TABLE OF CONTENTS

How To Use This User Guide

This user guide is designed to help you install, develop, and maintain your system. Each chapter begins with a list of specific objectives that should be met after you have read the chapter. This section is intended to help you find and use the information in this user guide.

Assumptions

This user guide assumes that you have the skills or fundamental understanding of the following information.

❏ Basic electronics concepts (voltage, switches, current, etc.)

❏ Basic motion control concepts (torque, velocity, distance, force, etc.)

Contents of This Manual

This user guide contains the following information.

Chapter 1: Introduction

This chapter provides a description of the product and a brief account of its specific features.

Chapter 2: Getting Started

This chapter contains a detailed list of items you should have received with your SX shipment. It will help you to become familiar with the system and ensure that each component functions properly.

Chapter 3: Installation

This chapter provides instructions for you to properly mount the system and make all electrical connections. Upon completion of this chapter, your system should be completely installed and ready to perform basic operations. Tuning considerations and procedures are also provided.

Chapter 4: Application Design

This chapter will help you customize the system to meet your application's needs. Important application considerations are discussed. Sample applications are provided.

Chapter 5: SXF Follower

This chapter explains the SXF Following function and the SXF's capability to support absolute and incremental encoders.

Chapter 6: Hardware Reference

This chapter contains information on system specifications (electrical, dimensions, and performance). It may be used as a quick-reference tool for proper switch settings and connections.

Chapter 7: Troubleshooting

This chapter contains information on identifying and resolving system problems.

Installation Process Overview

To ensure trouble-free operation, pay special attention to the environment in which the SX equipment will operate, the layout and mounting, and the wiring and grounding practices used. These recommendations are intended to help you easily and safely integrate SX equipment into your manufacturing facility. Industrial environments often contain conditions that may adversely affect solid-state equipment. Electrical noise or atmospheric contamination, may also affect the SX System.

Developing Your Application

Before you attempt to develop and implement your application, there are several issues that you should consider and address.

① Recognize and clarify the requirements of your application. Clearly define what you expect the system to do.

② Assess your resources and limitations. This will help you find the most efficient and effective means of developing and implementing your application (hardware and software).

③ Follow the guidelines and instructions outlined in this user guide. Do not skip any steps or procedures. Proper installation and implementation can only be ensured if all procedures are completed in the proper sequence.

Installation Preparation

Before you attempt to install this product, you should complete the following steps:

① Review this entire user guide. Become familiar with the user guide's contents so that you can quickly find the information you need.

② Develop a basic understanding of all system components, their functions, and interrelationships.

③ Complete the basic system configuration and wiring instructions (in a simulated environment, not a permanent installation) provided in *Chapter 2, Getting Started.*

④ Perform as many basic functions as you can with the preliminary configuration. You can only perform this task if you have reviewed the entire user guide. You should try to simulate the task(s) that you expect to perform when you permanently install your application (however, do not attach a load at this time). This will give you a realistic preview of what to expect from the complete configuration.

⑤ After you have tested all of the system's functions and used or become familiar with tll of the system's features, carefully read *Chapter 3, Installation.*

⑥ After you have read Chapter 3 and clearly understand what must be done to properly install the system, you should begin the installation process. Do not deviate from the sequence or installation methods provided.

⑦ Before you begin to customize your system, check all of the systems functions and features to ensure that you have completed the installation process correctly.

The successful completion of these steps will prevent subsequent performance problems and allow you to isolate and resolve any potential system difficulties before they affect your system's operation.

Conventions

To help you understand and use this user guide effectively, the conventions used throughout this user guide are explained in this section.

Commands

All commands that you are instructed to enter are shown in capital letters. The symbol >, is the SX command prompt. The command is displayed in boldface. A delimiter (space or carriage return) is required after each command. A description is provided next to each command example.

Command	Description
>MR	Sets motor resolution to 25,000 steps/rev

The system ignores command syntax that is not within the valid range for a specific command. A **?** prompt will be returned by the drive when the last command entered was not understood, or a parameter limit was exceeded.

Motors

S Series and SX Series motors are one in the same (interchangeable terms).

Warnings & Cautions

Warning and caution notes alert you to possible dangers that may occur if you do not follow instructions correctly. Situations that may cause bodily injury are present as warnings. Situations that may cause system damage are presented as cautions. These notes will appear in bold face and the word warning or caution will be centered and in all capital letters. Refer to the examples 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.

CAUTION

System damage will occur if you power up the system improperly.

Related Publications

❏ Current Parker Compumotor Motion Control Catalog

❏ SX Indexer/Drive Software Reference Guide

CHAPTER ①

Introduction

Chapter Objective

The information in this chapter will enable you to:

- ❑ Understand the product's basic functions and features

Product Description

The SX is a bipolar, recirculating, microstepping drive with built-in indexing capabilities. It is designed to drive two-phase permanent magnet hybrid step motors. The drive uses MOSFET technology to give high performance in a small package while providing short-circuit protection, brownout protection, over-temperature protection, and a built-in power supply. The built in Indexer is capable of storing 99 multiple move motion programs in battery backed RAM memory. Any of the programs can be selected in a variety of ways including BCD inputs, programmable controllers or a computer via RS-232C Interface.

Features

The SX also provides the following features:

- ❑ 16 selectable motor resolutions are available (200 - 50,800 steps/rev)
- ❑ Uses low-inductance motors for improved high-speed performance (23, 34, 42 frame size motors available with torques from 65 - 1,900 oz-in)
- ❑ Microprocessor controlled microstepping provides smooth operation over a wide range of speeds
- ❑ Closed-loop positioning interfaces to incremental or absolute encoder standards
- ❑ One registration input that is given the highest priority (position latched within 50µs)
- ❑ A complex motion profiling system that allows you to:
  - ❍ Change velocity based on distance without stopping
  - ❍ Change distance, or turn on outputs on-the-fly
- ❑ High-level programming commands such as:
  - ❍ **`IF/THEN/ELSE/WHILE`**
  - ❍ **`REPEAT/UNTIL`**
  - ❍ **`GOTO AND GOSUB`**
- ❑ Complex evaluations such as checking input levels, error conditions, boolean evaluations, and variable comparisons for basic programming branching decisions can be made
- ❑ An output can be configured to provide pulse and direction to second axis to control velocity and distance
- ❑ PLC functionality and interfacing capability using the eight inputs and four outputs
- ❑ Full short circuit protection for phase-to-phase and phase-to-ground short circuits
- ❑ Overtemperature and undervoltage protection
- ❑ Three state current control for reduced motor/drive heating
- ❑ LED status indicators: power, undervoltage, overtemperature, motor fault, and Indexer monitor fault

❑ Motor connector interlock to prevent connector damage

❑ A fault output to signal other equipment if a fault occurs

❑ 90VAC - 132VAC, 50/60Hz power input

❑ Operates linear motor forcers

Following Option (SXF)

The SXF option can perform velocity following and distance following moves at a *following ratio* of a master velocity. The SXF can follow from incremental or absolute encoders.

You can program the SXF for following applications with its command language and report back/ verification feature. You can enter following ratios via thumbwheels and change them on-the-fly.

You can perform preset moves at a specified velocity ratio. You can perform registration moves while in the Following mode. Registration moves can either follow at a ratio of the master velocity or be executed in the standard motion modes. The SXF can jog the motor in Following mode to help set up a system.

You can use the SXF's special Synchronization mode to compensate for system errors (e.g., stretching in a web processing system).

Following Option Features

The Model SXF provides these additional features:

❑ Controls a speed based on a ratio of a primary axis speed

❑ Makes preset moves at a velocity ratio of a primary axis

❑ Synchronizes speed or position to a primary axis based on registration marks on material

❑ Changes following ratio and other functions based on the encoder position of a primary axis (*Cam Following*)

CHAPTER ②

Getting Started

Chapter Objectives

The information in this chapter will enable you to:

- ❏ Verify that each component of your system has been delivered safely
- ❏ Become familiar with the system components and their interrelationships
- ❏ Ensure that each component functions properly by bench testing

What You Should Have

Inspect the SX upon receipt for obvious damage to its shipping container. Report any such damage to the shipping company. Parker Compumotor cannot be held responsible for damage incurred in shipment. The following items should be present and in good condition.

Description	Part Number
S Series Fan Kit	SFK***
Power Cable (SX 8 has 2 cables)	44-000054-01
SX or SXF Indexer/Drive	SX6/SXF6 or SX8/SXF8
SX Indexer/Drive User Guide	88-011850-01
SX Software Reference Guide	88-011871-01
Motor	Variety of sizes available**
Mounting Screws (2)	51-006037-01
Mounting Brackets (2)	53-006007-01

*The SX8 Drive includes a fan
**Refer to the following tables for specific motor sizes
***Optional Equipment

High and Low Power Drives

You should verify which type of SX you have before proceeding with this chapter. The high-power version of the drive (**SX8**) provides bipolar 0 - 8 amps/phase (up to 1,900 oz-in). The low-power version of the drive (**SX6**) provides bipolar 0 - 6 amps/phase (up to 400 oz-in). You can determine which drive you have by checking the label on the side of the drive. The label identifies the unit as **`SX8 DRIVE`** (SX106) or **`SX6 DRIVE`** (SX57 & SX83). You must be aware of the drive's type to set the motor current correctly (using DIP switches). There are different DIP switch settings for the two drive types.

Quick Test

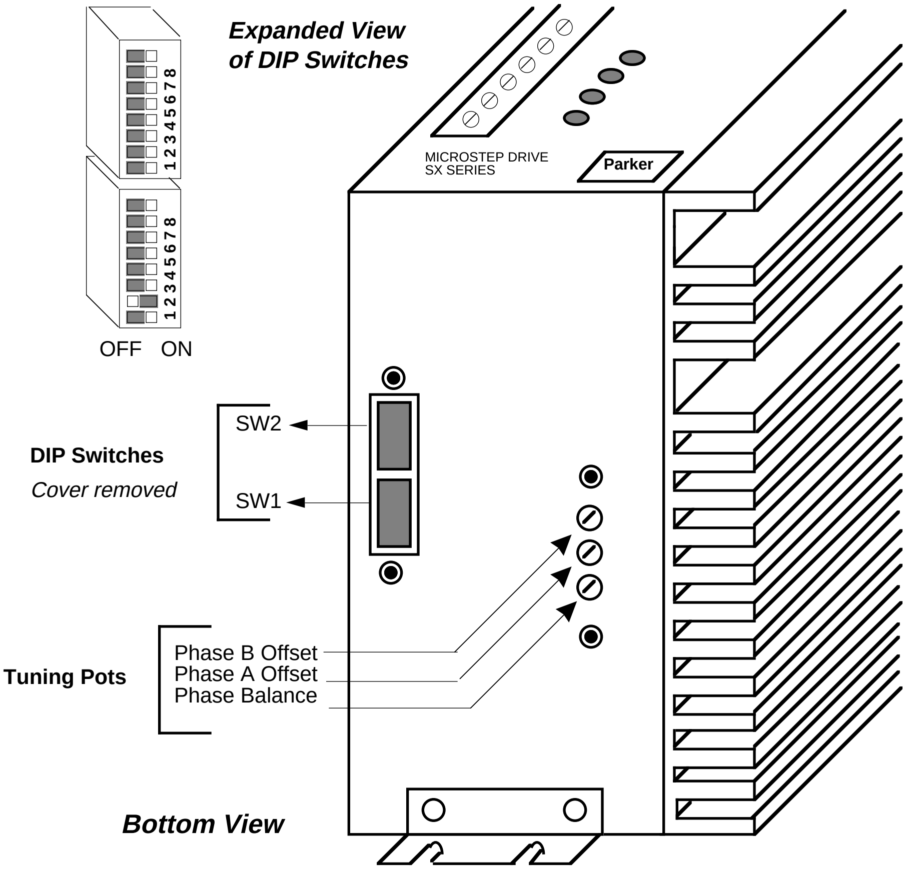
This section will show you how to set the SX's DIP switches and wire the unit to quickly ensure that your system is operating properly. Detailed installation instructions are provided in *Chapter 3, Installation*. You will need the following tools to complete these steps:

❑ A phillips-head screw driver (to move mounting brackets)

❑ A small flat screw driver (to adjust DIP switches and make connections)

CAUTION

Never adjust the DIP switches with a pencil. Lead from the pencil may contaminate the SX.



DIP Switch & Tuning Pot Locations

WARNING

Never attempt to change DIP switch settings while power is being applied to the unit.

① Set DIPSwitches

The SX has two sets of DIP switches. Each set of DIP switches has eight individual switches. The first set of switches will be referred to as **SW1** and the second set as **SW2**. The individual switch will be preceded by the **#** symbol. Hence, the third switch on **SW1** will be referred to as **SW1-#3**, while the third switch on **SW2** will be referred to as **SW2-#3.**

The first thing that you must do is set the motor current on the SX to match the motor that you are using. Use the directions below to set the DIP switches for your motor. (*Motor Drive combinations are configured at the factory for the proper current settings.*)

① Be sure that power is not applied to the unit.

② Remove the panel that covers the DIP switches. (Refer to earlier figure.)

③ Set the motor current for your Compumotor motor using the following tables. **SW1-#1** thru **SW1-#6** *control motor current* (refer to the previous figure for the location of **SW1**). Make the required adjustments to match the drive and motor types that you are using.

The following current tables are for S/SX Motors only.

Motor Size	Current	SW1-#1	SW1-#2	SW1-#3	SW1-#4	SW1-#5	SW1-#6
S57-51**S**	1.18	off	off	on	on	off	off
S57-51**P**	2.28	off	on	on	off	off	off
S57-83**S**	1.52	off	on	off	off	off	off
S57-83**P**	3.09	on	off	off	off	off	off
S57-102**S**	1.71	off	on	off	off	on	off
S57-102**P**	3.47	on	off	off	on	off	off
S83-62**S**	2.19	off	on	off	on	on	on
S83-62**P**	4.42	on	off	on	on	on	off
S83-93**S**	2.85	off	on	on	on	on	off
S83-93**P**	5.62	on	on	on	off	on	on
S83-135**S**	3.47	on	off	off	on	off	off
S83-135**P**	6.00	on	on	on	on	on	on

S: Series Configuration P: Parallel Configuration

Setting SX6 Motor Current (Compumotor Motors)

Motor Size	Current	SW1-#1	SW1-#2	SW1-#3	SW1-#4	SW1-#5	SW1-#6
S106-178**S**	6.02	on	off	on	on	on	on
S106-178**P**	8.0	on	on	on	on	on	on
S106-205**S**	3.55	off	on	on	on	off	off
S106-205**P**	6.99	on	on	off	on	on	on
S106-250**S**	6.23	on	on	off	off	off	on
S106-250**P**	8.0	on	on	on	on	on	on

S: Series Configuration P: Parallel Configuration

Setting SX8 Motor Current (Compumotor Motors)

The previous tables show motor current settings for series and parallel motor configurations. Refer to *Chapter 6, Hardware Reference* for specific motor configuration instructions. ***Compumotor ships all SX systems in series configurations***.

If you use a *non-Compumotor motor*, special precautions and instructions are required. Read the instructions in *Chapter 6, Hardware Reference* for non-Compumotor motors thoroughly before attempting to set the motor current or wire your motor.

④ Turn **SW2-#8 on** to enable the Automatic Test function. This function rotates the motor in an Alternating mode approximately 6 revolutions at 1 rps (12 inches at 2 ips if an L20 linear motor is used) as soon as power is applied.

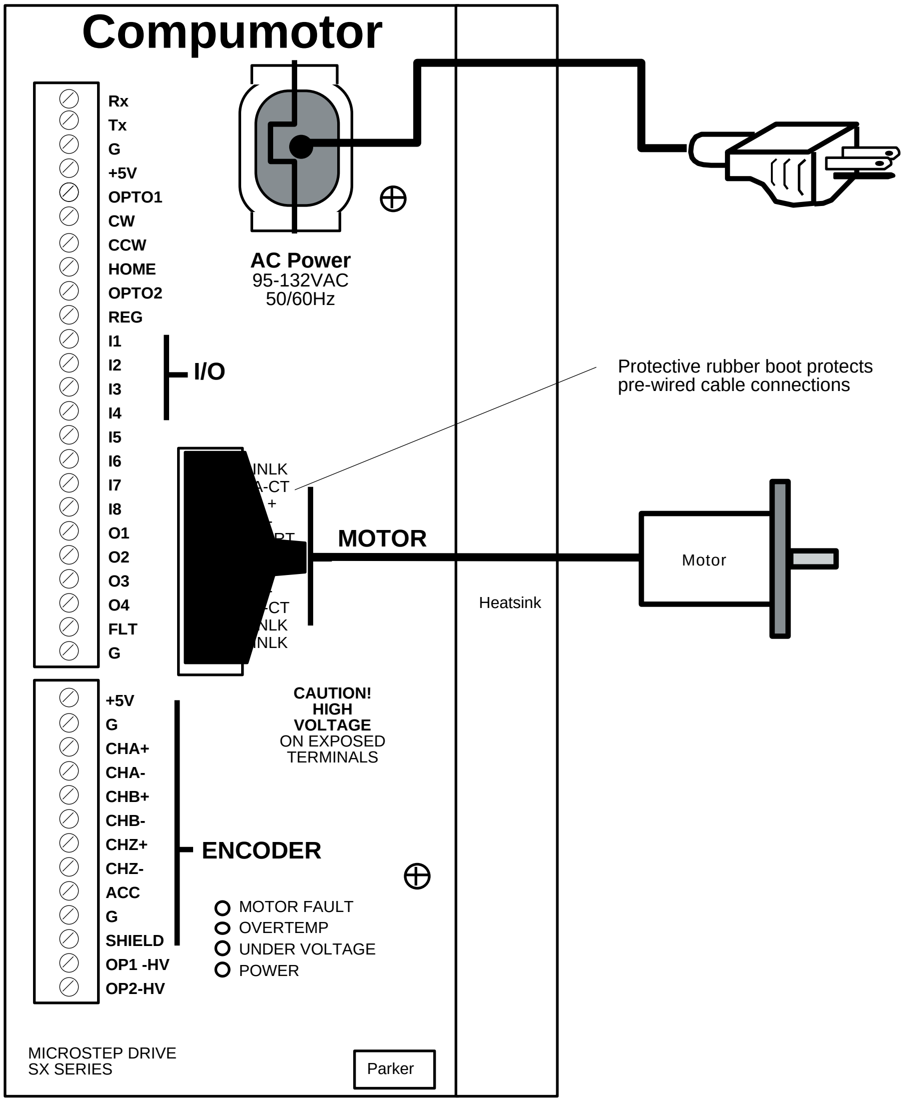
⑤ After you have properly set motor current (**SW1-#1** thru **SW1-#6**) and the Automatic Test function (**SW2-#8**), screw the plate that covers the DIP switches back onto the drive. **Do not change any other DIP switch settings**.

② Attach the Motor

WARNING

POWER MUST BE OFF before cabling the drive. Lethal voltages are present inside the drive and on its screw terminals.

The SX motor is pre-wired in series. Plug the pre-wired motor cable into the motor connector on the drive (refer to the following figure). If you use a non-Compumotor motor, refer to *Chapter 6, Hardware Reference* for instructions on wiring the motor to the drive. **Do not connect the motor to the load at this time.**



Test Configuration (S6 Drive shown)

③ Apply Power

① Plug the molded AC cable into the power connector on the drive (refer to the previous figure).

② Plug the other end of the cable into a 115VAC power source. ***If you are using the SX8 Drive (high power), you must also plug in the fan's power cable to 115VAC. The fan must be on when power is applied to the drive***. The motor should rotate in an Alternating mode approximately 6 revolutions at 1 rps. The **green** Power LED should be on.

③ **To stop the motor, you must unplug the power cable from the power source. The motor may continue to run for a few seconds after you remove power.**

The successful completion of this test indicates that the amplifier, motor, and microprocessor are operating properly.

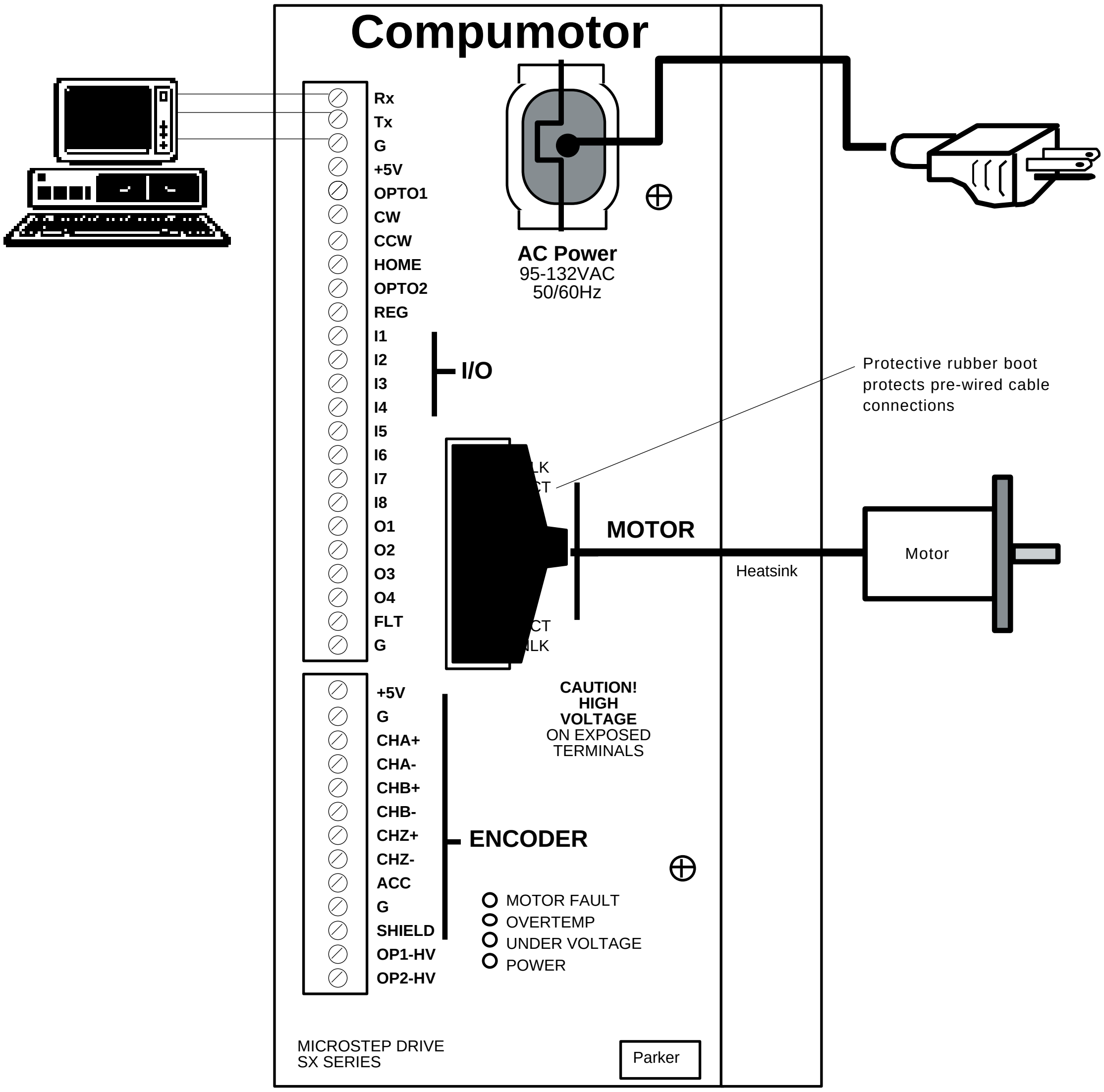
Indexer Test

This section will show you how to quickly test the RS-232C Indexer to assure proper operation.

With no power applied to the drive, perform the following steps to test the communications. ***The default motor resolution setting for the SX is set to 25,000 steps/rev.***

Step ①

Remove the panel that covers the DIP switches. Turn DIP switch **`SW2-#8`** (off) to disable the Automatic Test function. Ensure that switches **`SW2-#1`** through **`SW2-#4`** are **off. Do not change any other DIP switch settings**. Screw the panel back onto the drive.



Test Configuration (S6 Drive shown)

A terminal emulator program or communications driver is required to establish communications with the SX. Contact your distributor for a copy of Compumotor’s X-Ware if needed.

Step ②

Connect a terminal to the RS-232C connector (**`I/O`**) using the **`Rx`** (receive), **`Tx`** (transmit), and **`G`** (ground) connections. The basic communications parameters are listed below:

❑ Baud Rate: 9600

❑ Data Bits: 8

❑ Stop Bits: 1

❑ Parity: None

❑ Full Duplex mode

Refer to the operations manual that accompanied your terminal for instructions on setting the communication parameters listed above. Chapter 3, Installation provides more RS-232C information for the SX.

Step ③

Apply power to all system components now. The following response should appear on the terminal:

```
*READY
>
```

Step ④

You must now disable the hardware limits. Type the following command to disable the limits:

```
> LD3
```

To check the state of your hardware limits command, enter the following command:

```
> 1LD
```

You should receive the following response:

```
*3_No_Limits_Enabled
```

Step ⑤

Enter the following commands to move the motor 25,000 steps.

Command	Description
`> A1Ø`	Sets acceleration to 10 rps^2
`> AD1Ø`	Sets deceleration to 10 rps^2
`> V1`	Sets velocity to 1 rps
`> D25ØØØ`	Sets distance to 25,000 steps
`> G`	Initiates motion

Since the default drive resolution is 25,000 steps/rev, the motor should have moved 25000 steps or 1 revolution.

This completes the basic configuration tests. The successful completion of this test means that you wired the motor, terminal (optional), and power connections correctly. The components of your system are functioning properly. If the motor does not move as commanded, enter the following command:

```
>1RSE
```

You should receive the following response:

```
*NO_ERRORS
```

If an error message flashes, correct the problem and repeat step 5.

CHAPTER ③

Installation

Chapter Objectives

The information in this chapter will enable you to:

❏ Ensure that the complete system is installed correctly

❏ Mount all system components properly

Before proceeding with this chapter, you should have completed the steps and procedures in *Chapter 2, Getting Started*.

Installation Precautions

This section contains precautions that you must follow to configure and operate your SX system properly.

Environmental Considerations

An internal thermostat will shut down the drive if it reaches 158°F (70°C) internally. *Current settings in excess of 4A in high ambient temperature environments (above 113°F [45°C]) may require fan cooling to keep the drive's internal temperature within allowable limits and to keep the drive from shutting itself down due to over temperature.*

The maximum allowable motor case temperature is 212°F (100°C). Actual temperature rise is duty cycle dependent.

CAUTION

When connected in parallel, SX motors can overheat if operated at high speeds for extended periods of time.

Wiring Considerations

There are hazardous voltages present on the SX's connectors when power is applied. To prevent injuries to personnel and damage to equipment, note the following guidelines:

❏ Never connect/disconnect the motor from the drive when power is applied. If you do, the motor connector may be damaged. Power should never be applied to the drive when the motor is not connected.

❏ Never increase the current setting (using the drive's DIP switches) to more than 10% greater than the current specified for the motor you are using. Excessive current may cause the motor to overheat and result in a motor failure.

❏ Verify that there are no wire *whiskers* that can short out the motor connections.

❏ If the motor turns in the opposite direction (from the desired direction) after you connect the motor wires to the connector and the connector to the drive, you can change the direction by reversing the leads going to **A+** and **A-** on the motor terminal.

❏ Never extend the **`INLK`** jumper beyond the connector. This jumper is intended to protect the motor connector and should **not** be used as a system interlock.

❏ **Never** probe the drive. **Never** connect anything other than the motor to the motor terminals. Probing or opening the drive in any other way will void the warranty. Hazardous voltages are present within the drive. **The thermal interface will be broken if you open the drive. The thermal interface is critical to the reliability of the drive.**

❏ When connecting the motor to the drive, be sure the connector is firmly seated.

Preventing Electrical Noise Problems

The SX provides power to the motor by switching 170VDC (120VAC input) at 21 KHz (nominal). This has the potential to radiate or conduct electrical noise along the motor cable, through the motor, and into the frame to which the motor is attached. It can also be conducted out of the drive into the AC power line. Should the electrical noise generated by the SX cause problems for your other equipment use the following steps to prevent problems created by the SX:

① Ground the motor casing (*already done for you with Compumotor motors*).

