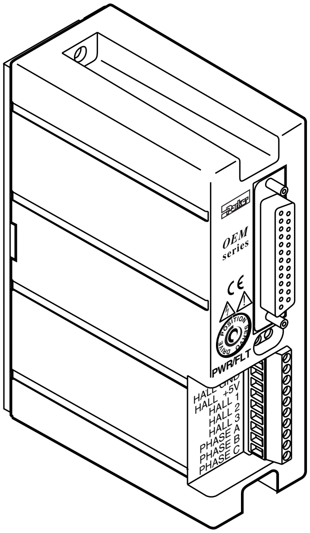
Compumotor

# OEM770X
# Servo Controller/Drive
# User Guide



Compumotor Division
Parker Hannifin Corporation
p/n 88-018468-01 A

# IMPORTANT

# User Information

## ⚠ WARNING ⚠

OEM Series products are used to control electrical and mechanical components of motion control systems. You should test your motion system for safety under all potential conditions. Failure to do so can result in damage to equipment and/or serious injury to personnel.

OEM Series products and the information in this user guide are the proprietary property of Parker Hannifin Corporation or its licensers, and may not be copied, disclosed, or used for any purpose not expressly authorized by the owner thereof.

Since Parker Hannifin constantly strives to improve all of its products, we reserve the right to change this user guide and software and hardware mentioned therein at any time without notice.

In no event will the provider of the equipment be liable for any incidental, consequential, or special damages of any kind or nature whatsoever, including but not limited to lost profits arising from or in any way connected with the use of the equipment or this user guide.

**Technical Assistance** ⇨ *Contact your local automation technology center (ATC) or distributor,* ***or ...***

**North America and Asia:**
Compumotor,
Division of Parker Hannifin
5500 Business Park Drive
Rohnert Park, CA 94928

Telephone: (800) 358-9070
or (707) 584-7558
Fax: (707) 584-3793
FaxBack: (800) 936-6939
or (707) 586-8586
e-mail: tech_help@cmotor.com
Internet: http://www.compumotor.com

**Europe** *(non-German speaking)***:**
Parker Digiplan
21 Balena Close
Poole, Dorset
England BH17 7DX
Telephone: +44 (0)1202 69 9000
Fax: +44 (0)1202 69 5750

**Germany, Austria, Switzerland:**
HAUSER Elektronik GmbH
Postfach: 77607-1720
Robert-Bosch-Str. 22
D-77656 Offenburg
Telephone: +49 (0)781 509-0
Fax: +49 (0)781 509-176

Parker
Automation

**Technical Support Email**
**tech_help@cmotor.com**

**Product Type:**
**OEM770X Position Servo Controller/Drive**

The above product is in compliance with the requirements of directives

- **72/23/EEC Low Voltage Directive**
- **93/68/EEC CE Marking Directive**

The OEM770X, when installed according to the procedures in the main body of this user guide, may not necessarily comply with the Low Voltage Directive (LVD) of the European Community. To install the OEM770X so that it complies with LVD, you must follow the additional procedures described in *Appendix A, LVD Installation Instructions*. If you do not follow these instructions, the LVD protection of the product may be impaired.

OEM770X drives are sold as complex components to professional assemblers. As components, they are not required to be compliant with Electromagnetic Compatibility Directive 89/336/EEC. However, information is offered in Compumotor's *EMC Installation Guide* on how to install the OEM770X in a manner most likely to minimize the effects of drive emissions and to maximize the immunity of drives from externally generated interference.

**Compumotor Division**

# CONTENTS

# PREFACE

## About This User Guide

You may not need to read this user guide from cover to cover! You can find essential information in the first three chapters—a product description in Chapter 1, installation instructions in Chapter 2, and specifications for the drive and motors in Chapter 3. This may be all you need to use the OEM770X.

Later chapters contain additional information about selected topics. Read them if you need a deeper understanding about these topics.

Special internal circuits, including an extended discussion of the current foldback circuit and the response circuit, are covered in Chapter 4. This chapter may interest you if you want to achieve optimum performance from the drive by adjusting the selectable resistors.

Hall effect sensors, and the way they affect commutation in brushless servo motors, are described in Chapter 5. If you use motors from manufacturers other than Compumotor, you may need this information to determine how to connect your motor to the drive.

Power supply selection is covered in Chapter 6. Read this chapter for information about calculating the power your system requires, how regeneration affects power supplies, and how you can specify a power supply for your system.

Troubleshooting procedures are covered in Chapter 7.

## Warnings and Cautions

Warning and caution notes alert you to problems that may occur if you do not follow the instructions correctly. Situations that may cause bodily injury are presented as warnings. Situations that may cause system damage are presented as cautions.

A typical warning note is shown below.

***WARNING***

Do not touch the motor immediately after it has been in use for an extended period of time. The motor may be hot.

A typical caution note is shown below.

***CAUTION***

Do not turn on power unless the motor's Hall effect sensors, Hall +5, and Hall GND are connected to the drive. The motor may be destroyed by overheating if these connections are not made.

# CHAPTER 1

## Introduction

### OEM770X Description

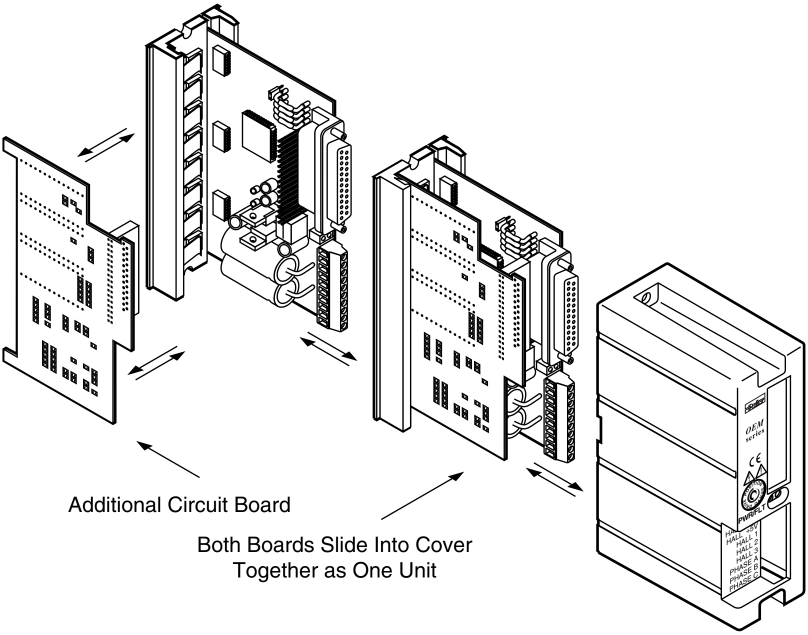
The OEM770X is a torque servo drive designed to operate standard 3 phase brushless DC servo motors equipped with Hall effect sensors, or equivalent feedback signals. It can also operate brushed DC servo motors. It is a high-performance module around which the Original Equipment Manufacturer (OEM) can design a motion control system. The drive offers a basic set of features designed to meet the needs of most customers. It is compatible with standard industry servo controllers, and is intended to be used in positioning applications. It uses three-state current control for efficient drive performance and cooler motor operation.

The OEM770X is small and convenient to use. It installs with only two screws (the screws also provide grounding and captivate the cover). Its right angle screw terminal allows side-by-side mounting, and its small footprint maximizes cabinet space. The snap-on molded cover is removable for drive configuration, and helps provide a barrier against environmental contamination. The drive is the same size as a 3U Eurorack card. Its standard 25 pin D-connector is compatible with universally available connectors.

The drive is designed for manufacturability and reliability. It uses surface mount components and a custom designed ASIC to conserve space, reduce cost, and improve reliability. More than 90% of the components are auto inserted, which reduces assembly time and cost, and further improves reliability.

### Operation & Block Diagram

A Compumotor product called the OEM770T Torque Drive is the “building block” in a family of servo drives. It has an internal slot where an additional circuit board can be inserted to make a different product.



Additional Circuit Board

Both Boards Slide Into Cover
Together as One Unit

*Additional Circuit Board Can Mount Internally*

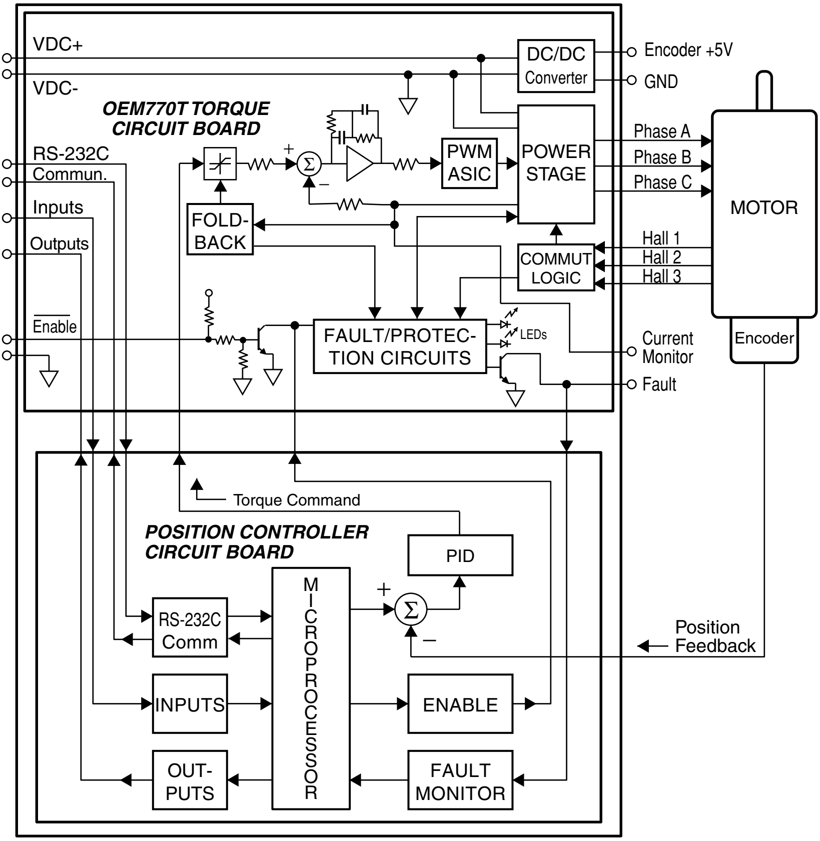
The additional circuit board is inserted at the factory, at the time of manufacture. Externally, the new product looks just like the OEM770T, except that the label is a different color.

## OEM770X Position Controller/Drive

The OEM770X Controller/Drive consists of the OEM770T with a position controller circuit board.

The OEM770X requires a single external power supply. The drive accepts 24VDC to 75VDC for its power input. Its internal DC-to-DC converter produces +5V to power Hall effect sensors and encoder electronics, and all internal voltages used for the drive’s circuits.

The block diagram for the OEM770X is shown in the next drawing.

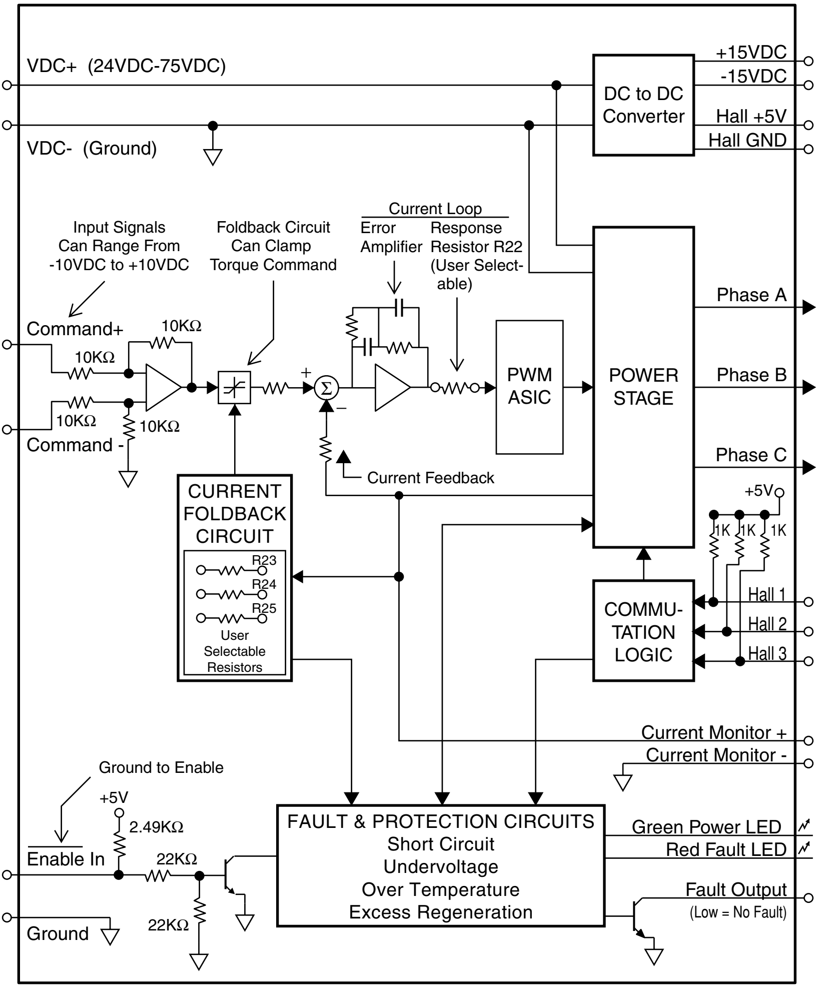


*OEM770X Position Controller/Drive — Block Diagram*

Inputs, outputs, and RS-232C communications are internally routed to the position controller board, where they interface with a microprocessor. The microprocessor generates a position command. It can also enable or disable the torque board.

The position controller board receives feedback about actual position from an encoder, and compares commanded position with actual position. It generates a torque command to correct any position errors. The torque command (which is an analog voltage) then goes to the torque board as a *command input* signal, passes through the foldback circuit, and proceeds through the remainder of the torque board’s circuits.

The detailed block diagram for the torque board is shown in the next drawing.



*Block Diagram — OEM770T Torque Servo Drive*

Input to the torque board is a voltage signal called *command input*. It can range from -10VDC to +10VDC. Output current is scaled so that each volt of command input corresponds to 1.2A of output current. For example, a command input of 5V results in a 6A output current. The maximum command input of 10V results in the full 12A output current.

When the command input signal enters the board, it is amplified, sent through a foldback circuit (which may or may not be active) and an inverter, and summed with a current feedback signal that is proportional to the actual output current.

An error signal—the difference between commanded and actual output current—goes through an error amplifier. The amplifier's output controls a pulse width modulation (PWM) circuit. If actual current is too low, the PWM circuit will send longer pulses to the power stage. These pulses keep the stage turned on longer, which results in more output current. If actual current is too high, the PWM circuit sends shorter pulses, resulting in less current.

A *response resistor* affects the signal level that goes into the PWM circuit. The user can choose a value for this resistor that produces the best current loop gain and system dynamics for a particular motor.

The power stage has three outputs—each connects to a particular motor coil. The drive gets inputs from the motor's Hall effect sensors, and determines which of six possible positions the rotor is in. It then uses a six-state commutation technique to send current into one coil and out of another (the third coil receives no current). The current creates a torque on the rotor, and the rotor turns to the next position. The drive reads the new position from the Hall sensors, and switches current to a different combination of coils. The rotor turns further, and the process repeats. (The drive can also be configured to commutate brushed servo motors.)

The drive has several fault and protection circuits. These monitor temperature, regeneration, undervoltage, and short circuits. They can shut down the drive if limits are exceeded. LEDs indicate power and fault status.

A foldback circuit monitors motor current, and protects the motor from overheating due to prolonged high currents. The user can install resistors to set levels for peak current, foldback current, and time constant. When the circuit invokes foldback, it clamps the command input signal at a voltage that reduces motor current to the preset level. After a period of time, the circuit may release its clamp on the command input signal, and normal operations can continue.

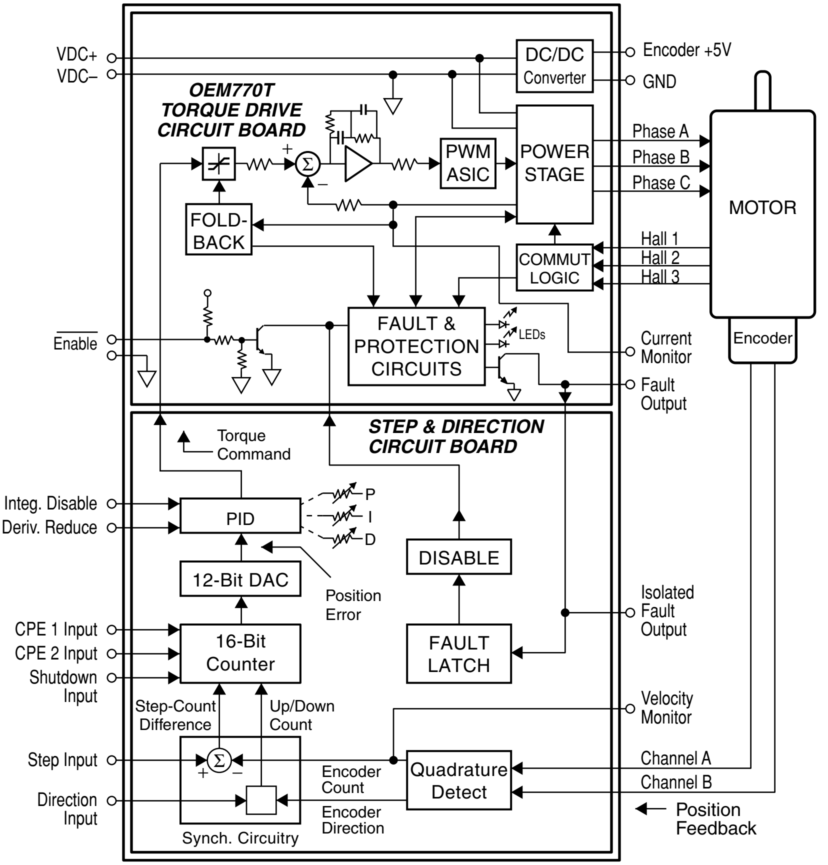
The drive has several other inputs and outputs. An enable input must be grounded to enable the drive. A fault output is held low if there are no faults. A current monitor output provides a voltage scaled to represent the actual output current. It can range from -10V to +10V, with one volt corresponding to 1.2 amps of output current.

## Related Products

In addition to the OEM770X and OEM770T, the other products described below comprise the family of Compumotor OEM Servo Products.

### OEM770SD Step & Direction Servo Drive

The OEM770SD Step & Direction Servo Drive consists of the OEM770T with a position controller circuit board added.



*Block Diagram – OEM770SD Step & Direction Servo Drive*

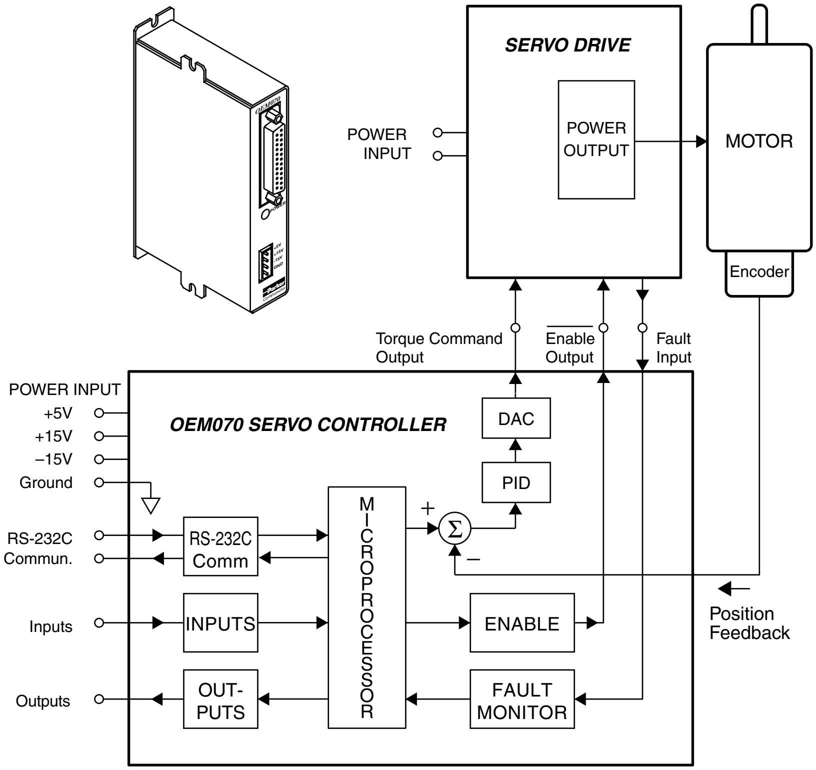
The controller accepts step and direction position commands from an indexer. It uses encoder signals for feedback. Its internal PID position control loop generates an analog command output voltage that is sent to the torque board.

Indexers intended for use with step motor systems can operate the OEM770SD. It emulates a stepper drive, but can achieve servo system levels of high speed performance and thermal efficiency.

## OEM070 Servo Controller

The OEM070 Servo Controller is a compact, stand-alone controller designed to operate with analog servo drives.

The OEM070 contains the same position controller board used in the OEM770X. The board is packaged by itself in a minimum depth, small footprint housing. It controls motor torque or velocity with a ±10V command output signal. Through its I/O and RS-232C ports, the OEM070 can interface with external devices such as incremental encoders, switches, computers, and programmable control units.



*OEM070 Servo Controller – Block Diagram*

## SM and NeoMetric Series Servo Motors

Compumotor offers SM Series and NeoMetric Series servo motors designed to operate with OEM Series servo drives. Each motor is equipped with Hall effect outputs and an encoder.

# CHAPTER 2

## Installation

Complete the following installation steps before you use the OEM770X drive.

**Installation Steps:**

1. Verify shipment is correct.
2. Install selectable resistors.
3. Mount the drive.
4. Mount the motor.
5. Connect the motor to the drive.
6. Connect inputs, outputs, and controller.
7. Connect a power supply to the drive.
8. Tune the drive.

The sections in this chapter give basic instructions about how to complete each of these steps.

## OEM770X Ship Kit

Inspect the OEM770X upon receipt for obvious damage to its shipping container. Report any damage to the shipping company. Parker Compumotor cannot be held responsible for damage incurred in shipment. You should receive one or more drives, depending upon what you ordered. Compare your order with the units shipped.

| Component | Part Number |
|---|---|
| OEM770X Drive | OEM770X |
| Resistor Kit | 73-018496-01 |
| Accessories | |
| OEM770X User Guide | 88-018468-01 |
| Heatsink | OEM -HS1 |

User guides are not sent with each product. They are available upon request. Please order user guides as needed.

The following SM and NeoMetric Series servo motors are designed to be used with the OEM770X. Compare your order with the motors shipped.

| Motor Size | Part Number |
|---|---|
| Size 16 | SM160A, SM160B, SM161A, SM161B<br>SM162A, SM162B |
| Size 23 | SM230A, SM230B, SM231A, SM231B,<br>SM232A, SM232B, SM233A, SM233B |
| Size 34 | NO341D, NO341F, NO342E, NO342F<br>JO341D, JO341F, JO342E, JO342F |
| Size 70mm | NO701D, NO701F, NO702E, NO702F<br>JO701D, JO701F, JO702E, JO702F |

## Installing Selectable Resistors and Jumper JU1

You must install four resistors into sockets on the OEM770X's circuit board. Three of these are *foldback resistors*; they determine the parameters for the current foldback circuit, which can protect your motor from overheating due to prolonged high currents. The fourth resistor is a *response resistor*—it affects the gain and frequency response of the current loop.

The OEM770X ships with resistors and jumper installed. These resistors are not appropriate for most applications. You *must* select other resistors and install them in the drive.

You can also install jumper JU1, located near the resistors, to adjust drive performance for your particular motor.

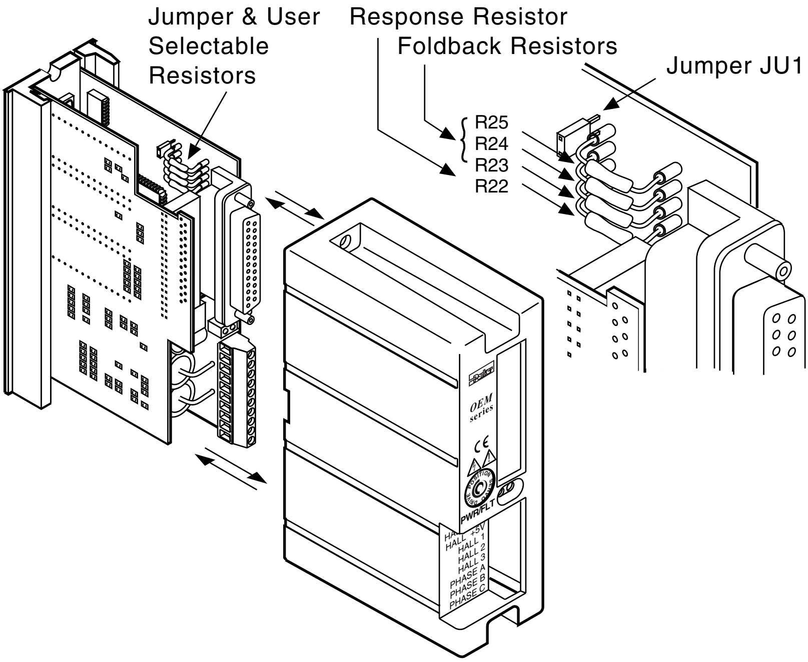
A resistor kit for use with Compumotor SM and NeoMetric Series motors is included with the drive. If the resistors are color coded, a key to the code is included in the kit. If the resistors have a numerical code, the first three digits are resistance values; the fourth digit is a multiplier.

Example: 3013 = 301 x $10^3$ = 301KΩ
6492 = 649 x $10^2$ = 64.9 KΩ
Note: zero ohm resistors may be color coded (black band)

To install resistors or the jumper, remove the drive's molded plastic cover. Apply pressure to the D-connector while you hold the cover's sides. The circuit board will slide out. The resistors and jumper are located at the corner of the board, near the 25 pin D-connector, as shown in the next drawing.

***WARNING***

Remove power from the OEM770X before installing selectable resistors.



*Selectable Resistor and Jumper Locations*

Remove any resistors that are in the sockets, and install those that you have selected. The next table shows recommended resistors for Compumotor SM and NeoMetric Series motors. For full details on further customizing the response and foldback circuits, or choosing resistors for non-Compumotor motors, see *Chapter 4 Special Internal Circuits*.

The next table also shows jumper position—installed or removed—for Compumotor motors.

## Resistor & Jumper Selection for Compumotor Motors

Use the table below to select resistors and jumper position for Compumotor motors. (The next section shows default values.)

### OEM770X – Resistor and Jumper Settings for Motors at 75VDC*

| Motor | R22 | R23 | R24 | R24 | R25 | Jumper Installed |
|---|---|---|---|---|---|---|
| | ($R_{response}$) | ($T_{c-therm}$) | ($I_{pk-tune}$) | ($I_{pk-final}$) | ($I_{fold}$) | |
| SM160A | 249 KΩ | 5.1 MΩ | 348 KΩ (5 A) | 150 KΩ (7.5 A) | 1.2 MΩ (2.2 A) | no |
| SM160B | 750 KΩ | 10 MΩ | 64.9 KΩ (10 A) | 0 Ω (12 A) | 124 KΩ (3.0 A) | no |
| SM161A | 301 KΩ | 5.1 MΩ | 450 KΩ (4 A) | 249 KΩ (6 A) | 1.2 MΩ (2.2 A) | no |
| SM161B | 750 KΩ | 10 MΩ | 124 KΩ (8 A) | 0 Ω (12 A) | 124 KΩ (3.0 A) | no |
| SM162A | 205 KΩ | 5.1 MΩ | 450 KΩ (4 A) | 249 KΩ (6 A) | 1.2 MΩ (2.2 A) | no |
| SM162B | 402 KΩ | 10 MΩ | 124 KΩ (8 A) | 0 Ω (12 A) | 124 KΩ (3.0 A) | no |
| SM230A | 402 KΩ | 5.1 MΩ | 348 KΩ (5 A) | 150 KΩ (7.5 A) | 1.2 MΩ (2.2 A) | no |
| SM230B | 1 MΩ | 10 MΩ | 64.9 KΩ (10 A) | 0 Ω (12 A) | 124 KΩ (3.0 A) | no |
| SM231A | 402 KΩ | 5.1 MΩ | 450 KΩ (4 A) | 249 KΩ (6 A) | 1.2 MΩ (2.2 A) | no |
| SM231B | 604 KΩ | 10 MΩ | 124 KΩ (8 A) | 0 Ω (12 A) | 124 KΩ (3.0 A) | no |
| SM232A | 205 KΩ | 5.1 MΩ | 450 KΩ (4 A) | 249 KΩ (6 A) | 1.2 MΩ (2.2 A) | no |
| SM232B | 500 KΩ | 10 MΩ | 124 KΩ (8 A) | 0 Ω (12 A) | 124 KΩ (3.0 A) | no |
| SM233A | 30.1 KΩ | 5.1 MΩ | 450 KΩ (4 A) | 249 KΩ (6 A) | 1.2 MΩ (2.2 A) | yes |
| SM233B | 700 KΩ | 10 MΩ | 124 KΩ (8 A) | 0 Ω (12 A) | 124 KΩ (3.0 A) | no |
| NO701D/NO341D | 205 KΩ | 10 MΩ | 249 KΩ (6 A) | 90.9 KΩ (9 A) | 1.2 MΩ (2.2 A) | yes |
| NO701F/NO341F | 750 KΩ | 10 MΩ | 90.9 KΩ (9 A) | 0 Ω (12 A) | 124 KΩ (3.0 A) | yes |
| NO702E/NO342E | 750 KΩ | 10 MΩ | 182 KΩ (7 A) | 64.9 KΩ (10 A) | 1.2 MΩ (2.2 A) | yes |
| NO702F/NO342F | 604 KΩ | 10 MΩ | 90.9 KΩ (9 A) | 0 Ω (12 A) | 124 KΩ (3.0 A) | yes |

* For supply voltages less than 75VDC, calculate R22 using the following equation: $R22_{new}$ = ($R22_{old}$ • $V_{bus}$)/75, where $R22_{old}$ is the value from the table above (at 75VDC). R23, R24, R25 remain the same as for 75VDC.

### *R24 – "pk-tune" and "pk-final"*

Note that there are two values recommended for R24. Use the first value (*pk-tune*) when you begin your tuning procedure. This keeps peak currents low, to avoid the damaging currents that instability during tuning can cause. As you refine your tuning settings, replace R24 with the second value (*pk-final*), if your application requires more torque.

## Resistor & Jumper Selection for Non-Compumotor Motors

The following sections describe how to choose resistor values and jumper position for other motors.

### *Selecting Foldback Resistors*

The OEM770X ships with resistors already installed.

**Default Foldback Resistors (as shipped)**

| Res. #: | Function | Value |
|---|---|---|
| R25 | Foldback Current | 23.7 KΩ (6A) |
| R24 | Peak Current | Ø Ω (12A) |
| R23 | Time Constant | 5.1 MΩ |

The default values above may not be suitable for your application. If your system cannot withstand the peak torque, you should determine foldback resistor values appropriate to your application and install them in your drive.

For full details about how to choose foldback resistor values, and about how the foldback circuit works, see *Chapter 4 Special Internal Circuits*.

### *Selecting a Response Resistor*

The OEM770X ships with a response resistor already installed.

**Default Response Resistor (as shipped)**

| Res. #: | Function | Value |
|---|---|---|
| R22 | Optimize gain and frequency response | 100 KΩ |

If your motor is not well matched to the default resistor, your system might not perform as well as you expect. In this case, improve your system's performance by selecting an appropriate response resistor, and installing it in the drive.

For full details about how to choose a value for the response resistor, and about how the circuit works, see *Chapter 4 Special Internal Circuits*.

### *Selecting Jumper Position for Non-Compumotor Motors*

You can adjust the performance of the OEM770X's internal error amplifier by installing or removing jumper JU1. The drive ships with the jumper installed.

For motors with long electrical time constants (L/R), such as Compumotor's NeoMetric motors, install the jumper. Remove the jumper for motors with short time constants, such as Compumotor's SM motors (except SM233A).

**Jumper Position Selection Procedure**

**1. Adjust R22 with Jumper JU1 Installed**

Starting with a high value, adjust R22 for optimum system response. For adjustment instructions, see *Selecting a Response Resistor* in *Chapter 4 Special Internal Circuits.*

**2. If Unable to Obtain an Optimum Response:**

*Chapter 4 Special Internal Circuits* describes optimum responses. If you could not obtain an optimum response in *Step 1*—your adjustments produced overdamped or underdamped responses, with no range of optimum responses in between—then:

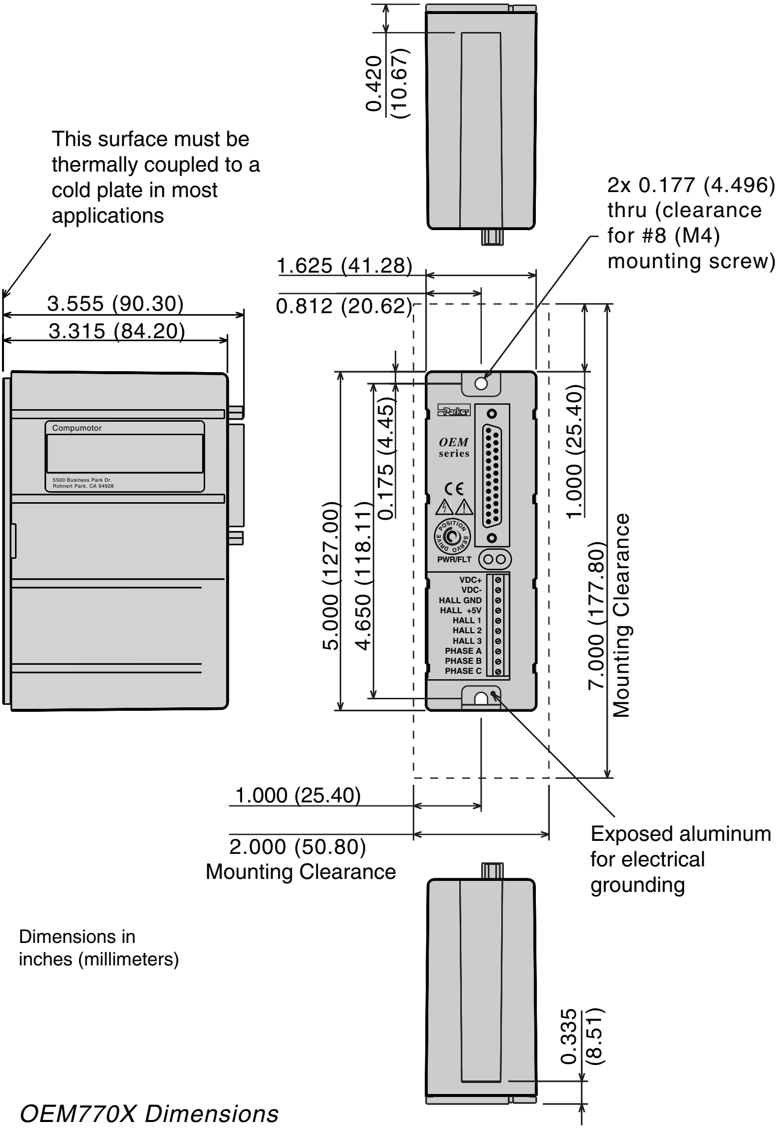
- Replace R22 with a high value, to limit oscillations during *Step 3* below.
- Remove Jumper JU1.

**3. Adjust R22 with Jumper JU1 Removed**

With Jumper JU1 removed, adjust R22 to achieve an optimum system response.

For further help, provide your motor's inductance (*L*) and resistance (*R*) values to Compumotor's Applications Department. We can calculate a recommended jumper position and R22 value, based on your motor's values.

# Drive Mounting



*OEM770X Dimensions*

## DRIVE DIMENSIONS

The OEM770X is designed to minimize panel area, or footprint, in an equipment cabinet. Dimensions are shown in the drawing. You can mount the drive in a "minimum depth" configuration if you use an optional heatsink. (See below.)

## Panel Layout

Move profiles and loads affect the amount of heat dissipated by the OEM770X. Applications with low average power (less than 3 Amps continuous motor current) and mild ambient temperatures may not require a heatsink.

The OEM770X is designed to operate within the following temperature guidelines:

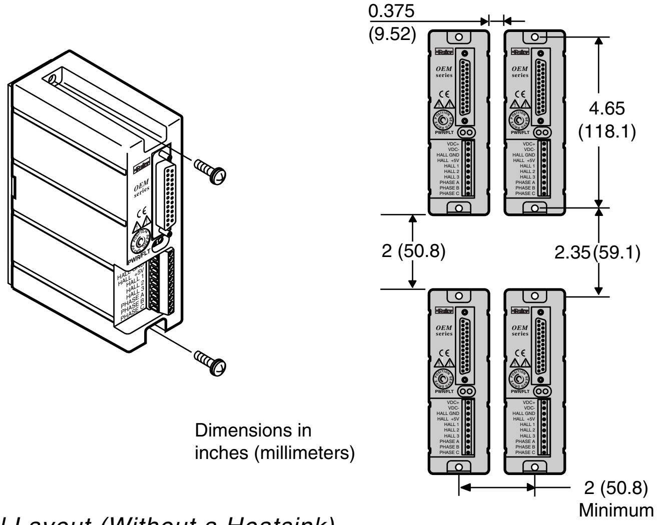
| | |
|---|---|
| • Maximum Ambient Temperature: | 45°C (113°F) |
| • Maximum Heatsink Temperature | 45°C (113°F) |

For applications with higher power or elevated ambient temperatures, you may need to mount the drive in a way that removes heat from it. The drive uses a heatplate design as a pathway to dissipate its excess heat; it should be mounted to a heatsink or a suitable heat sinking surface.

The OEM770X is overtemperature protected. (See *Chapter 4 Special Internal Circuits* for more information.)

### *Mounting Without a Heatsink*

The next drawing shows the recommended panel layout for mounting the OEM770X without a heatsink.



Dimensions in inches (millimeters)

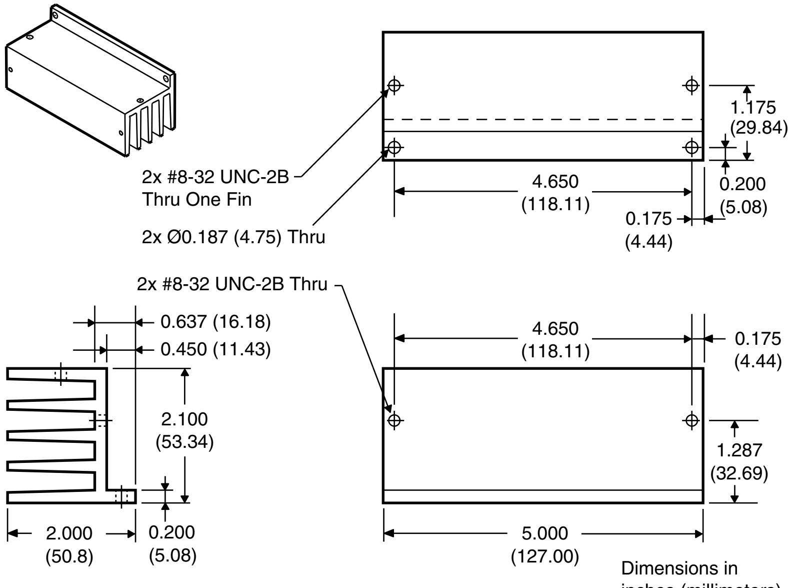
*Panel Layout (Without a Heatsink)*

### *Mounting With Compumotor Heatsink OEM-HS1*

A heatsink designed to work with the OEM770X can be purchased from Compumotor (Part Number OEM-HS1). This heatsink is sufficient for most applications operating in 45°C (113°F) or lower ambient temperatures.

The drive may be mounted in two different configurations. One configuration uses a minimum amount of mounting area (*minimum area*). The other configuration uses a minimum amount of mounting depth (*minimum depth*).

Heatsink dimensions are shown in the next drawing.



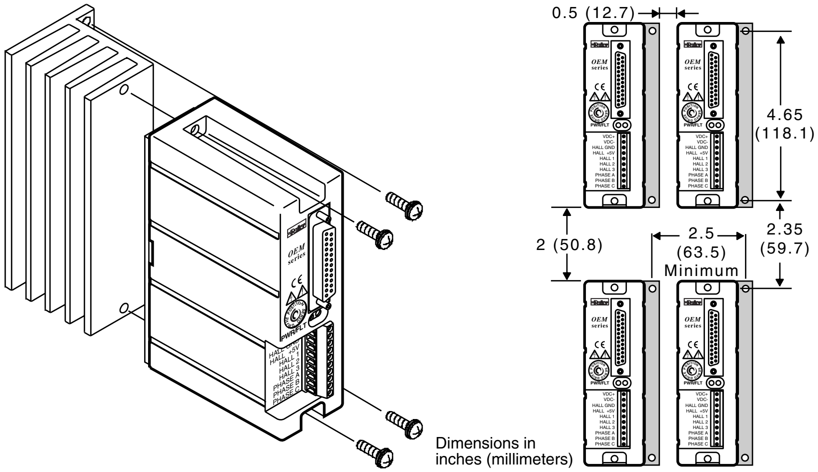
*OEM-HS1 Heatsink Dimensions*

Two #8-32 screws are needed to mount the OEM770X to the OEM-HS1 heatsink. Use a *star washer* on the bottom screw to ensure proper electrical grounding. Use two #8 screws to mount the OEM-HS1 to the cabinet.

Do not use a star washer between the back of the OEM770X heatplate and the mounting surface. The mounting surface must be flat. Use silicone thermal joint compound or thermal pads to facilitate heat transfer from the drive's heatplate to your mounting surface.

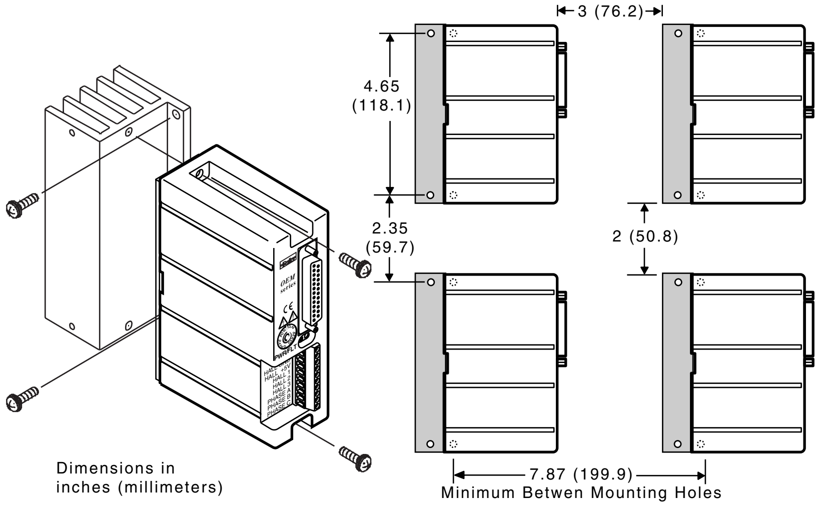
A heatsink with holes tapped for metric screws is available. Its part number is OEM-HS1-M4. See your Compumotor sales guide for more information.

The next drawing shows the panel layout for minimum area.



*OEM-HS1 Minimum Area Panel Layout*

The following drawing shows dimensions for a minimum depth panel layout.



*OEM-HS1 Minimum Depth Panel Layout*

## Motor Mounting

The following guidelines present important points about motor mounting and its effect on performance.

For mechanical drawings of SM and NeoMetric Series servo motors, see *Chapter ¬ Specifications*

**WARNING**
Improper motor mounting can reduce system performance and jeopardize personal safety.

Servo motors used with the OEM770X can produce large torques and high accelerations. This combination can shear shafts and mounting hardware if the mounting is not adequate. High accelerations can produce shocks and vibrations that require much heavier hardware than would be expected for static loads of the same magnitude.

The motor, under certain move profiles, can produce low-frequency vibrations in the mounting structure. These vibrations can cause metal fatigue in structural members if harmonic resonances are induced by the move profiles you are using. A mechanical engineer should check the machine design to ensure that the mounting structure is adequate.

**CAUTION**
Consult a Compumotor Applications Engineer (800-358-9070) before you machine the motor shaft. Improper shaft machining can destroy the motor's bearings. Never disassemble the motor.

Servo motors should be mounted by bolting the motor's face flange to a suitable support. Foot mount or cradle configurations are not recommended because the motor's torque is not evenly distributed around the motor case. Any radial load on the motor shaft is multiplied by a much longer lever arm when a foot mount is used rather than a face flange.

### Motor Heatsinking

Performance of a servo motor is limited by the amount of current that can flow in the motor's coils without causing the motor to overheat. Most of the heat in a brushless servo motor is dissipated in the stator—the outer shell of the motor. Performance specifications usually state the maximum allowable case temperature. Exceeding this temperature can permanently damage the motor.

If yours is a demanding application, your motor may become quite hot. The primary pathway through which you can remove the heat is through the motor's mounting flange. Therefore, mount the motor with its flange in contact with a suitable heatsink.

Specifications for Compumotor SM and NeoMetric Series servo motors apply when the motor is mounted to a ten inch by ten inch aluminum mounting plate, 1/4 inch thick. To get rated performance in your application, you must mount the motor to a heatsink of at least the same thermal capability. Mounting the motor to a smaller heatsink may result in decreased performance and a shorter service life. Conversely, mounting the motor to a larger heatsink can result in enhanced performance.

## ATTACHING THE LOAD

Your mechanical system should be as stiff as possible. Because of the high torques and accelerations of servo systems, the ideal coupling between a motor and load would be completely rigid. Rigid couplings require perfect alignment, however, which can be difficult or impossible to achieve. In real systems, some misalignment is inevitable. Therefore, a certain amount of flexibility may be required in the system. Too much flexibility can cause resonance problems, however.

These conflicting requirements are summarized below.

- Maximum Stiffness (in the mechanical system)
- Flexibility (to accommodate misalignments)
- Minimum Resonance (to avoid oscillations)

The best design solution may be a compromise between these requirements.

## MISALIGNMENT & COUPLERS

The type of misalignment in your system will affect your choice of coupler.

***Parallel Misalignment***
The offset of two mating shaft center lines, although the center lines remain parallel to each other.

***Angular Misalignment***
When two shaft center lines intersect at an angle other than zero degrees.

***End Float***
A change in the relative distance between the ends of two shafts.

There are three types of shaft couplings: single-flex, double-flex, and rigid. Like a hinge, a single-flex coupling accepts angular misalignment only. A double-flex coupling accepts both angular and parallel misalignments. Both single-flex and double-flex, depending on their design, may or may not accept endplay. A rigid coupling cannot compensate for any misalignment.

***Single-Flex Coupling***

When a single-flex coupling is used, one and only one of the shafts must be free to move in the radial direction without constraint. ***Do not use a double-flex coupling in this situation:*** it will allow too much freedom and the shaft will rotate eccentrically, which will cause large vibrations and catastrophic failure. ***Do not use a single-flex coupling with a parallel misalignment:*** this will bend the shafts, causing excessive bearing loads and premature failure.

***Double-Flex Coupling***

Use a double-flex coupling whenever two shafts are joined that are fixed in the radial and angular direction. (This is the most common situation. It results from a combination of angular and parallel misalignment).

***Rigid Coupling***

As mentioned above, rigid couplings would be ideal in servo systems, but are not generally recommended because of system misalignment. They should be used only if the motor or load is on some form of floating mounts that allow for alignment compensation. Rigid couplings can also be used when the load is supported entirely by the motor's bearings. A small mirror connected to a motor shaft is an example of such an application.

## RESONANCE ISSUES

A coupler that is too flexible may cause a motor to overshoot its commanded position. When the encoder sends a position feedback signal, the controller will command a correction move in the opposite direction. If the resonant frequency of the system is too low (too flexible), the motor may overshoot again and again. In extreme cases, the system could become an oscillator.

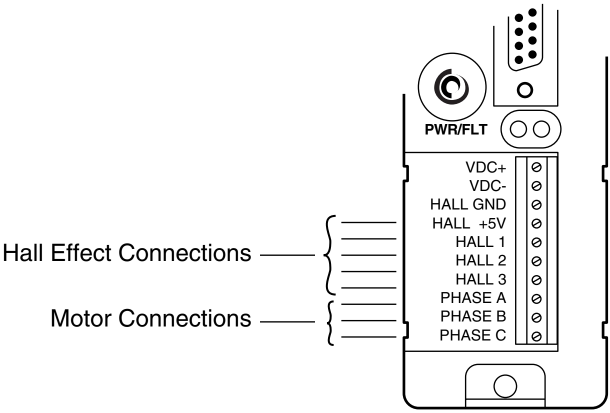
To solve resonance problems, increase the mechanical stiffness of the system to raise the resonant frequency so that it no longer causes a problem.

If you use a servo as a direct replacement for a step motor, you may need to modify your mechanical coupling system to reduce resonance. For example, we recommend using a bellows-style coupler with servo motors, rather than the helical-style coupler that is often used with step motors. Helical couplers are often too flexible, with resonant frequencies that can cause problems. Bellows couplers are stiffer, and perform better in servo systems.

## Connecting a Motor to the Drive

The OEM770X drive is designed to work with three-phase brushless motors equipped with Hall effect sensors or equivalent feedback signals. The typical motor has a permanent-magnet rotor with four poles (two pole pairs).

Connect your motor's phase wires and Hall effect sensor wires to the 10-pin screw terminal on the OEM770X. Each terminal is labeled with the name of the wire you should connect to it.



*10-Pin Screw Terminal*

14 AWG (2.5 mm$^2$) is the maximum wire size that can fit in the connector.

**CAUTION**

Do not turn on power unless the motor's Hall effect sensors, Hall +5, and Hall GND are connected to the drive. The motor may be destroyed by overheating if these connections are not made.

If the Hall effects are not connected, the drive determines that it is configured to run a *brushed* servo motor. With power and a command input applied, the drive will send the commanded DC current through the motor. If the motor is a *brushless* motor, it will not turn. Full current may flow in the motor and cause overheating, or destroy the motor within a short period of time.

## CONNECTING COMPUMOTOR SM AND NEOMETRIC SERIES MOTORS

To connect a Compumotor SM or NeoMetric Series motor to the OEM770X, follow the color code shown below for flying lead or cable versions. (These motors have additional wires not used by the OEM770X. See *Chapter 3 Specifications* for colors and functions of the additional wires.)

| Function | Wire Color |
|---|---|
| Hall Ground | White/Green |
| Hall +5V | White/Blue |
| Hall 1 | White/Brown |
| Hall 2 | White/Orange |
| Hall 3 | White/Violet |
| Phase A | Red/Yellow |
| Phase B | White/Yellow |
| Phase C | Black/Yellow |

Connect each motor wire to its appropriate screw terminal on the OEM770X. Wire sizes used for Compumotor motors are:

| | |
|---|---|
| Phase: | 18 AWG (O.75 mm$^2$) |
| Hall/Encoder: | 24 AWG (0.25 mm$^2$) |

## CONNECTING MOTORS FROM OTHER VENDORS

Before connecting a motor from another vendor, you must determine which motor phase wires correspond to Phase A, Phase B, and Phase C on the OEM770X. Similarly, you must determine which Hall effect wires correspond to Hall 1, Hall 2, and Hall 3.

