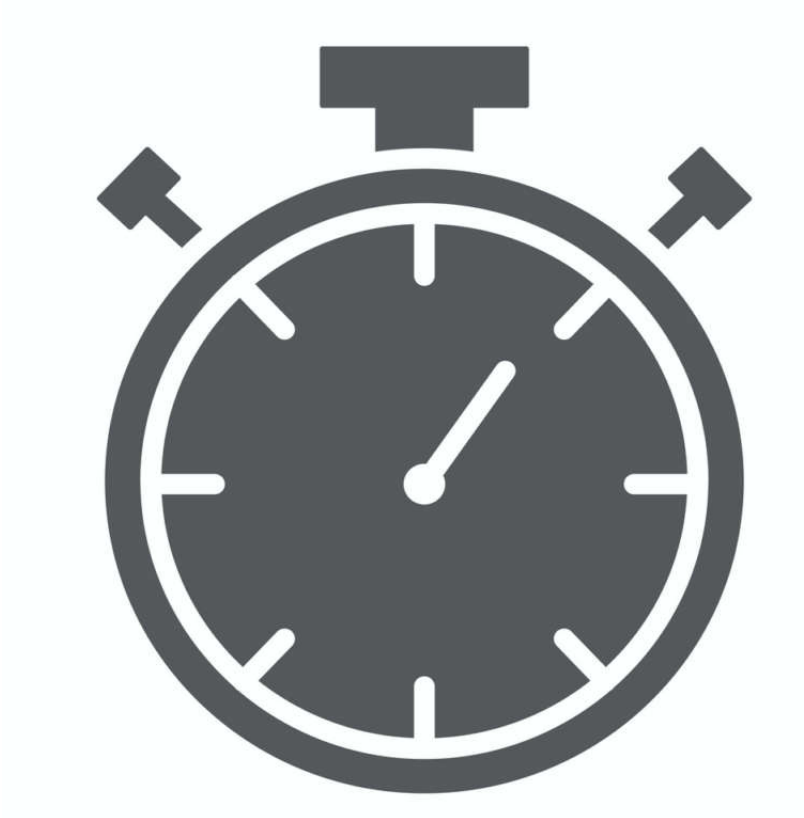
Working with STM32 and Timers part 1: Delay using Timer

Posted August 26, 2021 | *by* Husamuldeen | *in* Embedded Systems, Peripheral Drivers, STM32 |



In this guide, we shall investigate how to use timer to generate precise delay in down to microseconds (will use 1 second delay here for demonstrations purposes only).

In this guide, we will cover the following:

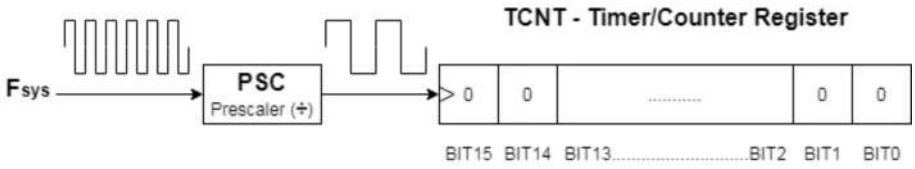
- What is the timer and how it works
- Configure the timer to generate delay
- Code
- Demo

1 What is the timer and how it works

A Timer Module in its most basic form is a digital logic circuit that counts up every clock cycle. More functionalities are implemented in hardware to support the timer module so it can count up or down. It can have a Prescaler to divide the input clock frequency by a selectable value. It can also have circuitry for input capture, PWM signal generation, and much more as we'll see in this tutorial.

Let's consider a basic 16-Bit timer like the one shown below. As a 16-Bit time, it can count from 0 up to 65535. Every clock cycle, the value of the timer is incremented by 1. And as you can see, the Fsys is not the frequency that is incrementing the timer module.

But it gets divided by the Prescaler, then it gets fed to the timer.