WARNING

You must ground the motor casing. Motor winding case capacitance can cause large potentials to develop at the motor. This can create a lethal shock hazard.

② Avoid extended motor cable runs. Mount the drive as close as is practical to the motor.

③ Mount equipment that is sensitive to electrical noise as far as possible from the SX and motor.

④ Filter power to the SX with a PI type filter and an isolation transformer (refer to the power rating tables later in this chapter). The filter reduces the AC line noise that the SX generates back into the AC line. The Corcom® EP Series filter works well with the SX.

Corcom
1600 Winchester Road
Libertyville, IL 60048 Telephone: (847) 680-7400

⑤ Provide a separate power line for the SX. Do not use the same power circuit for equipment that is sensitive to electrical noise and the SX.

⑥ Shield the motor cable in conduit separate from low voltage signal wires and ensure the conduit is taken to a low impedance earth ground at one point.

Installation Overview

The procedures in this chapter will enable you to configure and wire your system. The following figure shows the front panel of the SX. The following installation steps will be discussed:

❏ Series vs. Parallel Motor Wiring

❏ Motor/SX Configuration (Wiring & Motor Current)
❍ Compumotor Motors

❏ Set DIP Switches

❏ Fan Connection (for SX6 —fan is standard for SX8)

❏ I/O Connections
❍ RS-232C
❍ Limit Inputs
❍ Home Inputs
❍ Programmable Inputs and Outputs
❍ Registration Inputs
❍ Fault Output

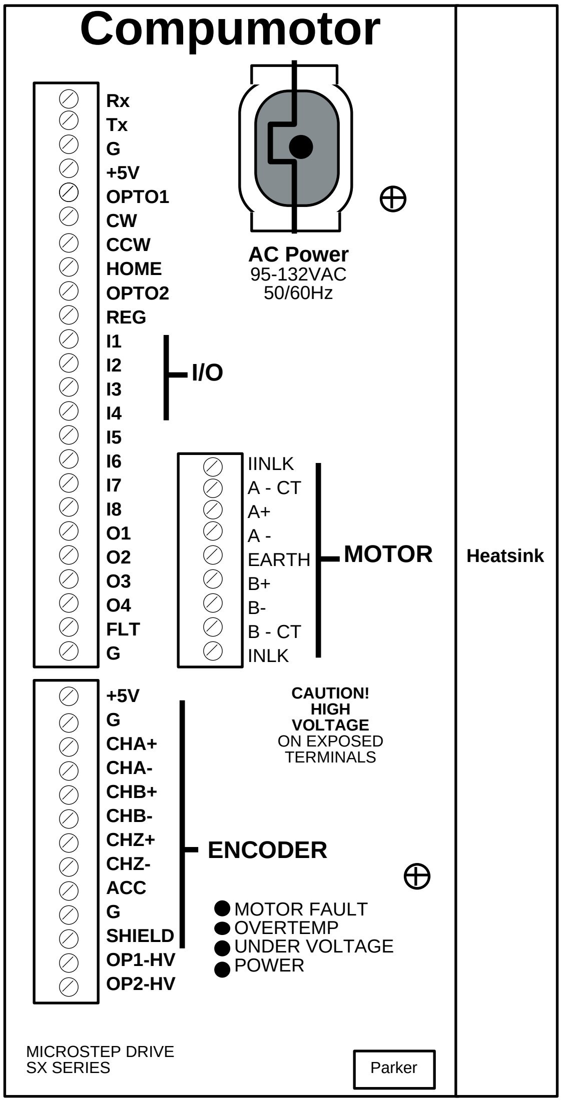
❏ Encoder Connections

❏ Apply Power to SX

❏ Test the System

❏ Mount the SX and the Motor

❏ Attach the Load



SX Wiring Diagram (S6 Drive shown)

Do not deviate from the steps in this chapter. Do not wire or apply power to the system until you are instructed to do so. If you do not follow these steps, you may damage your system.

Series vs. Parallel Motor Wiring

S Series motors are shipped from the factory wired in series. You may re-wire the motor (shown later in this chapter—*Motor Configurations*). Parallel configurations provide more torque than series configurations provide at high speeds (refer to the speed/torque curves in *Chapter 6, Hardware Reference*). You must observe certain precautionary measures to prevent overheating when using motors wired in parallel configurations.

Motor Heating

S Series motors that are wired in series can be run continuously at speeds that incur peak motor loss. S Series motors that are wired in parallel, however, cannot be run at peak motor loss levels continuously without overheating (unless extensive cooling measures are employed). Most applications do not require continuous operation at high speed. Therefore, the average motor loss will be within safe limits.

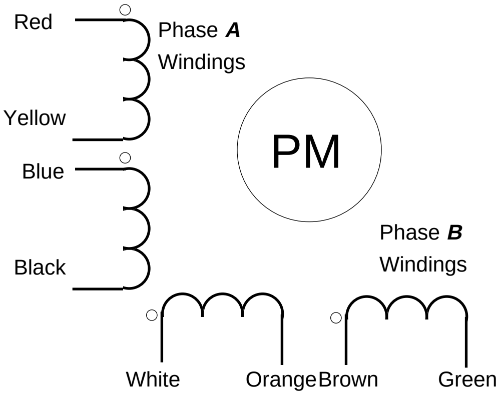
Motor Configurations

The SX Drive can run Compumotor and Non-Compumotor motors. This section provides instructions for configuring Compumotor and Non-Compumotor motors. ***Follow only the directions that apply to the type of motor that you are using***.

Compumotor Motors—Drive/Motor Connection

Compumotor motors are pre-wired in series and require no setup other than being plugged into the drive. If you plan to run the motor is series, no further motor wiring setup is required.

Frame size 23 and 34 motors (SX57 or SX83) are 8 lead motors. Frame size 42 (SX106) are 4 lead motors. The following figure represents the motor winding color code for 8 lead, 23 and 34 frame size motors.



8-Lead Motor Winding Color Code

S Series motors in the 23 and 34 frame sizes (SX57 and SX83 series) are constructed with an 8-conductor motor cable to allow you to change the motor configuration on the connector at the drive. The 42 frame size motors (SX106 series) are constructed with a 4 lead motor cable, but the motors can be configured by removing the cover plate on the back of the motor and rewiring at the screw terminals.

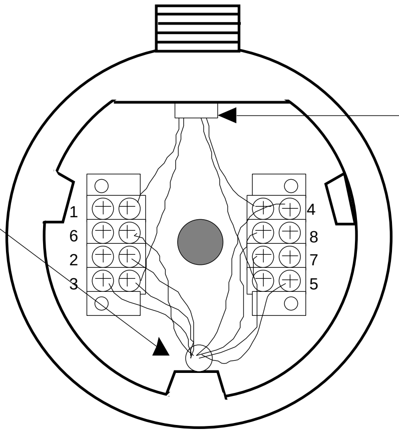
SX106-178 Series and Parallel Connections

The S106-178 is pre-wired in series. If you remove the motor's back panel, you can wire it in parallel.

Motor Terminal #	Wire Color
1	Red
3	Black
5	Green
4	White

Inside Motor Wiring

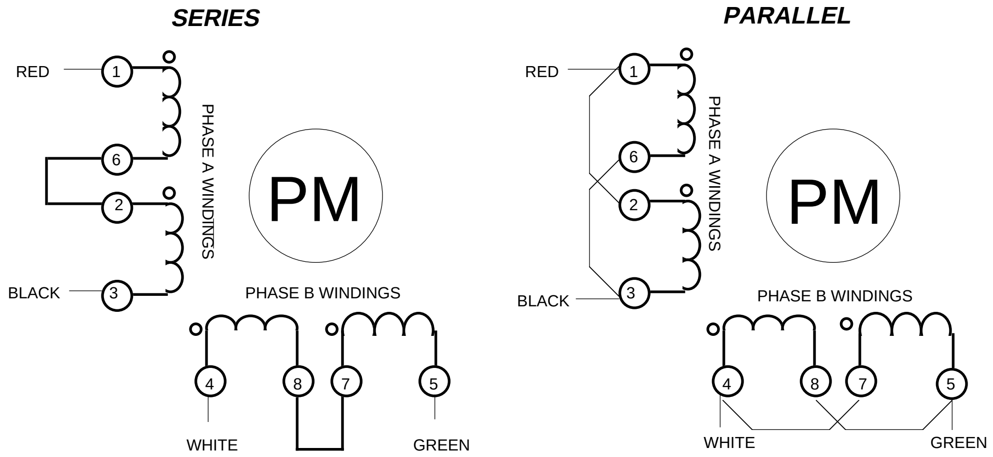
Wire Color	Pin #
Dark Orange	#1
Blue	#2
Black	#3
White	#4
Green	#5
Yellow	#6
Brown	#7
Orange	#8



Drive Terminal Wires

Motor Terminal #	Wire Color	To Drive Terminal
1	Red	A+
3	Black	A-
4	White	B+
5	Green	B-

S106-178 Motor Wiring Diagram



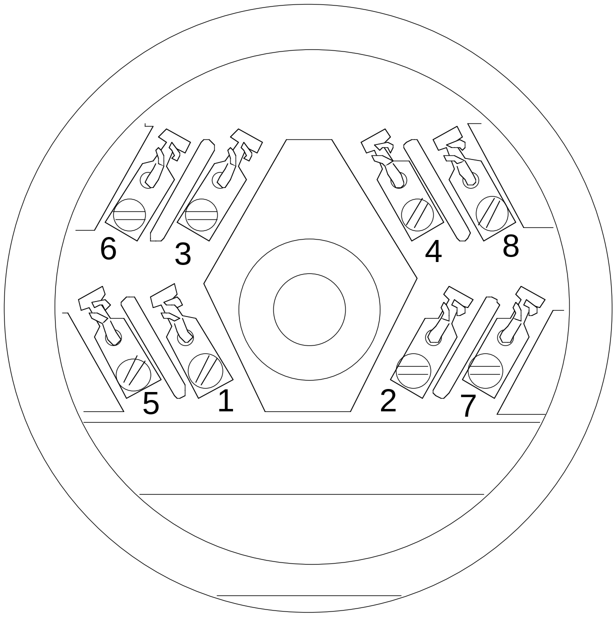
S106-178 Series and Parallel Connections

S106-205 Series and Parallel Connections

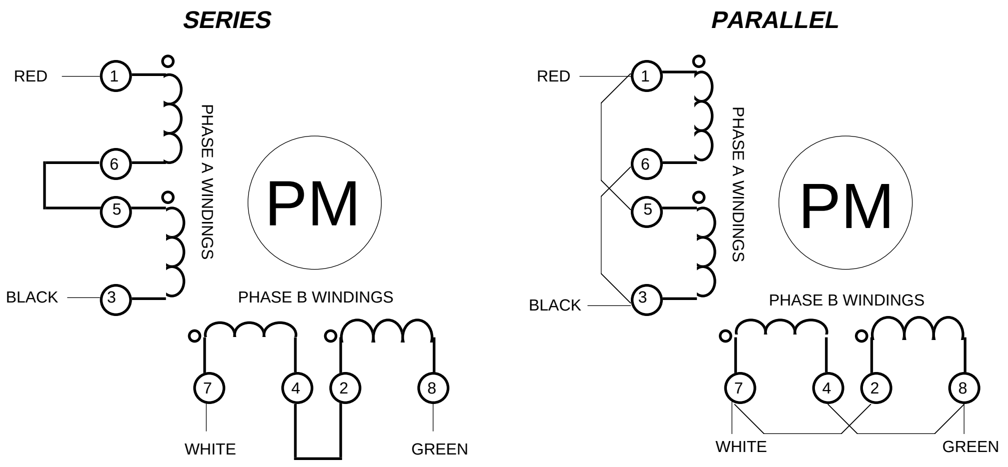
The S106-205 is pre-wired in series. If you remove the motor's back panel, you can wire it in parallel.

Motor Terminal #	Wire Color
1	Red
3	Black
8	Green
7	White

S106-205 Motor



S106-205 Motor Wiring Diagram



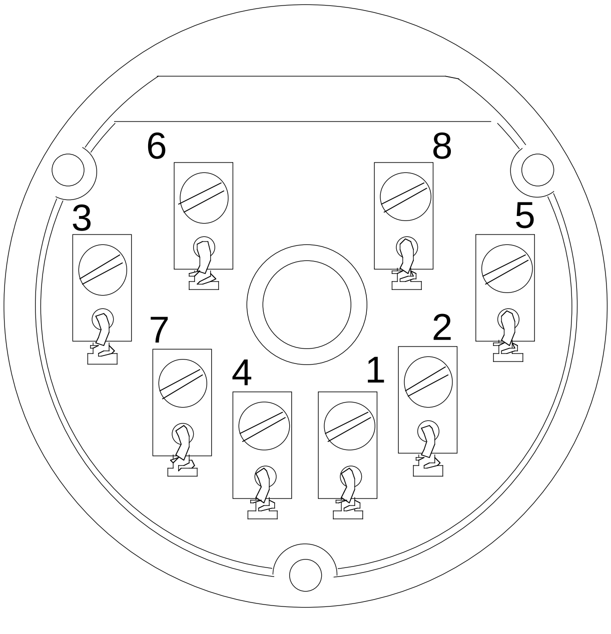
S106-205 Series and Parallel Connections

S106-250 Series and Parallel Connections

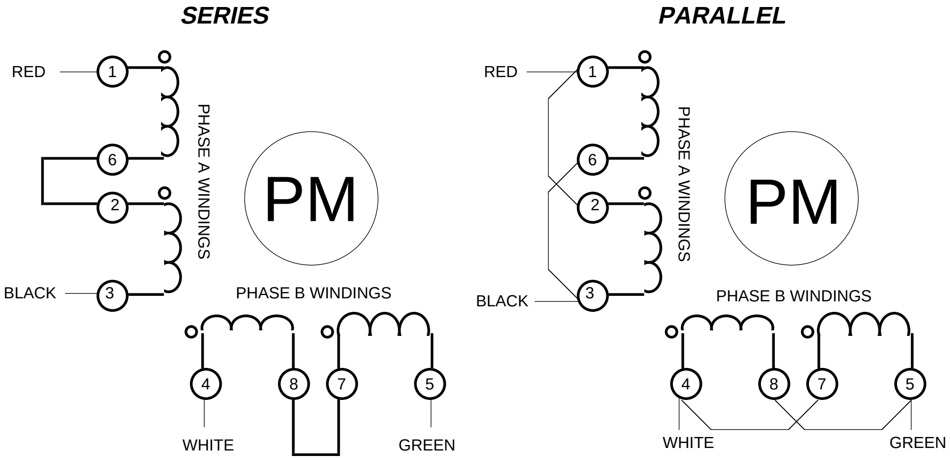
The S106-250 is pre-wired in series. If you remove the motor's back panel, you can wire it in parallel.

Motor Terminal #	**Wire Color**
1	Red
3	Black
4	White
5	Green

S106-250 Motor



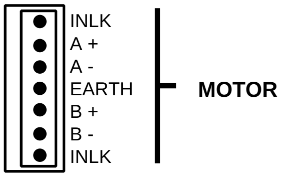
S106-250 Motor Wiring Diagram



S106-250 Series and Parallel Connections

7-Pin Motor Connector

The ***7-pin version***of the **MOTOR** connector is shown below. Before connecting the motor, determine which motor wires correspond to Phase A and Phase B. The 7-pin motor connector provides for easier installation when the motor is wired in series. **A-CT** and **B-CT** are not connections—they are terminal blocks.



S Drive 7-Pin Motor Connector

The following tables show the color codes for the following types of motor connections to the S Drive 7-pin **MOTOR** connector.

❏ 8 Lead Motors—Series (S57 and S83)

❏ 8 Lead Motors—Parallel (S57 and S83)

❏ 4 Lead Motors—Series or Parallel (S106)

Pin	7-Pin/8 Lead Series— Color	7-pin/8 lead Parallel— Color	7-pin/4 Lead S & P— Color
Connected	Yellow & Blue		
A+	Red	Red & Blue	Red
A-	Black	Black & Yellow	Black
EARTH	Shield	Shield	Shield
B+	White	White & Brown	White
B-	Green	Green & Orange	Green
Connected*	Orange & Brown	N.C.	N.C.
	Jumper INLK to INLK	Jumper INLK to INLK	Jumper INLK to INLK

*Refer to your local electrical code for proper termination of these center tap leads

Color Code—7-Pin Connector/8 Lead Motor (Series)

☞ Helpful Hint: *Scenario #1*

The resistance measurements to the two remaining motor leads are virtually identical. Label the two remaining motor leads **A+** and **A-**. Label the motor lead connected to the negative lead of the ohmmeter **A-CT** (this is the center tap lead for Phase A of the motor).

☞ Helpful Hint: *Scenario #2*

The resistance measurement to the second of the three motor leads measures 50% of the resistance measurement to the third of the three motor leads. Label the second motor lead **A-CT** (this is the center tap lead for Phase A of the motor). Label the third motor lead **A-**. Label the motor lead connected to the ohmmeter **A+**.

⑦ Repeat the procedure as outlined in step 6 for the three leads labeled **B** (**B-CT** is the center tap lead for Phase B of the motor).

⑦ Repeat the procedure as outlined in step 6 for the three leads labeled **B** (**B-CT** is the center tap lead for Phase B of the motor).

⑧ *If your S Drive has a 7-pin motor connector*, cover the two motor leads labeled **A-CT** and **B-CT** with electrical tape or shrink tubing to prevent these leads from shorting out to anything else. Do not connect these leads together or to anything else.

If your S Drive has a 9-pin motor connector, connect the **A-CT** motor lead to the **A-CT** pin on the **MOTOR** connector. Connect the **B-CT** motor lead to the **B-CT** pin on the **MOTOR** connector.

⑨ Proceed to the *Terminal Connections* section below.

Series Configuration

Use the following procedures for series configurations.

① *If your S Drive has a 7-pin motor connector*, connect the motor leads labeled A2 and A3 together and cover this connection with electrical tape or shrink tubing. Make sure these leads are not connected to the S Drive.

If your S Drive has a 9-pin motor connector, you can connect A2 and A3 to **A-CT**. You may also connect B2 and B3 to **B-CT**.

② Relabel the A1 lead to **A+**.

③ Relabel the A4 lead to **A-**.

④ If your S Drive has a 7-pin motor connector, connect the motor leads labeled B2 and B3 together and cover this connection with electrical tape or shrink tubing. Make sure these leads are not connected to the S Drive.

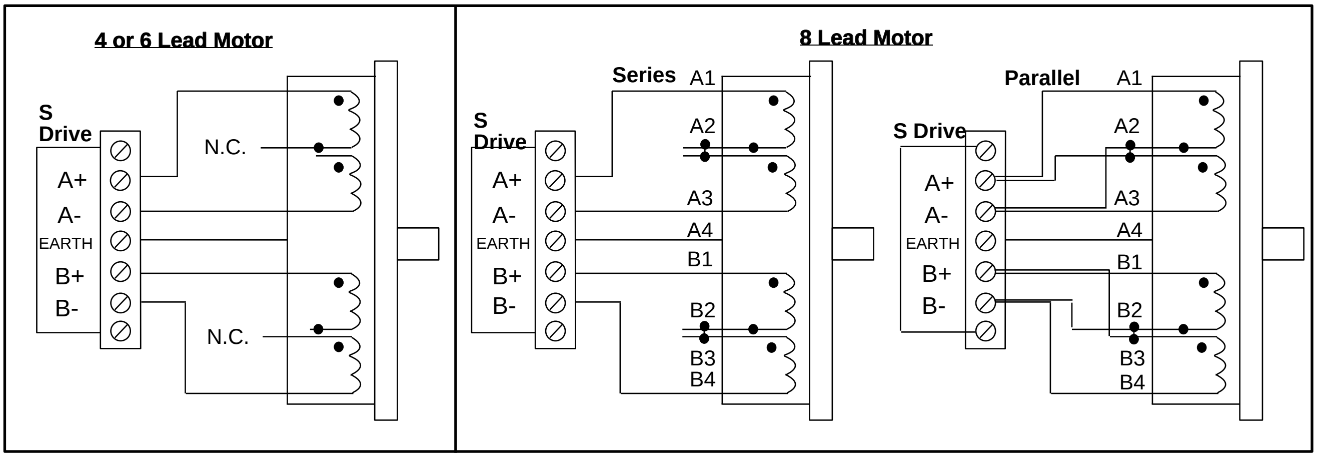
⑤ Relabel the B1 lead to **B+**.

⑥ Relabel the B4 lead to **B-**.

⑦ Proceed to the *Terminal Connections* section below.

Terminal Connections

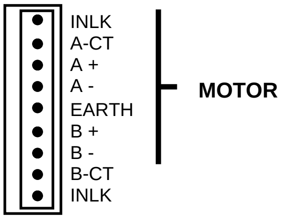
After determining the motor's wiring configuration, connect the motor leads to the 9-pin or 7-pin **MOTOR** connector using the diagrams below.



7-Pin Motor Connector (Non-Compumotor Motors)

9-Pin Motor Connector

The following figure shows the 9-pin version of the **MOTOR** connector. Before connecting the motor, determine which motor wires correspond to Phase A and Phase B. The 9-pin motor connector provides for easier installation when the motor is wired in series. **`A-CT`** and **`B-CT`** are not connections—they are terminal blocks.



SX 9-Pin Motor Connector

The following table shows the color codes for the following types of motor connections to the SX 9-pin **MOTOR** connector.

❑ 8 Lead Motors—Series (S57 and S83)

❑ 8 Lead Motors—Parallel (S57 and S83)

❑ 4 Lead Motors—Series or Parallel (S106)

Pin	9-Pin/8 Lead Series Color	9-pin/8 lead Parallel Color	9-pin/4 Lead S & P Color
A-CT	Yellow & Blue	N.C.	N.C.
A+	Red	Red & Blue	Red
A-	Black	Black & Yellow	Black
EARTH	Shield	Shield	Shield
B+	White	White & Brown	White
B-	Green	Green & Orange	Green
B-CT	Orange & Brown	N.C.	N.C.
	Jumper INLK to INLK	Jumper INLK to INLK	Jumper INLK to INLK

Color Code–9-Pin Connector

Once you determine the wiring configuration, connect the motor to the drive's screw terminals according to the appropriate color code table. The following instructions should also be completed.

① Connect shield to the **MOTOR** connector's shield. **This is a very important safety precaution**. If your motor does not have a ground (shield) wire, attach a lug to the motor case and connect the motor to **EARTH**.

② Connect a short jumper wire from **INLK** (first pin of connector) to **INLK** (last pin of connector). This is a connector interlock. The drive will not operate if this jumper is missing or extended.

Extended Motor Cables

This table contains the recommended motor cables for various motor types and the minimum recommended motor/driver wire size (AWG) and resistance.

Motor Series	**Maximum Current Per Winding (Amps)**	**Less than 100 ft. (20.5M)**	**100 - 200 ft. (30.5M - 71M)**
SX57	3	22 AWG	20 AWG
SX83	6	20 AWG	18 AWG
SX106	8	16 AWG	14 AWG

Recommended Motor Cables

Cable runs of more than 200 feet (71M) are not recommended. Cable runs greater than 50 feet may degrade system performance.

Setting Motor Current

You should verify which type of SX you have before setting motor current. The high-power drive (**SX8**) provides bipolar 0 - 8 amps/phase (up to 1,900 oz-in). The low-power drive (**SX6**) provides bipolar 0 - 6 amps/phase (up to 400 oz-in). You can determine which drive you have by checking the label on the top of the drive. The label identifies the unit as **`SX8 DRIVE`** (SX106) or **`SX6 DRIVE`** (SX57 or SX83). You must be aware of the drive's type to set the motor current correctly (using DIP switches). The tables below contain the proper motor current settings for Compumotor motors. **`SW1-#1`** thru **`SW1-#6`** control ***motor current***. Adjust the motor current to match the drive and motor that you are using. A complete list of all motor current settings is provided in *Chapter 6, Hardware Reference*.

Motor Size	Current	SW1-#1	SW1-#2	SW1-#3	SW1-#4	SW1-#5	SW1-#6
S57-51**S**	1.18	off	off	on	on	off	off
S57-51**P**	2.28	off	on	on	off	off	off
S57-83**S**	1.52	off	on	off	off	off	off
S57-83**P**	3.09	on	off	off	off	off	off
S57-102**S**	1.71	off	on	off	off	on	off
S57-102**P**	3.47	on	off	off	on	off	off
S83-62**S**	2.19	off	on	off	on	on	on
S83-62**P**	4.42	on	off	on	on	on	off
S83-93**S**	2.85	off	on	on	on	on	off
S83-93**P**	5.62	on	on	on	off	on	on
S83-135**S**	3.47	on	off	off	on	off	off
S83-135**P**	6.00	on	on	on	on	on	on

S: Series Configuration P: Parallel Configuration

SX6 Drive Motor Current (Compumotor Motors)

Motor Size	Current	SW1-#1	SW1-#2	SW1-#3	SW1-#4	SW1-#5	SW1-#6
S106-178**S**	6.02	on	off	on	on	on	on
S106-178**P**	8.00	on	on	on	on	on	on
S106-205**S**	3.55	off	on	on	on	off	off
S106-205**P**	6.99	on	on	off	on	on	on
S106-250**S**	6.23	on	on	off	off	off	on
S106-250**P**	8.00	on	on	on	on	on	on

S: Series Configuration P: Parallel Configuration

SX8 Drive Motor Current (Compumotor Motors)

Compumotor A/AX motors may be used with the SX. However, differences in motor design result in a significant reduction in performance as compared with an SX Motor/Drive system. **Compumotor strongly recommends using an S/SX motor with an SX Indexer/Drive.**

In a retrofit application, customers may order an SX option through Compumotor's Custom Products Group for increased performance when using an A Series motor. This is most important with the 57 frame motors. The custom product number for motor sizes A57-51 through 83-135 is **`CP*SX6-DRIVE-10261`**. This option is also recommended for better performance with motors rated 25-30mH per phase and above.

Motor Size	Current	SW1-#1	SW1-#2	SW1-#3	SW1-#4	SW1-#5	SW1-#6
A/AX57-51	0.32	off	off	off	off	on	on
A/AX57-83	0.51	off	off	off	on	off	on
A/AX57-102	0.70	off	off	off	on	on	on
A/AX83-62	0.80	off	off	on	off	off	off
A/AX83-93	1.37	off	off	on	on	on	off
A/AX83-135	1.90	off	on	off	on	off	off
A/AX106-120	1.90	off	on	off	on	off	off
A/AX106-178S	3.95	on	off	on	off	off	on
A/AX106-178P	6.00	on	on	on	on	on	on
A/AX106-205	6.00	on	on	on	on	on	on

S: Series Configuration P: Parallel Configuration

A/AX Drive Motor Current using an SX6 (Compumotor Motors)

Configuration of the Drive

In this section, you will set the following DIP-switch-selectable functions:

❏ Indexer Address function

❏ RS-232C Baud Rate setting

❏ Automatic Test function

Setting Indexer Address

Switches **SW2-#1** - **SW2-#4** control the device address (refer to the following table). Each SX is factory set to device address 1. If you want to daisy-chain you must establish a unique address for each SX Indexer/Drive. The device address can be changed with switches **SW2-#1** - **SW2-#4**.

	Address	SW2-#1	SW2-#2	SW2-#3	SW2-#4
*	**1**	**off**	**off**	**off**	**off**
	2	on	off	off	off
	3	off	on	off	off
	4	on	on	off	off
	5	off	off	on	off
	6	on	off	on	off
	7	off	on	on	off
	8	on	on	on	off
	9	off	off	off	on
	10	on	off	off	on
	11	off	on	off	on
	12	on	on	off	on
	13	off	off	on	on
	14	on	off	on	on
	15	off	on	on	on
	16	on	on	on	on

* Default Setting

Indexer Address Settings

Setting RS-232C Baud Rate

DIP switches **SW2-#5** thru **SW2-#7** allow you to set the RS-232C baud rate

	Baud Rate	SW2-5	SW2-6	SW2-7
*	**9600**	off	off	off
	9600	off	on	off
	4800	on	on	off
	2400	off	off	on
	1200	on	off	on
	600	off	on	on
	300	on	on	on

* Default Setting

Baud Rate Settings

Automatic Test Function

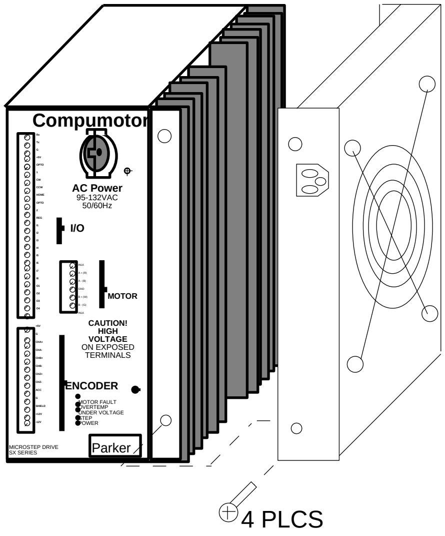
The Automatic Test (***DIP switch* SW2-#8**) function turns the motor shaft slightly less than six revolutions in Alternating mode at 1 rps. The Automatic Standby function and motor resolution settings are disabled when you use the Automatic Test function.

