No More Missed Steps: Unlocking Precision with Closed-Loop Stepper Control

By Jose Quinones, Senior Application Engineer

Introduction

Bipolar stepper motors provide precise position control while operating in an open loop. Industrial automation applications — such as robots, processing, and packaging machinery — and consumer products — such as 3D printers and office equipment — effectively take advantage of the stepper's inherent position retention. This eliminates the need for convoluted sensor technology, processing power requirements, or complex control algorithms.

However, driving a stepper motor in an open-loop methodology requires the motion profile to be errorless. Any glitch in which the stepper's load abruptly changes results in step loss, which desynchronizes the stepper position from the application's perceived position. In most cases, this position tracking loss is problematic. For example, in a label printer, step loss could cause the print to be skewed with the label, resulting in skewed label prints. This article will describe a simple implementation that gives the stepper motor the ability to sense its position and actively correct any error that might accrue during actuation.

Assumptions

For the purpose of this article, we will assume that a bipolar stepper motor with 200 steps per revolution is employed to drive a mechanism that is responsible for opening and closing some sort of flap or valve while servicing a production line. To make motion smooth, we will utilize a bipolar stepper driver with 8 degrees of microstepping, resulting in 1600 step commands per full rotor revolution. In order to fully open or close said mechanism, we will need multiple rotor turns; for simplicity, assume we need 10 full turns. In this case, the controller would need to send 16,000 step commands on each direction to successfully actuate the mechanism.

When the current is high enough to overcome any torque variation, the stepper moves accordingly and can fully open and close the control surface. In this scenario, the position is preserved. If steps are lost, however, the controller loses synchronization with the motor and the actuation becomes compromised. Newer technologies attempt to provide checks, such as stall detection, by measuring the motor winding's back electromotive force (BEMF) when the applied revolving magnetic field crosses the zero-current magnitude. Stall detection only tells the application whether the motor is moving; it fails to report how many steps have been effectively lost. In cases like this, it is worthwhile to explore closing the loop on the rotor position using sensing technology.

Sensor Selection

In some cases, using simple limit switches (e.g. magnetic, optical, or mechanical) might suffice to drive the stepper motor until the limits are met. However, there are plenty of cases where the available space does not allow for the use of such switches. If a switch cannot be used, it might make sense to populate an optical shaft encoder (relative or absolute) at the motor's back side shaft, but there is a high cost associated with these solutions. An affordable solution for this dilemma is a contactless angular position sensor. This type of sensor involves the use of readily available magnetics with precise and accurate semiconductors that employ Hall sensors, which extract the rotor's position with as much as 15 bits worth of resolution. That means each rotor revolution can be encoded to as much as 2^{15} = 32,768 units, or 0.01 degrees (360/32,768).

For this example, an 11.5-bit resolution was selected, as that will be sufficient to encode the 1600 microsteps. By using 11.5 bits of resolution, we can obtain 2,896.31 effective angle segments. A Hall-effect based contactless sensor such as the MA732 provides absolute position encoding with 11.5 bits of resolution.

When coupled to a diametrically magnetized round magnet, the sensor is periodically sampled through its serial peripheral interface (SPI) port at 1ms intervals (see Figure 1). When a read command is issued, the sensor responds with a 16-bit word. The application uses the 16 bits worth of information, although the system's accuracy is driven by the effective 11.5-bit resolution.

