HOW HARDWARE IMPLEMENTS PULSE-WIDTH MODULATION

Introduction

This supplementary material will cover how hardware implements pulse-width modulation (PWM). Most projects won't require such extensive knowledge of hardware implementation, so this section is primarily for a more thorough understanding.

Registers Used in PWM

Pulse-width modulation uses three registers for control: The timer register, TMRy, which counts the frequency of the peripheral bus clock divided by the timer's pre-scaler value; the period register, PRy, which resets the timer register when the value of the timer equals the value of the period register; and finally, we have the output compare register, OCxR, which toggles the pin OCx to either HIGH or LOW. The value of OCxR should be less than the value of PRy, otherwise it won't toggle the OCx pin. With pulse-width modulation enabled, the OCx pin is asserted HIGH when the timer resets (TMRy = 0), and it toggles to LOW when the value of the timer equals the value of the output compare register (TMRy = OCxR).

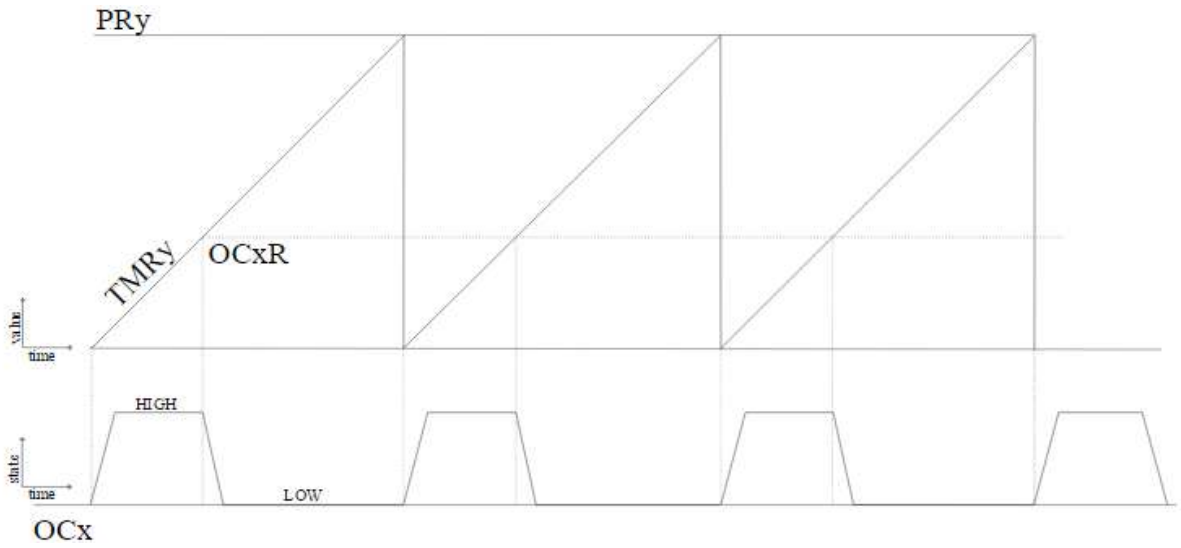

Figure 1. How TMR, PR, and OCxR affect OCx.

Calculations For PWM

Duty Cycle

Many aspects of pulse-width modulation are governed by a few equations, the simplest of which is the duty cycle equation. This equation can be written in two ways: either in terms of the time the output compare pin is HIGH in relation to the rest of the period, and in terms of the values of the period register and the output compare register. Figure 2a displays the former while Fig. 2b displays the latter.

$$\text{Duty Cycle} = \frac{T_{\text{HIGH}}}{T_{\text{P}}}$$

Figure 2a. Duty cycle in terms of time.

$$\text{Duty Cycle} = \frac{\text{OCxR}}{\text{PRy}+1}$$

Figure 2b. Duty cycle in terms of register value.

The reason that the duty cycle featured in Fig. 2b has a “1” added to the period register is because an additional cycle runs when the timer resets, resulting in a period that lasts the period register plus 1.

Relationship between the Peripheral Bus and PWM

We can find the period of the pulse-width modulation cycle by multiplying the period of the peripheral bus clock by the TMRy prescaler and the value of the period register, with one added due to the reset lag, PRy + 1. Figure 3a shows this equation.

We can then put this into terms of frequency by finding the reciprocal, i.e., put both sides to the -1 power, since frequency is one over the period. The resulting equation for frequency is shown in Fig. 3b.

$$T_{\text{PWM}} = (\text{PR} + 1)T_{\text{PB}}\text{Prescaler}$$

Figure 3a. Period of PWM in terms of period of peripheral bus.

$$\text{F}_{\text{PWM}} = \frac{\text{F}_{\text{PB}}}{(\text{PR}+1)\text{Prescaler}}$$

Figure 3b. Period of PWM in terms of frequency of peripheral bus.

Resolution of PWM

The resolution of pulse-width modulation is the number of individual steps between LOW and HIGH. Since the time that the output compare pin is HIGH is dependent on the ratio between OCxR and PRy, we can find the resolution (in number of bits) by taking the log base 2 of PR+1.

$$\text{Resolution} = \log_2 (\text{PR} + 1)$$

Figure 4. Resolution of PWM in terms of period register.

To find the smallest step in voltage that output compare can do, divide the voltage of what is considered HIGH, and then divide by PR+1. That being said, the chipKIT™ Uno32™ and Max32™ are limited to a resolution of 8 bits. Since the max value of analog Write() is 255, we use that as our PR and then find the log base 2 of 255 + 1. Thus, we find that the result is 8. So, for the chipKIT boards, the smallest step in average voltage for PWM is 3.3V divided by 256, which is about 12.9 mV per step.

Source: https://learn.digilentinc.com/Documents/381