Using the PIC16F874 Pulse Width Modulator*

CDR Charles B. Cameron

23 February 2005

The pulse width modulator relies on Timer 2 for its operation. However, whereas interrupts generated by Timer 2 use the Timer 2 Postscaler, the pulse width modulator does not. For this reason, the period of the pulse width modulator is given by

$$\boxed{T_{\mathrm{PWM}} = 4(y+1)T_{\mathrm{OSC}}\,x} \tag{1}$$

where

$$\begin{aligned} &x = \text{the Timer 2 prescaler value.} \\ &x \in \{1, 4, 16\}, \end{aligned} \tag{2}$$

and

$$\begin{aligned} &y = \text{PR2, the Timer 2 period register value} \\ &y \in [0, 255]. \end{aligned} \tag{3}$$

For example, with $f_{\mathrm{OSC}} = 4\,\mathrm{MHz}$, $x = 4$ and $y = 99$, $T_{\mathrm{PWM}} = 400\ \mu\mathrm{s}$.

The pulse width modulator's output remains high for time

$$\boxed{T_{\mathrm{HIGH}} = w\,T_{\mathrm{OSC}}\,x} \tag{4}$$

where

$$w = \mathrm{CCPRnL{:}CCPnCON}\langle 5:4\rangle) \tag{5}$$

and

$$n = \begin{cases} 1 & \text{for pulse width modulator 1} \\ 2 & \text{for pulse width modulator 2.} \end{cases} \tag{6}$$

The duty cycle of the pulse width modulator can be worked out by finding the ratio of on-time to period:

$$\begin{aligned} \eta &= \frac{T_{\mathrm{HIGH}}}{T_{\mathrm{PWM}}} \\ &= \frac{wT_{\mathrm{OSC}}x}{4(y+1)T_{\mathrm{OSC}}x} \end{aligned}$$

*Course notes for EE461 Microprocessor-based Digital Design

$$\boxed{\eta = \frac{w}{4(y+1)}.} \tag{7}$$

Now because w can take on values in the range $[0, 1023]$ and y can take on values in the range $[0, 255]$, the duty cycle can take on values in the range $[0, 1023/4]$ or 0 to 255.75. All values of the duty cycle that are larger than 1, that is, 100%, are indistinguishable. So unless we choose $y = 255$, some values of w will generate more than a 100% duty cycle. This is equivalent to saying that fewer than 10 bits of accuracy is actually available.

For example, if we choose $y = 127$ then all values of $w \geq 512$ produce $\eta = 100\%$ and only 9 bits of accuracy are effective.

To get maximum accuracy requires a specific value of y. This limits the use of Timer 2 for generating periodic interrupts. Whereas it is ordinarily true that this period can be chosen from among $3 \times 256 \times 16 = 12,288$ different values,[1] the restriction that $y = 255$ reduces this to $3 \times 16 = 48$ different values. Furthermore, the value of the prescaler x also influences the period T_{PWM} of the pulse width modulator.

In short, there are numerous constraints which influence our ability to get useful periods between Timer 2 interrupts at the same time we get useful parameters for pulse width modulation.

An example will illustrate some of the tradeoffs.

Example 1

Suppose $f_{\mathrm{OSC}} = 4\,\mathrm{MHz}$ and we want to know when 1 s has passed so that we can initiate an analog-to-digital conversion. At the same time, we want to generate a pulse width modulated signal with a period of 1 ms with 10 bits accuracy.

To get 10 bits accuracy, we require PR2= 255, as shown above. Now

$$\boxed{T_{\text{Timer 2 Interrupt}} = 4T_{\mathrm{OSC}}\, x\,(y+1)(z+1)} \tag{8}$$

where x and y are as defined in (2) and (3) and

$$\begin{aligned} z &= \text{the Timer 2 postscaler value} \\ z &\in [0, 15]. \end{aligned} \tag{9}$$

Because our desire for a full 10 bits of accuracy dictates that PR2= $y = 255$, we suppose that after some number n of Timer 2 Interrupts, 1 s will have passed

[1] There are three choices for the prescaler x, 256 choices for y =PR2, and 16 choices for the postscaler z.

and so:

$$\begin{aligned} nT_{\text{Timer 2 Interrupt}} &= 4T_{\text{OSC}}\, x\,(255+1)(z+1) && \text{(from (8))} \\ &= 1024\,(250\text{ ns})\, x\,(z+1) \\ 1\text{ s} &= (256\ \mu\text{s})\, x\,(z+1) \\ x\,(z+1) &= 3906.25 \end{aligned}$$

There are no integral solutions to this equation for x and z, even ignoring the constraints placed on them.