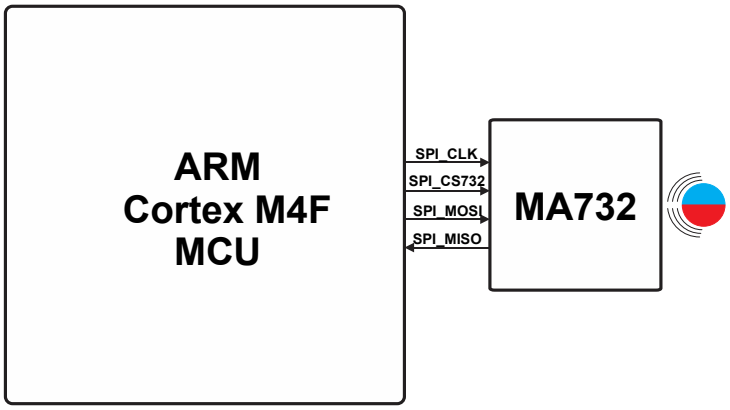


Figure 1: MA732 Connection to the MCU

Power Stage Selection

Driving bipolar steppers require two full H-bridges. The two main implementations to drive bipolar stepper motors are using a dual H-bridge power stage with a microcontroller unit (MCU) to generate sine/cosine wave pairs, or using a fully integrated step indexer engine with microstepping support. Using an MCU and dual H-bridge combination provides more flexibility in terms of how to regulate the sine wave currents, but it also increases complexity. For this article, a fully integrated step indexer with as much as 16 degrees of microstepping was selected (see Figure 2). The integrated step indexer in this article is the MP6602, which provides up to 4A of current drive and is capable of driving NEMA 17 and NEMA 23 bipolar stepper motors. Meanwhile, the MCU drives all control signals, communicates with the indexer through the SPI port, and samples fault information.

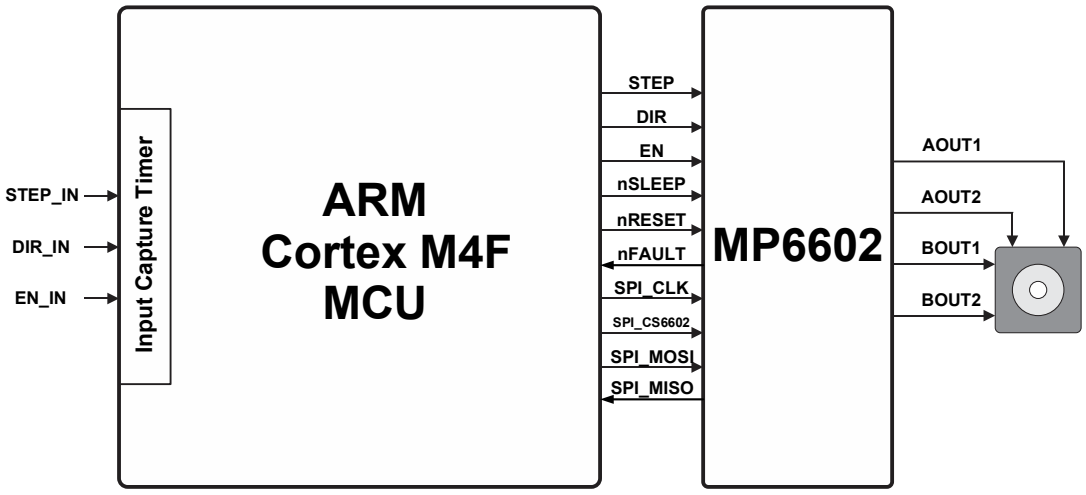
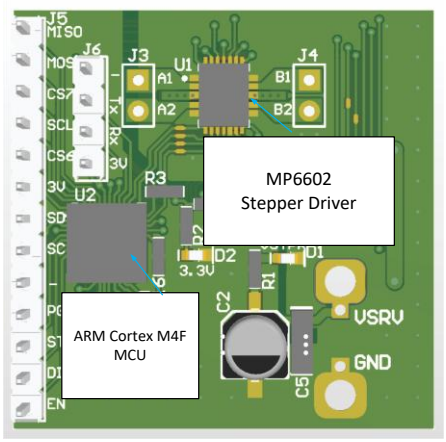


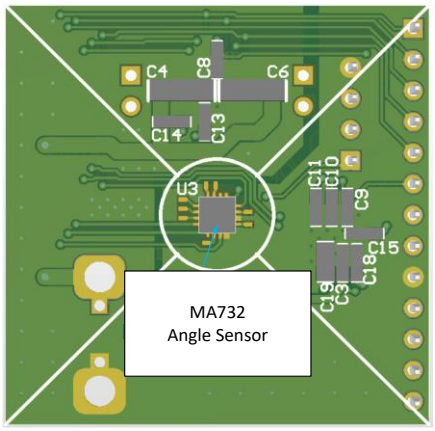
Figure 2: Bipolar Stepper Indexer Connections

Final Implementation

For a closed-loop stepper implementation, the sensor and power stage should be controlled by an off-the-shelf ARM Cortex M4F microcontroller (MCU). The MCU communicates with both devices through a single SPI port with two chip selects. An internal timer generates the steps. The board measures 1.35"x1.35", and is small enough to fit behind a NEMA 17 stepper motor (see Figure 3). This allows the reference design to be used in a larger motor frame size, such as the NEMA 23.