***** SW2-#8 OFF Disables Auto Test
SW2-#8 ON Enables Auto Test

* Default Setting

Fan Connection

The fan kit is a standard feature of the **SX8** (high-power). If you are using the **SX6** (low-power), you may order the fan kit from your Automation Technology Center (ATC) or Compumotor Distributor. *Ensure that the fan is powered when the SX8 is on*.



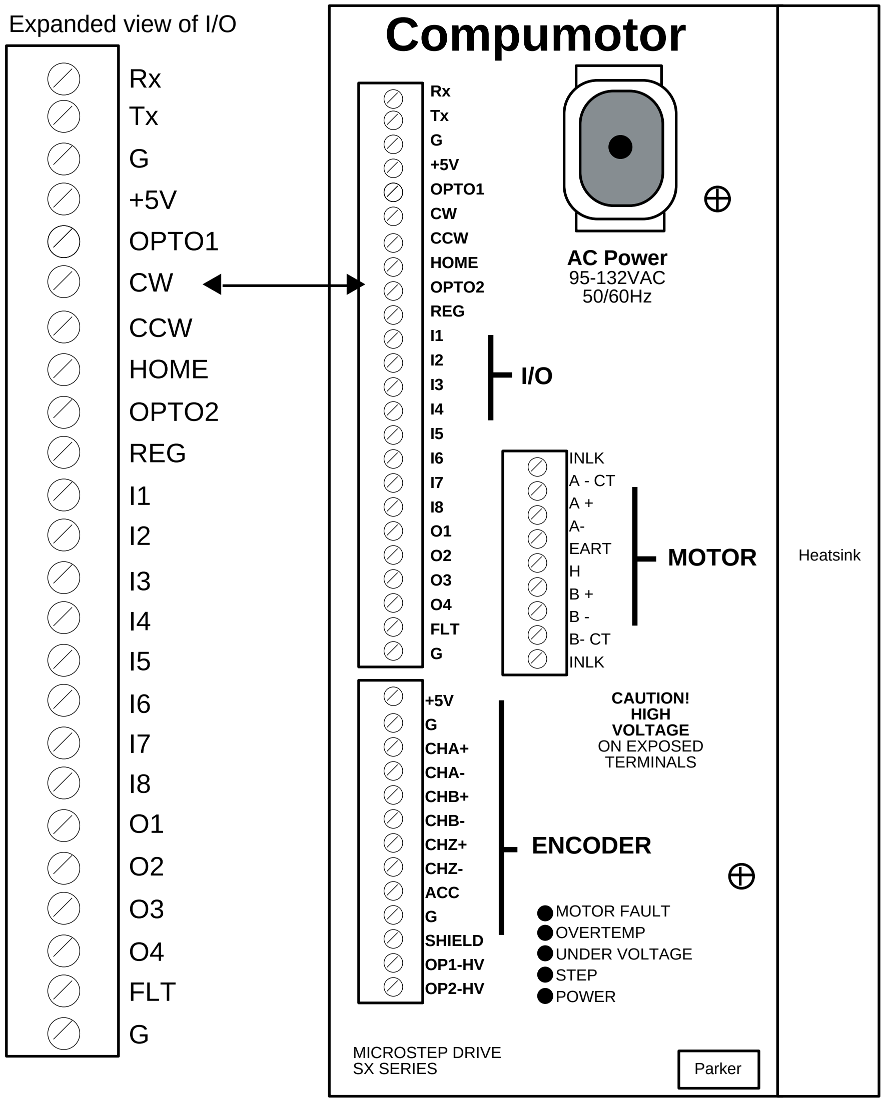
Fan Connection

I/O Connections

The SX's **`I/O`** connector provides the following communication, input, and output connections.

- Communication
  - RS-232C
- Inputs
  - +5 Volts
  - OPTO1-HV & OPT02-HV
  - End-of-Travel Limits
  - Home Position Input
  - Registration Input
  - Eight programmable inputs
  - OP1-HV & OP2-HV
- Outputs
  - Four programmable outputs
  - Fault Output

The following figure shows the location of these connections.

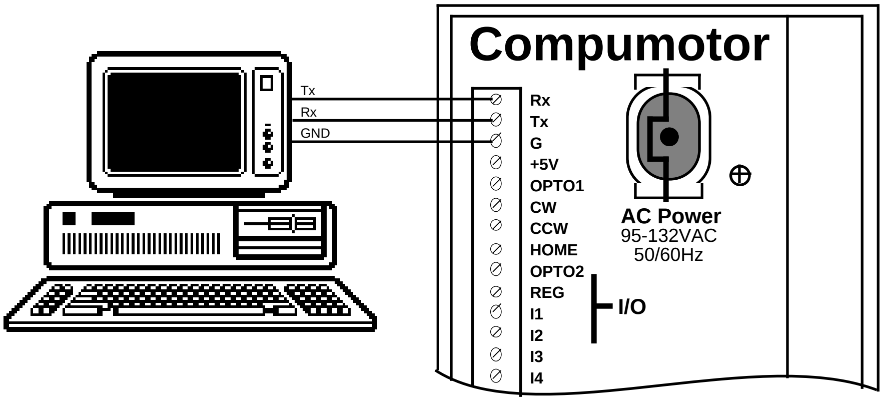


Screw Terminal I/0 (S6 Drive shown)

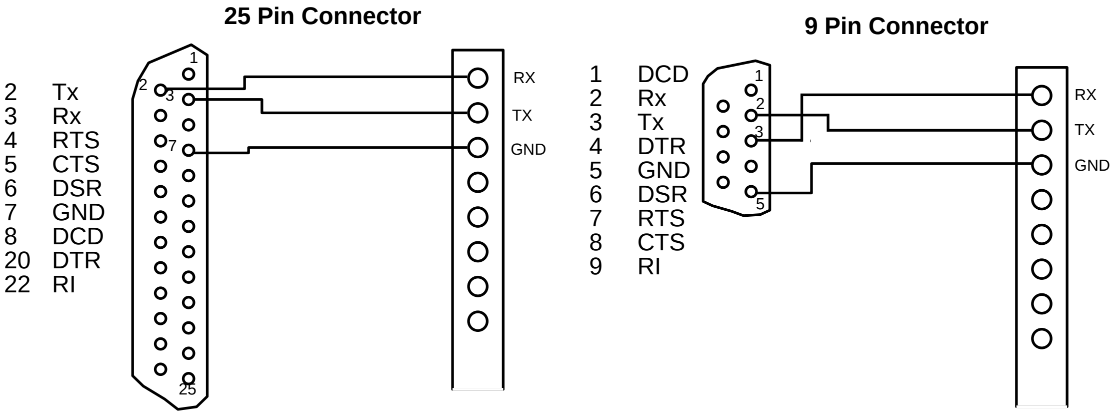
RS-232C Connections (RX, TX, GND) I/O

The SX can communicate to any terminal or host computer that can be configured for RS-232C. The SX has a set of commands that you can use to set up the drive, program the drive, and report back drive data. Compumotor supplies an editor/terminal emulator program (X-Ware) to facilitate communications from a host computer. Contact your local ATC or distributor for a copy. Any terminal emulator or communications driver capable of using the available communications parameters will also work.

The SX has a three-wire, optically isolated RS-232C interface that is compatible with RS-232C specifications. Receive Data (**Rx**), Transmit Data (**Tx**), and ground (**GND**) signals are connected on the screw terminal **I/O**. Proper shielding of the RS-232C signal wires is required. The shield should be connected to an earth ground point on the terminal. The following figure shows standard RS-232C connections. The second figure shows standard 25-pin and 9-pin outputs for serial communication ports.



RS-232C Connnections



RS-232C Standard Pin-Outs

The rest of the signals involve RS-232C handshaking. The SX does not support handshaking. If your system requires handshaking, connect RTS to CTS and DTR to DSR.

The default communication parameters are

❏ Baud Rate: 9600

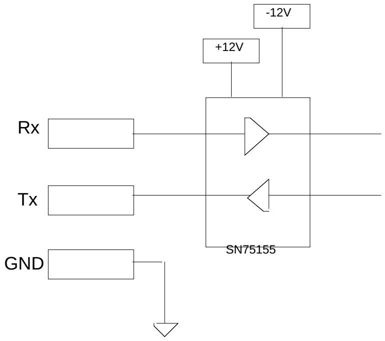
❏ Data Bits; 8

❏ Stop Bit: 1

❏ Parity: None

Handshaking is not supported. The terminal should be set for Full Duplex mode.

You can change the baud rate with the DIP switches (refer to previous section). Baud rates of 300, 600, 1200, 2400, 4800, and 9600 are available. The RS-232C communication interface is optically isolated. The following figure is a schematic of the RS-232C communication interface.



RS-232C Input

SX Daisy Chain Wiring

You may daisy chain up to 16 SXs. Individual drive addresses are set with the SX's DIP switches (refer to the previous section). When daisy chained, the units may be addressed individually or simultaneously. You should establish a unique device address for each SX. Refer to the following figure for SX daisy chain wiring configuration.

Commands prefixed with a device address instruct only the unit specified. Commands without a device address instruct all units on the daisy chain.

For example the Go (**G**) command instructs all units on the daisy chain to go, while **1G** tells only unit #1 to go.

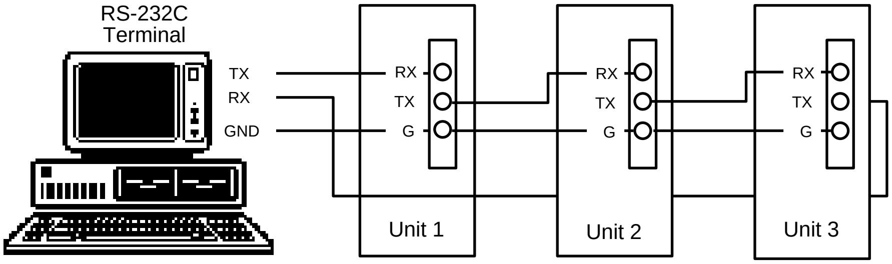
No SX executes a device-specific command unless the address number specified matches the SXs unit number. Device-specific commands include both buffered and immediate commands. This becomes critical if you instruct any Indexer to transmit information. To prevent all of the units on the line from responding to a command, you must precede the command with the device address of the designated unit.

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.

You must use status-request commands in an orderly fashion. Commands should only be issued when the host is ready to read the response. You should not send new commands until you receive a response from the previous status-request command. In particular, you should not issue a immediate-status command until the host receives a buffered command status response. If this is not followed, the command responses will be intertwined, rendering the data useless.

If you enable the Interactive mode (**SSIØ**), only the SX that is set to address #1 will respond with a prompt (>). This prevents all the SXs from sending out > in a daisy chain. Compumotor recommends disabling the interactive mode in all units when in a daisy chain configuration to prevent the > from address #1 being embedded in programs stored at other addresses. The default for the **SSI** command is enabled (**SSIØ**).

The following figure shows a multiple-drive configuration (daisy-chain) of RS-232C ports from one controlling terminal or computer.



RS-232C Daisy Chain Configuration

Sample Applications and Commands

Three SXs are on an RS-232C daisy chain. Send the following commands:

Command	Description
> MN	Sets unit to Preset mode
> A5	Sets acceleration to 5 rps^2 for all three controllers
> V1Ø	Sets velocity to 10 rps for all three controllers
> LD3	Disables limits (in case they are not connected)
> 1D25ØØØ	Sets Axis 1 distance to 25,000 steps
> 2D5ØØØØ	Sets Axis 2 distance to 50,000 steps
> 3D1ØØØØØ	Sets Axis 3 distance to 100,000 steps
> G	Moves all axes

Internal +5V Supply

This is the connection to the internal, isolated +5V supply. This supply is rated at 250mA maximum and is primarily designed to power an optical encoder. This supply may be used as a power source for the optically isolated I/O if an encoder is not being used.

CAUTION

Do not attempt to power both an encoder and the I/O from the +5V supply.

OPTO1

This terminal is the (5-12VDC) source input for the optically isolated **CW**, **CCW** and **HOME** inputs. The following figure is a schematic showing the **OPTO1** input. Refer to *Chapter 6, Hardware Reference* for the electrical specifications.

OP1-HV

☞ Note

Older SX units may not have OP1-HV connections. If not, 12-24VDC will require a zener diode to clamp the voltage at 12VDC. Applying 12-24VDC to OPTO1 without the zener diode may cause damage. Customers currently using zeners can continue using them on OPTO2 or choose to use OP1-HV, which do not require the zener diodes. Refer to *Chapter 6, Hardware Reference* for diode specifications and wiring.

CAUTION

OPTO1 and OP1-HV should not be used at the same time. Damage may occur if they are both wired to power supplies at the same time.

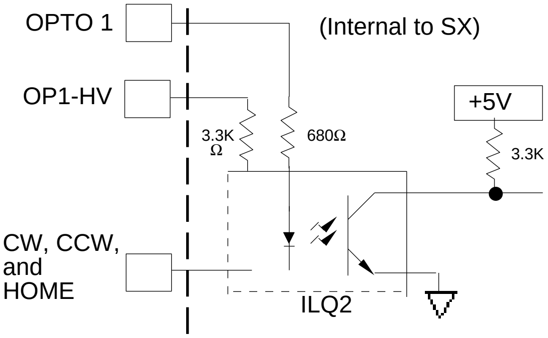
CW and CCW Limits

The SX has two dedicated hardware end-of-travel limits (**CCW** and **CW** on the front panel). When you power up the SX, these inputs are enabled and are expecting switches/sensors normally closed to ground (use the **OSA** command to change the limit active level). If you want to test the SX without connecting the CCW and CW switches, you must disable the limit inputs with the **LD3** command. If you command a move without disabling the inputs, the SX motor will not turn. You can use the **RA** (Limit Switch Status Report), **IS** (Input Status), and **IN** (Set Input Functions) commands to test the limits' status. The following figures are schematics showing the optically isolated limit inputs, typical 3-wire sensor wiring, and typical hard contact wiring. Refer to *Chapter 6, Hardware Reference* for the electrical specifications.

The SX also has software limit capabilities. The software limits are disabled when you power up the system. If you need software limit capabilities, you can enable and define these software limits. Refer to the ***SX Software Reference Guide***—Software Limits (**SL**) command.

Home Position Input

The SX's dedicated Home Position input [**HOME**] provides a reference for your applications motion. The following figures show typical switch wiring configurations. This input defaults expecting a switch/sensor that is normally open (use the **OSC** command to change home active levels) and may be used to command a machine to start an operation from a repeatable position. You can use this input in conjunction with the Go Home (**GH**) command or a Go Home input configured with the Set Input Functions (**IN**) command. When the SX executes a Go Home (**GH**) command, it scans the Home Position input until the switch activates the Home Position input. The following figure is a schematic showing the Home Position input. Refer to *Chapter 6, Hardware Reference* for the Home Position input's electrical specifications. The homing function is discussed in *Chapter 4, Application Design*.



Note: OPTO1 is for use with (5-12VDC) power supplies and OP1-HV is for use with (12-24VDC) power supplies.
They should not be used together.

CW, CCW, and Home Inputs

OPTO2

This terminal is the (5-12VDC) source input for the optically isolated **REG** and **I1** - **I8** inputs. The following figure is a schematic showing the **OPTO2** input. Refer to *Chapter 6, Hardware Reference* for the electrical specifications.

OP2-HV

✒ Note

Older SX units may not have OP2-HV connections. If not, (12-24VDC) will require a zener diode to clamp the voltage at 12VDC. Applying (12-24VDC) to OPTO2 without the zener diode may cause damage. Customers currently using zeners can continue using them on OPTO2 or choose to use OP2-HV, which do not require the zener diodes. Refer to *Chapter 6 Hardware Reference* for diode specifications and wiring.

CAUTION

OPTO2 and OP2-HV should not be used at the same time. Damage may occur if they are both wired to power supplies at the same time.

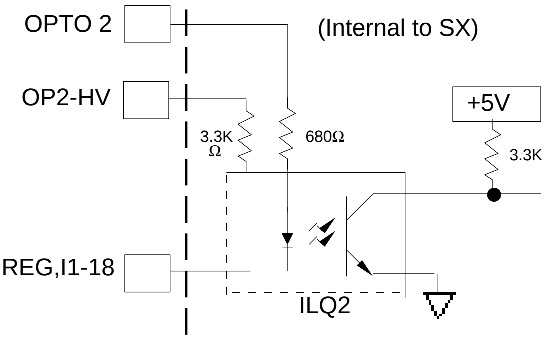
REG Input

The SX has a dedicated hardware registration input. The following figure is a schematic showing the optically isolated **REG** input. Refer to *Chapter 6, Hardware Reference* for the electrical specification. The registration function is discussed in *Chapter 4, Application Design*.

I1—I8 Inputs

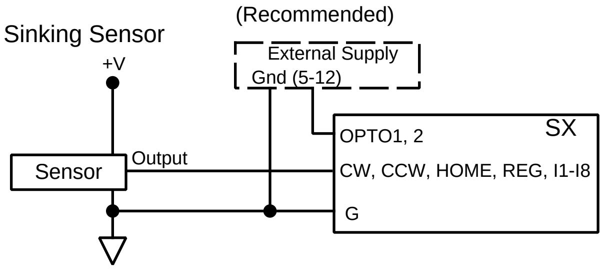
The SX has eight general purpose programmable inputs that default expecting switch/sensors that are normally open (use the **INL** command to change input active level). Each input can be programmed to perform 24 different functions. The inputs can be used with PLCs and configured with the outputs to interface with thumbwheel switches. The following figure is a schematic showing the optically isolated general-purpose programmable inputs. Refer to *Chapter 6, Hardware Reference* for the programmable inputs' electrical specifications.

REG and I1 - I8 Inputs



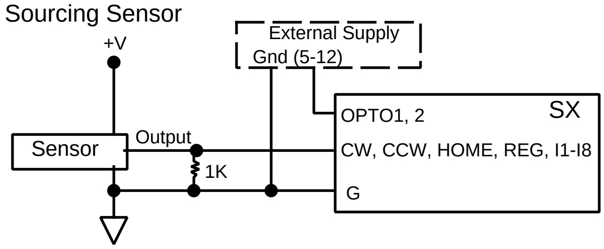
Note: OPTO2 is for use with (5-12VDC) power supplies and OP2-HV is for use with (12-24VDC) power supplies.
They should not be used together.

Typical 3-Wire Sensor Input Connections



+v may use the same external voltage supply as OPTO1, 2

Note: Use OP1-HV and/or OP2-HV in place of OPTO1 and/or OP1-HV if (12-24VDC) is being used.



Note: The 1K resistor value may vary depending on sensor type.

CAUTION

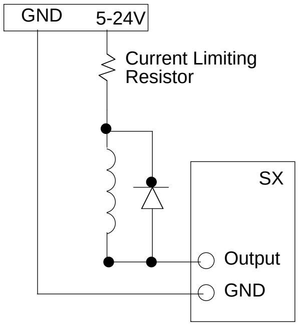
The maximum reverse voltage across OPTO1 & 2 and their corresponding inputs is 3VDC. A zener diode or blocking diode may be required (on the input as well) if applying 24VC to the inputs from a PLC output or other source.

O1—O4 Outputs

The SX has four general purpose programmable outputs. The output is an optically isolated open collector darlington transistor. You can program these outputs to perform 16 different functions.

☞ **Helpful Hint:**

Inductive Loads



If an inductive load is used, you ***must*** put a diode across the load, with the anode connected to the SX output (see previous figure). These outputs can sink up to 35mA.

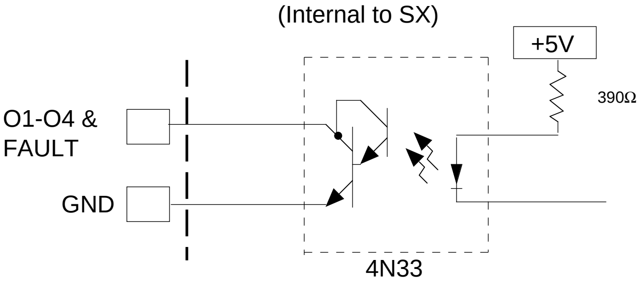
External Supply Voltage	**Minimum Resistor Value**	**Power Dissipation**
5	150	0.167W
15	350	0.411W
24	690	0.835W
30	860	1.050W

Refer to *Chapter 6, Hardware Reference* for the programmable output's electrical specifications. The following figure is a schematic showing the optically isolated general-purpose programmable outputs.

FAULT

The SX has one dedicated hardware fault output. The output is an optically isolated open collector darlington transistor. The following figure is a schematic showing the optically isolated fault output. Refer to *Chapter 6, Hardware Reference* for the electrical specifications. The fault output is normally conducting current in the non-faulted state. The transistor turns off when a fault occurs. The following conditions will cause the fault output to turn off:

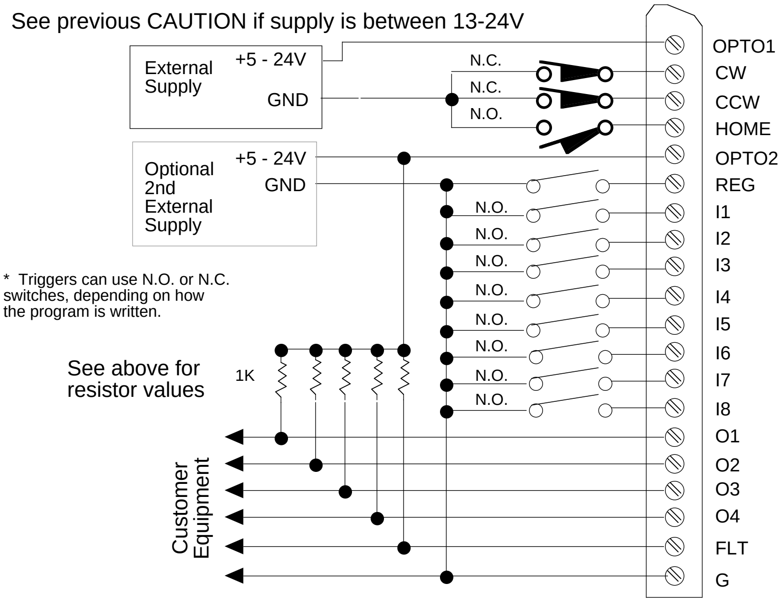
- ❑ User Fault Input
- ❑ Brown Out Condition
- ❑ Motor Fault
- ❑ Battery Back-up RAM Corrupted
- ❑ SX in Auto Run mode
- ❑ Excessive Positioning Error
- ❑ Amplifier Overheating



Outputs and Fault

GND Connection

This terminal is the ground reference for the open collector outputs.


See previous CAUTION if supply is between 13-24V
External Supply
+5 - 24V
GND
Optional 2nd External Supply
+5 - 24V
GND
N.C.
N.C.
N.O.
N.O.
N.O.
N.O.
N.O.
N.O.
N.O.
N.O.
N.O.
* Triggers can use N.O. or N.C. switches, depending on how the program is written.
See above for resistor values
1K
Customer Equipment
OPTO1
CW
CCW
HOME
OPTO2
REG
I1
I2
I3
I4
I5
I6
I7
I8
O1
O2
O3
O4
FLT
G

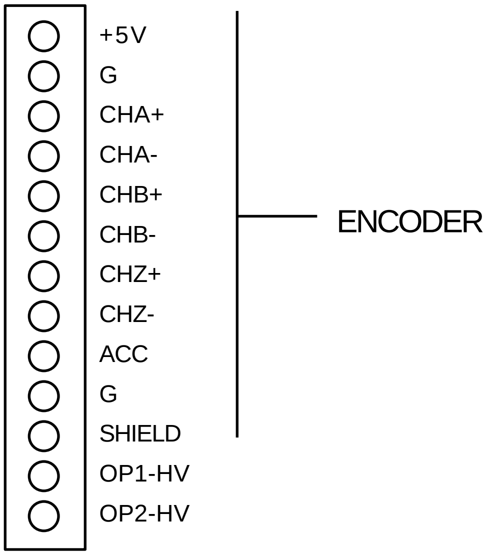
Typical I/O Connections

Encoder Connections

✒ Note

As of January 1, 1995, the SX/SXF will no longer have absolute encoder interface capability as a standard feature. The standard SX/SXF will not be compatible with the AR-C absolute encoder, but rather the SX/SXF absolute encoder interface will be an option to the standard system. For help in determining whether or not your SX/SXF has the absolute encoder interface, see the RVV command in the ***SX Software Reference Guide***.

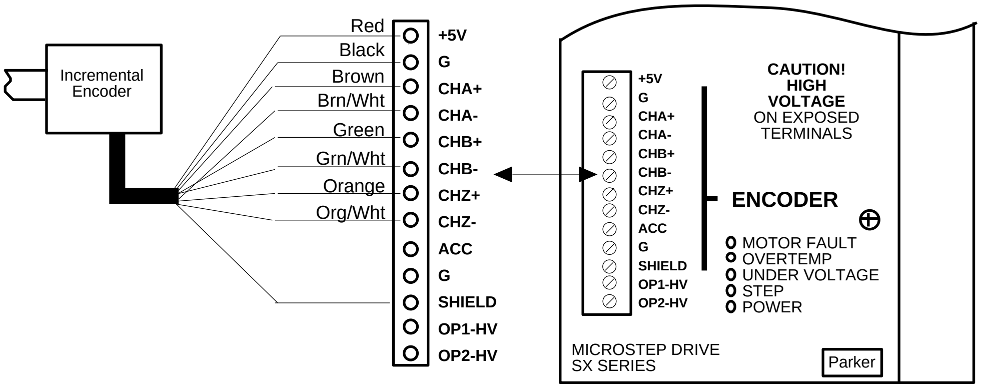
The SX Indexer/Drive supports incremental and absolute encoders. The following figure shows the SX's encoder terminals. All encoder connections for incremental or absolute encoders are made to these terminals.



Incremental Encoder Connection

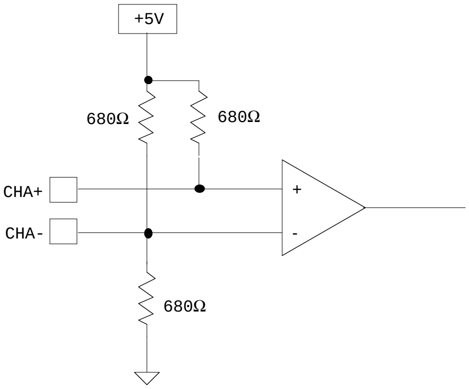
Connections for a typical incremental encoder are shown here.

Color codes shown are for Compumotor's -E optional incremental encoder.



Incremental Encoder Connections

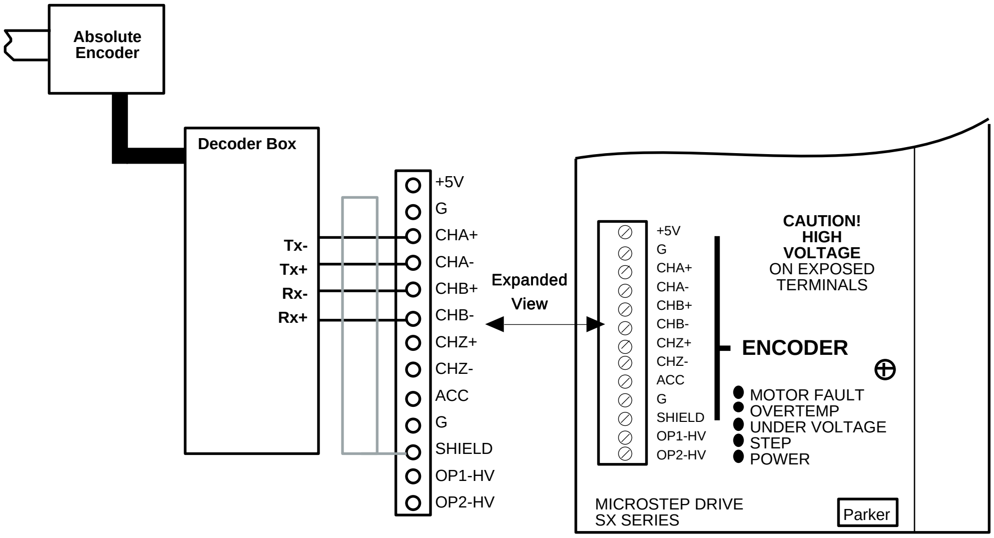
This figure shows the schematic for the incremental encoder inputs.



