Application Note, V 1.0, September 2005

AP0801510

XC866

Two Phase Stepper Motor Control with the XC866

Microcontrollers

Never stop thinking.

Revision History:	1.0	**V 1.0**
Previous Version:	-	

Page	Subjects (major changes since last revision)

Controller Area Network (CAN): License of Robert Bosch GmbH

We Listen to Your Comments

Any information within this document that you feel is wrong, unclear or missing at all? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:
mcdocu.comments@infineon.com

Edition 2005-09-01

Published by
Infineon Technologies AG
81726 München, Germany

1 Introduction

Small Two Phase Stepper Motors are popular for instrumentation since they provide easy to see information to the end user. Applications such as vehicle instrument clusters employ these types of motors in high volumes. This ApNote describes how to control two phase stepper motors with the powerful CAPCOM6E peripheral of the Infineon XC866 8-bit 8051 based flash microcontroller.

2 Two Phase Stepper Motor Structure

A two Phase Stepper Motor is a four pin device that contains two coils and a permanent magnet rotor as shown in Figure 1. The coils can be controlled independently and can carry either positive or negative current by controlling the voltages. A ferrite metal usually channels the flux created by the coils closer to the rotor so that the air gap can be reduced.

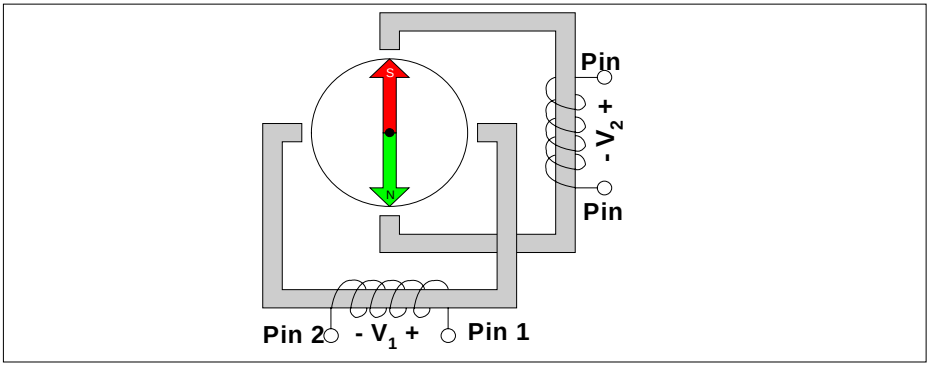

Figure 1 Two Phase Stepper Motor Structure

One or more gears are often connected to the rotor. A pointing device may also be connected to the gears so that one revolution of the rotor shaft moves the pointer only a few degrees or even just a fraction of a degree.

3 Two Phase Stepper Motor Operation

When current flows through the stator coils, flux is produced and channeled through the ferrite material and then jumps the air gap to interact with the rotor. The rotor tries to align itself with the stator flux. This is similar to the behavior of a switched reluctance motor. There are several different methods that can be used to control the position of the stepper motor.

3.1 Full Stepping

By energizing one coil at a time, the rotor can make one complete rotation by taking 4 steps as shown in figure 2. This is referred to as “Full Stepping”. Each full step is 90 electrical degrees (0°, 90°, 180°, 270°, 360°).

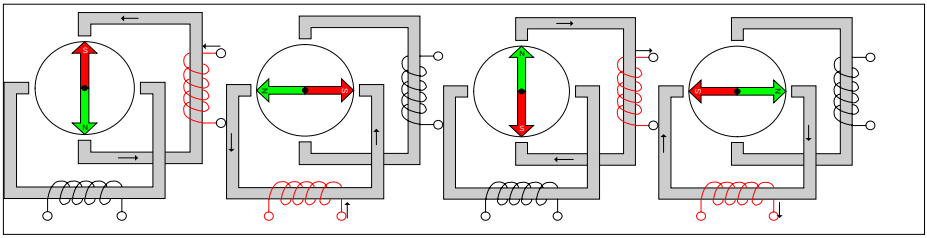

Figure 2 Full Stepping a Stepper Motor

3.2 Half Stepping

By energizing both coils at the same time the rotor can move into a position half way between two full steps (45°, 135°, 225°, 315°). This is called Half Stepping. Using a combination of Full Steps and Half Steps, the motor can be moved 8 steps per revolution as shown in figure 3 giving a resolution of 45 electrical degrees.

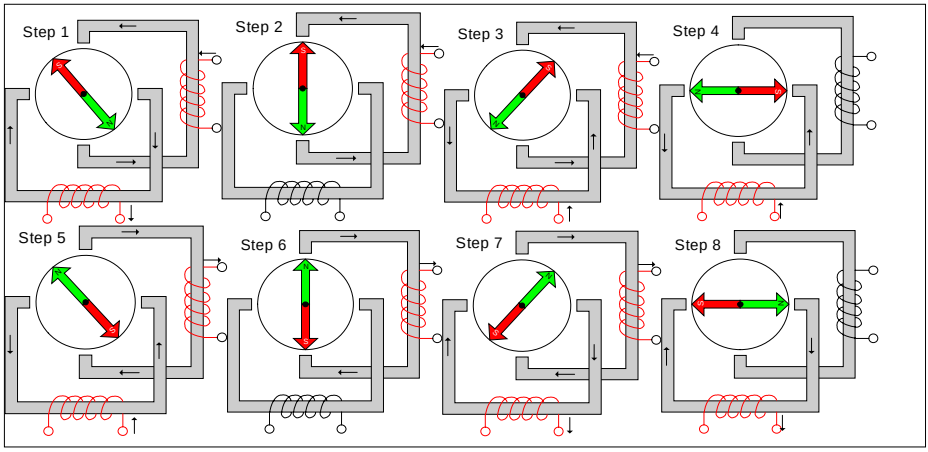

Figure 3 Half Stepping a Stepper Motor

To produce the currents in the motor coils for full/half stepping, both ends of both coils must be able to be driven high or low. The coil voltages for the full/half stepping pattern in Figure 3 are shown in Figure 4 (the block voltages). Figure 4 assumes the coils are labeled as shown in Figure 1.

