Applying acceleration and deceleration profiles to bipolar stepper motors

By Jose I. Quinones
Applications Engineer

Introduction

With a DC motor, ramping up the voltage (or duty cycle if pulse-width modulation is being used) controls how fast the motor's shaft reaches any given speed. With stepper motors, however, changing the voltage does not have any effect on the motor speed. While it is true that changing the voltage changes the rate of current charge across the windings and thus the maximum speed the stepper can reach, the motor speed is set by the rate at which the current through the windings is switched, or commutated.

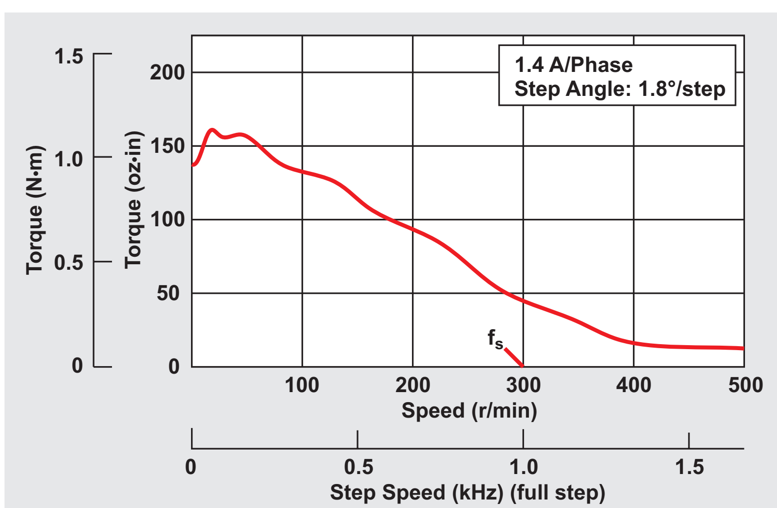
Can it be assumed that steppers are machines not requiring controlled acceleration profiles? If so, can steppers be run at any target speed desired without consequences? The truth is that stepper-motor motion needs to be actuated through acceleration and deceleration profiles more than any other motor topology. Trying to start at any speed may have dire effects.

In this article it is assumed that the reader is well-versed in how a commercially available integrated microstepping driver is used to control a stepper motor. The output of a stepper driver, such as the Texas Instruments (TI) DRV8818, is directly proportional to the frequency of a square wave (STEP input). Each STEP pulse equals a step (or microstep) as defined by the driver's stepping logic. Hence, changing the frequency of the square wave also changes the stepper's rate accordingly.

Figure 1 shows a motor manufacturer's conventional stepping rate/torque curve with an important parameter, f_S, called the starting frequency. It must be understood that, for this particular motor to start properly, a stepping rate smaller than f_S must be employed. To start the motor with a stepping rate larger than f_S may induce the motor to stall and lose synchronization. Once this happens, motion control is severely compromised. This appears to be a major problem but actually can be solved quite easily. All that is needed is to start the motor at a stepping rate below f_S and then increase the speed until the target speed is reached. Following this guideline, the stepper motor can be actuated with stepping rates far exceeding f_S—as long as the speed is kept below the shown torque/speed curve.

Equally important, one should not attempt to stop the motor simply by halting the STEP pulses. Instead, the stepping rate should be decreased from the target speed to a lower rate at which the motor can stop without the shaft inertia inducing extra and unwanted steps. Remember that if the stepper is being utilized in a positioning application, the motor shaft can lose position if it keeps on moving after it should have stopped. Since closed-loop position

Figure 1. Torque/speed curve for a bipolar constant-current stepper motor



feedback is seldom used for driving steppers, it is crucial to ensure that only the commanded steps take place.

Acceleration/deceleration profile

To accelerate a stepper from a starting speed to a desired target speed, the current speed just needs to be changed at periodic intervals. Most engineers use microcontrollers to achieve stepper control. The most common implementation uses only two timers. The first is a steps-per-second (SPS) timer used to generate an accurate timing function for the stepping rate. The second is an acceleration timer used to alter the first timer on a periodic basis. Since the speed is being changed at timely intervals, in essence the angular velocity with respect to time (dv/dt) is being derived. This derivation is called acceleration, or how speed changes across time. Figure 2 shows an enlarged view of a typical microcontroller-based acceleration profile and what is happening as the stepper is accelerated towards a target speed.

