Application Report
SPRA771 - August 2001

Motor Speed Measurement Considerations When Using TMS320C24x DSPs

Shamim Choudhury DCS Applications

ABSTRACT

The TMS320C24x™ generation of DSPs provide appropriate internal hardware for interfacing with low-cost, external-speed sensors for motor speed measurement applications. The periodic output signal from the speed sensor is applied to the capture input pin of the DSP and the signal's period is measured. This information is then used to calculate the motor speed. However, this calculation of motor speed depends on several system parameters. These parameters affect the scaling and normalization factors that must be used in the speed calculation routine for accurate measurements. This application report, therefore, gives an analysis of the speed measurement system to show the effect of system parameters on the calculated speed. The choice of appropriate scaling and normalization factors for a given system is also discussed. Finally, code examples are given to show the software implementation of the speed calculation routine.

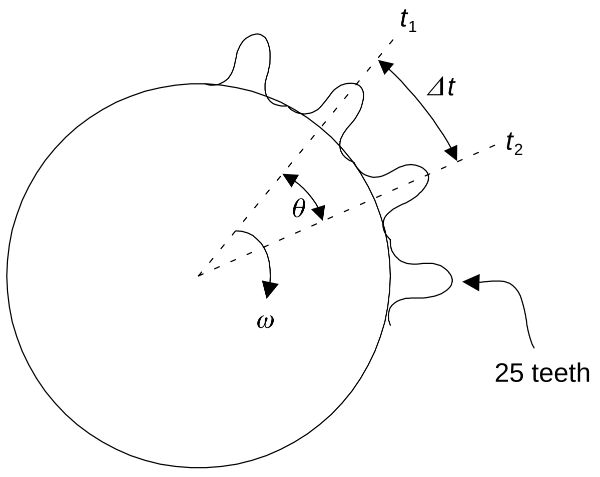
Contents

List of Figures

1 Motor Speed Measurements

Low-cost shaft sprockets, having multiple teeth, are widely used with the Hall effect gear-tooth sensor for motor speed measurements. Figure 1 shows some of the physical details of a sprocket that affect the speed measurement. The Hall effect sensor generates a square-wave output signal every time a tooth rotates within its proximity. If the sprocket has n teeth, then the resultant pulse rate is n pulses per revolution. The Hall effect sensor output is fed directly to the C24x™ Capture input pin. This unit captures the value of its base timer counter on either the rising or falling edges (whichever is specified) of the Hall effect sensor output. The captured value is then stored in a variable for later use in speed calculation. We will call this variable *time_stamp*.

During the implementation of the software for the speed calculation, for every time a new value of *time_stamp* becomes available the current value is first saved in a variable. Let us call this variable *time_stamp_old*. Then the new value is loaded in another variable, which we will call *time_stamp_new*. The two values are then compared and, thus, the tooth-to-tooth period (t_2–t_1) value is calculated. In order to reduce jitter or period fluctuation, an average of the most recent n period measurements can be performed each time a new pulse is detected.



$$\Delta t = t_2 - t_1 \text{ sec}$$

$$\theta = \frac{360}{25} = 14.4^\circ$$

$$\omega = \frac{2\pi}{\sum_{n-1}^{25} \Delta t_n} \text{ rad/sec}$$

Figure 1. Speed Measurement With a Sprocket

From the two consecutive *time_stamp* values, the difference between the captured values is calculated as,

$$\Delta = \text{time\_stamp\_new} - \text{time\_stamp\_old}$$

Then the time period in seconds is given by

$$\Delta t = t_2 - t_1 = K_P \times T_{CLK} \times \Delta$$

where K_P is the prescaler value for the capture unit time base. T_{CLK} is the CPU clock period in seconds. CPU clock is used as capture unit time base with user selectable prescaler K_P.

From Figure 1, the angle θ in radian is given by

$$\theta = \frac{2\pi}{n}$$

C24x is a trademark of Texas Instruments.

where n is the number of teeth in the sprocket. Then the speed ω, in radian/sec, and the normalized speed ω_N are calculated as

$$\omega = \frac{\theta}{\Delta t} = \frac{2\pi}{n\Delta t} = \frac{2\pi}{n \times K_P \times T_{CLK} \times \Delta}$$

$$\Rightarrow \omega_N = \frac{\omega}{\omega_{\max}} = \frac{\omega}{2\pi\left(\frac{1}{n \times K_P \times T_{CLK}}\right)} = \frac{1}{\Delta}$$

Here, ω_{max} is the maximum value of ω, which occurs when Δ=1. Therefore,

$$\omega_{\max} = \frac{2\pi}{nK_P T_{CLK}}$$

The corresponding maximum speed, N_{max} in rpm, that can be measured is calculated as

$$N_{\max} = \frac{60}{nK_P T_{CLK}}$$

2 Effect of System Parameters on Speed Calculations

TMS320C24x generation of DSPs provides different CPU clock frequencies and different clock prescaler values for it's capture units. These will affect the software implementation of the speed measurement algorithm. Also, depending on the number of teeth in the sprocket and the actual maximum speed of the motor being used, the implementation of the speed measurement algorithm needs simple modifications for accurate calculation of motor speed. In this section a few examples are provided to show the necessary modifications for correct measurement of motor speed.