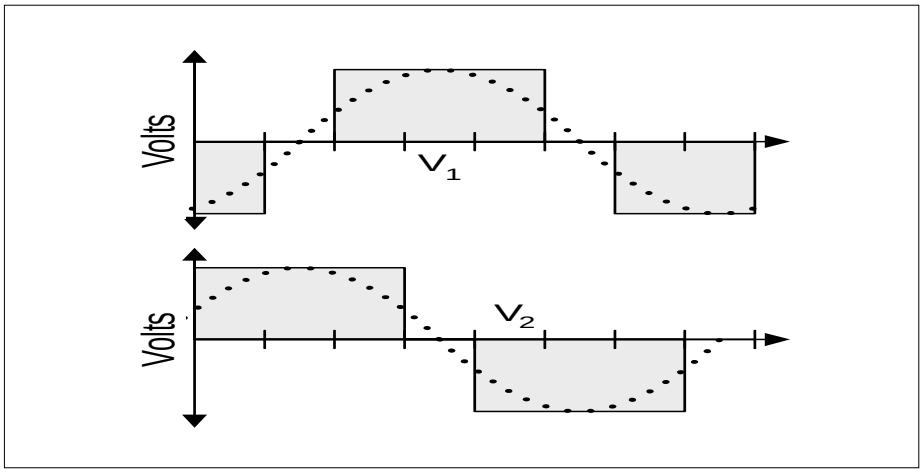

Figure 4 Coil Voltages for Half Stepping a Stepper Motor

The block voltages shown in figure 4 are shifted 90° from each other. The dotted lines in Figure 4 show a smoothed sinusoidal version of the coil voltages. One voltage could be considered a sine function of the desired rotor position, and the other can be considered a cosine function.

3.3 Micro Stepping

It is possible to get even higher resolution than 45 degrees if a wider range of voltages can be applied to the coils. The resolution of the rotor position is then dependent on the resolution of the voltage that can be applied (and mechanical tolerances of the motor and gears). Ideally, sinusoidal voltages can be applied to the coils to produce the maximum rotor resolution (see Figure 4) and smooth rotor movement. This is referred to as micro-stepping.

Pulse Width Modulation (PWM) signals can be generated by a microcontroller to produce sinusoidal voltages. To produce sinusoidal voltages on a coil, only one channel of PWM is required. One end of the coil can be driven by a PWM signal to control the magnitude of the voltage and the other end of the coil can be held high or low to control the direction of the voltage as shown in Figure 5.

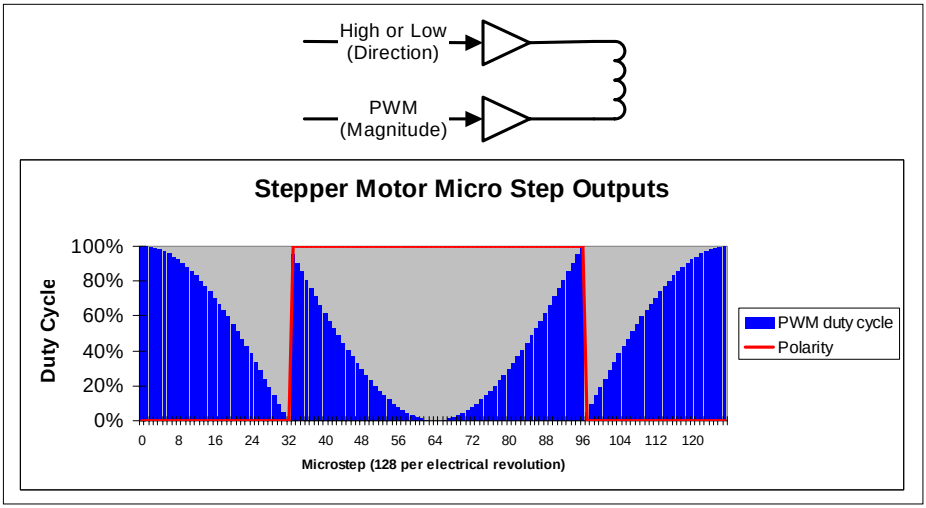

Figure 5 Using PWM to Produce Sinusoidal Voltages

By driving Coil 1 with a sine function and Coil 2 with a cosine function, the rotor angle can be controlled to a high degree of accuracy. If 8-bit PWM is used to drive the motor, then the rotor can achieve 512 discrete positions per revolution.

4 Determination of Step Timing

There is no rotor position measurement device in most stepper motor systems. Therefore the microcontroller must assume that the rotor aligns itself with the commanded angle. If the assumed and actual rotor angles differ enough, then the rotor can take a step backwards.

To ensure that the motor never misses a step, the microcontroller must ensure that the rotor is never commanded to accelerate or decelerate beyond its physical limits. Figure 6 shows the shape of a torque/speed curve for a hypothetical stepper motor. The torque that the motor can produce decreases as the motor speed decreases.

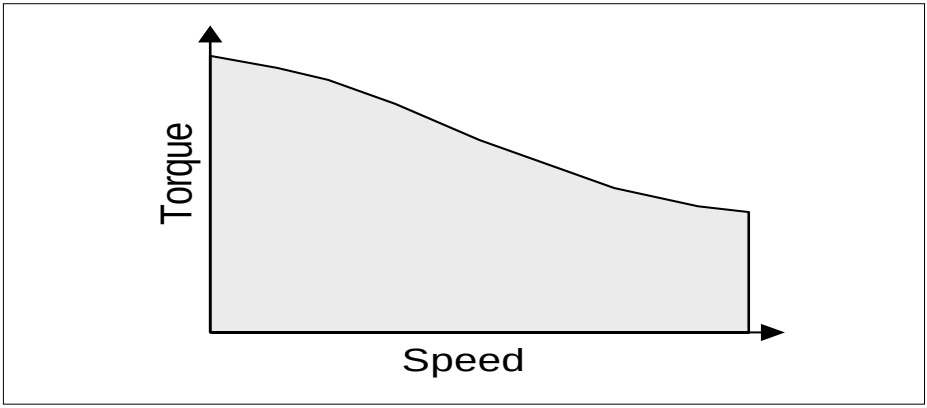

Figure 6 Typical Torque/Speed Curve a Two Phase Stepper Motor

So at higher speeds the motor cannot accelerate or decelerate as quickly as it can at lower speeds. For safe operation the speed and acceleration of the motor must be closely controlled.