Connect each wire to its appropriate terminal on the OEM770X. Ensure that the Hall effect sensors accurately transmit information about rotor position, and that motor current is commutated to the correct motor phases. See *Chapter 5 Hall Effect Sensors* for more information.

If your drive arrived with a response resistor installed, you should consider using a different response resistor. See *Chapter 4 Special Internal Circuits* for details about selecting a response resistor to improve your system's performance.

## CONNECTING A BRUSHED DC SERVO MOTOR

You can use the OEM770X as a drive for brushed DC servo motors. Follow these steps:

1. Connect HALL 1 and HALL 2 to HALL GND.
2. Make no connections to HALL 3.
3. Connect the drive’s Phase A to your motor’s positive input.
4. Connect the drive’s Phase C to your motor’s negative input.

Under these conditions, the drive’s internal logic determines that a brushed motor is connected. DC current will flow out of Phase A, through the motor, and back into the drive through Phase C. The amount and polarity of the current will be determined by the command input signal.

## SHIELDED MOTOR CABLES

Prevent electrical noise from interfering with the signals that the Hall effect sensors send to the drive. Position the motor as close to the drive as possible. If you need to connect a long cable between the drive and motor, we recommend you use a shielded cable for the Hall wires (Hall 1, Hall 2, Hall 3, +5V, GND). Run the power wires (phase A, B, and C) separately from the Hall wires.

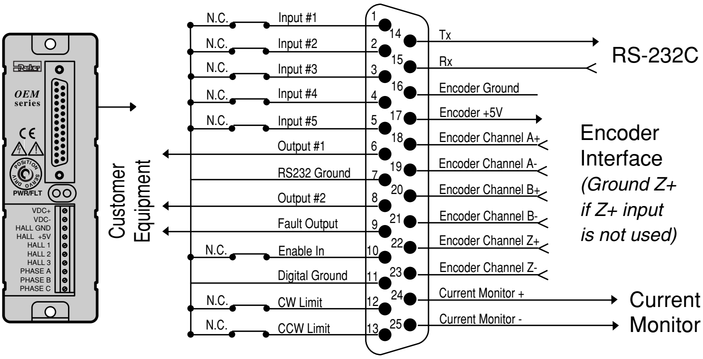
## MOTOR GROUNDING

For safety reasons, the motor should be grounded. Often, the motor can be grounded through the equipment to which it is mounted. This requires a good electrical connection between the motor’s mounting flange and the equipment, and that the equipment be connected to ground. Check with the National Electrical Code (NEC) and your local electrical code to ensure you use proper grounding methods.

Proper grounding can also reduce electrical noise.

# OEM770X Inputs and Outputs

Inputs and outputs are located on the 25 pin D-connector.



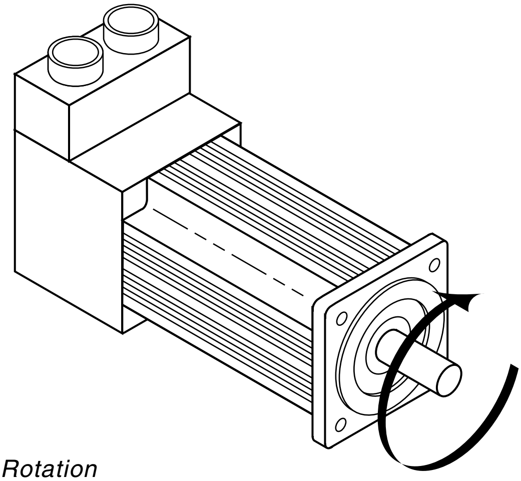
*OEM770X Inputs & Outputs*

***CAUTION***

I/O is not OPTO isolated, I/O GND (pins 7, 11, 16 & 24) are common to VDC- and HALL GND. For greater noise immunity, we recommend using optical isolation modules. For added noise immunity, the OEM770X has a digital filter; each input must be true for three successive cycles before it recognizes a given state.

### CLOCKWISE AND COUNTERCLOCKWISE – DEFINITIONS

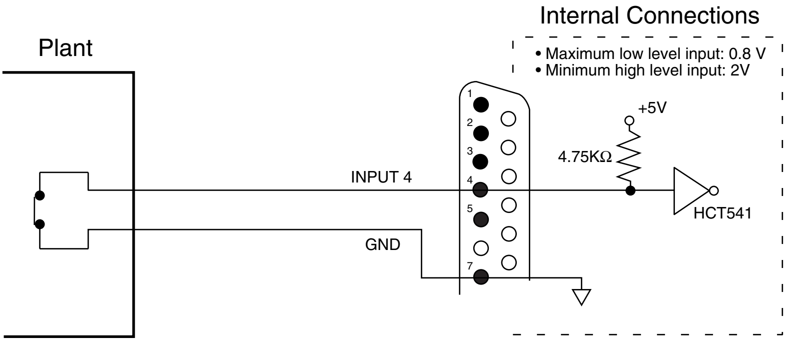
Shaft rotation is defined as the direction the shaft rotates, as viewed from the mounting flange end of the motor.



*Clockwise Shaft Rotation*

## General Purpose Inputs (Signal 1-Signal 5)

The OEM770X has 5 general purpose inputs. Each of these inputs may be configured to match the application needs. The figure represents a typical configuration of one of these inputs.


Internal Connections
Plant
• Maximum low level input: 0.8 V
• Minimum high level input: 2V
+5V
4.75KΩ
INPUT 4
GND
HCT541
1
2
3
4
5
7

Internal Connections for Pins 1-5 are identical

General Purpose Input Connected to a Switch

The **IN** command is used to configure the inputs to the following functions:

***Trigger Input***

The OEM770X can dedicate up to five Trigger inputs. These inputs are pulled up internally. These inputs are used with the Trigger (**TR**) command to control the OEM770X's trigger function. Minimum pulse width is 1 ms.

***Home Position Input***

The OEM770X can dedicate up to one Home input. The Home input allows you to establish a home reference position. This input is not active during power-up. Refer to the Go Home (**GH**) command for more information on setting up and using this function. Minimum pulse width is 1 ms.

***Sequence Select Input***

The OEM770X can dedicate up to three Sequence Select inputs that allow you to control seven different sequences. Sequences are executed remotely by using one of the following logic patterns in conjunction with the **XP** command.

| **Sequence #** | **Ø** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|---|
| SEQ Input #1 | Ø | 1 | Ø | 1 | Ø | 1 | Ø | 1 |
| SEQ Input #2 | Ø | Ø | 1 | 1 | Ø | Ø | 1 | 1 |
| SEQ Input #3 | Ø | Ø | Ø | Ø | 1 | 1 | 1 | 1 |

Ø = low, pulled to ground
1 = high, 5VDC

### *Stop or Kill Input*

The OEM770X can dedicate up to one Stop and one Kill input. The Stop or Kill input is identical in function to the effect of the **S** or **K** command respectively.

### *Go Input*

The OEM770X can dedicate up to one Go input. The active state is high. The Go input is identical in function to the effect of the GO (**G**) command.

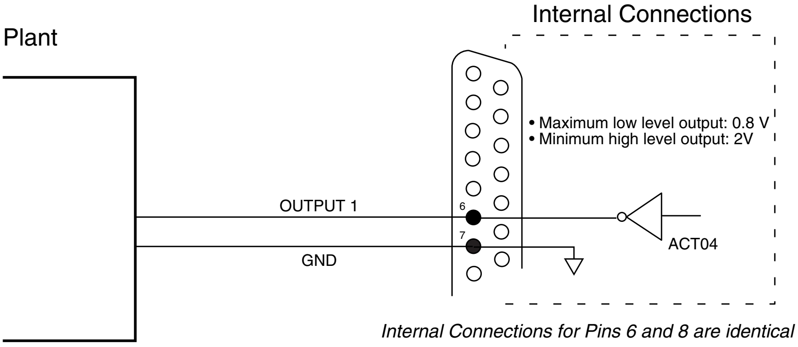
**CAUTION**

Unless configured otherwise (**SSH** command), the controller will dump the commands following the IN command in the buffer. Please pay special attention to the state of the inputs before entering the IN command.

## OUTPUT #1 (SIGNAL 6) AND OUTPUT #2 (SIGNAL 8)

The OEM770X has two dedicated programmable +5 volt outputs. They may be used to signal peripheral devices upon the start or completion of a move. The default state for Outputs #1 and #2 is logic low. Refer to the Output (**O**) command for information on using these outputs.

The next drawing shows the schematic for one of the outputs.

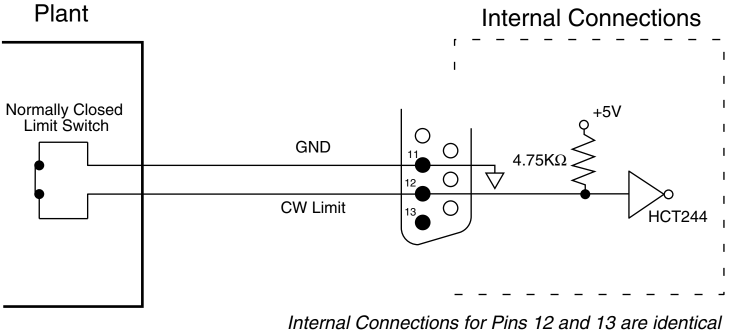

Internal Connections
Plant
• Maximum low level output: 0.8 V
• Minimum high level output: 2V
OUTPUT 1
6
7
GND
ACT04
Internal Connections for Pins 6 and 8 are identical

*General Purpose Outputs*

## CW (SIGNAL 12) & CCW (SIGNAL 13) LIMIT INPUTS

The OEM770X has two dedicated hardware end-of-travel limits (CCW and CW ). When you power up the OEM770X, these inputs are enabled (high). To test the OEM770X without connecting the CCW and CW limits, you must disable the limits with the **LD3** command. You can use the Limit Switch

Status Report (**RA**) and Input Status (**IS**) commands to monitor the limits' status. The figure represents a typical configuration of these inputs. Minimum pulse width is 1 ms.

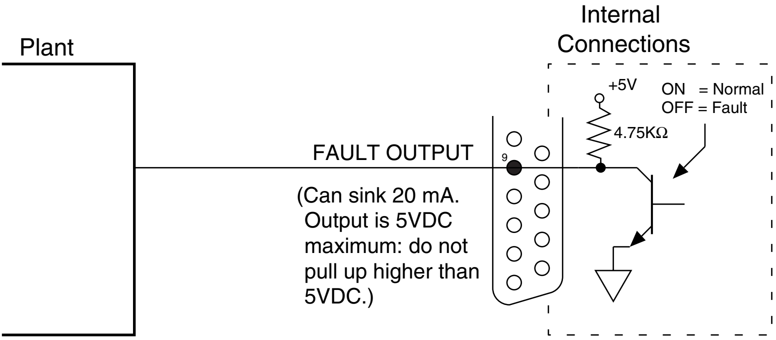


Plant
Internal Connections
Normally Closed Limit Switch
GND
CW Limit
11
12
13
+5V
4.75KΩ
HCT244
Internal Connections for Pins 12 and 13 are identical

*Limit Switch Inputs*

## DEDICATED FAULT OUTPUT (SIGNAL 9)

The OEM770X has one dedicated fault output**.** This output may be used to signal peripheral devices if a drive fault occurs. The Fault output's default state is logic high. When the front panel LED signals a drive fault, this signal will be active high. The **RSE** command will report the controller's error conditions. The next figure represents a typical configuration of this output.

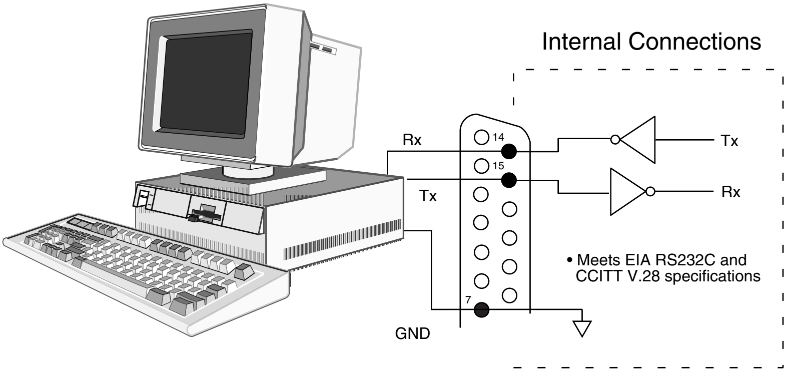


Plant
Internal Connections
+5V
ON = Normal
OFF = Fault
4.75KΩ
FAULT OUTPUT
9
(Can sink 20 mA. Output is 5VDC maximum: do not pull up higher than 5VDC.)

*Fault Output*

## RS-232C—Tx (Signal 14), Rx (Signal 15), and Ground (Signal 7)

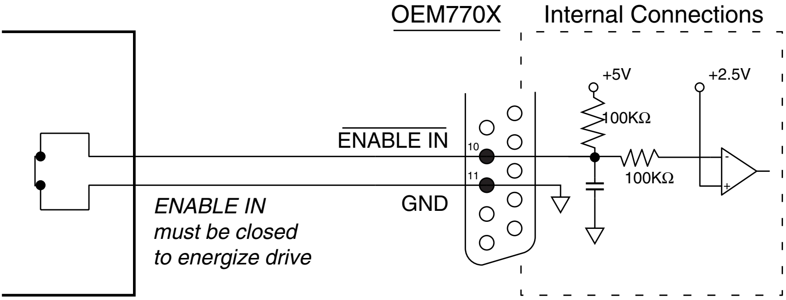
The OEM770X uses RS-232C as its communication medium. The OEM770X does not support handshaking. A typical three-wire (Rx, Tx, and Signal Ground) configuration is used. The figure represents a typical RS-232C configuration.



*RS232 Input*

## Enable Input (Signal 10)

The OEM770X has an enable input that must be connected to ground for the drive to be enabled. As the next drawing shows, you can connect the enable input to a switch for a hardware method of disabling the drive. When the switch is opened, the drive will be disabled.



*Enable Input Connected to a Switch*

If you need an emergency stop, do not use the hardware switch alone. There will be no torque on the motor when the drive is disabled. If the motor is moving, it will freewheel until it slows to a stop. Loads in vertical applications may drop due to gravity. Use a mechanical brake, in conjunction with the hardware disable switch, as an emergency stop.

***WARNING***

Disabling the drive with a switch is not an emergency stopping method. There is no torque on the motor when the drive is disabled. Therefore, you must also use a mechanical brake or some other means to stop the motor.

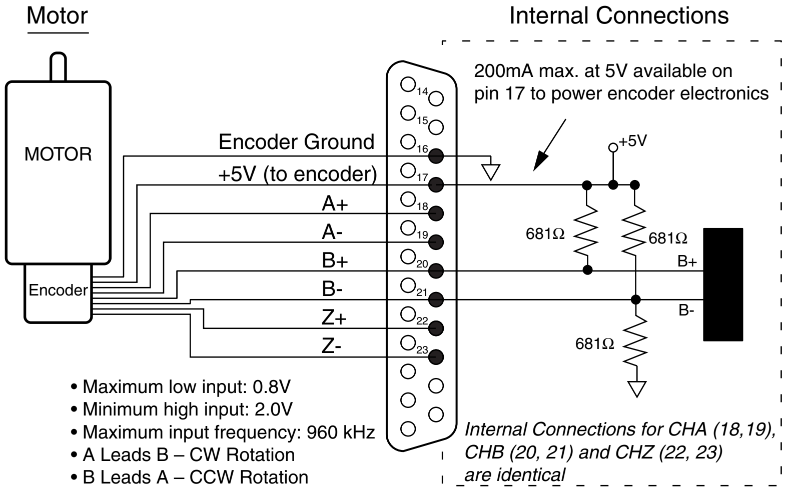
If you do not need to disable the drive by a hardware switch, connect a jumper wire from pin 10 to ground. The drive will then be enabled when it powers up. The OEM770X can also be disabled by software commands. For more information, see the shutdown command, **ST**, and the **OFF** command in the software reference section.

## ENCODER INPUTS +5V, A, B, Z, GND (SIGNALS 16 - 23)

The OEM770X has six dedicated inputs for use with a differential incremental encoder. These inputs provide the position information for the servo loop.

***CAUTION***

If you do not connect an encoder Z-channel output to the OEM770X, then you must ground the Z+ input on the OEM770X. To do this, connect a jumper wire between the Z+ input (pin 22), and the nearest available ground (pin 16 or 11).

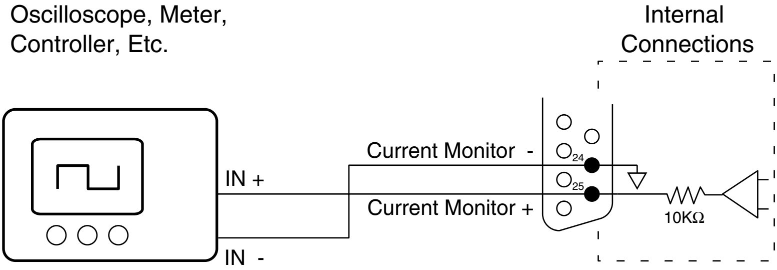

Motor
Internal Connections
MOTOR
Encoder
Encoder Ground
+5V (to encoder)
A+
A-
B+
B-
Z+
Z-
14
15
16
17
18
19
20
21
22
23
200mA max. at 5V available on pin 17 to power encoder electronics
+5V
681Ω
681Ω
681Ω
B+
B-
• Maximum low input: 0.8V
• Minimum high input: 2.0V
• Maximum input frequency: 960 kHz
• A Leads B – CW Rotation
• B Leads A – CCW Rotation
Internal Connections for CHA (18,19), CHB (20, 21) and CHZ (22, 23) are identical

*Encoder Input*

## Current Monitor

You can use the OEM770X's current monitor output to measure motor current. Connect pin 25 to the positive input of your oscilloscope, meter, etc. Use pin 24 as a signal ground for your oscilloscope or meter.

The OEM770X monitors the actual motor current. It puts out a voltage on pin 25 that is proportional to current, with 1 volt out = 1.2 amps of motor current. Positive voltages correspond to clockwise rotation (as viewed from the mounting flange end of the motor). Negative voltages correspond to counter-clockwise rotation.

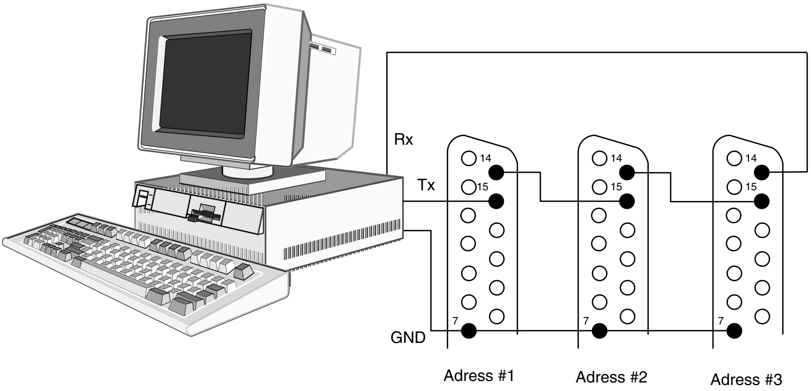


*Current Monitor Output Connections*

## Daisy Chaining

You may daisy chain up to 255 OEM770Xs. Individual drive addresses are set with the **#** (Address Numbering) command. When daisy chained, the units may be addressed individually or simultaneously. You should establish a unique device address for each OEM770X.

Refer to the figure below for OEM770X daisy chain wiring.



*Daisy Chain of 3 OEM770Xs*

Commands prefixed with a device address control only the unit specified. Commands without a device address control all units on the daisy chain. The general rule is: *Any command that causes the drive to transmit information from the RS-232C port (such as a status or report command), must be prefixed with a device address*. This prevents daisy chained units from all transmitting at the same time.

Attach device identifiers to the front of the command. The Go (**G**) command instructs all units on the daisy chain to go, while **1G** tells only unit 1 to go.

When you use a single communications port to control more than one OEM770X, all units in a daisy chain receive and echo the same commands. Each device executes these commands, unless this command is preceded with an address that differs from the units on the daisy chain. This becomes critical if you instruct any OEM770X to transmit information. To prevent all of the units on the line from responding to a command, you must precede the command with the device address of the designated unit.

## Connecting a Power Supply

The OEM770X requires a single external power supply with these features:

- 24VDC to 75VDC
- Fast Transient Response (can quickly supply enough current to meet your application's requirements)
- Power Dump (not required for all applications)

The power dump may be required if your system produces excess regenerated energy. To avoid damage, dissipate the regenerated energy in a power resistor, store it in extra capacitance (a blocking diode may be needed), or provide some other means to absorb regenerated energy.

For information about power supply selection, regeneration, and power dump methods, see *Chapter 6 Power Supply Selection*. The following table briefly lists the type of power supply you can use for different applications.

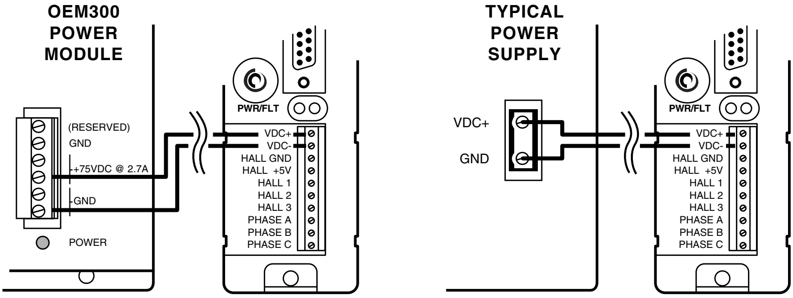
| **Application** | **Recommended Power Supply** |
|---|---|
| Very Low Power<br>(low regen) | 24-48VDC Switching Power Supply<br>24-48VDC Linear Unregulated Supply<br>OEM300 Power Module |
| Low Power<br>(with regen) | Switching Power Supply with blocking diode and extra capacitance.<br>Linear Unregulated Supply<br>OEM300 Power Module |
| High Power<br>(low regen) | Linear Unregulated Supply with Transformer<br>OEM1000 Power Supply |
| High Power<br>(with regen) | Linear Unregulated Supply with added Capacitance or added Power Dump<br>OEM1000 Power Supply |

The Compumotor OEM300 Power Module is a single unit that contains a 75VDC/300W power supply, integral power dump, and several protective circuits.

The Compumotor OEM1000 Power Supply is a linear power supply that can provide 1000W/15A at 70VDC.

## Connecting the Power Supply

Connect your power supply to the 10 pin screw terminal on the OEM770X. The next drawing shows connections for a typical power supply, and for an OEM300 Power Module.



*Power Supply Connections*

- Connect the positive DC terminal of your power supply to the VDC+ input on the OEM770X's 10-pin screw terminal.
- Connect the ground terminal of your power supply to VDC- on the OEM770X.

To reduce electrical noise, minimize the length of the power supply wires and twist them tightly together.

### *Grounding*

Internally, VDC- is connected to the Hall Ground and the grounds on the 25 pin D-connector (pins 7, 11, 16, 24). Do not connect your power supply's ground to these pins, however. Connect it only to VDC-.

The shell of the 25 pin D-connector and the heatplate are connected internally. They are not connected to VDC-, Hall Ground, or the D-connector grounds (pins 7, 11, 16, 24).

### *Wire size*

Use 18 AWG (0.75 mm$^2$) or greater diameter wire for power connections. For applications that use high peak power, use larger diameter wires. 14 AWG (2.5 mm$^2$) wire is the biggest wire that will fit in the 10-pin screw terminal.

## Tuning

The OEM770X uses a digital *Proportional Integral Derivative* (PID) filter to compensate the control loop. For best performance, you must tune the filter's parameters. A properly tuned system will exhibit smooth motor rotation, accurate tracking, and fast settling time.

All tuning is performed via RS232-C communications.

***Saving the Settings***

Tuning parameters are saved in memory on the OEM770X. The –M2 option (battery backed RAM) allows gains to be saved when the drive is turned off:

| | |
|---|---|
| *–M2 Option Installed* | Gains are saved in nonvolatile memory |
| *–M2 Option* **NOT** *Installed* | Gains are **not saved** when drive is off |

If you do not have the –M2 option, you must reload the gains each time power is cycled to the OEM770X.

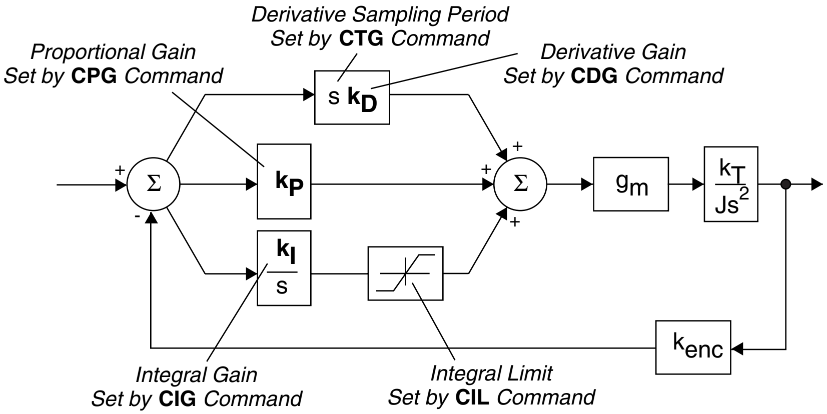
### PID TUNING

In the procedure described below, you will systematically vary the tuning parameters until you achieve a move that meets your requirements for accuracy and response time.

The OEM770X generates a move profile based upon the user supplied acceleration, velocity, and distance commands (**A**, **V**, and **D**). At each servo sampling period (every 266 microseconds), the OEM770X calculates the position the motor should reach as it follows the move profile. This is called *commanded position*, and is one of two inputs to a summing node. Position information from the encoder, which is called *actual position*, is the other input to the summing node. During a typical move, actual position will differ from commanded position by at least a few encoder counts. When actual position is subtracted from commanded position at the summing node, an error signal is produced. The error signal is the input to the PID filter.

The position specified by the distance command (**D**) is called the *target position*. During the move, commanded position is not the same as target position. The commanded position is incremented each sampling period. When it finally matches the target position, the move is over.

The servo block diagram is shown on the following page.



As the figure shows, you can adjust five different parameters to tune the PID filter. The relevant commands are:

| | |
|---|---|
| **CPG** | Configure Proportional Gain |
| **CDG** | Configure Derivative Gain |
| **CTG** | Configure Derivative Sampling Period |
| **CIG** | Configure Integral Gain |
| **CIL** | Configure Integral Limit |

To tune the system, you will iteratively increase **CDG** and **CTG** to their optimum values, then increase **CPG** to its optimum value. If necessary, you will also increase **CIG** and **CIL**.

In general, you will set **CDG** and **CPG** as high as possible, and **CIG** as low as possible. Trade-offs between response time, stability, and final position error will dictate the values you select. For loads that vary during operations, you can download new parameters by using buffered versions of the five tuning commands (**BCPG, BCDG, BCTG, BCIG, BCIL**).

***WARNING***

During servo tuning, the system can undergo accidental and violent movement due to improper gain settings and programming errors. Please use extreme caution while prototyping.

Each tuning parameter is described in the following sections.

### CPG – *Proportional Gain*

Proportional gain provides a torque that is directly proportional to the *magnitude* of the error signal. Proportional gain is similar to a spring—the larger the error, the larger the restoring force. It determines the stiffness of the system and affects the following error. High proportional gain gives a stiff, responsive system, but can result in overshoot and oscillation. Damping—provided by derivative gain—can reduce this overshoot and oscillation.

### *CDG – Derivative Gain; CTG – Derivative Sampling Period*

Derivative gain provides a torque that is directly proportional to the *rate of change* of the error signal. The previous error is subtracted from the present error each sampling period. The difference represents the error's instantaneous rate of change, or *derivative*. The difference is multiplied by the value set by the **CDG** command, and the product contributes to the motor control output.

Derivative gain opposes rapid changes in velocity. It will dampen the resonance effects of proportional gain. With higher derivative gain, you can use higher proportional gain.

You can use the **CTG** command to make the derivative sampling period longer than the system's sampling period. The system sampling period—266 µsec—is the period between updates of position error, and cannot be changed. The derivative sampling period is an integer multiple of the system sampling period. It can range from 266 µsec to 68 msec, in increments of 266 µsec (for example: CTG0 = 266 µs, CTG1 = 532 µs, CTG2 = 798 µs, etc.).

With a longer derivative sampling period, more time elapses between error measurements. The difference between previous and present error is still multiplied by the CDG value. The product contributes to the motor control output every *system* sampling period, but is only updated every *derivative* sampling period. This gives a more constant derivative term and improves stability. Low velocity systems in particular can benefit from a longer sampling period.

### *CIG – Integral Gain; CIL – Integral Limit*

Integral gain provides a torque that is directly proportional to the sum, over time, of the error values—the *integral* of the error. The controller reads the error value every sampling period, and adds it to the sum of all previous error values. The sum is multiplied by the value set by the **CIG** command, and the product contributes to the motor control output every system sampling period.

Integral gain can remove steady state errors that are due to gravity or a constant static torque. Integral gain can also correct velocity lag that can occur in a constant velocity system.

If error persists during a move, the sum of the error values may be quite high at the end of the move. In this case, the torque provided by the integral gain can also be very high, and can cause an overshoot. This effect is called *integral windup*. You can use **CIL**, the integral limit command, to set a maximum value for integral gain. The integral limit constrains the integral term to values less than or equal to **CIL**, which will reduce the overshoot caused by integral windup.

## TUNING PROCEDURE

You can manually tune the OEM770X by varying the tuning parameters while you empirically evaluate the system response. This manual method works well in most applications. You can also connect an oscilloscope to the OEM770X's current monitor output, and observe the motor current waveform. This will give you information about the system's step response. You can then adjust the parameters until you obtain the step response you desire. Both of these methods are explained below.

### *Tuning Procedure – Manual Method*

You will achieve best results by making a consistent, repetitive move that is representative of your application.

1. ***Issue a RETURN TO FACTORY SETTINGS command (RFS)***
   The RFS command will reset the gains to their default values (CDG240, CTG0, CPG16, CIG2, CIL2, CPE4000)

2. ***Decrease CPG***
   Decrease **CPG** to zero (**CPG0**). If your system has very little friction, internal offsets in the drive may cause the motor to run away (spin faster and faster) with **CPG** set to zero. If the motor runs away, issue an **OFF** command. When the motor stops, increase **CPG** by one unit (**CPG1**), and issue an **ON** command. If the motor continues to run away, repeat this procedure—incrementing **CPG** by one unit—until the motor remains stopped.

3. ***Decrease CIG***
   Set the integral gain to zero (**CIG0**).

4. ***Increase Derivative Gain (CDG) and Derivative Sampling Period (CTG)***
   Determine **CDG** and **CTG** iteratively. Increase **CDG** until the shaft begins high frequency oscillations, then increase **CTG** by one. With a higher **CTG**, the oscillations should be damped. Again increase **CDG** until oscillations occur, then increase **CTG** by one. Repeat this process until **CTG** reaches a value appropriate for the system.

   In general, you will want values for **CDG** and **CTG** that are as large as possible, without producing unacceptably high motor vibrations. However, many systems will require a low **CTG** value, to ensure that the derivative sampling period is shorter than one tenth of the system mechanical time constant. Therefore, start with a low **CTG** and gradually increase it, rather than immediately trying a large **CTG** value.

5. ***Increase Proportional Gain (CPG)***
   Determine the **CPG** value iteratively. Increase **CPG**, and evaluate the system damping. Repeat until the system is critically damped. You should increase **CPG** to the largest value that does not cause overshoot or ringing. Because proportional gain and derivative gain affect each other, you may need to repeat step 4 and step 5 several times to arrive at optimum values for **CPG** and **CDG**.

6. ***Determine Integral Gain (CIG) and Integral Limit (CIL) Values***
   High values for **CIG** will make the system respond quickly, but can cause other problems. In general, you should set **CIG** to the *lowest* value that will correct following errors and static position errors, but not increase overshoot or settling time. In a system without static torque loading, a **CIG** of zero may be appropriate.

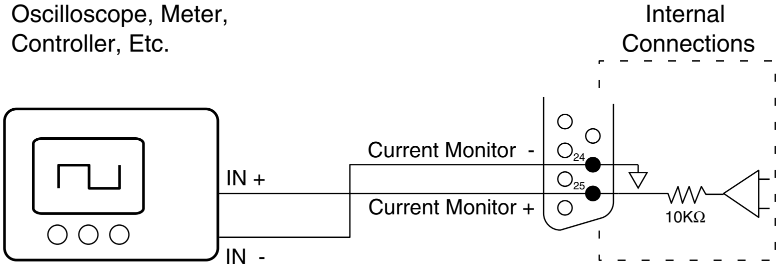
   **CIL** limits **CIG**—therefore, before you increase **CIG** to a particular value, you must first increase **CIL** to an equal or higher value.

### *Tuning Procedure – Step Response Method*

You can use the OEM770X's current monitor output to view a signal that is a scaled replica of motor current. Use this method if you were unable to obtain satisfactory results from the manual tuning method described above, or if you need more performance information to help you tune your system.

The step response for a motion control system is a graph of actual position versus a small and instantaneous change in commanded position. An oscilloscope, connected as described below, does not show position—it shows motor current. However, in most applications motor current is directly related to position. Viewing the current waveform, therefore, can give you insight into the system's step response.

Connect an oscilloscope to the current monitor, as shown below.

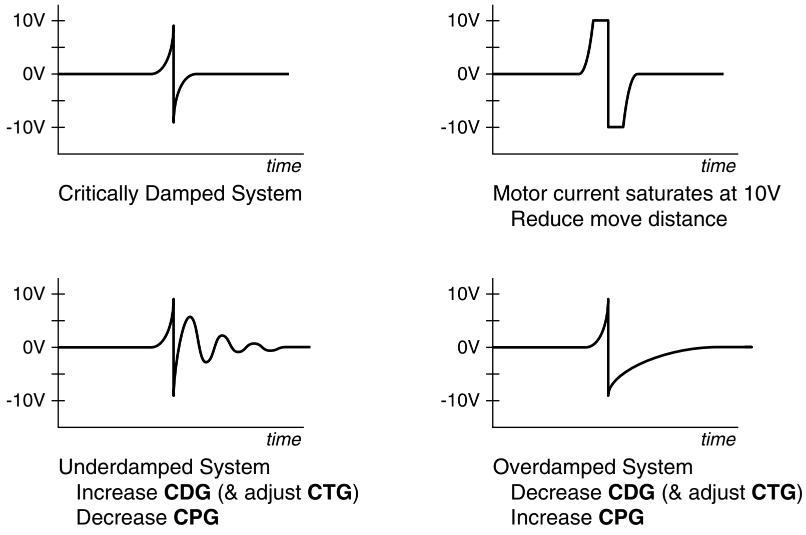


*Current Monitor Output Connections*

The voltage on pin 25 can range from -10VDC to +10VDC. It is scaled to be proportional to motor current, with 1 volt out = 1.2 amps of motor current. Positive voltages correspond to clockwise rotation of the motor shaft.

Command the longest move that does not cause the motor current to saturate (keep the current monitor output below 10V). Typical move distances are 250 to 500 encoder counts. Use a high velocity and high acceleration.

The next drawing shows typical response waveforms, along with suggestions on how to adjust the tuning parameters to achieve a critically damped system.



Critically Damped System

Motor current saturates at 10V
Reduce move distance

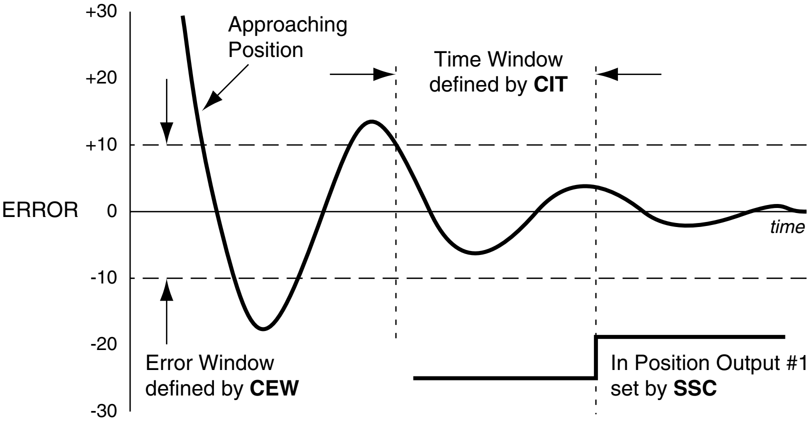
Underdamped System
Increase **CDG** (& adjust **CTG**)
Decrease **CPG**

Overdamped System
Decrease **CDG** (& adjust **CTG**)
Increase **CPG**

*Step Response Waveforms*

## CONFIGURING AN IN POSITION WINDOW

You can define an In Position Window, and use it to indicate that the preceding move is done. Two commands—**CEW** and **CIT**—determine the height and width of the window. A third command—**SSC**—can turn on output #1 when the In Positon criteria are met.


+30
+20
+10
ERROR 0
-10
-20
-30
Approaching Position
Time Window defined by CIT
time
Error Window defined by CEW
In Position Output #1 set by SSC

As the drawing shows, **CEW** defines the position error window at the end of a move. **CIT** specifies the length of time the motor must be within the error window. The motor is In Position when three conditions are satisfied:

1. The controller algorithm is finished (no input position command)
2. Position error is less than that specified by the **CEW** command
3. Condition *2* above has been true for the time period specified by **CIT**

If **SSC** has been set to 1, output #1 will turn on when these three conditions have been met. You can use output #1 to trigger external hardware from the In Position condition. The output will stay on until the next move command is issued, such as **GO** or **GO HOME**.

(*Note:* If the motor is held (mechanically, or against an end stop), and **CPE** is greater than **CEW**, the motor may become “trapped” between **CPE** and **CEW**: it will not execute the next move. In this rare situation, two things are happening:

1. **CPE** is not violated, and therefore no position error fault occurs;
2. in position criteria are not met.

If you were to execute a **1R**, the response would be ***B**, which means the drive is “busy” waiting for the move to be over. Why doesn’t the drive force the motor to finish the move? The motor is somehow held. To correct this situation, try touching the motor; this may complete the move, and the drive may execute the next move. Or, execute a **DPA** to read actual position, and verify that the move is not complete. You can also execute a **KILL** to reset the positions, and then do the next series of moves.)

# CHAPTER 3

## *Specifications*

Complete specifications for the OEM770X Drive and Parker Compumotor SM, NeoMetric, and J Series motors are listed in this chapter.

The motors are described by the following numbering system:

| SM | Frame, Magnet Length | Winding | Feedback | Shafting | Connections | Options |
|---|---|---|---|---|---|---|
| SM Series | 161<br>162<br>231<br>232<br>233 | A<br>B | D – 500 ppr encoder[1]<br>E – 1,000 ppr encoder[1]<br>H – Hall-effect only<br>R – resolver | N – normal<br>F – flat<br>K – keyway[2]<br>L – extended | MS – military style<br>10 – 10' cable[3]<br>25 – 25' cable[3]<br>FL – 18" leads<br>TQ – TQ amp series | N – none<br>V – shaft seal[4]<br>Gxx – Gearhead |

[1] includes Hall-effect
[2] not available on size 16
[3] cable is hard-wired
[4] size 23 w/MS or TQ connectors—IP65

| N | Frame, Magnet Length | Winding | Feedback | Shafting | Connections | Options |
|---|---|---|---|---|---|---|
| NeoMetric Series | xxxy<br>xxx = flange dia.<br>070 for 70 mm<br>034 for 34 frame<br>092 for 92 mm<br>y = magnet length reference 1 to 4 | Identifying character<br>D, E, F, etc. | D – 500 ppr encoder[2]<br>E – 1,000 ppr encoder<br>H – Hall-effect only<br>R – resolver | N – normal<br>F – flat<br>K – keyway | FL[2]<br>MS<br>TQ[4]<br>PT[1]<br>10[2] | B – brake<br>N – none<br>V – IP65[5]<br>W – IP67<br>Gxx – Gearhead[3] |

[1] 92 mm motors only
[2] 70 mm and 34 frame motors only
[3] 34 frame motors only
[4] Not available on 92 mm motors
[5] IP65 when specified with MS or TQ connections

| J | Frame, Magnet Length | Winding | Feedback | Shafting | Connections | Options |
|---|---|---|---|---|---|---|
| J Series | xxxy<br>xxx = flange dia.<br>070 for 70 mm<br>034 for 34 frame<br>092 for 92 mm<br>y = magnet length reference 1 to 3 | Identifying character<br>D, E, F, etc. | D – 500 ppr encoder<br>E – 1,000 ppr encoder<br>H – Hall-effect only<br>R – resolver | N – normal<br>F – flat<br>K – keyway | FL[2]<br>MS<br>TQ[4]<br>PT[1]<br>10[2] | B – brake<br>N – none<br>V – IP65[5]<br>Gxx – Gearhead[3] |

[1] 92 mm motors only
[2] 70 mm motors only
[3] 34 frame motors only
[4] Not available on 92 mm motors
[5] IP65 when specified with MS or TQ connections

## Specifications: OEM770X Controller/Drive

### OEM770X Controller/Drive – Specifications

| | | |
|---|---|---|
| **POWER INPUT** | | |
| | Voltage | 24-75VDC |
| | Current | Ø-12 amps |
| **POWER OUTPUT—MOTOR** | | |
| | Peak Current | 12A (approx. 2 sec maximum duration |
| | | at 45°C ambient temperature. |
| | | See *Current Foldback* for details) |
| | Continuous Current | 6A |
| | Voltage | 90VDC maximum |
| | Peak Power | 840W (1.1 hp) (@75V supply voltage) |
| | Continuous Power | 420W (0.56 hp) |
| | Switching Frequency | 20 kHz |
| | Bandwidth | 2 kHz typical (dependent on motor) |
| | Transconductance | 1 volt = 1.2 amp |
| | Commutation | 120° Hall Effect Sensors for |
| | | Six-State Commutation Method, |
| | | or Brushed DC Motor |
| | Short Circuit Protected | Yes |
| **POWER OUTPUT—HALL EFFECT SENSORS** | | |
| | Voltage | +5VDC ± 0.5VDC |
| | Current | 50 mA (maximum) |
| | Overload Protected | Yes |
| **PERFORMANCE** | | |
| | Position Range | ±1,073,741,823 |
| | Velocity Range | 0.01 to 200 rev/s |
| | Acceleration Range | 0.01 to 9999 rev/$s^2$ |
| | Velocity Accuracy | ±0.02% of maximum rate |
| | Velocity Repeatability | ±0.02% of set rate |
| **INPUTS** | | |
| | General Purpose Inputs | 5 user defined inputs |
| | Limit Inputs | 2 hardware limits |
| | Enable Input | Active LOW: Ø-0.8V @ 2mA |
| | | When disabled: Internal 2.49 KΩ |
| | | pullup resistor to +5VDC |
| | Encoder | 2-phase differential (recommended) or |
| | | single ended (+5VDC TTL compatible). |
| | | 960 kHz maximum frequency. |
| **HALL INPUTS** | | |
| | Low State | Ø-0.8V |
| | High State | Internal 1 KΩ pullup resistor to +5V |
| | Input Frequency | Ø-2 kHz maximum |

## OEM770X Controller/Drive – Specifications (contin.)

### OUTPUTS

| | |
|---|---|
| General Purpose | 2 user defined general purpose outputs |
| Fault Output | Active HIGH: open collector output |
| | Inactive LOW: Ø-0.4VDC at Ø-20 mA |
| Current Monitor | -10V to +10V analog voltage |
| | Scale: 1V corresponds to 1.2A output |
| | Output Impedance: 10 KΩ |
| LEDs | GREEN = power; RED = various faults |

### DIGITAL SERVO LOOP

| | |
|---|---|
| Update Time | 266 microseconds |
| Output | 12 bid DAC |
| Servo Tuning | RS232C command interface |
| Tuning Parameters | PID with digital filter |

### RS232C Interface

| | |
|---|---|
| Connections | 3-wire connection (Rx, Tx, and GND) |
| Maximum Daisy Chain | 255 units on daisy chain |
| Address Settings | Via # command |
| Parameters | 9600 baud; 1 stop, 8 data bits; no parity |

### PROTECTIVE CIRCUITS

| | |
|---|---|
| Short Circuit | Turns off outputs to motor; latched |
| Over Temperature | 55°C ± 5°C trip temperature; latched |
| Overvoltage | 95V ± 5V trip voltage; latched |
| Undervoltage | 21.5V maximum; not latched |
| Current Foldback | Configurable with 3 resistors |

### MOTOR CHARACTERISTICS

| | |
|---|---|
| Minimum Inductance | 50 µH (micro Henrys) |
| Minimum Resistance | 0.25 Ω |
| Loop Gain Adjustment | Configurable with one resistor |

### TEMPERATURE

| | |
|---|---|
| Minimum Temperature | Ø°C (32°F) |
| Maximum Temperature | 45°C (113°F) (max. heatplate temp.) |
| Storage Temperature | -30°C to 85°C (-22°F to 185°F) |
| Package Dissipation | Heatplate: 0 to 30W, depending on |
| | motor current; P = ($I_{AVG}$/12 A)30 W |
| | Cover: 3 watts maximum |
| Humidity | 0 to 95% non condensing |
| Contaminants | OEM770X is not waterproof, |
| | oilproof, or dustproof. |

### MECHANICAL

| | |
|---|---|
| Power Connector | 10 pin screw terminal |
| | 14 AWG (2.5 $mm^2$) maximum wire size |
| Input/Output Connector | 25 Pin D-connector |
| Size | 5x1.6x3.5 in (127x41x89 mm) approx. |
| Dimensions | see *Chapter 2 Installation* |
| Weight | 12 ounces (0.35 kg) |

# Motor Specifications: SM160

| Parameter | Symbol | Units | SM160A | SM160B |
|---|---|---|---|---|
| Stall Torque Continuous[1] | $T_{cs}$ | lb-in/oz-in | 0.8/13 | 0.8/13 |
| | | N-m | 0.09 | 0.09 |
| Stall Current Continuous[1] | $I_{cs}$ (trap) | Amps DC | 2.5 | 4.8 |
| Rated Speed[2] | $\omega_r$ | rpm | 7500 | 7500 |
| Peak Torque[6] | $T_{pk}$ | lb-in/oz-in | 2.5/40 | 2.5/40 |
| | | N-m | 0.28 | 0.28 |
| Peak Current[6] | $I_{pk}$ (trap) | Amps DC | 7.4 | 14.4 |
| Torque @ Rated Speed | $T_r$ | lb-in/oz-in | 0.6/10 | 0.6/10 |
| | | N-m | 0.07 | 0.07 |
| Rated Power–Output Shaft | $P_o$ | watts | 57 | 55 |
| Voltage Constant[3,4] | $K_b$ | volts/radian/sec | 0.038 | 0.020 |
| Voltage Constant[3,4] | $K_e$ | volts/KRPM | 4.02 | 2.08 |
| Torque Constant[3,4] | $K_t$ (trap) | oz-in/Amp DC | 5.43 | 2.81 |
| | | N-m/Amp DC | 0.038 | 0.020 |
| Resistance[3] | R | ohms | 3.43 | 0.90 |
| Inductance[5] | L | millihenries | 0.53 | 0.13 |
| Thermal Resistance | $R_{th}$ w-a | °C/watt | 3.2 | 3.2 |
| Motor Constant | Km | oz-in/√watt | 2.93 | 2.96 |
| | | N-m/√watt | 0.021 | 0.021 |
| Viscous Damping | B | oz-in/Krpm | 0.162 | 0.162 |
| | | N-m/Krpm | 1.13E-3 | 1.13E-3 |
| Static Friction | $T_f$ | oz-in | 0.10 | 0.10 |
| | | N-m | 7.0E-4 | 7.0E-4 |
| Thermal Time Constant | $\tau_{th}$ | minutes | 10 | 10 |
| Electrical Time Constant | $\tau_e$ | milliseconds | 0.16 | 0.15 |
| Mechanical Time Constant | $\tau_m$ | milliseconds | 11.7 | 11.5 |
| Rotor Inertia | J | lb-in-sec$^2$ | 4.4E-5 | 4.4E-5 |
| | | kg-m$^2$ | 5.0E-6 | 5.0E-6 |
| Weight | # | pounds | 0.7 | 0.7 |
| | kg | kilograms | 0.3 | 0.3 |
| Winding Class | | | H | H |

[1] @ 25°C ambient, 125°C winding temperature, motor connected to a 10"x10"x1/4" aluminum mounting plate, @ 40°C ambient, derate phase currents and torques by 12%.

[2] Maximum speed is 7500RPM with 500 line encoder. For 1000 line encoders, derate to 6000RPM. For higher speed operation, please call the factory.

[3] Measured line-to-line, ±10% line-to-line.

[4] Value is measured peak of sine wave.

[5] ±30% line-to-line, inductance bridge measurement @ 1 kHz.

[6] Initial winding temperature must be 60°C or less before peak current is applied.

**Note: These specifications are based on theoretical motor performance and are not specific to any amplifier.**

# Motor Specifications: SM161 and SM162

| Parameter | Symbol | Units | SM161A | SM161B | SM162A | SM162B |
|---|---|---|---|---|---|---|
| Stall Torque Continuous[1] | $T_{cs}$ | lb-in/oz-in | 1.6/26 | 1.6/26 | 2.9/47 | 3.1/49 |
| | | N-m | 0.18 | 0.18 | 0.33 | 0.34 |
| Stall Current Continuous[1] | $I_{cs}$ (trap) | Amps DC | 2.3 | 4.5 | 2.3 | 4.4 |
| Rated Speed[2] | $\omega_r$ | rpm | 7,500 | 7,500 | 7,500 | 7,500 |
| Peak Torque[6] | $T_{pk}$ | lb-in/oz-in | 4.9/78 | 4.9/78 | 8.8/141 | 9.1/145 |
| | | N-m | 0.55 | 0.54 | 0.99 | 1.02 |
| Peak Current[6] | $I_{pk}$ (trap) | Amps DC | 7.0 | 13.4 | 6.8 | 13.2 |
| Torque @ Rated Speed | $T_r$ | lb-in/oz-in | 1.1/18 | 1.1/18 | 2.3/37 | 2.3/37 |
| | | N-m | 0.13 | 0.13 | 0.26 | 0.26 |
| Rated Power–Output Shaft | $P_o$ | watts | 97 | 100 | 205 | 204 |
| Voltage Constant[3,4] | $K_b$ | volts/radian/sec | 0.079 | 0.041 | 0.147 | 0.078 |
| Voltage Constant[3,4] | $K_e$ | volts/KRPM | 8.27 | 4.29 | 15.39 | 8.17 |
| Torque Constant[3,4] | $K_t$ (trap) | oz-in/Amp DC | 11.19 | 5.81 | 20.82 | 11.04 |
| | | N-m/Amp DC | 0.078 | 0.041 | 0.146 | 0.077 |
| Resistance[3] | R | ohms | 4.53 | 1.24 | 6.50 | 1.73 |
| Inductance[5] | L | millihenries | 0.81 | 0.21 | 1.39 | 0.33 |
| Thermal Resistance | $R_{th}$ w-a | °C/watt | 2.70 | 2.70 | 2.00 | 2.00 |
| Motor Constant | Km | oz-in/√watt | 5.26 | 5.21 | 8.16 | 8.40 |
| | | N-m/√watt | 0.037 | 0.036 | 0.057 | 0.059 |
| Viscous Damping | B | oz-in/Krpm | 0.284 | 0.284 | 0.300 | 0.300 |
| | | N-m/Krpm | 1.99E-3 | 1.99E-3 | 2.1E-3 | 2.1E-3 |
| Static Friction | $T_f$ | oz-in | 0.15 | 0.15 | 0.20 | 0.20 |
| | | N-m | 1.05E-3 | 1.05E-3 | 1.40E-3 | 1.40E-3 |
| Thermal Time Constant | $\tau_{th}$ | minutes | 11.6 | 11.6 | 14.2 | 14.2 |
| Electrical Time Constant | $\tau_e$ | milliseconds | 0.18 | 0.17 | 0.21 | 0.19 |
| Mechanical Time Constant | $\tau_m$ | milliseconds | 7.7 | 7.8 | 5.5 | 5.2 |
| Rotor Inertia | J | lb-in-sec$^2$ | 9.4E-5 | 9.4E-5 | 1.6E-4 | 1.6E-4 |
| | | kg-m$^2$ | 1.1E-5 | 1.1E-5 | 1.8E-5 | 1.8E-5 |
| Weight | # | pounds | 1.1 | 1.1 | 1.6 | 1.6 |
| | kg | kilograms | 0.5 | 0.5 | 0.7 | 0.7 |
| Winding Class | | | H | H | H | H |

[1] @ 25°C ambient, 125°C winding temperature, motor connected to a 10"x10"x1/4" aluminum mounting plate, @ 40°C ambient, derate phase currents and torques by 12%.