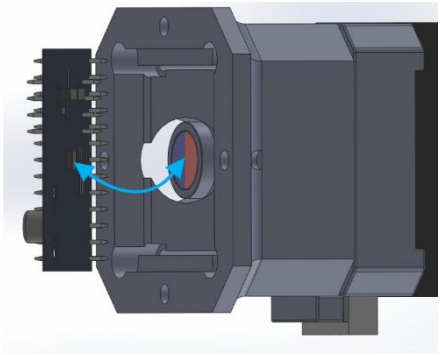
a) Top Side



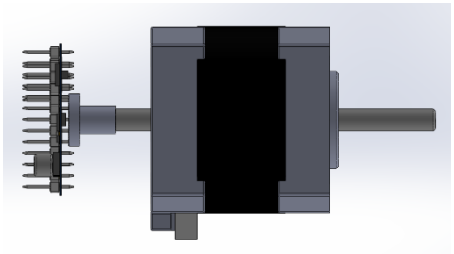
b) Bottom Side

Figure 3: PCB Layouts

Figure 4 shows the motor assembly, in which Figure 4a shows the motor assembly with a diametrically magnetized round magnet facing MA732 sensor, and Figure 4b shows the final solution.



a) Motor Assembly



b) Final Solution (Housing Invisible)

Figure 4: Motor Assembly

Absolute Position and Sensor Overflow

Although the contactless magnetic based sensor is considered to be an absolute position encoder, this is only true on a per-revolution basis. That is, throughout the rotor's angular travel through each revolution, the sensor provides a 16-bit number that the MCU reads, which essentially allows the firmware to learn the rotor's absolute position at any given time.

As the motor revolves, however, each new revolution is indistinguishable from the previous revolution. We can add angular position readings into a much larger number, which can be expressed as a variable that takes all of the angle readings to obtain the entire position as an absolute value (called Rotor_Angle_Absolute). This variable is a 32-bit signed integer. If the motor moves forward, increment the variable, and vice versa. Assuming 16-bit readings, 1600 microsteps per revolution, and a 1000rpm step rate, it would take 22.37 hours for the variable to overflow. The MCU must ensure that the sensor readings are added correctly, even as the rotor goes through its overflow region. This absolute position correction must be executed whether the motor is rotating clockwise or counterclockwise (i.e. the sensor position is incrementing or decrementing).

Figure 5 shows how the angle position changes over time.

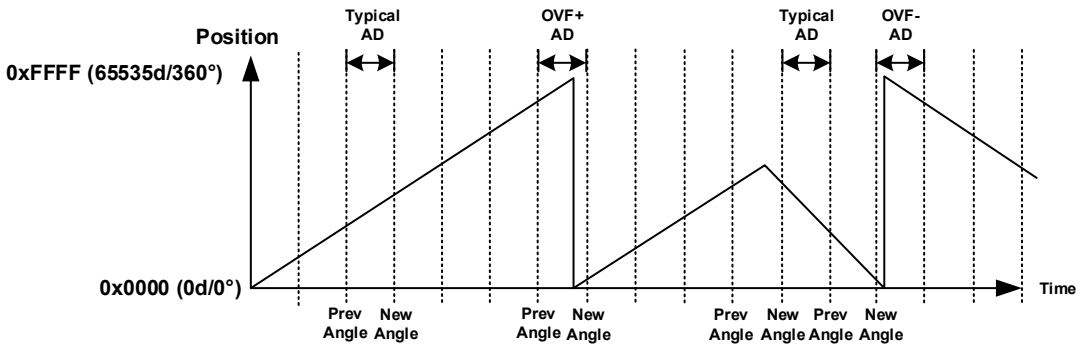


Figure 5: Angle Position Changes Over Time

Figure 5 shows that the the angular displacement (MA732_Angle_Delta, denoted as AD in Figure 5) is computed at periodic intervals (1ms). During each sample, the previous read is stored within MA732_Angle_Prev (denoted as Prev Angle in Figure 5), the new sample is stored at MA732_Angle_New (denoted as New Angle on Figure 5). MA732_Angle_Delta can be calculated with Equation (1):