Acceleration can be thought of as $\Delta\omega/\Delta t$. $\Delta\omega$ is the difference between current velocity and desired velocity. Δt is not quite so straight forward. For a stepper motor to safely increase or decrease its speed, it should reach the desired velocity after two full steps of rotation (180°). If the motor is capable of changing from the current velocity to the commanded velocity in the amount of time that it takes to make two full steps (at the desired velocity), no steps will be lost. If however, the actual rotor position ever falls more than 180° behind the commanded position, a step will be lost.

To ramp the motor from stand-still up to its maximum speed in the fastest amount of time without loosing any steps, the motor should be driven at a constant speed for 180°. Then the speed can be increased for another 180°, and so on, until the desired motor speed has been reached. To determine the maximum speed for each 180° of

rotation, some off-line calculations need to be done using the Torque/Speed curve of the motor.

The torque produced by a motor can be expressed as:

$$T = J \cdot \alpha \qquad [1]$$

... where J is the moment of Inertia of the motor and load in $\left[\text{Kg} \cdot \text{m}^2\right]$

... α is the rotor accelerati on in $\left[\frac{\text{rad}}{\text{sec}^2}\right]$

In this case we know J and T (as a function of speed) from the motor data sheet. We are trying to find the maximum values of α that will be just under the maximum torque that the motor can produce.

α as mentioned before is $\Delta\omega$/(2 full step times). The time for two full step is π/ω. So equation [1] can be re-written as:

$$T = J \cdot \frac{(\omega - \omega_0) \cdot \omega}{\pi} \qquad [2]$$

... where ω is the desired velocity $\left[\frac{\text{rad}}{\text{sec}}\right]$

... ω_0 is the current velocity $\left[\frac{\text{rad}}{\text{sec}}\right]$

4.1 Start/Stop Frequency

Given equation [2] and the motor torque speed curve, we can determine the start/stop frequency of the motor. If the motor is at rest, the start/stop speed is the maximum speed which the motor can achieve after 180° of rotation (two full steps). If the motor is spinning at the start/stop speed, then it can stop spinning within two full step times.

To determine the start/stop frequency of a stepper motor, equation [2] is used with ω_o equal to zero. If the equation for the motor torque/speed curve is known, then equation [2] (with ω_o=0) can be solved analitically for ω. However since many motor manufactures only specify the motor torque/speed curve graphically, the start/stop frequency must be found iteratively by using guesses for ω. The result of equation [2] can then be plotted on the torque/speed curve to find the intersection of the two lines.

For example, given a rotor and load intertia of $700 \cdot 10^{-9}$ $\text{Kg} \cdot \text{m}^2$ we can calculate the torque required to accelerate from a stop to various speeds as shown here:

$$T_{0-\omega} = J \cdot \frac{\omega^2}{\pi}$$

$$T_{0-200} = 700 \cdot 10^{-9} \cdot \frac{200^2}{\pi} = 8.91\,mNm$$

$$T_{0-400} = 700 \cdot 10^{-9} \cdot \frac{400^2}{\pi} = 35.7\,mNm$$

$$T_{0-600} = 700 \cdot 10^{-9} \cdot \frac{600^2}{\pi} = 80.2\,mNm$$

The results of the calculations above can be plotted directly onto the torque/speed curve from the motor data sheet as shown in Figure 7.

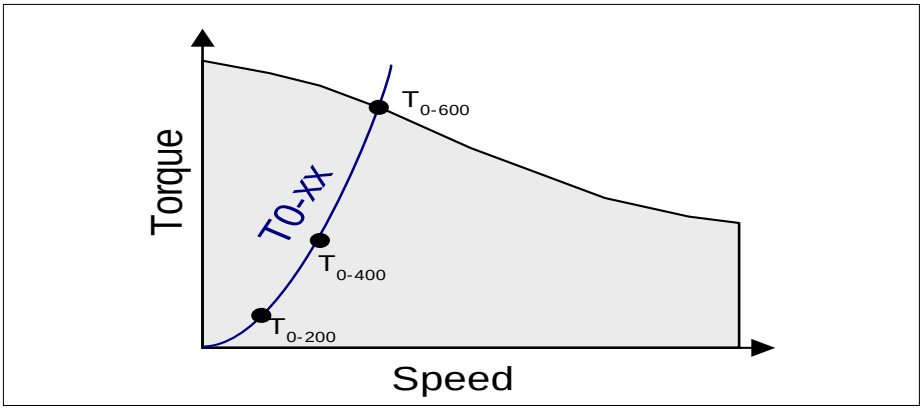

Figure 7 Graphical Determination of Maximum Start/Stop Frequency

Figure 7 shows that the maximum start/stop speed of the motor is approximately 600 rad/sec. So if the motor is initially stopped, it can be driven up to 600 rad/sec in two full steps.

4.2 Maximum Acceleration Frequency

Equation [2] can also be used to find the torque required to accelerate or decelerate from any initial speed to any desired speed. For example, if the motor described in the

previous section can accelerate from a stop to 600 rad/sec in two full steps, we can use 600 rad/sec for ω_0 in equation [2], and generate another Torque/Speed curve as was done previously. The intersection of the curve generated from equation [2] with the motor torque/speed curve will give the maximum speed that the motor can accelerate to from 600 rad/sec in two full steps. The process can then be repeated many times using the previous result for ω_0. Typical curves for such a process can be seen in Figure 8.