[2] Maximum speed is 7500RPM with 500 line encoder. For 1000 line encoders, derate to 6000RPM. For higher speed operation, please call the factory.

[3] Measured line-to-line, ±10% line-to-line.

[4] Value is measured peak of sine wave.

[5] ±30% line-to-line, inductance bridge measurement @ 1 kHz.

[6] Initial winding temperature must be 60°C or less before peak current is applied.

**Note: These specifications are based on theoretical motor performance and are not specific to any amplifier.**

# Motor Specifications: SM230 and SM231

| Parameter | Symbol | Units | SM230A | SM230B | SM231A | SM231B |
|---|---|---|---|---|---|---|
| Stall Torque Continuous[1] | $T_{cs}$ | lb-in/oz-in | 1.7/27 | 1.6/26 | 3.8/61 | 3.4/54 |
| | | N-m | 0.19 | 0.18 | 0.43 | 0.38 |
| Stall Current Continuous[1] | $I_{cs}$ (trap) | Amps DC | 2.4 | 4.7 | 2.5 | 4.8 |
| Rated Speed[2] | $\omega_r$ | rpm | 7500 | 7500 | 7,500 | 7,500 |
| Peak Torque[6] | $T_{pk}$ | lb-in/oz-in | 5.1/82 | 4.9/78 | 11.3/181 | 10.0/160 |
| | | N-m | 0.57 | 0.55 | 1.27 | 1.12 |
| Peak Current[6] | $I_{pk}$ (trap) | Amps DC | 7.1 | 14.2 | 7.6 | 14.3 |
| Torque @ Rated Speed | $T_r$ | lb-in/oz-in | 1.4/22 | 1.3/21 | 2.9/47 | 2.8/44 |
| | | N-m | 0.15 | 0.15 | 0.33 | 0.31 |
| Rated Power–Output Shaft | $P_o$ | watts | 122 | 116 | 261 | 244 |
| Voltage Constant[3,4] | $K_b$ | volts/radian/sec | 0.081 | 0.039 | 0.169 | 0.079 |
| Voltage Constant[3,4] | $K_e$ | volts/KRPM | 8.48 | 4.09 | 17.70 | 8.27 |
| Torque Constant[3,4] | $K_t$ (trap) | oz-in/Amps DC | 11.47 | 5.54 | 23.93 | 11.19 |
| | | N-m/Amps DC | 0.080 | 0.039 | 0.168 | 0.078 |
| Resistance[3] | R | ohms | 4.43 | 1.12 | 5.22 | 1.46 |
| Inductance[5] | L | millihenries | 1.19 | 0.28 | 1.64 | 0.44 |
| Thermal Resistance | $R_{th}$ w-a | °C/watt | 2.67 | 2.67 | 2.00 | 2.00 |
| Motor Constant | Km | oz-in/√watt | 5.45 | 5.23 | 10.47 | 9.26 |
| | | N-m/√watt | 0.038 | 0.037 | 0.073 | 0.065 |
| Viscous Damping | B | oz-in/Krpm | 0.160 | 0.160 | 0.250 | 0.250 |
| | | N-m/Krpm | 1.12E-3 | 1.12E-3 | 1.75E-3 | 1.75E-3 |
| Static Friction | $T_f$ | oz-in | 0.20 | 0.20 | 0.30 | 0.30 |
| | | N-m | 1.4E-3 | 1.4E-3 | 2.10E-3 | 2.10E-3 |
| Thermal Time Constant | $\tau_{th}$ | minutes | 18.3 | 18.3 | 20 | 20 |
| Electrical Time Constant | $\tau_e$ | milliseconds | 0.27 | 0.25 | 0.31 | 0.30 |
| Mechanical Time Constant | $\tau_m$ | milliseconds | 18.3 | 19.9 | 9.5 | 12.2 |
| Rotor Inertia | J | lb-in-sec$^2$ | 2.4E-4 | 2.4E-4 | 4.6E-4 | 4.6E-4 |
| | | kg-m$^2$ | 2.7E-5 | 2.7E-5 | 5.2E-5 | 5.2E-5 |
| Weight | # | pounds | 1.2 | 1.2 | 2.1 | 2.1 |
| | kg | kilograms | 0.5 | 0.5 | 1.0 | 1.0 |
| Winding Class | | | H | H | H | H |

[1] @ 25°C ambient, 125°C winding temperature, motor connected to a 10"x10"x1/4" aluminum mounting plate, @ 40°C ambient derate phase currents and torques by 12%.

[2] Maximum speed is 7500RPM with 500 line encoder. For 1000 line encoders, derate to 6000RPM. For higher speed operation, please call the factory.

[3] Measure line-to-line, ±10% line-to-line.

[4] Value is measured peak of sine wave.

[5] ±30%, line-to-line, inductance bridge measurement @ 1 kHz.

[6] Initial winding temperature must be 60°C or less before peak current is applied.

**Note: These specifications are based on theoretical motor performance and are not specific to any amplifier.**

# Motor Specifications: SM232 and SM233

| Parameter | Symbol | Units | SM232A | SM232B | SM233A | SM233B |
|---|---|---|---|---|---|---|
| Stall Torque Continuous[1] | $T_{cs}$ | lb-in/oz-in | 6.6/106 | 7.0/111 | 10.1/161 | 9.7/156 |
| | | N-m | 0.74 | 0.78 | 1.13 | 1.09 |
| Stall Current Continuous[1] | $I_{cs}$ (trap) | Amps DC | 2.4 | 4.7 | 2.4 | 4.5 |
| Rated Speed[2] | $\omega_r$ | rpm | 7,500 | 7,500 | 5,800 | 5,800 |
| Peak Torque[6] | $T_{pk}$ | lb-in/oz-in | 19.8/316 | 20.9/334 | 30.2/483 | 29.2/467 |
| | | N-m | 2.21 | 2.34 | 3.38 | 3.27 |
| Peak Current[6] | $I_{pk}$ (trap) | Amps DC | 7.2 | 14.0 | 7.1 | 13.6 |
| Torque @ Rated Speed | $T_r$ | lb-in/oz-in | 5.1/81 | 5.4/86 | 8.1/129 | 7.6/121 |
| | | N-m | 0.57 | 0.60 | 0.90 | 0.85 |
| Rated Power–Output Shaft | $P_o$ | watts | 449 | 477 | 553 | 519 |
| Voltage Constant[3,4] | $K_b$ | volts/radian/sec | 0.310 | 0.169 | 0.484 | 0.242 |
| Voltage Constant[3,4] | $K_e$ | volts/KRPM | 32.46 | 17.70 | 50.68 | 25.34 |
| Torque Constant[3,4] | $K_t$ (trap) | oz-in/Amp DC | 43.90 | 23.93 | 68.53 | 34.27 |
| | | N-m/Amp DCs | 0.307 | 0.168 | 0.480 | 0.240 |
| Resistance[3] | R | ohms | 7.50 | 2.00 | 9.65 | 2.58 |
| Inductance[5] | L | millihenries | 2.90 | 0.78 | 4.08 | 1.06 |
| Thermal Resistance | $R_{th}$ w-a | °C/watt | 1.54 | 1.54 | 1.25 | 1.25 |
| Motor Constant | Km | oz-in/√watt | 16.03 | 16.92 | 22.06 | 21.33 |
| | | N-m/√watt | 0.112 | 0.118 | 0.154 | 0.149 |
| Viscous Damping | B | oz-in/Krpm | 0.360 | 0.360 | 0.540 | 0.540 |
| | | N-m/Krpm | 2.52E-3 | 2.52E-3 | 3.78E-3 | 3.78E-3 |
| Static Friction | $T_f$ | oz-in | 0.70 | 0.70 | 1.00 | 1.00 |
| | | N-m | 4.90E-3 | 4.90E-3 | 7.00E-3 | 7.00E-3 |
| Thermal Time Constant | $\tau_{th}$ | minutes | 21.6 | 21.6 | 23.3 | 23.3 |
| Electrical Time Constant | $\tau_e$ | milliseconds | 0.39 | 0.39 | 0.42 | 0.41 |
| Mechanical Time Constant | $\tau_m$ | milliseconds | 7.2 | 6.5 | 5.4 | 5.8 |
| Rotor Inertia | J | lb-in-sec$^2$ | 8.2E-4 | 8.2E-4 | 1.2E-3 | 1.2E-3 |
| | | kg-m$^2$ | 9.3E-5 | 9.3E-5 | 1.3E-4 | 1.3E-4 |
| Weight | # | pounds | 3.0 | 3.0 | 3.9 | 3.9 |
| | kg | kilograms | 1.4 | 1.4 | 1.8 | 1.8 |
| Winding Class | | | H | H | H | H |

[1] @ 25°C ambient, 125°C winding temperature, motor connected to a 10"x10"x1/4" aluminum mounting plate, @ 40°C ambient, derate phase currents and torques by 12%.

[2] Maximum speed is 7500RPM with 500 line encoder. For 1000 line encoders, derate to 6000RPM. For higher speed operation, please contact factory.

[3] Measured line-to-line, ±10% line-to-line.

[4] Value is measured peak of sine wave.

[5] ±30%, line-to-line, inductance bridge measurement @ 1 kHz.

[6] Initial winding temperature must be 60°C or less before peak current is applied.

**Note: These specifications are based on theoretical motor performance and are not specific to any amplifier.**

# Motor Specifications: NeoMetric & J Series

| Parameter | Symbol | Units | N0701 | | N0702 | |
|---|---|---|---|---|---|---|
| | | | or N0341 | | or N0342 | |
| Winding Selection | | | D | F | E | F |
| Stall Torque Continuous[1] | $T_{cs}$ | lb-in | 5.7 | 5.6 | 10.4 | 10.4 |
| | | N-m | 0.63 | 0.63 | 1.17 | 1.16 |
| Stall Current Continuous[1] | $I_{cs}$ (trap) | Amps DC | 2.9 | 4.5 | 3.3 | 4.6 |
| Rated Speed[2] | $\omega_r$ | rpm | 7500 | 7500 | 7500 | 7500 |
| Peak Torque[6] | $T_{pk}$ | lb-in | 17.0 | 16.8 | 31.2 | 31.1 |
| | | N-m | 1.90 | 1.88 | 3.50 | 3.49 |
| Peak Current[6] | $I_{pk}$ (trap) | Amps DC | 8.7 | 13.5 | 10.0 | 13.9 |
| Torque @ Rated Speed | $T_r$ | lb-in | 4.7 | 4.6 | 7.1 | 7.9 |
| | | N-m | 0.53 | 0.52 | 0.80 | 0.88 |
| Rated Power–Output Shaft | $P_o$ | watts | 416 | 411 | 632 | 699 |
| Voltage Constant[3, 4] | $K_b$ | volts/radian/sec | 0.221 | 0.140 | 0.353 | 0.253 |
| Voltage Constant[3, 4] | $K_e$ | volts/KRPM | 23.14 | 14.66 | 36.97 | 26.49 |
| Torque Constant[3, 4] | $K_t$ (trap) | oz-in/Amp DC | 31.29 | 19.82 | 49.98 | 35.82 |
| | | N-m/Amp DC | 0.219 | 0.139 | 0.350 | 0.251 |
| Resistance[3] | R | ohms | 5.52 | 2.27 | 5.22 | 2.70 |
| Inductance[5] | L | millihenries | 12.98 | 5.23 | 15.80 | 8.16 |
| Thermal Resistance | $R_{th}$ w-a | °C/watt | 1.44 | 1.44 | 1.15 | 1.15 |
| Motor Constant | Km | oz-in/√watt | 13.32 | 13.16 | 21.88 | 21.80 |
| | | N-m/√watt | 0.093 | 0.092 | 0.153 | 0.153 |
| Viscous Damping | B | oz-in/Krpm | 0.2 | 0.2 | 0.4 | 0.4 |
| | | N-m/Krpm | 1.4E-3 | 1.4E-3 | 2.8E-3 | 2.8E-3 |
| Static Friction | $T_f$ | oz-in | 0.8 | 0.8 | 1.6 | 1.6 |
| | | N-m | 5.6E-3 | 5.6E-3 | 1.2E-2 | 1.2E-2 |
| Thermal Time Constant | $\tau_{th}$ | minutes | 16.6 | 16.6 | 21.7 | 21.7 |
| Electrical Time Constant | $\tau_e$ | milliseconds | 2.35 | 2.30 | 3.03 | 3.02 |
| NeoMetric Mech. Time Const. | $\tau_{mch}$ | milliseconds | 1.6 | 1.7 | 0.6 | 0.6 |
| J Series Mech. Time Const. | $\tau_{mch}$ | milliseconds | 14.7 | 14.7 | 5.7 | 5.7 |
| NeoMetric Rotor Inertia | J | lb-in-sec$^2$ | 1.1E-4 | 1.1E-4 | 1.7E-4 | 1.7E-4 |
| | | kg-m$^2$ | 1.2E-5 | 1.2E-5 | 2.0E-5 | 2.0E-5 |
| J Series Rotor Inertia | J | lb-in-sec$^2$ | 1.1E-3 | 1.1E-3 | 1.2E-3 | 1.2E-3 |
| | | kg-m$^2$ | 1.3E-4 | 1.3E-4 | 1.4E-4 | 1.4E-4 |
| NeoMetric Weight | # | pounds | 3.5 | 3.5 | 4.5 | 4.5 |
| | kg | kilograms | 1.6 | 1.6 | 2.1 | 2.1 |
| J Series Weight | # | pounds | 4.4 | 4.4 | 5.4 | 5.4 |
| | kg | kilograms | 2.0 | 2.0 | 2.5 | 2.5 |
| Winding Class | | | | H | | H |

@ 25°C ambient, 125°C winding temperature, motor connected to a 10"x10"x1/4" aluminum mounting plate, @ 40°C ambient, derate phase currents and torques by 12%.

[2] Maximum speed is 7500RPM with 500 line encoder. For 1000 line encoders, derate to 6000RPM. For higher speed operation, please contact factory.

[3] Measured line-to-line, ±10% line-to-line.

[4] Value is measured peak of sine wave.

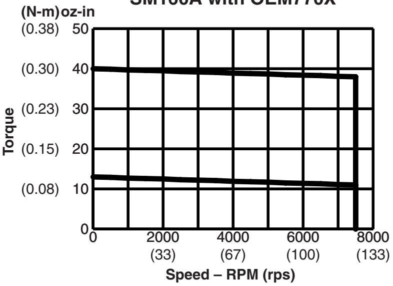
[5] ±30%, line-to-line, inductance bridge measurement @ 1 kHz.

[6] Initial winding temperature must be 60°C or less before peak current is applied.

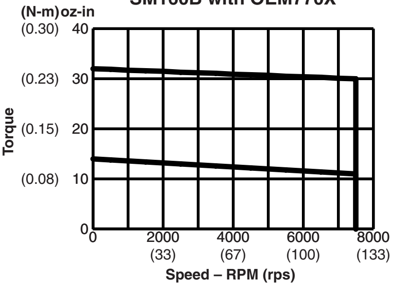
**Note: These specifications are based on theoretical motor performance and are not specific to any amplifier.**

# Speed/Torque Curves[2]: SM 160, SM161 and SM162

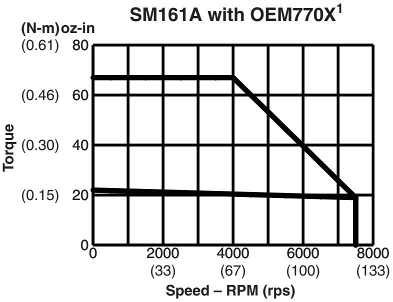
SM160A with OEM770X[1]



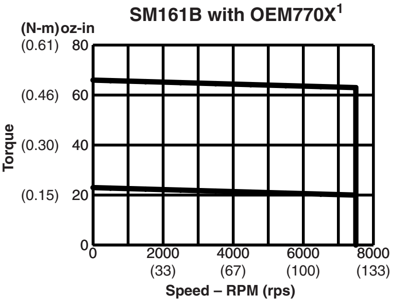
SM160B with OEM770X[1]



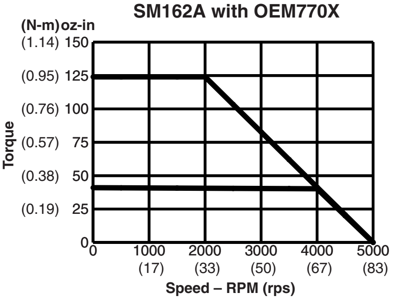
SM161A with OEM770X[1]



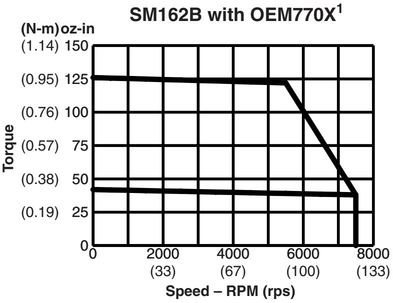
SM161B with OEM770X[1]



SM162A with OEM770X



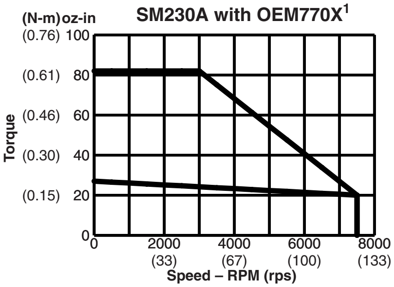
SM162B with OEM770X[1]

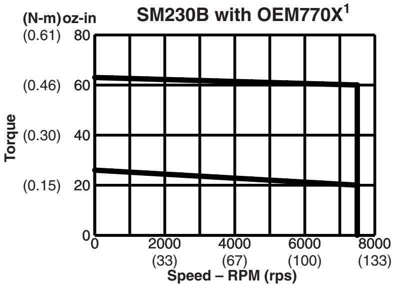


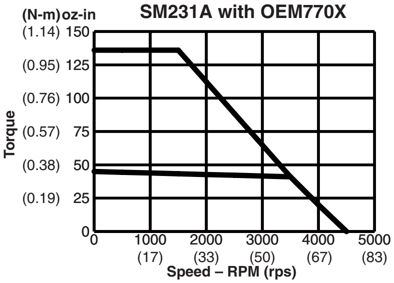
[1] For "E" encoder option (1000 ppr), maximum velocity is 6,000 rpm (100 rps).

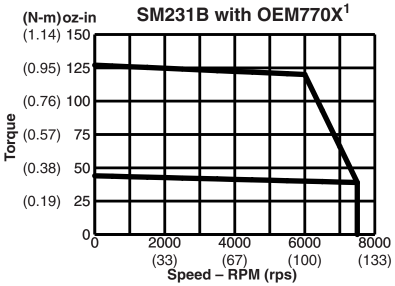
[2] With 75VDC bus voltage; 25°C (77°F) ambient temperature.

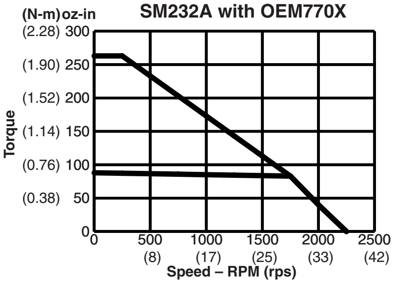
# Speed/Torque Curves[2]: SM230, SM231, SM232, SM233

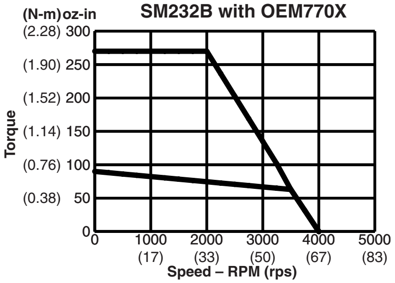

**SM230A with OEM770X[1]**

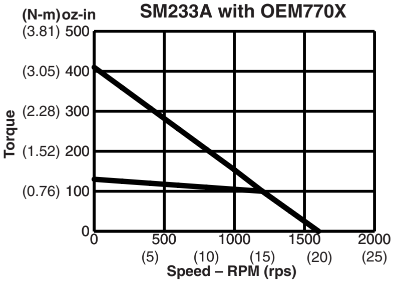

**SM230B with OEM770X[1]**

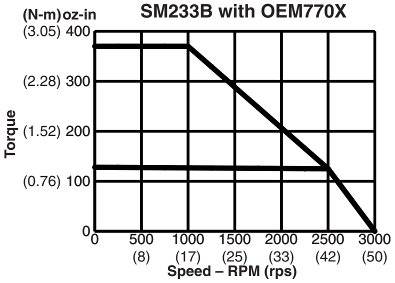

**SM231A with OEM770X**


**SM231B with OEM770X[1]**


**SM232A with OEM770X**


**SM232B with OEM770X**


**SM233A with OEM770X**

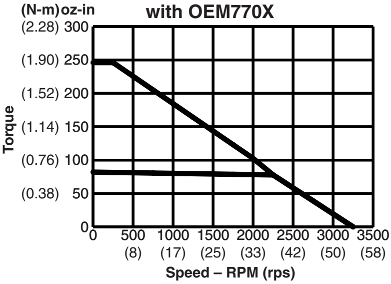

**SM233B with OEM770X**

[1] For "E" encoder option (1000 ppr), maximum velocity is 6,000 rpm (100 rps).

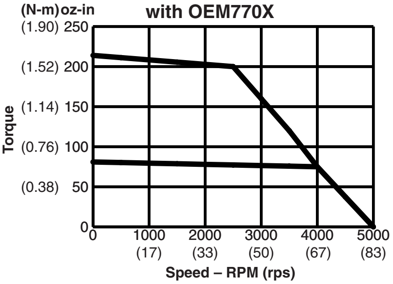
[2] With 75VDC bus voltage; 25°C (77°F) ambient temperature.

# Speed/Torque Curves[1]: NeoMetric & J Series Motors

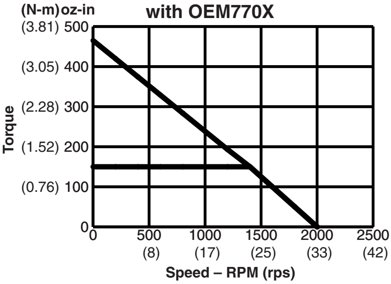
NO701D/NO341D/JO701D/JO341D with OEM770X



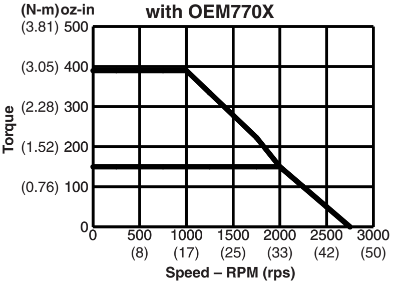
NO701F/NO341F/JO701F/JO341F with OEM770X



NO702E/NO342E/JO702E/JO342E with OEM770X



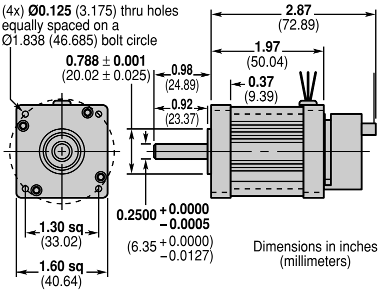
NO702F/NO342F/JO702F/JO342F with OEM770X



[1] With 75VDC bus voltage; 25°C (77°F) ambient temperature.

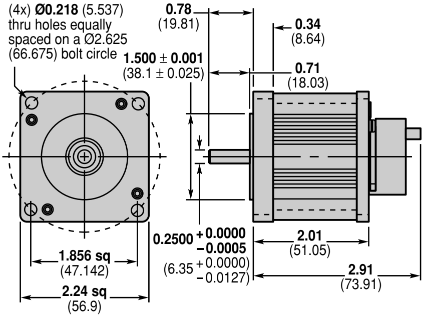
# Motor Dimensions: Compumotor SM160 and SM230

## SM160

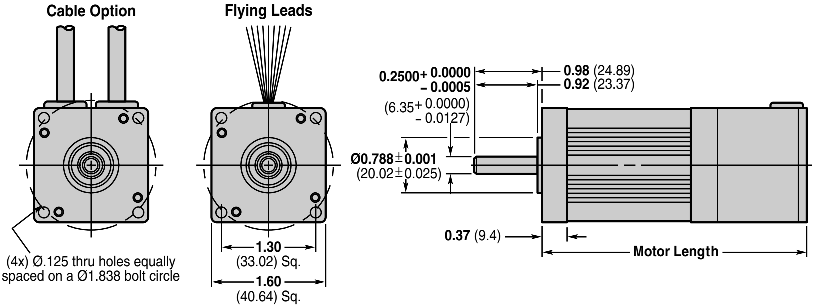


Dimensions in inches (millimeters)

## SM230



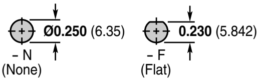
# Motor Dimensions: Compumotor SM Series, Size 16



| Cable Options | |
|---|---|
| Part # | Description |
| – FL | 18" Flying Leads |
| – 10 | 10 ft. Cable |

Longer lengths available
Consult Compumotor for information

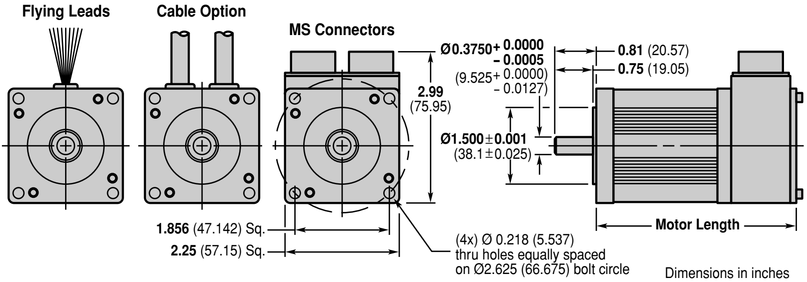
Shaft Options



Dimensions in inches
(millimeters)

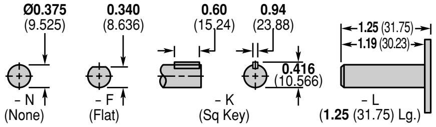
| Motor Sizes | |
|---|---|
| Motor Length | Model |
| 4.79 (121.66) | 162 Motor |
| 3.79 (96.27) | 161 Motor |

# Motor Dimensions: Compumotor SM Series, Size 23



Dimensions in inches
(millimeters)

Shaft Options

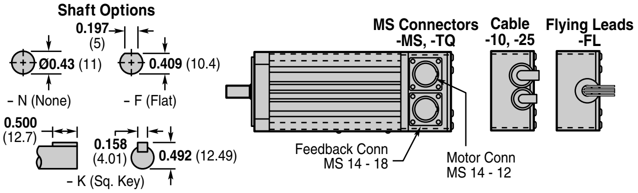


| Cable Options | |
|---|---|
| Part # | Description |
| – FL | 18" Flying Leads |
| – 10 | 10 ft. Cable |

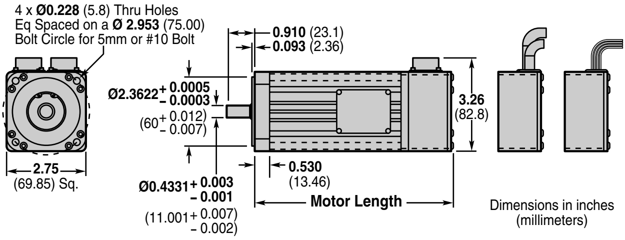
Longer lengths available
Consult Compumotor
for information

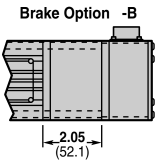
| Motor Sizes | |
|---|---|
| Motor Length | Model |
| 5.98 (151.89) | 233 Motor |
| 4.98 (126.49) | 232 Motor |
| 3.98 (101.09) | 231 Motor |

# Motor Dimensions: NeoMetric & J Series, Size 70

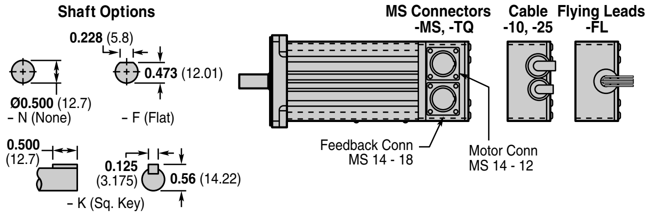


| Motor Sizes | |
|---|---|
| Length | Model |
| 4.94 (125.5) | NO701 |
| 5.94 (150.9) | NO702 |
| 5.94 (150.9) | JO701 |
| 6.94 (176.3) | JO702 |
| 7.00 (177.8) | NO701 w/Brake |
| 8.00 (203.2) | NO702 w/Brake |
| 8.00 (203.2) | JO701 w/Brake |
| 9.00 (228.6) | JO702 w/Brake |

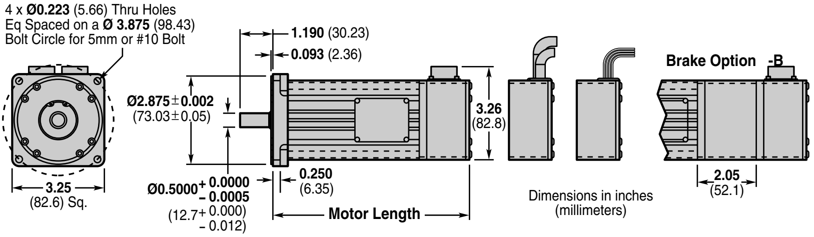




# Motor Dimensions: NeoMetric & J Series, Size 34



| Motor Sizes | |
|---|---|
| Length | Model |
| 4.94 (125.5) | NO341 |
| 5.94 (150.9) | NO342 |
| 5.94 (150.9) | JO341 |
| 6.94 (176.3) | JO342 |
| 7.00 (177.8) | NO341 w/Brake |
| 8.00 (203.2) | NO342 w/Brake |
| 8.00 (203.2) | JO341 w/Brake |
| 9.00 (228.6) | JO3422 w/Brake |


4 x Ø0.223 (5.66) Thru Holes
Eq Spaced on a Ø 3.875 (98.43)
Bolt Circle for 5mm or #10 Bolt
1.190 (30.23)
0.093 (2.36)
Brake Option -B
Ø2.875±0.002 (73.03±0.05)
3.26 (82.8)
3.25 (82.6) Sq.
Ø0.5000+ 0.0000 - 0.0005
(12.7+ 0.000) - 0.012)
0.250 (6.35)
Motor Length
Dimensions in inches (millimeters)
2.05 (52.1)

## Encoder Specifications

The same type of encoder is used on all SM and NeoMetric Series motors. Encoders have either 500 lines ("-D") or 1000 lines ("-E).

| | |
|---|---|
| **Mechanical** | |
| Accuracy | ±2 min of arc |
| **Electrical** | |
| Input power | 5 VDC ±5%, 135 mA |
| Operating frequency | 100 kHz max |
| Output device | 26LS31 |
| Sink/Source, nominal | 20 mA |
| Suggested user interface | 26LS32 |

## Hall Effect Specifications

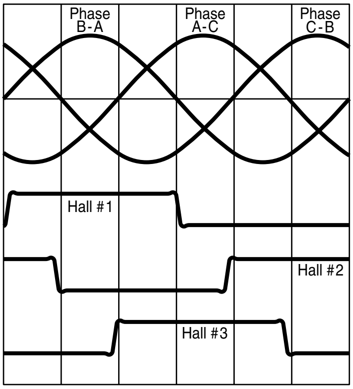
Specifications for Hall effect outputs on SM and NeoMetric Series motors are listed below.

| | |
|---|---|
| **Electrical** | |
| Input power | 5 VDC ±5%, 80 mA |
| Output device, open collector | LM339 |
| Maximum pull up | 12 VDC |
| Sink | 16 mA |

### COMMUTATION CHART

This chart shows the relationship between motor back EMF and Hall state.

Clockwise rotation as viewed from front shaft.



# Motor Wiring Information

## SM MOTORS – SIZE 16 AND SIZE 23

### Motor Phase

| | | | | -FL Option<br>-10 Option<br>-25 Option |
|---|---|---|---|---|
| | -MS Option | -TQ Option | -H Option | |
| | Pin No. | Pin No. | Pin No. | Wire |
| **Designation** | **MS14-12** | **MS14-12** | **MS14-12** | **Color** |
| Phase A | J | J | J | Red/Yellow |
| Phase B | K | K | K | White/Yellow |
| Phase C | L | L | L | Black/Yellow |
| Ground | M | M | M | Green/Yellow |
| Shield | NC | NC | NC | — |
| Temp | G | G | G | Orange/Yellow or Yellow |
| Temp | H | H | H | Orange/Yellow or Yellow |

### Encoder

| | Pin No. | Pin No. | Not | Wire |
|---|---|---|---|---|
| **Designation** | **MS14-18** | **MS14-18** | **Applicable** | **Color** |
| Vcc | H | H | — | Red |
| Ground | G | G | — | Black |
| CH A+ | A | A | — | White |
| CH A- | B | B | — | Yellow |
| CH B+ | C | C | — | Green |
| CH B- | D | D | — | Blue |
| Index + | E | E | — | Orange |
| Index - | F | F | — | Brown |
| Shield | NC | NC | — | — |

### Hall-effect

| | Pin No. | Pin No. | Pin No. | Wire |
|---|---|---|---|---|
| **Designation** | **MS14-18** | **MS14-12** | **MS14-12** | **Color** |
| Hall GND | K | F | F | White/Green |
| Hall +5 | M | B | B | White/Blue |
| Hall 1 | T | C | C | White/Brown |
| Hall 2 | U | D | D | White/Orange |
| Hall 3 | P | E | E | White/Violet |

Wiring color is provided for flying lead or cable versions.

## NeoMetric & J Series Motors – Size 070 (Size 034)

### Motor Phase

| Designation | Pin No.<br>MS14-12 | Wire Color |
|---|---|---|
| Phase A | J | Red/Yellow |
| Phase B | K | White/Yellow |
| Phase C | L | Black/Yellow |
| Ground | M | Green/Yellow |
| Shield | NC | — |
| Continue for “H” or “TQ” Options | | |
| Temp | G | Orange/Yellow or Yellow |
| Temp | H | Orange/Yellow or Yellow |
| Hall GND | F | White/Green |
| Hall +5 | B | White/Blue |
| Hall 1 | C | White/Brown |
| Hall 2 | D | White/Orange |
| Hall 3 | E | White/Violet |

### Encoder/Commutation Connections

| Designation | Pin No.<br>MS14-18 | Wire Color |
|---|---|---|
| Encoder | | |
| +5 VDC | H | Red |
| Ground | G | Black |
| CH A+ | A | White |
| CH A- | B | Yellow |
| CH B+ | C | Green |
| CH B- | D | Blue |
| Index + | E | Orange |
| Index - | F | Brown |
| Commutation | | |
| Hall GND | K | White/Green |
| Hall +5 | M | White/Blue |
| Hall 1 | T | White/Brown |
| Hall 2 | U | White/Orange |
| Hall 3 | P | White/Violet |
| Temp | L | Orange/Yellow or Yellow |
| Temp | N | Orange/Yellow or Yellow |
| Brake Option | | |
| Brake[1] | R | Red/Blue |
| Brake[1] | S | Red/Blue |

[1] Brake will operate regardless of polarity of connection.

# CHAPTER 4

## *Special Internal Circuits*

The OEM770X has several internal circuits that can protect the drive, protect equipment connected to the drive, or change the drive's performance characteristics.

Four of the built-in circuits work automatically. Their performance cannot be changed or altered.

- Short Circuit Protection
- Undervoltage
- Overvoltage
- Overtemperature

Two of the circuits use removable resistors in sockets. You can change these resistors to alter the circuit parameters.

- Response Circuit
- Current Foldback Circuit

This chapter explains the performance of these six circuits.

## Short Circuit Protection

The OEM770X continuously monitors the current it sends to the motor. If it detects excessive current, it interprets the excessive current as a short circuit fault in the motor or cabling. The OEM770X then disables its power output to the motor terminals—Phase A, Phase B, and Phase C.

To show that a short circuit fault has occurred, the drive illuminates the red LED, turns off the green LED, and causes the fault output (pin 9) to go high. Other power outputs—Hall +5, Encoder +5V—remain on.

The short circuit fault is a *latched* condition. Latched means that the output will remain off until power is cycled. To *cycle power*, turn off the power to the drive, wait approximately 30 seconds, then turn on the power.

The other power outputs (Hall +5, Encoder +5V) are also short circuit protected.

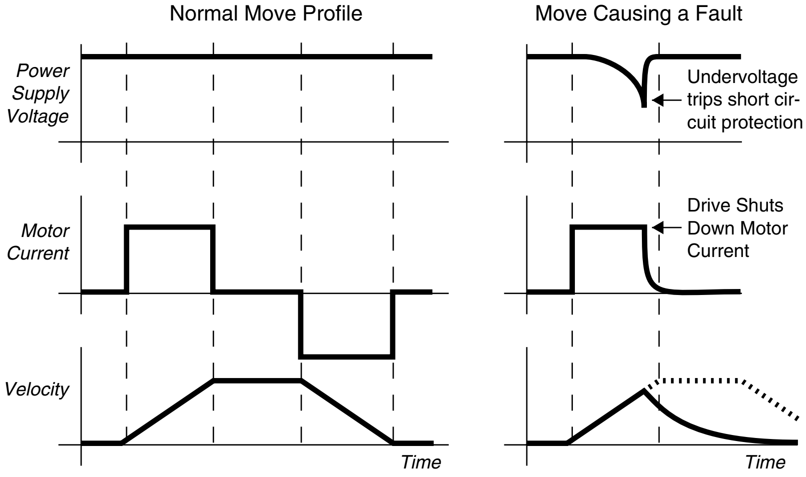
Short circuit protection features are summarized below.

| | |
|---|---|
| • Power to motor is turned OFF | |
| • Red LED is turned ON | (Illuminated) |
| • Green LED is turned OFF | (Not Illuminated) |
| • Fault output goes HIGH | |
| • Latched | |
| • Hall +5, Encoder +5V remain powered | |
| • Hall +5VDC, Encoder +5V are also short circuit protected | |

(*Troubleshooting Note:* Other faults will also turn on the red LED, but they leave the green LED illuminated. Short circuit protection is the only fault that will turn off the green LED when it turns on the red LED.)

A short circuit fault is not the only event that can trigger this circuit. A power supply fault can also trigger short circuit protection. The fault can occur if the supply is undersized, and cannot provide enough power during demanding move profiles.

The next drawing shows graphs for motor current and power supply voltage during a normal move profile.



*Power Supply Fault*

The drawing also shows what happens to voltage if the power supply is inadequate. During the first part of the move, energy stored in the power supply (in the capacitors, for example) can provide power for the move. As this energy is used up, the power supply cannot replenish it fast enough, and the voltage drops. If the voltage gets too low, short circuit protection is turned on, and shuts down motor current. At this point the power supply no longer needs to provide power to the drive. It can now direct power into its own capacitors. They recharge, and the supply voltage quickly returns to normal levels.

This is a transient event. Without short circuit protection, it may go undetected. Your system's performance could be less than you expected, and you might not know why. Short circuit protection latches the drive off during the transient event, however. This allows you to realize there is a problem, and find the cause. Once you determine there is no short circuit in your motor or cabling, you can inspect your power supply.

If your system runs while the motor is stopped or turning slowly, but faults during demanding move cycles, then your power supply may be causing the fault because it is inadequate for the task. Consider using a larger power supply, or altering your move profile so that the move requires less power.

The same condition—a momentary power supply fault—can sometimes turn on the undervoltage circuit, rather than short circuit protection. The undervoltage circuit is explained in the next section.

There are two potential warning signals, then, to alert you about power supply problems. Short circuit protection will latch, and shut down the drive. Undervoltage protection will momentarily turn on the red LED (but not turn off the green LED), and will not latch.

## Undervoltage

The undervoltage circuit monitors power supply voltage. If the voltage falls below a threshold level—21.5VDC or less—the undervoltage circuit will illuminate the red LED, and cause the fault output (pin 9) to go high. The green LED remains illuminated.

This condition *is not latched*. If the power supply voltage rises above the threshold, the red LED turns off, and the fault output goes low.

Undervoltage circuit features are summarized below.

- 21.5VDC threshold (Maximum)
- Red LED is turned ON (Illuminated)
- Green LED stays ON (Illuminated)
- Fault output goes HIGH
- Not latched

The undervoltage circuit ensures an orderly startup and shutdown process. During startup, when the power supply's voltage is rising, the undervoltage circuit will not allow the drive to turn on until the voltage rises above the threshold, and there is enough power to maintain the drive's circuits. During shutdown, when the power supply voltage falls below the threshold, the circuit will turn off the drive's circuits in an orderly and systematic manner. You may see the red LED come on briefly when the drive is turned on or off. This is normal, and does not indicate a problem.

One problem situation—a power supply undervoltage fault—can trigger the undervoltage circuit. (See the power supply fault explanation above, under *Short Circuit Protection*.) An undervoltage fault can trigger either the undervoltage circuit or short circuit protection. Sometimes the undervoltage circuit will react first, and turn on the red LED and send the fault output high. At other times, short circuit protection will react first, and latch the drive off. Which circuit reacts first depends on the dynamics of the fault, and is not easily predictable.

The undervoltage circuit can help you diagnose power supply problems.

*Example*: You use a 24VDC power supply to power an OEM770X. During certain parts of the move, your system's performance is less than you expect, and you notice that the red LED flashes. The flashing LED indicates that either the drive is in current foldback, or that the power supply's voltage is too low. If you monitor the fault output and notice that Pin #9 goes high when the LED flashes, you can rule out foldback. Foldback does not make the fault output go high. The problem is a power supply undervoltage fault. Try a larger power supply, or a less demanding move profile.

## Overvoltage

The overvoltage circuit protects the drive from regeneration. The OEM770X monitors voltage at its motor output terminals—Phase A, Phase B, and Phase C. If the motor regenerates energy, and the voltage rises above a threshold level—95VDC ± 5VDC—the circuit will disable power output to the motor. This is a latched condition. You must cycle power to restart the drive.

The circuit also turns on the red LED, and activates the fault output. Other power outputs—Hall +5, Encoder +5V—remain on. The green LED also stays on.

Overvoltage circuit features are listed below.

- 95VDC ± 5VDC threshold
- Power to motor is turned OFF
- Red LED is turned ON (Illuminated)
- Green LED stays ON (Illuminated)
- Fault output goes HIGH
- Latched
- Motor freewheels to a stop
- Does not protect against power supply overvoltage

After an overvoltage fault, the drive does nothing to stop the motor. When it stops receiving current, the motor will freewheel to a stop. If you have components that could be damaged by a freewheeling motor, consider using an external brake.

For example, in a system that raises and lowers a load, regeneration may occur while the load is being lowered. If the regeneration exceeds the 90VDC threshold and the overvoltage circuit shuts down motor current, the motor might freewheel, and the load could plunge to the floor. To avoid damage, a brake could be employed to stop the load in the event of a sudden loss of motor torque.

***CAUTION***

The overvoltage protection circuit can shut down current to the motor. This can cause a sudden and unexpected loss of motor torque. The motor will freewheel to a stop. Consider using a brake to arrest motion if your system regenerates energy.

Another possible concern is power supply overvoltage. The overvoltage circuit only monitors voltage at the output terminals to the motor. It does not monitor power supply voltage. This means that the drive is not protected from a defective power supply that produces excessive voltage. To protect the drive in this situation, use a power supply with built-in overvoltage protection on its outputs, such as Compumotor's OEM300 Power Module.

## Overtemperature

The overtemperature circuit protects the OEM770X from damage due to overtemperature conditions. This circuit monitors the temperature of the drive's heatplate. A temperature rise above 50°C (122°F) will cause an overtemperature fault. The protection circuit will disable power output to the motor, turn on the red LED, and activate the fault output. This is a latched condition. Other power outputs—Hall +5, Encoder +5V—remain on.

Overtemperature circuit features are listed below.

- 55°C ± 5°C (131°F ± 9°F) threshold
- Power to motor is turned OFF
- Red LED is turned ON (Illuminated)
- Green LED stays ON (Illuminated)
- Fault output goes HIGH
- Latched
- Cool below 40°C (104°F) and cycle power to restart

The overtemperature protection circuit has built-in thermal hysteresis. This means that the OEM770X cannot operate again until it has had time to cool below approximately 40°C (104°F). Once it has cooled, you must cycle power to restart the drive.

***Design Tip***

Use 50°C (122°F) as the maximum heatplate temperature allowed for continuous operation of the drive. Because of manufacturing tolerances on circuit components, different OEM770X units will shut down at different temperatures in the 50°C to 60°C range (122°F to 140°F). For predictability, use 50°C (122°F) as the shutdown temperature.

***Troubleshooting Note***

An overtemperature fault is a sign that something is wrong with your installation. Typical causes of overtemperature faults are:

- Inadequate Ventilation (broken fan, blocked vent, etc.)
- Inadequate Heatsink (too small, missing, not cooled properly, etc.)
- Assembly mistakes (mounting screw not tight, poor thermal contact, etc.)

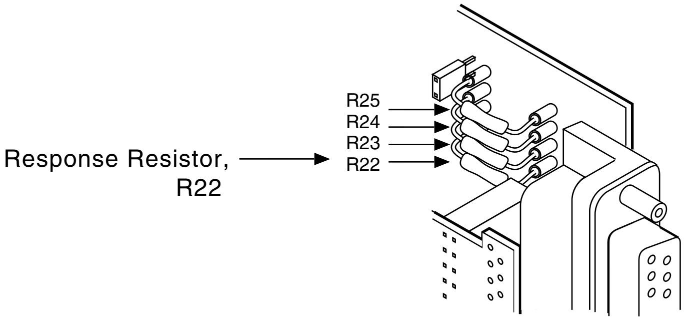
If your drive has an overtemperature fault, do not simply cool the drive, cycle power, and resume operations. Instead, find the problem that caused the fault, and fix the problem.

## Response Circuit

All servo motors are not the same! The inductance of different motors covers a wide range. When you select a motor for use with the OEM770X, its inductance affects the gain and frequency response of the current feedback loop, and thus the performance of your system.

To accommodate the wide range of motors that customers are likely to use, the drive has a response circuit that is adjustable. You can tailor the response circuit to match your motor. This can help you achieve optimum performance.

You can adjust the response circuit by changing the response resistor, R22, on the drive's circuit board.



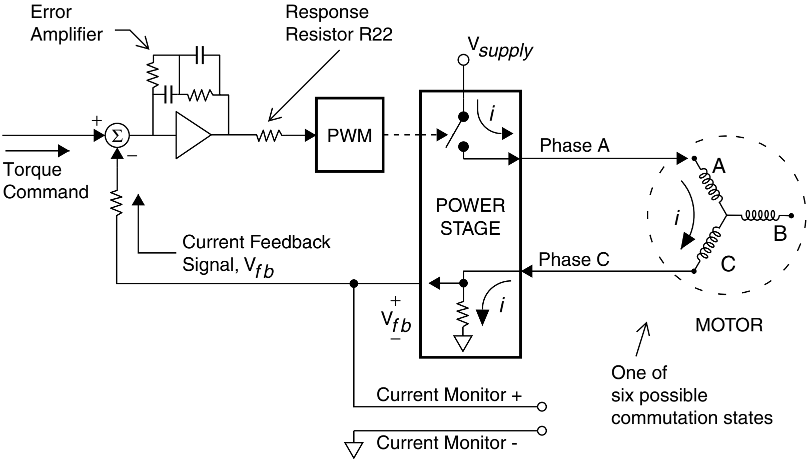
*Response Resistor Location*

See *Installing Selectable Resistors* in *Chapter 2 Installation* for instructions about installing a different response resistor, and for a list of resistors to use with Compumotor motors.

In the following sections, we will explain how the current feedback loop works, how motor inductance affects the loop, and how the response resistor can adjust drive performance to compensate for different motor inductances. Then, we will give detailed instructions for selecting a response resistor.

### CURRENT FEEDBACK LOOP

The following section of the OEM770X's block diagram shows the main components in the *current feedback loop*. This diagram shows the drive in one particular Hall state, with current flowing into phase A and out of phase C. Five other Hall states are possible. Their diagrams are similar.



*Current Feedback Loop*

The torque command is a signal that tells the drive how much current to produce. This desired current is called *commanded current*. It enters the loop through a summing node, where it is combined with a current feedback signal.

The feedback signal is a voltage that represents *actual current* flowing in the motor. The signal's polarity is adjusted so that it is inverted at the summing node. (Inverters and other components that accomplish this are not shown in the diagram.) This makes it a negative feedback signal. If actual current is identical to commanded current, the sum of the two signals will be zero.

If the two currents are not identical, the summing node will produce an *error signal*, which enters an *error amplifier*. This amplifier has very high gain at low frequencies, and will amplify even very small signals by a factor of thousands.

The amplified error signal next passes through the *response resistor*, which can change the level of the error signal, and thus modify the gain of the error amplifier. Higher resistor values will reduce the signal; lower values will increase it. (More information about selecting a response resistor will be given at the end of this section.)

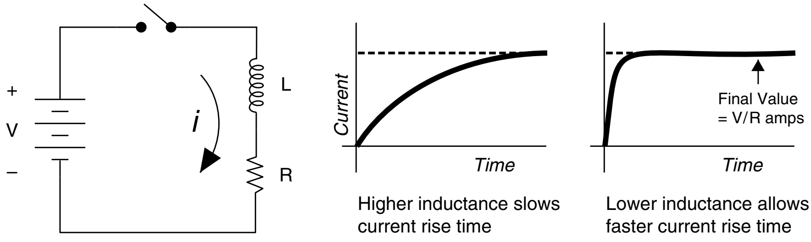
The error signal enters a *pulse width modulation* (PWM) circuit that controls the power stage. The drawing shows a simplified conceptual representation of how this control is accomplished. Voltage from a PWM pulse causes a switch to close. Current can then flow from an external power supply, through two coils of the motor, a sense resistor, and to ground. When the PWM pulse stops, the voltage controlled switch opens, which disconnects the power supply from the motor.