Incremental Encoder Schematic

Absolute Encoder Connection

The connection for Compumotor’s AR-C Absolute Encoder is shown below. For SX’s purchased after January 1st, 1995, the -A option must be purchased to have Absolute Encoder capabilities.



Absolute Encoder Connection

AC Power Connection

The SX includes a standard molded power cable. Simply plug the power cable into the drive's power connector and a 90VAC - 132VAC power source. If your SX is equipped with a fan kit, plug in a second power cable to the fan kit's power connector and a 90-132VAC power source.

CAUTION

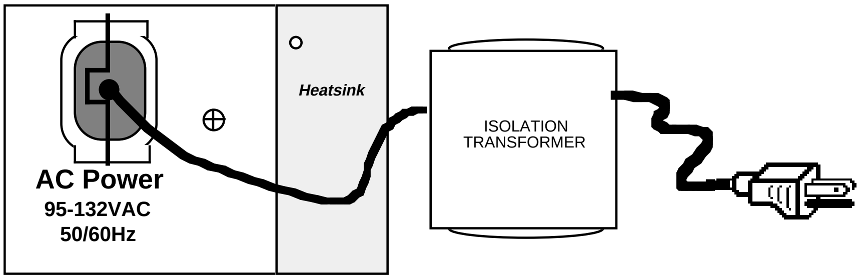
AC power to the SX is limited to 132VAC. Higher voltages will damage the drive. The low-voltage limit is 90VAC.

Transformers

An isolation transformer (optional) can enhance the system's electrical noise immunity. Refer to the *Transformer Specifications* section for instructions on sizing a transformer for your application. Use the transformer user guide and the figure below to connect the transformer leads to the AC power connector on the drive.

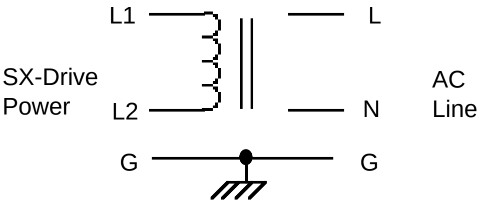
WARNING

Do not connect the transformer to the SX while power is applied to the transformer. Do not touch the wiring studs or terminals on the transformer after it is plugged into an AC outlet. Lethal voltages are present.



Transformer Connections

When powering the SX from a transformer, it is very important that the earth ground terminal is connected.



Earth Ground Terminal

Transformer Specifications

The following tables contain power rating data to help system designers cool drives and motors, and size isolation transformers. Each of the tables' fields is explained below. Combinations of motors and current levels other than those discussed in this section will result in power values that are not specified in this discussion.

Power Ratings

Motor Type	Cabinet Loss (Watts)	Peak Motor Loss (Watts)	Peak Shaft Power (Watts	Peak Total Power (Watts)	Volt-Amp Rating (VA)
S57-51S	11.2	25	55	90	140
S57-51P	15.8	50	110	180	270
S57-83S	12.7	27	72	110	170
S57-83P	19.8	54	144	218	335
S57-102S	14.5	30	95	140	215
S57-102P	25.1	60	190	280	420
S83-62S	14.5	50	120	190	280
S83-62P	25.1	100	240	370	560
S83-93S	18.2	52	172	240	370
S83-93P	36.6	104	343	480	740
S83-135S	21.8	57	205	280	440
S83-135P	40.0	114	410	560	870

S: Series Configuration   P: Parallel Configuration

SX6 Power Ratings

Motor Type	Cabinet Loss (Watts)	Peak Motor Loss (Watts)	Peak Shaft Power (Watts	Peak Total Power (Watts)	Volt-Amp Rating (VA)
S106-178S	20	140	350	510	790
S106-178P	30	280	700	1010	1570
S106-205S	40	150	230	420	650
S106-205P	40	290	460	790	1230
S106-250S	30	160	360	550	860
S106-250P	40	300	700	1040	1620

S: Series Configuration   P: Parallel Configuration

SX8 Power Ratings

Calculations

❏ To convert watts to horsepower, **divide by 746**.

❏ To convert watts to BTU/hr, **multiply by 3.413**.

❏ To convert watts to BTU/min, **multiply by 0.0569**.

Motor Type

Compumotor S/SX Series motors are custom-made for use with SXs. They are not available as a standard model from any other manufacturer. These motors are designed for low loss at rest and at high speed. Motors in the same frame sizes from other manufacturers may sustain considerably higher iron losses than an S/SX Series motor. S/SX Series motors are wound to render inductances within a particular range suitable for SXs. If you intend to use a motor other than an S/SX Series motor, you should consult Compumotor's Applications Engineering Department for motor heating and drive performance consequences (800-358-9070). The SX is intended for use with 2 phase PM step motors only. Do not use variable reluctance or DC motors.

Current (Amps)

Compumotor has assigned the current ratings (previously shown) to S/SX Series motors to produce the highest possible torque while maintaining smoothness. Use of higher currents will produce higher static torque; however, the motor will run roughly and may overheat. The selected motor current setting for motors wired in parallel is twice the value of the motor current setting selected for motor motors wired in series. Do not run the parallel rated current into a motor that is wired in series—it will destroy the motor's windings. Remember, a motor run in parallel must have a limited duty cycle or overheating and damage in the motor's windings will occur.

Cabinet Loss

The total thermal dissipation in the SX is almost constant, regardless of whether the motor is stationary or in motion. The current output switch settings determine the motor phase currents that cause the power losses shown in the previous tables. The cabinet's thermal resistance is approximately 0.35°C/W in still air with the heatsink fins vertically positioned. For 6A operation, the cabinet will rise approximately 15°C above ambient temperature. The fan kit (which is optional for SX6s) will reduce this temperature rise to 2°C. Application/Product design must prevent ambient temperature around the drive from exceeding 45°C (temperature above 45°C will activate the drive's thermal shutdown feature). If the appropriate temperature cannot be maintained, the fan kit must be installed.

Peak Motor Loss

As the speed of a motor increases, the *core losses* (hysteresis and eddy current) increase to the level where the motor loses torque. The peak dissipation includes core and copper losses. The data in the previous tables does not indicate average power unless the motor is run almost continuously at high speed. Average motor loss will generally be less than these figures depending on the duty cycle and dwell times. Motor losses are almost entirely independent on the mechanical load. ***Motor losses are not related to shaft power***.

S/SX Series motors that are wired in series can be run continuously at speeds that incur peak motor loss. S/SX Series motors that are wired in parallel, however, cannot be run at peak motor loss levels continuously without overheating (unless extensive cooling measures are employed). Most applications do not require continuous slewing at high speed. Therefore, the average motor loss will be within safe limits (refer to the motor sizing information provided in Compumotor's sizing software).

WARNING

Do not run the SX with motors in a parallel configuration without inspecting the thermal behavior of the system. A parallel motor that operates at peak motor loss does not sustain damage immediately. Approximately 10 - 30 minutes of continuous operation may be required to reveal an overheating problem. In general, the motor's case temperature should not exceed 100°C.

Peak Shaft Power

Peak shaft power is the product of torque and velocity in the region where the speed/torque curve appears as a hyperbola. In that speed range, the available shaft power is essentially constant at this peak value. Most applications do not use more than 50% of the available peak shaft power. You should use the peak shaft power values shown in the previous tables to determine the maximum demand on the primary power source.

Peak Total Power

Peak total power is the sum of cabinet loss + peak motor loss + peak shaft power. The average demand will be significantly less than the values provided in the tables depending on duty cycles at high speed and dwell times at rest.

Volt-Amp Rating

SXs obtain DC power by directly rectifying 120VAC, 60 Hz voltage. This is a low-cost, lightweight, small size method of obtaining power. However, such a power supply represents a low-power factor to the line (approximately 0.65 for SXs). The volt-amp ratings provided in the previous tables were calculated by dividing *peak total power* by 0.65.

Summary

Selecting an isolation transformer based on these power ratings will provide you with a conservatively rated system. For slow-speed or light-duty applications, smaller VA ratings may be appropriate.

Installation Verification

After you have completed all of the wiring instructions, you should complete the steps in this section to ensure that you have wired the limits, home, registration, inputs, outputs, motor, and encoder correctly.

Input Conventions

All of the inputs on the I/O connector are optically isolated and are activated by causing current to flow from the OPTO terminal to the appropriate input terminal typically to logic ground through a sinking resistor or contact closure. This is the ***energized*** state.

Input Tests

All of the inputs (limits, **HM**, **REG**, and **I1** - **I8**) can be tested using the Input Status (**IS**) command. The **IS** command will respond with the status of all the inputs on the I/O connector. Refer to the figures titled *Typical 3-wire Sensor Input Connection* and *Typical I/O Connections* for a typical input circuit. The format will be as follows:

```
> 1IS
*ØØØØ_ØØØØ_ØØØØ
```

The **Ø** in the response string represent the state of the different input functions. From left to right, the inputs are as follows:

CW LIMIT
CCW LIMIT
HOME INPUT
REGISTRATION INPUT

INPUT 1
INPUT 2
INPUT 3
INPUT 4

INPUT 5
INPUT 6
INPUT 7
INPUT 8

Each input may be tested by energizing the desired input and issuing the **IS** command. The response string should indicate a **1** in the position that corresponds to the input that was energized. Refer to the ***SX Software Reference Guide*** and the **DSA**, **OSA**, **INL** commands for changing the various active levels.

Example

Only the **REG** input is energized and then the **IS** command is issued. The response should be as follows: **`*ØØØ1_ØØØØ_ØØØØ`**

Output Conventions

The outputs on the I/O connector (O1 - O4) are optically isolated open collector darlington transistors. To view the output signal as a voltage, an external pull-up resistor must be used. Energizing an output will cause the transistor to turn on, this will result in a low signal if the output is being viewed as a voltage. If the output is being used as a current node, then energizing an output will cause current to flow.

Output Test

All of the outputs (O1-O4) can be tested using the Immediate Output (**IO**) command. By issuing the following sequence of commands you will be able to verify that the outputs are wired correctly.

```
> 1IO1ØØØ (energizes only O1)
> 1IOØ1ØØ (energizes only O2)
> 1IOØØ1Ø (energizes only O3)
> 1IOØØØ1 (energizes only O4)
```

Fault Output Convention

The fault output will be energized (conducting current), when ever the Indexer thinks that everything is operating correctly. Normally, if the shuts down the amplifier because an amplifier shutdown command (**`ST1`** or **`OFF`**) was issued the fault output would not be de-energized. This situation can be changed by using the **`SSR`** command (fault de-energized upon amplifier shutdown). Refer to the fault output description earlier in this chapter for a listing of the conditions that will cause the fault output to de-energize.

Fault Test

The fault output can be tested by issuing the following sequence of commands. The fault output follows the same conventions as the general-purpose outputs.

Command	Description
`> 1SSR1`	De-energize the fault output upon commanded shutdown
`> 1ST1`	Shutdown Amplifier

This should have resulted in the amplifier being disabled and the fault output being de-energiz ed.

Command	Description
`> 1STØ`	Enable amplifier

This should have resulted in the amplifier being enabled and the fault output being energized.

Motor Test

By issuing the following sequence of commands, you will be able to verify that the motor is connected correctly.

Command	Description
`> A2Ø`	Set acceleration at 20 rps^2
`> V2`	Set velocity at 2 rps
`> MN`	Set move to Normal mode
`> MR11`	Set the motor resolution at 25K
`> D+25ØØØ`	Set distance at 1 revolution CW
`> LD3`	Disable end-of-travel limits
`> G`	Execute the move (Go)

The motor should have turned 1 revolution CW. If the motor moved in the CCW direction, then the motor is not wired to the drive correctly. The motor direction may be changed by reversing the leads connected to the **`A+`** and **`A-`** terminal on the motor connector.

Incremental Encoder Test

By issuing the following sequence of commands, you will be able to verify that the incremental encoder is connected correctly.

Command	Description
`> A2Ø`	Set acceleration at 20 rps^2
`> V2`	Set velocity at 2 rps
`> MN`	Set move to Normal mode
`> MR11`	Set the motor resolution at 25K
`> D+25ØØØ`	Set distance at 1 revolution CW
`> LD3`	Disable end-of-travel limits
`> PZ`	Disable end-of-travel limits
`> G`	Execute the move (Go)
`> 1PR`	Request motor position
`*+ØØØØØ25ØØØ`	Verifies motor moved 25000 steps CW
`> 1PX`	Request encoder position
`*+ØØØØØØ4ØØØ`	Verifies motor moved 4000 steps CW

If the encoder position report responded with the correct number of encoder counts but in the wrong direction (**`*-ØØØØØØ4ØØØ`**), the encoder is connected backwards. This is easily remedied by switching channel A and channel B. **It is very important for closed-loop operation that the motor direction and encoder direction match.**

Absolute Encoder Test

First confirm that the absolute encoder version of the SX is what you have. Reset the unit with the **Z** command. Then issue the **1RVV** command to report back encoder interface status. By issuing the following sequence of commands, you will be able to verify that the absolute encoder is connected correctly.

Command	Description
> A2Ø	Set acceleration at 20 rps^2
> V2	Set velocity at 2 rps
> FSM1	Set to Absolute encoder mode
> ER16384	Set encoder resolution to 16384
> MN	Set motor to Normal mode
> MR11	Set the motor resolution to 25K
> D+25ØØØ	Set distance to 1 revolution CW
> LD3	Disable end-of-travel limits
> PZ	Set absolute position to zero
> G	Execute the move (Go)
> 1PR	Request motor position
***+ØØØØØ25ØØØ**	Verifies motor moved 25000 steps CW
> 1PX	Request encoder position
***+ØØØØØ16384**	Verifies encoder position

If the encoder position report responded with the correct encoder count, but wrong direction (*-ØØØØ16384), the absolute encoder is counting backwards. Flip the direction DIP switch inside the AR-C Decoder Box (refer to the ***AR-C User Guide***). **It is very important for closed-loop operation that the motor direction and encoder direction match.**

Drive Mounting

You can mount the SX in either a minimum depth or width configuration, depending on the position of the mounting clips (refer to the following figure). **Use only 6-32 X 3/8" screws to attach the mounting clips to the drive. Longer screws may damage the drive.**

WARNING

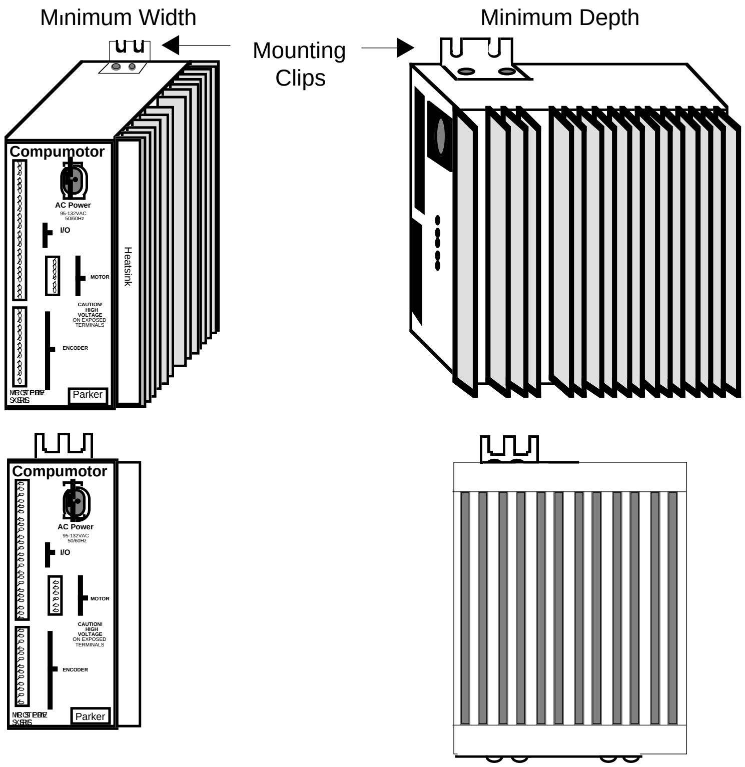
Use 6-32 X 1/4" screws to mount the switch cover only. Longer screws will damage the internal printed circuit board.

Minimum Width

Two clips are attached to the side of the drive away from the power connectors for minimum width. This provides the maximum amount of panel space. **The drive is shipped in this configuration**.

Minimum Depth

You can move the clips from the minimum-width position to the side opposite the heatsink to create a minimum-depth configuration. Three clips are used in the minimum-depth position—one on top and two on the bottom.



Mounting the Drive

WARNING

If you mount the SX in the minimum-depth configuration, the screws (6-32) used to attach the clips to the SX/SXF must not be longer than 3/8". Longer screws will damage the internal printed circuit board.

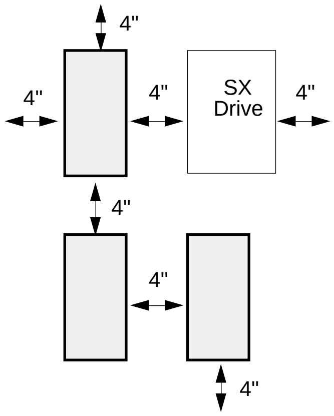
Panel Layout

If you mount the SX in an enclosure, observe the following guidelines:

① The vertical and horizontal clearance between the SX and other equipment, or the top or bottom of the enclosure, should be no less than 4".

② Do not mount large, heat-producing equipment directly beneath the SX.

③ Do not mount the SX directly below an Indexer (the SX produces more heat than an Indexer). Fan cooling may be necessary if air flow is not adequate.



Panel Layout Guidelines

Motor Mounting

Rotary stepper motors should be mounted using flange bolts and positioned with the centering flange on the front face. Foot-mount or cradle configurations are not recommended because the torque of the motor is not evenly distributed around the motor case and they offer poor registration. 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.

WARNING

Improper mounting can compromise system performance and jeopardize personal safety.

The motors used with the SX can produce very large torques. These motors can also produce 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 profiles, can produce low-frequency vibrations in the mounting structure. These vibrations can also 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. **Do not attach the load to the motor yet. Coupling the load to the motor is discussed later in this chapter.**

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 (it will cause a significant loss of torque). Modifying or altering the motor or shaft may void the warranty.

Attaching the Load

This section discusses the main factors involved when attaching the load to the motor. The following three types of misalignments can exist in any combination.

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.

Couplings

The motor and load should be aligned as accurately as possible. Any misalignment may degrade your system's performance. 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 end-play. 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 because it will allow too much freedom and the shaft will rotate eccentrically; this will cause large vibrations and immediate failure.*

Double-flex Coupling

Use a double-flexed coupling whenever two shafts are joined that are fixed in the radial and angular direction (angular misalignment). *Do not use a single-flex coupling with a parallel misalignment; this will bend the shafts, causing excessive bearing loads and premature failure.*

Rigid Coupling

Rigid couplings are generally not recommended. They should be used only if the motor is on some form of floating mounts, which allow for alignment compensation.

☛ Coupling Manufacturers

HELI-CAL
901 McCoy Lane
P.O. Box 1460
Santa Maria, CA 93456
(805) 928-3851

ROCOM CORP
5957 Engineer Drive
Huntington Beach, CA 92649
(714) 891-9922

For unusual motor installations contact a Compumotor Applications Engineer for assistance.

Tuning

This section contains the issues and concerns that you should be aware of as you tune and develop your system.

❏ Resonance

❏ Mid-Range Instability

Resonance

Resonance exists in all stepper motors and is a function of the motor's mechanical construction. It can cause the motor to stall at low speeds. Most full step motor controllers *jump* the motor to a set minimum starting speed that is greater than the resonance region. The SX's microstepping capability allows you to operate a motor smoothly at low speeds.

Motors that will not accelerate past 1 rps may be stalling due to resonance. You can add inertia to the motor shaft by putting a drill chuck on the shaft. The drill chuck may provide enough inertia to test the motor when it is ***not loaded.*** *In extreme cases, a viscous damper may also be needed. Refer to Chapter 6, Hardware Reference for the maximum inertia ratings for your motor.*

The SX is factory tuned to minimize resonance problems. If you are running the SX at motor resolutions of 200 or 400 steps/rev, you may need to implement the Start/Stop Velocity (**VS**) command.

Mid-Range Instability

All step motors are subject to mid-range instability, also referred to as parametric oscillations. These oscillations may stall the motor at speeds from 6 to 16 rps.

Tuning Procedures

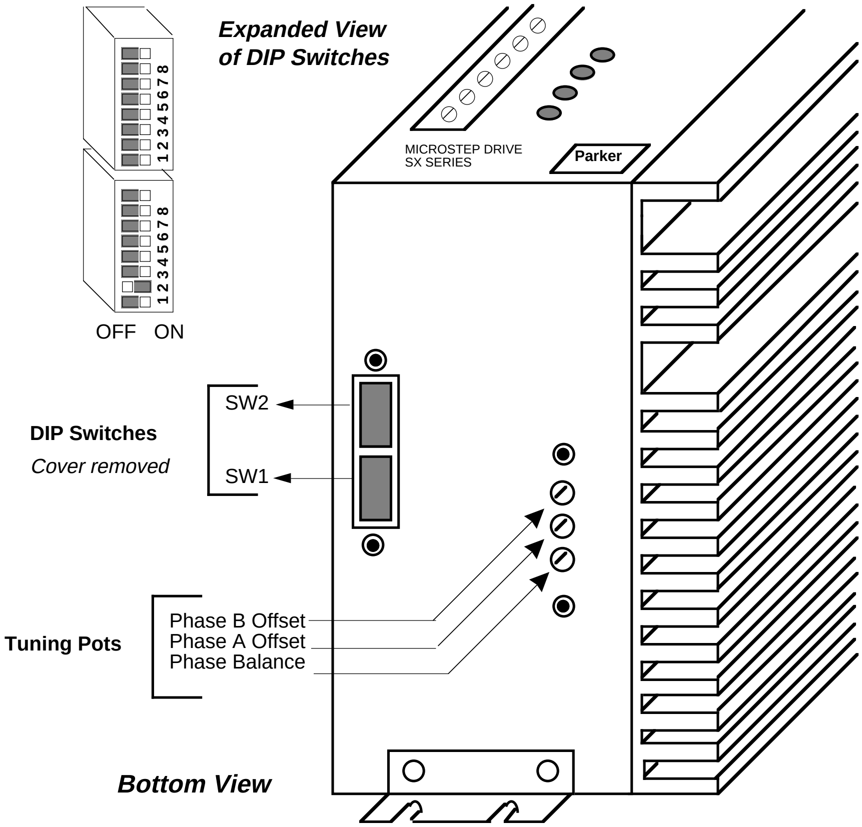
You can tune the SX to minimize resonance and optimize smoothness by adjusting the small potentiometers (pots) on the bottom of the unit. The following figure shows the location of the potentiometers and their functions. A description of each function is listed below.

❏ Phase A Offset: Adjusts the DC offset of the phase current for Phase A.

❏ Phase B Offset: Adjust the DC offset of the phase current or Phase B.

❏ Phase Balance: Adjust the phase current of Phase B to approximately ±10% of Phase A.

It is not usually necessary to adjust these pots, tuning is done at the factory. Adjustments should be made only if the load inertia is greater than 2-3 times that of the rotor inertia. For best results, the drive and motor should be on, connected to the load, and warmed up for 30 minutes prior to tuning.



Location of Tuning Pots

Gauging Motor Resonance

There are several methods that you can use to determine the level of motor resonance in your system.

☛ Tachometer Method

Use an oscilloscope to gauge the output of a tachometer attached to the motor shaft. The tachometer will output a DC voltage, proportional to speed. This voltage will oscillate around an average voltage when the motor is resonating. The amplitude of this oscillation will be at its maximum when you run the motor at its ***resonance speed***. The goal of this tuning method is to tune the motor for its lowest oscillation amplitude.

☛ Sounding Board Method

You can practice your tuning skills with an unloaded motor placed on a sounding board or table. When you command a velocity that is near the motor's ***resonance speed***, the phenomenon will cause an audible vibration. The goal of this tuning method is to tune the motor for the least amount of vibration.

☛ Stethoscope Method

When you tune your motor under loaded conditions, you can hear the audible vibration caused by the motor's natural frequency by placing the tip of a screw driver against the motor casing and placing the handle of the screw driver close to your ear (as you would a stethoscope). You will also be able to hear the different magnitudes of vibration caused by the motor's natural frequency. The goal of this tuning method is to tune the motor for the least amount of vibration.

☛ Touch Method

After you have had some experience with tuning, you should be able to locate the motor's ***resonance speed*** by placing your fingertips on the motor shaft and adjusting the motor's velocity. Once the ***resonance speed*** is located, you can tune the motor for maximum smoothness in the same way.

Tuning the Drive to the Motor

Please note that system tuning has been done at the factory. To tune the drive, it is suggested that you first return the potentiometers to their center positions. To tune the SX, follow the directions below:

① Locate the motor's natural resonant frequency.

A table of resonant frequencies for unloaded Compumotor motors is shown below.

Motor Size	1st Tuning Speed	2nd Tuning Speed
SX57-51	5.15 rps	2.57 rps
SX57-83	3.92 rps	1.96 rps
SX 57-102	3.75 rps	1.88 rps
SX83-62	3.00 rps	1.50 rps
SX83-93	2.97 rps	1.48 rps
SX83-135	2.95 rps	1.47 rps
SX106-178	2.11 rps	1.06 rps
SX106-1250	2.07 rps	1.04 rps
SX106-205	2.67 rps	1.34 rps

By varying the speed slightly from the values given in the table above, locate the speed of worst resonance.

Adjust the Phase A and Phase B offset potentiometers (located on the bottom of the drive), for best smoothness. Iterative tuning is recommended. That is, adjust Phase A Offset, then B, then C, etc., until no further improvement in smoothness is noted.

② Decrease the motor's velocity to half the value used in step 1.

Adjust the Phase Balance Potentiometer for best smoothness.

Optional Fine Tuning:

③ Once again, decrease the motor's velocity by half.

Adjust the Waveform Symmetry with the Motor Waveform (`MW`) command, for best smoothness.

Repeat the above procedure until no further improvement in motor smoothness is noted.

Motor Waveforms

Step motor manufacturers make every effort to design step motors that work well with sinusoidal current waveforms. However, due to physical limitations, most motors operate best with a current waveform other than a pure sine wave.

The purpose of adjusting motor current waveforms is to cause the step motor to move with equal step sizes as the current waveforms are sequenced through the motor. This *waveform matching* will also help the motor run more smoothly. The motor waveform can be changed with the **MW** command (refer to the ***SX Software Reference Guide*** for the command syntax).

Motor waveforms are usually adjusted after the drive has been tuned to its motor. If you do not have precision measurement equipment, you may select the correct motor waveform with one of the three methods described previously in this chapter (Tachometer Method, Sounding Board Method, Stethoscope Method, and Touch Method). These empirical methods generally yield acceptable results.

Anti-Resonance

As of serial number 95Ø626XXXXX, the S series contains an anti-resonance circuit designed to reduce mid-frequency resonance in your system. The circuit acts on the power being supplied to the motor, which varies greatly when resonance occurs. It will inject a signal opposite to that caused by the resonance in order to apply torque against the resonance and cancel it.

This circuit does not guarantee resonance won't cause problems in your system if it is bad enough. It simply works to reduce the affect mid-frequency resonance has on your system. The effect of this will be smoother motion over the mid-range frequencies and better use of the torque available at those speeds.

CHAPTER ④

Application Design

Chapter Objectives

The information in this chapter will enable you to:

- ❏ Recognize and understand important considerations that must be addressed before you implement your application
- ❏ Understand the capabilities of the system
- ❏ Use examples to help you develop your application

Motion Profile Application Considerations

This section contains information that you should consider and evaluate when designing and developing your system.

