

#15: Voltage to Frequency Conversion for Stepper Control

By design, no motor is capable of changing its momentum faster than the system's electrical and mechanical time constants will allow. When operating a DC motor, if the command voltage changes faster than the motor's ability to keep up, extra current is pumped into the motor, increasing its torque, and in turn, increasing its speed. This means that a typical motor (DC, AC, Brush, Brushless, etc.) can actually LAG behind the velocity command signal indefinitely. The worst thing that will happen is that the motor will be "late" when arriving at the commanded position. This phenomenon is known as following error. Following error is the difference between where the control wants the motor to be and where the motor actually is, at any time.

A stepper motor, however, is a zero-following-error device. This means that when the controller generates the motion command, it expects the stepper motor to move.

But, the stepper motor has electrical and mechanical time constants just like any other motor. Put these two facts together and we have a requirement for zero-following error. In other words, the motion command signal cannot change its trajectory (or profiles) path faster than the stepper motor's ability to keep; this is a constraint of the system. Since the stepper is controlled by a pulse stream, it is required that the pulse stream be extremely stable in both frequency, and rate of frequency change (accel. and decel. profiles). If continuity is not maintained, the motor will slip, causing mispositioning, and possibly stall.

There are many methods of pulse generation in use today for stepper motor control. Following are some of the ways in which stepper motor pulses are generated, and a discussion of their benefits and/or traps.

Processor-generated pulse stream

Microprocessors have the ability to generate pulse streams. This method requires the use of an internal timer interrupt to ensure continuity of pulse stream generation. Since this is a "real time" effort, the only problem is in determining the maximum interrupt frequency (MIF) that can be tolerated by both the stepper (its maximum speed in PPS) and the processor (its interrupt handling capability). The MIF sets the top frequency that can be generated which, in turn, sets the top RPM of the stepper motor. When using a processor such as an Intel 8051, the top frequency is limited to around 20Khz. When using the Dallas 8051, the top frequency is limited to around 40Khz with the same control software. A DSP might be capable of generating pulses up to 100Khz for the similar control software. The

limitation using processor control, therefore, is the top frequency that the processor can generate while still doing the other control work (I/O sampling, operator interface, etc.).

ASIC-generated pulse stream

ASIC stands for Application Specific Integrated Circuit. This means is that the ASIC is designed to perform specific tasks. For stepper motor operation, the ASIC is the programmable frequency generator. Ultra precise ramping and slew pulses are produced from the ASIC, ensuring the best possible continuity for stepper motor control. Many of the stepper ASICs allow that the stepper's approximate velocity, and in some cases, the acceleration and deceleration ramps to be changed *on the fly*.

The limitation in using an ASIC to control a stepper motor is simply

ensuring that you understand all of the mathematical requirements for loading all of the ASIC's internal registers. Many ASIC manufactures supply math-handling routines which take the work out of this requirement, many don't. The math ritual, however, is always given in the ASIC documents. Moves such as multiple axis interpolation can then be reduced to simple software driver routines have provided a complete understanding of the ASIC exists. ASICs can generate pulse streams up

to 3MHz for high speed, microstepping applications.

V>F conversion of an Analog Signal

A V>F converter is a device that changes an analog voltage into a variable frequency. For example, it might be desirable to convert a voltage ranging from zero to ten volts into a frequency ranging from 100Hz to 100Khz. In theory, this seems like a good way of allowing a servo motor controller to control the actions of a stepper motor. There are, however, several major traps lurking in the woods:

The first trap is *Non-Linearity*. It is very difficult (and expensive) to convert a purely linear voltage into a purely linear frequency without digitizing. The result of simple analog conversion could yield 5% linearity right from the start, but remember that any sudden surge in frequency could cause stepper slippage, or stall. There is also an inability to produce a quality frequency when below 10%, and above 90% of the V>F limits.

The second trap is the problem of maintaining a stable frequency anywhere within the V>F's frequency range. The usual stability is in excess of 2%. Since a stepper motor requires continuity (ie: linearity)

A customer asked if the use of a V>F converter was practical for stepper motor control. My question to him was: How critical is your application?