Together, the error amplifier, PWM circuit, and power stage form a voltage–to–current converter. A voltage that represents commanded current is converted to an actual current flowing in the motor. Longer PWM pulses will cause more current to flow; shorter pulses will cause less current to flow.

Notice that the motor current goes through a *sense resistor* before it reaches ground. The sense resistor is a current–to–voltage converter. Motor current flowing through it generates a voltage across the resistor. This voltage is proportional to actual current. It is used as the current feedback signal, $v_{fb}$, which is fed back to the summing node. This signal is also accessible to the user at the current monitor output.

## MOTOR INDUCTANCE AFFECTS FEEDBACK

So far, we have seen that there is motor inductance in the feedback loop, but we have not discussed its significance. To understand how inductance can affect a circuit, let us first look at a very simple circuit.



Higher inductance slows current rise time

Lower inductance allows faster current rise time

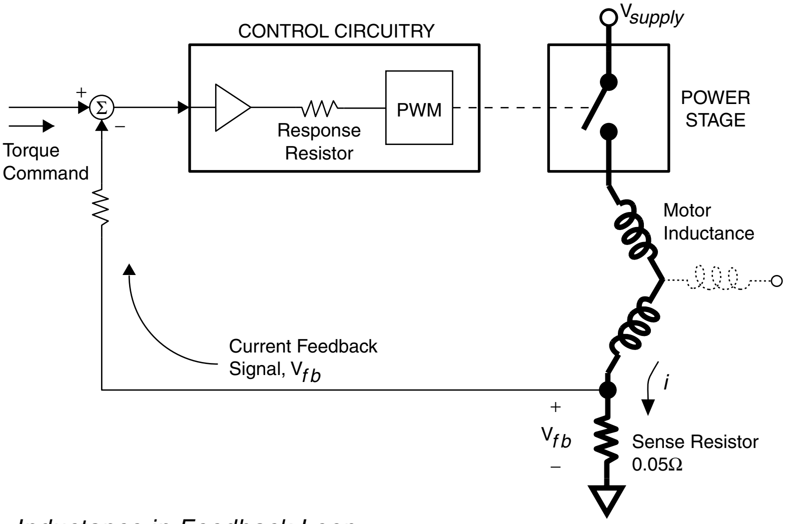
*Inductance Controls Rise Time*

This circuit consists of a battery, an inductor, a resistor, and a switch. When the switch is closed, current begins to flow in the circuit. The final value of the current depends on the voltage of the battery, V, and the size of the resistor, R. Its value is

$$I_{final} = V/R \text{ amps}$$

How long until the current reaches this final value? The rise time is determined by the size of the inductor. The inductor opposes the *change* in current flow. A large inductor will cause slow rise times. A small inductor will allow much faster rise times.

This circuit, although quite simple, is actually very similar to the OEM770X's current feedback loop, which is redrawn below.



*Inductance in Feedback Loop*

Compare this circuit to the simple circuit with a battery, switch, inductor and resistor. In this circuit, the battery has been replaced with a  power supply, the switch has been replaced by the power stage, the inductor has been replaced by the motor inductance, and the resistor has been replaced with a sense resistor (and motor and cabling resistance).

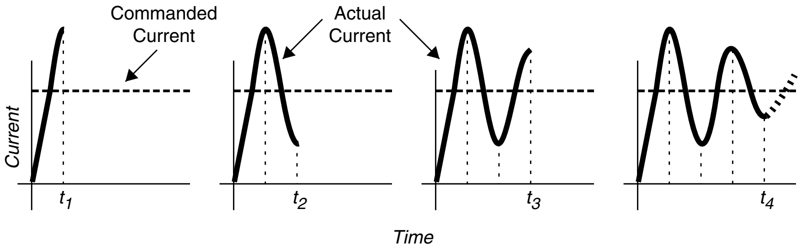
Most importantly, the switch is no longer controlled manually—it is now automatically controlled by a feedback loop. The most important control elements are shown together in the box labeled *Control Circuitry.*

In the feedback loop, commanded current is compared with actual current 20,000 times each second. After each comparison, the control circuit increases or decreases current flow by changing the width of PWM pulses. Feedback about results of the change is not instantaneous, however, because time delays are built into each step of the feedback loop. Each PWM setting is maintained for 50 microseconds, until the next comparison is made. At that time, the control circuit compares the feedback signal to the command signal, adjusts PWM pulses—and the whole process repeats.

How does motor inductance affect feedback and the current control process? We will consider several situations in which the only variable that changes is motor inductance. In each of the following examples, assume that the power supply voltage and error amplifier gain do not change.

### *Response with Low Inductance Motor*

The first drawing shows what can happen when the motor's inductance is low.

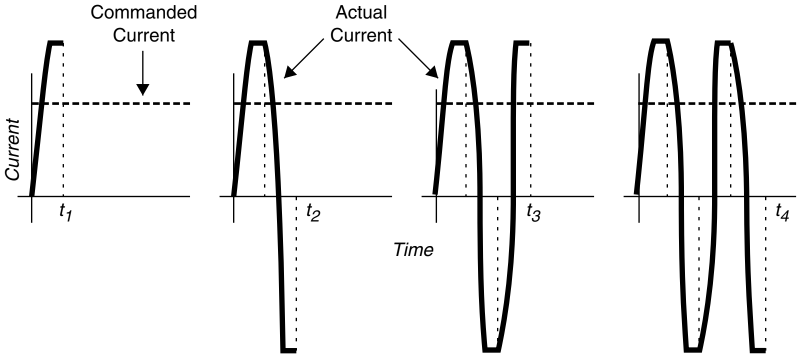


*Underdamped Response*

Recall that a low inductance permits a fast current rise. In this drawing, the system is given a commanded current. The drive compares actual current with commanded current, sees a large error, and directs the PWM circuit to produce maximum current. Motor inductance barely opposes current rise; because of the error amplifier's high gain, the current quickly rises to a level *higher* than commanded current.

At time $t_1$, the drive again compares actual with commanded current, and sees that actual current is too high. As a result, it *reduces* the power stage's current output. The change quickly results in an actual current that, at the next sample time $t_2$, is too low. Current is *increased*, and by the next sample point, time $t_3$, it is once again too high. Adjustments continue in this manner, and eventually the amount of actual current settles near the commanded current level.

This type of response is called an *underdamped* response. For a given loop gain and power supply voltage, the main component influencing this response is the inductance of the motor. If the inductance is *very* low, the system can oscillate, with actual current never settling near commanded current. The next drawing shows this case.

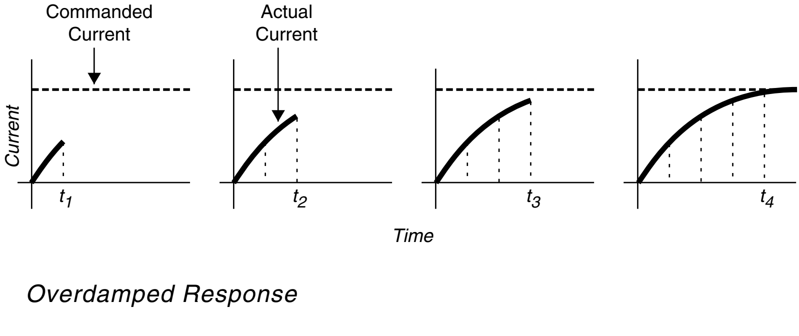


*Oscillating Response*

Here we see that current rise is so fast that the current output saturates at its maximum level before each successive sample. With 12A set as the maximum current, for example, actual current will oscillate between +12A and -12A. The motor will probably not turn—it can not respond as fast as the quickly changing currents—but it may become excessively hot due to the oscillating currents.

***Response with High Inductance Motor***

Next, we consider the effects that a high inductance motor has on the feedback loop. The drawing below shows the *overdamped* response that is typical with high inductance.

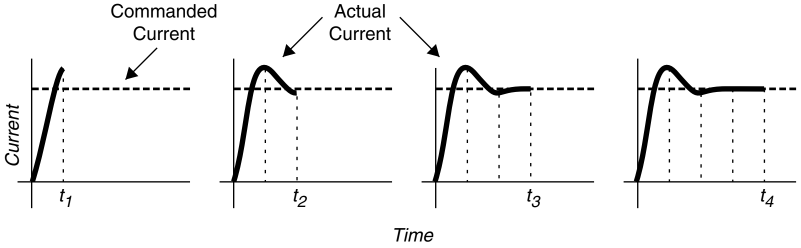


*Overdamped Response*

We see that actual current slowly rises to meet commanded current. The high inductance limits the current rise so much that by times $t_1$, $t_2$, and $t_3$, actual current is still too low. In overdamped situations, we can achieve very good control, with no overshoot—but the response time is very slow.

### *Optimum Response*

What type of response, then, is best? We want a fast current rise, so the system can quickly get to the commanded current level. But, the rise should not be so fast that the system repeatedly overshoots, and is underdamped. The next drawing shows an *optimum* response.



*Optimum Response*

In this example, the motor's inductance is well matched with the gain and timing of the current feedback loop. The inductance allows a fast current rise—but just fast enough so that, when the actual current level is rising past the commanded current level, it is time for the next sample. The control circuit compares commanded with actual current, and makes an adjustment. There is little overshoot, with a minimum settling time before actual current reaches commanded current.

If you change one component in this well-matched system—motor inductance, for example—you may need to adjust some other component to maintain the system's optimum response.

## SELECTING A RESPONSE RESISTOR

In the previous section, we discussed the effect different motors have on the drive's response. Once you have chosen a motor, the inductance in your system is fixed—it is no longer a variable. To adjust the response of your system for the motor you have chosen, you can install the correct response resistor.

If yours is a Compumotor motor, use the response resistor recommended for your motor in *Installing Selectable Resistors* in *Chapter 2 Installation*. If yours is a non-Compumotor motor, examine the motor specification tables for Compumotor motors in *Chapter 3 Specifications*; find a motor with inductance and resistance similar to yours, and use the resistor recommended for that motor. In either case, you may have to make further adjustments as described below.

Once you have chosen a resistor, there are three possibilities for what to do next, based upon the response of your system.

- OPTIMUM RESPONSE – Use the resistor you have chosen.
- UNDERDAMPED RESPONSE – Use a higher value than the resistor you have chosen.
- OVERDAMPED – Use a lower value than the resistor you have chosen.

We will discuss each of these options below, and show a method for viewing response waveforms on an oscilloscope.

### *Optimum Response*

If your system has an optimum response with the resistor you have chosen, no further adjustments are necessary.

### *Underdamped Response*

To optimize if your system is underdamped, use a resistor whose value is *larger*. The increased resistance will reduce the gain of the error amplifier, and diminish the signal that goes into the PWM circuit. Consequently, the power stage will be on for a shorter period of time, current rise will be slowed, and damping in your system will be increased.

### *Overdamped Response*

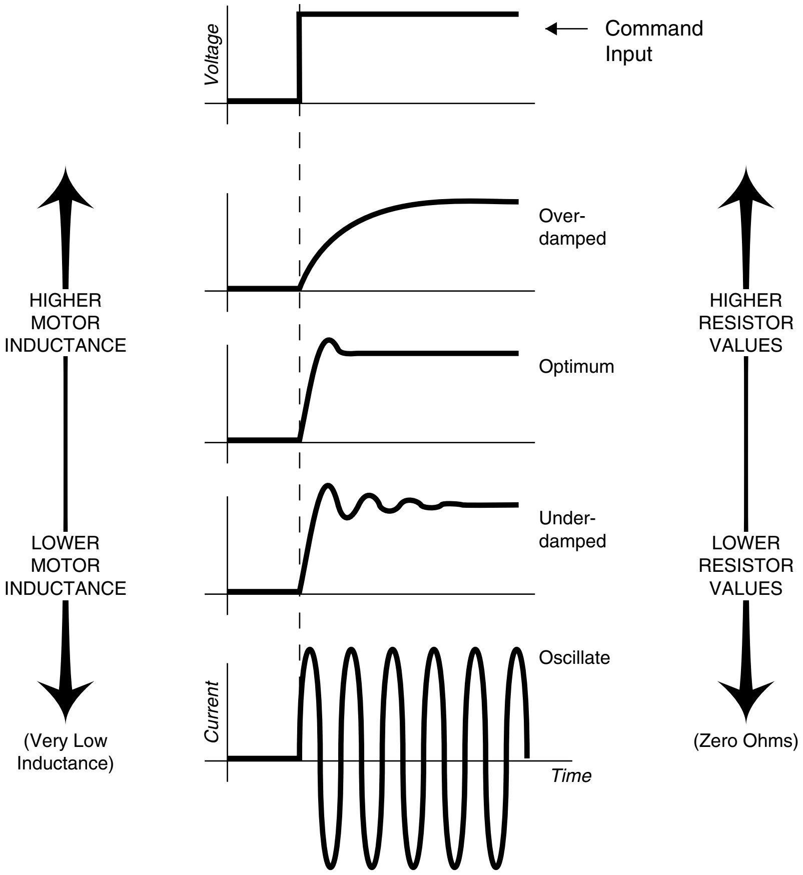
To optimize if your system is overdamped, use a resistor whose value is *smaller*. With less resistance, the error amplifier's gain will be higher, a larger signal will reach the PWM circuit, and the power stage will stay on longer. More current will flow, which will cause a faster current rise. The system will have less damping, and will respond more quickly.

### *A Graphical Representation*

The next drawing provides a visual summary of effects you can expect from changing either the motor inductance or the response resistor.

Arrows on the left side of the drawing show the effects of changing the motor inductance while keeping other components unchanged. Increasing the inductance will cause overdamping; decreasing the inductance will cause underdamping.

Arrows on the right side of the drawing show the effects of changing the response resistor while keeping other components unchanged. Increasing the resistance will make your system overdamped; decreasing the resistance will make it underdamped.

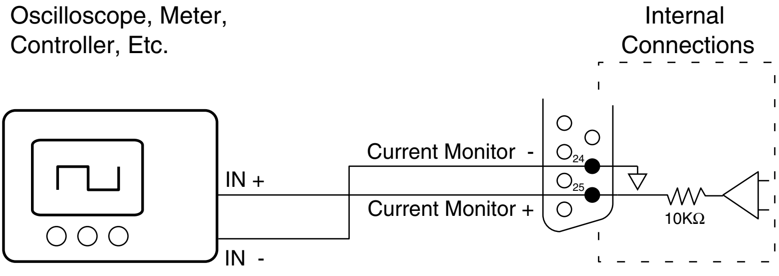


*Response Waveforms*

### *Viewing the Response Waveform*

You can view your system's response waveforms on an oscilloscope, and compare them to the drawings we have presented throughout this section.

Connect an oscilloscope to the drive's current monitor output, as shown in the next drawing.



*Current Monitor Output Connections*

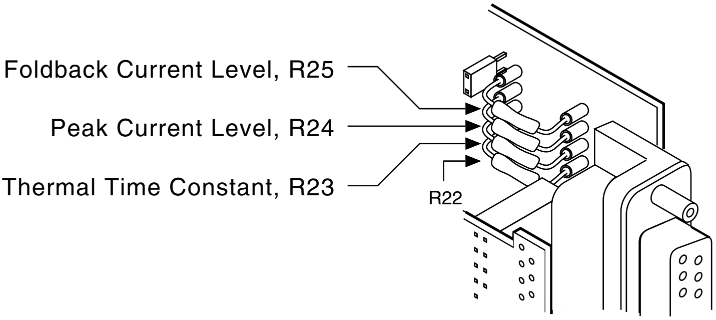
From the picture on your oscilloscope screen, you can see if your system is overdamped or underdamped. If necessary, change the value of the response resistor to improve performance. Monitor the waveforms until you get the response you want.

## Current Foldback

A mechanical jam in a servo system can cause the motor to overheat. In contrast to a stepper motor, which does not run hotter when jammed, a servo will apply full current (for full torque) while it attempts to move as commanded. Usually, this current will be *much* higher than the motor can withstand continuously. If it persists indefinitely, it may damage the motor's windings.

To protect the motor from overheating, the OEM770X has a *current foldback* circuit. If high motor current continues for too long, the circuit reduces current to a lower level, which decreases the rate of motor heating.

You can adjust the foldback circuit by changing three resistors on the drive's circuit board—R23, R24, and R25.



*Foldback Resistor Locations*

See *Installing Selectable Resistors* in *Chapter 2 Installation* for an explanation on how to change foldback resistors.

You have two options for choosing resistors for current foldback:

- Select resistors to use with Compumotor SM and NeoMetric Series motors.
- Select resistors to use with motors from other vendors.

The following sections will explain when you should use foldback, how the current foldback circuit works, and how to choose resistor values.

## WHEN DO YOU NEED FOLDBACK?

If you have properly sized the motor for your application, and you set up your OEM770X so that it can detect a mechanical jam, *you do not need foldback*. The OEM770X can protect the motor more quickly and completely than a foldback circuit can. It can also keep the machine from producing bad parts, which sometimes happens when one axis folds back and others continue to run normally.

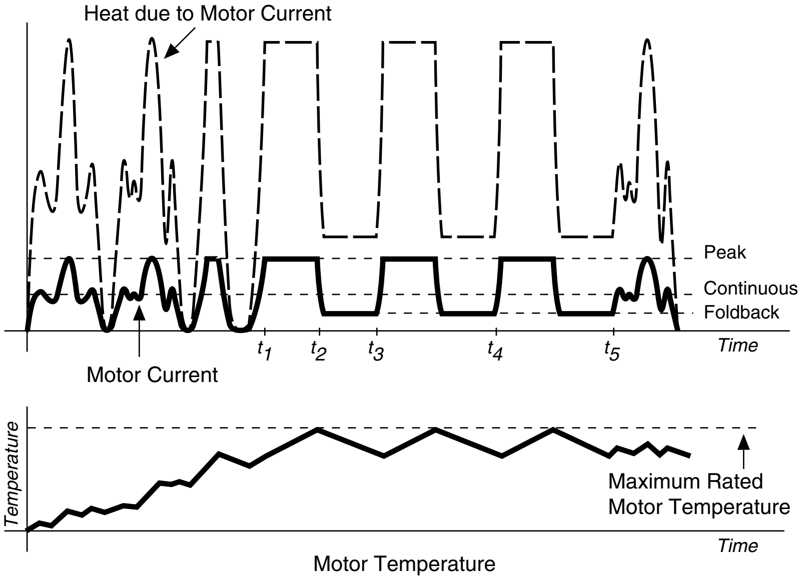
In the most common method of detecting a jam, the OEM770X shuts down the system if the actual position is significantly different from the commanded position. All servo applications should include a position error shutdown, if possible.

If your OEM770X cannot detect a jam, or if you need to limit peak torque in your system, you should use the foldback circuit.

## CURRENT FOLDBACK—HOW DOES IT WORK?

The OEM770X does not directly measure motor temperature. Instead, it uses an electrical circuit to model the motor's thermal performance. Actual current flows in the motor; a replica of the actual current flows in the foldback circuit. Current in the motor is converted to heat, and the motor temperature rises; current in the foldback circuit charges a capacitor, and the voltage on the capacitor rises. The drive uses the capacitor voltage to represent motor temperature.

The following drawing shows the relationship between current, heat, and temperature in the motor. (For clarity, only positive motor currents are shown.)



*Current Foldback*

The current waveforms for several moves are shown. The rotor becomes locked at time t$_1$, and peak current flows in the motor (for maximum torque). Current is converted to heat, and the motor temperature rises. When the temperature reaches the motor's maximum rating at time t$_2$, the foldback circuit takes control, and reduces motor current to a lower level. The motor can then cool down.

At times t$_3$ and t$_4$, the foldback circuit permits full current to flow again. Because the rotor is still locked, the foldback cycle repeats. By time t$_5$, however, the rotor has been released. Normal operations can now continue. (*Note*: Sometimes when the drive goes into foldback, it stays in foldback until the command input voltage is reduced. The system's parameters determine whether the drive goes in and out of foldback, as shown in the drawing above, or stays in foldback.)

While the rotor was locked, the foldback circuit reduced the rate of motor heating.

Notice the relationship between current, heat, and motor temperature. Current is converted to heat in the motor. The heat's magnitude is proportional to the *square* of the current. As this heat is dumped into the motor, the motor's temperature rises. The temperature is the accumulation, over time, of the net heat in the motor. It is also proportional to the square of the motor current.

You can match the foldback circuit to your particular motor and application by selecting three resistors. The following sections describe the function of each resistor.

### *Peak Current: $I_{pk}$, R24*

Peak current is the maximum current the OEM770X will produce in the motor. You can set it as high as 12 amps. For Compumotor SM motors with "A" windings, and NeoMetric motors with "D" or "E" windings, recommended peak currents are in the 6 – 10 amp range.

In applications where you wish to limit peak current, or the peak torque applied to mechanical assemblies, use R24 to reduce the peak current the drive supplies to your motor.

### *Foldback Current: $I_{fold}$, R25*

When the foldback circuit takes control, it reduces motor current to a lower level, which is called the *foldback current*. R25 sets the foldback current level.

To ensure that the rate of motor heating is reduced, the foldback circuit enforces a limited duty cycle between operations at high current and operations at foldback current. The *average* power in the motor during this period is approximately equal to the power that would be produced if the motor operated at its rated continuous current level for the same period of time.

The motor's continuous current rating specifies the maximum current at which the motor can run indefinitely without overheating. Try to match your motor’s current rating to your application and operating conditions.

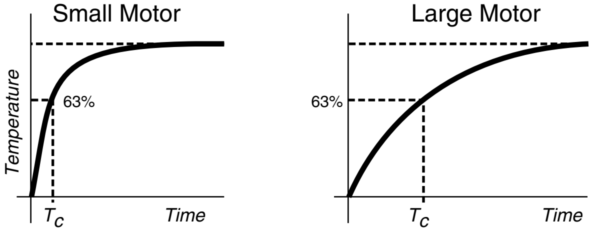
If you use R24 to limit peak current, be sure to also change R25, so that the foldback current is lower than the peak current.

### *Thermal Time Constant: $T_{c\text{-}therm}$, R23*

Every motor has its own particular winding-to-stator time constant. This is the time it takes for the *motor winding* to reach 63% of its equilibrium temperature, after application of rated current. The time for the *motor case* to reach equilibrium temperature is different, and is usually much longer.

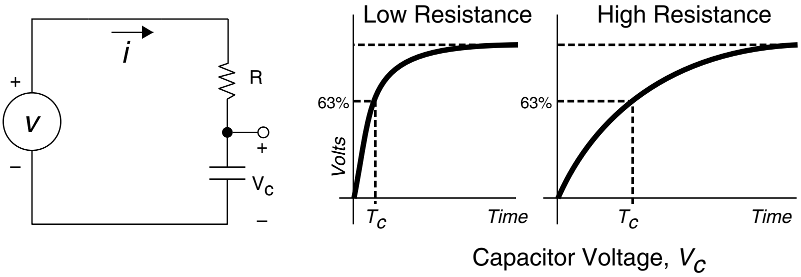
Small motors usually have much shorter time constants than large motors. Heat dumped into a small motor causes a fast rise to the equilibrium temperature. A large motor has a much greater thermal mass—consequently, the same quantity of heat will cause a much lower temperature rise. The large motor can absorb heat over a longer period of time before it reaches its maximum rated winding temperature.

The next drawing shows time constants for a small motor and a large motor.



*Motor Time Constant*

The drive uses an electrical circuit to model the motor's thermal characteristics. The next drawing shows the part of the circuit that models the motor's thermal time constant.



*Foldback Time Constant*

In this circuit, the voltage source *v* is proportional to heat in the motor.

$$v \propto (I_{replica})^2 - (\mathrm{I}_{foldback})^2$$

where

$I_{replica}$ is a scaled replica of the motor current

$(I_{replica})^2$ represents heat entering the motor

$(I_{foldback})^2$ represents heat leaving the motor

$v_c$, the voltage on the capacitor, represents motor temperature. R23 controls how fast $v_c$ can change.

If you select an appropriate value for R23, the *RC* time constant of the circuit will match the thermal time constant of your motor. In the drawing above, the two graphs on the right show that a low resistance produces a time constant similar to a small motor's time constant; a high resistance gives a longer time constant, similar to that in a large motor.

R23, therefore, controls the time constant in the foldback circuit. It is scaled to one second per megohm. Very small motors should use a lower (faster) value for R23. For larger motors that need peak power for long acceleration times, you can increase R23 to as high as 10 megohms. Values higher than this are not recommended.

These points are summarized below.

| | |
|---|---|
| • SCALING: | 1 sec per MΩ |
| • MAXIMUM: | 10 MΩ |

Notice that the time constant *averages* the flow of heat in the motor. This means that previous circuit behavior will affect foldback. If the motor has been working hard, then suddenly demands peak current, the time to foldback will be short. On the other hand, if the motor has been idle much of the time, its average heat will be low. The circuit will recognize this—if the motor demands peak current, the time before foldback occurs will be longer.

As a general guideline, if you reduce R23 by half, then time to foldback will be cut almost in half.

## Resistor Selection

The following sections describe three application situations.

- High Torque Permitted: OEM770X Can Detect a Jam
- High Torque Not Permitted
- OEM770X Cannot Detect a Jam

To select foldback resistors, determine which of the situations apply to your system, and follow the instructions in the relevant section below.

### High Torque Permitted: OEM770X Detects Jam

If your mechanical system can withstand the peak torque of your motor with 12 amps in it, and your OEM770X can detect a jam, you can probably use the resistors in the table below. These resistors allow 12 amps peak current for 0.5 – 2 seconds before foldback occurs (depending on the level of current *before* the peak), and will allow currents up to 6 amps continuously.

**Foldback Resistors for 12A Peak, 6A Continuous**

| Res. #: | Function | Resistor Value | Current |
|---|---|---|---|
| R25 | Foldback Current | 23.7 KΩ | 6A |
| R24 | Peak Current | Ø Ω | 12A |
| R23 | Time Constant | 5.1MΩA | |

To verify that these resistors are suitable for your application, test your system as described below.

If you experience undesired foldback (red LED lights, but goes out when the command input voltage is reduced), the foldback circuit can be disabled by replacing R25 with a 0 – 10 ohm resistor. Even with foldback disabled, you can still limit peak current (and thus peak torque), by installing an appropriate resistor value for R24, $I_{pk}$.

### High Torque Not Permitted

If your mechanical system cannot withstand the peak torque that the OEM770X can produce, you can limit peak current, and thus peak torque, with R24. See the *Peak Current* table below for appropriate resistor values.

### OEM770X Cannot Detect a Jam

If your OEM770X *cannot* detect a jam, you should determine foldback resistor values appropriate for your application and install them in your drive. When a jam occurs with these resistors installed, the OEM770X will reduce the motor current to a lower level

This mode of operation greatly reduces the rate of motor heating, and allows more time for the machine operator to notice that there is a problem and shut the system down. As a warning to the operator, the red LED on the front panel will be illuminated while the drive is in foldback.

If you use Compumotor servo motors, the table *Resistors for SM and NeoMetric Motors* in *Chapter 2 Installation* lists suggested resistors for you to use.

These values will be appropriate for most applications. However, there are many variables that affect the actual motor operating temperature (see the list below in *Application Conditions Affect Foldback)*. You may need to adjust these resistors further.

The next table gives resistor values for specific peak currents and foldback currents.

| **R24, PEAK CURRENT** | | | **R25, FOLDBACK CURRENT** | | |
|---|---|---|---|---|---|
| $I_{pk}$ (*amps*) | R24 | | $I_{fold}$ (*amps*) | R25 | |
| 3 | 845 | KΩ | 2 | 1.2 | MΩ |
| 4 | 450 | KΩ | 3 | 124 | KΩ |
| 5 | 348 | KΩ | 4 | 53.6 | KΩ |
| 6 | 249 | KΩ | 5 | 36.5 | KΩ |
| 7 | 182 | KΩ | 6 | 26.1 | KΩ |
| 8 | 124 | KΩ | 7 | 18.7 | KΩ |
| 9 | 86.6 | KΩ | 8 | 13.3 | KΩ |
| 10 | 56.2 | KΩ | | | |
| 12 | Ø | Ω | | | |

A starting point for $I_{fold}$ is to choose R25 so that the foldback current is 70% of the motor's continuous current rating.

If you experience "nuisance" foldback where the current is reduced, but the motor is not too hot and no jam exists, try increasing the foldback current.

To disable current foldback, replace R25 with a 0 – 10 ohm resistor. You can still specify peak current with R24—but the drive will never reduce current with R25 below 10 ohms.

### *Application Conditions Affect Foldback*

The foldback circuit is well defined, but it is a simplified, approximate model of what actually occurs in the motor. Circuit limitations and differences in application conditions can cause widely varying results.

Some conditions that affect motor temperature are:

- Ambient temperature
- Air flow on the motor
- Heatsinking of motor (size, composition, and temperature of the motor mounting surface)
- Move profile and duty cycle
- Motor core losses

Other conditions may be important in your system.

Because many variables affect motor temperature, we recommend that you treat the suggested resistor values as a starting point in developing your thermal management strategy. You may need to determine the best values empirically. For optimum motor protection, choose values as conservatively as possible. Finally, test your system as described below.

### *Application Examples*

If you have a load that is primarily frictional (for example, a spindle drive), you can set the peak current limit resistor, R24, to a value that will keep the current below the continuous current rating of your motor. This will ensure that the current cannot exceed the motor's rating. Check the motor temperature under actual operating conditions.

If you have a load that is primarily inertial (for example, a point-to-point move with low friction), you can set the foldback current resistor, R25, to a low value that will protect against a jam but still allow full peak current for the acceleration portion of the move. If the move duty cycle is low, the overall average power will also be low, even though the peak power may be quite high. Therefore, you can use a low foldback current setting.

## TEST YOUR SYSTEM

Once you have selected and installed foldback resistors, you should perform two tests to verify that the foldback circuit adequately protects your motor.

- Measure Motor Temperature
- Simulate a Jam

These tests are described below.

### *Measure Motor Temperature*

Measure the motor case temperature under actual operating conditions. Make your measurements after the motor temperature has reached equilibrium (which can take several hours). Compare the results with the motor's ratings.

Compumotor servo motors have an internal thermoswitch, with normally closed contacts. If the motor windings exceed predetermined temperature levels, the contacts will open. Monitor the thermoswitch to verify that the contacts remain closed during operating conditions.

### *Simulate a Jam to Verify Resistor Values*

Set OEM770X position-error shutdown limits to appropriate values. To avoid motor overheating, follow these steps:

1. With foldback resistors installed, start your test with a cold motor. Command full current while you simulate a jam.
3. Monitor the red LED. It will illuminate when the drive goes into foldback. ***Do not overheat the motor!*** If the drive does not go into foldback when you expect it to, stop the test immediately.
4. Monitor the fault output. It should be low at the start of your test, and should remain low when the drive goes into foldback. (Foldback is the only condition where the red LED illuminates, but the fault output is low.)
5. Watch to see that the drive comes out of foldback, indicated by the red LED turning off. If the drive does not come out of foldback on its own, reduce the command input voltage; the red LED should then turn off.

The results of your test indicate how much time an operator has to shut down the system in the event of an actual jam.

## How Long Will Foldback Protect Your System?

Ideally, foldback should prevent the motor from overheating under all conditions of improper application. In practice, because of the many variables affecting motor temperature, foldback can only *delay* motor overheating. This will allow more reaction time for an operator or control system to detect that the machine is jammed.

With foldback, the time before motor overheating occurs can be increased from a few minutes to 10 – 30 minutes for large motors, or from seconds to 1 – 2 minutes for small motors.

The degree of expected operator attention is also a factor. If the machine will be running unattended, we strongly recommend you set up your OEM770X so that it can detect a jam.

If your OEM770X cannot detect a jam, use a conservative approach and select foldback resistors that limit worst-case motor temperature to a safe value for an indefinitely long period of time.

If the machine operator is nearby and will notice within a reasonable period of time that the machine is jammed, you can use a more aggressive approach to selecting resistors. Different resistors may allow higher motor performance, yet still limit the rate of rise of motor temperature so that the operator has time to react, and shut the machine down.

# CHAPTER 5

# *Hall Effect Sensors*

The OEM770X works with three-phase brushless motors equipped with Hall effect sensors or equivalent feedback signals. In this chapter we will explain how Hall effect sensors are used in brushless motors, and how the OEM770X uses Hall effect outputs from Compumotor servo motors for commutation.

If you are using a motor from another vendor, obtain information about your motor's Hall signals and commutation sequence. Then use the information in this chapter to help you connect your motor to the OEM770X.

## Hall Effect Sensors and Commutation

To move the rotor in the commanded direction, the drive will send current through two of the motor's stator coils. This current produces electromagnetic fields that develop a torque on the rotor, and the rotor turns. The rotor will stop if it can reach a position where its permanent magnets are next to the magnetic fields that attract them. Before the rotor can get to this position, though, the drive switches the current to a new combination of stator coils, and creates a new set of electromagnetic fields that cause the rotor to continue its movement.

The process of continually switching current to different motor coils to produce torque on the rotor is called *commutation*.

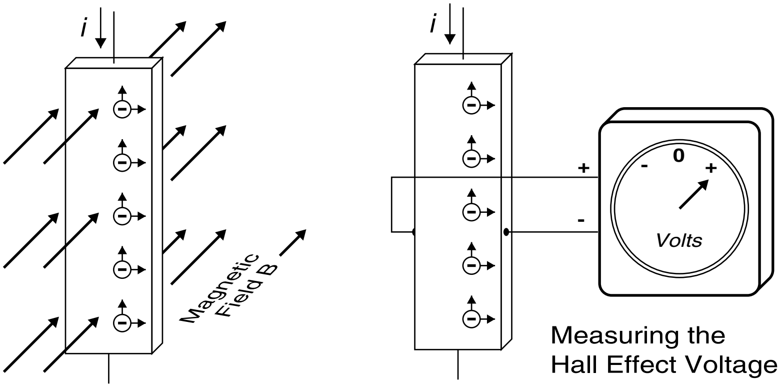
If the drive knows the position of the rotor's permanent magnets, it can set up magnetic fields in the stator that have the correct location and polarity to cause the rotor to turn. How can the drive know rotor position? Three Hall effect sensors located in the motor are affected by the rotor's permanent magnets. The three sensors transmit a unique pattern of signals for each rotor position. The drive uses these signals to determine the position of the rotor.

## The Hall Effect

Electrically charged particles moving through a magnetic field experience a deflecting force perpendicular to both the direction of their motion and the direction of the magnetic field.

The *Hall effect* is a phenomenon which shows that if a magnetic field is perpendicular to a thin strip of conductive material, and an electric current flows lengthwise through the strip, the mobile charges that carry the current will drift to one edge as they move along the strip.

In the example shown in the next drawing, assume that the conductive strip is metal. Electrons are the mobile charges. With a current *i* as shown in the drawing, the electrons will move upwards through the strip. In the presence of the magnetic field *B*, shown in the drawing, the electrons will drift toward the right edge of the strip.



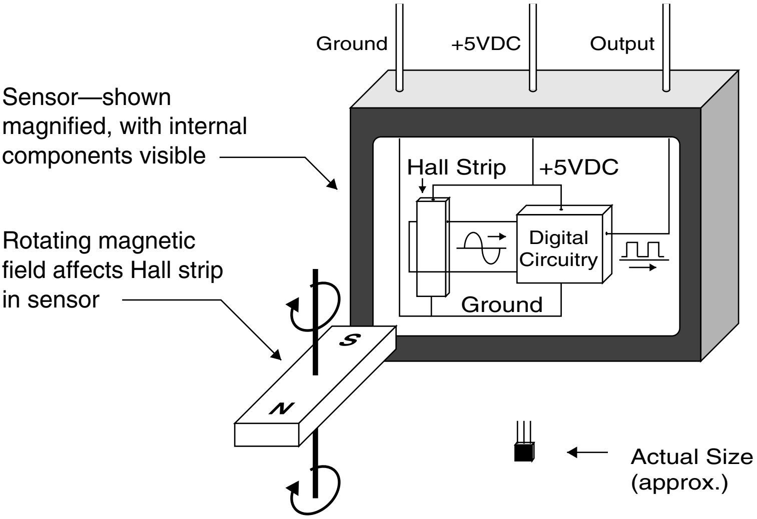
*The Hall Effect*

Because electrons are concentrated along one edge, there is a potential voltage difference across the strip. This voltage is known as the *Hall effect voltage*. The drawing shows a voltmeter connected across the strip to measure Hall effect voltage.

If the magnetic field is removed, the Hall effect voltage disappears. If the magnetic field is reversed, the Hall effect voltage will also be reversed.

## HALL EFFECT SENSORS

Many types of sensors use the Hall effect to sense the presence of magnetic fields. The next figure is a conceptual drawing of a Hall effect sensor.


Ground
+5VDC
Output
Sensor—shown magnified, with internal components visible
Hall Strip
+5VDC
Digital Circuitry
Ground
Rotating magnetic field affects Hall strip in sensor
S
N
Actual Size (approx.)

*Hall Effect Sensor*

A constant current runs through a conductive Hall strip inside the sensor. The drawing shows a rotating magnet near the sensor. The alternating field from this rotating magnet will cause an alternating Hall effect voltage to be generated across the strip.

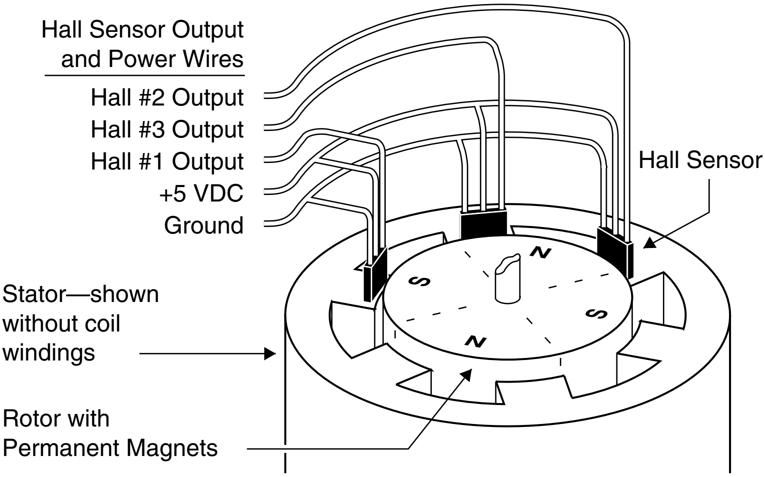
This alternating voltage waveform is fed into circuitry that shapes the waveform. The output of the circuitry is a digital signal that is either +5VDC or ØVDC.

Sensors are available with a variety of output voltages and polarities. In the following discussion, we assume that the sensor is turned ON by a south magnetic pole, and remains on after the south pole is removed. When a north magnetic pole approaches, the north pole will turn the sensor OFF.

Note from the drawing that the sensor requires power connections for its internal circuitry (+5VDC and Ground). Also note that although the actual Hall effect voltage generated inside the sensor is an analog signal, the *output* from the sensor is a digital signal that is either ON or OFF.

## HALL EFFECT SENSORS USED INSIDE BRUSHLESS MOTORS

There are three Hall effect sensors inside a motor. The next figure shows a conceptual drawing of the inside of the motor, and the three sensors.



*Hall Sensor Location (Shown Mounted Above Stator Pole Faces)*

For clarity, the stator is depicted in simplified form, without its coil windings. The Hall effect sensors are located at one end of the stator, near the pole faces of the rotor. They are positioned approximately as shown in the figure.

Five wires are shown for making connections to the Hall sensors. Three wires are for individual outputs. The fourth and fifth wires are  for +5VDC and Ground, which are internally connected to all three sensors.

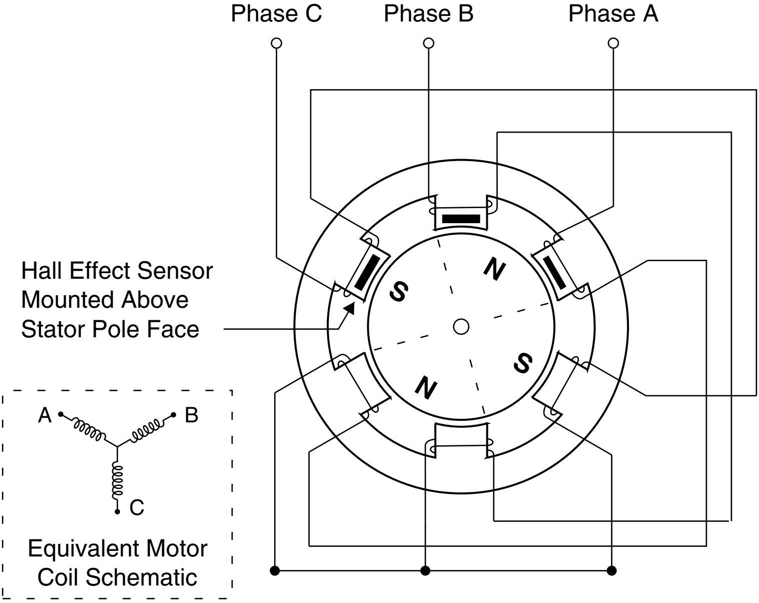
Note that Hall #3 is positioned between Hall #1 and Hall #2.

***Do Compumotor Motors Have Hall Effect Sensors?***

Most Compumotor servo motors do not use Hall effect sensors. Instead, the motor's encoder has an extra *commutation track*, with three outputs. These outputs mimic signals that would be obtained from Hall sensors; in fact, the outputs are called *Hall outputs*. For conceptual reasons, in the discussion that follows we assume the motor contains Hall sensors. Keep in mind that no matter how the original signals are generated—from sensors or from an encoder—the result is the same: three output wires that deliver commutation information to the drive.

## Windings in a Three Phase Brushless Motor

The next drawing depicts an end view of the motor, with the separate phase windings shown in their relative positions around the stator. The three phases share a center connection, as the detail within the dotted line shows.


Phase C
Phase B
Phase A
Hall Effect Sensor Mounted Above Stator Pole Face
S
N
S
N
A
B
C
Equivalent Motor Coil Schematic

*3-Phase Servo Motor with Hall Effect Sensors*

The physical spacing of the Hall effect sensors is very important. Notice that one pole of the rotor can affect two sensors at the same time. In this drawing, the rotor's north pole is adjacent to both Hall 2 and Hall 3. Since south turns a sensor ON and north turns it OFF, the Hall outputs in this drawing would be 1ØØ. (In this example, 1 = ON and Ø = OFF. 1ØØ, therefore, means that Hall 1 is ON, Hall 2 is OFF, and Hall 3 is OFF.)

The OEM770X will send current into one phase and out of another—the third phase receives no current. When current flows through a phase, two magnetic poles of the same sign are formed on opposite sides of the motor. We will use the convention in these drawings that when current flows from the drive into a coil, it will produce a north pole. When it flows from a coil to the drive, it will form a south pole.

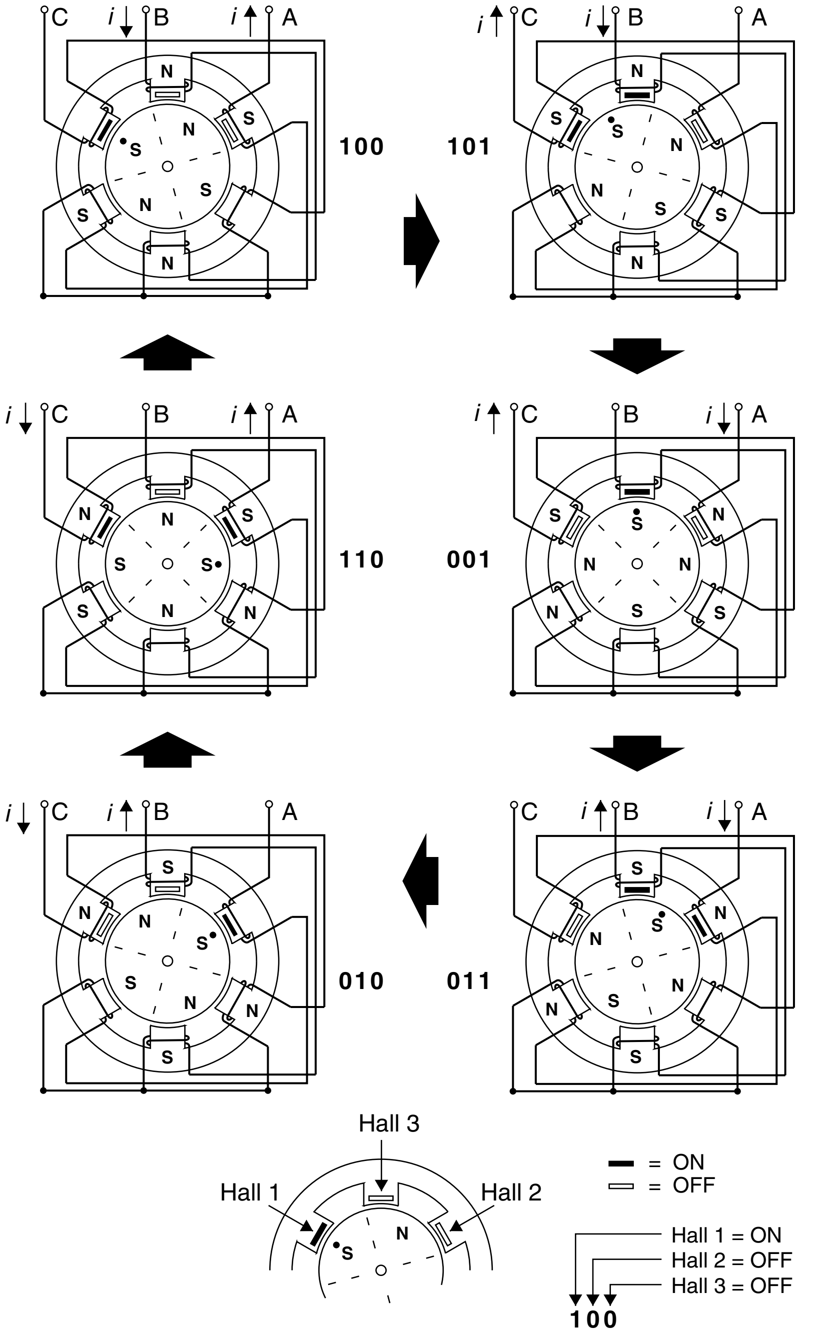
For example, suppose current goes into the motor through Phase A, and exits through Phase B. (Phase C has no current in it.) The current will flow through the windings in A and form north magnetic poles on opposite sides of the stator. The current flows through the center connection, and enters B's windings, where, because of the direction of the current, south magnetic poles are formed on opposite sides of the stator. (Refer to the previous drawing.)

From this example, notice that, although the stator has six locations for pole faces, there are only four poles at any one time. The other two pole faces have windings that carry no current—therefore no magnetic poles are formed by those windings.

## The Six Possible Hall States

The next figure illustrates that, as the rotor turns, six different Hall states will be produced in a predictable and repeatable sequence.

This drawing shows the rotor, stator, phase coils, and Hall sensors. A small black dot has been drawn next to one of the south poles, to help show the motion of the rotor as it turns. (The two south poles in the rotor are actually indistinguishable from each other, as are the north poles.)



*Hall Sensor States*

For each of the six different rotor positions in the drawing, a current is shown that will cause the rotor to rotate in a clockwise direction. The stator is labeled with N or S, to show the magnetic fields the current produces. These fields exert the torque on the rotor that causes it to move.

Each rotor position is labeled with its corresponding Hall state (100, 101, 001, etc.). These numbers represent the three Hall sensors, and whether they are on or off. The first digit corresponds to Hall 1, the second to Hall 2, and the third to Hall 3.

What voltage levels correspond to on and off? We use this convention:

- 1 = ON = +5VDC
- Ø = OFF = ØVDC
- Voltage is measured at the OEM770X's Hall input, with the Hall wire connected to the input, and the drive turned on.
- If no drive is available, connect the Hall wire to a 1KΩ pullup resistor. Connect the resistor to +5VDC. Connect Hall +5 and Hall Gnd to your power supply. Measure the voltage at the point where the Hall wire is connected to the resistor.

To understand this drawing, examine the rotor position at Hall state 100. The south pole turns Hall 1 on. The north pole turns off Hall 2 and Hall 3. The Hall state, therefore, is 100. (Hall 1 = ON, Hall 2 = OFF, Hall 3 = OFF)

If current flows into phase B and out of phase A, north and south poles form in the stator. These poles exert a strong torque on the rotor's north pole, and it will turn clockwise.

If the rotor could turn far enough so that its north pole was aligned with the south pole in the stator, the rotor would stop. However, immediately before the rotor reaches this position, the Hall state changes. The south pole (with a dot on it, in this figure) moves into position next to Hall 3 and turns it on. The Hall state is now 101 (Hall 1 = ON, Hall 2 = OFF, Hall 3 = ON. Remember, Hall 3 is located between Hall 1 and Hall 2. See the detail at the bottom of the drawing.)

If current is now directed into phase B and out of phase C, a new set of magnetic fields forms in the stator that exert a strong torque on the rotor's south pole. The rotor moves further in a clockwise direction, and when it turns far enough, the Hall state changes to 001. At this point, directing current into phase A and out of phase C will keep the rotor turning to state 001.

The next Hall states the rotor will pass through are 010 and 110. When the south pole *without* the dot reaches state 100, a complete electrical cycle has occurred, and the rotor has rotated through 360 *electrical* degrees. (Physically, it has rotated through 180 *mechanical* degrees.) At this point, the same sequence of Hall states begins again.

Notice that the Hall states are not determined by the current flowing in the stator. They simply report information about the position of the rotor. Whether you turn the rotor by hand, or cause it to turn by directing current through the motor's coils, the Hall effect sensors are influenced only by the magnetic fields of the rotor.