The SPS is the desired number of steps per second, or the stepping rate, at which the motor should move. The SPS timer must be programmed to issue pulses at this rate. Depending on the timer's oscillator frequency, a typical equation is

$$\text{SPS\_timer\_register} = \frac{\text{timer\_oscillator}}{\text{SPS}},$$

where SPS_timer_register is a 16-bit number that tells the timer how long it takes to generate subsequent STEP pulses, and timer_oscillator is a constant of how fast the timer is running in megahertz.

This equation is stored in a function because it is used quite frequently. To see how it works, assume that the timer oscillator is running at 8 MHz and the desired stepping rate for the motor is 200 SPS. According to the equation, the program code makes the value of SPS_timer_register equal to 40,000. So every 40,000 timer clicks, a STEP pulse is generated. This results in a timer-based output of 200 pulses per second and a shaft rotation equal to 200 SPS.

Every time such an event takes place, an interrupt is generated and the timer is cleared. The timing of the rising edge at the STEP input is crucial to the microstepping driver's accuracy, but the falling edge can happen at almost any time as long as it is well before the next STEP rising edge.

Two parameters are needed to define the acceleration curve: (1) how often to change the SPS value, and (2) by how much. The acceleration curve is directly proportional to both parameters; that is, the more often the SPS value is updated and the higher its value, the steeper will be the acceleration curve. The acceleration timer handles both parameters: The timer function fires as many times per second as is desired to change the SPS value, and the timer's interrupt-service routine (ISR) determines what the new speed is by incrementing the current SPS by a predetermined factor.

The acceleration rate is measured in steps per second per second (SPSPS), or by how many times per second the current SPS rate is changed. If the SPS value is changed by adding a one, the acceleration timer's ISR must be called (triggered) for each change in the acceleration rate. For example, with an acceleration rate of 1000 SPSPS, the motor speed can be started at 200 SPS and incremented by one until it reaches 1200 SPS. The acceleration timer's ISR would then need to be called 1000 times.

Another option is to call the acceleration timer half as frequently and then increment the SPS by two. Compared to the previous example, the acceleration timer's ISR is called only 500 times, but the motor still starts up at 200 SPS and reaches 1200 SPS within a second. The difference is more real-time availability at the expense of resolution. In other words, to achieve an accurate acceleration rate of 999 SPSPS, the first option must be used.

Figure 2. Close-up of a typical acceleration profile

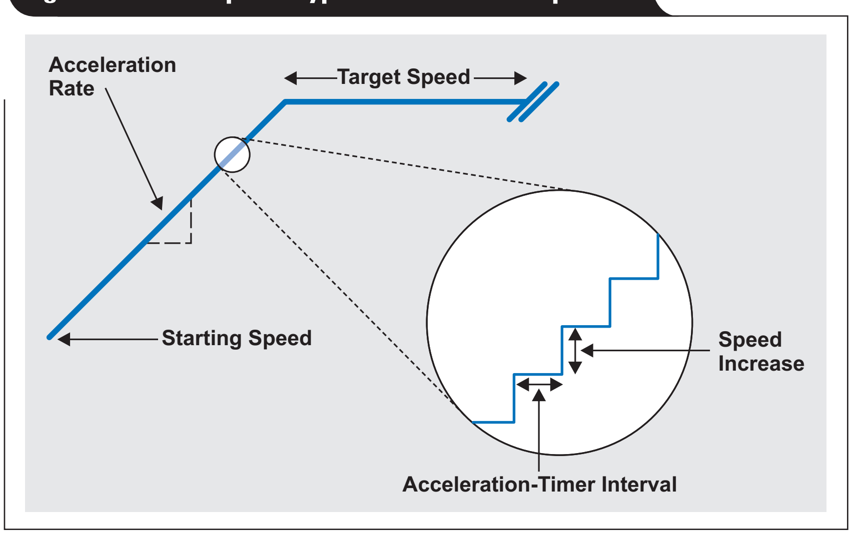
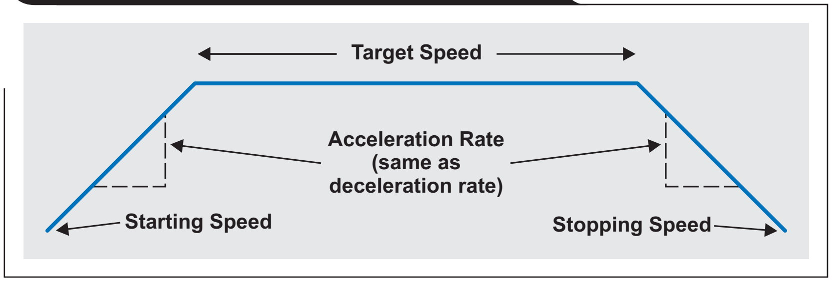