2.1 Case1

For a system based on 20MHz CPU clock (T_{CLK} = 50 x 10^{-9} sec), K_P = 32 and n = 25, the normalized speed ω_N, when normalized with respect to maximum measurable speed is

$$\omega_N = \frac{\omega}{2\pi(25000)} = \frac{1}{\Delta}$$

where the maximum measurable speed, in rad/sec, is

$$\omega_{\max} = \frac{2\pi}{nK_P T_{CLK}} = 2\pi(25000)$$

This corresponds to a maximum speed of 1,500,000 rpm that can be measured using this system. Now, in any practical implementation the maximum motor speed will be significantly lower than this maximum measurable speed. So, for example, if the motor used has a maximum operating speed of 23,000 rpm, then the calculated speed can be expressed as a normalized value with a base value of normalization of at least 23,000 rpm. If we choose this base value of normalization as 23,438 rpm (the reason for the choice of such a value is explained later), then the corresponding base value of normalization, in rad/sec, is

$$\omega_{max1} = \frac{23438 \times 2\pi}{60} \approx 2\pi(390)$$

Therefore, the scaled normalized speed is calculated as

$$\omega_{N1} = \frac{\omega}{2\pi(390)} \approx \frac{64}{\Delta} = 64 \times \omega_N$$

This shows that the scaling factor, in this case, is 64. Notice that for this system having a maximum measurable speed of 1,500,000 rpm, the choice of a base speed of 23,438 rpm results in a scaling factor of 64 (= 1500000/23438). With this scaling factor, the above multiply operation can also be implemented simply by left shifting ω_N by 6 bits. While the multiply operation can be performed in either way, the latter results in faster execution time.

Now the maximum value of the scaled normalized speed ω_{N1} is 1. When expressed as a Q15 number, this maximum value of ω_{N1} is 32,767 (7FFFh). Therefore, in this case the maximum value of ω_N is 32767/64 when expressed in Q15. Now, during the software implementation, this is saved as a 16-bit number. So for the maximum accuracy of ω_N, while avoiding an overflow condition, it should be expressed in Q21. This maximum value of ω_N, in Q21, is 32,767.

The speed, in rpm, is calculated as

$$N_1 = 23438 \times \omega_{N1} = 23438 \times \frac{64}{\Delta}$$

Note that when the motor runs at it’s maximum speed of 23,000 rpm, the measured value of Δ will be at its minimum. This minimum value will be $\Delta = 65$ for this system (fastest time for 1 rev = $65 \mathrm{x} 25 \mathrm{x} 32 \mathrm{x} 50 \mathrm{x} 10^{-9}$ sec = 0.0026 sec).

2.2 Case 2

If the system parameters remain the same as in case 1 but the motor used has a maximum operating speed of 5500 rpm, then we have, as before,

$$\omega_N = \frac{\omega}{2\pi(25000)} = \frac{1}{\Delta}$$

and

$$\omega_{\max} = \frac{2\pi}{nK_P T_{CLK}} = 2\pi(25000)$$

Now, assume that we normalize the calculated speed with respect to a base value of 5859 rpm. The corresponding base value of normalization, in rad/sec, is

$$\omega_{max2} = \frac{5859 \times 2\pi}{60} \approx 2\pi(98)$$

Therefore, the scaled normalized speed is calculated as

$$\omega_{N2} = \frac{\omega}{2\pi(98)} \approx \frac{256}{\Delta} = 256 \times \omega_N$$

This shows that the scaling factor in this case is 256. Here again, the choice of the base value of 5859 rpm results in the scaling factor of 256 (= 1500,000/5859). With this value the multiply operation can also be performed by left-shifting ω_N by 8 bits, which saves computation time.

Again, the maximum value of the scaled normalized speed ω_{N2} is 1. In Q15, this is 32767 (7FFFh). Therefore, in this case, the maximum value of ω_N is 32767/256 in Q15. So, for maximum accuracy of ω_N while avoiding an overflow condition, it should be expressed in Q23. This maximum value of ω_N, in Q23, is 32767.