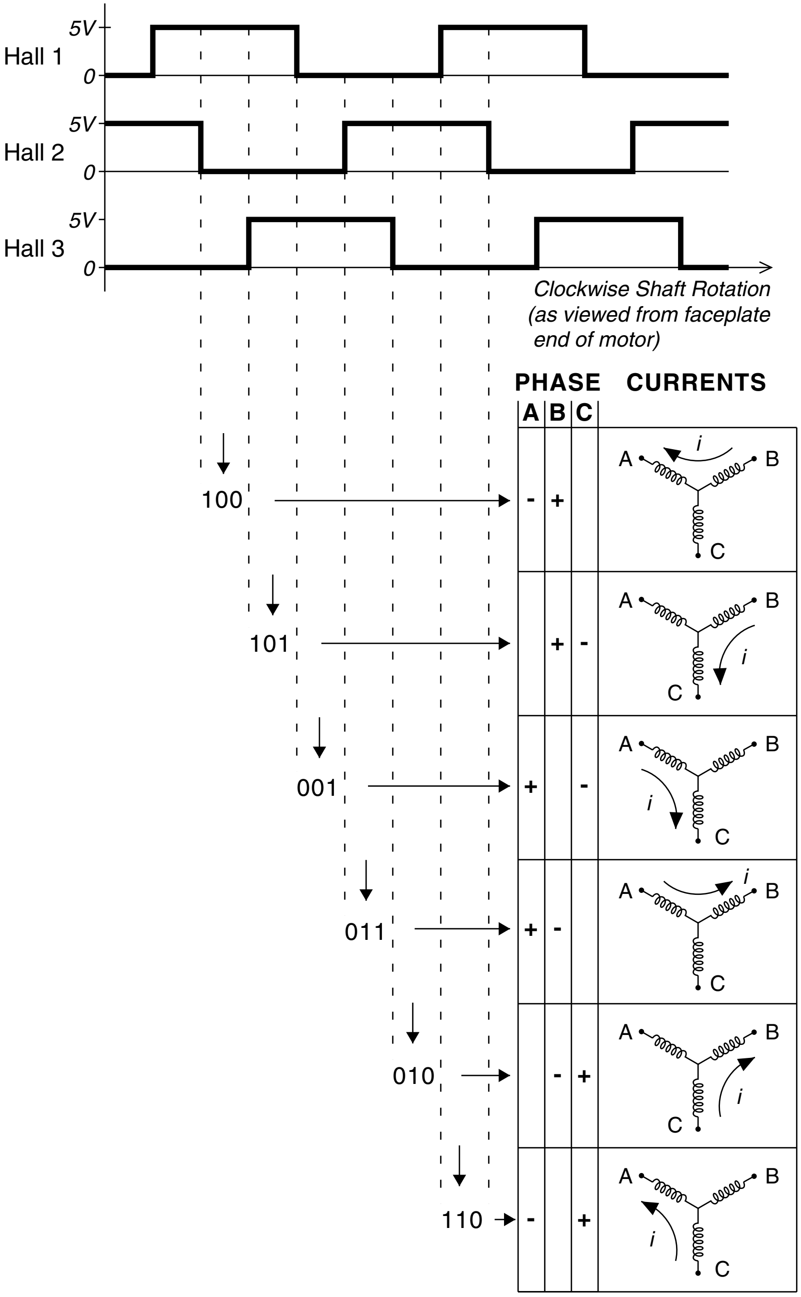
The Hall effect outputs in Compumotor servo motors divide the electrical cycle into three equal segments of 120° (*electrical* degrees, not *mechanical* degrees). Outputs used in this arrangement are called 120° Hall effect outputs. The Hall states 111 and 000 never occur in this configuration.

Another arrangement, rarely used in modern servo motors, uses a 60° Hall effect sensor configuration, in which the states 111 and 000 can occur. Do not attempt to use such a motor with the OEM770X. It will not operate properly.

## COMMUTATION BASED ON HALL STATES

The OEM770X monitors its three Hall inputs. It uses internal logic circuitry to assign a rotor position to each of the six Hall states, and then direct a motor current that results in rotor movement in the commanded direction.

The three Hall signals produced by clockwise shaft rotation are shown at the top of the next drawing. The Hall states are also listed, along with the table of phase currents the OEM770X uses for each Hall state.



| | PHASE A | PHASE B | PHASE C |
|---|---|---|---|
| 100 | - | + | |
| 101 | | + | - |
| 001 | + | | - |
| 011 | + | - | |
| 010 | | - | + |
| 110 | - | | + |

*Commutation for Clockwise Shaft Rotation—Based on Hall States*

For counterclockwise rotation, two changes are made. First, as the rotor moves counterclockwise, it passes through the same Hall states, but in the opposite order. (In this drawing, read the Hall states from the bottom up for counterclockwise rotation.) The drive sends currents through the same coils shown in this picture, but the direction of the current is reversed from that shown. As a result, a torque is produced in each state that causes the rotor to turn counterclockwise.

## Connecting Motors from Other Vendors

The previous discussion described Compumotor servo motors, and how the OEM770X drive operates them. If you use a motor from another vendor, obtain information from the motor's manufacturer about its sequence of Hall states, commutation scheme, etc. Use the above information about Compumotor motors for guidance on how to connect your motor to the OEM770X.

### Improper Wiring Can Result in Poor Performance

Assume that you arbitrarily connect your motor's three Hall wires to the OEM770X's Hall inputs. For any particular Hall wiring pattern, there are six different ways you can connect wires to Phase A, Phase B, and Phase C.

Of these six possible phase wiring combinations, only one will work properly. Three will not work at all. The other two deserve particular attention: if the motor is wired in one of these two configurations, the motor will turn, but its performance will be severely impaired.

How can you tell if your motor is wired improperly? If it is in one of the two poor-performance configurations, its torque will be much lower than the torque level of a properly wired motor. Also, torque ripple will be very pronounced as the motor turns.

The best way to determine whether or not your motor is wired correctly is to find the three wiring configurations that enable the motor to turn. Compare the motor's torque in each configuration. The configuration with the most torque will be the proper configuration.

## TRIAL AND ERROR METHOD

You can use a trial and error method to connect your motor to the OEM770X. Follow these steps:

1. Arbitrarily assign numbers to your motor's three Hall output wires, and connect them to Hall 1, Hall 2, and Hall 3 on the OEM770X.

2. Connect Hall +5V and Hall GND.

3. Arbitrarily assign letters (A, B, C) to your motor's phase wires, and connect them to Phase A, Phase B, and Phase C on the OEM770X.

4. If the motor turns, find the best phase wiring configuration:
   - Move each phase wire over one position ( A B C —> C A B ). Compare torque and torque ripple.
   - Move each phase wire one position further ( C A B —> B C A ). Compare torque and torque ripple.
   - Use the wiring configuration that gives highest torque and lowest torque ripple.

5. If the motor does not turn, exchange two of the phase wires. The motor should now turn. Go to *Step 4*, compare the three wiring configurations that make the motor turn, and use the proper one.

6. If your motor turns in the opposite direction than you want, you can reverse it using one of several methods.
   - Reverse the command input wires.
   - Reverse the appropriate encoder connections.
   - Exchange two Hall input wires, then follow *Steps 2 – 5* above.

# CHAPTER 6

## Power Supply Selection

To choose a power supply for the OEM770X, you need to answer some important questions.

- How many watts does your system need?
- Will regeneration be a concern ?
- At what voltage should your system operate?
- Should you use a linear power supply or a switching power supply?

The sections in this chapter will help you answer these questions.

### *A Word About Units*

We want a solution for power that is expressed in *watts*. To be consistent with watts, we will express all quantities in SI (metric) units, derived from kilograms, meters, and seconds. The quantities and units we will use are:

| QUANTITY | SYMBOL | UNITS | |
|---|---|---|---|
| Torque | T | Nm | (newton meter) |
| Shaft Velocity | v | rps | (revs per second) |
| | $\omega$ | rad/s | ($2\pi v = \omega$) |
| Shaft Acceleration | a | rps$^{-2}$ | (revs per sec$^2$) |
| | $\alpha$ | rad/s$^{-2}$ | ($2\pi a = \alpha$) |
| Motor Resistance | $\Omega$ | ohms | |
| Torque Constant | $k_T$ | Nm/A | |
| Current | I | A | (amps) |
| Inertia | J | kg-m$^2$ | |

If you want to use other units, apply conversion factors in the appropriate places.

## How Much Power Does Your System Need?

The first step in choosing a power supply is to analyze your motion control system, and determine two quantities:

- Peak Power
- Average Power

*Peak power* is the maximum number of watts the power supply must provide during the most demanding part of the move.

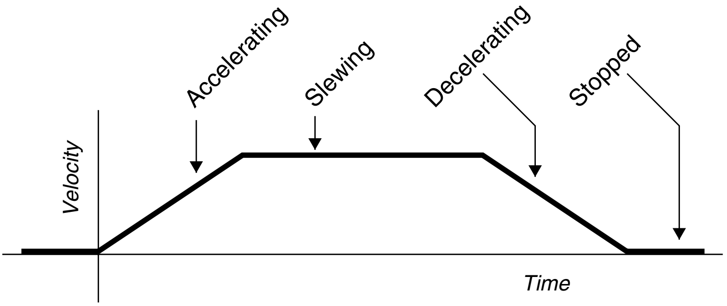
*Average power* is the number of watts required for a repetitive move, averaged over the entire move cycle, including time spent at rest.

In the sections below, we show several ways to determine how much power your system needs: a calculation method; a graphical method; and an empirical method.

It is not our goal to calculate power *precisely*. A full analysis of power in a servo system can be quite complicated and time consuming. Rather, our goal is to easily arrive at a reasonably accurate *estimate* of power needs, and then use this estimate for power supply decisions.

### PEAK POWER—A CALCULATION METHOD

Servo applications vary widely, with many possible move profiles. We will show how to calculate power requirements for the most common move profile, a trapezoidal move.



*Trapezoidal Move Profile*

In the calculation method, we follow these steps:

1. Calculate power required for copper losses
2. Calculate shaft power
3. Add shaft power and copper losses, for total power
4. Add 10% to total power, for miscellaneous losses

Each of these steps will be explained below. To simplify the analysis, we make the following assumptions:

- Equal acceleration and deceleration rates
- Friction is negligible, and can be ignored

***Power for Copper Losses***

During the acceleration portion of a trapezoidal move, constant current in the motor produces constant torque. With a constant torque applied, the motor accelerates at a constant rate until it reaches slew velocity.

Torque is directly proportional to the current in the motor.

$$T = k_T I, \ \ or \ \ I = \frac{T}{k_T}$$

The proportionality constant, $k_T$, is called the torque constant, and is determined by the motor's physical parameters.

The current that produces torque flows through the resistance, *R*, of the motor's copper coils, and causes heat. The power to produce this heat comes from the power supply. (The coil resistance *R* may change with temperature. When you use the equations that follow, use the resistance of your motor at its actual operating temperature.)

Power converted to heat, rather than useful work, is called a *loss*. The losses resulting from current flowing in the motor's copper coils are called *copper losses*, or *$I^2R$ losses,* so named from the formula used to calculate them:

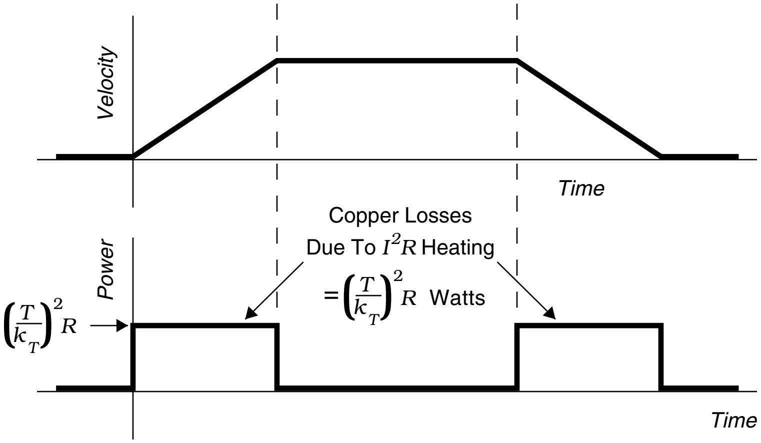
$$P_{copper} = I^2 R$$

$P_{copper}$ represents power used for copper losses.

You can calculate copper losses, even if you do not know the motor current *I*. The following equation uses the relationship between current and torque to express copper losses in terms of torque, resistance, and the torque constant.

$$P_{copper} = I^2 R = \left(\frac{T}{k_T}\right)^2 R$$

Copper losses are shown in the next drawing.



*Copper Losses*

The supply must deliver power only during acceleration and deceleration. During slew with no friction, there is no torque on the motor shaft, and no motor current—consequently, there are no copper losses

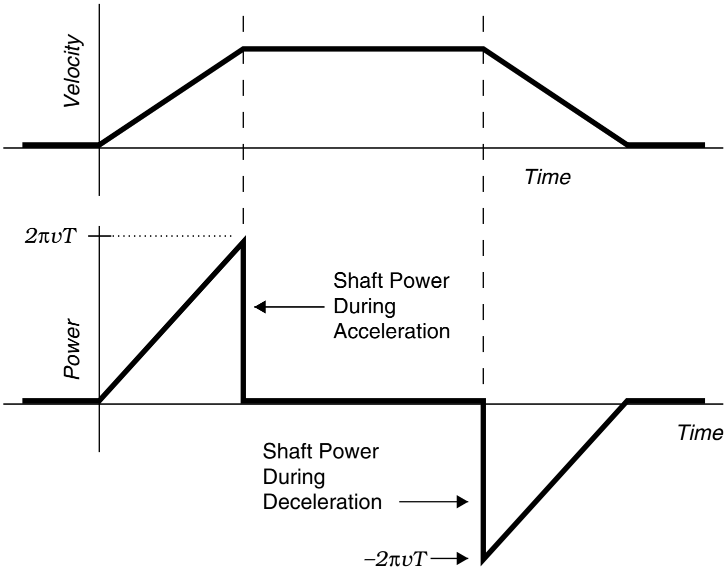
***Shaft Power***

A motor uses shaft power to accelerate or decelerate a load. The equation for shaft power, the product of torque and shaft velocity, is

$$P_{shaft} = \omega T = 2\pi v T$$

where $P_{shaft}$ is shaft power, in watts.

The graph for shaft power is shown in the next drawing.



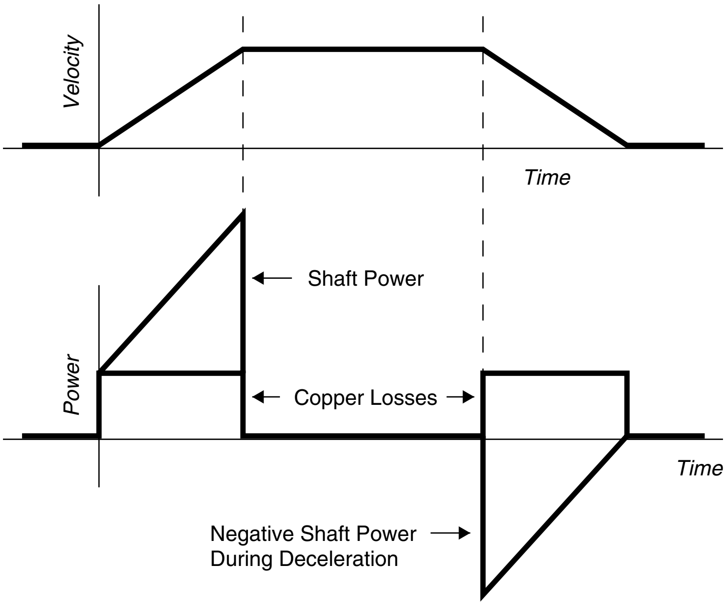
Shaft Power

Torque and velocity are both positive during acceleration. Shaft power, therefore, is also positive.

During deceleration, velocity is still positive, but torque is applied in the opposite direction, and thus is negative. Shaft power, then, is *negative* during deceleration. Negative power is *regeneration*—power flows from the motor, and back into the drive. Later in this chapter, we will discuss regeneration in detail.

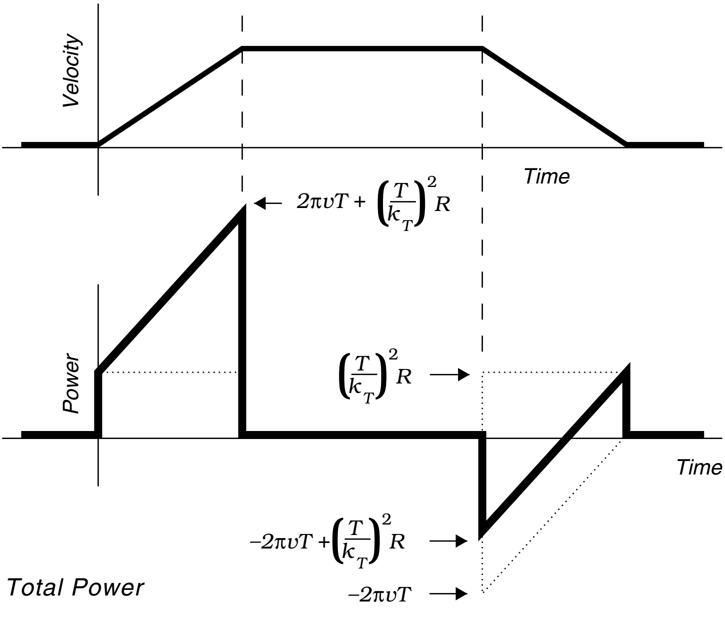
### *Total Power*

In the next drawing, we have combined the graphs for copper losses and shaft power.



*Copper Losses & Shaft Power*

To obtain the total power, we can add together copper losses and shaft power. The heavy line in the next drawing shows the total power that the power supply must provide.


Velocity
Time
$2\pi v T + \left(\frac{T}{k_T}\right)^2 R$
Power
$\left(\frac{T}{k_T}\right)^2 R$
Time
$-2\pi v T + \left(\frac{T}{k_T}\right)^2 R$
$-2\pi v T$

*Total Power*

The equation for power, then, at any velocity during acceleration or deceleration, is:

$$P_{total} = P_{shaft} + P_{copper} = 2\pi vT + \left(\frac{T}{k_T}\right)^2 R$$

The first term on the right represents shaft power. The second term represents copper losses.

Notice that power demand increases as velocity increases during acceleration, and reaches a peak just before the motor reaches its slew velocity. The equation for peak power is:

$$P_{peak} = 2\pi v_{slew} T + \left(\frac{T}{k_T}\right)^2 R$$

***Estimation Factor***

The power equations above show how much power the supply must deliver for shaft power and copper losses. There are other losses, which are usually smaller and less significant, such as:

- Drive Losses
- Core Losses
- Switching Losses

Core losses are dependent on velocity. To approximate their effect, use the power equation from above, and add 10% to it.

$$P = \left[2\pi vT + \left(\frac{T}{k_T}\right)^2 R\right](1.1)$$

For clarity and simplicity in the rest of this chapter, we will omit the 10% figure that represents miscellaneous losses. If you need more accuracy in your estimate, you should include this estimation factor.

Drive losses are not dependent on velocity. When the motor is at rest, or during slew, drive losses are approximately 5–10W.

### Power Supply Current Does Not Equal Motor Current

The equation we have developed represents power that the *power supply* must deliver to the system. This is not the same as *motor power*, or *drive power*. Similarly, current from the power supply will not be the same as current flowing in the motor.

These distinctions can be confusing! To help clarify the situation, think of the equation as an accounting system. All terms on the right side of the equation represent places where power is used in the system: motor heating, shaft power, drive losses, hysteresis, etc. We add up these amounts of power, find the total, and then insist that this total power must have come from the power supply. Therefore, the equation shows how much power the supply must provide for every use on the right side of the equation.

### What about Acceleration and Inertia?

To use the equation we have developed, you only need four pieces of information about your system:

| | |
|---|---|
| $T$ | Torque |
| $v$ | Velocity |
| $k_T$ | Motor Torque Constant |
| $R$ | Motor Resistance |

You may be wondering why acceleration, rotor inertia, or load inertia do not appear in the equation, and what effect these parameters have on power requirements.

The answer is that acceleration and inertia *are* in the equation—they are hidden within the values for torque and velocity. Recall that torque is equal to the product of acceleration and inertia.

$$T = \alpha J = 2\pi a J$$

When you analyze your system, you can derive torque and velocity terms based on acceleration requirements, load inertia, and rotor inertia. Acceleration and inertia, therefore, are implicit in the equation we have developed (and are also implied in speed/torque curves for motors).

## Peak Power—A Graphical Method

Given a speed/torque curve for a particular motor, you can overlay a family of curves that show peak power levels for various moves. To do this, start with the equation for peak power that we developed above. Next, set *P* equal to a fixed value, and then solve for velocity.

$$v = \frac{P - \left(\frac{T}{k_T}\right)^2 R}{2\pi T}$$

For any given torque, you can determine a velocity such that the peak power required to reach that velocity is equal to *P* watts. The graphical method is illustrated in the next example.

***Example***

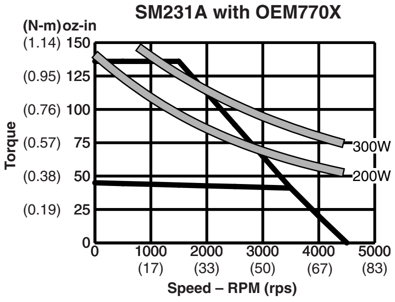
For the SM231A motor at 75VDC, we wish to determine a curve that shows all of the possible speed/torque combinations that require 300W peak power. So, set P = 300W. We then have;

$$v = \frac{300 - \left(\frac{T}{k_T}\right)^2 R}{2\pi T}$$

| *Torque* | *Velocity* |
|---|---|
| *oz–in* (*Nm*) | (*rps*) |
| 75 (0.53) | 73 |
| 100 (0.71) | 45 |
| 125 (0.88) | 26 |

For each torque listed in the table, the peak power required to reach the corresponding velocity is 300W.

In the next drawing, we have plotted these values on the speed torque curve for the SM231A motor. We have also plotted a similar curve, corresponding to moves of 200W peak power.



*Peak Power Curves: SM231A at 200W and 300W*

Any move that falls on the 300W curve will require 300W peak power from the power supply. Moves that lie above the curve will use more torque, a faster velocity, or both, and consequently will need more peak power. Moves that lie below the curve will need less power.

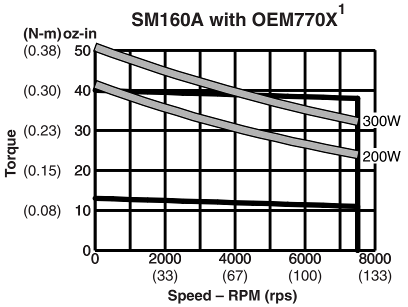
Compumotor’s OEM300 Power Module produces 300W peak and 200W continuous. You could use it to power any move on or below the 300W curve. You could use it continuously for any move below the 200W curve.

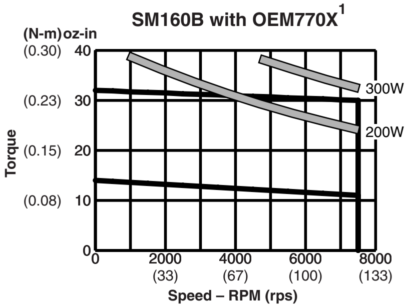
Compumotor's OEM1000 Power Supply produces 1000W. You could use it to power any move within the speed/torque curve.

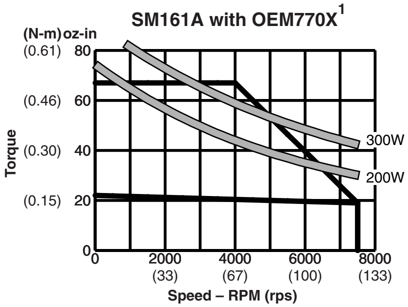
## Peak Power Curves

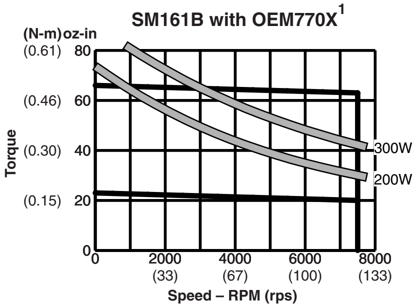
The following drawings show speed/torque curves for SM16, SM23, and NeoMetric servo motors, with peak power curves added.

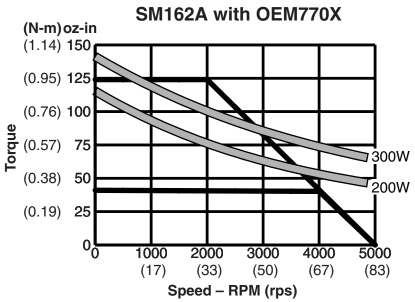
## Peak Power Curves: SM Motors, Frame Size 16 [2]

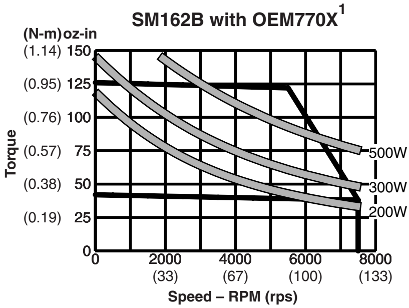

SM160A with OEM770X[1]


SM160B with OEM770X[1]


SM161A with OEM770X[1]


SM161B with OEM770X[1]


SM162A with OEM770X

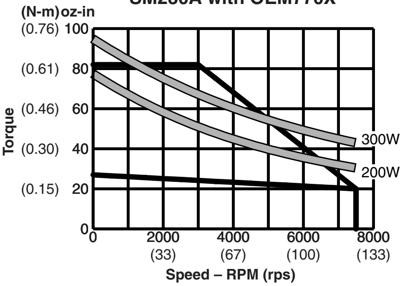

SM162B with OEM770X[1]

[1] For "E" encoder option (1000 ppr), maximum velocity is 6,000 rpm (100 rps).

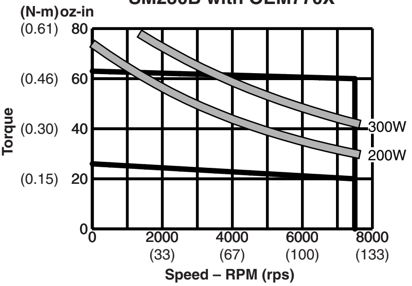
[2] With 75VDC bus voltage; 25°C (77°F) ambient temperature.

## Peak Power Curves: SM Motors, Frame Size 23 [2]

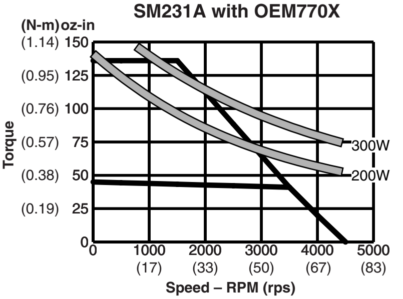
### SM230A with OEM770X[1]



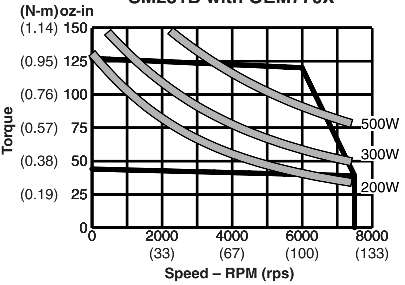
### SM230B with OEM770X[1]



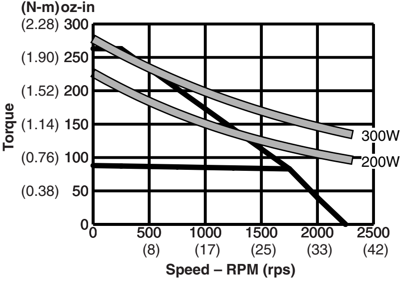
### SM231A with OEM770X



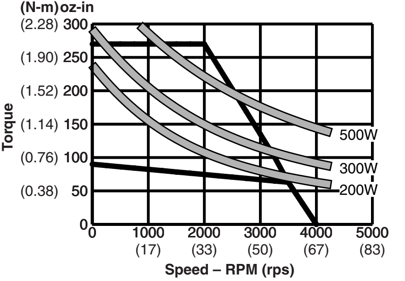
### SM231B with OEM770X[1]



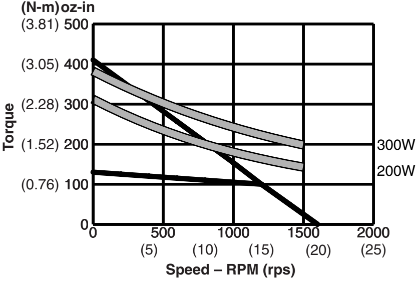
### SM232A with OEM770X



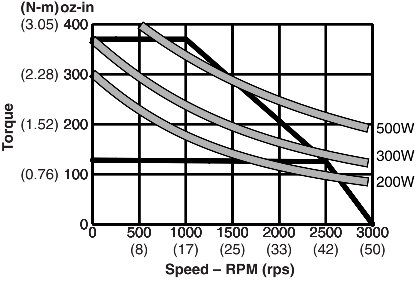
### SM232B with OEM770X



### SM233A with OEM770X



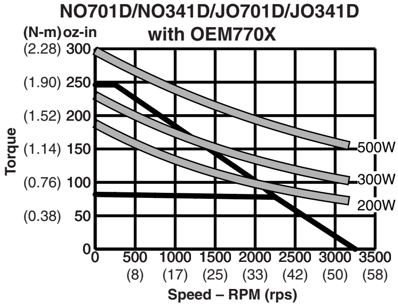
### SM233B with OEM770X

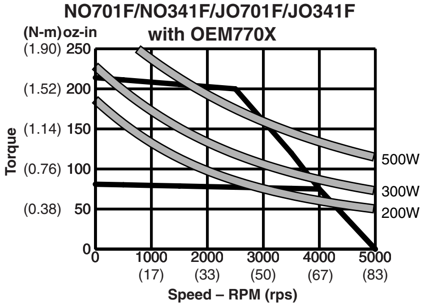


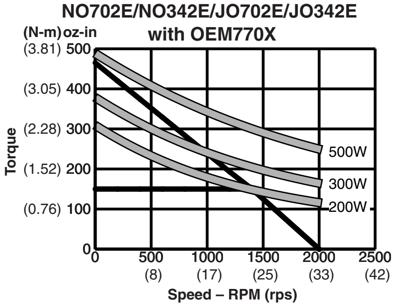
[1] For "E" encoder option (1000 ppr), maximum velocity is 6,000 rpm (100 rps).

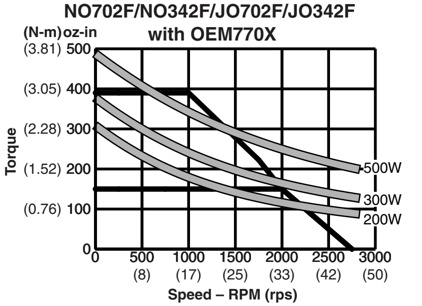
[2] With 75VDC bus voltage; 25°C (77°F) ambient temperature.

*Peak Power Curves: NeoMetric & J Series Motors* [1]









[1] With 75VDC bus voltage; 25°C (77°F) ambient temperature.

### *Example*

Use the peak power curves to choose a power supply to use with a system consisting of an OEM770X with an SM233B motor. The motor must accelerate with a torque of 200 oz-in (1.52 Nm), until it reaches a velocity of 1,500 rpm (25 rps). It then slews at constant velocity until it decelerates.

From the peak power curves, observe that this move requires approximately 300W peak power. Choose a power supply that provides at least 330W peak to accomplish this move. (330W includes an extra 10% for miscellaneous losses.)

### *Example*

A system must make a trapezoidal move, and reach 2,000 rpm (33.3 rps) at a torque of 125 oz-in (0.88 Nm). Which size 23 motor requires the smallest power supply to make this move?

From the peak power curves:

| Motor | Peak Power | Peak + 10% |
|---|---|---|
| SM230A | n/a | n/a |
| SM230B | n/a | n/a |
| SM231A | 340W | 374W |
| SM231B | 365W | 401W |
| SM232A | n/a | n/a |
| SM232B | 240W | 264W |
| SM233A | n/a | n/a |
| SM233B | 220W | 242W |

This move is beyond the speed/torque range of four motors. Of the remaining motors, the SM233B requires a 242W power supply to make the move. The other motors need larger power supplies.

## Friction, Gravity, and Different Move Profiles

The techniques we have discussed so far apply to trapezoidal moves with negligible friction. Below, we will briefly mention some salient points about other types of moves. If your system has moves similar to one of these, apply the techniques developed above to your application.

### *Friction*

The presence of friction requires additional torque to overcome the friction. We will consider Coulomb friction in a trapezoidal move. (Coulomb friction does not change with velocity. Viscous friction, which *does* depend on velocity, is much more difficult to analyze.)

During acceleration, total torque is equal to the torque required for acceleration plus the torque required to overcome friction.

$$T_t = T_a + T_f$$

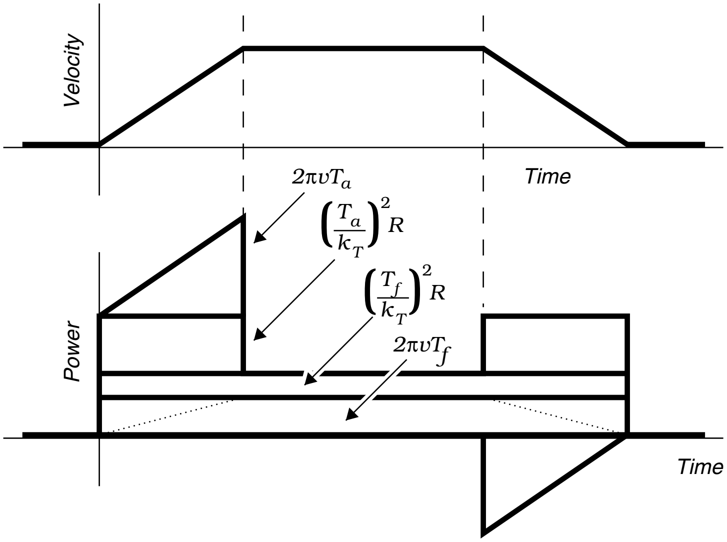
where

$T_t$ = Total Torque

$T_a$ = Acceleration Torque

$T_f$ = Friction Torque

The next drawing illustrates how friction affects a system.


Velocity
Time
$2\pi \upsilon T_a$
$\left(\frac{T_a}{k_T}\right)^2 R$
$\left(\frac{T_f}{k_T}\right)^2 R$
$2\pi \upsilon T_f$
Power
Time

*Friction*

Observe that friction adds additional plateaus to the drawing. The actual shape of the plateau due to frictional shaft power is shown by the dotted lines. For simplicity, we approximate the shape with a rectangle.

The equation for peak power becomes

$$P_{peak} = 2\pi v\left(T_a + T_f\right) + \left(\frac{T_a + T_f}{k_T}\right)^2 R$$

The power supply must also provide power while the motor is slewing at constant velocity. The equation for power during slew is:

$$P_{slew} = 2\pi v_{slew} T_f + \left(\frac{T_f}{k_T}\right)^2 R$$

You can use the peak power curves (discussed in the previous section) to predict the peak power and slew power that the power supply must provide. Be sure that you include the friction torque in the appropriate places, however. The next example illustrates this.

### *Example*

Determine peak power and slew power that an SM232B motor will require. Acceleration torque is 100 oz-in (0.71 Nm). Friction torque is 50 oz-in (0.35 Nm). The slew velocity is 2,000 rpm (33 rps).

Total torque during acceleration is 150 oz-in (1.06 Nm), the sum of acceleration and friction torque. On the curves, the intersection of 150 oz-in and 2,000 rpm lies on the 300W line.

During slew, the only torque present is friction torque. At 50 oz-in and 2,000 rpm, the curves show that 80W is required.

The power supply must be capable of providing at least 330W peak and 88W continuous power (these values include a 10% estimation factor).

### *Gravity*

We can distinguish two distinct situations when gravity is involved in an application.

- Lifting a load against gravity
- Lowering a load with gravity

These situations must be analyzed separately.

When your system lifts a load, gravity imposes a force downward. The motor must exert an additional torque to counteract this force. This is similar to a system that has friction, where the motor must exert an additional torque to overcome the friction. One possible difference can occur if the motor must provide holding torque while the load is stationary, to prevent the load from moving downward. In this case, the supply must provide power for the copper losses due to the holding torque, even when the motor is not moving.

The analysis for lowering a load can be much more complicated. The basic power equation can still be used, but you must take care to use the proper algebraic sign for the various torques, forces, velocities, etc.  A full analysis of the calculation method is beyond the scope of this text. The easiest way to determine your system's power needs may be the empirical method, discussed in the next section.

As an example of the complexity of the calculation involved, consider just one part of the move profile—acceleration from rest, with the load moving downward. Depending upon whether the acceleration is faster, slower, or equal to gravitational acceleration, net power can be positive, negative (regeneration), or even zero! Other parts of the move profile are equally complicated.

### *Other Move Profiles*

Many other move profiles and application conditions are possible. For example, moves can be sinusoidal, s–curve, or random, with or without friction, with or without or gravity.

To calculate power needs for moves such as these, you may be able to follow the methods we have developed above, and modify the equations to suit your application. Or, you may need to use the empirical method, presented below.

## POWER REQUIREMENTS—AN EMPIRICAL METHOD

You can use an empirical approach to measure the voltage and current going from a power supply to an OEM770X, and directly determine your system's power requirements.

You will need the following equipment:

- DC Current Probe
- Oscilloscope
- Large Power Supply

This method also requires that you make a prototype of your system.

### *Prototype Your System*

Make a working prototype of your system. For the power supply, temporarily use a large power supply that is capable of providing enough power for all the moves your system makes. The temporary power supply should operate at the same voltage at which you intend your final system to run. Once you determine the power requirements, you can replace the temporary power supply with a permanent one.

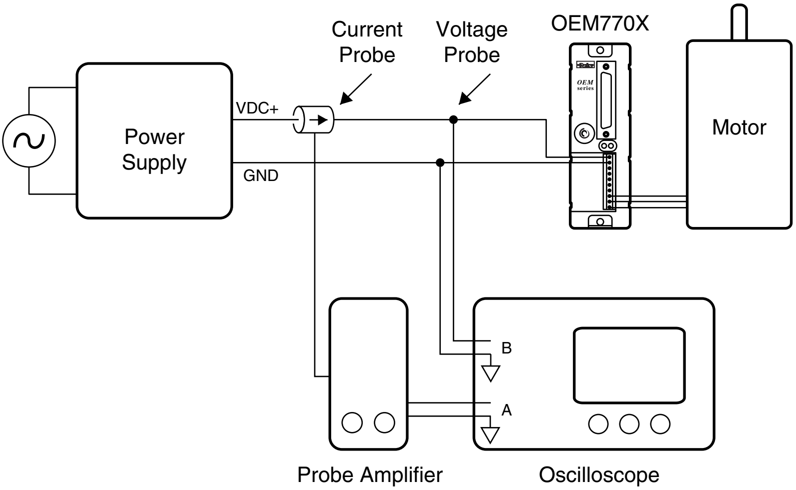
### *Measure Current*

Connect a current probe to one channel of an oscilloscope. (Connect the probe in the correct direction. With the motor at rest, the probe should measure *positive* current.) Measure current going from the power supply to the OEM770X while your system performs its moves under actual operating conditions. Current going from the OEM770X to the motor is not relevant in this procedure; you do not need to measure it.

Your current probe must be of the type that connects to an oscilloscope, and is fast enough to show current variations (such as a Tektronix A6302

Current Probe and AM 503 Current Probe Amplifier). The current probe in a digital multimeter will not work in this situation, nor will an AC current probe.

Connect an oscilloscope probe to the second channel of the oscilloscope, and use it to monitor power supply bus voltage.



*Setup for Current Measurement*

The bus voltage should drop no more than 10% during peak power events. If it drops more than 10%, use a larger power supply.

### *Determine Power Needs*

At any moment the power used by your system is

$$P = V_{supply} I_{supply}$$

When the current is positive, current flows from the supply to the drive, and the supply delivers power to your system. When current is negative, the system is regenerating—power flows from your system, and back into the supply.

To determine the peak power that the supply must deliver, measure the highest current (as seen on the oscilloscope screen). Substitute this current in the power equation, to get:

$$P_{peak} = V_{supply} I_{peak}$$

Once you know the peak power that your system demands, you can select a supply that can deliver enough power.

### AVERAGE POWER CALCULATIONS

Many power supplies have a peak power rating and an average power rating. The peak power may be much higher than the average power rating.

For example, the OEM300 Power Module can deliver 300W peak for 30 seconds, at a 10% duty cycle. It can deliver 200W continuously.

To determine the average power in your system, calculate the area under the graph of power, and multiply by the repetition frequency.

***Example***

Consider a trapezoidal move with acceleration *a*, velocity *v*, and repetition frequency $f_{rep}$. Ignore friction, and assume that regeneration provides power for deceleration. Therefore, the power supply only delivers power during acceleration.

The average power is

$$P_{avg} = f_{rep} \frac{v}{a} \left[ \tfrac{1}{2}(2\pi vT) + \left( \frac{T}{k_T} \right)^2 R \right]$$

If your system needs power to decelerate, you should add a term to the equation that represents power needed to decelerate, and include this power in the average.

## Regeneration

At certain times during a move, particularly during deceleration or while lowering a load, energy can be transferred from the motor and load, and back to the power supply. This is called *regeneration*.

The following sections will describe methods to calculate the power and energy that regeneration can produce during deceleration in a trapezoidal move. You can use this information to help you select a power supply that can deal with regenerated energy.

## POWER FLOW DURING DECELERATION

In the trapezoidal moves we have analyzed, we used the convention that torque and velocity are positive during acceleration. During deceleration, however, torque is applied in the opposite direction. Therefore, torque is negative, and shaft power, the product of torque and shaft velocity, is also negative.

$$P_{shaft} = \omega(-T) = 2\pi v(-T)$$

Negative shaft power means that power flows from the motor back to the drive. Does this mean that deceleration always causes regeneration? Not necessarily. Current must flow in the motor to produce the negative torque. The heat that this current produces is proportional to the *square* of the torque. Copper losses, therefore, are always positive.

$$P_{copper} = \left(\frac{-T}{k_T}\right)^2 R$$

The total power during deceleration, then, is the sum of shaft power and copper losses.

$$P_{decel} = -2\pi vT + \left(\frac{-T}{k_T}\right)^2 R$$

If the magnitude of the first term is larger than the magnitude of the second term, then the net power is negative—power will flow from the system, and back into the power supply. When the second term is larger than the first, the power supply must provide power for deceleration.

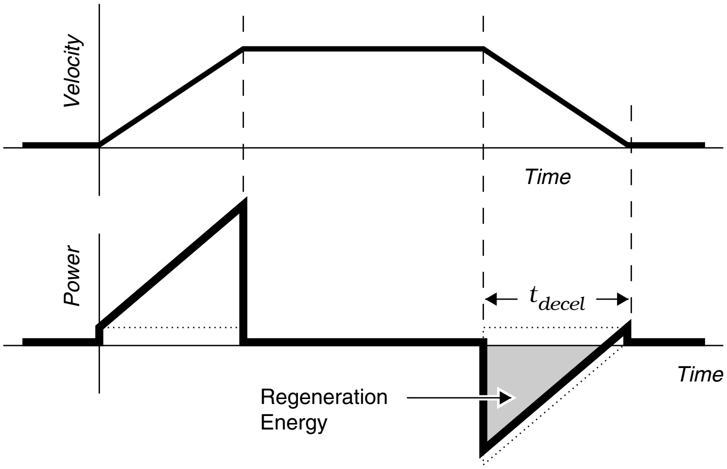
## ENERGY DURING REGENERATION

The power supply must be capable of absorbing or dissipating energy that flows into it during regeneration. The amount of energy is related to the power that we discussed above.

Recall from physics that the *joule* is the unit of energy in the SI system, and that power is the rate of energy flow. One watt is equal to an energy flow of one joule per second.

1 watt = 1 joule/second

Energy is also the integral of power. Therefore, you can determine the total energy produced during deceleration by finding the area under the peak power curve. The next drawing shows this area, for a situation where copper losses are small, and shaft power is large.

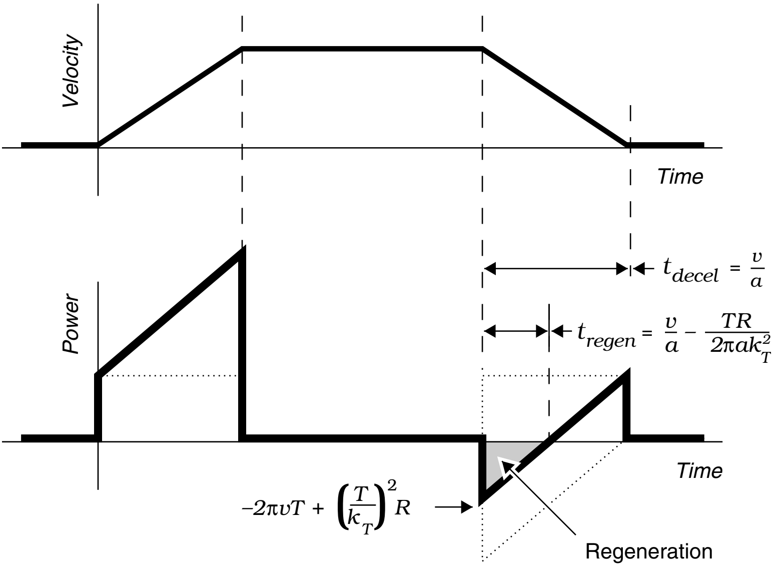


*Regeneration with Low Torque*

To approximate the total energy from regeneration, find the area of the triangle representing shaft power. You can ignore the copper losses, because they are small.

$$E_{regen} = \tfrac{1}{2} \cdot base \cdot height = \tfrac{1}{2}(-2\pi vT)(t_{decel})\ , in\ joules$$

The next drawing shows the deceleration portion of a move that uses a higher torque to decelerate the motor. Consequently, the copper losses are greater.


Velocity
Time
Power
Time
$t_{decel} = \frac{v}{a}$
$t_{regen} = \frac{v}{a} - \frac{TR}{2\pi a k_T^2}$
$-2\pi v T + \left(\frac{T}{k_T}\right)^2 R$
Regeneration Energy

*Regeneration with High Torque*

If you ignore copper losses when you calculate energy from regeneration in this type of situation, the answer will be much larger than the actual energy produced. To accurately calculate the energy, use the next equation to find the area of the regeneration triangle.

$$E_{regen} = -\tfrac{1}{2}\left[2\pi v T - \left(\frac{T}{k_T}\right)^2 R\right]\left[\frac{v}{a} - \frac{TR}{2\pi a k_T^2}\right] \text{, } \textit{in joules}$$

In this equation, *v* is the slew velocity, and *a* is the deceleration rate.

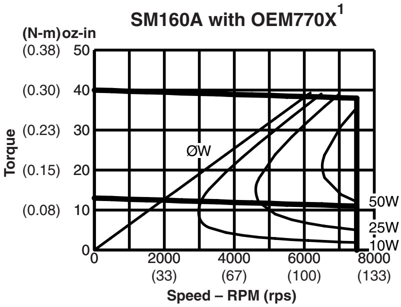
## Regeneration Curves

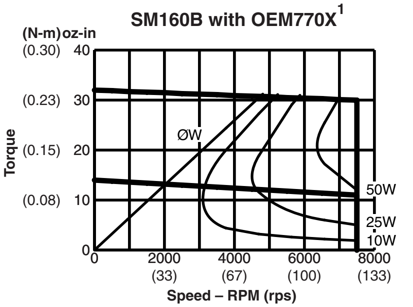
In the following version of the regeneration equation:

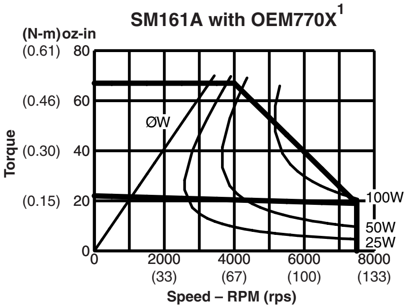
$$v = \frac{P + \left(\frac{T}{k_T}\right)^2 R}{2\pi T}$$

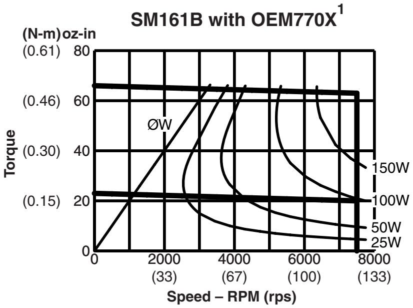
If we set power equal to a specific value, and solve for velocity at various torques, we can plot a family of curves that represent peak regeneration watts. We have done this on the following page for Compumotor servo motors.

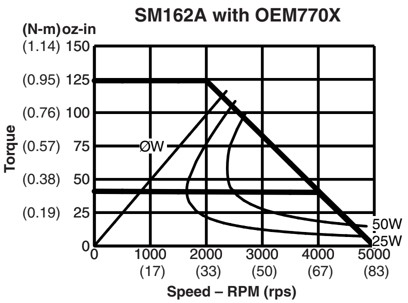
*Peak Regeneration Curves: SM Motors, Frame Size 16* [2]

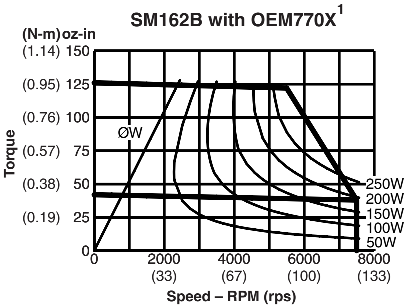










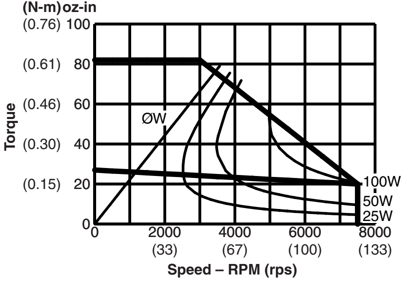


[1] For "E" encoder option (1000 ppr), maximum velocity is 6,000 rpm (100 rps).

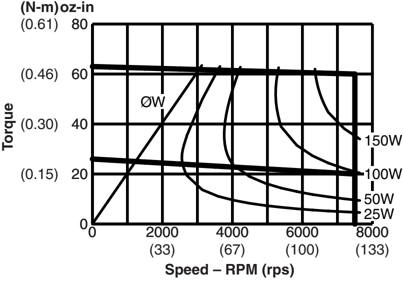
[2] With 75VDC bus voltage; 25°C (77°F) ambient temperature.

## Peak Regeneration Curves: SM Motors, Frame Size 23 [2]

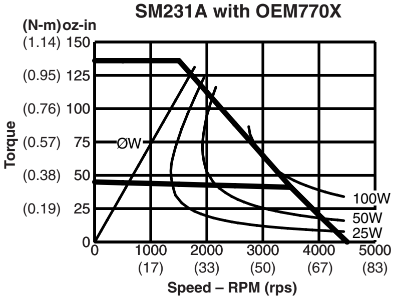
### SM230A with OEM770X[1]



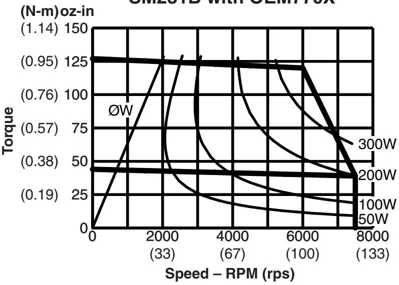
### SM230B with OEM770X[1]



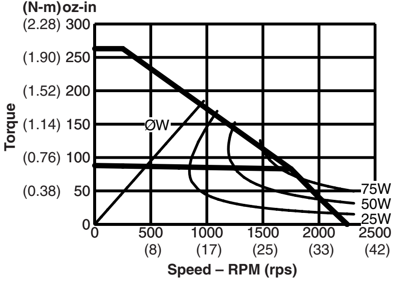
### SM231A with OEM770X



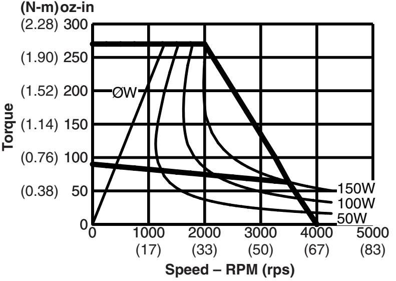
### SM231B with OEM770X[1]



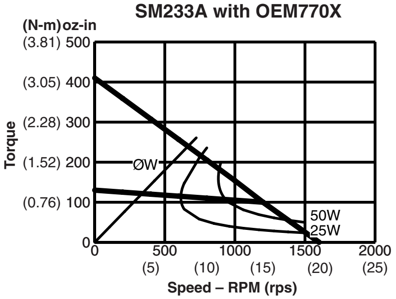
### SM232A with OEM770X



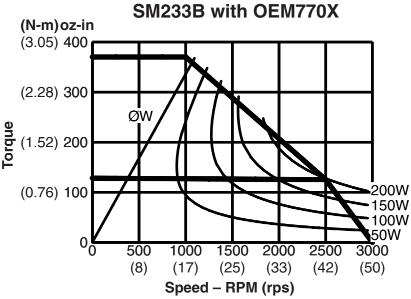
### SM232B with OEM770X



### SM233A with OEM770X



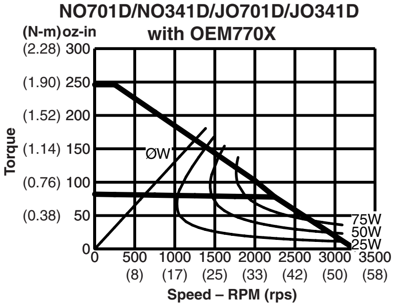
### SM233B with OEM770X

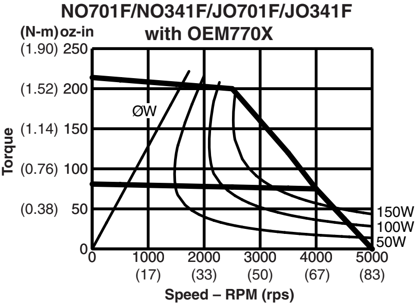


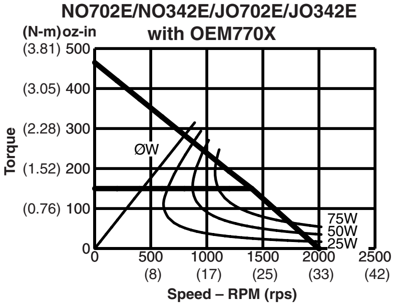
[1] For "E" encoder option (1000 ppr), maximum velocity is 6,000 rpm (100 rps).