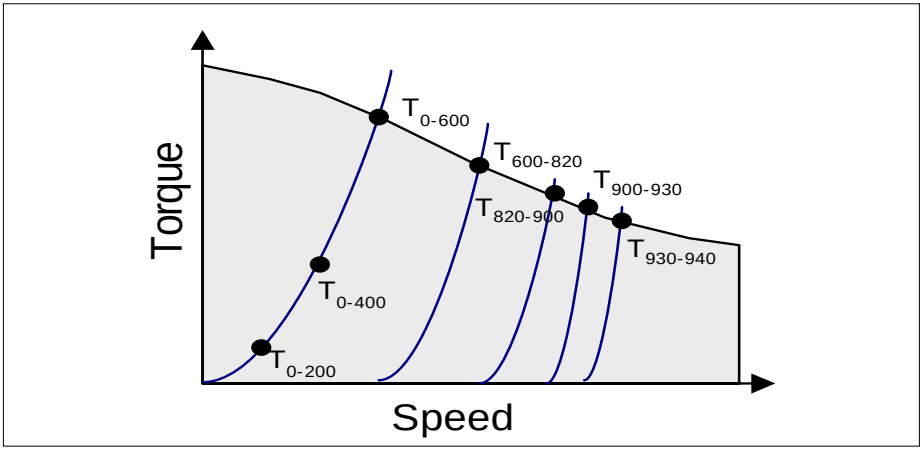

Figure 8 Graphical Determination of Maximum Acceleration Ramp

As Figure 8 shows, to accelerate the motor from rest to its maximum speed as fast as possible without loosing any steps, the motor should be run at several discrete speeds. Each speed should be maintained for two full steps. The relationship between the motor speed and the amount of time that the speed should be maintained (two full steps) is straight forward:

$$TwoStepTimes = \frac{\pi}{\omega}$$

Table 1 shows the time that each speed must be maintained for the motor shown in Figure 8. The motor can be accelerated from a stop to full speed (940 rad/sec) after 10 full steps.

Table 1 Full Step Timing for Maximum Motor Acceleration

Speed [rad/sec]	Time For each Speed [s]
0	N.A.
600	π÷600
820	π÷820
900	π÷900
930	π÷930
940	π÷940

If the motor is to be driven in full step mode, the time for each full step would be half of that shown in table 1. If the motor is to be driven in half step mode, then the time for each half step would be ¼ that shown in table 1. In micro-step mode, the time for each micro-step would be the value from table 1 divided by the number of micro-steps in 180°. In any case, the time between steps should be maintained for 180° of rotation.

4.3 Vibration and Safety Factor

Using above methods to determine the maximum motor acceleration does not leave any margin for safety, error or unbalanced vibration. To take vibration and a safety factor into account, the motor torque should be adjusted as shown in Equation 3.

$$T_{Adjusted} = (T - T_{Vibration}) \cdot SafetyFactor \qquad [3]$$

$$T_{Vibration} = OffBalance[N \cdot m] \cdot VibrationLevel[G's]$$

Vibration effects any part of the load that is out of balance. So the “vibration torque” is calculated as the load off-balance in Newton-Meters multiplied by the applications maximum G’s.

The Safety Factor is just a proportional de-rating of the motor torque to provide an acceptable margin for volume production of the system.

Figure 9 shows the graphical determination of the maximum acceleration taking into account vibration and a safety factor.

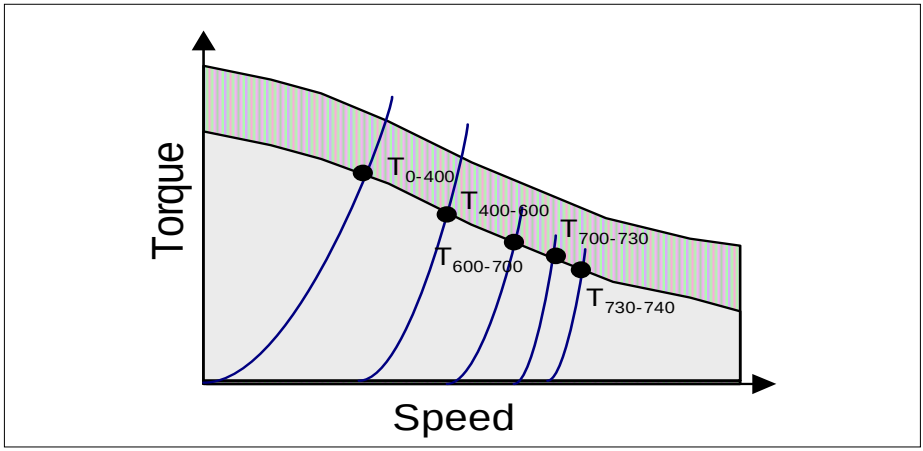

Figure 9 Graphical Determination of Maximum Acceleration Ramp

5 XC866 Implementation of Stepper Motor Control

The Infineon XC866 8-bit microcontroller has a powerful motor control peripheral, the CAPCOM6E. The CAPCOM6E can generate glitch free patterns for controlling a stepper motors in full-step, half-step or micro-step modes.

Note: The control algorithm and implementation proposed in this application note is only one of many possible solutions. Your application may have different requirements. The implementation proposed in the application notes is not optimized.

The stepper motor used in this application is geared, with a gear reduction ratio of 180:1. This means that when the rotor of the motor moves 180 degrees, the pointer that is connected to the gears moves one degree. This gives the pointer much higher resolution.

To spin the motor smoothly and further increase the resolution of the pointer, the motor will be driven in micro-step mode. The software is setup so that 24 micro-steps will be performed per revolution of the rotor.

5.1 Software Overview

The main object of the software is to drive the pointer that is connected to the motor via the gears to any desired position smoothly and safely. The software to control the motor consists of two parts. The first part is a periodic position and speed controller. The second part is a micro-step function that is executed every micro-step synchronously to the motor. Figure 10 shows how these two functions work together.