Positional Accuracy vs. Repeatability

Some applications require high absolute accuracy. Others require repeatability. You should clearly define and distinguish these two concepts when you address the issue of system performance.

If the positioning system is taken to a fixed place and the coordinates of that point are recorded. The only concern is how well the system repeats when you command it to go back to the same point. For many systems, what is meant by accuracy is really repeatability. Repeatability measures how accurately you can repeat moves to the same position.

Accuracy, on the other hand, is the error in finding a random position. For example, suppose the job is to measure the size of an object. The size of the object is determined by moving the positioning system to a point on the object and using the move distance required to get there as the measurement value. In this situation, basic system accuracy is important. The system accuracy must be better than the tolerance on the measurement that is desired. Consult the technical data section of the Compumotor Catalog for more information on accuracy and repeatability.

Move Times—Calculated vs Actual

You can calculate the time it takes to complete a move by using the acceleration, velocity, and distance values that you define. However, you should not assume that this value is the actual move time. There is calculation delay and motor settling time that makes your move longer. You should also expect some time for the motor to settle into position. The SX has minimal calculation-delay time associated with a Go (**G**) command. This delay can be as low as 500 µs. The SX has an internal timer that allows you to monitor the elapsed time of your move. The response of the **TM** command shows you the previous move's execution time.

Preset Mode Moves

A preset move is one in which the distance is specified in motor steps. You can select preset moves by putting the SX into Normal mode with the Mode Normal (**MN**) command. Preset moves allow you to position the motor in relation to the motor's previous stopped position (incremental moves) or in relation to a defined zero reference position (absolute moves). You can select incremental moves with the Mode Position Incremental (**MPI**) command. You can select absolute moves with the Mode Position Absolute (**MPA**) command. At any time, you can request the state in which the SX is configured by issuing the **DR** command.

Incremental Mode Preset Moves

When you are in Incremental mode (**MPI**) and Preset mode (**MN**), the motor moves the specified distance from its current position. The preset distance is specified with the **D** command. You can specify the direction with the optional sign (**D+8ØØØ** or **D-8ØØØ**), or you can define it separately with the Change Direction (**H+** or **H-**) command. If no sign is specified with the **D** command it defaults to positive.

Command	Description
> LD3	Disables CW and CCW limits (**not needed if limits are installed**)
> MPI	Sets unit to Incremental Position Mode
> MN	Places the SX in Preset mode
> PZ	Zeroes the position counter
> A25	Sets acceleration to 25 rps^2
> AD25	Sets deceleration to 25 rps^2
> V5	Sets velocity to 5 rps
> D8ØØØ	Sets distance to 8,000 steps
> G	Executes the move (Go)
> D12ØØØ	Sets distance to 12,000 steps
> G	Executes the move (Go)
> 1PR	Reports the setpoint (commanded) position Response: ***+ØØØØØ2ØØØØ**

Absolute Mode Preset Moves

A preset move in the Absolute mode (**MPA**) and Preset mode (**MN**) moves the motor to the distance in an absolute coordinate system that you specify relative to an absolute zero position. You can set the absolute position to zero with the Position Zero (**PZ**) command or by cycling the power to the Indexer. The absolute zero position is the initial power-up position.

The direction of an absolute preset move depends upon the motor position at the beginning of the move and the position you command it to move to. If the motor is at absolute position +12,800, and you instruct the motor to move to position +5,000, the motor will move 7,800 steps in the negative direction to reach the absolute position of +5,000.

The SX powers up in Incremental mode. When you issue the Mode Position Absolute (**MPA**) command, it sets the mode to absolute. When you issue the Mode Position Incremental (**MPI**) command the unit switches to Incremental mode. The SX retains the absolute position, even while the unit is in the Incremental mode. You can use the Position Report (**PR**) command to read the absolute position.

In the following example, the motor performs the same commands as the incremental position example. In this case, the **PR** command will report a different position because it is working in an absolute coordinate system.

Command	Description
> LD3	Disables CW and CCW limits (**not necessary if limits are installed**)
> MPA	Sets unit to Absolute Position mode
> PZ	Zeroes the position counter
> A25	Sets acceleration to 25 rps^2
> AD25	Sets deceleration to 25 rps^2
> V5	Sets velocity to 5 rps
> D8ØØØ	Sets distance to 8,000 steps
> G	Executes the move (Go)
> D12ØØØ	Sets distance to 12,000 steps
> G	Initiates motion
> 1PR	Reports the setpoint (commanded) position Response: *+**ØØØØØ12ØØØ**

The motor will move to absolute position 8,000. The second move is 4,000 more steps to the absolute position of 12,000 steps. The **PR** command reports a setpoint (commanded position) of 12,000 steps.

Continuous Mode Moves

The Continuous mode (**MC**) command accelerates the motor to the velocity that you last specified with the Velocity (**V**) command. The motor continues to move at the specified velocity until you issue the Stop (**S**) or Kill (**K**) command or specify a velocity change. To interactively change velocity while the motor is moving, use the instantaneous velocity command (**IV**). To change velocity on-the-fly in a sequence, use the Motion Profiling mode (**MPP**).

In Motion Profiling mode, all buffered commands are executed immediately—therefore you only have to enter the **V** command to change the velocity. No **G** is needed following the **V**. Continuous mode is useful for applications that require constant movement of the load, and the motion is not based on distance but is based on internal variables or external inputs, or when the motor must be synchronized to external events such as trigger input signals. In this example, velocity is changed after a time delay of about 1 second.

Command	Description
> PS	Pauses motion until a **C** command is reached
MPP	Places the SX in the **MPP** mode
IN1A	Sets up **I1** (Input 1) as trigger bit 1
IN2A	Sets up **I2** (Input 2) as trigger bit 2
LD3	Disables CW and CCW limits (**not necessary if limits are installed**)
MC	Sets unit to the Continuous mode
A25	Sets acceleration to 25 rps^2
AD25	Sets deceleration to 25 rps^2
V1	Sets velocity to 1 rps
G	Executes the move (Go)
T1	Waits 1 second after the move is started
V5	Sets velocity to 5 rps
TR1Ø	Waits for trigger bit 1 to go on and bit 2 to go off
STOP	Stops the motor
C	Continues execution of commands

These commands cause the SX to run in Continuous mode. The motor starts accelerating, waits for 1 second, changes velocity to 5 rps, waits for you to turn **I1** (input 1) on and turn **I2** (input 2) off, and then stops. *The* **VØ** *and* **STOP** *commands stop the motor (the* **S** *command is not a buffered command and cannot be used in this situation, unless you wish to halt the operation in the middle of the program).* The **DIN** command (an immediate command) simulates the state you want the inputs to be in. In the example above, you could simulate the activation of the trigger state without physically toggling the inputs, by using the **DIN** command as follows.

```
> DINEEEE1ØEEEEEE
```

E means *do not affect the input.* A 1 makes the input one, a 0 makes the input zero. Each **E**,1, or 0 represents an input bit. There are 12 inputs. The 1 and Ø in this example correspond to **I1** and **I2** on the front panel. The first 4 **E'**s correspond to CCW, CW Home limit, and Registration Input.

Closed Loop Operation

As of January 1, 1995, the SX/SXF will no longer have absolute encoder interface capability as a standard feature. The standard SX/SXF will not be compatible with the AR-C absolute encoder, but rather the SX/SXF absolute encoder interface will be an option to the standard system. For help in determining whether or not your SX/SXF has the absolute encoder interface, see the RVV command in the ***SX Software Reference Guide.***

This section explains the closed-loop operation of the SX. Closed-loop moves use external sensors to provide position verification signals. Closed-loop operation provides the SX with the ability to detect a system stall or to adjust the load position to compensate for the system's mechanical slop.

The standard SX-A can interface only to an incremental encoder. The SX-A can interface to an incremental encoder or a Compumotor absolute encoder (AR-C). The encoder may be used as a means of creating a closed-loop system or as an independent means of verifying motor position. The following functions are added to a system when an encoder is used:

❏ Encoder referenced positioning
❏ Encoder position correction
❏ Motor stall detection
❏ Higher accuracy homing function

To implement the closed-loop functions, you must connect an incremental encoder or a Compumotor AR-C absolute encoder to the SX. The SX can supply up to 250mA/+5VDC to power the incremental encoder (the AR-C provides it's own power). When you use incremental encoders with single-ended outputs, do not connect channels A-,B-, and, Z- to the SX's encoder connector. Refer to *Chapter 3, Installation* for details on wiring encoders to the SX.

Selecting Encoder Resolution Values: Incremental

The number of encoder steps that the SX system recognizes is equal to four times the number of encoder lines. For example, a 1000-line encoder mounted directly on the motor will generate 4000 encoder steps per revolution of the motor shaft. The SX reads in a quadrature signal.

Selecting Encoder Resolution Values: Absolute

The number of encoder steps that the SX system recognizes is equal to the specified number of absolute encoder steps that are selected at the AR-C decoder box. The absolute encoder is capable of providing either 16384, 8192, 4096, or 2048 steps per revolution.

Motor-to-Encoder Ratios

A minimum of three motor steps per encoder step is required for successful operation of the Position Maintenance function. If a 1000-line incremental encoder is mounted directly to the motor shaft, the motor must have a resolution of 12,000 steps/rev or higher. Ratios above three motor steps per encoder step ensure stability of the position maintenance correction function. **When operating in Encoder Step mode, system accuracy depends on the accuracy of the encoder and not on the accuracy of the drive and motor combination in Open-Loop mode.**

If you install a reducer (gear box) between the motor shaft and the encoder, the number of encoder steps that the Indexer receives is equivalent to the number of encoder steps divided by the encoder gear ratio.

For example, using a 12,800 steps/rev motor, a 1,000-line encoder, and a 10:1 reducer, the ratio of motor revolutions to encoder steps would be changed as described in the following table.

Parameter	1:1 Ratio	10:1 Ratio
Number of encoder steps recognized by the SX (per motor rev)	4,000	400
Required ratio of motor revs to encoder steps	1/4,000	1/400
Motor-to-encoder step ratio	3.2/1	32/1

Setting Encoder Resolution

Incremental Encoder

Encoders are available in a wide range of resolutions. Using the Encoder Resolution (**ER**) command, you must specify the resolution of the encoder that is connected to the SX.

Absolute Encoder

Compumotor offers an absolute encoder for use with the SX-A system. The AR-C absolute encoder is capable of providing one of four resolutions (16384, 8192, 4096, or 2048 steps/rev). The encoder resolutions are selected using DIP switches inside of the AR-C decoder box.

Using the following example as a guide, you can specify the encoder's resolution to the with the Encoder Resolution (**ER**) command.

Commmand	Description
> ER4000	Sets encoder's post-quadrature resolution to 4000 steps/ rev (1,000 lines)
> ER8192	Sets absolute encoder resolution to 8192 steps/rev

The number entered with the ER command is the number of encoder steps that will be counted when the motor has moved 1 revolution. If a 1000-line encoder is directly mounted to a 25000 step motor, there will be 4000 encoder steps for 1 motor revolution.

Encoder Step Mode

The Indexer can perform moves in either motor steps or encoder steps. In Motor Step mode (**FSBØ**), the distance (**D**) command defines moves in motor steps. In Encoder Step mode (**FSB1**), the distance command defines moves in encoder steps. When the AR-C is used, the Enable Absolute Encoder command (**FSM1**) must be used prior to enabling the Encoder Step mode (**FSB1**). You must set up the Indexer for the correct encoder resolution. The Encoder Resolution (**ER**) command defines the encoder resolution. The sample move below gives examples of incremental and absolute encoder step moves.

Incremental Encoder

Command	Description
> LD3	Disables CW and CCW limits (not necessary if limits are installed)
> MN	Sets motor to Normal mode
> ER4000	Sets encoder resolution to 4000 counts/rev
> FSB1	Sets move to Encoder Step mode
> A10	Sets acceleration to 10 rps^2
> V5	Sets velocity to 5 rps
> D4000	Sets distance to 4000 encoder steps
> G	Executes the move (Go)

The motor will turn CW until 4000 encoder steps (1 rev) are received. If this move does not work, refer to *Chapter 7 , Maintenance & Troubleshooting*.

If the encoder is accidentally disconnected during the execution of this move, the motor will continue to move CW indefinitely. This is because the system continues to move, while waiting for the remaining encoder pulses. To prevent this, use the Stop-On-Stall (**FSD1**) command.

If this example causes a very slow, creeping motion, the encoder is not wired properly or the encoder resolution being seen is too low. Confirm the encoder count is increasing with CW motion using the PX command.

Absolute Encoder

Command	Description
> LD3	Disables CW and CCW limits (not necessary if limits are installed)
> MN	Sets motor to Normal mode
> MR15	Sets motor resolution to 50000 steps/rev
> FSM1	Enable absolute encoder
> 1RSE	Check for AR-C problems
> ER16384	Select (default) AR-C encoder resolution
> FSB1	Sets move to encoder step mode
> A10	Sets acceleration to 10 rps^2
> V5	Sets velocity to 5 rps
> D16384	Sets distance to 16384 encoder steps
> G	Executes the move (Go)

The motor will turn CW until 16,384 encoder steps (1 rev) are received. If this move does not work, refer to *Chapter 7, Maintenance & Troubleshooting.*

If the encoder is accidentally disconnected during the execution of this move, the motor will stop immediately as if stop-on-stall was enabled and the motor has stalled. **1RSE** will respond with ***ABSOLUTE_ENCODER_FAULT** and **1R** will respond with ***S** (attention required). To clear this fault condition, execute the **ON** command.

If this example causes a very slow, creeping motion, the encoder is not wired properly or the encoder resolution being seen is too low. Confirm the encoder count is increasing with CW motion using the PX command.

If you need to power up the SX and an absolute encoder simultaneously, enter the following commands into sequence 100.

Command	Description
> FSM0	Disable absolute encoder
> T1	Pause for 1 second
> FSM1	Enable absolute encoder

This gives the absolute encoder time to fully power-up before the SX checks the interface.

Position Maintenance

The Position Maintenance (**FSC**) command enables and disables the position maintenance function. You must enable Position Maintenance (**FSC1**) to activate closed-loop position correction and ensure that encoder step moves are positioned within the specified Deadband (**DW**) of the commanded encoder position. Enabling position maintenance will cause the Indexer to servo the motor until the correct encoder position is reached. This occurs at the end of a move (if the final position is incorrect) or any time the Indexer senses a change in position while the motor is at Ø velocity. To enable position maintenance, you must have an encoder connected and the Indexer set in Encoder Step mode (**FSB1**).

If a stall occurs during position maintenance, the position error is set equal to zero and the motor does not move. If you disable Stop-on-Stall (**FSDØ**), the SX will continue to try to move the motor the commanded distance.

☛ Note Position Maintenance is not cancelled by a Stop (**S**) or a Kill (**K**) command. Use the **FSDØ** or **OFF** commands.

The SX's Position Maintenance mode corrects for final positional accuracy after a move is completed. If a 4,000 encoder step move is commanded, the motor will attempt to go 25,000 motor steps (or 1 rev). After the move is complete, the system checks the encoder to ensure the encoder reads 4,000 steps or 1 revolution. If it is not at 4,000, it will correct the motor's final position by commanding the motor to move until the encoder reads 4,000 steps. You can use the following commands to set up a position maintenance application.

Command	Description
DPA	Displays actual motor position indicated by the encoder—displayed in encoder steps.
PR	Displays number of steps motor was commanded to go—motor or encoder steps.
PX	Displays actual motor position indicated by the encoder—displayed in encoder steps.
DPE	**D**isplays the **P**osition **E**rror between where the motor was commanded to go (setpoint) and where the encoder indicates the motor is(actual position). In the Motor step mode, the error is displayed in motor steps. In the encoder step mode, the error is displayed in encoder steps.
CPE	**C**onfigures the maximum **P**osition **E**rror allowed. The actual motor position that is off from the setpoint or commanded position. This determines when a stall is detected (FSD1)
CPG	**C**onfigure **P**osition **G**ain. This is the percentage gain that is applied to the position error to determine a correction velocity.
CPM	Maximum gain value for which the **CPG** determines the % of to use for correction.
MV	The maximum velocity that the SX can command when correcting for position while in Position Maintenance mode.
FSC	Enters and exits Position Maintenance mode.
FSD	Enters and exits the **S**tall **D**etect mode.
DW	This is the deadband window, when the error is greater than this number of steps, the SX will perform position maintenance. A wider window Prevents dither.

In Position Maintenance mode, the SX performs an end-of-move correction. It corrects for any position error that is detected between the commanded position and the encoder position which is greater than the deadband window. The SX has a 1-ms update rate. Every ms it will correct for any position error above the deadband window. It will continue to send pulses to the motor until the motor is in position according to the encoder. Position maintenance can be used only in Encoder Step mode.

In the encoder mode, the SX determines the motor position from the encoder. A move is considered complete when the encoder position reaches the commanded position. When a step motor is accelerated or decelerated, it will ring about the commanded move trajectory. The higher the acceleration, the larger the amplitude of the ringing. If the amplitude of the ringing is larger than one motor pole, the motor and drive will lose synchronism and a stall will result. Microstepping reduces the violence of this ringing, but does not totally eliminate it.

Because of this ringing, it is possible for the motor to overshoot or undershoot its commanded goal. This occurs because at the moment the encoder indicates the correct terminal position and the SX stops commanding motion, the motor may still ring back to a different resting position. The magnitude of this final error can be anywhere from a few steps to 30-50 steps depending on the acceleration. The purpose of Position Maintenance is to correct this end of move error.

Position maintenance is also very useful in correcting for system errors if the encoder is mounted on the load rather than directly behind the motor. Mounting an encoder on the load is preferred when the system mechanics allow.

If the Position Maintenance mode (**FSC1**) is on, the motor will servo to the commanded position after the move is completed. With both Stall Detect (**FSD1**) and Position Maintenance (**FSC1**) modes activated, the SX will apply the position error to the following algorithm: If **DPE** (position error) > **CPE** (maximum allowable position error), a stall condition exists. Stall errors occur and the step and direction output is turned off. *If **DPE**(position error) > **DW** (deadband window), position maintenance occurs, unless a stall is detected.* Stalls are checked for each sample period, and position maintanance is checked each sample period at the end of a commanded move.

Correction velocity is determined by the following equation:

Velocity = **CPG** (Proportional gain percentage) • **CPM** (Proportional gain maximum) • **DPE** (position error)

If this is less than the **MV** value (maximum correction velocity), it is the commanded correction velocity. If it is greater than the **MV** value, the **MV** value is the commanded position maintenance correction velocity.

If **DPE** is < **DW**, no correction occurs. **If oscillation occurs after the move, the Position Maintenance gains may be set too high.**

Example

Perform the following steps to make a move with stall detect and position maintenance activated:

Step ① Type the following commands:

Command	Description
> 1CPE2ØØØ	Maximum position error is 2,000 motor steps
> 1CPG5Ø	Sets position gain to 50% of maximum gain (CPM)
> 1CPM6Ø	Maximum position gain is 60
> 1MV1Ø	Sets maximum correction velocity to 10 rps
> 1DW5	Sets deadband window to 5 steps
> 1FSD1	Invokes Stall Detect mode
> 1FSB1	Set system to Encoder Step mode
> 1FSC1	Invokes Position Maintenance mode

This enables the Position Maintenance mode. The above selected parameters may make the motor unstable when unloaded.

Step ② Type the following commands to make a move:

Command	Description
> A1ØØ	Sets the acceleration to 100 rps^2
> AD1ØØ	Sets the deceleration to 100 rps^2
> V1Ø	Sets velocity to 10 rps
> D1ØØØØØ	The motor is set to move 100,000 encodersteps
> G	Initiates motion

Step ③ Use the following commands to observe the position error and actual position.

Command	Description
> 1DPA	Display the actual position
> 1DPE	Display the position error

The motor should move to the commanded position, +100,000 encoder steps.

Stop-On-Stall

You can enable the Stop-on-Stall function with the FSD1 command. The move will terminate, without any delay, as soon as a stall is detected. This function works either in Motor Step or Encoder Step mode and is independent of position maintenance.

CAUTION

Disabling the Stop-on-Stall function with the FSDØ command will allow the SX to attempt to finish the move profile regardless of a stall detection, even if the load is jammed. This can potentially damage user equipment.

The Stop-on-Stall feature depends on the maximum allowed position error (set with the Configure Position Error [**CPE**] command). The factory default setting for maximum position error is 4000 encoder steps.

Command	Description
> CPE1ØØ	Sets maximum position error to 100 steps
> ER4ØØØ	Sets encoder resolution to 4,000 steps/rev
> FSD1	Enables Stop on Stall mode

While at rest the position error will normally never exceed 30 to 50 steps. When operating at high speeds, the difference between the command and actual position (position error) can easily exceed 50 steps. This is because the motor has already been moving for 1 ms before the position error is calculated. For example, at **MR11** (25,000 steps/rev) the position error can be 1,250 steps at 50 rps. This number could be larger by a ringing motor or motor windage.

Stall detection does not occur until the error exceeds the maximum position error (set with the **CPE** command). Consequently, if the commanded motor position and the encoder position differ by 100 motor steps, the SX will detect a stall and stop the motor immediately.

Output-On-Stall

You can define one of the outputs to act as an output-on-stall. By defining the output type as **L** it becomes an output-on-stall. Whenever a stall condition occurs the output will be activated. You can be in either Motor Step mode or Encoder Step mode, and you do not have to be in Position Maintenance mode. By selecting an output as an output-on-stall you are not causing the motor to stop on a stall. The motor will not stop on a stall unless you enable it with **FSD1**.

Command	Description
> MN	Sets the system in the Preset mode
> FSB1	Enables Encoder mode
> CPE3Ø	Selects maximum position error of 30 motor steps
> OUT1L	Enables stall detect function on output #1
> FSD1	Stops motor if a stall is detected
> A2	Sets acceleration to 2 rps^2
> V.1	Sets velocity to 0.1 rps
> D128ØØ	Sets distance to 12,800 encoder steps
> G	Execute the move (Go)

While the motor is moving, you can cause a stall by holding the shaft. If you can not manually stall the motor, you can simulate a stall by carefully disconnecting the +5V encoder lead from pin #1 on the SX encoder connector. When the stall occurs, output #1 is turned on and the motor stops (this signals you that the motor has stalled).

Fault Sequence Execution with Stall Detect

If the user has an error or kill sequence defined (**SFKn**), a stall will cause the program to execute the SFX sequence. This can be very useful in alerting machine operators to error conditions or mechanical difficulties.

Mechanical Resonance

Resonance, a characteristic of all stepper motors, can cause the motor to stall at low speeds. Most full-step motor controllers *jump* the motor to a set minimum starting speed to avoid this resonance region. This causes poor performance below 1 rps. In nearly all cases, the stepping features of the SX will overcome these problems. However, in some cases the drive will need to be optimized with some simple adjustments to overcome resonance.

Resonance occurs at speeds which approach the natural frequency of those speeds. It causes the motor to vibrate at these speeds. The speed at which fundamental resonance occurs is typically between 0.3 and 0.8 rps and is highest for small motors and lowest for large motors.

Motors that will not accelerate past 1 rps may be stalling due to resonance. The resonance point may be lowered to some extent by adding inertia to the motor shaft. This may be accomplished by putting a drill chuck on the back shaft. *This technique is applicable only to double-shaft motors with the shaft extending from both ends of the motor.* In extreme cases, you may also need a viscous damper to balance the load. One of the manufacturers of viscous dampers is listed below:

Ferrofluidics Corporation
40 Simon Street
Nashua, NH 03061
(603) 883-9800

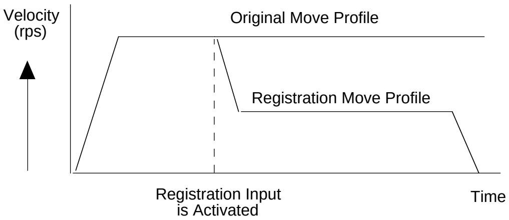
Adjusting the waveform (**`MW`** command) or changing the velocity (**`V`** command) and acceleration (**`A`** command) may also help a resonance problem. Refer to the Tuning Procedures in *Chapter 3, Installation.*

Ringing or Overshoot

The motor's springiness, along with its mass, form an underdamped resonant system that *rings* in response to acceleration transients (such as at the end of a move). Ringing at the end of a move prolongs settling time. *Overshoot* occurs when the motor rotates beyond the actual final position. The actual settling time of a system depends on the motor's stiffness, the mass of the load, and any frictional forces that may be present. By adding a little friction, you can decrease the motor's settling time.

Registration

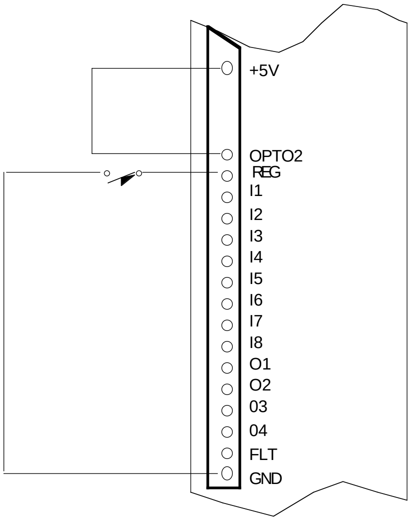
Registration with the SX provides the ability to change the move profile which is being executed to an unrelated move profile defined as a registration move. **This unrelated registration move is executed when an input to the SX transitions from an inactive to active state.** You can only use the input labeled **`REG`** as a registration input. This input will interrupt the SX microprocessor at the highest priority level in the system. The registration input and the motor's current position will be captured within 50 μs. The registration profile will be executed at the next update period. The registration move uses the actual position captured within 50 μs as its reference point.



Registration Move

The interrupt is *edge-sensitive* to the voltage transition (if you have a bouncy switch for the registration input, you must use the debounce (**`TDR`**) command to ensure that false registration interrupts do not occur). With the SX, another registration interrupt can interrupt the current registration move. *When the* ***`REG input`*** *is* **`enabled`**, *it* ***cannot*** *be toggled at a frequency higher than 1 kHz, or once every ms.*

`REG` defines the move when the **`REG`** input is enabled (refer to the figure below). The **`INRE`** command enables registration, **`INRØ`** will disable it (see ***SX Software Reference Guide*** for further explanation.)



Partial View of SX

Note: Use OP2-HV if
(12-24VDC) is used.

To Program

```
> INRE
> REG1,A1Ø,AD5,V5,D25ØØØ
```

The syntax for defining a registration move using the **REG** input on the front panel is:
REG1,A1Ø,AD1Ø,V1Ø,D25ØØØ

The registration move, **REG1**, will be performed when the **REG** input is activated. The acceleration will be 10 rps^2, the deceleration will be 10 rps^2, the velocity will be 10 rps, and the distance traveled will be 25,000 steps from the captured position. The SX motor must be in motion for a registration move to be performed.

Bouncy Registration Inputs

The registration switch may be bouncy or noisy and may take a few ms to settle. With a bouncy switch, each edge appears like a registration interrupt and the registration move is made from the distance position at which the latest edge was detected. The SX allows you to debounce the inputs (with a software command). You can ignore any bouncing transitions from your switch after the initial registration interrupt has occurred. The time in which the interrupts or false edges are ignored is determined by the number you enter for the **TDR** command. **TDR** is the debounce time in ms that you specify so that the inputs cannot cause another registration interrupt until the switch is settled.

As an example, an application needs a registration move. The motor has a resolution of 25,000 steps per revolution. The registration move must turn the motor 1 revolution at 10 rps. If the input does not occur, the move will be a 500,000-step move at 5 rps. The SX is configured as follows:

Command	Description
> INRE	Enables registration input (REG)
> REG1,A1Ø,AD1Ø,V1Ø,D25ØØØ	Defines the registration move
> D5ØØØØØ	Sets distance to 500,000 steps
> V5	Sets velocity to 5 rps
> G	The preset move is initiated

If **REG** is toggled, the corresponding registration move is performed. The **DIN** command will not activate registration input (the registration input is hardware oriented).

Jogging the Motor

In some applications, you may want to move the motor manually. You can configure the SX to allow you to move the motor manually with the Configure Input (**IN**) command. You can define the jogging velocity with the Jog Velocity High (**JVH**) and Jog Velocity Low (**JVL**) commands. You can define three different inputs for jogging: CW Jog input (**IN#J**), CCW Jog Input (**IN#K**), and Jog Speed Select High/Low (**IN#L**). You must also enable the jogging feature with the **OSE1** command. Once you set up these parameters, you can attach a switch to the jog inputs that you defined and perform jogging (**#** represents digits 1 - 8, which you enter). The following example shows how you can define power-up sequence #100 to set up jogging.