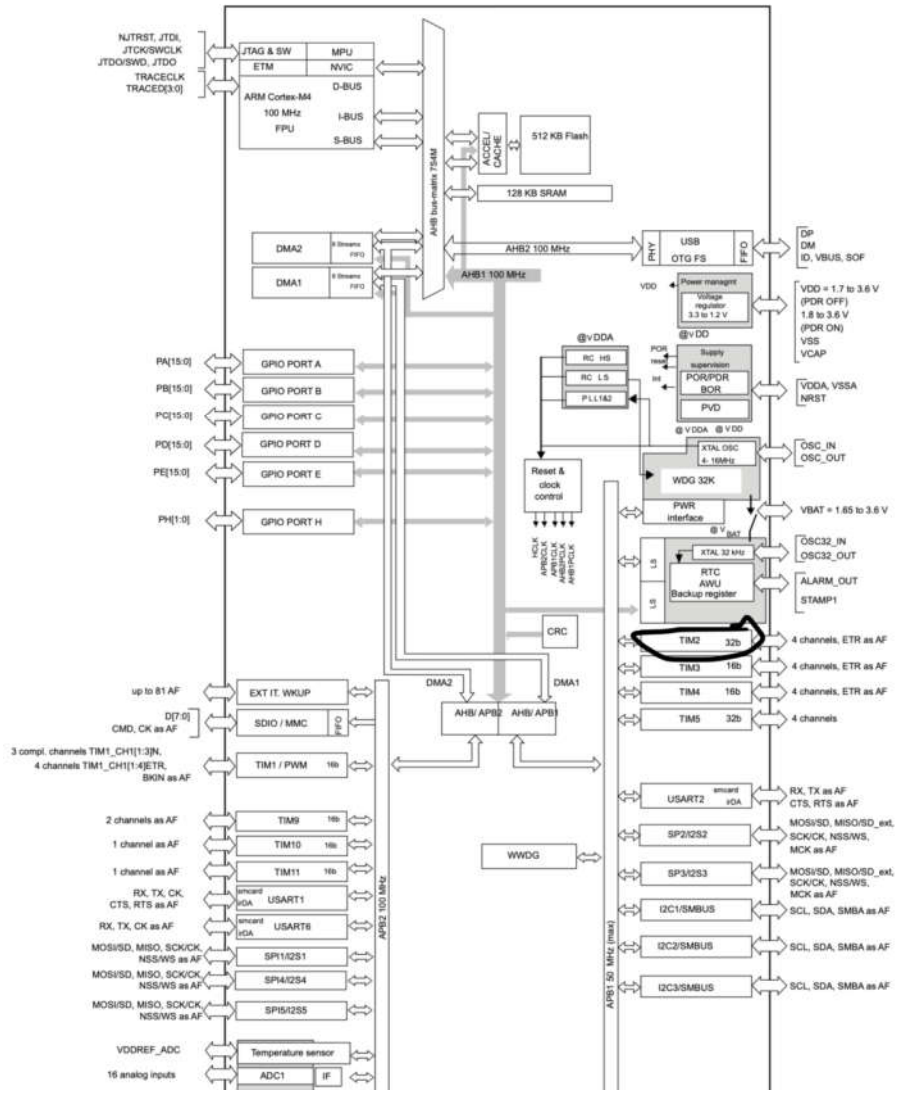
Basically, in timer mode, the TCNT register is incremented by 1 each clock cycle @ the following frequency (Fsys/PSC). This means if the Fsys is 80MHz & PSC is 1:1024, the TCNT gets incremented by 1 every 12.8µSec. Therefore, if you start this timer to count from 0 until it reaches overflow (at 65535), a flag will be set and starts over from zero

2. Configure the timer to generate delay

We starting by getting to know which timer to use, in this case , we will use TIMER2.

Hence, we need to know which bus is connected to. From the diagram, TIMER2 is connected to APB1 bus

Figure 3. STM32F411xC/xE block diagram



STM32F411 Block Diagram

Then we enable clock access to the timer as following

C

```
RCC->APB1ENR|=RCC_APB1ENR_TIM2EN;
```

After we enabled clock access to TIM2 we shall set the prescaller and auto-reload value as following

C

```
TIM2->PSC=249;
TIM2->ARR=63999;
```

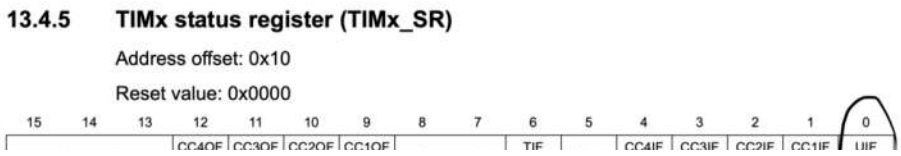
The ARR value is the maximum value can the counter counts. Since TIM2 is 32-bit timer, it can counts up to (2^32 -1 = 4294967295).

now we enable timer as following:

C

```
TIM2->CR1=1;
```

For delay purposes, we shall wait to the UIF to be set in SR register then reset it as following



TIMER STATUS REGISTER

C

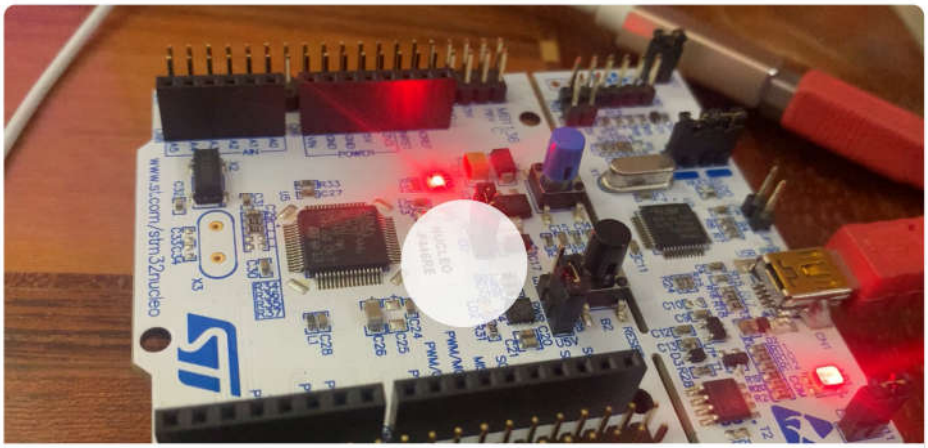
```
while(!(TIM2->SR&TIM_SR_UIF)){}
TIM2->SR&=~TIM_SR_UIF;
```

3. Code

C

```
#include "stm32f4xx.h"                  // Device header
int main(void)
{
RCC->AHB1ENR |=RCC_AHB1ENR_GPIOAEN ;//enable gpio a clock
GPIOA ->MODER|= GPIO_MODER_MODER5_0; //PA5 as output
//Timer1 configure
RCC->APB1ENR|=RCC_APB1ENR_TIM2EN;
TIM2->PSC=249;
TIM2->ARR=63999;
TIM2->CNT=0;
TIM2->CR1=1;
while(1)
{
GPIOA->ODR^=GPIO_ODR_OD5;//toggle PA5
while(!(TIM2->SR&TIM_SR_UIF)){} //wait UIF to be set
TIM2->SR&=~TIM_SR_UIF; //reset UIF
}
}
```

4. Demo



0:00 / 0:05