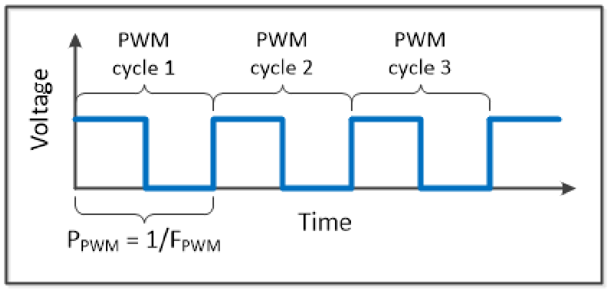
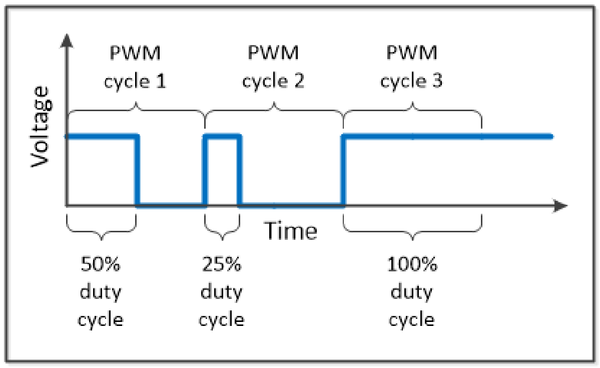


With PWM, the duty-cycle is the part of the PWM-cycle in which the signal is high, as illustrated below. Configuring a timer to generate PWM output includes specifying the PWM frequency (F_{PWM}) and the duty-cycle. For dimmable LEDs, the PWM frequency should be at least 100Hz, to avoid visible flicker [STM08, pg5]. An STM tutorial on dimmable LEDs uses an F_{PWM} of 1KHz, so that frequency will be used here [STMa, pg33].



The LED brightness is proportional to the duty-cycle value. With a duty-cycle of 0% the LED is off. At 100% the LED is at its brightest. If the ratio between duty-cycle and brightness is linear, then a 50% duty-cycle, would make the LED half as bright as 100%. (I don't know what the actual ratio is.)

2.2 How the PWM frequency is configured

The PWM signal is generated by a clock, clock prescaler, and timer. This section describes how they are configured to generate a particular PWM frequency. An earlier figure shows the components, and how they are configured.

STM32 supports many different types of PWM. The type presented here appears to be commonly used for dimming an LED, and it's used in the Chapter 13 example-programs. In particular, the type presented here is: PWM mode 1, in edge-aligned mode, with up-counting. Those PWM attributes are not explained here. However, to use the cited STM references, it can be necessary to know the PWM type being used.

The terminology used here is adapted from an STM tutorial [STMa, pgs31-33]. Among the tutorials I found, it was the easiest to understand. However, its equations on duty-cycle and PWM-resolution are incorrect, as described in the bibliography, below.

As shown in the figure, the timer's input is a *scaled clock-signal*, with frequency $\mathbf{F_{SCALED}}$. The scaled clock-signal is generated by using a *system clock* and a *clock prescaler*. The system clock's frequency is referred to as $\mathbf{F_{TIM}}$ (*timer operating frequency*). The clock-prescaler is used to reduce the F_{TIM}. The clock-prescaler is embedded in the timer. Its **PSC** register is used to set F_{SCALED}:

$$F_{SCALED}\ \text{tics/sec} = [F_{TIM}\ \text{tics/sec}]\ /\ (PSC + 1)$$

The PSC register is set by the user, and it can be 0 to 65,535 [STMa, pg 6]. `(PSC + 1)` is the *scaling value*. Adding one allows scaling values to potentially be 1 to 65,536.

The scaled clock-signal's period ($\mathbf{P_{SCALED}}$) is:

$$P_{SCALED}\ \text{sec/tic} = 1\ /\ (F_{SCALED}\ \text{tics/sec})$$

The dev-board used here is a NUCLEO-F767ZI. It's maximum CPU frequency is 216MHz, and the timers can be run at that rate, or slower [STM21a, pages 1, 18]. In the earlier figure, the example values shown are:

$$F_{TIM} = 216\text{MHz}$$