TECH TIPS

#15: Voltage to Frequency Conversion for Stepper Control

in frequency generation, duty cycle, and frequency ramping, the V>F converter is not applicable nor is it practical for accurate position control. It is, however, a qualified device for simple *velocity* control.

Another trap is that the *typical* servo algorithm requires position error in order to generate the analog command signal for V>F use (position error is the difference between where the controller wants the motor to be, and where the motor position actually is). Since the stepper is a zero position error device, the implementation of velocity, and possibly acceleration feedforward gain *must* be present in the servo's gain algorithm. If it is not, the particular servo controller cannot be used without trajectory control modification. In other words, it becomes a *special device*.

And finally, because the servo algorithm expects a position error to occur, if the stepper begins to slip, the servo gain structure will try to speed up the motor in order to maintain a constant following error at any commanded velocity. Any attempt to speed up a slipping stepper motor will force it to prematurely stall.

When trying to implement the V>F approach, be certain you test it to ensure it can do what you want. If not, you may find yourself replacing the entire system with a properly qualified stepper or servo device that should have been used.

The use of a PWM servo output signal

Pulse width modulation (PWM) is another form of control used in the servo motor world. What

PWM does is change the On/Off duty cycle of a pulse to produce either an effective voltage to the servo amplifier (voltage averaging), (or it can happen in a window within which a time can be measured or current can be generated). As the requirement for a motor to speed up or slow down is invoked (accel. or decel.), the duty cycle of the PWM control signal will change. In order to use a PWM signal for stepper motor control, the PWM signal must first meet the minimum pulse width requirement of the stepper motor amplifier, and second, it must meet the frequency stability requirement of the stepper operation. A typical PWM signal generates a pulse stream based on a division of the clock frequency used to produce the PWM timing. The top pulse frequency is typically in the 10 to 30Khz range. (This is not very high when you consider an ASIC can do 3Mhz).

In order to maintain the PWM output within the stability requirements for good stepper control, the encoder resolution must match the step resolution of the stepper motor. In other words, one pulse out to the stepper must equal one pulse of feedback from the encoder. Since this is not possible continuously, the PWM pulse width (duty cycle) can vary causing the stepper to stall. In order to overcome this problem, a "virtual" encoder is developed within the servo controller to maintain perfect synchronization of motion and feedback. Does it work? Yes. How well? Reasonably. The problems are the PWM frequency, the clock frequency, the required stepper pulse width, and other tight coordination requirements between the servo controller's trajectory generator and the stepper motor's

ability to respond. In other words, if you're doing a simple velocity move without position qualification, it could work. But, if you need to be position accurate, this is not the approach to use.

In the haste to make things simpler, cheaper, faster, smoother, etc., the requirements of each control device must be qualified. If it was so good, or so easy, everyone would be using it. But there's more to it. Systems require careful analysis to ensure that the control fits the mechanical requirements, not the other way around. Although the stepper motor is a good cost effective method of obtaining motion, do not cheapen up the system to the point where it fails to perform ■

About the Author:

In his more than two decades in the industry, **Chuck Raskin, PE, CMCS**, has contributed to many industry publications, including *Motion*, *Motion Control*, & *PCIM*, and finished on the fifth edition of the *Designing with Motion Handbook*.

Chuck is currently the manager of technical services for Technology 80 and a board member of the American Institute of Motion Engineers (AIME).



Liked what you've read? Now you can buy the book!

Now in its fifth edition, Chuck Raskin's *Designing With Motion Handbook* will be a valuable resource for you regardless of your level of expertise. Considered to be the single best text in the motion control industry, it is used as a core text in the American Institute of Motion Engineers' Certification Program.

With more than three decades of industry experience, Chuck's 450+ page book contains valuable insights and examples that will help you balance all of the mechanical, electrical and software considerations of virtually any motion control application.

As a special offer, you can receive a FREE copy of The PC Handbook, a 96-page pocket-size reference that's a valuable resource when designing PC-based applications (a \$9.95 value). To order the handbook for only \$50, call 800-545-2980 and ask for the "TECH TIPS HANDBOOK OFFER."

Please include \$5 for shipping & handling. CA and MN residents also include appropriate sales tax.