$$MA732\_Angle\_Delta = (MA732\_Angle\_New - MA732\_Angle\_Prev) \quad (1)$$

The result of Equation (1) is added to MA732_Angle_Absolute. If the rotor moved clockwise (forward), the displacement is positive; if the motor moves counter clockwise (reverse), the displacement is negative.

A special consideration must be made during angle sensor overflows. If the sensor moves forward past the maximum of 0xFFFF (denoted as OvF+AD on Figure 5), or if the sensor decrements its position past 0x0000 (denoted as OvF-AD on Figure 5), the previous equation can no longer be used. In both of these scenarios, the FW logic chooses one of the following equations, depending on which case we are servicing. If the angle displacement overflows when counting up and exceeds the maximum (OvF+AD), then MA732_Angle_Delta can be calculated with Equation (2):

$$MA732\_Angle\_Delta = 65535 + MA732\_Angle\_Delta \quad (2)$$

If the angle displacement overflows when counting down and falls below the minimum (OvF-AD), then MA732_Angle_Delta can be calculated with Equation (3):

$$MA732\_Angle\_Delta = -(65535 - MA732\_Angle\_Delta) \quad (3)$$

Position Control Algorithm

Once the absolute position derived from the sensor samples is obtained, we can close the loop on position by comparing the current position to a position command (Position_Command). In this implementation, a simple serial communication protocol is used to send the position command through an external controller and into the motor's MCU. For example, consider the stepper motor in this article, which is tasked with opening and closing a flap. From a current position that reads as 0, the motor can be commanded to move to 655,530, which represents 10 turns of clockwise motion. Every time the sensor is sampled and the absolute position obtained, the absolute position command error (POS_Command_Error) can be calculated with Equation (4):

$$POS\_Command\_Error = POS\_Command - MA732\_Angle\_Absolute \quad (4)$$

If the error is positive, the MP6602 DIR input is made high. If the error is negative, the MP6602 DIR input is made low. At this time, POS_Command_Error is converted into an absolute number (ABS_Error). It is now a matter of employing a timer to generate the step commands, which decreases the error until it approaches 0. This timer's frequency is controlled via a proportional gain, such that the step frequency is larger with large errors and grows smaller as the error decreases. Once the error approaches zero, the motor stops and actuation ends.

To avoid unwanted vibrations while attempting to preserve the commanded position, a dead band of about 3 microsteps was put into place. If the error needs to be even closer to zero, a multiple-order proportional gain could be implemented. For example, a larger gain could be employed to articulate the motor while the error is big, until the error is close to 3 microsteps. From then on, a much smaller gain can be used to bring the motor home as close to within ±1 microstep.

Conclusion

Using an off-the-shelf microcontroller, we can interface the stepper motor driver and Hall-sensor based sensor via an SPI port. The firmware can then continuously interrogate the position sensor and extrapolate the motor rotor position at all times. By comparing this position to a commanded position, the motor can be commutated to reach the commanded position in a timely fashion. If an external force causes the motor to lose steps, the sensor information tracks how many steps were lost, which then allows the microcontroller to close the loop on position and successfully bring the stepper motor to the commanded position.

Although stepper motors are mostly used in open-loop applications, there are plenty of advantages in closing the loop on position. By employing cost-effective, Hall-sensing technologies, such as the MA732, and an easy-to-use index based stepper drivers, such as the MP6602, the application can now add servo-like properties to their stepper-based applications. MPS provides a wide array of position sensors and stepper drivers to suit each specific design's needs.