TMS320 DSP
DESIGNER'S NOTEBOOK
Number 86
TEXAS INSTRUMENTS

Symmetric PWM Outputs Generation with the TMS320C14 DSP

Contributed by Zhenyu Yu

Design Problem

How do I generate symmetric PWM outputs with the TMS320C14?

Solution

Pulse Width Modulated (PWM) signal generation is crucial to many motor and motion control applications. PWM signals are pulse trains with fixed frequency and magnitude and variable pulse width. There is one pulse of fixed magnitude in every PWM period. However, the width of the pulses changes from period to period according to a modulating signal.

When a PWM signal is applied to the gate of a power transistor, it causes the turn on and turn off intervals of the transistor to change from one PWM period to another PWM period, according to the same modulating signal. The frequency of a PWM signal is usually much higher than that of the modulating signal, or the fundamental frequency, such that the energy delivered to the motor and its load depends mainly on the modulating signal.

Figure 1 shows two types of PWM signals: symmetric and asymmetric. The pulses of a symmetric PWM signal are always symmetric with respect to the center of each PWM period. The pulses of an asymmetric PWM signal always have the same side aligned with one end of each PWM period.

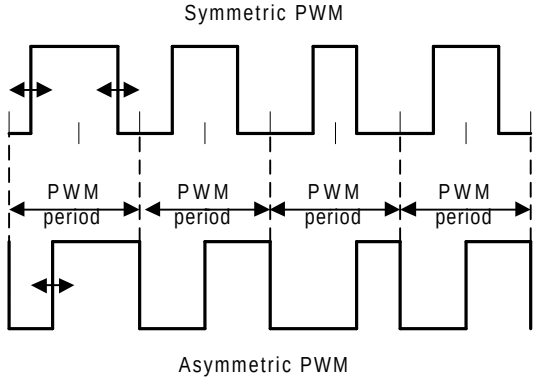

Figure 1. Symmetric and asymmetric PWM signals

It has been shown that symmetric PWM signals generate fewer harmonics in the output currents and voltages.

The TMS320C14 can generate 6 high-resolution asymmetric PWM outputs as represented by CMP0 and CMP1 in Figure 2. A small amount of off-chip logic and code lines can be used to easily generate 3 symmetric PWM outputs from the 6 asymmetric PWM outputs. Figure 2 shows how to generate one symmetric PWM output.

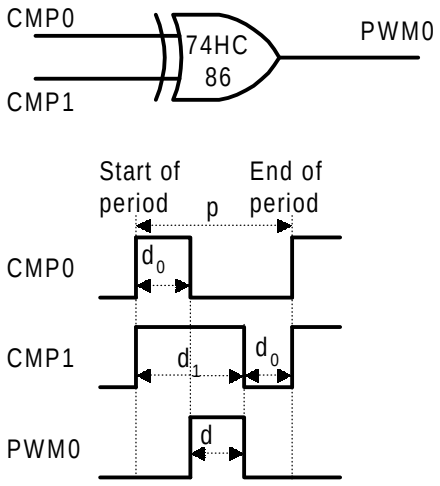

Figure 2. Symmetric PWM output generation with TMS320C14

Figure 2 also shows the waveforms for PWM0 with compare/PWM outputs CMP0 and CMP1 of the 'C14, where:

p is the period value for the selected timer and determines the PWM carrier period,
d is the desired pulse width for the symmetric PWM output for this PWM period,
d0 is the derived compare value for Compare Unit 0, and
d1 is the derived compare value for Compare Unit 1,
all in terms of CPU clock periods.

The actual PWM carrier period is given by: p*(CPU clock period). The smallest CPU clock period for the TMS320C14 is 160nS. The desired pulse width d in terms of CPU clock period is obtained by: d=(desired pulse width)/(CPU clock period). Given the desired pulse width d in terms of CPU clock period, the compare values for Compare Units 0 and 1 are determined by: d0=0.5(p-d), d1=d0+d.

Example

Using the scheme in Figure 2, one can evaluate the variables needed to generate a pulse width of 16uS assuming a PWM carrier period of 49.92uS (that is, a PWM frequency of 20.032KHz) and a CPU clock period of 160nS. The calculations are as follows:

p = 49.92 uS / 160 nS = 312;
d = 16 uS / 160 nS = 100;
d0 = 0.5 * (312 - 100) = 106;
d1 = d0 + d = 106 + 100 = 206.

Code lines to do the above are as follows:

```
;
; Other code lines to determine d
;
; Calculate d0 and d1 from d based on d0=0.5*(p-d), d1=d0+d
lac    p               ; load period
sub    d               ; calculate p-d
sacl   temp            ; save p-d
lac    temp,15         ; load p-d, left shift 15 bits
sach   temp0           ; save d0=0.5*(p-d)
addh   d               ; calculate d1=d0+d
sach   temp1           ; save d1
;
; Load compare (shadow) registers
lack   #BANK4          ;
sacl   temp            ;
out    temp,BSR        ; select shadow reg bank
out    temp0,ACT0      ; load shadow register 0
out    temp1,ACT1      ; load shadow register 1
;
; Continue
```

The logic in Figure 2 can be repeated for CMP2 and CMP3, and for CMP4 and CMP5. This way, up to 3 symmetric PWM outputs can be generated with the 6 asymmetric PWM outputs of a ‘C14 and one 74HC86. The calculation needed to determine the pulse width, d, for each symmetric PWM output is performed just like in any normal motor and motion control application. The only software overhead is the subtraction, left shift and addition needed to determine d0 and d1.