Figure 3. Acceleration/deceleration profile



The trade-offs of choosing one option versus the other must not be ignored, as the choice defines what kind of motion quality can be obtained. For instance, if a lot of granularity is required in order to achieve every possible acceleration profile, the acceleration timer's ISR will need to be called as much as possible.

However, in the SPS-timer equation given earlier, there is a division operation. Depending on which processor core is being employed, this division may considerably limit how many times the ISR can effectively be called and still correctly generate the new SPS rate. In an implementation using TI's MSP430™ with the CPU running at 16 MHz, a division operation takes about 500 µs. As a result, the most the ISR can be called per second is 2000 times. This limit then defines the incrementing factor. For any acceleration rate larger than 2000, an increment larger than one must be used.

The acceleration rate is computed once, shortly before the motor is started. The software in charge of this computation determines what the acceleration timer's interval and increment factor will be, then configures the variables accordingly. These variables are used concurrently until the SPS rate is modified enough to reach the target speed. Once the target speed is met, the acceleration profile ends.

The deceleration profile is basically identical to the acceleration profile, except that the increment factor is negative rather than positive. Also, a new target speed must be specified at which the motor can be safely stopped. Figure 3 shows an acceleration/deceleration profile where the acceleration and deceleration rates are symmetric. Asymmetric rates can also be employed.

Position control

Up to this point, operating the motor in a speed-control loop has seemed fairly simple. The motor is brought into a target speed and at some point commanded to stop. However, what happens when a predetermined number of steps needs to be executed in a predetermined amount of time? The acceleration/deceleration profiles then become more important than ever. In this motion-control topology, it is crucial that the motor stop when all the programmed steps have been executed. The variable that specifies how many steps will be issued is called number_of_steps.

The motion profile must be coded to make the motor stop at the required time rather than wait for a command to start deceleration. One way to achieve this is to program a variable called steps_to_stop to be smaller than number_of_steps. The software then determines when deceleration needs to be engaged by monitoring steps_to_stop.

Acceleration will not complete execution until the target speed has been reached. Once this happens, the stepper is allowed to run until it reaches the steps_to_stop count, at which time deceleration begins. For example, for a 1000-step run, steps_to_stop is set to 800. Hence, the motor is started via an acceleration profile and runs until step 800 is reached, at which time the motor decelerates until it stops.

Depending on how all of the system's variables are configured, five important scenarios need to be examined (see Figure 4).

Scenario 1: All steps are issued before the motor reaches the target speed.

Scenario 2: All steps are issued while the motor is at the target speed.

Scenario 3: All steps are issued before the stopping speed is reached.