Step ① Define a power-up sequence.

Command	Description
> XE1ØØ	Erase sequence #100
> XD1ØØ	Define sequence #100
LD3	Disables the limits (*not needed if the limit switches installed*)
JA25	Set jog acceleration to 25 rps^2
JAD25	Set jog deceleration to 25 rps^2
OSE1	Enables Jog function
JVL.5	Sets low-speed jog velocity to 0.5 rps
JVH5	Sets high-speed jog velocity to 5 rps
IN1J	Sets **I1** as a CW jog input
IN2K	Sets **I2** as a CCW jog input
IN3L	Sets **I3** as a speed-select input
XT	Ends sequence definition

Step ② Reset the SX.

Command	Description
> Z	Reset the SX

Step ③ Turn **I1** input on to move the motor CW at 0.5 rps (until you turn off **I1**).

Step ④ Turn **I2** input on to move the motor CCW at 0.1 rps (until you turn off **I2**).

Step ⑤ Turn **I3** input on to switch to high-speed jogging.

Step ⑥ Repeat steps 3 and 4 to perform high-speed jogging.

Changing **I3** during a current jog will change the velocity on the fly.

Backlash Compensation

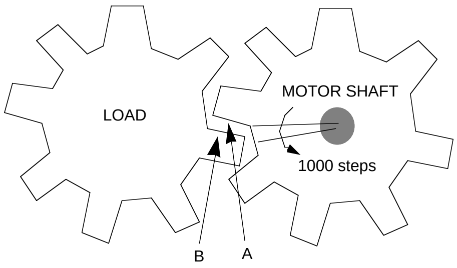
The SX can compensate for backlash in the gearing of your system. You can specify different compensations depending on the direction the motor is moving. You will use the **BL** command for backlash compensation. Refer to the ***SX Software Reference Guide*** for a detailed description of the backlash command. The command syntax is as follows:

Command	Description
> BLn,m	Variable **n** is the amount of CCW steps that should be compensated. Variable **m** is the number of CW steps that will be compensated.

The **BL** *command is only functional in Motor Step mode*. If a CW move is made, an extra **m** steps will be made to account for backlash. The concept is that the load will not begin to move until the motor moves enough to make contact with the gearing (refer to the figure below). This is also true for CCW moves. The backlash compensation may be different in either direction, so you can program them independently. The load will not move until point A contacts point B. This backlash is associated with changing direction. If the desired distance is 1,000 motor steps, the **BL** command will have the following parameters:

Command	Description
> BL1ØØØ,2ØØØ	**1000** is the amount of CCW steps that should be compensated. **2000** is the number of CW steps that will be compensated.

For a 25,000-step CCW move, the motor will actually move 26,000 steps to remove the backlash, but the position counter will only change by 25,000 steps. This is done because the load is expected to move whenever the motor moves. The load should move a certain distance when the motor moves a certain distance. If there is backlash in the system, the load will not move the correct distance when the motor is commanded to move in the opposite direction from its previous move. In this example, the motor was moving CW. The gearing was flush and the teeth were touching. When the direction changes, the teeth must move 1,000 steps before they are again in contact with the load gear. Thus, for the load to move 25,000 steps, the motor will have to move 1,000 steps until the teeth are in contact, then move 25,000 steps so that the load will actually move 25,000 steps. This mode allows you to compensate for the error between the motor and the actual load position. *The compensation only occurs when you change direction.*



Backlash Compensation

Defining a Home Location

The SX's go home function brings your motor to a home reference position. The homing function allows you to back up to a home switch and come to a stop on a specific edge of the switch. You can program the active level of the switch. The homing function also allows you to home to the Z channel of the SX motor's encoder.

Homing to a Switch

There are two homing modes. Normal mode (**OSB0**) is where the motor decelerates from the go home velocity (**GHV**) to a stop, upon the first home input transition, regardless of direction.

Back Up to Home mode (**OSB1**) is where the motor decelerates from the go home velocity to a stop, upon the first home input transition, and either changes direction or continues on at the go home final velocity (**GHF**), looking for another transition of the home switch. This part of the back up to home move is dependant upon the initial homing direction, the final homing direction (**OSG**), and whether the initial deceleration takes you through the other side of the home sensor. **Refer to the included homing diagrams for your needs and set-up parameters.**

A Go Home move is initiated with the **GH** command or with an input defined as a Go Home Input (**INnS**). The **GH** command can also be used to specify the initial homing velocity and direction. These parameters can also be set with the **GHV** command, which does not initiate motion by itself. For example, GH-2 would initiate the go home move in the negative direction at 2 rps. This could also be accomplished with GHV-2 and GH or GHV-2 and a go home input. The **GHA** and **GHAD** commands control the go home acceleration and deceleration. The **OSC** command allows you to adjust the active level of the home input. **Refer to the *SX Software Reference Guide* for further descriptions of all the noted commands.**

Go Home Status Report

The Go Home Status Report (**RG**) can be used to determine if the home move was successful or not.

Homing to a Z Channel

The SX also allows you to home to an encoder's Z channel after locating the home sensor and doing its back up to home move. It is required to see the home input transitions and to be doing a back up to home move in order to home to the Z channel. You do not have to be in encoder mode. Homing to a Z channel is enabled with the **OSD** command and can be configured with the **OSJ** command to search for the Z channel until it is received or through one revolution only. The **RG** command will report back *A, home successful, as long as the home input transitions have been seen. It is independent of whether the Z channel is seen.

Homing and End-of-Travel Limits

Hitting an end-of-travel limit (hardware or software) once during a go home move will cause the motor to reverse direction and seek the desired home transition in the other direction. The home move will then act as though the new direction was the original direction. Hitting the other end of travel limit (hard or soft) in the course of the same go home move will cause the motor to decelerate to a stop. If a home transition has not been found before the two limits are hit, the **`RG`** command will report back *@, home unsuccessful. However, if a home transition has been encountered before the second limit is hit, the **`RG`** command will report back **`*A`**, home successful, so this condition should be avoided for accurate homing. Hitting the second limit after encountering home transitions will not cause a fault (run the fault sequence), since this is considered a successful home.

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 following parameters were used for the diagrams shown below.

Set-up Parameters: **`GHV1 GHF0.2 GHA1 GHAD1`**

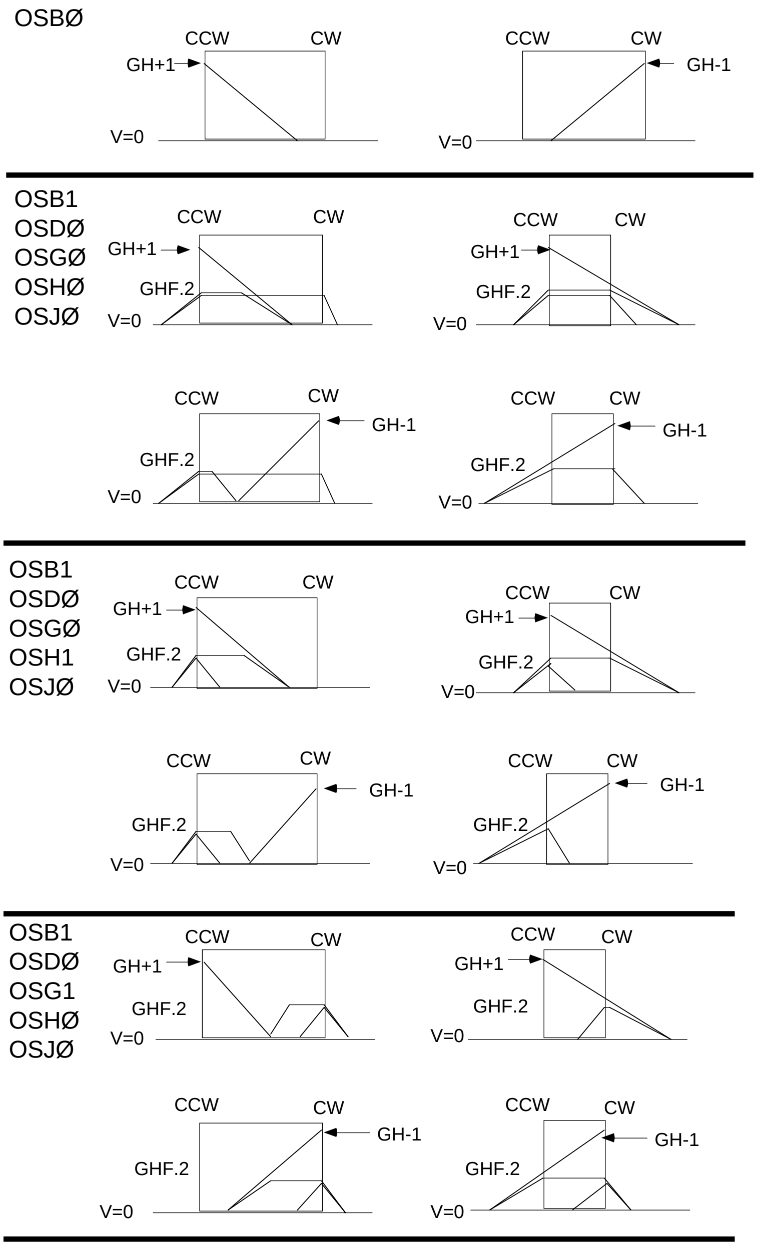
The CW side of the home pulse is the side closest to the CW limit. The CCW side of the home pulse is the side closest to the CCW limit.

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

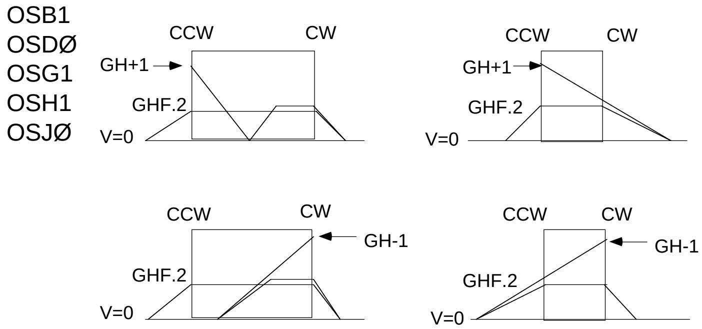
If an end-of-travel limit is hit during the initial homing, refer to the homing diagram for the opposite direction of travel.

☞ Note

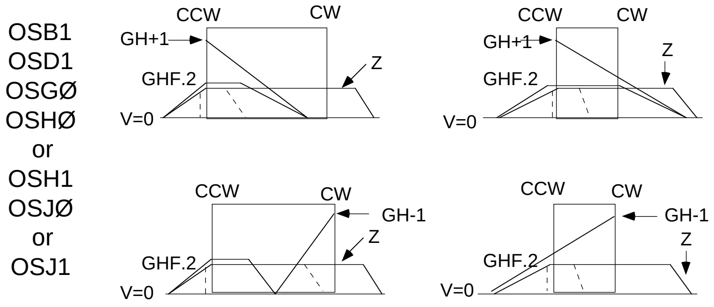
The diagrams are drawn as a general guide. Velocity levels and slopes are drawn to indicate the general move profile the motor will make during the go home move. The vertical axis is velocity and the horizontal axis is position in relation to the home input transitions. Some lines are drawn as closely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

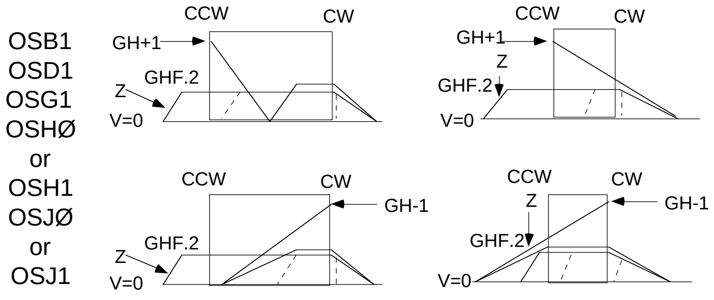


Homing Examples



Note: The final stopping point in the diagrams below is dependant upon the location of the encoder's Z channel. As shown, the Z channel can occur anywhere in relation to the home switch, once the final homing move is started.





Homing Examples

Motion Programs and Sequences

You must program the SX to perform motion functions. A motion program typically consists of initialization (or SX set-up), move profiles, and an I/O or RS-232C interface to execute motion instructions. Refer to the ***SX Software Reference Guide*** to determine if a command is buffered or immediate.

Sequence Commands

Sequences are the building blocks of SX motion programs. A sequence can be 1 command or up to 8K bytes of commands. The sequences are stored in battery backed RAM. All commands are followed by a delimiter. You can use either the space character or a carriage return. Either format is acceptable. The following commands define, erase, and run sequences as well as other specialized sequence functions. Refer to the ***SX Software Reference Guide*** for detailed descriptions and syntax of the following commands.

Sequence Status Commands

Command	Description
XBS	Reports the number of bytes available for sequence programming
XC	Sequence checksum report
XDIR	Reports the defined sequences and the bytes of memory they occupy

Sequence Programming Commands

Command	Description
XD	Starts sequence definition
XEALL	Deletes all sequences from battery-backed RAM
XT	Ends sequence definition

Sequence Execution Commands

Command	Description
XQ	Sets/resets interrupted Run mode
XRP	Runs a sequence with a pause
XR	Runs a sequence
SSJ1	Runs a sequence defined by binary weighted sequence inputs

Sequence Branching Commands

Command	Description
XG	Exits current sequence to execute another sequence
GOTO	Exits current sequence to execute another sequence
XR	Jumps to execute another sequence then returns to the originating sequence
GOSUB	Jumps to execute another sequence then returns to the originating sequence

Sequence Debugging Commands

Command	Description
XTR	Sequence Trace mode
XST	Sequence Single Step mode
XS	Sequence Execution status
#	Step sequence command
DIN	Simulate input state command
DOUT	Simulate output state command

Special Sequence Commands

Command	Description
WHEN	Special condition command
XWHEN	Special condition sequence
XFK	Fault sequence

A sequence is a series of commands. These commands are executed in the order in which they are programmed when the sequence is run. Immediate commands cannot be stored in a sequence, just as they cannot be stored in the command buffer. Only buffered commands may be used in a sequence. Refer to the ***SX Software Reference Guide*** to determine if a command is buffered or immediate.

The SX has 8,000 bytes of nonvolatile memory to store 100 sequences. You can use the **XBS** command to determine how many bytes are available in the sequence buffer and the **XDIR** command to determine what sequences have been programmed. The sequence buffers may have variable lengths, so you may have one long sequence or several short ones, as long as the combined total length does not exceed the 8,000 bytes of allocated space.

To begin the definition of a sequence, enter the Define Sequence (**XD**) command immediately followed by sequence identifier number (1 to 100) and a delimiter. The Terminate Sequence (**XT**) command ends the sequence definition. All commands that you enter after the **XD** command and before the **XT** command will be executed when the sequence is run. An example is provided below. Type **1DR** to see the state of the SX.

Command	Description
> 1DR	Displays the present state of the SX.

Perform the following commands:

Command	Description
> MPI	Places the SX in Incremental mode
> MN	Places the SX in Preset mode
> FSIØ	Places the SX in Indexer mode
> LD3	Disables the SX's limits

Command	Description
> XE1	Erases sequence #1
> XD1	Begins definition of sequence #1
A25	Sets acceleration to 25 rps^2
AD25	Sets deceleration to 25 rps^2
V1Ø	Sets velocity to 10 rps
D5ØØØ	Sets distance to 5,000 steps
G	Executes the move (Go)
H	Reverses direction
G	Executes the move (Go)
XT	Ends definition of sequence
> XR1	Runs sequence #1

You can run a sequence by entering the **XR** command immediately followed by a sequence identifier number (1 - 100) and a delimiter. **Once you define a sequence, it cannot be redefined until you delete it.** You can delete a sequence by entering the **XE** command immediately followed by a sequence identifier (1 - 100) and a delimiter. You may then redefine that sequence. *You can use* **XEALL** *to delete all defined sequences from battery-backed RAM*. **Use these commands with extreme caution. Erased sequences cannot be retrieved**.

Sequence #100 is a power-up sequence (if you have defined it). It is always run when you power up the system or when you reset the Indexer with the Reset (**Z**) command. For convenience, you may find it advantageous to place all of your set-up commands in sequence #100. Sequences that you define are automatically saved into the SX's nonvolatile memory. The only way to erase these sequences individually is by using the Erase Sequence (**XE**) command.

Creating and Executing Sequences

Sequences can be created via RS-232C serial communications. Before you create sequences, you must understand the types of motion and the required user interfaces. To determine the proper user interface, you should be familiar with the methods of selecting sequences within your application.

Selecting Sequences

After you define the sequences from the RS-232C interface, you can execute the sequences by using one of the following modes of operation:

❏ Stand Alone: Select sequences using discrete inputs or thumbwheels See, *Stand Alone Operation* in this chapter for an example of this feature.

❏ Computer Interface: Use the Execute Sequence (**XR**) command to run sequences. See, *Stand Alone Operation* in this chapter for an example.

❏ PLC (Programmable Logic Controller): Use the sequence select inputs to run sequences. See *Stand Alone Operation* in this chapter for an example.

Subroutines

When you use the **GOTO** Sequence (**XG**) and the Run a Sequence (**XR**) commands, you can execute new sequences from within the current sequence. These commands can replace the **GOTO** and **GOSUB** commands respectively. If you use **XG** or **GOTO**, the program will move to the specified sequence. After executing the specified sequence, the system will not return to the original sequence. It will remain in the current sequence, unless it receives another execution command (**XG/GOTO** or **XR/GOSUB**). However, if you use **XR** or **GOSUB**, the program will return control to the original sequence that contained **XR** or **GOSUB**, when **XT** is reached in the subroutine. This prompts the program to return to the sequence that initiated the move to another sequence.

You can nest as many as 16 different levels of sequences within one program. For example, when you exit sequence #1 to execute sequence #2 with **XR2**, you could **GOSUB** to sequence #3 from sequence #2. This procedure of nesting **XR** commands can be repeated 16 times. The **XG** command has no nesting limit since the program will not return control to the original sequence.

Command	Description
> XE2	Erases sequence #2
> XD2	Defines sequence #2
A1ØØ	Sets acceleration to 100 rps^2
AD1ØØ	Sets deceleration to 100 rps^2
V5	Sets velocity to 5 rps
D25ØØØ	Sets distance to 25000 steps
G	Executes the move (Go)
XT	Ends sequence #2 definition
> XE3	Erases sequence #3
> XD3	Defines sequence #3
A1Ø	Sets acceleration to 10 rps^2
V5	Sets velocity to 5 rps
D-25ØØØ	Sets distance to 25000 steps (CCW direction)
G	Executes the move (Go)
XT	Ends sequence #3 definition
> XE1	Erases sequence #1
> XD1	Defines sequence #1
XR2	Executes sequence #2
GOSUB3	Executes sequence #3 (same as an **XR** command)
XT	Ends sequence #1 definition
> 1XTR1	Enables the Trace mode
> XR1	Executes sequence #1

In the previous example, when you execute sequence #1, the program moves to sequence #2. After executing sequence #2, the program returns to sequence #1. The program then moves to execute sequence #3. The Trace mode (**XTR1**) was enabled to help you see how the sequence was executed.

Asynchronous Events—FAULT and WHEN

The SX has special sequences that can be run when a certain condition occurs.

❑ The power-up sequence has already been explained (sequence #100). This sequence is always run on power up and is often used to store the set-up commands for the application.

❑ **FAULT** sequence

❑ **WHEN** sequence

FAULT Sequence

The fault sequence is a sequence that is automatically executed when a fault condition or a Kill (**K**) occurs. Any condition that causes the SX to fault invokes the fault sequence (if one is defined). Refer to the list below for conditions that will prompt execution of the fault or kill sequence. A sequence is designated as a fault or kill sequence with the **XFK** command. You can use the fault sequence to place the SX in a safe state and turn off outputs that may be harmful to the rest of the system. You can use an **IF** command to determine what condition caused the fault so that the appropriate action can be performed. The **IF** statement is explained in the next section. The following steps illustrate the use of a fault sequence. The **IF** statement section explains more about the **ERnnnnn** flag that indicates what fault condition occurred.

Example

Step ① Define an input as a user fault input. You can use this input to indicate that a fault has occurred somewhere external to the system. This input will cause a fault condition.

Command	Description
> IN1U	Defines input #1 as a user fault input

Step ② Designate sequence #10 as the fault sequence.

Command	Description
> XFK1Ø	Designates sequence #10 as the fault sequence
> XE1Ø	Erases sequence #10
> XD1Ø	Defines sequence #10
1"External_Fault	Quote command
XT	End definition of sequence #10

This defines the fault sequence. The quote command sends a message over the RS-232C link to the terminal to tell the operator that a fault has occurred. You can use the Quote command to write statements to the terminal. Remember, the space character is a delimiter so the underscore character should be used to separate words in the quote command. Sequence #10 will now be executed whenever a fault occurs. In the following example, an alternating loop will be performed.

Step ③ The normal *state* of this example application will be an alternating loop.

Command	Description
> A5Ø	Acceleration is 50 rps^2
> AD4Ø	Deceleration is 40 rps^2
> D5ØØØØ	Distance is 50000 steps
> V7	Velocity is 7 rps
> L	Loop command
G	Initiates motion
H	Change direction
N	End the loop

The motor will alternate back and forth continuously.

Step ④ You will now cause a system user fault error. Input states can be simulated using the **DIN** command. Type the following to simulate the activation of the user fault input.

Command	Description
> DINEEEE1	Input #1 is activated

The user fault should have occurred and sequence #10 should have automatically been executed when the fault occurred. To clear the fault from the display, enter the following command.

Command	Description
> DINEEEEØ	Input #1 is de-activated

An **IF** statement could have been used in the fault sequence to determine what fault condition occurred then branch to a sequence that handled that fault condition appropriately. For example, the fault sequence will run for several faults. These faults are limits reached, general product faults, and user faults. Each bit in the error flag (**ERnnnn**) will correspond to one of these fault conditions. Use the **IF** statement to determine which one occurred. Retype Sequence #10 (the fault sequence) as follows:

Command	Description
> 1XE1Ø	Erases sequence #10
> 1XD1Ø	Defines sequence #10
IF(ER1)	If CCW limit is hit,
1"CCW_LIMIT_WAS_HIT	
NIF	End the **IF (ER1)** statement
IF(ERX1)	If CW limit is hit,
1"CW_LIMIT_WAS_HIT	
NIF	End the **IF (ERX1)** statement
IF(ERXXXXXX1)	If user fault occurs
1"USER_fault	
NIF	Ends the **IF (ERXXXXXX1)** statement
XT	End fault sequence statement

Depending on which error caused the program to branch to the fault sequence, the appropriate message will be displayed. Before continuing with the user guide, disable the fault sequence with the **XFKØ** command.

List of Fault or Kill Sequence Causes

❑ Over Temperature Fault (motor de-energized)

❑ Motor Fault (motor de-energized)

❑ Under Voltage Fault (motor de-energized)

❑ Commanded Motor Shutdown (motor de-energized)

❑ End-of-Travel Limit Hit (hard or soft) (motor not de-energized)

❑ Excessive Position Error (motor not de-energized)

❑ Absolute Encoder Rollover (motor not de-energized)

❑ User Fault Input Active (motor de-energized)

❑ Commanded Kill (**K** command) (motor not de-energized)

❑ Kill Input Transition (motor not de-energized)

WHEN Sequence

The **WHEN** sequence is a sequence that can be designated to run when a specific condition evaluates true. This could be a variable having a certain value, the inputs being in a specific state, or the user flag having a set state. Whatever sequence or program is currently being run will be interrupted when the condition is true and the sequence designated by the **XWHEN** command will be executed. There are certain commands which will not be interrupted by the **WHEN** condition. These are **G**, **GH**, **TR**, **U**, **PS**, or a jogging move. The **XWHEN** sequence can be executed during a **G** move or a jogging move if in Position Profile mode (**MPP**). However, the move itself will not be stopped automatically. In the following example, when variable 3 is greater than 50, or Input #1 is on, the SX will execute the **XWHEN** sequence.

Command	Description
> WHEN(VAR3>5Ø_OR_INXXXX1)	Defines the **WHEN** statement that must be true in order for the **XWHEN** sequence to be run.

Step ① The **WHEN** sequence is now defined. Sequence #8 is designated as the **WHEN** sequence.

Command	Description
> XWHEN8	Sequence #8 is designated as a **WHEN** sequence
> XE8	Erases sequence #8
> XD8	Defines sequence #8
A5Ø	Acceleration is 50 rps^2
AD4Ø	Deceleration is 40 rps^2
D3ØØØØ	Distance is 30000 steps
V5	Velocity is 5 rps
G	Initiates motion
XT	Ends the sequence definition

Step ② The normal program to be executed is defined and executed here.

Command	Description
> VAR3=Ø	Variable 3 is initialized to Ø
> A5Ø	Acceleration is 50 rps^2
> AD4Ø	Deceleration is 40 rps^2
> D5ØØØØ	Distance is 50000 steps
> V7	Velocity is 7 rps
> L	Loop command
G	Initiates motion
VAR3=VAR3+1	Variable 3 is increased
N	End the loop

Step ③ You can activate the **WHEN** sequence in two ways.

❑ The **DIN** command can activate the input that will satisfy the **WHEN** condition.

❑ The current program can be run 50 times, then the **WHEN** sequence will be run.

Step ④ Using an input to cause program execution to jump to the **XWHEN** sequence can change or interrupt the program. Within the **XWHEN** sequence, you can use an **IF** statement to check the state of one or more inputs. Based on the state of the inputs, different sequences can be executed. **Disable the XWHEN sequence before continuing with this procedure by entering XWHENØ.**

Power-Up Sequence Execution

You can program the SX to execute a sequence of commands on power up (sequences can be used as subroutines). Sequence #100 always runs on power up. To run another sequence on power up, put an **XR**<num> (or **XG**<num>) at the end of sequence #100. If sequence #100 is empty, nothing happens on power up. Refer to the ***SX Software Reference Guide*** for detailed descriptions and syntax of the following commands.

Command	Description
> XE1ØØ	Erases sequence #100
> XD1ØØ	Begins definition of sequence #100
LD3	Disables limits
A2Ø	Sets acceleration to 20 rps^2
AD2Ø	Sets deceleration to 20 rps^2
V5	Sets velocity to 5 rps
D125ØØ	Sets distance to 12,500 steps
MN	Sets SX in Preset mode
MPI	Sets SX in Incremental mode
FSIØ	Sets unit to the Indexer mode (non Follower)
XG1	Go to sequence #1
XT	Ends sequence definition
> XE1	Erases sequence #1
> XD1	Defines sequence #1
G	Executes a go command
XT	Ends sequence definition
> Z	Resets the Indexer and runs sequence #100

A power-up sequence is typically used to store set-up or initialization parameters that your application requires. Having motion in your power up sequence is not recommended. Examples of these set up commands are listed below.

Command	Description
SSJ1	Continuous Sequence Scan Mode
SN	Scan time
JA	Jog acceleration
JVL	Jog velocity low
JVH	Jog velocity high

SS, **FS**, **OS**, **IN**, and **OUT** commands are examples of set-up commands typically put into sequence #100. You can put any buffered commands into sequence #100 (if you want to execute them during power up).

Sequence Debugging Tools

After creating your sequences, you may need to debug them to ensure that they are performing properly. The SX provides several debugging tools.

❏ In Trace mode, you can trace a sequence as it is executing.

❏ You can set the desired state of the SX ‘s I/O via software commands.

❏ Error messages can be enabled explaining why the SX has stopped execution due to a programming error.

Trace Mode

You can use the Trace mode to debug a sequence or a program of sequences. The Trace mode allows you to track, command-by-command, the entire sequence. It displays to your terminal, over the RS-232C serial link, all of the commands as they are executed. The following example demonstrates the Trace mode.

Step ① Create the following sequence:

Command	Description
> XE1	Erases sequence #1
> XD1	Begins definition of sequence #1
A1Ø	Acceleration is 10 rps^2
AD1Ø	Deceleration is 10 rps^2
V5	Velocity is 5 rps
L5	Loop 5 times
GOSUB3	Jump to sequence #3
N	Ends the loop
XT	Ends the definition of sequence #1

Step ② Define sequence #3.