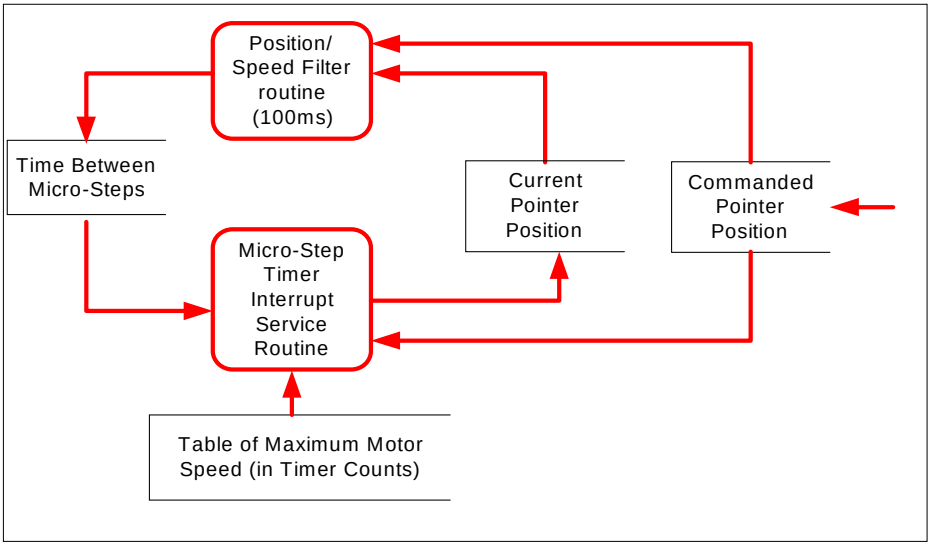

Figure 10 Data Flow Diagram of Stepper Motor Software

5.2 Postion/Speed Control Function

The position and speed controller is executed periodically, in this case every 100ms. The 16-bit Timer 0, (T0) is used to trigger this function. The job of this function is to filter the motor movement so that the pointer can come slowly to rest at the requested position. This type of movement is more pleasing to the eye and also more beneficial to the user since they can approximate the final pointer position before it is reached. The effect of noise on the requested pointer position is also reduced.

A first order filter is used to calculate the desired path that the pointer should follow as shown in the following formula:

$$NewPos = CurrentPos + K \cdot (RequestedPos - CurrentPos) \qquad [4]$$

...where K is the filter constant and less than one

If the pointer position follows Equation [4], the path that the pointer would follow after a step change in the requested pointer position would look similar to the exponential curve shown in Figure 11.

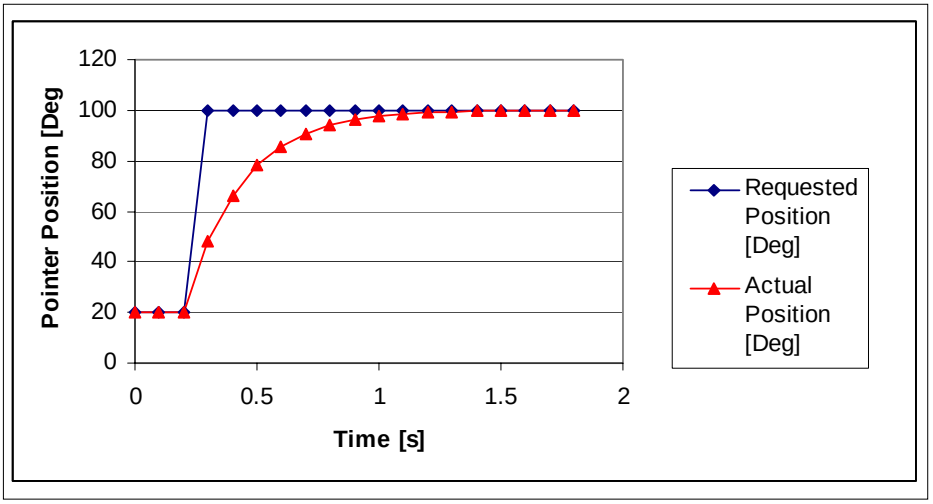

Figure 11 Fitler Response to step change in requested poiner position

To get the motor to follow the desired path, the time between micro-steps must be controlled. Each time the filter is activated (every 100ms) the commanded speed of the motor is calculated. Equation [5] shows how the motor speed is calculated.

$$velocity = \frac{NewPos - CurrentPos}{100ms}$$

$$velocity = \frac{K}{100ms} \cdot (RequestedPos - CurrentPos) \qquad [5]$$

The position is measured in pointer degrees, therefore the velocity is calculated in degrees per second. Once the desired motor speed is known, it must be converted into timer ticks between micro-steps. The number of timer ticks between micro-steps is used by the micro-step timer interrupt service routine to adjust the motor speed. The flow chart for the 100ms routine is shown in Figure 12.