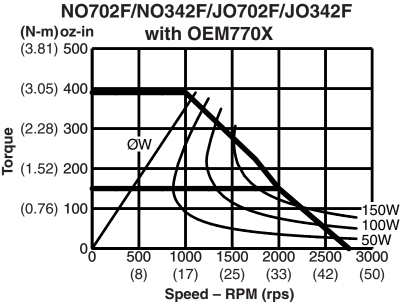
[2] With 75VDC bus voltage; 25°C (77°F) ambient temperature.

## *Peak Regeneration Curves: NeoMetric & J Series Motors* [1]









[1] With 75VDC bus voltage; 25°C (77°F) ambient temperature.

On each of these charts, there is a straight line corresponding to zero watts. This is where

$$2\pi v T = \left(\frac{T}{k_T}\right)^2 R$$

In areas to the left of this line, copper losses are always greater than shaft power, and the power supply must always provide power. In other words, for any move to the left of this line, the power supply will not receive regeneration energy from the system, because copper losses will be greater than negative shaft power.

***Example***

An SM232B motor performs a trapezoidal move. It slews at 50 rps, and decelerates at 100 rps$^{-2}$ with a torque of 75 oz-in (0.53 Nm). Does the power supply receive regenerated energy? If so, how much? The motor has a torque constant $k_T$ = 0.169 Nm/A, and a resistance $R$ = 2.01 ohms.

Using the regeneration equation,

$$\begin{aligned}
E_{regen} &= \tfrac{1}{2}\left[2\pi vT - \left(\frac{T}{k_T}\right)^2 R\right]\left[\frac{v}{a} - \frac{TR}{2\pi a k_T^2}\right] \\
&= \tfrac{1}{2}\left[2\pi(50)(0.53) - \left(\frac{0.53}{0.169}\right)^2 2.01\right]\left[\frac{50}{100} - \frac{(0.53)2.01}{2\pi(100)(0.169)^2}\right] \\
&= \tfrac{1}{2}[166.5 - 19.8][0.5 - 0.06] \\
&= \tfrac{1}{2}[146.7 \quad \text{watts}][0.44 \quad \text{seconds}] \\
&= 32.3 \quad \text{joules}
\end{aligned}$$

At the moment deceleration began, the peak regenerated shaft power was 166.5W, and copper losses were 19.8W. The peak regeneration power was therefore 146.7W, which you can also read directly from the chart for the SM232B motor. To determine regeneration energy (joules), however, you need to perform the calculation.

The last term in the equation shows that total deceleration time (*v/a*) was 0.5 seconds. The power supply received regenerated energy for the first 0.44 seconds, and had to supply power for the final 0.6 seconds.

## What Voltage Do You Need?

The OEM770X uses the DC power supply voltage as the supply voltage for the motor. The motor's performance depends on the voltage at which it runs. Therefore, the power supply voltage you choose will affect motor performance. We will use Compumotor servo motors as examples to illustrate this, but the points presented below apply to any servo motor.

Because the OEM770X accepts such a wide range of input voltage (24 – 75VDC), you have several options for choosing a power supply voltage. These options are explained on the following pages.

## MATCH THE POWER SUPPLY TO THE MOTOR

Manufacturers wind servo motors for optimum performance at a specific voltage. They publish speed/torque curves measured at that voltage. If you select a motor because you need the performance shown in the curves, choose a power supply that produces at least as much voltage as that for which the motor was designed.

For example, Compumotor servo motors specified in this user guide are wound for 75VDC operation. The speed/torque curves were measured with a 75VDC power supply. If you want the full performance shown in the curves, use a power supply that operates at 75 volts.

## USE AVAILABLE POWER, AND CUSTOM WIND A MOTOR

In many machines, the motion control system is but one component among many in the entire machine. Power may be available from a large power supply that runs other parts of the machine. We designed the OEM770X so that you can take advantage of available power.

If power is available, but at a voltage lower than specified for the motor you have chosen, you can contact the manufacturer to see if the motor can be made with the voltage rating you need. Motor manufacturers can design a motor's windings so that it can have similar performance characteristics at different voltages.

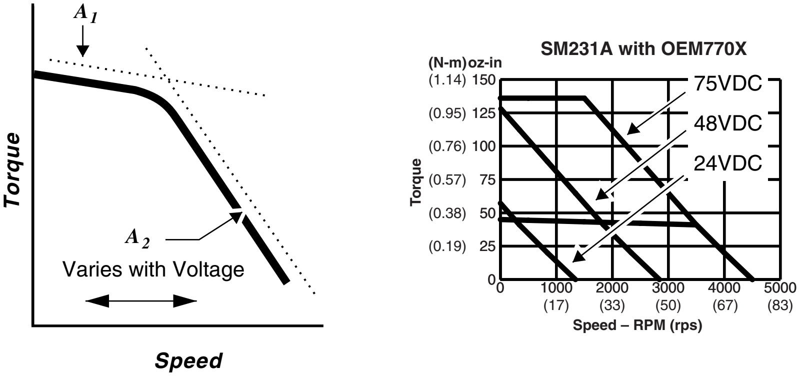
For example, suppose you decide to use the SM231A motor. You want to make moves that lie within the 75VDC speed/torque curve, but you only have 48VDC available. If you cannot get the performance you need from the standard motor at 48VDC, you should call Compumotor. We can make the motor with a special winding to obtain performance similar to that shown in the 75VDC speed/torque curve, but at 48VDC.

## USE AVAILABLE POWER AND AN AVAILABLE MOTOR

You can use a power supply whose voltage is less than the voltage at which your motor's speed/torque curve was specified. The motor will not be able to perform the full range of moves shown on the speed/torque curve, however.

The next drawing shows how varying the power supply voltage affects a motor's speed/torque curve. The speed/torque curve can be approximated by two asymptotes, labeled $A_1$ and $A_2$ in the curve on the left. $A_1$ is not affected by voltage changes, but $A_2$ is. As the voltage is decreased, $A_2$ will shift to the left. The slope of $A_2$ will not change.

$A_2$ will move a distance proportional to the decrease in voltage. If the voltage is cut in half, $A_2$ will move halfway to the origin. If voltage is reduced by two thirds, $A_2$ will move two thirds of the way toward the origin.



*Voltage Affects the Speed/Torque Curve*

To illustrate how voltage affects performance for a specific motor, the drawing shows the speed/torque curve for the SM231A motor at 75VDC, 48VDC, and 24VDC.

## Power Supply Choices

If you have worked through the previous sections, then by this point you have:

- Determined how much power your system needs.
- Determined whether regeneration is a concern.
- Selected a power supply voltage.

Armed with this information, you are now ready to choose a power supply! You have three main choices:

- Linear Unregulated Power Supply (OEM1000)
- Switching Power Supply
- OEM300 Power Module

In the following sections, we will explain the advantages and disadvantages of linear and switching supplies. We will also present information about Compumotor's OEM300 Power Module and OEM1000 Power Supply.

## LINEAR POWER SUPPLY

The simplest linear power supply consists of a transformer, bridge rectifier, and capacitor. The transformer changes the level of the AC input voltage. Diodes in the rectifier change the AC to DC. The capacitor filters the DC, and stores energy. Such linear supplies are unregulated.

Some models have a fuse to provide overcurrent protection. To improve the transient response, the single output capacitor can be replaced by combinations of capacitors and inductors.

Compumotor's OEM1000 is a linear power supply.

### *Advantages of Linear Power Supplies*

- **Simplicity** – Linear supplies are simple, robust, and repairable. They have very few parts. Once the supply is working, it usually keeps working for a long time. If a part fails, diagnosing the failure is straightforward, and the part can be replaced.

- **Low Cost** – In many applications, a linear supply costs less than a switching supply. (This depends upon power level and number of units.)

- **Low Noise** – Linear supplies are virtually free of electrical noise, and give excellent results in noise-sensitive applications.

### *Disadvantages of Linear Power Supplies*

- **Poor Line Regulation** – If the input line voltage rises or falls, the power supply’s output voltage will also rise or fall.

- **Poor Load Regulation** – When the load uses more power, the power supply’s output voltage may drop.

- **Voltage Ripple** – Large ripple voltage in the output requires a relatively large output capacitor for smoothing.

- **Large Size** – Compared to a switching supply of the same power level, a linear supply is larger, heavier, and takes up more space.

- **Low Efficiency** – The linear supply suffers losses in the transformer and other components. This dissipation can result in heat and higher operating temperatures.

- **Slow Transient Response** – The linear supply may not be capable of keeping up with the rapidly changing load requirements of some servo systems. Designing a linear supply for a high performance system can be quite complex.

### *Regeneration and Linear Power Supplies*

Dealing with regeneration is simpler with linear supplies than with switching supplies. The linear supply's transformer and rectifier will continue to operate during regeneration.

During regeneration, the supply's capacitors will absorb energy from the load. As the energy is stored in the capacitors, the supply's output voltage will rise. If it goes higher than the threshold of 90VDC, the OEM770X's overvoltage protection will disable the drive. To avoid overvoltage shutdowns, you can use larger capacitors to store more energy, or use a power supply that operates at a lower bus voltage.

## SWITCHING POWER SUPPLY

A switching power supply takes an AC input voltage at power line frequency, and uses switching transistors to increase the frequency. Various techniques are used to modify the high frequency voltage and obtain the desired DC output voltage. The chief advantage of operating at higher frequency is that many components, particularly transformers and capacitors, can be much smaller, and operate more efficiently.

A switching power supply is regulated. It actively monitors the input line voltage, and keeps its output voltage constant, even when the input voltage varies. If the load demands more power, the supply will increase its output current, but its output voltage will stay at a constant level.

### *Advantages of Switching Power Supplies*

- **Regulation** – The supply will try to keep its output at a constant voltage, regardless of line or load variations. (There are limitations on how well it can do this.)

- **Small Voltage Ripple** – The output voltage ripple is small, and at a high frequency. Therefore, a relatively smaller output capacitor can be used for smoothing.

- **Small Size** – A switching supply will be much smaller than a linear supply of the same power rating.

- **Efficiency** – Switching supplies are efficient—they dissipate less power as heat than linear supplies.

- **Fast Transient Response** – Because a switching supply monitors its output, it can quickly adapt its performance to provide changing amounts of power for changing load conditions. (Power supply transient response depends upon the supply's design.)

### *Disadvantages of Switching Power Supplies*

- **High Cost** – In most applications, a switching supply will cost more than a linear supply. (Depends upon power level and number of units.)

- **Electrical Noise**– Switching supplies produce electrical noise, which may be transmitted to load equipment and power lines. They may not be suitable for noise-sensitive applications.

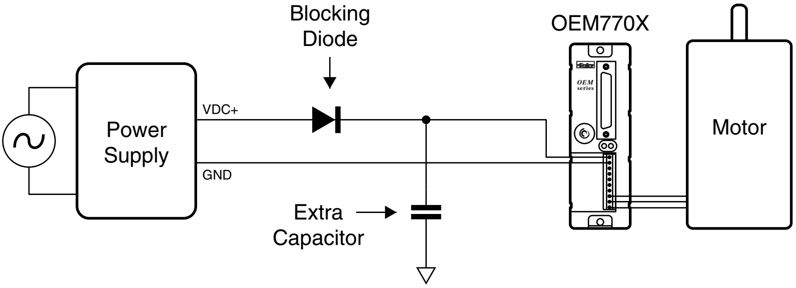
- **Less Reliable** – Switching supplies are much more complex than linear supplies. More components means that more things can go wrong. Consequently, the time before failure may be shorter for switching supplies.

- **Less Repairable** – If a switching supply fails, it usually can only be repaired by its manufacturer. The user probably cannot repair it, and may need to replace the entire unit.

### *Regeneration and Switching Power Supplies*

Regenerated energy flowing from the load to a switching supply may cause the supply to behave erratically and unpredictably. Accommodating regeneration is more difficult with a switching supply than with a linear supply.

You may need to install a *blocking diode* if regeneration causes problems with your switching supply. The next drawing shows where the diode should be positioned.

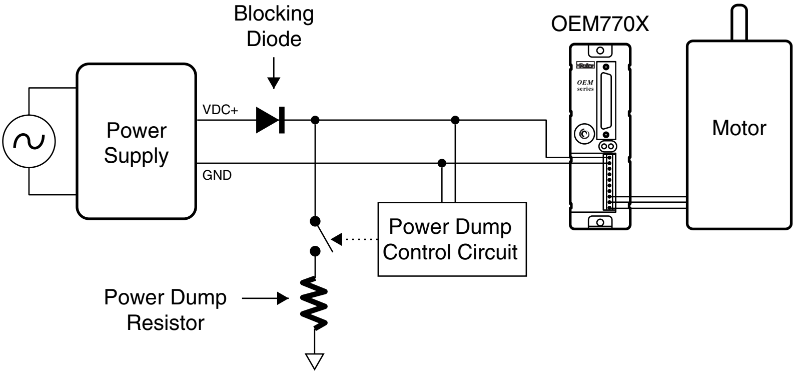


*Blocking Diode with Extra Capacitor*

The blocking diode will prevent regenerated energy from entering the power supply. This energy *must* go somewhere. If it is not absorbed by the supply, it will charge up the drive's internal capacitors, and cause an overvoltage fault. (In a vertical application, it may damage the drive.)

The drawing above shows one possibility for removing regenerated energy. You can install extra capacitors on the power bus, and allow the energy to charge up the capacitors.

The next drawing shows another possibility for removing regenerated energy. You can install a power dump resistor, and circuitry to monitor the voltage on the power bus.



*Blocking Diode with Power Dump*

Design the circuit so that when regeneration causes a voltage rise, the power dump will turn on and dissipate regenerated energy in the resistor.

### OEM300 Power Module

The OEM300 Power Module is a Compumotor product that contains a switching power supply, and several additional circuits that make it an ideal power supply for many servo applications. Its features are summarized below. For additional information, contact Compumotor at 800-358-9070, and request a copy of the OEM300 User Guide.

***Power Supply***

The switching power supply in the OEM300 has characteristics that are highly compatible with OEM Series servo drives and microstepping drives. It can provide 300W peak/200W continuous power, at 4.0A/2.7A, respectively. The transient response of the OEM300 is matched to that of OEM Series drives.

***Power Dump***

The OEM300 contains a power dump circuit that turns on at 85VDC. The power dump can dissipate as much as 400 joules of energy, at a peak dissipation rate of 722.5 watts.

***Short Circuit Protection***

The OEM300 will shut down its output if its current exceeds 9 amps.

***Overtemperature Protection***

An internal temperature sensor will shut down the OEM300 if its temperature reaches 60°C (140°F).

***Overvoltage Protection***

The OEM300 will shut down its output if an overvoltage condition lasts longer than 0.5 seconds.

## Powering Multiple Axes

So far in this chapter, we have presented several methods for choosing a power supply for a single axis system—one drive and one motor. You can also use a supply to provide power to multiple axes.

To choose a power supply for multiple axis operation, the first step is to determine the power each individual axis requires, using any of the methods we presented above.

Next, determine how the power requirement of each axis relates, in time, to the other axes. There are two possibilities: each axis moves independently; or, the various axes move in a coordinated way, with the motion of each axis depending upon the other axes.

For independent moves, the largest power demand will occur if all axes simultaneously reach their peak power points. Choose a power supply that can provide enough power for this peak demand.

For dependent moves, find the times when the maximum power is required. Add together the power requirements for each axis at these times, to find the peak power requirement. Choose a power supply that can satisfy the peak requirement.

# CHAPTER 7

## Troubleshooting

When a problem occurs in your system, use the following strategy to isolate and identify the problem:

- Check Light Emitting Diodes (LEDs) and the Fault Output for an indication of the cause of the problem.
- Check other possible causes.

When we refer to LEDs,

- **ON** means *illuminated.*
- **OFF** means *not illuminated.*

When we refer to the Fault Output,

- **HIGH** means +5VDC.
- **LOW** means ground, or ØVDC to Ø.8VDC.

The next table summarizes LED & Fault Output information. The table after that summarizes other possible sources of problems. Detailed troubleshooting procedures follow the tables.

## Check LEDs First!

If you encounter problems, you may be able to quickly identify the problem by looking at the LEDs and the fault output. The next table summarizes possible LED and fault output states.

### LEDs and FAULT OUTPUT

| GREEN LED | RED LED | FAULT OUT | CONDITION |
|---|---|---|---|
| OFF | OFF | HI | No Power; or<br>Short on Hall +5VDC or short on Encoder +5VDC |
| ON | OFF | LO | Normal operating condition |
| ON | INT | LO | Foldback |
| ON | ON | LO | Foldback (Red LED turns off within 10 seconds if command input is reduced to ØV) |
| ON | INT | INT | Normal condition while drive is powering up, or turning off; or,<br>Weak power supply, VDC is too low for operation |
| ON | ON | HI | Power supply undervoltage; or,<br>No enable |
| | ↑ | | ***NOT LATCHED*** Can recover from above conditions without cycling power. |
| | ↓ | | ***LATCHED*** Cycle power to reset drive, and recover from conditions below. |
| ON | ON | HI | Overvoltage from regeneration; or,<br>Overtemperature |
| OFF | ON | HI | Short circuit in load or cables; or bad Hall state (all high or low); or,<br>Power supply fault (typically, a transient undervoltage) |
| | | | |
| ***Legend*** | | | **ON** = LED is ON (Illuminated); or,<br>**HI** = Fault Output is HIGH (+5VDC) |
| | | | **OFF** = LED is OFF ( Not Illuminated); or,<br>**LO** = Fault Output is LOW (ØVDC, or Ground) |
| | | | **INT** = LED turns ON, then turns OFF; or,<br>**INT** = Fault Output goes LOW, then goes HIGH |

For a detailed description of the various fault conditions, see the basic troubleshooting procedure below.

## Other Possible Problems

If the drive is powered up, enabled, and operating properly:

- The green LED is ON
- The red LED is OFF
- The fault output is LOW

These conditions indicate that the OEM770X is probably not the source of the problem. The next table summarizes other possible sources of problems.

### TROUBLESHOOTING TABLE

| *Possible Source of Problem* | *SOLUTION* |
|---|---|
| **PROGRAM** | Verify motion is being commanded. |
| **MOTOR** | Check for motor problems. Check motor coils for continuity, shorts, proper resistance. Check Hall and Phase wiring. |
| **MECHANICAL SYS.** | Check for jams, binds, increased friction, etc. |
| **WIRING** | Check motor wiring: Phases, Hall Effects.<br>Check power supply wiring. |
| **OVERHEATING** | Verify that drive's heatplate has good thermal contact with heatsink. Check mounting screws. Provide sufficient ventilation. |
| **POWER SUPPLY** | Verify power supply delivers enough power during entire move, without undervoltage, or overvoltage caused by regeneration. |
| **MOVE PROBLEMS** | Check speed/torque limitations. Check for excessive friction, regeneration, problems with gravity, transient undervoltage, etc. |
| **RS232 PROBLEMS** | Verify proper communications. |
| **ELECTRICAL NOISE** | Check for problems caused by electrical noise. Consult Compumotor's *EMC Installation Guide* for possible solutions. |

Details on these problems are discussed after the next section.

| Symptoms | Possible Causes | Solutions | Useful Commands |
|---|---|---|---|
| No motion | Limits active | Check hard wiring<br>Disable limits | R,RA,LD |
| | Exceeded maximum distance range | Reset absolute counter with PZ command | PZ, D, MPI, PR |
| | Position error (red fault light is on) | CPE command too small<br>Check encoder wiring | R,RSE,CPE,DP E |
| | Already executing a command | Check status commands | R,RB |
| | Servo gains too small | Increase gains | CPG,CDG,CIL, CIG |
| | Stop, kill, trigger inputs active | Check wiring to inputs<br>Check input setup | R,RB,IN,IS |
| | Loop, pause active | Check status commands | R,RB |
| | Incorrect unit address | Verify unit address<br>Re-issue unit address | R,# |
| | In absolute mode (already at position) | Issue new or different absolute distance<br>Issue MPI command | R,MPI,MPA |
| | Unit shutdown (red fault light is on) | Enable drive<br>Verify hardwire enable input is shorted | ST, ON<br>R,IS |
| No communications | RS-232 miswired | Check wiring for RS-232 | |
| | Wrong RS232 settings | 9600 Baud, 8 data bits, 1 stop bit, no parity | |
| | F command enabled | Issue an E command | E,F |
| Motor runs away | Encoder miswired | Check encoder wiring<br>Verify power to encoder | |
| | Controller not receiving encoder pulses | With drive disabled turn shaft clockwise (front of motor) PX should read positive counts | ST0,OFF,ST1, ON,DPA,PX |
| Fault LED & output active | Position error | Increase allowable position error<br>Decrease moveparameters | CPE,DPE, RSE A,V |
| | Unit disabled | Enable unit | ST0,ON |
| Unit not responding to commands | Defined a sequence and didn't close with an XT | Issue an XT at the end of a sequence | XD,XT |
| | Communications disabled | Enable communications | E,F |

## RS-232C PROBLEMS

Use the following procedure to troubleshoot communication problems that you may have with the OEM770X.

1. Be sure the host computer's transmit (Tx) wire is wired to the peripheral's receive (Rx) connection, and the host computer's receive (Rx) wire is wired to the peripheral's transmit (Tx) connection. Switch the receive and transmit wires on either the host or peripheral if the problem persists.

**CAUTION**

OEM770X Rx, Tx, and GND pin outs are not 2, 3, and 7 like most devices.

2. Confirm that the host and peripheral are configured for the same baud rate, 8 data bits, 1 stop bit, and no parity.

3. Use DC common or signal ground as a reference, not earth ground.

4. Cable lengths should not exceed 50 feet (15 m) unless you are using some form of line driver, optical coupler, or shield. As with any control signal, be sure to shield the cable-to-earth ground at one end only.

5. To test the terminal or terminal emulation software and the RS-232C cable for proper three-wire communication, unhook the OEM770X and enter a character. You should not receive an echoed character. If you do, you are in half duplex mode. Connect the host's transmit and receive lines together and send another character. You should receive the echoed character. If not, consult the manufacturer of the host's serial interface for proper pin outs.

6. *The following applies only to firmware 92-016637-01 or higher*:
   If the OEM770X echoes back **&** for each byte sent to it, a data communication error has occured. To re-establish communication, see the **%** command.

7. *The following applies only to firmware 92-016637-01 or higher*:
   To extend cable length and/or improve noise immunity, an RS-485 option is available as a custom product. Contact Compumotor's Custom Products Department for details.

## SOFTWARE DEBUGGING TIPS

This section offers helpful tips for debugging programs and understanding errors and fault conditions. The OEM770X has several tools that you can use to debug a problem in the system. The software tools are:

**R**—Report Status
**RA**—Limit Switch Status Report
**RB**—Loop, Pause, Shutdown, Trigger Status report
**RSE**—Report Servo Errors
**IS**—Input Status Report
**BS**—Buffer Status Report
**B**—Buffer Status Report

The troubleshooting table also offers possible causes for typical symptoms.

### ENCODER PROBLEMS

Since the OEM770X relies on feedback information, encoder connections are critical for the unit to operate properly. If you suspect the OEM770X is not receiving good position data, use the following procedure to verify.

1. Disable the drive (**ST1** or **OFF**).
2. Enter a **PZ** command.
3. Rotate the motor CW by hand approximately one revolution.
4. Enter a PX. It should read approximately 4000 (for a 1000 line encoder).
5. Rotate CCW one revolution.
6. Enter PX command. It should read approximately 0.

If your OEM770X did not respond with similar values, it is not receiving encoder information. Either the encoder is miswired or in need of repair. If the green LED is *off*, verify that the Encoder/Hall +5V power is present. If not, repair encoder and Hall wiring and cycle power.

## Homing Diagrams

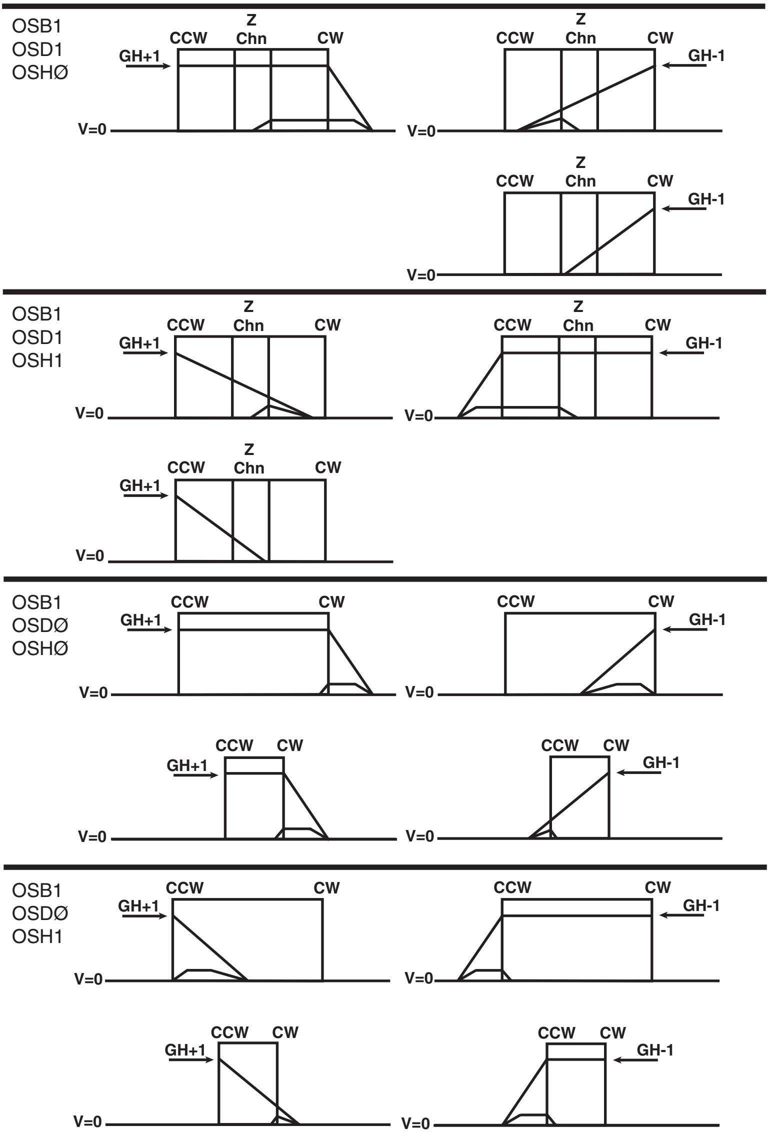
The following diagrams are examples of the many possible homing set-ups. Your parameters may vary and the results may vary slightly depending on your settings.

The CW side of the home pulse is the side closest to the CW limit. The CCW side of the home pulse is the side closest to the CCW limit.

The long pulse diagrams are indicative of situations where the motor decelerates while remaining inside the home pulse width due to the rapid homing deceleration or a very wide home pulse. The short pulse diagrams are indicative of situations where the motor decelerates through the home pulse width due to slow deceleration or a very narrow pulse width.

If an end-of-travel limit is hit during the initial homing, refer to the homing diagram for the opposite direction of travel.

The diagrams are drawn as a general guide. Velocity levels and slopes are drawn to indicate the general move profile the motor will make during the go home move. The vertical axis is velocity and the horizontal axis is position in relation to the home input transitions. Some lines are drawn as closely as possible together to indicate identical velocities, yet remain discernible.



## Basic Troubleshooting Method

To identify the cause of a problem, find the condition below that matches your situation. Then follow the detailed procedure listed under that condition.

### ARE BOTH LEDS OFF?

***Possible Problems:***

- No power from power supply

***Procedure***

1. Remove power. Disconnect all wiring except VDC+ and VDC–. Reapply power. Verify that power supply voltage is in the 24VDC–75VDC range. Is the green LED now on?
2. If the green LED is still off, return the drive to Compumotor.

### IS THE GREEN LED OFF, AND RED LED ON?

***Possible problem:***

- Short circuit in motor or cabling

***Procedure***

1. Remove power.
2. Disconnect all wiring except VDC+ and VDC–.
3. Reapply power.
4. Green LED should now be on, and red LED should be off. This indicates the problem is a short circuit in the cabling or motor.
5. Fix the short, and cycle power.

***Possible problem:***

- Bad Hall state (all three HIGH or all three LOW)

***Procedure***

1. Remove power.
2. Disconnect all wiring except VDC+ and VDC–.
3. Connect a jumper wire from any Hall input to HALL GND.
4. Apply power. The green LED should now be *on*.

5. Next, remove power again. Connect Hall wires to motor (Hall 1, Hall 2, Hall 3, Hall GND, Hall +5). *Do not connect motor phase wires.*
6. Apply power.
7. If green LED is off, and red LED is ON, then problem is a bad Hall state (all three HIGH or all three LOW). Possible causes are Hall miswiring, a damaged motor, or a short in Hall or encoder power wiring. Check Hall wiring, and voltage levels at Hall terminals. Check motor for faulty Hall sensors.

8. Measure HALL +5V with respect to HALL GND. If there is no HALL +5V, disconnect Encoder +5V and cycle power. If HALL +5V returns after you cycle power, then the encoder power wiring is bad (possibly a short on Encoder +5 wiring).

***Possible problem:***

- Power supply undervoltage during move

***Procedure***

1. Cycle power. Green LED should now be on, red LED off.
2. Make the move.
3. If the move causes a fault, the problem is probably a power supply undervoltage during the move. Try a larger power supply.

## Is Green LED ON, Red LED OFF—But No Motion?

These conditions indicate that the OEM770X is powered up, enabled, and operating properly. It is probably not the source of the problem. Look for the cause of the problem elsewhere in your system.

***Possible problems:***

- Wrong motor phase wiring
- Wrong motor Hall effect wiring
- Mechanical jam
- Program is not commanding motion

***Procedure***

1. Verify program is commanding motion.
2. Try to rotate the shaft manually. If you can, then the motor phases are probably miswired. Or, the motor may be damaged—check its phases for proper resistance, continuity, shorts, shorts to the case, etc.

3. If you cannot rotate the shaft, disable the drive. Try to rotate the shaft manually.
4. If you can rotate the shaft, then Hall wires are probably miswired. Check them, and check the motor temperature. (Without proper Hall inputs, the drive may command maximum current and overheat the motor—but no motion will result.)
5. If you cannot rotate the shaft, the machine is mechanically jammed.

## ARE GREEN AND RED LEDS BOTH ON?

***Possible problems:***

- Not Enabled
- Foldback
- Power Supply problem
- Overvoltage
- Overtemperature

***Procedure***

1. Check the enable input to see if it is low (grounded). If not, then the drive is not enabled.

2. With the drive enabled, reduce command input to ØVDC. If the red LED goes out within 10 seconds, then foldback was the problem. Check motor temperature. Check for a mechanical jam in your system.

   (***NOTE***: The fault output stays LOW during foldback. Foldback is the only condition that turns the red LED ON, but keeps the fault output LOW)

3. If red LED is still on (with a ØVDC command input), measure power supply voltage at the drive terminals, VDC+ and VDC–. It should be in the 24VDC – 75VDC range. If not, there is a power supply or power cabling problem.

4. With proper power supply voltage at the drive, measure the temperature of the drive's heatplate. Is it hot? If so, the problem could be an overtemperature shutdown. Wait 30 minutes for the drive to cool. Check for proper drive mounting and heatsinking. Check for a mechanical jam. When the drive has cooled, cycle power to resume operations. If overheating persistently causes shutdowns, you can try several remedies: change move profile or duty cycle; improve drive

mounting or heatsinking; reduce drive ambient temperature; add forced air cooling.

5. With proper power supply voltage at the drive, and if the drive is not hot, the problem could be an overvoltage fault. Regeneration during deceleration could have caused the overvoltage fault. Cycle power to resume operations. If regeneration repeatedly causes overvoltage faults, you can try several remedies to solve the problem: reduce deceleration rate; reduce bus voltage; add bus capacitance; add power dump circuitry.

   ***NOTE***: Overvoltage and overtemperature faults both have identical indicators: red and green LEDs both ON; fault output HIGH; fault condition is *latched*. To distinguish between the two faults, monitor conditions while the drive runs. Monitor heatplate temperature to see if it gets too high, which could cause an overtemperature fault. Monitor power bus voltage, to see if it gets too high, particularly during deceleration. This could cause an overvoltage fault.

## Miscellaneous Problems

The basic troubleshooting procedure, presented above, will identify most problems, particularly those that affect the LEDs or the fault output. Some problems, however, occur transiently during a move, or do not affect the LEDs. Others may be due to wiring mistakes, or failure of other components in the system (encoder, motor, etc.). The sections below will help you identify such problems.

### Problems During Move

***Speed/Torque Limitations***

Make sure that you are not commanding a move that requires the motor to go faster than it can, or use more torque than it can produce. Check the motor's speed/torque curve for your operating voltage.

***Weak Power Supply***

A weak power supply may not produce sufficient power during all parts of the move. It can cause an undervoltage problem. Undervoltage can affect the drive in two ways:

- Temporary Fault – the red LED will turn ON and the fault output will go HIGH during the undervoltage condition. The fault is *not latched*, and will disappear when the voltage goes above approximately 24VDC.
- Latched Fault—The undervoltage trips the short circuit protection. The green LED is turned off, the Red LED is turned ON, and the fault output goes HIGH. This is a *latched* condition.

For a full description of faults caused by a weak power supply, see the section on *Undervoltage*, and the section on *Short Circuit Protection*, in *Chapter 4 Special Internal Circuits*.

***Excessive Friction***

Too much friction in your system might cause move problems. Excessive friction can cause trouble when mechanical components in a system age. As friction increases, problems may occur in a system that had previously been working well.

### Mechanical Problems

Check for binds, jams, increased friction, or other problems in the mechanical system. If a system was working properly, but then suddenly develops new problems, check for changes in the mechanical system that could be causing the problems.

### ELECTRICAL NOISE PROBLEMS

Electrical noise can cause problems, depending on the application and the sensitivity of equipment in the system. For more information on identifying problems caused by electrical noise, and solutions to those problems, consult the technical section in Compumotor's *EMC Installation Guide*.

## Product Return Procedure

If you must return the OEM770X for repairs, use the following steps:

1. Get the serial number and the model number of the defective unit, and a purchase order number to cover repair costs in the event the unit is determined to be out of warranty.

2. In the USA, call your Automation Technology Center (ATC) for a Return Material Authorization (RMA) number. Returned products cannot be accepted without an RMA number. If you cannot obtain an RMA number from your ATC, call Parker Compumotor's Customer Service Department at (800) 722-2282.

Ship the unit to:

Parker Hannifin Corporation
Compumotor Division
5500 Business Park Drive, Suite D
Rohnert Park, CA 94928
Attn: RMA # xxxxxxxxx

3. In the UK, call Parker Digiplan for a GRA (Goods Returned Authorization) number. Returned products cannot be accepted without a GRA number. The phone number for Parker Digiplan Repair Department is 0202-690911. The phone number for Parker Digiplan Service/Applications Department is 0202-699000.

Ship the unit to:

Parker Digiplan Ltd.
21, Balena Close,
Poole, Dorset,
England. BH17 7DX

4. Elsewhere: Contact the distributor who supplied the equipment.

# APPENDIX A

## LVD Installation Instructions

For more information about LVD, see 73/23/EEC and 93/68/EEC, published by the European Economic Community (EEC).

## Environmental Conditions

### *Pollution Degree*

The OEM770X is designed for pollution degree 2.

### *Installation Category*

The OEM770X is designed for installation category II.

## Electrical

### *Connecting and Disconnecting Power*

The OEM770X's protective earth connection is provided through its heatsink. You must reliably earth the OEM770X's protective earth connection.

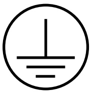
Attach or remove the OEM770X's power connections only while input power is OFF.

The OEM770X's supply voltage is limited to 75 VDC.

### *Connecting the Protective Conductor Terminal to Earth*

You must provide a connection from the OEM770X's protective conductor terminal to a reliable earth point.

The protective conductor terminal is marked with a label on the product bearing the following symbol:



*Protective Conductor Terminal Marking*

To connect the protective conductor terminal to earth, complete these steps:

1. Use a ring terminal in combination with a star washer to make good contact with the exposed metal surface surrounding the lower mounting hole on the OEM770X. (The dimension drawing in *Chapter 2 Installation* indicates that the lower mounting hole is surrounded by exposed metal.)

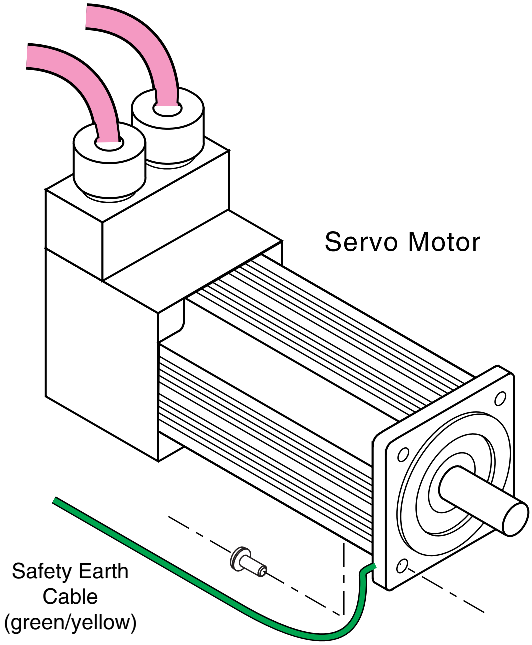
2. Use a VDE approved green/yellow protective conductor terminal wire to reliably earth the protective conductor terminal. Wire gauge must be no thinner than the current-carrying wire in the product's mains supply.

3. Resistance between the protective conductor terminal and earth must be no greater than 0.1 ohm. Use thicker gauge wire if the resistance is too high.

### *Providing a Protective Earth Connection for Motors*

You must provide a connection from the motor to a reliable protective earth. This connection provides a protective earth for the motor contact point. The motor's protective earth connection is important for safety reasons, and *must not be omitted*.

Make connections according to the following instructions and diagram:



1. Use a ring terminal in combination with a star washer and mounting bolt to make good contact with the bare metal surface of the motor's mounting flange.

2. Use a VDE approved green/yellow protective conductor terminal wire to make the connection between the motor and earth. Wire gauge must be no thinner than the current carrying wire in the motor's power cable.

3. Resistance between the motor and earth must be no greater than 0.1 ohm. Use thicker gauge wire if the resistance is too high.

## MECHANICAL

### *Installing in an Enclosure*

The OEM770X must be installed within an enclosure. The enclosure's interior must not be accessible to the operator. The enclosure should be opened only by skilled or trained service personnel.

### *Do Not Operate the OEM770X Without Cover*

The cover provides mechanical support to the circuit assemblies inside.

## SERVICING THE OEM770X

### *Changing Firmware*

Only skilled or trained personnel should change firmware.

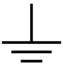
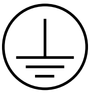
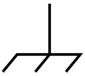
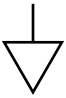
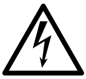
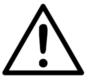
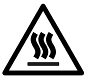
## THERMAL SAFETY

### *The Motor May Be Hot*

The motor may reach high temperatures during normal operations, and may remain hot after power is removed.

## Table of Graphic Symbols and Warnings

The following symbols may appear in this user guide, and may be affixed to the products discussed in this user guide.

| Symbol | Description |
|---|---|
|  | Earth Terminal |
|  | Protective Conductor Terminal |
|  | Frame or ChassisTerminal |
|  | Equipotentiality |
|  | Caution, Risk of Electric Shock |
|  | Caution, Refer to Accompanying Text |
|  | Hot Surface |

# APPENDIX B

## *Software Reference*

Use this chapter as a reference for the function, range, default, and sample use of each command for the OEM770X.

When we refer to "the controller", we are referring to the OEM770X.

## Command Descriptions

### ① A—Acceleration *(Example Command)*

| | |
|---|---|
| ② Command Type: Motion | ⑥ Valid Software Version: A |
| ③ Syntax: <a>An | ⑦ Units: revs/sec$^2$ |
| ④ Range: n = 0.01-999.99 | ⑧ Default Value: A = 100 |
| ⑤ Attributes: Buffered,<br>Savable in Sequence | ⑨ See Also: D, G, MR, V |
| | ⑩ Response to aA is *An |

① ***Command Mnemonic***

The beginning of each command entry contains the command's mnemonic value and full name.

② ***Command Type***

*Set-Up*—Set-up commands define application conditions. These commands establish the output data's format from the controller.

*Motion*—Motion commands affect motor motion, such as acceleration, velocity, distance, go home, stop, direction, mode, etc.

*Programming*—Programming commands affect programming and program flow for trigger, output, all sequence commands, time delays, pause and continue, enable and disable, loop and end-loop, line feed, carriage return, and backspace.

*Status*—Status commands respond (report back) with data. These commands instruct the system to send data out from the serial port for host computer use.

③ ***Syntax***

The proper syntax for the command is shown here. The specific parameters associated with the command are also shown. If any of these parameters are shown in brackets, such as <a>, they are optional. The parameters are described below.

**a**—An *a* indicates that a device address must accompany the command. Only the device specified by this parameter will receive and execute the command. Valid addresses are 1-255.

**n**—An *n* represents an integer. An integer may be used to specify a variety of values (acceleration, velocity, etc.).

**s**—An *s* indicates that a sign character, either positive or negative (+ or -), is required.

**x**—An *x* represents any character or string of characters.

### ④ Range

This is the range of valid values that you can specify for n (or any other parameter specified).

### ⑤ Attributes

This first attribute indicates if the command is *immediate* or *buffered.* The system executes immediate commands as soon as it receives them. Buffered commands are executed in the order that they are received with other buffered commands. Buffered commands can be stored in a sequence.

The second attribute explains how you can save the command.

- Savable in Sequence
- Never Saved
- Automatically Saved

*Savable in Sequence* commands are saved when they are defined in a sequence (see `XT` command). *Savable in Sequence* commands can be stored in system memory (nonvolatile) and retained when power is removed from the system. A command that is *Never Saved* is executed without being saved into the system's permanent memory . *Automatically Saved* commands are automatically saved into memory upon execution.

### ⑥ Valid Software Version

This field contains the current revision of the software in which the command resides at the time this user guide was released.

### ⑦ Units

This field describes what unit of measurement the parameter in the command syntax represents.

### ⑧ Default Value

The default setting for the command is shown in this box. A command will perform its function with the default setting if you do not provide a value.

### ⑨ See Also

Commands that are related or similar to the command described are listed here.

### ⑩ Response

A sample status command and system response are shown. When the command has no response, this field is not shown.

# A—Acceleration

❏ Command Type: Motion
❏ Syntax: <a>An
❏ Range: n = 0.01-9999.99
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: revs/sec$^2$
❏ Default Value: A = 100
❏ See Also: D, G, MR, V
❏ Response to aA is *An

The Acceleration command specifies the rotary acceleration rate to be used for the next Go (`G`) command. The acceleration remains set until you change it. You do not need to reissue this command for subsequent Go (`G`) commands. Accelerations outside the valid range cause the acceleration to remain at the previous valid **A** setting.

If the Acceleration command is entered with only a device address (`1A`), the controller will respond with the current acceleration value. If a move is commanded without specifying an acceleration rate, the previously commanded acceleration rate will be used. Acceleration cannot be changed on the fly. The minimum acceleration is

Min Accel = Encoder resolution (**ER** command) x .00465

| **Command** | **Description** |
|---|---|
| **A1Ø** | Sets acceleration to 10 revs/sec$^2$ |
| **V1Ø** | Sets velocity to 10 revs/sec |
| **D2ØØØ** | Sets distance to 2,000 encoder counts |
| **G** | Executes the move |

# B—Buffer Status

❏ Command Type: Status
❏ Syntax: aB
❏ Range: N/A
❏ Attributes: Immediate, Never Saved
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ Response to aB is *B or *R
❏ See Also: BS

The buffer status command will report the status of the command buffer. If the command buffer is empty or less than 95% full, the controller will respond with a ***R**.

The command buffer is 512 bytes long. A ***B** response will be issued if less than 5% of the command buffer is free.

*R = More than 5% of the buffer is free
*B = Less than 5% of the buffer is free

This command is commonly used when a long series of commands will be loaded remotely via RS-232C interface. If the buffer size is exceeded, the extra commands will not be received by the controller until more than 5% of the command buffer is free.

| Command | Response |
|---|---|
| **1B** | ***B** (less than 5% of the command buffer is free) |

## BCDG—Buffered Configure Derivative Gain

- □ Command Type: Set-up
- □ Syntax: <a>BCDGn
- □ Range: n = 0-32,767
- □ Attributes: Buffered
  Savable in Sequence
- □ Valid Software Version: A
- □ Units: N/A
- □ Default Value: 240
- □ Response to aBCDG is *CDGn
- □ See Also: BCIG, BCPG, BCTG, CIG, CPG, CTG

This buffered command is used for system tuning. This term represents the gain applied to the derivative of the position error—in other words, the rate at which the position error is changing. This gain produces a damping effect similar to velocity feedback.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| Command | Description |
|---|---|
| **BCDG400** | Set the derivative gain term to 400 |
| **1BCDG** | Reports derivative gain term (*CDG400) |

## BCIG—Buffered Configure Integral Gain

- □ Command Type: Set-up
- □ Syntax: <a>BCIGn
- □ Range: n = 0-32,767
- □ Attributes: Buffered
  Savable in Sequence
- □ Valid Software Version: A
- □ Units: N/A
- □ Default Value: 2
- □ Response to aBCIG is *CIGn
- □ See Also: BCDG, BCPG, BCTG, CDG, CPG, CTG

This buffered command is used for system tuning. This term represents the gain applied to the integral of the position error—the net accumulation of the position error over time. Thus integral gain will contribute when a position error is not being reduced over time, as may be caused by the effects of friction or gravity. This gain will improve overall accuracy but may increase settling time and, if excessive, may cause a low frequency oscillation around the commanded position.

Before you increase **BCIG**, you must first increase the integral limit (**BCIL**) to an equal or higher value.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| Command | Description |
|---|---|
| **BCIL40** | Set the integral limit term to 40 |
| **BCIG10** | Set the integral gain term to 10 |
| **1BCIG** | Reports integral gain  term (*CIG10) |

## BCIL—Buffered Configure Integral Limit

❏ Command Type:  Set-up
❏ Syntax:  <a>BCILn
❏ Range:   n = 0-32,767
❏ Attributes:  Buffered
Savable in Sequence

❏ Valid Software Version:  A
❏ Units:  N/A
❏ Default Value:   2
❏ Response to aBCIL is *CILn
❏ See Also:  BCDG, BCIL, BCPG, BCTG, CDG, CIL,CPG, CTG

This buffered command is used for system tuning.  This term represents the limit applied to the integral gain contribution of the PID equation.  A high integral gain  with a large inertial load can cause a non-ringing overshoot of the commanded position. By limiting the contribution of integral action, this overshoot can be minimized.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| Command | Description |
|---|---|
| **BCIG10** | Set the integral gain term to 10 |
| **BCIL40** | Set the integral limit to 40 |
| **1BCIL** | Reports integral limit term (*CIL40) |

## BCPE—Buffered Configure Position Error

❏ Command Type: Set-up
❏ Syntax: <a>BCPEn
❏ Range: n = 0-32,767
❏ Attributes: Buffered
Savable in Sequence

❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: 4000
❏ Response to aBCPE is *CPEn
❏ See Also: DPE, CPE

This buffered command defines the maximum allowable position or following error. If the actual position error ever exceeds the allowable position error, the controller will generate a fault condition and shut down the drive. If the maximum allowable position error is set to Ø, the function is disabled and no amount of position error will generate the fault condition.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| **Command** | **Description** |
|---|---|
| **BCPE400** | Set the maximum allowable position error to 400 encoder counts |
| **BCPE0** | Disable fault generation due to position error |
| **1BCPE** | Reports maximum position error setting (*CPE0) |

## BCPG—Buffered Configure Proportional Gain

❏ Command Type: Set-up
❏ Syntax: <a>BCPGn
❏ Range: n = 0-32,767
❏ Attributes: Buffered
Savable in Sequence

❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: 16
❏ Response to aBCPG is *CPGn
❏ See Also: BCDG, BCIG, BCTG, CDG, CIG, CTG

This buffered command is used for system tuning. This term represents the gain applied directly to the position error. The proportional gain sets how actively the system will respond to position error. High proportional gain will give a stiff, responsive system, but may result in overshoot and oscillation.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| **Command** | **Description** |
|---|---|
| **BCPG50** | Set the proportional gain term to 50 |
| **1BCPG** | Reports proportional gain term (*CPG50) |

## BCTG—Buffered Configure Derivative Sampling Period

❏ Command Type: Set-up
❏ Syntax: <a>BCTGn
❏ Range: n = 0-255
❏ Attributes: Buffered Savable in Sequence
❏ Valid Software Version: A
❏ Units: 266 µsec
❏ Default Value: 0
❏ Response to aBCTG is *CTGn
❏ See Also: BCDG, BCIG, BCPG, CDG, CIG, CPG

This buffered command is used for system tuning. Use **BCTG** to adjust the derivative sampling period. The *system* sampling period—266 µsec—is the period between updates of position error. The *derivative* sampling period is an integer multiple of the system sampling period. In general, a longer derivative sampling period gives a more constant derivative term and improves stability. Many systems require a low **BCTG** value to prevent oscillations, however. Therefore, start with a low value and increase it incrementally.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| **Command** | **Description** |
|---|---|
| **BCTG5** | Set the derivative sampling period to 1596 µsec |
| **1BCTG** | Reports derivative sampling period (*CTG5) |

## BS—Buffer Size Status

❏ Command Type: Status
❏ Syntax: aBS
❏ Range: N/A
❏ Attributes: Immediate, Never Saved
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ Response to aBS is *n
❏ See Also: B

This command reports the number of bytes remaining in the command buffer. When entering long string commands, check the buffer status to be sure that there is enough room in the buffer. Otherwise, commands may be lost. Each character (including delimiters) uses one byte. The range for the response is 0 - 512 bytes.

| **Command** | **Response** |
|---|---|
| **1BS** | ***122** (122 bytes available in the buffer) |

# C—Continue

❏ Command Type: Motion
❏ Syntax: <a>C
❏ Range: N/A
❏ Attributes: Immediate, Never Saved

❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: PS, U

The Continue (C) command ends a pause state. It enables your controller to continue executing buffered commands. After you enter a Pause (PS) or the Pause and Wait for Continue (U) command, you can clear it with a Continue (C) command. This command is useful when you want to transmit a string of commands to the buffer before you actually execute them.

| **Command** | **Description** |
|---|---|
| **MC** | Sets move to continuous mode |
| **A1Ø** | Sets acceleration to 10 revs/sec$^2$ |
| **V1Ø** | Sets velocity to 10 revs/sec |
| **PS** | Pauses system until controller receives **C** command |
| **G** | Accelerates the motor to 10 revs/sec |
| **C** | Continues executing commands in the buffer |

The motor will not execute the **G** command until the **C** command is issued.