The speed, in rpm, is calculated as

$$N_2 = 5859 \times \omega_{N2} = 5859 \times \frac{256}{\Delta}$$

Note that in this case, when the motor runs at it's maximum speed of 5500 rpm, the minimum value measured for Δ will be Δ = 273. This means that the fastest time for 1 revolution is, 273 x 25 x 32 x 50 x 10^{-9} sec (= 0.01092 sec).

2.3 Case 3

Now, if we have a different system based on 20MHz CPU clock (T_{CLK} = 50 x 10^{-9} sec), K_P = 4, and n = 25, then the normalized speed is

$$\omega_N = \frac{\omega}{2\pi(200000)} = \frac{1}{\Delta}$$

where the maximum measurable speed used for the above normalization, in rad/sec, is

$$\omega_{\max} = \frac{2\pi}{nK_P T_{CLK}} = 2\pi(200000)$$

Now, assume that the motor used has a maximum operating speed of 5000 rpm. If we normalize the calculated speed with respect to the same maximum value of 5000 rpm, then the corresponding base value of normalization, in rad/sec, is

$$\omega_{max3} = \frac{5000 \times 2\pi}{60} \approx 2\pi(83)$$

Therefore, the scaled normalized speed is calculated as

$$\omega_{N3} = \frac{\omega}{2\pi(83)} \approx \frac{2400}{\Delta} = 2400 \times \omega_N$$

In this case, the calculated scaling factor is 2400. Here again, the maximum value of ω_{N3} is 1. In Q15, this is 32767 (7FFFh). Therefore, the maximum value of ω_N in this case is 32767/2400 when expressed in Q15. So for maximum accuracy of ω_N while avoiding an overflow condition, it should now be expressed in Q26. This maximum value of ω_N, in Q26, is 27961.

The speed, in rpm, is calculated as

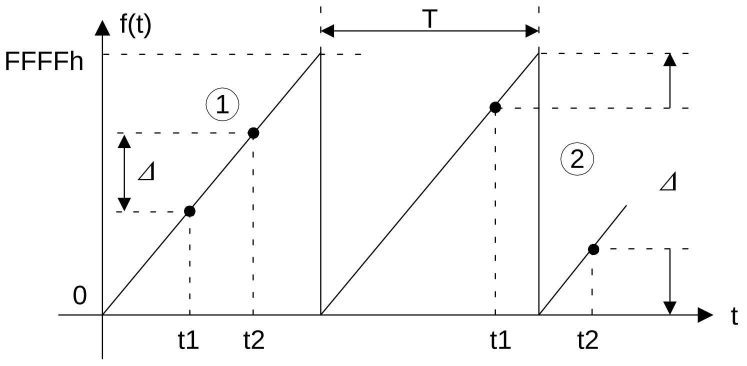
$$N_3 = 5000 \times \omega_{N3} = 5000 \times \frac{2400}{\Delta}$$

Note that in this case when the motor runs at it's maximum speed of 5000 rpm, the measured minimum value of Δ will be Δ = 2400. This means the fastest time for 1 revolution is, 2400 x 25 x 4 x 50 x 10^{-9} sec (= 0.012 sec).

For all the three cases discussed above, it is evident that the scaled normalized speeds ω_{N1}, ω_{N2}, and ω_{N3} are all represented as Q15 numbers. Because of this and the choice of the system parameters, the normalized speed ω_N (speed normalized with respect to maximum measurable speed) should be represented in different Q formats to achieve maximum accuracy. Once ω_N is represented in a particular format for maximum accuracy, the range of speed calculation will be limited to the corresponding chosen base speed of normalization. For example, to achieve maximum accuracy for the speed range in Case 3 (normalized with respect to 5000 rpm), if ω_N is expressed in Q26, then the maximum speed that can be calculated is 5000 rpm. However, if ω_N is expressed in Q21, then the maximum speed that can be calculated is 23438 rpm (when appropriate values are selected for the maximum speed and the scaling factor). However, this also means that the ability to calculate the speed in a wider range is achieved by compromising the accuracy of speed calculation at the lower speed range.

3 Speed Calculation from Period Measurements

The capture unit in the C24x allows accurate time measurement (in multiples of clock cycles and defined by a prescaler selection) between events. In this case the events are selected to be the rising edge of the incoming pulse train. What we are interested in here is the delta time between events. To implement this, Timer 1 is allowed to free run with a user-selectable prescaler K_P and a CPU clock with a time period of T_{CLK} sec. The delta time Δ, in scaled clock counts, is calculated as shown in Figure 2.