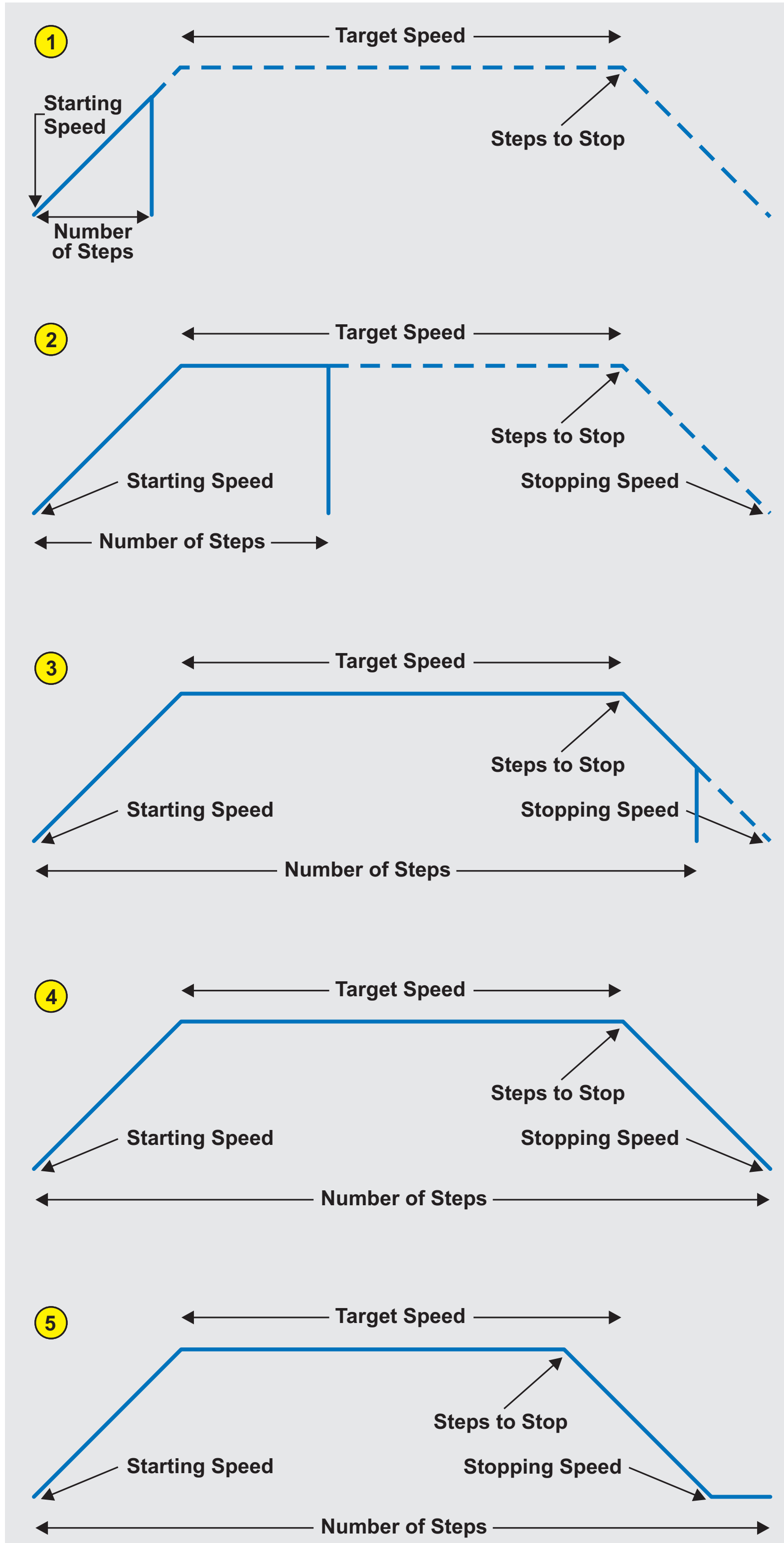
Scenario 4: All steps are issued as the stopping speed is reached.

Scenario 5: All steps are issued after the stopping speed is reached.

Stopping the motor right as the stopping speed is reached (Scenario 4) is the ideal case. Stopping the motor shortly before the stopping speed is reached (Scenario 3) or after it is reached (Scenario 5) can be acceptable depending on how many steps away from the ideal case these events occur. For instance, if all steps are issued while the motor is moving too fast, the motor shaft may lose position due to rotor inertia. But if the stopping speed is reached before all the steps are executed, the total time needed to execute the profile can become too long.

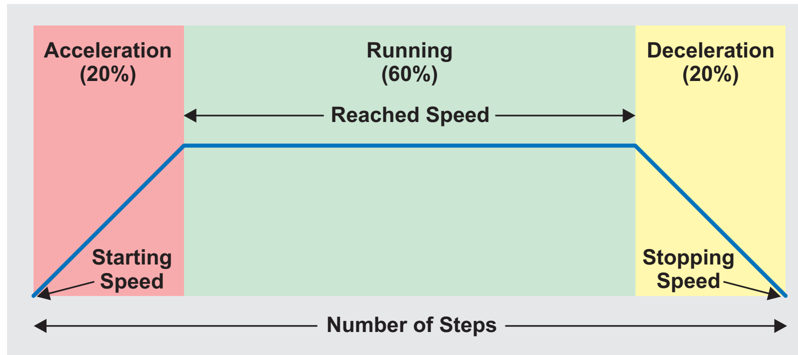
Scenarios 1 and 2, portrayed for illustrative purposes only, should not take place, as the designer should always ensure that steps_to_stop is smaller than number_of_steps. Knowing all the possible scenarios, the designer can easily tune the system to acquire the optimal response.