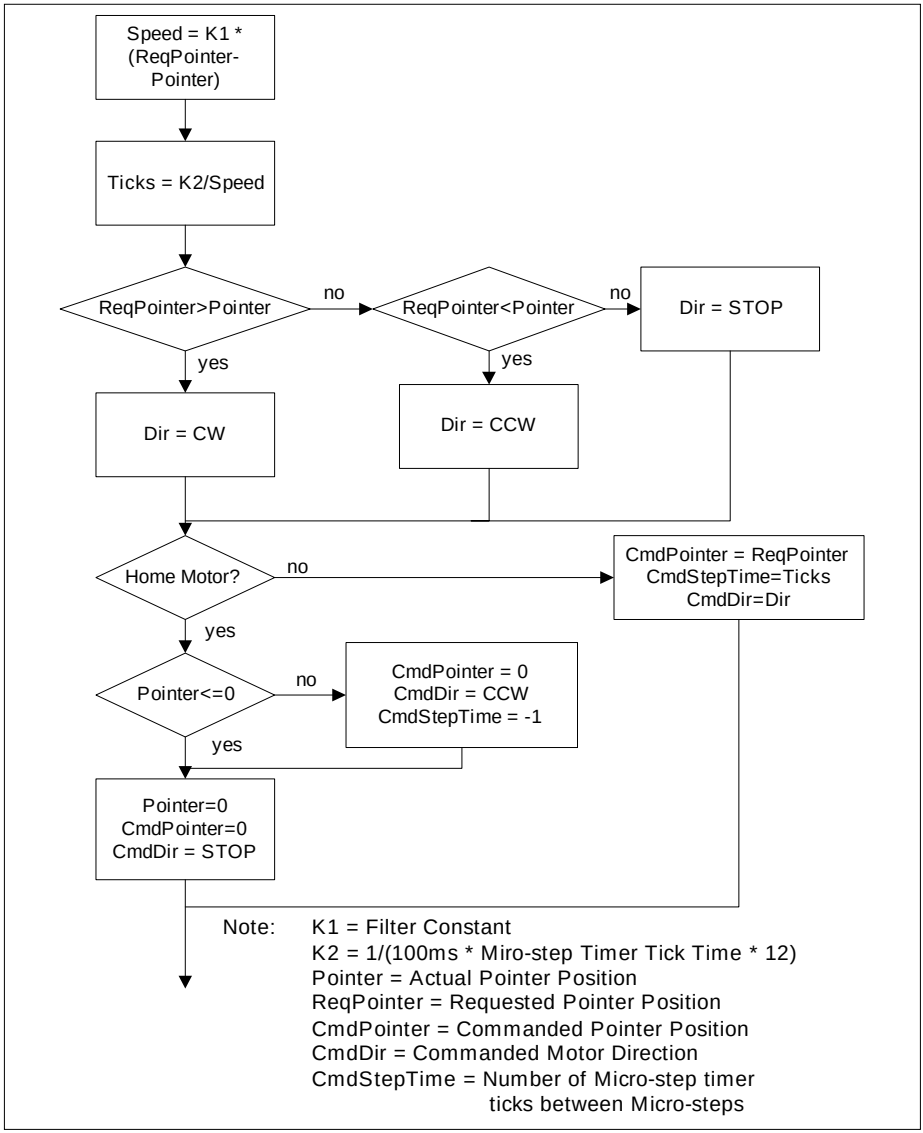

Figure 12 Filter Routine Flow Chart

5.3 Micro-step Control Function

The micro-steps are controlled by two 16-bit timers on the XC866. Timer 12 (T12) is used to as the time base for PWM generation. PWM in produced on two channels to drive the two coils of the motor. Although T12 is 16-bits wide, the period register is set to 0x00FF so the timer acts like an 8-bit timer.

Timer 2 (T2) is a general purpose 16-bit timer on the XC866. Timer 2 is used to generate an interrupt whenever it is time to take another micro-step. So the T2 interrupt service routine could also be called the micro-step interrupt service routine. T2 can only interrupt the CPU after every overflow, so to time the micro-steps accurately, after each interrupt the timer is manually re-loaded so that the next interrupt occurs after the desired micro-step time (which was previously calculated in the 100ms filter routine).

Since this routine operates synchronously to the motor, it is called many times more often than the 100ms filter function when the motor speed is high. If the motor speed is low, the micro-step function may be called less often than the 100 ms filter function.

The micro-step timer interrupt service routine is responsible for controlling the PWM duty cycles and pin states required to make the motor micro-step. This routine also keeps tract of the position of the pointer.

5.3.1 Micro-stepping the motor

Two tables of sinusoidal PWM (one sine table and one cosine table) and a table of pin states (high or low), each containing 24 entries (since there are 24 micro-steps per revolution) are used to control the micro-steps. Each time the micro-step interrupt occurs (and the motor is supposed to be moving), an index into these tables is incremented or decremented depending on the direction the motor is supposed to spin. The PWM values from the tables are copied into PWM shadow registers and the pin state values are copied from a table into a shadow registers. When the PWM timer reaches zero, the new values of the PWM and output pins are simultaneously applied (from the shadow registers) to the motor for a glitch-less micro-step. This is done automatically by hardware. Figure 13 shows a graphical representation of the tables used in the software.

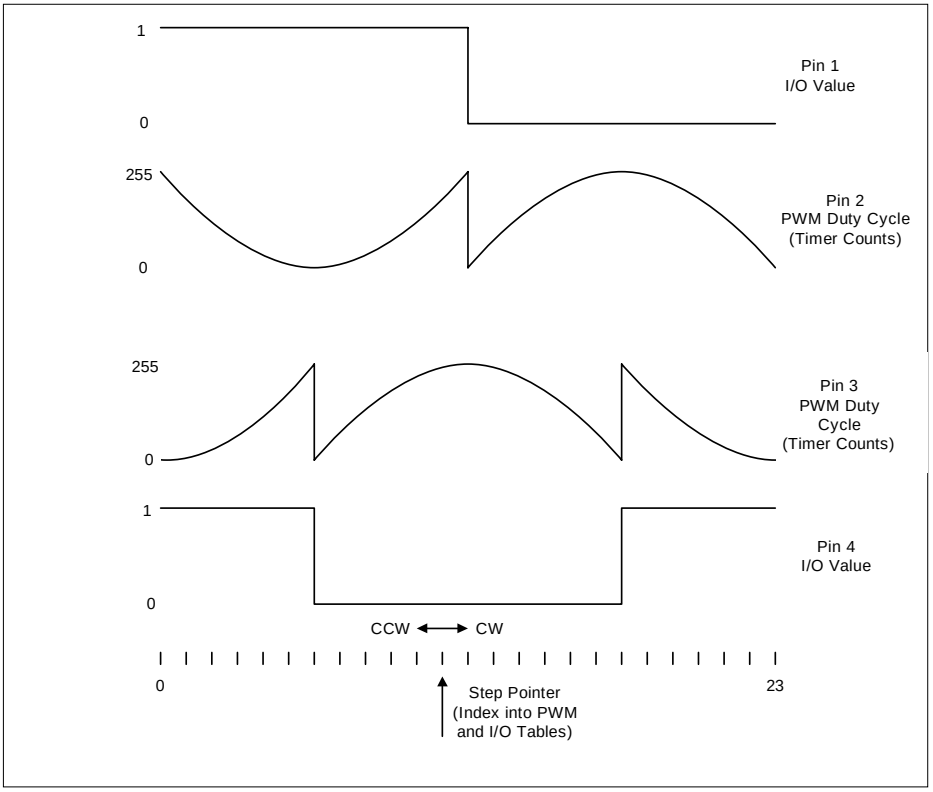

Figure 13 Graphical Representation of Tables used the the Software

Incrementing the index into the tables causes the motor to spin clockwise and decrementing the index causes the motor to spin counter clockwise. This makes it convenient to make a “direction” variable that can hold the values -1, 0 and 1. Adding the direction variable to the index will cause the motor to spin counter clockwise, stop or clockwise (respectively).