# CDG—Configure Derivative Gain

❏ Command Type: Set-up
❏ Syntax: <a>CDGn
❏ Range: n = 0-32,767
❏ Attributes: Immediate Automatically Saved

❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: 240
❏ Response to aCDG is *CDGn
❏ See Also: CIG, CPG, CTG, RFS

This command is used for system tuning. This term represents the gain applied to the derivative of the position error, in other words, the rate at which the position error is changing. This gain produces a damping effect similar to velocity feedback.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| **Command** | **Description** |
|---|---|
| **CDG400** | Set the derivative gain term to 400 |
| **1CDG** | Reports derivative gain term (*CDG400) |

# CEW—Configure In Position Error Window

❑ Command Type: Set-up
❑ Syntax: <a>CEWn
❑ Range: n = 0-32,767
❑ Attributes: Buffered
Savable in Sequence

❑ Valid Software Version: A
❑ Units: encoder counts
❑ Default Value: 50
❑ Response to aCEW is *CEWn
❑ See Also: CIT, SSC

This command, together with the **CIT** command, can be used to configure an In Position window, which can be used to indicate that the preceding move has terminated.

The In Position condition is met when:

- The controller algorithm has finished (no input position command)
- The CEW condition is met (the position error is less than that specified by the **CEW** command).
- The above condition has been true for the length of time specified by the **CIT** command

The position error range, specified by *n* in **CEW*n***, is the maximum number of encoder counts allowed on either side of the desired position. For example, if *n* = 10, then the In Position window is 20 encoder counts wide.

Output 1 can be configured with the **SSC** command to show the state of the In Position detector. This allows the user to trigger external hardware from the In Position condition.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| **Command** | **Description** |
|---|---|
| **CEW10** | Configure an In Position Error Window ±10 encoder counts either side of desired position |
| **1CEW** | Reports ± number of encoder counts (*CEW10) |

# CIG—Configure Integral Gain

❑ Command Type: Set-up
❑ Syntax: <a>CIGn
❑ Range: n = 0-32,767
❑ Attributes: Immediate
Automatically Saved

❑ Valid Software Version: A
❑ Units: N/A
❑ Default Value: 2
❑ Response to aCIG is *CIGn
❑ See Also: CIL, CDG, CPG, CTG, RFS

This command is used for system tuning. This term represents the gain applied to the integral of the position error—the net accumulation of the position error over time. Thus integral gain will contribute when a position error is not being reduced over time, as may be caused by the effects of friction or gravity. This gain will improve overall accuracy but may increase settling time and, if excessive, may cause a low frequency oscillation around the commanded position.

Before you increase **CIG**, you must first increase the integral limit (**CIL**) to an equal or higher value.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| Command | Description |
|---|---|
| **CIL40** | Set the integral limit term to 40 |
| **CIG10** | Set the integral gain term to 10 |
| **1CIG** | Reports integral gain term (*CIG10) |

# CIL—Configure Integral Limit

❑ Command Type: Set-up
❑ Syntax: <a>CILn
❑ Range: n = 0-32,767
❑ Attributes: Immediate
Automatically Saved

❑ Valid Software Version: A
❑ Units: N/A
❑ Default Value: 2
❑ Response to aCIL is *CILn
❑ See Also: CIG,CDG, CPG, CTG, RFS

This command is used for system tuning. This term represents the limit applied to the integral gain contribution of the PID equation. A high integral gain with a large inertial load can cause a non-ringing overshoot of the commanded position. By limiting the contribution of integral action, this overshoot can be minimized.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| Command | Description |
|---|---|
| **CIG10** | Set the integral gain term to 10 |
| **CIL40** | Set the integral limit to 40 |
| **1CIL** | Reports integral limit term (*CIL40) |

# CIT—Configure In Position Time

| | |
|---|---|
| ❏ Command Type: Setup | ❏ Valid Software Version: A |
| ❏ Syntax: <a>CITn | ❏ Units: Milliseconds |
| ❏ Range: n = 0-32,767 | ❏ Default Value: 20 |
| ❏ Attributes: Buffered<br>Savable in Sequence | ❏ Response to aCIT is *CITn<br>❏ See Also: CEW, SSC |

This command is used to specify the time period that the servo is to be within the In Position window before the "In Position" signal is generated. The range is 0 to 32,767, and is the number of milliseconds to be used as the testing time frame.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| Command | Description |
|---|---|
| **1SSC1** | Set output 1 as "In Position" |
| **1CIT30** | Set "In Position" time to 30ms |
| **1CEW20** | Set allowable position error to ±20 encoder counts |

# CPE—Configure Position Error

| | |
|---|---|
| ❏ Command Type: Set-up | ❏ Valid Software Version: A |
| ❏ Syntax: <a>CPEn | ❏ Units: N/A |
| ❏ Range: n = 0-32,767 | ❏ Default Value: 4000 |
| ❏ Attributes: Immediate<br>Automatically Saved | ❏ Response to aCPE is *CPEn<br>❏ See Also: DPE, RFS |

This command defines the maximum allowable position or following error. If the actual position error ever exceeds the allowable position error, the controller will generate a fault condition and shut down the drive. If the maximum allowable position error is set to Ø, the function is disabled and no amount of position error will generate the fault condition.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| Command | Description |
|---|---|
| **CPE400** | Set the maximum allowable position error to 400 encoder counts |
| **CPE0** | Disable fault generation due to position error |
| **1CPE** | Reports maximum position error setting (*CPE0) |

# CPG—Configure Proportional Gain

❏ Command Type: Set-up
❏ Syntax: <a>CPGn
❏ Range: n = 0-32,767
❏ Attributes: Immediate
Automatically Saved

❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: 16
❏ Response to aCPG is *CPGn
❏ See Also: CDG, CIG, CTG, RFS

This command is used for system tuning. This term represents the gain applied directly to the position error. The proportional gain sets how active the system will be to position error. High proportional gain will give a stiff, responsive system, but may result in overshoot and oscillation.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| **Command** | **Description** |
|---|---|
| **CPG50** | Set the proportional gain term to 50 |
| **1CPG** | Reports proportional gain term (*CPG50) |

# CR—Carriage Return

❏ Command Type: Programming
❏ Syntax: <a>CR
❏ Range: N/A
❏ Attributes: Buffered
Savable in Sequence

❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: LF

The Carriage Return (CR) command determines when the controller has reached a particular point in the execution buffer. When the controller reaches this command in the buffer, it responds by issuing a carriage return (ASCII 13) over its interface back to the host computer or terminal. If you place the CR command after a Go (**G**) command, it indicates when a move is complete. If you place the **CR** command after a Trigger (**TR**) command, it indicates when the trigger condition is met.

You can use Carriage Return (**CR**) and Line Feed (**LF**) commands with the Quote (“) command to display multiple-line messages via the RS-232C interface.

| **Command** | **Description** |
|---|---|
| **MN** | Sets mode to preset mode |
| **A5Ø** | Sets acceleration to 50 revs/sec$^2$ |

| | |
|---|---|
| **V5** | Sets Velocity to 5 revs/sec |
| **D5ØØØ** | Sets distance to 5,000 encoder counts |
| **G** | Executes the move (Go) |
| **1CR** | Sends a carriage return after move is completed |

The motor moves 5,000 encoder counts. When the motor stops, the controller sends a carriage return over its interface.

## CTG—Configure Derivative Sampling Period

❏ Command Type: Set-up
❏ Syntax: <a>CTGn
❏ Range: n = 0-255
❏ Attributes: Immediate
Automatically Saved

❏ Valid Software Version: A
❏ Units: 266 µsec
❏ Default Value: 0
❏ Response to aCTG is *CTGn
❏ See Also: CDG, CIG, CPG, RFS

This command is used for system tuning. Use **CTG** to adjust the derivative sampling period. The *system* sampling period—266 µsec—is the period between updates of position error. The *derivative* sampling period is an integer multiple of the system sampling period. In general, a longer derivative sampling period gives a more constant derivative term and improves stability. Many systems require a low **CTG** value to prevent oscillations, however. Therefore, start with a low vlaue and increase it incrementally.

Refer to *Tuning* in *Chapter ② Installation* for more information.

| **Command** | **Description** |
|---|---|
| **CTG5** | Set the filter time constant to 1596 µsec |
| **1CTG** | Reports filter time constant (*CTG5) |

## D—Distance

❏ Command Type: Motion
❏ Syntax: <a>Dn
❏ Range: n = ±1,073,741,823
❏ Attributes: Buffered
Savable in Sequence

❏ Valid Software Version: A
❏ Units: encoder counts
❏ Default Value: 4000
❏ Response to aD is *Dn
❏ See Also: A, G, MN, MPA, MPI, V, H

The Distance (**D**) command defines either the number of encoder counts the motor will move or the absolute position it will seek after a Go (**G**) command is entered. In incremental mode (**MPI**), the value set with the Distance (**D**) command will be the distance (in encoder counts) the motor will travel on all subsequent Go (**G**) commands. In absolute mode (**MPA**), the distance moved by the motor will be the difference between the present motor position and the position (referenced to the zero position) set with the **D** command. In either mode, the direction is controlled by the direction (+ or -) that precedes the distance value. The **D** command has no effect on continuous moves (**MC**).

In Mode Normal (**MN**) the position may not exceed the maximum distance range of 1,073,741,823 encoder counts. If the motor approaches the absolute maximum (plus or minus), the controller will not execute any **GO** commands that would cause the distance to exceed the absolute maximum. To proceed further, use the **PZ** command to reset the absolute counter to zero, and then resume operations.

If **D** is entered with only a device address (**1D**), the controller will respond with the current distance value. If a move is commanded without specifying a distance, the previously commanded distance will be applied to the move.

| Command | Description |
|---|---|
| **MN** | Sets controller to Normal mode |
| **MPI** | Sets controller to Incremental Position mode |
| **A1Ø** | Sets acceleration to 10 revs/sec$^2$ |
| **V1Ø** | Sets velocity to 10 revs/sec |
| **D4ØØØ** | Sets distance to 4000 encoder counts |
| **G** | Executes the move |

A servo motor with a 4000 count encoder will travel 1 rev (CW) after G is issued.

# DPA—Display Position Actual

- □ Command Type: Status
- □ Syntax: aDPA
- □ Range: n = ±1,073,741,823
- □ Attributes: Immediate Device Specific, Never saved
- □ Valid Software Version: A
- □ Units: Encoder counts
- □ Default Value: NA
- □ Response to aDPA is *DPAn
- □ See Also: D, PZ

Single display of actual motor position as measured by the encoder. This command is functionally identical to the **PX** command. The response is the position in encoder counts as referenced to the last power reset or position zero command (**PZ**).

| Command | Description |
|---|---|
| **1DPA** | Report the actual motor position of axis 1 (*+0004000000) |

## DPE—Display Position Error

❑ Command Type: Status
❑ Syntax: aDPE
❑ Range: n = ±2,147,483,646
❑ Attributes: Immediate
Device Specific, Never saved

❑ Valid Software Version: A
❑ Units: Encoder counts
❑ Default Value: NA
❑ Response to aDPE is *DPEn
❑ See Also: D, DPA

Single display of position error. The response is the difference in encoder counts of the actual motor position and the commanded motor position. This information is used by the control algorithm to control the torque command to the motor. It is normal for the position error to be present during a move but large position errors are usually caused by inappropriate gain settings. Issuing a DPE command before the motor has settled into final position will also result in larger position errors.

| Command | Description |
|---|---|
| **1DPE** | Report the position error of axis 1 (*+0000000005). |

## DVA—Display Velocity Actual

❑ Command Type: Status
❑ Syntax: aDVA
❑ Range: n = 0 – 20000
❑ Attributes: Immediate
Device Specific, Never saved

❑ Valid Software Version: A
❑ Units: (rev/sec)∗100
❑ Default Value: NA
❑ Response to aDVA is *DVAn
❑ See Also: DPA, V

Single display of actual motor velocity in revolutions per sec. There is an implied decimal point before the last two digits.

| Command | Description |
|---|---|
| **MC** | Mode continuous |
| **V2** | Velocity of 2 rev/sec |
| **G** | Go |
| **1DVA** | Report the actual motor velocity of axis 1 (*00200) |

# E—Enable Communications

❏ Command Type: Programming
❏ Syntax: <a>E
❏ Range: N/A
❏ Attributes: Immediate
Never Saved
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: Enabled
❏ See Also: F

The Enable Communications (**E**) command allows the controller to accept commands over the serial communications interface. You can re-enable the communications interface with this command if you had previously disabled the RS-232C interface with the Disable Communications Interface (**F**) command. If several units are using the same communications interface, the **E** and **F** commands can help streamline programming.

| Command | Description |
|---|---|
| **F** | Disables all units (axes) on the communications interface |
| **1E** | Enables serial interface on Device 1 |
| **4E** | Enables serial interface on Device 4 |
| **A1Ø** | Set acceleration to 10 revs/sec$^2$ |
| **V5** | Set velocity to 5 revs/sec |
| **D5ØØØ** | Sets distance to 5000 encoder counts |
| **G** | Executes the move (Go—only axes 1 & 4 will move) |

# ER—Encoder Resolution

❏ Command Type: Set-up
❏ Syntax: <a>ERn
❏ Range: n = 400 - 65,532
❏ Attributes: Buffered,
Savable in Sequence
❏ Valid Software Version: A
❏ Units: n = encoder counts/rev
❏ Default Value: 4000
❏ Response to aER is *ERn
❏ See Also: CPE

The encoder resolution defines the number of encoder counts the controller will see per revolution of the motor. Multiply the number of lines on an encoder by 4 to arrive at the correct **ER** value per revolution of the motor. (In other words, one line of an encoder will produce 4 encoder counts due to quadrature detection)

Enter the **ER** value before you enter Acceleration (**A**) and Velocity (**V**) commands.

| Command | Description |
|---|---|
| **ER4ØØØ** | Sets encoder resolution to 4000 encoder counts per 1 motor revolution |
| **1ER** | Reports Encoder Resolution (*ER4000) |

# F—Disable Communications

❏ Command Type: Programming
❏ Syntax: <a>F
❏ Range: N/A
❏ Attributes: Immediate
Never Saved
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: None
❏ See Also: E

The Disable Communications (**F**) command is useful when you are programming multiple units on a single interface. Axes that are not intended to process global commands should be disabled using device specific **F** commands. This allows you to program other units without specifying a device identifier on every command. If you do not disable other units in a daisy chain, uploading programs may cause other units on the daisy chain to perform uploaded commands.

| Command | Description |
|---|---|
| **1F** | Disables the communications interface on unit #1 |
| **3F** | Disables the communications interface on unit #3 |
| **G** | All controllers (except 1 & 3) will execute a move |

# G—Go

❏ Command Type: Motion
❏ Syntax: <a>G
❏ Range: N/A
❏ Attributes: Buffered
Savable in Sequence
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: None
❏ See Also: A, D, MC, MN, S, V

The Go (**G**) command instructs the motor to make a move using motion parameters that you have previously entered. You do not have to re-enter Acceleration (**A**), Velocity (**V**), Distance (**D**), or the current mode (**MN** or **MC**) commands with each **G** (if you do not need to change them).

In the Normal mode (**MN**), moves can be either incremental or absolute. In the Incremental Preset mode (**MPI**), a **G** will initiate the move distance you specified with the **D** command. A **G** command in the Absolute Preset mode (**MPA**) will not cause motion unless the distance (**D**) value is different from the present motor position (**PR**) .

In Continuous mode (**MC**), you only need to enter the Acceleration (**A**) and Velocity (**V**) commands prior to **G**. The system ignores the Distance (**D**) command in this mode.

No motor motion will occur until you enter **G** in either the Normal (**MN**) or Continuous (**MC**) modes. If motion does not occur with **G**, an activated end-of-travel limit switch may be on. Check the hard limit switches or use the limit disable command (**LD3**—see **RA** command also). The next buffered command will not be executed until after the move is completed.

| Command | Description |
|---|---|
| **MN** | Sets Normal mode (preset) |
| **A5** | Sets acceleration to 5 revs/sec$^2$ |
| **V1Ø** | Sets velocity to 10 revs/sec |
| **D2ØØØ** | Sets distance to 2,000 encoder counts |
| **G** | Executes the move (Go) |
| **A1** | Sets acceleration to 1 rev/sec$^2$ |
| **G** | Executes the move (Go) |

Assuming the controller is in Incremental Preset mode, the motor turns 2,000 encoder counts and repeats the 2,000 count move using the new acceleration value of 1 rev/sec$^2$ (Total distance moved = 4,000 encoder counts).

# GH—Go Home

❏ Command Type: Motion
❏ Syntax: <a>GHn
❏ Range: n = ± .01 - 200
❏ Attributes: Buffered
Savable in Sequence
❏ Valid Software Version: A
❏ Units: Revs/sec
❏ Default Value: n = 0
❏ See Also: OS, RC, V,IN

The Go Home (**GH**) command instructs the controller to search for an absolute home position in the positive or negative (+ or -) direction. It defines home as the position where the home limit signal changes states. To use the GH command, one of the general purpose inputs (pins 1-5) must be configured as a home input (**IN** command).

Homing can be as simple as decelerating to a stop when the edge of the home limit is detected. By using the **OS** commands, the homing process can be tailored to meet the application needs.

**OSB**—*Back up to home* makes homing more repeatable by backing off the home switch and re-approaching at low speed. The final approach to home switch is always from the CW direction.

**OSC**—*Define active state of home* allows the use of a normally closed or normally open limit switch.

**OSD**—*Enable Z Channel for home* uses the Z channel of the encoder, in conjunction with a home switch, to determine the final home position. The Z channel is a more accurate home position than the edge of a switch.

**OSH**—*Reference edge of home switch* allows either edge of the home switch to be used as the final edge position.

The controller will reverse direction if an end-of-travel limit is activated while searching for home. However, if a second end-of-travel limit is encountered in the new direction, the Go Home procedure will stop and the operation will be aborted. The Homing Status (**RC**) command will indicate if the homing operation was successful.

The Go Home command will use acceleration set by the A command. The Go Home velocity will not affect the standard velocity (**V**) value.

| Command | Description |
|---|---|
| **INE1** | Configure input #1 as home input |
| **OSB1** | Back up the home switch |
| **OSD1** | Reference Z channel as final home |
| **GH-2** | The motor moves CCW at 2 revs/sec and looks for the Home Limit input to go active. |

Since the motor is turning CCW, it will see the CW edge of the limit first. It will decelerate to a stop and turn at 0.1 rev/sec in the CW direction until it detects the Z channel.

## ^H—Delete

❑ Command Type: Programming
❑ Syntax: ^H
❑ Range: N/A
❑ Attributes: Immediate
Never Saved

❑ Valid Software Version: A
❑ Units: N/A
❑ Default Value: None

This command allows you to delete the last character that you entered. The **^H** command will not prevent execution of an immediate command. A new character may be entered at that position to replace the existing character. (**^H** indicates that the Ctrl key is held down when the H key is pressed.) This command prompts the controller to backup one character in the command buffer, regardless of what appears on the terminal. On some terminals, the Ctrl and the left arrow (<—) keys produce the same character.

This command will *not* delete characters beyond the last delimiter issued. Pressing the delete key will not delete the previous character.

# H—Set Direction

❏ Command Type: Programming
❏ Syntax: <a>H(s)
❏ Range: s = + or -
❏ Attributes: Buffered
Savable in Sequence

❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: +
❏ See Also: D

The Set Direction (**H**) command changes or defines the direction of the next move that the system will execute. This command does not affect moves already in progress.

H+ **=** Sets move to CW direction
H– **=** Sets move to CCW direction
H **=** Changes direction from the previous setting

In preset moves, a Distance (**D**) command entered after the **H** command overrides the direction set by the **H** command. In Continuous mode (**MC**), only the **H** command can set the direction of motion.

| Command | Description |
|---|---|
| **MN** | Sets Normal mode |
| **A5** | Sets acceleration to 5 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **D4ØØØ** | Sets distance to 4,000 encoder counts |
| **G** | Executes the move (Go) in CW direction |
| **H** | Reverses direction |
| **G** | Executes the move (Go) in CCW direction |
| **MC** | Sets mode to continuous |
| **H+** | Sets direction to CW |
| **G** | Moves continuously in CW direction |

# IN—Set Input Function

❏ Command Type: Set-up
❏ Syntax: aINxn
❏ Range: x = A – F, n = 1 – 5
❏ Attributes: Buffered
Savable in sequence

❏ Valid Software Version: A
❏ Units: NA
❏ Default Value: AAAAA
❏ Response to aIN is *xxxxx
❏ See Also: IS, TR, XP, #, K, GH

This command configures the function of each of the 5 general purpose inputs. You can configure each input to perform one of the following functions:

Function A—*Trigger Input*
Used with the **TR** command as a comparison input. The TR command defines active high, low, or "don't care." Up to 5 inputs can be configured as trigger inputs.

Function B—*Sequence Select Input*
Executes predefined sequences from remote inputs based on the **XP** command. Active state (sequence selected) is high. Up to 3 inputs can be configured as sequence select inputs.

Function C—*Kill Input*
Immediately halts execution of the move. Same as kill (**K**) command. Active state (kill initiated) is high. Up to one input can be configured as a kill input.

Function D—*Stop Input*
Decelerates the motor to a stop using the value specified in the Acceleration (**A**) command (dumps the sequence or command buffer if configured by **SSHØ**). Same as Stop (**S**) command. Active state (stop initiated) is high. Up to one input can be configured as a stop input.

Function E—*Home Input*
Defines the motor home or origin position as executed by the **GH** command and configured with the Homing Function (**OS**) commands. **OSC** determines the active state of the input. Up to one input can be configured as a home input.

Function F—*Go Input*
Accelerates the motor to a velocity and distance specified in the Acceleration (**A**), Velocity (**V**), and Distance (**D**) commands. Same as Go (**G**) command. Active state (go initiated) is high. Up to one input can be configured as a go input.

Some of the functions (stop, kill, home, go) can only have one input configured to that function. If you try to configure another input to that function, the controller will not recognize the new function and revert back to the previous definition. For example, if input 1 is a kill function, and you want input 5 to be the kill function, you must first change input 1 to another function.

Function *B* (sequence select) can configure up to three inputs as sequence select inputs. If you try to configure more than three inputs, the controller will only recognize the first three. It will revert back to the previous definitions for the additional inputs.

| Command | Description |
|---|---|
| **1INA1** | Configure input one as trigger input 1 |
| **1INB2** | Configure input two as sequence select input 1 |
| **1INB3** | Configure input three as sequence select input 2 |
| **1INC4** | Configure input four as kill input |
| **1INF5** | Configure input five as go input |
| **1IN** | Reports input configuration (*ABBCF) |

# IS—Input Status

❏ Command Type: Status
❏ Syntax: aIS
❏ Range: N/A
❏ Attributes: Immediate
Never Saved

❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: IN, LD, RSE
❏ Response to aIS is *nnnnnnnn

This command reports the status of all hardware inputs. The response is 8 ASCII digits ( Ø or 1) corresponding to the following I/O bits:

1—IN1 (Ø = Low, 1 = High)
2—IN2 (Ø = Low, 1 = High)
3—IN3 (Ø = Low, 1 = High)
4—IN4 (Ø = Low, 1 = High)
5—IN5 (Ø = Low, 1 = High)
6—CW limit (Ø = Low, 1 = High)
7—CCW limit (Ø = Low, 1 = High)
8—Fault (Ø = Low, 1 = High)

This is not a software status. It will report the actual hardware status of the inputs. **IS** can help you troubleshoot an application, to verify that limit switches, trigger inputs and home switches work.

| **Command** | **Response** |
|---|---|
| **2IS** | ***ØØØ1ØØØ1** (The input status of device 2 is reported: I/O bits 1-3 and 5-7 are low (grounded), and I/O bits 4 (IN4), and 8 (Fault), are high) |

# K—Kill

❏ Command Type: Motion
❏ Syntax: <a>K
❏ Range: N/A
❏ Attributes: Immediate
Never Saved

❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: IN,S

This command causes commanded motion to cease immediately. There is *NO* deceleration of the motor. The motor will servo around the position where the kill was entered. The load could be driven past limit switches and cause damage to the mechanism and possibly to the operation. In addition to stopping the motor, the **K** command will terminate a loop, end a time delay, abort downloading a sequence (**XD**), and clear the command buffer.

***WARNING***

The Kill (**K**) command is not an emergency stop. The motor is not disabled. Motion caused by instability or incorrect wiring will not be stopped. An emergency stop should cut power to the amplifier or interrupt the hardware enable input (pin 10), and mechanically prevent the motor from turning.

| Command | Description |
|---|---|
| **A5** | Sets acceleration to 5 revs/sec$^2$ |
| **V2** | Sets velocity to 2 revs/sec |
| **MC** | Sets mode to continuous |
| **G** | Executes the move (Go) |
| **K** | Stops the motor instantly |

# L—Loop

| | |
|---|---|
| ❏ Command Type: Programming | ❏ Valid Software Version: A |
| ❏ Syntax: <a>Ln | ❏ Units: number of loops |
| ❏ Range: n = 0 - 65,535 | ❏ Default Value: None |
| ❏ Attributes: Buffered<br>Savable in Sequence | ❏ See Also: C, N, U, Y |

When you combine the Loop (**L**) command with the End-of-Loop (**N**) command, all of the commands between **L** and **N** will be repeated the number of times indicated by n. If you enter **L** without a value specified for n, or with a Ø, subsequent commands will be repeated continuously. If you specify a value greater than 65,535, the loop will be repeated continuously.

The **N** command prompts the controller to proceed with further commands after the designated number of loops have been executed. The **Y** command stops loop execution after completing the current loop cycle. The Immediate Pause (**U**) command allows you to temporarily halt loop execution after completing the current loop cycle. You can use the Continue (**C**) command to resume loop execution.

| Command | Description |
|---|---|
| **L5** | Loop 5 times |
| **A5** | Sets acceleration to 5 revs/sec$^2$ |
| **V1Ø** | Sets velocity to 10 revs/sec |
| **D1ØØØØ** | Sets distance to 10,000 encoder counts |
| **G** | Executes the move (Go) |
| **N** | End of loop |

The commands in the loop will be executed 5 times.

# LD—Limit Disable

□ Command Type: Set-Up
□ Syntax: <a>LDn
□ Range: n = 0 - 3
□ Attributes: Buffered
Savable in Sequence
□ Valid Software Version: A
□ Units: See Below
□ Default Value: Ø
□ See Also: RA

The Limit Disable (**LD**) command allows you to enable/disable the end-of-travel limit switch protection. The **LDØ** condition does not allow the motor to turn without properly installing the limit inputs. If you want motion without wiring the limits, you must issue **LD3**. For machine and operator safety, hardware limits are highly recommended.

- Enable CCW and CW limits—***n = Ø (Default)***
- Disable CW limit—n = 1
- Disable CCW limit—n = 2
- Disable CCW and CW limits—n = 3

| Command | Description |
|---|---|
| **1LDØ** | Enables CW and CCW limits. The motor will move only if the limit inputs are bypassed or connected to normally-closed limit switches. |
| **1LD3** | Allows you to make any move, regardless of the limit input state. |

# LF—Line Feed

❏ Command Type: Programming
❏ Syntax: <a>LF
❏ Range: N/A
❏ Attributes: Buffered
Savable in Sequence

❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: CR

When you issue the Line Feed (**LF**) command, the system transmits a line feed character over the communications link. When the controller reaches this command in the buffer, it responds by issuing a line feed (ASCII 10) over its interface back to the host computer. If you place the **LF** command after a Go (**G**) command, it indicates when a move is complete. If you place the **LF** command after a Trigger (**TR**) command, it indicates when the trigger condition is met.

You can use the Carriage Return (**CR**) and **LF** commands with the Quote (**"**) command to display multiple-line messages via the RS-232C interface.

| Command | Description |
|---|---|
| **A5** | Sets acceleration to 5 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **D15ØØØ** | Sets distance to 15,000 encoder counts |
| **G** | Executes the move (Go) |
| **1LF** | Transmits a line feed character over the communications interface after the move is completed |

# MC—Mode Continuous

❏ Command Type: Motion
❏ Syntax: <a>MC
❏ Range: N/A
❏ Attributes: Buffered
Savable in Sequence

❏ Valid Software Version: A
❏ Units: N/A
❏ Default Status: Inactive
❏ See Also: MN, T, TR, V

The Mode Continuous (**MC**) command causes subsequent moves to ignore any distance parameter and move continuously. You can clear the **MC** command with the Mode Normal (**MN**) command.

The controller uses the previously defined Acceleration (**A**) and Velocity (**V**) commands to reach continuous velocity. Using the Time Delay (**T**), Trigger (**TR**), and Velocity (**V**) commands, you can achieve basic velocity profiling. After a new parameter is entered a Go (**G**) command is required. Acceleration (**A**) cannot be changed on the fly.

| Command | Description |
|---|---|
| **MC** | Sets mode to continuous |
| **A5** | Sets acceleration to 5 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **G** | Executes the move (Go) |
| **T1Ø** | Move at 5 revs/sec for 10 seconds |
| **V7** | Set velocity to 7 revs/sec |
| **G** | Change velocity to 7 revs/sec |
| **T1Ø** | Move at 7 revs/sec for 10 seconds |
| **VØ** | Set velocity to 0 rev/sec (stop) |
| **G** | Executes the **VØ** command |

The motor turns at 5 revs/sec for 10 seconds, then moves at 7 revs/ sec for 10 seconds before decelerating to a stop.

## MN—Mode Normal

- ❏ Command Type: Motion
- ❏ Syntax: <a>MN
- ❏ Range: N/A
- ❏ Attributes: Buffered
  Savable in Sequence
- ❏ Valid Software Version: A
- ❏ Units: N/A
- ❏ Default Status: Active
- ❏ See Also: A, D, G, MC, MPA, MPI

The Mode Normal (**MN**) command sets the positioning mode to preset. In Mode Normal, the motor will move the distance specified with the last distance (**D**) command. To define the complete move profile, you must define Acceleration (**A**), Velocity (**V**), and the Distance (**D**). The **MN** command is used to change the mode of operation from Mode Continuous (**MC**) back to normal or preset. To use the **MPA** or **MPI** command, you must be in Mode Normal (**MN**).

| Command | Description |
|---|---|
| **MN** | Set positioning mode to preset |
| **A5** | Set acceleration to 5 revs/sec$^2$ |
| **V5** | Set velocity to 5 revs/sec |
| **D1ØØØ** | Set distance to 1,000 encoder counts |
| **G** | Executes the move (Go) |

Motor turns 1,000 encoder counts CW after the **G** command is issued.

## MPA—Mode Position Absolute

- ❏ Command Type: Set-Up
- ❏ Syntax: <a>MPA
- ❏ Range: N/A
- ❏ Attributes: Buffered
  Savable in Sequence
- ❏ Valid Software Version: A
- ❏ Units: N/A
- ❏ Default Status: Inactive
- ❏ See Also: D, MN, MPI, PZ

This command sets the positioning mode to absolute. In this mode all move distances are referenced to absolute zero. In Mode Position Absolute (**MPA**), giving two consecutive Go (**G**) commands will cause the motor to move only once, since the motor will have achieved its desired absolute position at the end of the first move.

**MPA** is most useful in applications that require moves to specific locations while keeping track of the beginning position.

You can set the absolute counter to zero by cycling power or issuing a Position Zero (**PZ**) command. You must be in Normal mode (**MN**) to use this command. In continuous mode (**MC**), **MPA** is ignored.

| Command | Description |
|---|---|
| **MN** | Sets Normal mode (preset) |
| **PZ** | Resets absolute counter to zero |
| **MPA** | Sets position mode absolute |
| **A5** | Sets acceleration to 5 revs/sec$^2$ |
| **V1Ø** | Sets velocity to 10 revs/sec |
| **D4ØØØØ** | Sets destination to absolute position 40,000 |
| **G** | Motor will move to absolute position 40,000 |
| **D1ØØØØ** | Sets destination to absolute position +10,000 |
| **G** | Motor will move to absolute position +10,000 |

The motor will move 40,000 encoder counts in the CW direction (if starting from position Ø) and then move 30,000 encoder counts in the CCW direction to reach the absolute position 10,000.

## MPI—Mode Position Incremental

- ❏ Command Type: Set-Up
- ❏ Syntax: <a>MPI
- ❏ Range: N/A
- ❏ Attributes: Buffered
  Savable in Sequence
- ❏ Valid Software Version: A
- ❏ Units: N/A
- ❏ Default Status: Active
- ❏ See Also: D, MN, MPA

This command sets the positioning mode to incremental. In incremental mode all move distances specified with the Distance (**D**) command will be referenced to the current position. Mode Position Incremental (**MPI**) is most useful in applications that require repetitive movements, such as feed to length applications.

You must be in normal mode (**MN**) to use this command. In continuous mode (**MC**), this command is ignored.

| Command | Description |
|---|---|
| **MN** | Set positioning mode normal (preset) |
| **MPI** | Set positioning mode incremental |
| **A5** | Sets acceleration to 5 revs/sec$^2$ |

| | |
|---|---|
| **V1Ø** | Sets velocity to 10 revs/sec |
| **D1ØØØØ** | Sets distance of move to 10,000 encoder counts |
| **G** | Move 10,000 encoder counts CW |
| **G** | Move another 10,000 encoder counts CW |

The motor moves 10,000 encoder counts CW after each **G** command (total move is 20,000 encoder counts).

## N—End of Loop

❑ Command Type: Programming
❑ Syntax: <a>N
❑ Range: N/A
❑ Attributes: Buffered
Savable in Sequence

❑ Valid Software Version: A
❑ Units: N/A
❑ Default Value: N/A
❑ See Also: C, L, PS, U

This command marks the end of a loop. You must use this command in conjunction with the Loop (**L**) command. All buffered commands that you enter between the **L** and **N** commands are executed as many times as the number that you enter following the **L**

| **Command** | **Description** |
|---|---|
| **MN** | Sets move to Normal mode |
| **A5** | Sets acceleration to 5 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **D1ØØØØ** | Sets move distance to 10,000 encoder counts |
| **L5** | Loops the following commands five times |
| **G** | Executes the move (Go) |
| **N** | Ends the loop |

The move will be executed five times.

## OFF—De-Energize Drive

❑ Command Type: Programming
❑ Syntax: <a>OFF
❑ Range: N/A
❑ Attributes: Immediate
Never Saved

❑ Valid Software Version: A
❑ Units: N/A
❑ Default Value: N/A
❑ See Also: ST, ON

This command de-energizes the drive immediately. This command can be used to shutdown the drive in an emergency. This command is functionally identical to the **ST1** command. With the drive de-energized you may still communicate with the controller.

| **Command** | **Description** |
|---|---|
| **OFF** | De-energize the drive |
| **1IS** | *00000001 (fault bit active) |

# ON—Energize Drive

- ❏ Command Type: Programming
- ❏ Syntax: <a>OFF
- ❏ Range: N/A
- ❏ Attributes: Immediate
  Never Saved
- ❏ Valid Software Version: A
- ❏ Units: N/A
- ❏ Default Value: N/A
- ❏ See Also: ST, OFF

This command energizes the drive immediately. This command can be used to re-energize the drive after a shutdown. This command is also used to re-energize after a fault condition such as excessive position error.

| Command | Description |
|---|---|
| **ON** | Energize the drive |
| **1IS** | *00000000 (fault bit inactive) |

# O—Output

- ❏ Command Type: Programming
- ❏ Syntax: <a>Onn
- ❏ Range: Ø, 1 or X (See Below)
- ❏ Attributes: Buffered
  Savable in Sequence
- ❏ Valid Software Version: A
- ❏ Units: on, off, or unchanged
- ❏ Default Value: ØØ
- ❏ See Also: OS, S, TR

The Output (**O**) command turns the programmable output bits on and off. This is used for signaling remote controllers, turning on LEDs, or sounding whistles. The output can indicate that the motor is in position, about to begin its move, or is at constant velocity, etc.

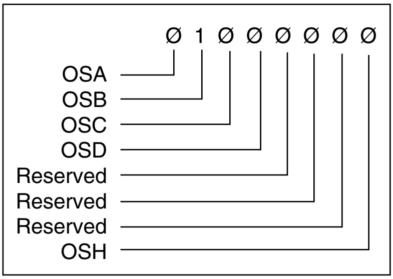
**n=1** = Turns output bits on
**n=Ø** = Turns output bits off
**n=X** = Leaves output bits unchanged

| Command | Description |
|---|---|
| **MN** | Set to Mode Normal |
| **A1Ø** | Set acceleration to 10 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **D2ØØØØ** | Set move distance to 20,000 encoder counts |
| **OØ1** | Set programmable output 1 off and output 2 on |
| **G** | Executes the move (Go) |
| **OØØ** | After the move ends, turn off both outputs |

# OS—Report Homing Function Set-Ups

❏ Command Type: Status
❏ Syntax: <a>OS
❏ Range: N/A
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: OS(A-H)
❏ Response to aOS is nnnnnnnn

This command results in a report of which software switches have been set by **OS** commands. The reply is eight digits. This command reports **OSA** through **OSH** Set-up status in binary format. The digit 1 represents ON (enabled), the digit Ø represents OFF (disabled). The default response is ***Ø1ØØØØØØ**.



# OSA—Define Active State of End-of-Travel Limits

❏ Command Type: Set-Up
❏ Syntax: <a>OSAn
❏ Range: n = Ø, 1
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: NA
❏ Default Value: Ø
❏ See Also: LD, OSC

**OSAØ**: Normally Closed Contacts
**OSA1**: Normally Open Contacts
This command sets the active state of the CW and CCW end-of-travel limit inputs. It enables you to use either normally closed or normally open switches.

| Command | Description |
|---|---|
| **OSA1** | Sets active state for normally open limit switches |
| **OSCØ** | Sets active state of home input closed (low) |
| **OSH1** | Selects the CCW side of the home signal as the edge on which the final approach will stop |

# OSB—Back Up To Home

| | |
|---|---|
| ❏ Command Type: Set-Up | ❏ Valid Software Version: A |
| ❏ Syntax: <a>OSBn | ❏ Units: See Below |
| ❏ Range: n = Ø, 1 | ❏ Default Value: 1 |
| ❏ Attributes: Buffered, Savable in Sequence | ❏ See Also: GH, OSC, OSD, OSH |

**OSBØ**: Back up to home
**OSB1**: Back up to selected edge

This command is used to make homing more repeatable. With Back Up to Selected Home (**OSB**) command enabled, homing is a two step process. First it approaches the home switch at high speed (set by the GH command) until it sees the switch. Then it decelerates and returns to the switch at slow speed. Since it always makes its final approach from the same direction, the system will act differently whether the active edge of the switch is the first or second edge encountered.

If the selected edge for final home position is the first edge encountered the motor will decelerate to 0 velocity, when that edge is detected. The motor will then reverse direction and stop on the selected edge.

If the selected edge for the final home position is the second edge encountered the motor will travel until that edge is detected. The motor will decelerate to a 0 velocity. The controller will then position the motor 1/2 of a revolution on the outside of the selected edge. Finally the motor will creep at 0.1 rev/sec in the direction of the active home region, until home is detected.

If the motor is already in the active home region and the Go Home (**GH**) command is given, the motor will travel in the direction of the edge for the final home position. The motor will decelerate to 0 velocity, reverse direction and approach home at the creep velocity until home is detected.

With **OSB** disabled, the motor will decelerate to 0 velocity after encountering the active home region, and will be considered to be at home if the home limit input is still active. If the deceleration overshoots the active home region the motor will reverse direction and travel back to the active home region.

| Command | Description |
|---|---|
| **OSB1** | Sets back up to home switch active |
| **OSCØ** | Sets active state of home input closed (low) |
| **OSH1** | Selects the CCW side of the home signal as the edge on which the final approach will stop |

## OSC—Define Active State of Home Switch

❏ Command Type: Set-Up
❏ Syntax: <a>OSCn
❏ Range: n = Ø, 1
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: NA
❏ Default Value: Ø
❏ See Also: GH, OSB, OSD, OSH

**OSCØ**: Active state of home input is n = Ø (closed)
**OSC1**: Active state of home input is n=1 (open)

**OSCØ** requires that a normally open (high) switch be connected to the home limit input. **OSC1** requires that a normally closed (low) switch be connected to the home limit input.

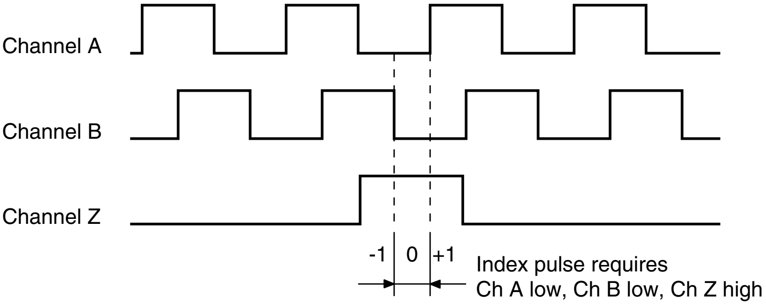
| **Command** | **Description** |
|---|---|
| **OSC1** | Sets the active state of the home input to open |

## OSD—Enable Encoder Z Channel for Home

❏ Command Type: Set-up
❏ Syntax: <a>OSDn
❏ Range: n = 0, 1
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: 0
❏ See Also: OSB,OSC,OSH,GH

**OSDØ** = Do not reference Z Channel during homing
**OSD1** = Reference Z Channel during homing

The encoder Z channel is used (in conjunction with a load activated switch connected to the home limit) to determine the home position. The switch determines the home region, and the Z channel determines the exact and final home position inside the home region. As the next drawing shows, the final home position occurs when Channel A is low, Channel B is low, and Channel Z is high.



For OSD1 to be selected, OSB1 must also be selected.

| **Command** | **Description** |
|---|---|
| **OSD1** | Recognizes Z channel as final home reference |

# OSH—Reference Edge of Home Switch

❑ Command Type: Set-Up
❑ Syntax: <a>OSHn
❑ Range: n = Ø, 1
❑ Attributes: Buffered, Savable in Sequence
❑ Valid Software Version: A
❑ Units: NA
❑ Default Value: Ø
❑ See Also: GH, OSB, OSC, OSD

**OSHØ**: Selects the CW side of the Home signal as the edge on which the final approach will stop
**OSH1**: Selects the CCW side of the home signal as the edge on which the final approach will stop

The CW edge of the Home switch is the first switch transition seen by the controller when traveling from the CW limit in the CCW direction. If n = 1, the CCW edge of the Home switch will be referenced as the Home position. The CCW edge of the Home switch is the first switch transition seen by the controller when traveling from the CCW limit in the CW direction. If n = Ø, the CW edge of the Home switch will be referenced as the Home position.

| Command | Description |
|---|---|
| **OSB1** | Sets back up to home switch active |
| **OSCØ** | Sets active state of home input closed (low) |
| **OSH1** | Selects the CCW side of the home signal as the edge on which the final approach will stop |

The home limit becomes active when the home limit input is closed. The controller recognizes the CCW edge of the switch as the home limit and backs up to that edge to complete the Go Home move.

# PR—Absolute Position Report

❑ Command Type: Status
❑ Syntax: aPR
❑ Range: ±1,073,741,820
❑ Attributes: Buffered, Savable in Sequence
❑ Valid Software Version: A
❑ Units: Encoder counts
❑ Default Value: N/A
❑ See Also: D, MPA, MPI, MN, PZ, PX
❑ Response to aPR is *±nnnnnnnnnn

This command reports the commanded motor position relative to the power-up position. When a **D** command is issued, the distance is relative to the value of **PR**. The difference between the commanded position (**PR**) and the actual encoder position (**PX**) is the position error (**DPE**). The controller is always trying to minimize the position error. You can reset the encoder position counter to zero by using the position zero (**PZ**) command or reset (**Z**). After **PZ** the encoder position (**PX**) will be set to zero. If there was a position error before the **PZ** was issued, the value of **PR** will differ from the value of **PX** by the amount of the position error. Increasing integral gain (**CIG**) can help reduce the position error at rest thus insuring the value of **PR** equals the value of **PX**.

| Command | Description |
|---|---|
| **1PR** | Commanded position report. (*+ØØØØØØ2ØØØ) |
| **1PX** | Encoder position report (*+ØØØØØØ2ØØ5) |

The actual motor position is 5 encoder counts from the commanded position.

## PS—Pause

❏ Command Type: Programming
❏ Syntax: <a>PS
❏ Range: N/A
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: C, U

This command pauses execution of a command string or sequence until the controller receives a Continue (**C**) command. **PS** lets you enter a complete command string before running other commands. **PS** is also useful for interactive tests and synchronizing multiple controllers that have long command strings.

| Command | Description |
|---|---|
| **PS** | Pauses execution of commands until the controller receives the Continue (C) command |
| **A5** | Sets acceleration to 5 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **D4ØØØ** | Sets move distance to 4,000 encoder counts |
| **G** | Executes the move (Go) |
| **T2** | Delays the move for 2 seconds |
| **G** | Executes the move (Go) |
| **C** | Continues Execution |

When the controller receives the **C** command, the motor moves 4,000 encoder counts twice with a 2 second delay between moves.

## PX—Report Absolute Encoder Position

❏ Command Type: Status
❏ Syntax: aPX
❏ Range: ±1,073,741,820
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: Encoder counts
❏ Default Value: N/A
❏ See Also: PR, PZ
❏ Response to aPX *±nnnnnnnnn

This command reports the actual motor position as measured by the encoder. When a **D** command is issued, the distance is relative to the value of **PR** not the value of **PX**. The difference between the commanded position (**PR**) and the actual encoder position (**PX**) is the position error (**DPE**). The controller is always trying to minimize the position error. You can reset the encoder position counter to zero by using the position zero (**PZ**) command or reset (**Z**). After **PZ** the encoder position (**PX**) will be set to zero. If there was a position error before the **PZ** was issued, the value of **PR** will differ from the value of **PX** by the amount of the position error. Increasing integral gain (**CIG**) can help reduce the position error at rest thus insuring the value of **PR** equals the value of **PX**.

| Command | Description |
|---|---|
| **MN** | Set to mode normal |
| **PZ** | Sets the absolute counter to zero |
| **A1Ø** | Sets acceleration to 10 rev/sec2 |
| **V5** | Sets velocity to 5 rev/sec |
| **D46ØØ** | Sets move distance to 4,600 encoder counts |
| **G** | Executes the move (Go) |
| **1PX** | After the motor executes the move, the encoder position is reported: The response is *+0000004600. |

## PZ—Set Absolute Counter to Zero

❏ Command Type: Programming
❏ Syntax: <a>PZ
❏ Range: N/A
❏ Attributes: Buffered, Never Saved
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: D, MN, PR, PX

This command sets the absolute encoder position counter to zero. If there was a position error before the **PZ** was issued, the new value of **PR** will differ from the value of **PX** by the amount of the position error. Absolute counter will also be set to zero when you cycle power or when you successfully execute a homing (**GH**) function.

| Command | Description |
|---|---|
| **MN** | Enter position mode (mode normal) |
| **MPA** | Makes preset moves from absolute zero position |
| **PZ** | Sets absolute position counter to zero |
| **A1Ø** | Sets acceleration to 10 rev/sec$^2$ |
| **V5** | Sets velocity to 5 rev/sec |
| **D4ØØØØ** | Sets move distance to 40000 encoder counts |
| **G** | Executes the move (Go) |
| **1PX** | Reports absolute encoder position (*+0000040000) |

| | |
|---|---|
| **PZ** | Sets the absolute counter to zero |
| **1PX** | Reports absolute encoder position (*+0000000000) |

## “—Quote

- ❏ Command Type: Programming
- ❏ Syntax: “x
- ❏ Range: x = up to 17 ASCII characters
- ❏ Attributes: Buffered, Savable in Sequence
- ❏ Valid Software Version: A
- ❏ Units: N/A
- ❏ Default Value: N/A
- ❏ See Also: CR, LF
- ❏ Response to “x is x

Up to 17 characters entered after the quotation marks (**“**) will be transmitted, exactly as they are entered, over the RS-232C link. A space entered by the space bar indicates the end of the command. A space is always sent after the last character in the string. This command is used during buffered moves or sequences to command other devices to move, or to send the message to a remote display. On a daisy chain of multiple units, if more than one unit is reporting a message at once, the messages will overlap and be garbled.

| **Command** | **Description** |
|---|---|
| **PS** | Pause execution until Continue (C) is entered |
| **A5** | Set acceleration to 5 revs/sec$^2$ |
| **V5** | Set velocity to 5 revs/sec |
| **D2ØØØ** | Set distance to 2,000 encoder counts |
| **G** | Executes the move (Go) |
| **“MOVE_DONE** | Transmits message |
| **C** | Continue move |

After the move, the controller will send the message MOVE_DONE via the RS-232C port

## QØ—Exit Velocity Profiling Mode

- ❏ Command Type: Set-Up
- ❏ Syntax: <a>QØ
- ❏ Range: N/A
- ❏ Attributes: Immediate, Never Saved
- ❏ Valid Software Version: A
- ❏ Units: N/A
- ❏ Default Value: N/A
- ❏ See Also: Q1, RM

The **QØ** command exits the Velocity Profiling mode. The motor will stop when **QØ** is issued. Entering this command will cause the controller to enter Normal mode (**MN**).