Command	Description
> XE3	Erases sequence #3
> XD3	Begins definition of sequence #3
D5ØØØØ	Sets the distance to 50000 steps
G	Initiates motion
XT	Ends the definition of sequence #3

Step ③ Enter the following command to enable the Trace mode.

Command	Description
> 1XTR1	Enables the Trace mode

Step ④ You will now execute the sequence. The commands will be displayed on the terminal as each command in the sequence is run. Enter the following command.

Command	Response
> XR1	Run sequence #1—response:

```
*SEQUENCE_ØØ1                    COMMAND_A1Ø
*SEQUENCE_ØØ1                    COMMAND_AD1Ø
*SEQUENCE_ØØ1                    COMMAND_V5
*SEQUENCE_ØØ1                    COMMAND_L5
*SEQUENCE_ØØ1                    COMMAND_GOSUB3____LOOP_COUNT_1
*SEQUENCE_ØØ3                    COMMAND_D5ØØØØ____LOOP_COUNT_1
*SEQUENCE_ØØ3                    COMMAND_G____LOOP_COUNT_1
*SEQUENCE_ØØ3                    COMMAND_XT____LOOP_COUNT_1
*SEQUENCE_ØØ1                    COMMAND_N____LOOP_COUNT_1
*SEQUENCE_ØØ1                    COMMAND_GOSUB3____LOOP_COUNT_2
*SEQUENCE_ØØ3                    COMMAND_D5ØØØØ____LOOP_COUNT_2
*SEQUENCE_ØØ3                    COMMAND_G____LOOP_COUNT_2
*SEQUENCE_ØØ3                    COMMAND_XT____LOOP_COUNT_2
*SEQUENCE_ØØ1                    COMMAND_N____LOOP_COUNT_2
*SEQUENCE_ØØ1                    COMMAND_GOSUB3____LOOP_COUNT_3
*SEQUENCE_ØØ3                    COMMAND_D5ØØØØ____LOOP_COUNT_3
 .
 .
 .
*SEQUENCE_ØØ3                    COMMAND_G____LOOP_COUNT_5
*SEQUENCE_ØØ3                    COMMAND_XT____LOOP_COUNT_5
*SEQUENCE_ØØ1                    COMMAND_N____LOOP_COUNT_5
*SEQUENCE_ØØ1                    COMMAND_XT
```

The format for the Trace mode display is: **`Sequence Number_Command_Loop Count`**

Step ⑤ To exit the Trace mode, enter the following command:

Command	Description
> 1XTRØ	Exits Trace mode

Single-Step Mode

You can debug your program with another level of debugging with the Single-Step mode. Single-Step mode allows you to execute one command at a time when you want the command to be executed. Use the **XST** command to enable Single-Step mode. Once you are in the mode, you can execute a sequence one command at a time. To execute a command, you must use the **#** sign. By entering a **#** followed by a delimiter, you will execute the next command in the sequence. If you follow the **#** sign with a number (*n*) and a delimiter, you will execute the next *n* commands. To illustrate Single-Step mode, use the following steps:

Step ① Enter Single-Step mode and Trace mode.

```
> XST1
> 1XTR1
```

Step ② Begin execution of sequence #1.

```
> XR1
```

Step ③ You will not execute any commands until you use the # command. Type the following.

Command	Description
#	Executes one command

The response will be:

```
*SEQUENCE_ØØ1____COMMAND_A1Ø
```

Step ④ To execute more than one command at a time follow, the # sign with the number of commands you want executed.

Command	Description
#3	Executes 3 commands, then pauses sequence execution

To complete the sequence, use the **#** sign until all the commands are completed. To exit Sequence-Step mode, type:

```
> XSTØ
```

Simulating I/O Activation

If your application has inputs and outputs that integrate the SX with other components in your system, you can simulate the activation of these inputs and outputs so that you can run your sequences without activating the rest of your system. Thus, you can debug your program independently of the rest of your system. This is the same way in which a PLC program can be debugged by simulation of input and output states to run various portions of the program. The SX uses two commands that allow you to simulate the input and output states desired. The **DIN** command controls the inputs and the **DOUT** command controls the programmable outputs.

You will generally use the **DIN** command to cause a specific input pattern to occur so that a sequence can be run. Use the **DOUT** command to simulate the output patterns that are needed to prevent an external portion of your system from operating. You can set the outputs in a state that will be the inactive state of your external system. When you execute your program, a part in the program that will activate the outputs will not actually turn the outputs on to their active state because the **DOUT** command overrides this output and holds the external portion of the machine in an inactive state. When the program is running smoothly without problems you can activate the outputs and the SX will affect the external system.

Outputs

The following steps describe the use and function of the **DOUT** command.

Step ① Display the state of the outputs with the **OUT** command and the **O** command.

Command	Description
> 1OUT	Displays the state of the outputs

The response will be:

```
*1_A_PROGRAMMABLE_OUTPUT______(STATUS_OFF)
*2_A_PROGRAMMABLE_OUTPUT______(STATUS_OFF)
*3_A_PROGRAMMABLE_OUTPUT______(STATUS_OFF)
*4_A_PROGRAMMABLE_OUTPUT______(STATUS_OFF)
*5_DEDICATED_FAULT_OUTPUT_____(STATUS_OFF)
```

Command	Response
> 1O	*ØØØØØ
> DOUT11EE	

Step ② Change the output state using the **O** command.

Command	Description
> O111Ø	

Step ③ Display the state of the outputs with the **OUT** command and the **O** command.

Command	Description
> 1OUT	Displays the state of the outputs

The response will be:

```
*1_A_PROGRAMMABLE_OUTPUT       (DISABLED_ON)
*2_A_PROGRAMMABLE_OUTPUT       (DISABLED_ON)
*3_A_PROGRAMMABLE_OUTPUT       (STATUS_ON)
*4_A_PROGRAMMABLE_OUTPUT       (STATUS_OFF)
*5_DEDICATED_FAULT_OUTPUT      (STATUS_OFF)
```

Command	Response
> 1O	*111Ø

Step ④ You can now disable the outputs into the inactive state using the **DOUT** command. An **E** does not affect the output.

```
> DOUT00EE
```

Step ⑤ Display the state of the outputs with the **OUT** command and the **O** command.

Command	Description
> 1OUT	Displays the state of the outputs

The response will be:

```
*1_A_PROGRAMMABLE_OUTPUT       (DISABLED_OFF)
*2_A_PROGRAMMABLE_OUTPUT       (DISABLED_OFF)
*3_A_PROGRAMMABLE_OUTPUT       (STATUS_ON)
*4_A_PROGRAMMABLE_OUTPUT       (STATUS_OFF)
*5_DEDICATED_FAULT_OUTPUT      (STATUS_OFF)
```

Command	Response
> 1O	*ØØ1Ø

Inputs

The following steps describe the use and function of the **DIN** command. You can use it to activate an input state. The inputs will not actually be in this state, but the SX treats them as if they are in the given state and will use this state to execute its program.

Step ① This sequence will wait for a trigger state to occur and will then begin moving in Continuous mode. An input that is configured as a stop (**S**) input will stop motion.

Command	Description
> 1IN1A	Input #1 is a trigger input
> 1IN2A	Input #2 is a trigger input
> 1IN3D	Input #3 is a stop input
> 1INLØ	The active input level is low
> 1XE1	Erase sequence #1
> 1XD1	Define sequence #1
TR11	Waits for the input trigger state to be 11
A1ØØ	Acceleration is set to 100 rps^2
AD1ØØ	Deceleration is 100 rps^2
V5	Velocity is 5 rps
MC	The Continuous mode is activated
TR1Ø	Waits for a trigger input state of 10
G	Begins motion
XT	Ends sequence definition

Step ② Turn on Trace mode so that you can view the sequence as it is executed.

Command	Description
> 1XTR1	Turns on the Trace mode

Step ③ Execute the sequence.

Command	Description
> XR1	Runs sequence #1

Step ④ The sequence will execute until the **TR11** command is encountered and will then pause waiting for the trigger condition to be satisfied. Simulate the input state using the **DIN** command. Inputs with an **E** value are not affected.

```
1DINEEEE11EEEEE
```

Step ⑤ The sequence will now execute until the **TR1Ø** trigger is encountered and will pause waiting for the new input pattern. Use the **DIN** command to simulate the desired input state.

```
1DINEEE1ØEEEEE
```

Step ⑥ The motor will now move continuously until a the stop input is activated. Activate the stop input with the **DIN** command.

```
> 1DINEEEEEE1EEEE
```

The motor will stop. This illustrates the use of the **DIN** command to simulate the activation of input commands. To deactivate the I/O simulation commands, type the following commands:

```
> 1DINEEEEEEEEEEE
> 1DOUTEEEE
```

Error Messages

The SX has an Error Message mode that can display error messages when an invalid command is attempted. This error message can be useful in debugging a sequence to determine what was wrong with the command that caused the sequence to not run properly. **SSN1** enables Error Message mode.

Command	Description
> 1SSN1	Enters Error Message mode
> 1D1ØØØØØØØØØØØ	*Invalid data field (distance specified is too large)*

The SX will respond with an error message: ***INVALID_DATA_FIELD**

Command	Description
> 1SSN1	Enters Error Message mode
> 1DK1ØØØØ	*Invalid command (DK is not a valid X Series command)*

The SX will respond with an error message: ***INVALID_COMMAND**

SSNØ disables Error Message mode.

Command	Description
> 1SSNØ	Exits the error message mode

High-Level Programming Tools

The Model SX's X-language includes several commands that are common in most high-level programming languages (Pascal, Fortran or BASIC). In addition to these commands, 50 variables **VAR1-VAR5Ø** are provided for performing mathematical functions and boolean comparisons. Along with these variables are certain system variables that you can access—**POS** (Commanded Position or Setpoint), **FEP** (Following Encoder Position) or **ABS** (Actual Encoder Position). This section will introduce and explain how to use these structures with the SX.

Variables

The SX has up to 50 variables that can be used to perform multiplication, division, addition, and subtraction. You can assign these variables to various motion parameters. These parameters and the syntax of assigning a variable to them are listed here.

Command	Description
D(VAR1)	Loads the distance with the value of variable 1
V(VAR4)	Loads the velocity with the value of variable 4
A(VAR5)	Loads the acceleration with the value of variable 5
AD(VAR7)	Loads the deceleration with the value of variable 7
FP(VAR9)	Loads the following port with the value of variable 9
DP(VAR1)	Loads the distance point with the value of variable 1
L(VAR3Ø)	Loads the loop count with the value of variable 30
XR(VAR21)	Executes the sequence number held in variable 21
T(VAR3)	Loads the time delay with the value of variable 3
FOL(VAR29)	Loads the following ratio with the value of variable 29

Variable assignments can be made in sequences or in Immediate Terminal mode. If a variable that has a fractional portion is assigned to a parameter that requires a whole number (such as distance), only the whole number portion of the variable gets assigned to the parameter. As an example of the use of the variables and the math functions they can perform, complete the steps below.

Assigning Variables to Constants

You can set up parameters as variables in a sequence (**D(VAR)**). You can define these variables by assigning a constant value. When the sequence is executed, this value is assigned to the corresponding parameter in the sequence.

Step ① The sequence below executes a move and travels the distance provided in variable 1 and the velocity provided in variable 2. Enter the following commands.

Command	Description
> LD3	Disables limits (*Not needed if limits are installed*)
> MPI	Places the SX in Incremental mode
> MN	Enters the Normal mode
> FSIØ	Sets to Indexer mode (not needed if you do not have Following option—SXF)
> XE1	Erases sequence #1
> XD1	Defines sequence #1
A1Ø	Sets Acceleration to 10 rps^2
AD1Ø	Sets deceleration to 10 rps^2
V(VAR2)	Assigns variable 2 to the velocity term
D(VAR1)	Assigns variable 1 to the distance term
G	Initiates motion
XT	Ends the sequence definition

Step ② You will now assign numbers to the variables **VAR1** and **VAR2**.

Command	Description
> VAR1=25ØØØ	The distance parameter is assigned 25,000
> VAR2=5	The velocity parameter is assigned 5

Step ③ Execute sequence #1 with (**XR1**). The motor will move 25000 steps at 5 rps. Verify that the distance (**D**) and the velocity(**V**) have been assigned these values.

Command	Response
> 1V	*VØ5.ØØØØØ
> 1D	*D+ØØØ25ØØØ

Step ④ Now change the values of the variables as follows.

Command	Description
> VAR1=75ØØØ	The distance parameter is 75,000
> VAR2=1Ø	The velocity parameter is assigned 10

Step ⑤ Repeat steps 3 and 4.

Variables Via RS-232C

When using variables in sequences, the **RSIN** command is a useful tool for interactively prompting the user to enter a variable number. An example of **RSIN** is given in the following steps.

Command	Description
> 1XE1	Erases current sequence #1
> 1XD1	Defines sequence #1
1"ENTER_THE_NUMBER_OF_PARTS	Prompts you for the # of parts to make
1CR	Inserts a carriage return
1LF	Inserts a line feed
VAR5=RSIN	Sequence execution stops until you enter a #
1CR	Inserts a carriage return
1LF	Inserts a line feed
L(VAR5)	The loop count is loaded with the # of parts to make
D25ØØØ	Distance is 25000 steps
G	Initiates motion
N	Ends the loop
1"FINISHED_WITH_PARTS_RUN	Indicates that the run is finished
1CR	Carriage return
1LF	Line feed
XT	Ends definition of sequence #1

Entering The **RSIN** command prompts you to enter a variable via RS-232C that will be used in a sequence. When a variable is assigned to **RSIN**, the execution of the sequence is stopped until you enter the variable . To enter the variable, you must precede the number with an exclamation point (**!**).

Step ① Turn on the Trace mode and run the sequence.

```
> 1XTR1
> XR1
```

Step ② The program will stop at the **RSIN** command and wait for you to be enter a variable number. Enter the variable number.

```
> !1Ø
```

The sequence executes a 25,000-step move 10 times making 10 parts.

Math Operations

In addition to assigning constants to variables, three system parameters can be assigned to a variable.

❑ **POS**—Commanded Position
❑ **FEP**—Following Encoder Position
❑ **ABS**—Actual Encoder Position

These are *read-only* variables. They allow you to read commanded positions, following encoder positions, or actual encoder positions at any time by setting a general-purpose variable equal to the read only variable. They can be also be used for conditional comparisons (i.e., **IF(POS>25ØØØ)**).

Command	Response
> VAR1=POS	Assigns the current value of the command position to variable #1

Performing Math Operations with Variables

The SX has the ability to perform simple math functions with its variables (add, subtract, multiply, and divide). The following sequence of steps illustrates the math capabilities of the SX. Note; Only one math function can be performed per command line (i.e., **`VAR1=4*5+6`** is illegal).

Step ①

Addition:

```
Command                       Response
> VAR1=5
> VAR23=1ØØØ.565
> VAR11=VAR1+VAR23
> VAR23=VAR11+VAR1
> VAR1=VAR1+5
> 1VAR1                       *+ØØØØØØØØØØØØØ1Ø.ØØØØØ
> 1VAR11                      *+ØØØØØØØØØØØ1ØØ5.565ØØ
> 1VAR23                      *+ØØ1Ø1Ø.565ØØ
> VAR3=POS+25ØØØ
> 1VAR3                       (Commanded Position plus 25ØØØ)
```

Step ②

Subtraction:

```
Command                       Response
> VAR3=1Ø
> VAR2Ø=15.5
> VAR3=VAR3-VAR20
> VAR19=ABS-25ØØØ
> 1VAR3                       *-ØØØØØØØØØØØØØØ5.5ØØØØ
> 1VAR2Ø                      *+ØØØØØØØØØØØØØ15.5ØØØØ
> 1VAR19                      (Actual Encoder Position minus 25ØØØ)
```

Step ③

Multiplication:

```
Command                       Response
> VAR3=1Ø
> VAR2Ø=15.5
> VAR3=VAR3*VAR2Ø
> VAR19=ABS*POS
> 1VAR3                       *+ØØØØØØØØØØØØ155.ØØØØØ
> 1VAR2Ø                      *+ØØØØØØØØØØØØØ15.5ØØØØ
> 1VAR19                      (Actual Encoder Position multiplied by commanded
                              position)
```

Step ④

Division:

```
Command                       Response
> VAR3=1Ø
> VAR2Ø=15.5
> VAR3=VAR3/VAR2Ø
> VAR3Ø=75
> VAR19=VAR3Ø/VAR3
> 1VAR3                       *+ØØØØØØØØØØØØØØØ.64516
> 1VAR2Ø                      *+ØØØØØØØØØØØØØ15.5ØØØØ
> 1VAR3Ø                      *+ØØØØØØØØØØØØØ75.ØØØØØ
> 1VAR19                      *+ØØØØØØØØØØØØ116.25Ø23
```

Data

Inputs can be defined as data inputs to allow for external entry of motion data, loop counts, sequence select, time delays, and variable values. The following commands will read and enter data from the inputs:

❑	**VARD**	Variable Read
❑	**DRD**	Distance
❑	**VRD**	Velocity
❑	**LRD**	Loop Count
❑	**TRD**	Time Delay
❑	**XRD**	Sequence Number
❑	**FRD**	Following Ratio

The weighting for data inputs is *binary coded decimal* or BCD. This weighting allows you to enter data via thumbwheels. To use the data inputs to enter data, the outputs must also be used. Outputs 1 - 3 must be configured as data strobe outputs. They are used to select or strobe the appropriate digit while reading data. Up to 16 digits of data and one sign bit may be entered. The recommended and most common method of controlling the input lines to read data is through thumbwheels. A PLC may also be used to enter data. An explanation of interfacing to thumbwheels or a PLC is introduced later in this chapter.

Delays

You can use the Time (**T**) command to halt the operation of the Indexer function for a preset time. If you are in the Continuous mode and Position Profile Mode, you may use the Time (**T**) command to run the motor at continuous velocity for a set time, then change to a different velocity. In Preset mode, the motor finishes the move before the Indexer executes the time delay.

Command	Description
> PS	Waits for the SX to receive a **C** command before executing the next command
D25ØØØ	Sets distance to 25000 steps
G	Moves motor 25,000 steps
T5	Waits 5 seconds after the move ends
H	Changes motor direction
G	Moves motor 25,000 steps in the opposite direction
C	Continues execution

Complex Branching and Looping

The SX supports the high-level language structures for branching and looping. Each conditional branch or loop evaluates a condition statement. Depending on whether this condition statement evaluates true or not determines where the SX will branch to. The unconditional branching and looping statements have been introduced already. These are the **GOTO** (Branch), **GOSUB** (Branch & Return) and **L** (Loop) command. The **L** command is explained further here.

Unconditional Looping

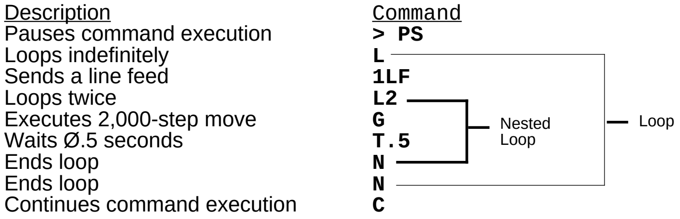
The Loop (**L**) command is an unconditional command. You may use **L** to repeat a series of commands. You can nest loop commands up to 16 levels deep.

Command	Description
> PS	Pauses command execution until the SX receives a **C** command
MPI	Sets unit to Incremental mode
A5Ø	Sets acceleration to 50 rps^2
V5	Sets velocity to 5 rps
L5	Loops 5 times
D2ØØØ	Sets distance to 2,000 steps
G	Executes the move (Go)
T2	Delays 2 seconds after the move
N	Ends loop
C	Continue—Initiates command execution to resume

The example below shows how to nest a loop inside a loop. The motor makes two moves and returns a line feed. The unit repeats these procedures until you instruct it to stop.

☞ **Helpful Hint:**

This command execution continues until you issue the Stop (**S**) or Kill (**K)** commands.



Description	Command
Pauses command execution	**> PS**
Loops indefinitely	**L**
Sends a line feed	**1LF**
Loops twice	**L2**
Executes 2,000-step move	**G**
Waits Ø.5 seconds	**T.5**
Ends loop	**N**
Ends loop	**N**
Continues command execution	**C**

Unconditional Branching

The unconditional branching commands **GOTO** and **GOSUB** were explained earlier in the sequence section.

Conditionals

The branching commands evaluate condition statements to make branching decisions. If the condition is true, one set of commands is processed. If the condition is false, another set of commands may be executed. The commands that evaluate conditions are listed below.

IF (condition true)—*execute these commands*
ELSE—*execute these commands*
NIF

WHILE (condition true)—*execute these commands*
NWHILE

REPEAT—*execute these commands*
UNTIL (condition true)

Condition statements can be very complex.

You can use the following types of conditional statements with the SX.

- Error Flags
- User Flags
- Input State
- Variable Comparisons
- Boolean Comparisons

Error Flags

The error flag (**ERXXXXXXXX**) is useful if you want to trap different error conditions and create different sequences to respond to them. The ***SX Software Reference Guide*** explains the errors that can be trapped in the evaluation command description. An example of the statement's use is provided below.

```
 > IF(ER110XXXX)
 > GOSUB2
 > NIF
```

User Flags

You can set the user flag **(FLØØØ111XØ)** and modify it within sequences to mark where the program has gone or to indicate any special state so that a conditional statement can be made. An example of the statement's use is provided below. The **SFL** command is used to set the user flag.

```
> WHILE(FLØØ11XXXX)
> D1ØØ
> G
> NWHILE
```

Input State

An example of this statement's use (**IN11111ØØØ1Ø**) is provided below.

```
> REPEAT
> GOSUB4
> UNTIL(INØØ1XX11XXXXX)
```

Variable Comparisons

Typical variable comparisons are shown below.

```
> VARn>VARm
> VARn<VARm
> VARn=VARm
> IF(VAR1>VAR2)
> WHILE(VAR3=1Ø)
> GOSUB2
> NWHILE
> NIF
> IF(VAR1>POS)
> GOSUB3
> NIF
> REPEAT
> IF(INXXXX1)
> GOTO3
> NIF
> UNTIL(POS>1ØØØØØ)
```

Boolean Comparisons

```
AND
OR
```

An example of the statement's use is provided below.

```
>WHILE(VAR3>1Ø_AND_IN11ØØ1_OR_VAR4=1)
>GOSUB8
>NWHILE
```

Condition statements are explained in the ***SX Software Reference Guide***.

Conditional Looping

The SX supports two conditional looping structures—**REPEAT/UNTIL** & **WHILE/NWHILE**.

REPEAT/ UNTIL

With the **REPEAT** command, all commands are repeated between **REPEAT** and **UNTIL**, until the condition is true. Enter the following sequence.

Step ①

Command	Description
> VAR5=Ø	Initializes variable 5 to 0
> 1XE1Ø	Erases sequence #10
> 1XD1Ø	Defines sequence #10
REPEAT	Begins the **REPEAT** loop
A5Ø	Acceleration is 50 rps^2
AD5Ø	Deceleration is 50 rps^2
V5	Sets velocity to 5 rps
D25ØØØ	Distance is 25,000 steps
G	Executes the move (Go)
VAR5=VAR5+1	Variable 5 counts up from 0
UNTIL(INXXXX111Ø_OR_VAR5>1Ø)	When the 111Ø input condition occurs, the programmable inputs or VAR5 > 10, the loop will stop
1"DONE_LOOPING	Quote command indicates that the loop is finished
1CR	Inserts a carriage return
1LF	Inserts a line feed
XT	Ends definition of sequence #10

Step ②

Use the Trace mode to display the commands as they are run.

```
> 1XTR1
> XR1Ø
```

The loop can be exited either by using the **DIN** command to satisfy the input state or letting the **VAR5** counter count to 10.

WHILE

With the **WHILE** command, all commands are repeated between **WHILE** and **NWHILE** until the **WHILE** condition is true. Enter the following sequence.

Step ①

Command	Description
> VAR5=Ø	Initializes variable 5 to 0
> 1XE1Ø	Erases sequence #10
> 1XD1Ø	Defines sequence #10
WHILE(INXXX111Ø_OR_VAR5<1Ø)	While input pattern is XXX111Ø or variable 5 is<10, repeat loop
A5Ø	Acceleration is 50 rps^2
AD5Ø	Deceleration is 50 rps^2
V5	Velocity is 5 rps
D25ØØØ	Distance is 25000 steps
G	Executes the move (Go)
VAR5=VAR5+1	Variable 5 counts up from 0
NWHILE	
1"DONE_LOOPING	Indicates that the loop is done
1CR	Inserts a carriage return
1LF	Inserts a line feed
XT	Ends definition of sequence #10

Step ②

Use the Trace mode to display the commands as they are run.

```
> 1XTR1
```

```
> XR1Ø
```

You can exit the loop with the **DIN** command (the input state does not match the **IN** command). You can also exit the loop by letting the **VAR5** counter get to 10. If the input pattern is not **XXX111Ø**, the loop will not be run.

Conditional Branching

You can use the **IF** statement for conditional branching. All commands between **IF** and **ELSE** are executed if the condition is true. If the condition is false, the commands between **ELSE** and **NIF** are executed. **ELSE** may not be required. The commands between **IF** and **NIF** are executed if the condition is true. Examples of these statements are provided.

☛ Note

If statements are evaluated at the time the command is executed they are not continually evaluated in the background.

❑ Error Flag

❑ User Flag

❑ Input State

Error Flag

The **IF** command checks to see if any error condition exists to execute a conditional command. This command is useful if you wish to trap different error conditions (Drive Disabled, User Fault Input Activated, Excessive Position Error, etc). Refer to the ***SX Software Reference Guide.***

Command	Description
> XE1Ø	Erases sequence #10
> XD1Ø	Defines sequence #10—if a fault occurs, sequence #10 will execute—defined with Set Fault or Kill Sequence (**XFK1Ø**) command
IF(ER1)	If hardware CCW limit switch is reached, run the following commands:
1"CCW_LIMIT_HIT	Display the error message
NIF	Ends IF statement
IF(ERX1)	If hardware CW limit switch is reached, perform the following commands:
1"CW_LIMIT_HIT	Display the error message
NIF	Ends IF statement
XT	Ends sequence
> XFK1Ø	Sets sequence #10 as the Fault sequence

User Flag

This example uses the pattern set by the User Flag (**SFL**) command to run the conditional commands. This command is useful if you wish to make a decision based on previous sequence executions that will set or clear the user flag bits. For example, if an application has several sequences, you can assign different bit patterns with the **SFL** command at the end of each sequence. If you select these sequences from the host computer, you may wish to make different moves depending on the sequence you ran. Refer to the ***SX Software Reference Guide*** for a detailed description.

Command	Description
> PS	Waits for the SX to receive a **C** command before executing the next command
SFL1Ø1Ø	Sets user flag bits 7 and 5 and clears bits 6 and 4, remaining bits are not altered
IF(FL1Ø1Ø)	If user flag bits 5 & 7 are set, and bits 6 & 4 are clear, perform these commands
A1Ø	Sets acceleration to 10 rps^2
V5	Sets velocity to 5 rps
D25ØØØ	Sets distance to 25,000 steps
G	Executes the move (Go)
NIF	Ends **IF** statement
C	Continues execution

If the **FL** pattern matches the **SFL** setting, the motor moves 25,000 steps. You can change the **SFL** pattern at different points in sequences to map a path for sequence execution.

Input State

The (IN) command compares the input pattern (**CW**, **CCW**, **HOME**, **REG**, and **I1** - **I8**) to execute the conditional commands. This command is useful for branching and performing conditional moves using the programmable inputs. For a detailed description of this command, refer to the ***SX Software Reference Guide***.

Command	Description
> 1XE5	Erases sequence #5
> 1XD5	Defines sequence #5
IF(INXXXX1Ø)	If **I1** is active and **I2** is not active, issue the following commands:
A1Ø	Sets acceleration to 10 rps^2
V5	Sets velocity to 5 rps
D25ØØØ	Sets distance to 25,000 steps
G	Executes the move (Go)
NIF	Ends IF statement
IF(INXXXXØ1)	If **I1** is inactive (open) & **I2** is active (closed), run the following commands:
A1Ø	Sets acceleration to 10 rps^2
V5	Sets velocity to 5 rps
D-5ØØØ	Sets distance to 5,000 steps in the opposite direction
G	Executes the move (Go)
NIF	Ends IF statement
IF(INXXXX1)	If **I1** is active, do the following command.
1"DONE	Ends message saying done
NIF	Ends **IF** statement
1XT	Ends sequence definition

Use the **DIN** command or the inputs themselves to execute the different trigger input states. You can use the Trace mode to see what commands are executed. The input state represents the state of all inputs regardless of whether they are dedicated limits or programmable inputs. This is different than the trigger (**TR**) command where only the inputs defined as triggers are used in the command.