5.3.2 Verification of Commanded Micro-step Time

Since the motor must maintain a speed for 180 degrees of rotor movement (one degree of pointer movement), the micro-step timer (T2) reload value is only updated every 12 micro-steps. Before the value is updated, it is checked to see if the update would cause acceleration or deceleration that would violate the motor capabilities.

The process of checking to see if the speed that is commanded by the periodic filter function can be achieved by the motor is implemented in a way so that the CPU time required to do the calculations is reduced. However other methods may be used to optimize this process even more.

First a table is made of the maximum motor acceleration ramp from standstill to the max motor speed using the method described in Chapter 4. The shaded values in Table 2 show the results of the calculations for the motor and pointer that is used in this Application Note. For this Application Note, T2 is counting at a rate of 0.45 µs per tick, and there are 12 micro-steps for each full step (180° of rotor movement, 1° of pointer movement). The non-shaded lines are the speeds that are half way between two shaded speeds. The shaded and non-shaded values are both used by the software as will be described latter.

Table 2 Timing for Maximum Motor Acceleration

Pointer Speed [Deg/sec]	Time for full step (1 deg.) [msec]	T2 Ticks per micro-step
0	N.A	N.A.
32	31.25	5787
64	15.63	2894
84	11.90	2205
104	9.615	1781
119	8.403	1556
134	7.463	1382
146	6.849	1268
158	6.329	1172
168	5.952	1102
178	5.618	1040
187	5.348	990
196	5.102	945
204	4.902	908
212	4.717	874

219	4.566	846
226	4.425	819
233	4.292	795
240	4.167	772
246	4.065	753
252	3.968	735

To control the acceleration of the motor, a look-up table is created similar to the last column of Table 2. The software has a variable that is an index into the table. The algorithm to control the motor acceleration works like this:

Assume that the current motor speed is 320 deg/sec, and the index into the table is pointing to the entry for 233 deg/sec. Also assume that the speed control function commands a new speed of 250 degrees per second. The software in the T2 ISR will see that the commanded speed cannot be safely achieved, so instead it increments the index into the table (so it points to the entry for 240 deg/sec) and sets the T2 reload value so that the timer will overflow again after 772 ticks.

After 12 micro-steps have passed (2 full-steps of the motor and 1 degree of pointer movement), if the commanded speed remains 250 deg/sec, the software will again increment the pointer into the table (so it point to the entry for 246 deg/sec) and set the T2 reload value so that the timer will overflow after 753 ticks.

After another 12 micro-steps, if the commanded speed remains 250 deg/sec, the software will set the T2 reload value so that the motor will spin at 250 deg/sec and the index into the table will not be incremented.

The table used in the software contains both positive and negative reload values to make the math easier (only positive values are actually used for the T2 reload).

A high level flow chart of the micro-step timer interrupt service routine can be seen in figure 14.