Figure 4. Five acceleration/deceleration scenarios



Another option that may result in less tuning is to segment the total number of steps into percentages assigned to each particular region of the acceleration/deceleration profile. In this algorithm implementation, 20% of the total number of steps can be selected to accelerate the motor, 60% to run the motor at a constant (reached) speed, and the remaining 20% to decelerate the motor (see Figure 5). If number_of_steps is 1000, the stepper accelerates at the programmed acceleration rate for 200 steps and stops acceleration at whatever step rate it reaches. It then executes 600 steps at this rate, with the last 200 steps being executed throughout the deceleration profile.

Figure 5. Acceleration/deceleration profile based on percentages



Notice that with an algorithm of this nature, assuming that the percentages are selected correctly, it is impossible to run out of steps on the wrong portion of the motion profile. For the example in Figure 5, since both the acceleration and deceleration portions are balanced, the motor most likely starts and stops at the same speed. The disadvantage of this method is that it is very hard to ensure what the target speed will be. If the target speed is not important, then this algorithm can be used to ensure that the motor will always stop at a safe speed.

If the speed reached is too slow for the application, the only means to speed up the motor shaft with this algorithm is to increase the acceleration rate or increase the percentages of the number of steps used in the acceleration/deceleration regions. However, the designer must be careful not to take the motor into a speed that violates the motor's torque/speed curve.

Conclusion

Accelerating and decelerating a bipolar stepper motor is a crucial part of designing any application that uses one. While power-stage control has been simplified considerably throughout the last decade, the application of acceleration and deceleration profiles still resides in the realm of the application's processor. Because of the wide availability of stepper solutions, the algorithms to process proper motion control for the application's stepper motor are easier to code and tune. By accelerating and decelerating the motor properly, the designer ensures that the application will operate efficiently and according to specifications.

Please see Reference 1 for more information about the code structure for an acceleration/deceleration-based implementation that revolves around a power stage similar to the DRV8818 and uses an MSP430 microcontroller.

Reference

For more information related to this article, you can download an Acrobat® Reader® file at www.ti.com/lit/*litnumber* and replace "*litnumber*" with the **TI Lit. #** for the materials listed below.

Document Title	**TI Lit. #**
1. Jose Quinones, "Intelligent stepper motor driver with DRV8811/18/24/25," Application Report	SLVA488

Related Web sites

www.ti.com/motor
www.ti.com/product/*partnumber*
Replace *partnumber* with DRV8811, DRV8818, DRV8824, or DRV8825

TI Worldwide Technical Support

Internet

TI Semiconductor Product Information Center Home Page
support.ti.com

TI E2E™ Community Home Page
e2e.ti.com

Product Information Centers

Americas	Phone	+1(972) 644-5580
Brazil	Phone	0800-891-2616
Mexico	Phone	0800-670-7544
	Fax	+1(972) 927-6377
	Internet/Email	support.ti.com/sc/pic/americas.htm

Europe, Middle East, and Africa

Phone

European Free Call	00800-ASK-TEXAS (00800 275 83927)
International	+49 (0) 8161 80 2121
Russian Support	+7 (4) 95 98 10 701

Note: The European Free Call (Toll Free) number is not active in all countries. If you have technical difficulty calling the free call number, please use the international number above.

Fax	+(49) (0) 8161 80 2045
Internet	www.ti.com/asktexas
Direct Email	asktexas@ti.com

Japan

Phone	Domestic	0120-92-3326
Fax	International	+81-3-3344-5317
	Domestic	0120-81-0036
Internet/Email	International	support.ti.com/sc/pic/japan.htm
	Domestic	www.tij.co.jp/pic

Asia

Phone

International	+91-80-41381665
Domestic	Toll-Free Number

Note: Toll-free numbers do not support mobile and IP phones.

Australia	1-800-999-084
China	800-820-8682
Hong Kong	800-96-5941
India	1-800-425-7888
Indonesia	001-803-8861-1006
Korea	080-551-2804
Malaysia	1-800-80-3973
New Zealand	0800-446-934
Philippines	1-800-765-7404
Singapore	800-886-1028
Taiwan	0800-006800
Thailand	001-800-886-0010

Fax	+8621-23073686
Email	tiasia@ti.com or ti-china@ti.com
Internet	support.ti.com/sc/pic/asia.htm