We could choose $x = 4$ and $z = 0$. This would lead to interrupts every 1.024 ms, an amout of time which does not divide evenly into 1 s. If the error is not important, we could ignore it. If it is important, we could treat the interrupts as if they occurred every 1 ms and accumulate the error every interrupt. Whenever the error exceeds 500 ns, we could add 1 to our interrupt counter and subtract 1 ms from the accumulated error. In this manner we would never be off by more than 500 ns.

An alternative would be to abandon the hope of achieving 10 bits of accuracy and insist that the Timer 2 Interrupts occur, say, every 1 ms. For this alternative we simply choose PR2= y as big as we can get it. This requires making x (the prescalar value) and z (the postscaler value) as small as possible:

$$\begin{aligned} T_{\text{Timer 2 Interrupt}} &= 4T_{\text{OSC}}\, x\,(y+1)(z+1) && (8) \\ y &= \frac{T_{\text{Timer 2 Interrupt}}}{4T_{\text{OSC}}\, x\,(z+1)} - 1 \\ &= \frac{1\text{ ms}}{4(250\text{ ns})\, x\,(z+1)} - 1 \\ &= \frac{1000}{x\,(z+1)} - 1 && (10) \end{aligned}$$

By choosing $x = 4$ and $z = 0$ we find that $y = 249$. Note that we could not pick $x = 1$ since that would require $y = 999$, a number too large to fit in the 8 bits of the PR2 register. As it happens, we chance also to get the pulse repetition period we desire, although this will not always be the case:

$$\begin{aligned} T_{\text{PWM}} &= 4(y+1)T_{\text{OSC}}\, x && \text{(from (1))} \\ &= 4(249+1)(250\text{ ns})\,4 \\ &= 1\text{ ms.} \end{aligned}$$

$$\begin{aligned} T_{\text{HIGH}} &= w\, T_{\text{OSC}}\, x && \text{(from (4))} \\ &= w\,(250\text{ ns})\,(4) \\ &= w\,(1\ \mu\text{s}). \end{aligned}$$

Now values of $w \geq 1000$ lead to a 100% duty cycle so, for practical purposes, we should limit the range of w to $w \in [0, 1000]$.

Let us now determine the number of bits of accuracy that can be achieved by the pulse width modulator. The 10-bit value w which can be used to control T_{HIGH} should be constrained so that

$$\eta = \frac{w}{4(y+1)} \leq 1. \qquad \text{(from (7))}$$

In other words, the maximum value $w_{\max}$ for w is given by

$$w_{\max} = 4(y+1). \tag{11}$$

But from (1)

$$T_{\text{PWM}} = 4(y+1)T_{\text{OSC}}\, x$$

so

$$w_{\max} = 4(y+1) = \frac{T_{\text{PWM}}}{T_{\text{OSC}}\, x}.$$

or

$$w_{\max} = \frac{f_{\text{OSC}}}{x\, f_{\text{PWM}}}. \tag{12}$$

Because the number of bits needed to represent a number is given by its logarithm to the base 2, we have

$$\boxed{\#\text{ of bits of resolution} = \log_2\left(\frac{f_{\text{OSC}}}{x\, f_{\text{PWM}}}\right)} \tag{13}$$

or, equivalently,

$$\#\text{ of bits of resolution} = \frac{\log\left(\frac{f_{\text{OSC}}}{x\, f_{\text{PWM}}}\right)}{\log 2} = \frac{\ln\left(\frac{f_{\text{OSC}}}{x\, f_{\text{PWM}}}\right)}{\ln 2}. \tag{14}$$

The PIC16F87xA data sheet states the maximum number of bits for the pulse width modulator as

$$\#\text{ of bits of resolution} = \log_2\left(\frac{f_{\text{OSC}}}{f_{\text{PWM}}}\right). \tag{15}$$

This is correct as stated—but it's not very helpful if $x \neq 1$. Equation (13) is the most general result.

Continuing the example above, where we already have realized that we need to keep w in the range $w \in [0, 1000]$, we can now calculate the effective resolution:

$$\begin{aligned} \text{Resolution} &= \log_2 \left(\frac{f_{\text{OSC}}}{x\, f_{\text{PWM}}} \right) \text{ bits} \\ &= \log_2 \left(\frac{4\,\text{MHz}}{4\,(1\,\text{kHz})} \right) \text{ bits} \\ &= 9.97. \end{aligned} \tag{16}$$

So although we no longer achieve the full 10 bits accuracy we sought originally, we do get a value that is close to it. Whether it is sufficient depends on the application.