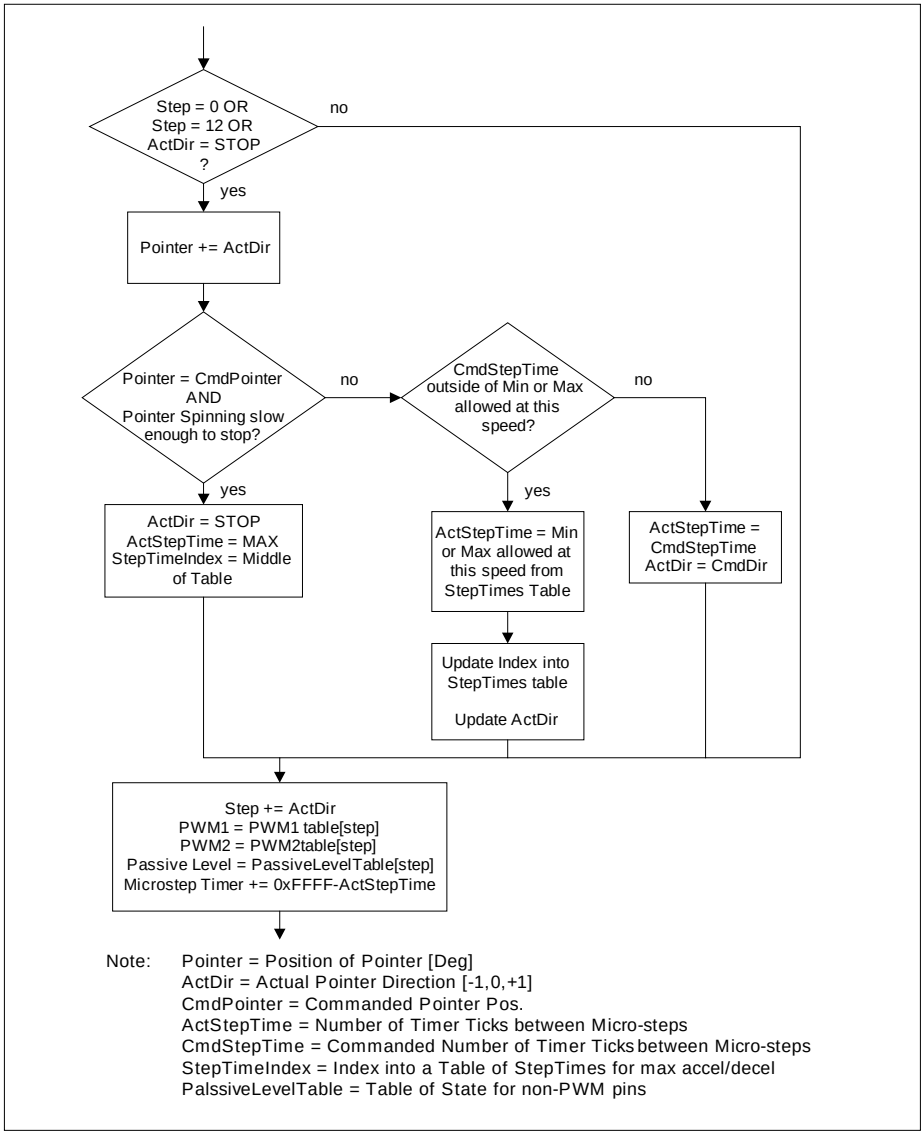

Figure 14 Micro-step Routine Flow Chart

5.4 Using the CAPCOM6E Peripheral for Glitch-Free Micro-stepping

Although it is not technically required for a stepper motor to operate, it is generally desired to have PWM and pin states that can be updated synchronously and without glitches. The CAPCOM6E unit of the XC866 (and many other Infineon microcontrollers) is capable of driving motors in this way.

There are two types of glitches that are eliminated by the CAPCOM6E. First, the PWM compare registers contain shadow registers. The actual PWM values are not updated until a shadow transfer bit is set and the PWM timer (T12) reaches 0. This prevents glitches that can occur if the actual PWM registers are updated asynchronously to the PWM timer.

Another kind of output glitch can occur if the PWM values are not updated at the same time as the non-PWMed pins that control the direction of the coil current. The same shadow transfer event that updates the PWM values can also update the state of the non-PWMed pins.

The state of the I/O pins can be controlled by the Passive State Level Registers (PSLR) in the XC866. The PSLR contains one bit per CAPCOM6 I/O pin. Updating the value of this register updates the actual pin value at the same instant as the PWM is updated.

Figure 15 shows the glitches referred to above and how they are avoided using the CACPOM6E.

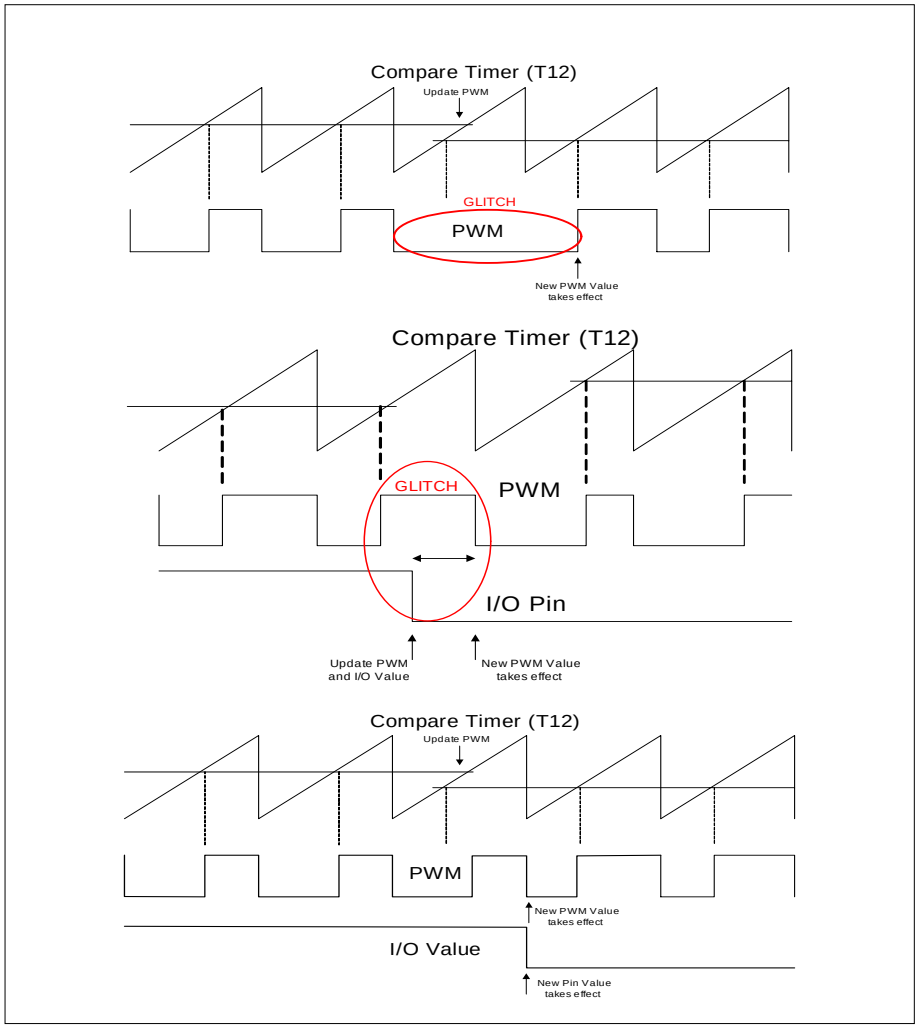

Figure 15 PWM Glitch (top), Pin State Glitch (middle) and Glitchless (bottom) micro-stepping

The software for this Application Note was written using DAvE, the free code generation tool from Infineon, and the Keil uVision3 embedded development environment and C51 compiler.

DAvE is available for download at: www.infineon.com/DAvE

An evaluation version of the Keil C51 tools is available at: www.keil.com

6 Conclusions

In this application note, the construction of two phase stepper motors was described. In addition various methods to control the motor were described (full-stepping, half-stepping, micro-stepping). Finally an actual implementation of a typical stepper motor control system using Infineon's XC866 8-bit microcontroller was described, and the software for such a system is provided as a reference.

In particular, the capability of the CAPCOM6E peripheral to easily and efficiently generate the necessary stepper motor signals in a glitchless fashion was demonstrated.

The companion application code to this application note uses less than 2k of code space and only 40 bytes of RAM.

http://www.infineon.com

Published by Infineon Technologies AG