Case 1 $f(t_2) \geq f(t_1)$

$\Delta = f(t_2) - f(t_1)$

Case 2 $f(t_2) \leq f(t_1)$

$\Delta = 1 + f(t_2) - f(t_1)$

Note: only true if $t_2 - t_1 < T$

Figure 2. Calculation of Speed

In Figure 2, the vertical axis *f*(*t*) represents the value of the timer counter, which is running in continuous up count mode and resetting when the period register = FFFFh. Note that two cases need to be accounted for: the simple case where the timer has not wrapped around and where it has wrapped around. By keeping the current and previous capture values, it is easy to test for each of these cases.

Once the value of Δ is known, the speed is calculated using the appropriate equations explained before. In order to maintain high precision in the calculation for the full range of motor speeds, a 32-bit/16-bit division is performed, as shown in Figure 3.

$$\frac{1}{period} = \frac{7FFFFFFF(Q31)}{period(Q0)} = speed(Q31 \rightarrow 32bit)$$

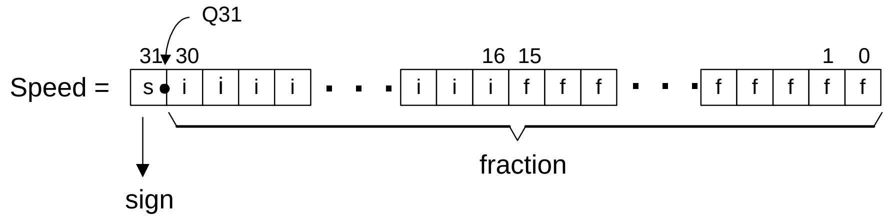


Figure 3. 32-Bit/16-Bit Division

After the division is performed, the result will be a 32-bit value in Q31 format. This value is subsequently scaled to a 16-bit, Q15 format.

4 Code Examples

Two code examples for speed calculation are presented here to illustrate the software implementation of the three cases explained before. It is assumed that the difference in captured values is saved in the variable called *event_period*.

4.1 Case 1

```
SPEED_SCALER_     .set  64
RPM_MAX_          .set  23438 ;Max rpm value. Base value for normalization

;Init variables:
      LDP   #speed_scaler
      SPLK  #SPEED_SCALER_, speed_scaler
      SPLK  #RPM_MAX_, rpm_max
      …………….
      …………….

;Calculate speed, i.e. speed = 1/period
;Numerator (i.e. 1) is in Q31 format
;Phase 1
      LACC  #07FFFh           ;Load Numerator Hi
      RPT   #15
      SUBC  event_period
      SACL  speed_hi
      XOR   speed_hi
      OR    #0FFFFh           ;Load Numerator Lo
;Phase 2
      RPT   #15
      SUBC  event_period
      SACL  speed_lo
      LACC  speed_lo
      ADDH  speed_hi          ;Result is in Q31 in 32 bit format

;End of 32bit/16bit division. The result is in ACC, in Q31, 32 bit format.

      RPT   #5
      SFL                     ;Q37
      SACH  speed_prd_max     ;Q21

      SPM   0
      LT    speed_prd_max     ;Q21
      MPY   speed_scaler      ;Q0*Q21
      PAC                     ;Q21
      RPT   #2
      SFL                     ;Q24
      SACH  speed_prd,7       ;Q15, speed_prd= speed_scaler*speed_prd_max

      LT    speed_prd         ;Q15
      MPY   rpm_max           ;Q0
      PAC
      SACH  speed_rpm,1       ;Q0
```

4.2 Case 2

In this case, the initialization and the division part remains the same. If the user also wants to use the same code for a wider speed range (0~23438 rpm) in a different application, then the only change needed here is in the max speed and the scaling factor (as explained before). However, if maximum accuracy is desired for this speed range (0~5000 rpm), then the normalized speed needs to be expressed in Q26. In that case, the later part of the code changes as shown below.

```
SPEED_SCALER_     .set  2400
RPM_MAX_          .set  5000  ;Max rpm value. Base value for normalization

;Init variables:
(Same as before)

;Calculate speed, i.e. speed = 1/period
(Same as before)

;End of 32bit/16bit division. The result is in ACC in Q31, 32 bit format.

      RPT   #10
      SFL                     ;Q42
      SACH  speed_prd_max     ;Q26

      SPM   0
      LT    speed_prd_max     ;Q26
      MPY   speed_scaler      ;Q0*Q26
      PAC                     ;Q26
      SACH  speed_prd,5       ;Q15, speed_prd= speed_scaler*speed_prd_max

      LT    speed_prd   ;Q15
      MPY   rpm_max           ;Q0
      PAC
      SACH  speed_rpm,1       ;Q0
```