# Q1—Enter Velocity Profiling Mode

❏ Command Type: Set-Up
❏ Syntax: <a>Q1
❏ Range: N/A
❏ Attributes: Immediate, Never Saved
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: QØ, RM

**Q1** activates Velocity Profiling mode. Subsequent **RM** commands will immediately change motor velocity. **QØ** exits this mode.

| Command | Description |
|---|---|
| **ER2ØØØ** | Set encoder resolution to 2000 |
| **Q1** | Enter Velocity Streaming mode |
| **RMØØØØ22ØC** | Accelerate to 0.25 revs/sec$^2$ |
| **RMØØØØ4418** | Accelerate to 0.5 revs/sec$^2$ |
| **RMØØØØ8831** | Accelerate to 1 revs/sec$^2$ |
| **RMØØØ11Ø62** | Accelerate to 2 revs/sec$^2$ |
| **RMØØØØ8831** | Decelerate to 1 revs/sec$^2$ |
| **RMØØØØ4418** | Decelerate to 0.5 revs/sec$^2$ |
| **RMØØØØ22ØC** | Decelerate to 0.25 revs/sec$^2$ |
| **RMØØØØØØØØ** | Decelerate to 0 revs/sec$^2$ |
| **QØ** | Exit Velocity Streaming mode |

# R—Request Controller Status

❏ Command Type: Status
❏ Syntax: aR
❏ Range: N/A
❏ Attributes: Immediate, Never Saved
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: RA, RB, RC, XSR, XSS
❏ Response to aR is *x

The Request Controller Status (**R**) command can be used to indicate the general status of the controller. Possible responses are:

| Character | Definition |
|---|---|
| ***R** | Ready |
| ***S** | Ready, Attention Needed |
| ***B** | Busy |
| ***C** | Busy, Attention Needed |

When the controller is not prepared to accept another command, the following conditions will cause a controller is busy (***B**) response:

* Performing a move
* Accelerating/decelerating during a continuous move
* A time delay is in progress. (T command)
* In RM mode
* Paused
* Waiting on a Trigger
* Going Home
* In Power-on sequence mode
* Running a sequence
* Executing a loop

The following conditions will cause an error (***S** or ***C**) response:

* Go home failed
* Limit has been encountered
* Sequence execution was unsuccessful
* Sequence memory checksum error
* Undervoltage
* Drive recently enabled
When the response indicates that attention is required, the **RA**, **RB**, **RC**, **XSR**, or **XSS** commands can provide details about the error.

It is not recommended that this command be used in tight polling loops that could result in microprocessor overload. Time delays can alleviate this problem.

This command is not intended to be used to determine if a move is complete. It should be used after a move is complete to determine if errors or faults exist. Use a buffered status request (**CR** or **LF**) command or a programmable output to indicate move completion.

| Command | Response |
|---|---|
| **1R** | ***R** (Controller ready to accept a command, and no error conditions exist.) |

## RA—Limit Switch Status Request

❑ Command Type: Status
❑ Syntax: aRA
❑ Range: N/A
❑ Attributes: Immediate, Never Saved
❑ Valid Software Version: A
❑ Units: N/A
❑ Default Value: N/A
❑ See Also: R, RB
❑ Response to aRA is *x

The **RA** command responds with the status of the end-of-travel limits during the last move as well as the present condition. This is done by responding with one of 12 characters representing the conditions listed below.

| ***Response*** | ***Last Move Terminated By*** | | ***Current Limit Status*** | |
|---|---|---|---|---|
| ***Character*** | ***CW Limit—*** | ***CCW Limit*** | ***CW Limit—*** | ***CCW Limit*** |
| *@ | No | No | Off | Off |
| *A | Yes | No | Off | Off |
| *B | No | Yes | Off | Off |
| *D | No | No | On | Off |
| *E | Yes | No | On | Off |
| *F | No | Yes | On | Off |
| *H | No | No | Off | On |
| *I | Yes | No | Off | On |
| *J | No | Yes | Off | On |
| *L | No | No | On | On |
| *M | Yes | No | On | On |
| *N | No | Yes | On | On |

The **RA** command is useful when the motor will not move in either or both directions. The report back will indicate if the last move was terminated by one or both end-of-travel limits. This command is not intended to be used to determine if a move is complete. It should be used after a move to determine if errors or faults exist. If you are hitting a limit switch, the Request Status (**R**) will return a ***S**.

| Command | Response |
|---|---|
| **1RA** | ***@** (the last move was not terminated by a limit and no limits are currently active.) |

## RB—Loop, Pause, Shutdown, Trigger Status Request

- ❑ Command Type: Status
- ❑ Syntax: aRB
- ❑ Range: N/A
- ❑ Attributes: Immediate, Never Saved
- ❑ Valid Software Version: A
- ❑ Units: N/A
- ❑ Default Value: N/A
- ❑ See Also: L, PS, R, RA, ST, TR
- ❑ Response to aRB is *x

This command receives a response from *@ to ***O**, as defined below. The four conditions for which status is indicated are as follows:

**Loop Active**: A loop is in progress.
**Pause Active**: Buffered commands waiting for a Continue (**C**).
**Shutdown Active**: The motor is shutdown by the **ST1** command.
**Trigger Active**: At least one trigger is active.

| *Response Character* | *Loop Active* | *Pause Active* | *Shutdown Active* | *Trigger Active* |
|---|---|---|---|---|
| *@ | No | No | No | No |
| *A | Yes | No | No | No |
| *B | No | Yes | No | No |
| *C | Yes | Yes | No | No |
| *D | No | No | Yes | No |
| *E | Yes | No | Yes | No |
| *H | No | No | No | Yes |
| *I | Yes | No | No | Yes |
| *J | No | Yes | No | Yes |
| *K | Yes | Yes | No | Yes |
| *L | No | No | Yes | Yes |
| *M | Yes | No | Yes | Yes |
| *N | No | Yes | Yes | Yes |
| *O | Yes | Yes | Yes | Yes |

This command is not intended to be used to determine if a move is complete. It should be used after the move is complete to determine if errors or faults exist.

| Command | Response |
|---|---|
| **1RB** | ***A** (After issuing a 1RB command, the response came back as *A. This means that the controller is currently executing a loop.) |

# RC—Homing Status Request

❏ Command Type: Status
❏ Syntax: aRC
❏ Range: N/A
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ Response to aRC IS *x
❏ See Also: R, RA, RB, FS, GH

The RC command has the same response format of RA and RB. The condition for which status is indicated is:

*Homing Function Failure:*
In this condition, the controller has encountered both End-of-Travel limits or one of several possible Stop commands or conditions. Go Home motion was concluded, but not at Home.

| Response Character | Go Home Successful? |
|---|---|
| *@ | YES |
| *B | NO |

| Command | Description |
|---|---|
| **1RC** | *B Go home was unsuccessful. |

# RFS—Return Servo Gains to Factory Settings

❏ Command Type: Status
❏ Syntax: aRFS
❏ Range: N/A
❏ Attributes: Immediate, Never Saved
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: RA, RB, RC, XSR, XSS
❏ Response to aRFS is *x

This command can be used to return all the servo gains to the factory defaults immediately. This command is useful when the current servo gain values are inappropriate and retuning is required. By returning to factory defaults the gains will be in proper relationship to one another allowing fine tuning of the system. The factory defaults are:

| | | |
|---|---|---|
| Proportional Gain | (**CPG**) | 16 |
| Integral Gain | (**CIG**) | 2 |
| Derivative Gain | (**CDG**) | 240 |
| Derivative Sampling Period | (**CTG**) | 0 |

| | | |
|---|---|---|
| Position Error | (**CPE**) | 4000 |
| Integral Limit | (**CIL**) | 2 |

| Command | Description |
|---|---|
| **1RFS** | All servo gains returned to the factory defaults |

## RM—Rate Multiplier in Velocity Streaming

❑ Command Type: Motion
❑ Syntax: <a>RMn
❑ Range: n = Ø - FFFFFFFF
❑ Attributes: Immediate, Never Saved
❑ Valid Software Version: A
❑ Units: revs/sec
❑ Default Value: None
❑ See Also: D, H, QØ, Q1

The **RM** command sets an immediate velocity where n represents an 8-digit hexadecimal value. The value for n is determined with the following formula:

*( desired revs/sec) • (encoder resolution ) * 17.432576* = **decimal # for velocity value to be rounded off to the closest whole number.**

The resulting decimal number must be converted to a hexadecimal number to obtain the value for n.

The velocity change is instant—there is no acceleration/deceleration ramp between velocities. A limit switch closure will stop movement in Velocity Profiling mode, but does not cause the controller to exit Velocity Streaming mode. To recover from a limit stop in **RM** mode, **QØ** must be issued and the direction must be changed. Velocity Profiling mode is unidirectional. The last direction set either from a move or from a Distance (**D**) or Direction (**H**) command will be used. Bidirectional moves can be made in this mode by returning to velocity zero (Ø), turning **RM** mode off, changing the direction, and re-enabling **RM** mode. Exiting **RM** mode with **QØ** causes the controller to enter Normal mode (**MN**).

| Command | Response |
|---|---|
| **ER2ØØØ** | Set encoder resolution to 2000 |
| **Q1** | Enter Velocity Streaming mode |
| **RMØØØØ22ØC** | Accelerate to 0.25 revs/sec$^2$ |
| **RMØØØØ4418** | Accelerate to 0.5 revs/sec$^2$ |
| **RMØØØØ8831** | Accelerate to 1 revs/sec$^2$ |
| **RMØØØ11Ø62** | Accelerate to 2 revs/sec$^2$ |
| **RMØØØØ8831** | Decelerate to 1 revs/sec$^2$ |
| **RMØØØØ4418** | Decelerate to 0.5 revs/sec$^2$ |
| **RMØØØØ22ØC** | Decelerate to 0.25 revs/sec$^2$ |
| **RMØØØØØØØØ** | Decelerate to 0 revs/sec$^2$ |
| **QØ** | Exit Velocity Streaming mode |

# RSE—Report Servo Errors

❏ Command Type: Status
❏ Syntax: aRSE
❏ Range: N/A
❏ Attributes: Immediate, Never Saved
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: R,RA, RB, RC
❏ Response to aRSE is *x

The Report Servo Errors (**RSE**) command can be used to indicate the general status of the controller. Possible responses are:

| Character | Definition |
|---|---|
| ***0** | No errors |
| ***2** | Excessive position error |
| ***4** | Drive fault or not enabled |
| ***6** | Commanded shutdown |
| ***8** | Undervoltage, or drive was recently enabled |

During a fault condition, the RSE command can be used to interrogate the controller for the reason of the fault. The following can cause a latched fault condition—the red LED will be illuminated, and the fault output will be active:

-Drive commanded shutdown via the **OFF** or **ST1** commands

-Position error exceeded the value set in the **CPE** command

-Hardware enable line (pin 11) not grounded

-Drive fault (overvoltage, overtemperature, etc.)

The next two conditions are not latched, and do not illuminate the red LED. They will cause a ***8** response to **RSE**, until the next move command is issued.

-Undervoltage (voltage at drive's DC input drops below 21.5VDC)

-Drive was recently enabled

If a fault condition occurs but RSE reports no errors, then the drive is in current foldback mode. Please consult *Chapter ⑦ Troubleshooting.*

| Command | Response |
|---|---|
| **1RSE** | ***0** (Controller ready to accept a command, and no error conditions exist.) |

# RV—Revision Level

❑ Command Type: Status
❑ Syntax: aRV
❑ Range: N/A
❑ Attributes: Buffered, Savable in Sequence

❑ Valid Software Version: A
❑ Units: N/A
❑ Default Value: N/A
❑ See Also:
❑ Response to aRV is nn-nnnnnn-nnnn

The Revision (**RV**) command responds with the software part number and its revision level. The response is in the form shown below:

***92-nnnn-nn**<***xn***>[cr]
(**part number**, ***revision level***)

The part number identifies which product the software is written for, as well as any special features that the software may include. The revision level identifies when the software was written. You may want to record this information in your own records for future use. This type of information is useful when you consult Parker Compumotor's Applications Department.

| **Command** | **Response** |
|---|---|
| **1RV** | **92-Ø16637-Ø1A** |

The product is identified by *92-Ø16637-Ø1A, and the revision level is identified by A.

# S—Stop

❑ Command Type: Motion
❑ Syntax: <a>S
❑ Range: N/A
❑ Attributes: Immediate, Never Saved

❑ Valid Software Version: A
❑ Units: N/A
❑ Default Value: N/A
❑ See Also: A, K, QØ, SSH, ST

This command decelerates the motor to a stop using the last defined Acceleration (**A**) command. This command clears the command buffer (at the end of a move, if one is in progress). The Sequence Definition (**XD**) command is aborted and a time delay is terminated. If **SSH1** is set the controller will stop the current move but it will not clear the command buffer.

| **Command** | **Description** |
|---|---|
| **MC** | Sets move in continuous mode |
| **A1** | Sets acceleration to 1 revs/sec$^2$ |
| **V1Ø** | Sets velocity to 10 revs/sec |
| **G** | Executes the move (Go) |
| **S** | Stops motor (motor comes to a stop at a deceleration rate of 1 revs/sec$^{2)}$ |

The **S** command is not buffered. As soon as the controller receives the **S** command, it stops motion.

# SN—Scan

❏ Command Type: Set-Up
❏ Syntax: <a>SNn
❏ Range: 1 - 1000
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: n = mS
❏ Default Value: 50
❏ See Also: XP

The Scan (**SN**) command allows you to define the *debounce time* (in milliseconds) for external sequence selection inputs. The debounce time is the amount of time that the sequence inputs must remain constant for a proper reading from a remote controller, such as a programmable logic controller (PLC). If you are using a PLC you should change the debounce time to match the *on time* of the PLC outputs.

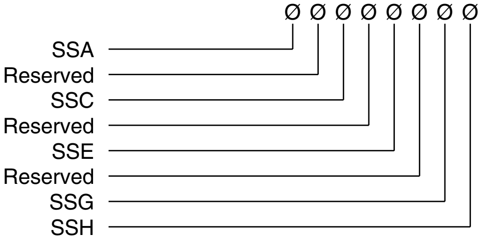
This command allows you to select the best possible trade-off between noise immunity and speed for a given application. If you make your scan time too short, the controller may respond to an electrical glitch. If you issue the Scan command with only a device address (**1SN**), the controller will respond with the current debounce time (***SNn**).

| Command | Description |
|---|---|
| **SN1Ø** | Sets scan time of sequence select inputs to 10 ms |

# SS—Software Switch Function Status

❏ Command Type: Status
❏ Syntax: aSS
❏ Range: N/A
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: SSA, SSC, SSG, SSH
❏ Response to aSS is *nnnnnnnn

This command reports the status of the **SS** commands. From left to right, the 8-character response corresponds to **SSA** through **SSH**.



## SSA—RS-232C Echo Control

❏ Command Type: Set-Up
❏ Syntax: <a>SSAn
❏ Range: n = Ø, 1
❏ Attributes: Buffered, Savable In Sequence
❏ Valid Software Version: A
❏ Units: See Below
❏ Default Value: Ø
❏ See Also:

This command turns the RS-232C echo (transmission of characters received from the remote device by the controller) on and off.

**SSAØ** = Echo on
**SSA1** = Echo off

In the Echo On (**SSAØ**) mode, characters that are received by the controller are echoed automatically. In the Echo Off (**SSA1**) mode, characters are not echoed from the controller. This command is useful if your computer cannot handle echoes. In a daisy chain, you must have the echo on (**SSAØ**) to allow controllers further down the chain to receive commands. *Status commands do not echo the command sent, but transmit the requested status report.*

| Command | Description |
|---|---|
| **SSA1** | Turns echo off (Characters sent to the controller are not echoed back to the host.) |

## SSC—Output #1 on In Position

❏ Command Type: Set-up
❏ Syntax: <a>SSCn
❏ Range: n = 0 or 1
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: 0
❏ See Also: CEW, CIT

With **SSC** set to 1, output 1 will turn on when the motor is within the In Position window for the specified time defined by the **CEW** and **CIT** commands

b = 0: Normal
b = 1: Output 1 is configured as an "In Position" output

| Command | Response |
|---|---|
| **SSC1** | Set output 1 as an In Position output |

# SSE—Enable/Disable Communication Error Checking

- ❏ Command Type: Set-Up
- ❏ Syntax: <a>SSEn
- ❏ Range: 0 (disable), 1 (enable)
- ❏ Attributes: Buffered, Savable in Sequence
- ❏ Valid Software Version: E
- ❏ Units: N/A
- ❏ Default Value: 0 (disable)
- ❏ See Also: %

This command setting determines whether or not each byte received at the controller is checked for communication errors. **SSE1** enables error checking for all bytes received at the controller, and **SSE0** disables error checking. See the **%** command for the types of errors detected.

# SSG—Clear/Save the Command Buffer on Limit

- ❏ Command Type: Set-Up
- ❏ Syntax: <a>SSGn
- ❏ Range: n = Ø, 1
- ❏ Attributes: Buffered, Savable In Sequence
- ❏ Valid Software Version: A
- ❏ Units: See Below
- ❏ Default Value: Ø
- ❏ See Also: LD

**SSGØ** = Clears command buffer on limit
**SSG1** = Saves command buffer on limit

In most cases, it is desirable that upon activating an end-of-travel limit input all motion should cease until the problem causing the over-travel is rectified. This will be assured if all commands pending execution in the command buffer are cleared when hitting a limit. This is the case if **SSGØ** is specified. If **SSG1** is specified and a limit is activated, the current move is aborted, but the remaining commands in the buffer continue to be executed.

| Command | Description |
|---|---|
| **SSG1** | Saves buffer on limit |
| **A1Ø** | Sets acceleration to 10 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **D4ØØØ** | Sets distance to 4,000 encoder counts |
| **G** | Executes the move (Go) |
| **O11** | Turn on outputs 1 and 2 |

If a limit switch is encountered while executing the move, outputs 1 and 2 will still go on.

## SSH—Clear/Save Command Buffer on Stop

❏ Command Type: Set-Up
❏ Syntax: <a>SSHn
❏ Range: n = Ø, 1
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: See Below
❏ Default Value: Ø
❏ See Also: S

**SSHØ** = Clears command buffer on stop
**SSH1** = Saves command buffer on stop

In Normal Operation (**SSHØ**) the Stop (**S**) command or a dedicated stop input will cause any commands in the command buffer to be cleared. If you select the Save Command Buffer On Stop (**SSH1**) command, a Stop (**S**) command will only stop execution of a move in progress. It will not stop execution of any commands that remain in the buffer. However, it will terminate a loop in the current pass.

| Command | Description |
|---|---|
| **SSHØ** | Clears buffer on stop |
| **A1Ø** | Sets acceleration to 10 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **D4ØØØ** | Sets distance to 4,000 encoder counts |
| **L5Ø** | Loops 50 times |
| **G** | Executes the move (Go) |
| **T.5** | Pauses the motor 500 ms |
| **N** | Ends Loop |
| **S** | Stops motion |

When **S** is issued, the controller will clear the buffer and stop the move.

## ST—Shutdown

❏ Command Type: Programming
❏ Syntax: <a>STn
❏ Range: n = Ø, 1
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: See Below
❏ Default Value: Ø
❏ See also: OFF, ON

The Shutdown (**ST1**) command rapidly decreases motor current to zero. The system ignores move commands that are issued after **ST1**. Torque on the motor is not maintained after you issue **ST1**.

The **STØ** command rapidly increases motor current to normal. Once you restore the current, you can execute moves. **ST1** reduces motor heating, and allows you to manually position the load. The position counter is set to the current position when you enter an **STØ** command. If you re-enable the drive using **STØ**, you must wait 500 ms before using other commands. The **ST** command is functionally identical to the **ON** and **OFF** commands

For the **ST** command to work, the enable line (pin 11) must be connected to ground.

| Command | Description |
|---|---|
| **ST1** | Shuts off current to the motor |

## T—Time Delay

- ❏ Command Type: Programming
- ❏ Syntax: <a>Tn
- ❏ Range: n = 0.01 - 99999.99
- ❏ Attributes: Buffered, Savable in Sequence
- ❏ Valid Software Version: A
- ❏ Units: seconds
- ❏ Default Value: None

The Time (**T**) command causes the controller to wait the number of seconds that you specify before it executes the next command in the buffer. This command is useful whenever you need to delay the motor's actions or when you wish to move the motor in continuous velocity for preset time.

| Command | Description |
|---|---|
| **MN** | Sets  Normal mode |
| **A5** | Sets acceleration to 5 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **D4ØØØ** | Sets distance to 4,000 encoder counts |
| **T1Ø** | Pauses motor movement 10 seconds |
| **G** | Executes the move (Go) |
| **T5** | Pauses the motor for 5 seconds after the move |
| **G** | Executes the move (Go) |

## TR—Wait For Trigger

- ❏ Command Type: Programming
- ❏ Syntax: <a>TRnnnnnn
- ❏ Range: n = Ø, 1, or X
- ❏ Attributes: Buffered, Savable in Sequence
- ❏ Valid Software Version: A
- ❏ Units: See Below
- ❏ Default Value: None
- ❏ See Also: IN

This command allows you to specify a trigger configuration to be matched before continuing execution of the move, where *nnnnn* corresponds to triggers 1, 2, 3, 4 and 5  respectively.  The possible values for *n* are as follows:

**n = 1** Wait for the trigger input to be high (opened)
**n = Ø** Wait for the trigger input to be low (grounded)
**n = X** Ignore the trigger input
The lowest numbered input will be the first trigger.  For example, if input 3, 4 and 5 are configured with the IN command as triggers, they will be trigger 1, 2 and 3 respectively.

| Command | Description |
|---|---|
| **IN3A** | Configure input 3 as  trigger input 1 |
| **IN5A** | Configure input 5 as trigger input 2 |
| **TR1Ø** | Wait for input 1 to be opened and input 2 to be grounded before going on to the next command |
| **A1Ø** | Sets acceleration to 10 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **D4ØØØ** | Sets distance to 4,000 encoder counts |
| **G** | Executes the move (Go) |

Motion will not occur until trigger conditions are true.

## U—Pause and Wait for Continue

❏ Command Type:  Programming
❏ Syntax:  <a>U
❏ Range:  N/A
❏ Attributes:  Immediate, Never Saved
❏ Valid Software Version:  A
❏ Units:  N/A
❏ Default Value:  N/A
❏ See Also:  C, PS

This command causes the controller to complete the move in progress, then wait until it receives a Continue (**C**) to resume processing.  Since the buffer is saved, the controller continues to execute the program (at the point where it was interrupted).  The controller continues processing when it receives the **C** command.  This command is typically used to stop a machine while it is unattended.

| Command | Description |
|---|---|
| **MN** | Sets move to Normal mode |
| **A5** | Sets acceleration to 5 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **LØ** | Loops indefinitely |
| **D46ØØ** | Sets distance to 4,600 encoder counts |
| **G** | Executes the move (G) |
| **T1Ø** | Waits 10 seconds after the move |
| **N** | Ends loop |
| **U** | Halts execution until the controller receives the Continue command **(C)** |

This command string pauses when the **U** command is entered. A **C** command resumes execution where it was paused. In this example, the loop stops at the end of a move, and resumes when the controller receives the **C** command. In reaction to the **T1Ø** command in the loop, there may be a 10 second delay before motion resumes after the **C** is executed, depending on when the **U** command is completed.

## V—Velocity

❑ Command Type: Motion
❑ Syntax: <a>Vn
❑ Range: n = 0.01 - 200.00
❑ Attributes: Buffered, Savable in Sequence

❑ Valid Software Version: A
❑ Units: revs/sec
❑ Default Value: 1
❑ See Also: A, D, G, GH, MR

The **V** command defines the maximum speed at which the motor will run when given the Go (**G**) command. The maximum encoder frequency the controller can accept is 960 kHz. In preset Mode Normal (**MN**), the maximum velocity may be limited when the resulting move profile is triangular. In Mode Continuous (**MC**), when a Go (**G**) command is issued the controller moves to the next command in the buffer.

Once you define the velocity, that value will be valid until you define another velocity, cycle DC power, or issue a **Z** (Reset) command.

*If the value specified for the **V** command is not valid, the controller ignores that value and defaults to the value specified in the last **V** command. If **V** is issued with only a device address (**1V**), the controller will respond with the current velocity value (*Vn).*

| Command | Description |
|---|---|
| **MC** | Sets move to continuous |
| **A5** | Sets acceleration to 5 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **G** | Go (begin motion) |

# XC—Sequence Checksum

❏ Command Type: Status
❏ Syntax: aXC
❏ Range: N/A
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: None
❏ See Also: XD, XE

**XC** computes the BBRAM checksum. After the unit is programmed, the response can be used for system error checking. The three-decimal response (ØØØ - 255) is followed by a [cr]. The response does not indicate the number of bytes programmed. This response is designed to be used for comparison. As long as the controller is not reprogrammed, the checksum response should always be the same.

| Command | Response |
|---|---|
| **1XC** | ***149** |

# XD—Sequence Definition

❏ Command Type: Programming
❏ Syntax: <a>XDn
❏ Range: n = 1 - 7
❏ Attributes: Buffered, Never Saved
❏ Valid Software Version: A
❏ Units: Sequences
❏ Default Value: None
❏ See Also: XE, XR, XRP, XT

This command begins sequence definition. All commands between the **XD** command and Sequence Termination (**XT**) command are defined as a sequence. The sequences will automatically be defined when **XT** is issued. If a sequence you are trying to define already exists, you must erase that sequence before defining it using the Erase Sequence (**XE**) command. A sequence cannot be longer than 255 characters. Immediate commands cannot be entered into a sequence. Sequences can only be permanently saved with the -M2 (BBRAM) option. Without the -M2 option sequences can be saved in operating RAM, and *will* be retained after a reset (Z) but not after a power cycle.

| Command | Description |
|---|---|
| **XE1** | Erases sequence #1 |
| **XD1** | Defines sequence #1 |
| **MN** | Sets to Normal mode |
| **A1Ø** | Sets acceleration to 10 revs/sec$^2$ |
| **V5** | Sets acceleration to 5 revs/sec |
| **D1ØØØØ** | Sets distance to 10,000 encoder counts |
| **G** | Executes the move (Go) |
| **XT** | Ends definition of Sequence #1 |
| **XR1** | Executes Sequence #1 |

## XE—Sequence Erase

- ❏ Command Type: Programming
- ❏ Syntax: <a>XEn
- ❏ Range: n = 1 - 7
- ❏ Attributes: Buffered, Never Saved
- ❏ Valid Software Version: A
- ❏ Units: Sequences
- ❏ Default Value: None
- ❏ See Also: XD, XR, XRP, XT

This command allows you to delete a sequence. The sequence that you specify (n) will be deleted when you issue the command. *Compumotor recommends that you delete a sequence before redefining it.*

| Command | Description |
|---|---|
| **XE1** | Deletes Sequence 1 |
| **XD1** | Defines Sequence 1 |

## XP—Set Power-up Sequence Mode

- ❏ Command Type: Set-Up
- ❏ Syntax: <a>XPn
- ❏ Range: n = 0 - 9
- ❏ Attributes: Buffered, Automatically Saved
- ❏ Valid Software Version: A
- ❏ Units: Sequences
- ❏ Default Value: 0
- ❏ See Also: IN, XQ, XSP, XSR

This command executes a single sequence or multiple sequences on power-up. If *n* = 1-7, the sequence whose value = *n* will be executed on power up. Control will then be passed to the RS-232C interface.

If *n* = 8, the sequence whose number appears on the sequence select inputs (configured with the **IN** command) will be executed on power-up. Control will then be passed to the RS-232C interface.

If *n* = 9, the sequence whose number appears on the Sequence Select inputs (configured with the **IN** command) will be executed on power-up. When the first sequence is finished in **XP9** mode, the controller will scan the Sequence Select inputs again and execute the next sequence. This cycle will continue until a Stop (**S**) or Kill (**K**) command is issued, a limit is encountered, or the unit is powered down. The possible settings for this command are as follows:

**n = Ø:** No sequence is executed on power-up
**n =1-7:** Sequence 1 - 7 is executed on power-up
**n = 8:** Sequence select inputs are read (single run) on power-up
**n = 9:** Sequence select inputs are read (continuous run) on power-up

In **XP9** mode, you can use the **XQ1** command to stop the controller from selecting the next sequence until all the sequence select inputs are first opened.

Sequences can only be permanently saved with the -M2 (BBRAM) option. Without the -M2 option sequences can be saved in operating RAM, and *will* be retained after a reset (Z) but not after a power cycle.

| **Command** | **Description** |
|---|---|
| **XP1** | Executes Sequence #1 on power-up |
| **XE1** | Erases Sequence #1 |
| **XD1** | Defines Sequence #1 |
| **LD3** | Disables CW & CCW limits |
| **A1Ø** | Sets acceleration to 10 revs/sec2 |
| **V5** | Sets velocity to 5 revs/sec |
| **D4ØØØ** | Sets distance to 4,000 encoder counts |
| **G** | Executes the move (Go) |
| **XT** | Ends definition of Sequence #1 |
| **Z** | Resets the controller |

The motor moves 4,000 encoder counts during power-up (with -M2 option only) or reset (Z).

## XQ—Sequence Interrupted Run Mode

❏ Command Type: Set-Up
❏ Syntax: <a>XQn
❏ Range: n = Ø, 1
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: Sequences
❏ Default Value: Ø
❏ See Also: XP

n = 1: Interrupted Run mode is set (on)
n = Ø: Interrupted Run mode is reset (off)

This command can be used only when the controller is stand-alone power-up sequencing in **XP9** mode. In **XP9** mode, if **XQ1** is executed, the controller will not accept a sequence select input until all sequence select inputs are OFF (closed). After all lines have simultaneously been brought to a low state (OFF), the controller will then read the sequence select lines and execute the sequence whose number appears there. This paused mode will continue until an **XQØ** command is executed. You may use **S** or **K** command to stop sequence execution. **XQ1** must be the first command entered in the sequence.

| Command | Description |
|---|---|
| **XE1** | Erases sequence #1 |
| **XD1** | Defines sequence #1 |
| **XQ1** | Turns Interrupted Run mode on |
| **LD3** | Disables CW & CCW limits |
| **XT** | Ends Sequence #1 |
| **XP9** | Sets power-up sequences as sequence select inputs |
| **Z** | Resets the controller to start sequence scanning |

If you execute Sequence #1 during power up by setting the sequence select inputs (configured with the **IN** command) inputs properly, Interrupted Run mode will be set.

# XR—Run a Sequence

- ❏ Command Type: Programming
- ❏ Syntax: <a>XRn
- ❏ Range: n = 1 - 7
- ❏ Attributes: Buffered, Savable in Sequence
- ❏ Valid Software Version: A
- ❏ Units: Sequence
- ❏ Default Value: 0
- ❏ See Also: XD, XE, XRP, XT

This command loads a predefined sequence into the command buffer (clears the buffer first) and executes these commands as a normal set of commands. **XR** automatically recalls the sequence from BBRAM.

**XR** can be used within one sequence to start execution of another sequence; however, all commands in the first sequence following **XR** will be ignored (in this respect an XR acts like a GOTO not a GOSUB). An **XR** command placed within a loop will be ignored.

Sequences can only be permanently saved with the -M2 (BBRAM) option. Without the -M2 option sequences can be saved in operating RAM, and *will* be retained after a reset (Z) but not after a power cycle.

| Command | Description |
|---|---|
| **XE1** | Erases sequence #1 |
| **XD1** | Defines sequence #1 |
| **A1Ø** | Sets acceleration to 10 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **D1ØØØØ** | Sets distance to 10,000 encoder counts |
| **G** | Executes the move (Go) |
| **XT** | Ends Sequence #1 definition |
| **XR1** | Executes Sequence #1 |

Sequence #1 is defined (**XD1**) and executed (**XR1**).

# XRP—Sequence Run With Pause

| | |
|---|---|
| ❏ Command Type: Programming | ❏ Valid Software Version: A |
| ❏ Syntax: <a>XRPn | ❏ Units: Sequence |
| ❏ Range: n = 1 - 7 | ❏ Default Value: 0 |
| ❏ Attributes: Buffered, Savable in Sequence | ❏ See Also: XD, XE, XR, XT |

This command is identical to the Sequence Run (**XR**) command, except that it automatically generates a pause condition. You must clear this condition with the Continue (**C**) command before the controller executes the command buffer. The pause condition is invoked only if the sequence is valid. This allows you to execute a sequence without the delay of buffering that sequence.

| Command | Description |
|---|---|
| **XE5** | Erases Sequence #5 |
| **XD5** | Defines Sequence #5 |
| **A1Ø** | Sets acceleration to 10 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **D1ØØØØ** | Sets distance to 10,000 encoder counts |
| **G** | Executes the move (Go) |
| **XT** | Ends definition of Sequence #5 |
| **XRP5** | Runs Sequence #5 with a pause |
| **C** | Controller executes Sequence #5 |

Upon issuing XRP5, Sequence #5 is entered in the command buffer, but is not executed. Issue a C command to execute Sequence #5.

# XSD—Sequence Status Definition

❏ Command Type: Programming
❏ Syntax: aXSD
❏ Range: N/A
❏ Attributes: Buffered, Savable in Sequence
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: XD, XE, XT
❏ Response to aXSD is *n

This command reports the status of the previous sequence definition (**XD**...**XT**). The response is 0 - 2. The valid values and descriptions of possible responses are shown below:

n = Ø: Download O.K.
n = 1: A sequence already exists with the number you have specified.
n = 2: Out of memory. The sequence buffer is full.

**XSD** verifies that the last sequence definition was successful.

| Command | Response |
|---|---|
| **1XSD** | ***1** (A sequence already exists as sequence 1) |

# XSP—Sequence Status Power-up

❏ Command Type: Status
❏ Syntax: aXSP
❏ Range: N/A
❏ Attributes: Buffered, Never Saved
❏ Valid Software Version: A
❏ Units: N/A
❏ Default Value: N/A
❏ See Also: XP, XQ, XSR
❏ Response to aXSP is *n

The Sequence Status Power-up (**XSP**) determines which, if any, sequence will be executed on power-up. After setting a power-up sequence using the Sequence Power-up (**XP**) command, you can check to make sure that proper sequence will be executed on power-up with **XSP**. The command reports which sequence the system will execute during power-up. The range of sequences is Ø - 9.

| Command | Description |
|---|---|
| **1XSP** | ***3** (Indicates that sequence #3. If it exists, will be executed upon power-up or reset.) |

# XSR—Sequence Status Run

❑ Command Type: Status
❑ Syntax: aXSR
❑ Range: N/A
❑ Attributes: Immediate, Never Saved
❑ Valid Software Version: A
❑ Units: N/A
❑ Default Value: N/A
❑ See Also: XR, XRP
❑ Response to aXSR is *n

This command allows you to check whether or not the last sequence issued was executed successfully without hitting limits, Stop (**S**), or Kill (**K**). The valid values and responses are shown below.

* **Ø** = Last sequence was successful
* **1** = In a loop
* **2** = Invalid sequence
* **3** = Erased
* **4** = Bad checksum
* **5** = Running
* **6** = Killed, stopped

| **Command** | **Response** |
|---|---|
| **1XSR** | ***Ø** (Sequence ran OK) |

# XSS—Sequence Status

❑ Command Type: Status
❑ Syntax: aXSSn
❑ Range: n = 1 - 7
❑ Attributes: Buffered, Never Saved
❑ Valid Software Version: A
❑ Units: Sequences
❑ Default Value: None
❑ See Also: XD, XE, XT
❑ Response to aXSSn is *x

**XSS** reports whether the sequence specified by n (representing one of the sequences 1 - 7) is empty, has bad checksum, or is OK.

Ø = Empty
1 = Bad Checksum
3 = O.K.

**XSS** verifies the existence of sequences and if that portion of memory has been corrupted.

| **Command** | **Response** |
|---|---|
| **1XSS1** | ***Ø** (Sequence #1 of device 1 is not defined.) |

# XT—Sequence Termination

- ❏ Command Type: Programming
- ❏ Syntax: <a>XT
- ❏ Range: N/A
- ❏ Attributes: Buffered, Never Saved
- ❏ Valid Software Version: A
- ❏ Units: N/A
- ❏ Default Value: N/A
- ❏ See Also: XD, XE, XR, XRP

**XT** is a sequence terminator. This command flags the end of the sequence currently being defined. Sequence definition is not complete until this command is issued. Properly defined sequences are saved into BBRAM (*-M2 Option Only*) automatically by issuing this command.

**NOTE**: In your communication program, use sufficient time delays after downloading a sequence before you send more commands to the OEM770X. In particular, after sending the **XT** command, wait at least 12.5 msec before sending a **Z** command.

| Command | Description |
|---|---|
| **XE1** | Erases Sequence #1 |
| **XD1** | Defines Sequence #1 |
| **MN** | Sets to Normal mode |
| **A1Ø** | Sets acceleration to 10 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **D4ØØØ** | Sets distance to 4,000 encoder counts |
| **G** | Executes the move (Go) |
| **XT** | Ends sequence definition |

# XU—Upload Sequence

- ❏ Command Type: Status
- ❏ Syntax: aXUn
- ❏ Range: n = 1 - 7
- ❏ Attributes: Buffered, Never Saved
- ❏ Valid Software Version: A
- ❏ Units: Sequences
- ❏ Default Value: N/A
- ❏ See Also: F, XD, XE, XT
- ❏ Response to aXUn is contents of sequence n

This command sends the contents of sequence *n* to the host computer via the RS-232C interface, terminated by a carriage return [cr]. The contents of that sequence will appear on the computer monitor. All command delimiters in the sequence will be shown as spaces (2ØH). Any device identifiers that were included in the original sequence will also be eliminated (they are not stored in the sequence).

When using a daisy-chain, **XU** must be used cautiously as the contents of the sequence will go to all controllers in the loop between the controller that is uploading and the host. The **F** command may be used to turn off communication on units you are not uploading from.

| Command | Description |
|---|---|
| **2F** | Turns off communication to unit #2 |
| **3F** | Turns off communication to unit #3 |
| **1XU1** | Uploads sequence #1 from unit #1 |

## Y—Stop Loop

❏ Command Type: Programming

❏ Syntax: <a>Y

❏ Range: N/A

❏ Attributes: Immediate, Never Saved

❏ Valid Software Version: A

❏ Units: N/A

❏ Default Value: N/A

❏ See Also: L, N

The Stop Loop (**Y**) command takes you out of a loop when the loop completes its current pass. This command does not halt processing of the commands in the loop until the controller processes the last command of the current loop. At that time, the controller executes the command that follows the End Loop (**N**) command. You cannot restart the command loop unless you enter the entire command structure, including the Loop (**L**) and End Loop (**N**) commands.

| Command | Description |
|---|---|
| **L** | Loops indefinitely |
| **A1Ø** | Sets acceleration to 10 revs/sec$^2$ |
| **V5** | Sets velocity to 5 revs/sec |
| **D4ØØØ** | Sets distance to 4,000 encoder counts |
| **T2** | Waits 2 seconds |
| **G** | Executes the move (Go) |
| **N** | Ends loop |
| **Y** | Stops loop |

The loop requires the motor to move 4,000 encoder counts CW and then wait for 2 seconds. The loop terminates at the end of the loop cycle it is executing when it receives the **Y** command.

# Z—Reset

- ❏ Command Type: Programming
- ❏ Syntax: <a>Z
- ❏ Range: N/A
- ❏ Attributes: Immediate, Never Saved
- ❏ Valid Software Version: A
- ❏ Units: N/A
- ❏ Default Value: N/A
- ❏ See Also: K, S

The Reset (**Z**) command is equivalent to cycling DC power to the controller. This command returns all internal settings to their power-up values. It clears the command buffer.

When you use the **Z** command, the controller is busy for 1,000 ms and ignores all commands. This command sets all position counters to zero and returns all values except values stored in battery backed RAM to factory defaults.

| Command | Description |
|---|---|
| **1Z** | Resets controller with address 1 |

# #—Address Numbering

- ❏ Command Type: Set-up
- ❏ Syntax: <a>#n
- ❏ Range: n=1-255
- ❏ Attributes: Immediate, Automatically Saved
- ❏ Valid Software Version: A
- ❏ Units: Address number
- ❏ Default Value: 1
- ❏ See Also: E,F

This command sets the individual unit address for each OEM770X, allowing addresses up to 255. Upon receipt of the command, the OEM770X will assign itself the address in the command and will pass on the daisy chain the address *plus one*, thus enabling automatic addressing of all units on the daisy chain. The address may also be set individually if preferred.

**#1** - Automatic addressing of all units
Response - #(number of units plus one)

If the unit addresses exceed 255 , then the response will be #?. A <CR> or LF must be used with this command.

| Command | Description |
|---|---|
| **#1** | #5 ( for a daisy chain of 4 units, the units will assign themselves addresses 1 through 4 and return #5 as confirmation). |

# %—Reset Communication

❑ Command Type: Status
❑ Syntax: %
❑ Range: N/A
❑ Attributes: Immediate, Never Saved

❑ Valid Software Version: E
❑ Units: N/A
❑ Default Value: N/A
❑ See Also: E, F, SSE

When the OEM770X detects a communication error, it ignores all *external* commands and echoes an **&** for each byte it receives from the host. You can use this command to re-establish communication, and to identify the cause of the communication error.

In a daisy-chained environment, units located downstream from the unit detecting a communication error will also disable external command processing. Units upstream in a daisy chain are not affected.

(Note: Error detection will only occur if **SSE1** is enabled. Detection of a communication error has no effect on internal command processing, or sequence execution. A communication error will not stop motion.)

Possible responses are:

| Character | Definition |
|---|---|
| * | No errors |
| ***0** | Unit upstream (daisy-chained) |
| ***1** | Overrun, data received too fast |
| ***2** | Framing error |

| Command | Response |
|---|---|
| **%** | ***2** (Either host or controller has lost synchronization.) |

(Note: For daisy chained environments, the response values are in reverse order.)

| | |
|---|---|
| **%** | ***0*0*0*1***** (First 5 units report no error, 6th unit detected an overrun error, and the last 3 units turned communication off because of unit 6.) |

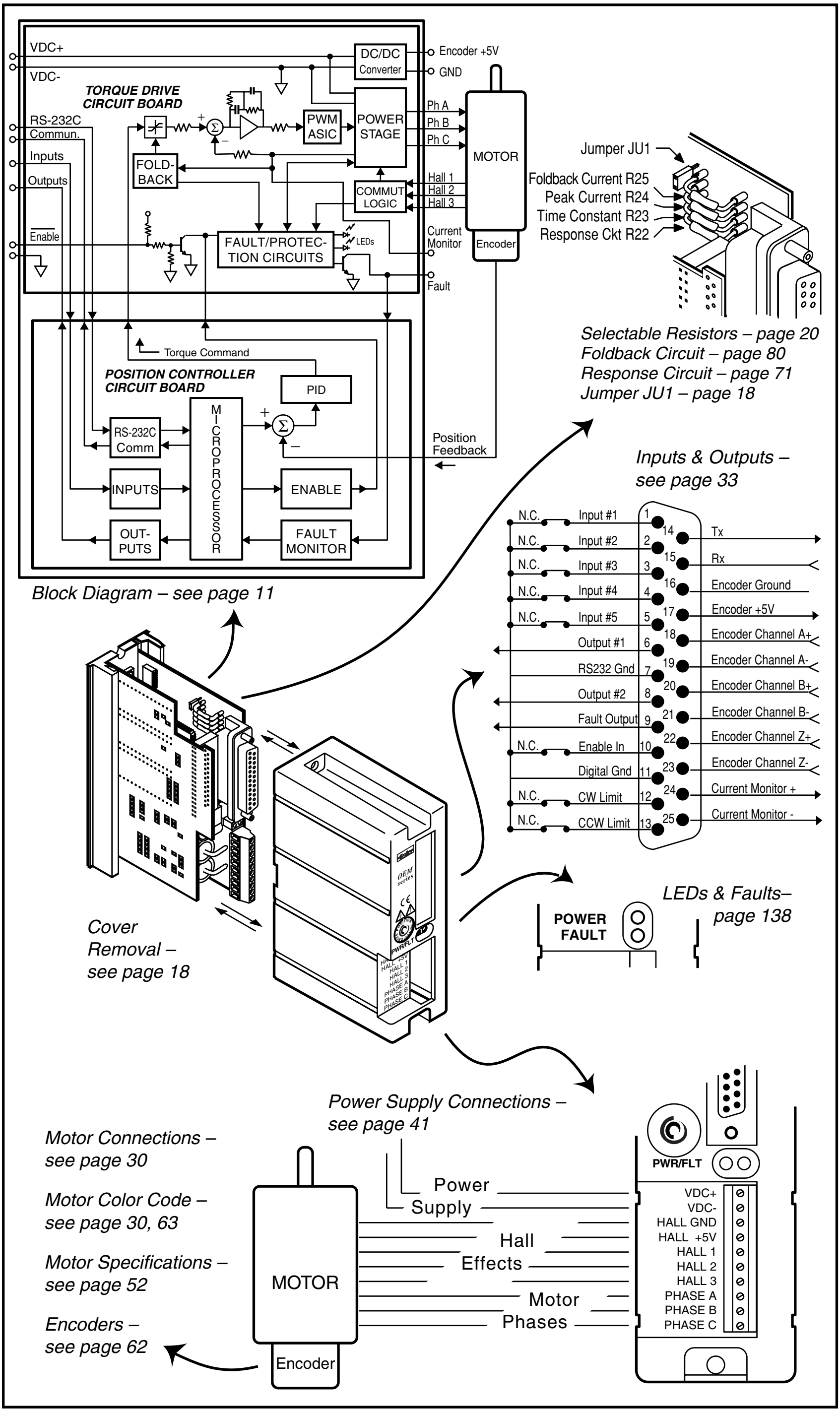
# Summary of Commands

A—Acceleration
B—Buffer Status
BCDG—Buffered Configure Derivative Gain
BCIG—Buffered Configure Integral Gain
BCIL—Buffered Configure Integral Limit
BCPE—Buffered Configure Position Error
BCPG—Buffered Configure Proportional Gain
BCTG—Buffered Configure Derivative Sampling Period
BS—Buffer Size Status
C—Continue
CDG—Configure Derivative Gain
CEW—Configure In Position Error Window
CIG—Configure Integral Gain
CIL—Configure Integral Limit
CIT—Configure In Position Time
CPE—Configure Maximum Position Error
CPG—Configure Proportional Gain
CR—Carriage Return
CTG—Configure Filter Time Constant
D—Distance
DPA—Display Position Actual
DPE—Display Position Error
DVA—Display Velocity Actual
E—Enable Communications
ER—Encoder Resolution
F—Disable Communications
G—Go
GH—Go Home
^H—Delete
H—Set Direction
IN—Set Input Functions
IS —Input Status
K—Kill
L—Loop
LD—Limit Disable
LF—Line Feed
MC—Mode Continuous
MN—Mode Normal
MPA—Mode Position Absolute
MPI—Mode Position Incremental
N—End of Loop
O—Output
OFF—Servo Disable
ON—Servo Enable
OS—Report Homing Function Set-Ups
OSA—Define Active State of End-of-Travel Limits
OSB—Back Up To Home
OSC—Define Active State of Home Switch
OSD—Enable Encoder Z Channel for Home
OSH—Reference Edge of Home Switch
PR—Position Report
PS—Pause
PX—Report Absolute Encoder Position
PZ—Set Absolute Counter to Zero
“—Quote
Q1—Enter Velocity Profiling Mode
QØ—Exit Velocity Profiling Mode
R—Request Controller Status
RA—Limit Switch Status Report
RB—Loop, Pause, Shutdown, Trigger Status Report
RC—Homing Status Report
RFS—Return Servo Gains to Factory Settings
RM—Rate Multiplier in Velocity Streaming Mode
RSE—Report Servo Erros
RV—Revision Level
S—Stop
SN—Scan
SS—Software Switch Function Status
SSA—RS-232C Echo Control
SSC—Output #1 on In Position
SSE—Enable/Disable Communication Error Checking
SSG—Clear/Save the Command Buffer on Limit
SSH—Clear/Save Command Buffer on Stop
ST—Shutdown
T—Time Delay
TR—Wait For Trigger
U—Pause and Wait for Continue
V—Velocity
XC—Sequence Checksum
XD—Sequence Definition
XE—Sequence Erase
XP—Set Power-up Sequence Mode
XQ—Sequence Interrupted Run Mode
XR—Run a Sequence
XRP—Sequence Run With Pause
XSD—Sequence Status Definition
XSP—Sequence Status Power-up
XSR—Sequence Status Run
XSS—Sequence Status
XT—Sequence Termination
XU—Upload Sequence
Y—Stop Loop
Z—Reset
#—Address Numbering
%—Reset Communication

# INDEX

# Summary of Commands

A—Acceleration
B—Buffer Status
BCDG—Buffered Configure Derivative Gain
BCIG—Buffered Configure Integral Gain
BCIL—Buffered Configure Integral Limit
BCPE—Buffered Configure Position Error
BCPG—Buffered Configure Proportional Gain
BCTG—Buffered Configure Derivative Sampling Period
BS—Buffer Size Status
C—Continue
CDG—Configure Derivative Gain
CEW—Configure In Position Error Window
CIG—Configure Integral Gain
CIL—Configure Integral Limit
CIT—Configure In Position Time
CPE—Configure Maximum Position Error
CPG—Configure Proportional Gain
CR—Carriage Return
CTG—Configure Filter Time Constant
D—Distance
DPA—Display Position Actual
DPE—Display Position Error
DVA—Display Velocity Actual
E—Enable Communications
ER—Encoder Resolution
F—Disable Communications
G—Go
GH—Go Home
^H—Delete
H—Set Direction
IN—Set Input Functions
IS —Input Status
K—Kill
L—Loop
LD—Limit Disable
LF—Line Feed
MC—Mode Continuous
MN—Mode Normal
MPA—Mode Position Absolute
MPI—Mode Position Incremental
N—End of Loop
O—Output
OFF—Servo Disable
ON—Servo Enable
OS—Report Homing Function Set-Ups
OSA—Define Active State of End-of-Travel Limits
OSB—Back Up To Home
OSC—Define Active State of Home Switch
OSD—Enable Encoder Z Channel for Home
OSH—Reference Edge of Home Switch
PR—Position Report
PS—Pause
PX—Report Absolute Encoder Position
PZ—Set Absolute Counter to Zero
“—Quote
Q1—Enter Velocity Profiling Mode
QØ—Exit Velocity Profiling Mode
R—Request Controller Status
RA—Limit Switch Status Report
RB—Loop, Pause, Shutdown, Trigger Status Report
RC—Homing Status Report
RFS—Return Servo Gains to Factory Settings
RM—Rate Multiplier in Velocity Streaming Mode
RSE—Report Servo Erros
RV—Revision Level
S—Stop
SN—Scan
SS—Software Switch Function Status
SSA—RS-232C Echo Control
SSC—Output #1 on In Position
SSE—Enable/Disable Communication Error Checking
SSG—Clear/Save the Command Buffer on Limit
SSH—Clear/Save Command Buffer on Stop
ST—Shutdown
T—Time Delay
TR—Wait For Trigger
U—Pause and Wait for Continue
V—Velocity
XC—Sequence Checksum
XD—Sequence Definition
XE—Sequence Erase
XP—Set Power-up Sequence Mode
XQ—Sequence Interrupted Run Mode
XR—Run a Sequence
XRP—Sequence Run With Pause
XSD—Sequence Status Definition
XSP—Sequence Status Power-up
XSR—Sequence Status Run
XSS—Sequence Status
XT—Sequence Termination
XU—Upload Sequence
Y—Stop Loop
Z—Reset
#—Address Numbering
%—Reset Communication



Block Diagram – see page 11

Selectable Resistors – page 20
Foldback Circuit – page 80
Response Circuit – page 71
Jumper JU1 – page 18

Inputs & Outputs – see page 33

LEDs & Faults– page 138

Cover Removal – see page 18

Power Supply Connections – see page 41

Motor Connections – see page 30

Motor Color Code – see page 30, 63

Motor Specifications – see page 52

Encoders – see page 62