Branching Using Variables and Boolean Logic

You can use the **IF** statement to branch based on variable values. Multiple comparisons can be made in one condition statement using the Boolean, **OR**, and **AND** functions as long as the statement doesn't exceed 40 characters.

Command	Description
> XE8	Erases sequence 8
> XD8	Defines sequence 8
VAR5=15	Variable 5 = 15
IF(VAR5<1Ø_AND_VAR4=2Ø)	If variable 5 < 10 & variable 4 = 20, run commands up to the **ELSE** command
A1ØØ	Sets acceleration to 100 rps^2
AD1ØØ	Sets deceleration to 100 rps^2
V5	Sets velocity to 5 rps
D25ØØØ	Sets distance to 25,000 steps
G	Executes the move (Go)
VAR5=VAR5-1	Variable 5 decrements one
ELSE	Ends IF statement
A1ØØ	Sets acceleration to 100 rps^2
AD1ØØ	Sets deceleration to 100 rps^2
V5	Sets velocity to 5 rps
D-5ØØØ	Sets distance to 5,000 steps in the CCW direction
G	Executes the move (Go)
NIF	Ends the IF statement
1XT	Ends the sequence definition

Motion Profiling Mode—On-the-Fly Changes

Motion Profiling mode allows you to execute buffered commands while a move is being made (on-the-fly). When you enter this mode, the SX will execute commands while the move profile is in progress. You can enter and exit this mode from within a sequence. This mode allows you to change velocity on-the-fly based on distance, turn on outputs based on distance, perform math and other commands while in motion. Changing the acceleration, deceleration, or distance parameters will not affect any move already in progress, but will be in effect for any subsequent moves. The following commands are used with Motion Profiling mode.

- ❑ **MPP**—Enter Motion Profiling Mode
- ❑ **NG**—Exits Motion Profiling Mode
- ❑ **DP**—Sets Distance Points within Motion Profiling Mode

While the SX is in Motion Profiling mode, you can execute most command while a move is being made. The exceptions are other move commands such as **G** or **GH**. When the SX reaches an **MPP** command, all subsequent commands will be executed until the **NG** command is encountered. An example of the **MPP** command is provided below.

```
XD1 D5ØØØØ V1 MPP G O1 TR1X1 V4 NG XT
```

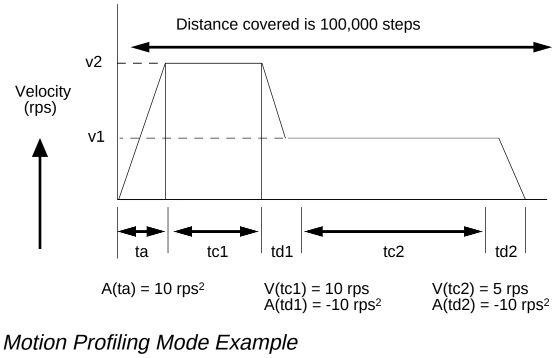
In this example, a 50,000-step move is made. The initial velocity is 1 rps. Motion begins with the **G** command. Output #1 is turned on with the **O1** command. The SX then runs the trigger command (**TR**) until the condition is true, at which point, it changes the velocity. When the SX encounters the **NG** command, it will not receive any additional commands until the 50,000-step move is completed. The 50,000-step move will be completed even if the **TR1X1** is not satisfied. The SX will however, wait until the **TR** condition is satisfied, at which point it will update the last specified velocity from **V1** to **V4**.

Changes Based on Distance

Changes are made based on distance using the distance point (**DP**) command. This command causes a delay in the processing of the commands until the motor reaches the specified distance point. Processing will continue once the distance point is reached. In this way, velocity changes and the activation of outputs can be based on distance.

The distance point is interpreted differently for Absolute mode versus Incremental mode. To change velocity on-the-fly based on distance, the Motion Profiling Mode (**MPP**) command must be used with the Distance Point (**DP**) command. The following sequence example executes the profile shown in the figure below.

```
1XD1 PZ D1ØØØØØ V2 MPP G DP25ØØØ O1 DP5ØØØØ V1 OØ NG XT
```



Motion Profiling Mode Example

In Incremental mode, commands are processed until the **DP** command is reached. The SX pauses at **DP** until the motor moves 25,000 steps. The SX then turns on output #1. The 25,000 steps are counted from the point at which **DP** is encountered. When **DP5ØØØØ** is reached, the SX pauses until the motor moves an additional 50,000 steps. The SX then turns output #1 off and decreases the velocity to 1 rps. In this example, if the SX is in Incremental mode, the output will be turned on at 25,000 steps and turned off after the motor has moved a total of 75,000 steps from the beginning of the first move.

In Absolute mode, the value specified with **DP** is interpreted as an absolute position relative to the zero point location. In this example, the output would be turned on at 25,000 steps and turned off after the motor had passed the 50,000-step point.

Stopping Motion with a Stop Input

A common use of the Motion Profiling mode is to perform a continuous move and stop the move from the inputs. This can be done in two ways. You can define an input as a stop input. Motion can be stopped by activating the stop input.

The other method is to use the **STOP** command and place it within a sequence. The following example illustrates the steps of these two methods.

Step ① Define the input as a stop input.

Command	Description
> IN1D	Defines input #1 as a stop input

Step ② Create the sequence with a continuous move.

Command	Description
> XE1	Erase previous sequence #1
> XD1	Begins definition of sequence #1
V5	Sets velocity to 5 rps
A1ØØ	Sets acceleration to 100 rps^2
MC	Sets the SX to Continuous mode
MPP	Sets the SX to Motion Profiling mode
G	Executes the move (Go)
XT	Ends sequence #1 definition

Step ③ Execute sequence #1 (**XR1**). The motor will move at 5 rps and will not stop until you activate the stop input (input #1). Activate the stop input.

The motor will stop at a controlled deceleration. In this case, the buffer will be dumped and the sequences will not be finished. The SX can also be set to stop only the motion when the stop input is activated. Enabling the **SSH1** command will cause the stop input to stop the move in progress and whatever command is currently being executed and go onto the next command in the sequence or input buffer.

Step ④ Issue Display Parameters (**1DR**) command. The SX is still in Motion Profiling mode. Enter the **NG** command to exit the mode.

Stopping Motion with the STOP Command

The following step-by-step example illustrates the method of stopping a continuous move with the **STOP** command.

Step ① Configure input #1 as a trigger input.

Command	Description
> IN1A	Configures input #1 as a trigger input
> SSHØ	Disable **SSH** mode

Step ②

Create a sequence with a **STOP** command after the trigger.

Command	Description
> XE1	Erases previous sequence #1
> XD1	Begins the definition of sequence #1
V5	Sets velocity to 5 rps
A1ØØ	Sets acceleration to 100 rps^2
MC	Sets the SX to Continuous mode
MPP	Sets the SX to Motion Profiling mode
G	Executes the move (Go)
TR1	Activates trigger #1
STOP	Stops motion when the trigger condition is met
NG	Exits Motion Profiling mode
XT	Ends sequence #1 definition

Step ③

Issue the Display Parameters (**1DR**) command. The SX is still in Motion Profiling mode. In this example, an **NG** command was required after motion was stopped. When a **STOP** command is issued, the command buffer is emptied. Therefore, the commands that have not been executed in the sequence at the time the stop occurs will not be executed. In this example, the **NG** command is not executed because motion was stopped. In fact, **NG** can never be executed under these conditions. *If the* **STOP** *command is used to stop continuous motion, the* **NG** *must be issued either at the beginning of the next sequence, directly via the RS-232C interface or at some point in the sequence prior to the stop.*

To prevent the SX from stopping without finishing the sequence that it is currently executing, a software switch has been provided that will cause the SX to continue executing the sequence it was running when the **STOP** was issued. By entering the Clear/Save the Command Buffer on Stop (**SSH1**) command, the SX will stop motion when it encounters a **STOP** and continue processing the commands in the sequence. *Enter* **SSH1** *and repeat the previous step. Notice how the Motion Profiling mode is exited.*

Sequence Scan Mode and the Stop Command

In applications that use the Sequence Scan mode (see Sequence Select, Sequence Scanning, and Programmable Inputs and Outputs sections in this chapter) and the **STOP** command, you must use the (**SSH1**) command to prevent the Sequence Scan mode from being disabled. Under normal conditions, the Sequence Scan mode is aborted when the SX encounters a **STOP** command. If the **STOP** command is issued from a stop input or from within a sequence, the Sequence Scan mode will be aborted. In some cases, you will not want to abort the mode. **SSH1** allows the SX to complete the current sequence from the point at which the **STOP** was issued and continue in Sequence Scan mode. The **OSI1** command would allow continuing in Sequence Scan mode on a **STOP** but not continuing in the current sequence. The following example illustrates such a sequence.

Step ①

Configure input #1 as a trigger input and input #2 as a sequence-select input.

Command	Description
> IN1A	Configures input #1 as a trigger input
> IN2B	Configures input #2 as a sequence-select input

Step ②

Enable the Sequence Scan mode with the **SSJ1** command and the Clear/Save Command Buffer on Step mode **(SSH1**).

Step ③ Create a sequence with a **STOP** command after the trigger.

Command	Description
> XE1	Erase Previous Sequence #1
> XD1	Begins the definition of sequence #1
V5	Sets velocity to 5 rps
A1ØØ	Sets acceleration to100 rps^2
MC	Sets the SX to Continuous mode
MPP	Sets the SX to Motion Profiling mode
G	Executes the move (Go)
TR1	Activates trigger #1
STOP	Stops motion when the trigger condition is met
NG	Exits Motion Profiling mode
XT	Ends sequence #1 definition

Step ④ Execute the sequence by activating input #2. Stop the move at any time by activating input #1.

Step ⑤ The SX is still in Sequence Scan mode and the move can be repeated by activating Input #2 again (and stopped with Input #1). The SX is not in MPP mode in-between the sequence execution since the SX executed the **NG** command at the end of the sequence due to **SSH1**. **OSI1** would have saved the Sequence Select mode but not allowed executing the **NG** command after the **STOP**.

Other Uses of Motion Profiling Mode

Motion Profiling mode allows a great amount of flexibility in the complexity of the control of the SX during motion. You can turn on outputs, change velocity, and perform math functions. **The primary application concern to consider during sequence execution is the amount of time required to perform the commands.** In some cases, the execution of commands may depend on the motion. The following examples show additional uses of the Motion Profiling mode.

Turning on Inputs, Using Time Delays, and Math

Command	Description
> XE1	Erase previous Sequence #1
> 1XD1	Begins the definition of sequence #1
1PZ	Sets axis #1 position counter to Ø
1MC	Sets the SX to Continuous mode
1MPP	Sets SX to Motion Profiling mode
1G	Executes the move (Go)
1T.5	Sets a 0.5 second delay
1O11Ø	Turns outputs #1 and #2 on, #3 off
1T.5	Sets a 0.5 second delay
1OØØ1	Turns outputs #1 and #2 off, #3 on
REPEAT	Starts repeat loop
VAR1=VAR1+1	Increases variable #1 by 1
T.1	Sets a 0.1 second delay
UNTIL(POS>4ØØØØØ)	Continues looping until the SX's position is > 400,000
STOP	Halts command processing
NG	Exits SX from Motion Profiling mode
XT	Ends the definition of sequence #1

Triggers, input states (**INXX111**), time delays, and the distance points allow you to control when and where procedures occur during motion in your program. Motion Profiling mode offers you the flexibility to satisfy a variety of different application needs. In the above example, **SSH1** should be enabled to exit **MMP** (**NG**) after the **STOP** is executed.

Interfacing to the SX

This section discusses interfacing the SX to other equipment in a system using the programmable inputs and outputs.

- ❑ Input and Output Function Types
- ❑ Switches
- ❑ Sequence Selecting
- ❑ Thumbwheels
- ❑ Sequence Selecting from a Thumbwheel
- ❑ PLC Operation
- ❑ Sequence Selecting from a PLC
- ❑ Miscellaneous Control from a PLC
- ❑ RP240 Operator Panel

Programmable Inputs and Outputs

The SX has a very flexible input and output scheme for defining I/O in a way that is suitable for almost any application. There are 8 programmable inputs (**`I1 - I8`** on the front panel). The other four inputs are dedicated for limits, home, and registration. There are also 4 programmable outputs in addition to a dedicated Fault output. This section explains some of the functions that the inputs and outputs can perform and explains how to use thumbwheels for an interface with the SX. Using the inputs in combination with the outputs you can use up to 32 digits of thumbwheels with the SX. Refer to *Chapter 3, Installation* for more information on wiring the inputs and outputs to other equipment and later in this cchapter for wiring to the Compumotor TM8 Module.

Output Functions

You can turn the programmable outputs (**`O1`** - **`O4`**) on and off with the Output (**`O`**) and Immediate Output (**`IO`**) commands. Outputs **`O1`** - **`O4`** are factory set as programmable outputs. The Fault output is dedicated as a fault output. However, you can configure all of the programmable outputs to perform different functions (Moving/Not Moving, Amp Off, Strobe, etc.) with the Configure Output (**`OUT`**) command. Refer to the **`OUT`** command in the ***SX Software Reference Guide*** for descriptions of the available functions. You can use these outputs to turn on and off other devices (i.e., lights, switches, relays, etc.). The output functions have unique letter assignments.

A:	Programmable Output	**L:**	Position Error Fault
B:	Moving/Not Moving	**N:**	CW Software Limit Reached
C:	Sequence in Progress	**P:**	CCW Software Limit Reached
D:	At Soft or Hard Limits	**R:**	CW Hardware Limit Reached
E:	At Position Zero	**S:**	CCW Hardware Limit Reached
F:	Fault Indicator	**T:**	Output Based on Position
H:	Shutdown Commanded	**U:**	Pulse Output
J:	Strobe Out	**Z:**	No Function Assigned
K:	Invalid Command Error		

Command	Description
> PS	Pauses command execution until the SX receives a Continue (**C**) command
MN	Sets unit to Normal mode
LD3	Disables the SX's limits
A1Ø	Sets acceleration to 10 rps²
V5	Sets velocity to 5 rps
D25ØØØ	Sets distance to 25,000 steps
OUT1A	Sets **O1** as a programmable output
OUT2A	Sets **O2** as a programmable output
OUT3B	Sets **O3** as a Moving/Not Moving output
O1Ø	Turns **O1** on and **O2** off
G	Executes the move
OØ1	Turn **O1** off and **O2** on
C	Initiates command execution to resume

This example defines **O1** and **O2** as programmable outputs and **O3** as a Moving/Not Moving output. Before the motor moves 25,000 steps, **O1** is turned on and **O2** is turned off. These outputs will remain in this state until the move is completed, then **O1** will turn off and **O2** will be turned on. While the motor is moving, **O3** remains on.

The active level of the programmable outputs can be changed with the **OUTL** command. Refer to the ***SX Software Reference Guide*** for more details.

Programmable Output
This output type gives the user on/off control over an output using the **O** or **IO** commands.

Moving/Not Moving Output
This output type indicates motion due to a **G** or **GH** command. This output does not indicate motion due to a position maintenance move.

Sequence in Progress Output
This output type indicates the SX is busy running a defined sequence.

At Soft or Hard Limit Output
This output type will indicate whenever any type of end of travel limit is hit during a move if the limits are enabled. This output is reset with the **ST0** or **ON** command.

At Position Zero Output
This output type indicates when the absolute position counter (**1PR** command) is equal to zero.

Fault Indicator
This output type indicates when certain fault conditions are true. The conditions that will activate it are:Over-temperature faultMotor faultLow-line fault (brown out)Excessive position error (cleared by **ST0** or **ON** command)Auto run mode activeUser fault input activeThese errors are only cleared by resetting the unit or cycling power unless otherwise noted.

Shutdown Commanded
This output type indicates a drive disabled condition caused by the **ST1** or **OFF** command. This output is reset with the **ST0** or **ON** command.

Strobe Output
This output type is used in conjunction with the SX's programmable inputs configured as data inputs. The strobe outputs will cycle through different output patterns to tell external devices which digit of information to put on the data inputs. Refer to later in this chapter for more information on this topic.

Invalid Command Error
This output type indicates that an invalid command was seen by the SX, either due to a value out of range, a syntax error, or an impossible request This output is reset by resetting or cycling power.

Position Error Fault
This output type indicates when the encoder input counter (**1PX**) differs from the number of pulses sent out (**1PR**) by more than the acceptable position error (set with the **CPE** command). The **DPE** command also reflects this positional error. This output is reset with the **ST0** or **ON** command.

CW Software Limit Hit
This output type indicates that the CW software limit was hit. This output is reset with the **ST0** or **ON** command.

CCW Software Limit Hit
This output type indicates that the CCW software limit was hit. This output is reset with the **ST0** or **ON** command.

CW Hardware Limit Hit
This output type indicates that the CW hardware limit was hit. This output is reset with the **ST0** or **ON** command.

CCW Hardware Limit Hit
This output type indicates that the CCW hardware limit was hit. This output is reset with the **ST0** or **ON** command.

Output Based on Position
This output type indicates that a certain position has been reached, either an incremental or absolute position, depending on the setting of the **OUTP** command. Refer to the **OUTP** command for more details and how to calculate the accuracy of this output.

Pulse Output
This output type generates a pulse train output that can be used for a limited motion axis output. The number of pulses sent out and the appropriate pulse width will be set with the **PUL** command. It is not intended as a full axis output and has no acceleration or deceleration ramp.

No Function Assigned
This output type has no function assigned to it and can be used to disable an output when needed.

Input Functions

The inputs can individually be programmed to perform any of the following functions. Each function has an assigned letter:

A:	Trigger
B:	Sequence Select
C:	Kill
D:	Stop
E:	Command Enable
F:	Pause/Continue
G:	Go
H:	Direction
I:	Synchronization
J:	Jog+ (CW)
K:	Jog- (CCW)
L:	Jog Speed Select
M:	Terminate Loop
N:	Data
P:	Memory Lock
R:	Reset
S:	Go Home
T:	Position Zero
U:	User Fault
V:	Data Valid
W:	Data Sign
X:	Increase Following Ratio
Y:	Decrease Following Ratio
Z:	No Function Assigned

The input functions are level sensitive unless otherwise specified.

Trigger Input
This input type is mainly used in conjunction with the **TR** command to pause command processing until the specified input pattern is satisfied. This input function is explained in more detail later in this chapter.

Sequence Select Input
This input type is used to remotely execute predefined sequences using the inputs. This input function is explained in more detail later in this chapter.

Kill Input
This input type is used to immediately halt all motion with no deceleration, stops the current sequence execution, and dumps the command buffer. Functions the same as the K command.

Stop Input
This input type is used to immediately stop the motor at the specified deceleration rate (**AD** command). It will also dump the sequence and command buffer unless the **SSH** or **SSL** command has been set previously.

Command Enable Input
This input type is used to enable or disable the drive. Functions the same as the **ON** and **OFF** commands. This input type is edge sensitive.

Pause/Continue Input
This input type is used to pause and continue command execution. This input will not pause motion in progress. The input is a pause in the active state and a continue in the inactive state. Functions the same as the **U** and **C** commands

Go Input
This input type is used to initiate a move. Functions the same as the **G** command. This input type is edge sensitive.

Direction Input
This input type is used to change the direction of the motor. The direction change will not affect a move in progress. Functions the same as the **H** command. This input type is edge sensitive.

Synchronization Input
This input type is used in the following self correction mode to adjust the following ratio. Refer to the **FSK** and **FSL** commands as well as *Chapter 5, SXF Following* for more information.

Jog CW Input
This input type is used to jog the motor in the CW direction. The jogging velocity is set with the **JVL** and **JVH** commands. Jogging is enabled with the **OSE** command. This input is normally level sensitive, except after stopping or killing a jog. In this case the input needs to see the edge transition again.

Jog CCW Input
This input type is used to jog the motor in the CCW direction. The jogging velocity is set with the **JVL** and **JVH** commands. Jogging is enabled with the **OSE** command. This input is normally level sensitive, except after stopping or killing a jog. In this case the input needs to see the edge transition again.

Jog Speed Select Input
This input type is used to select between the **JVH** and **JVL** velocities. If not defined or used, the jogging velocity defaults to **JVL**. This input will affect a jog in progress.

Terminate Loop Input
This input type is used to terminate an **L**, **N** command loop at the end of the current iteration. It will continue program execution with the statement immediately after the **N** command. This input type is edge sensitive.

Data Input
This input type is used to load parallel bytes of data into the SX. The data is put on the SX inputs in BCD format. This input type is explained in more detail later in this chapter.

Memory Lock Input
This input type is used to lock out sequence editing commands and a couple others. These commands are **XE**, **XD**, **RIFS**, **CPE**, **CPG**, and **CPM**. The SX will report back what they are currently set to but will not allow them to be changed while the memory lock input is active.

Reset Input
This input type is used to execute a software reset of the SX. Upon recovery, the power-up sequence will execute, unless a serious fault condition still exists. This input is equivalent to the **Z** command. This input type is edge sensitive.

Point Zero Input
This input type is used to zero the absolute position counter. It is equivalent to the **PZ** command. This input type is edge sensitive.

User Fault Input
This input type is used to tell the SX that a fault condition exists external to the Indexer. It may come from a pushbutton, PLC, or other device. This fault condition is latched and must be cleared by cycling power, resetting the SX, or issuing the **ON** or **STØ** command. This input type is edge sensitive.

Data Valid Input
This input type is used in conjunction with data inputs when loading data from an external source. If an input is defined as data valid, the SX will not load information with one of the data entry commands (**DRD**, **VRD**, **LRD**, **FRD**, **TRD**, **VARD**, and **XRD**) unless the data valid input is active. A data read is attempted as often as the **STR** command setting allows. If no inputs are defined as data valid, the data is read in synchronously according to the **STR** command. This input function is explained in more detail later in this chapter.

Data Sign Input
This input type is used in conjunction with data inputs to indicate to the SX whether the data is a positive or negative value. This input is evaluated at the end of the data read and an active input will give a negative sign. If no input is defined as a data sign input the sign will default to positive.

Increase Following Ratio Input
This input type is used to increase the following ratio on the fly when following is enabled and a move is in progress. This input will increase the following ratio by the set **FIN** value every 1 msec that the input is seen as active.

Decrease Following Ratio Input
This input type is used to decrease the following ratio on the fly when following is enabled and a move is in progress. This input will decrease the following ratio by the set **FIN** value every 1 msec that the input is seen as active.

No Function Assigned
This input type is used to disable the input from any functions. It can be used to temporarily prevent an input from executing any functions even though the input is still being activated externally. It will remain this type until redefined.

To designate each input to a particular function, use the Set Input Functions (**IN**) command. To see what the inputs are currently defined as, type **1IN**. To see the inputs' states, use the **1IS** command. Enter the following commands as an example.

```
> 1IN
```

Change input 1 to be a stop input by entering: **> IN1D**

Check that it was assigned properly by again entering: **> 1IN1**

With this method, you can assign all the inputs to any of the input functions listed above.

Switches

This section contains information on SX triggers and sequence scanning with inputs. Refer to *Chapter 3, Installation* for more information on wiring the inputs to other equipment.

Triggers

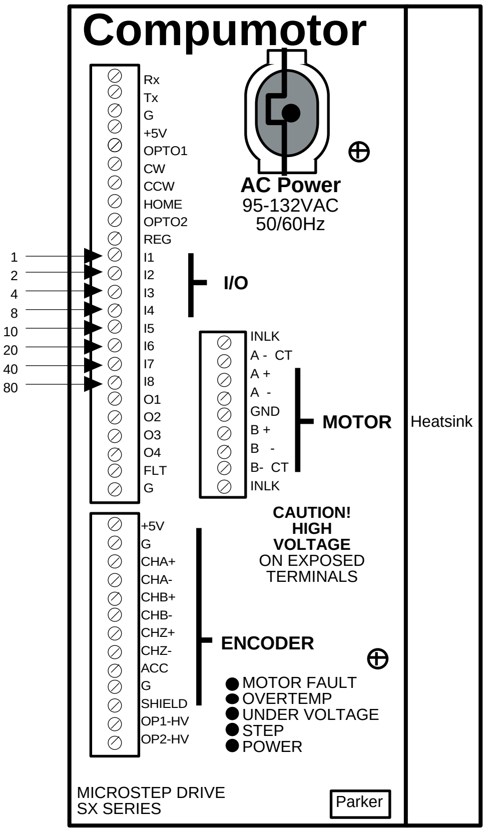
You can use the Wait for Trigger (**TR**) command to pause a sequence of buffered commands until one or more inputs reach a preferred state. Inputs **I1 - I8** are set (default setting) to function as trigger inputs.

Command	Description
> IN1D	Sets **I1** as Stop input
> IN2A	Sets **I2** as Trigger # 1
> IN3A	Sets **I3** as Trigger # 2
> IN4A	Sets **I4** as Trigger # 3
> MC	Sets the unit to Continuous mode
> MPP	Enters the Motion Profiling mode
> V1Ø	Sets velocity to 10 rps
> A15	Sets acceleration to 15 rps^2
> AD15	Sets deceleration to 15 rps^2
> TR1	Waits for trigger input 1 (**I2**)to be on
G	Executes a go (Go) command
L	Loops infinitely
V5	Sets velocity to 5 rps
TRXØ1	Waits for trigger input 2 (**I3**) to be off and trigger input 3 (**I4**) to be on
V1	Sets velocity to 5 rps
TRX1Ø	Waits for trigger input 2 (**I3**) to be on and trigger input 3 (**I4**) to be off
N	Ends the loop

This example program configures **I1** as a stop input and **I2**, **I3**, and **I4** as trigger inputs. The command execution will pause (be buffered) until the first **TR** condition is satisfied by activating Trigger #1 (**IN2**). Because the SX is in Motion Profiling mode, it will execute the loop and subsequent commands during the move. As it reaches each trigger statement, it waits for that input state to become true and then executes the commands following each **TR** command. The loop is infinite so it will continuously toggle between 1 rps and 5 rps as the trigger statements come true and will not stop until the stop input is activated. If you activate **I1** during the operation of the SX, the Indexer immediately decelerates the motor at 15 rps^2 and clears the command buffer.

Sequence Select & Sequence Scanning

Inputs can be defined as sequence-select inputs. This allows you to execute sequences defined via RS-232C and stored in the SX's memory, by activating the sequence-select inputs. Sequence-select inputs are assigned BCD (binary coded decimal) weightings. **The lowest input number assigned as a sequence-select input will have the least significant value.** The following figure shows the BCD weights of the SX's inputs when all 8 inputs are configured as sequence-select inputs.



BCD Weight of SX Inputs

The following table illustrates **one possible** input configuration and binary weighting.

Input	Function	BCD Weighting
I1	Sequence Select	1
I2	Trigger	—
I3	Stop	—
I4	Sequence Select	2
I5	Sequence Select	4
I6	Sequence Select	8
I7	Sequence Select	10
18	Sequence Select	20

Input Configuration/BCD Weighting Example

The **`IN`** command is used to configure an input as a sequence select input. For example, **`IN1B`** defines Input #1 as a sequence select input. **`IN5B`** defines input #5 as a sequence select input.

If the inputs are configured as in the following table (Inputs 1-8 all defined as sequence select inputs), Sequence #6 will be executed by activating Inputs #2 and #3. Sequence #19 will be executed by activating Inputs #1, #4, and #5.

Input	BCD Weight
1	1
2	2
3	4
4	8
5	10
6	20
7	40
8	80

BCD Weighting of Sequence Select Inputs

To execute sequences, the SX must be in Sequence Scan mode. In this mode, the SX will continuously scan the input lines and execute the sequence selected by the active sequence-select lines. The **SSJ** command is used to enable/disable the Sequence Scan mode. When **SSJ1** is entered, the Sequence Scan mode is enabled. To disable the mode, enter **SSJØ**. The sequence select inputs are not latched and are only looked at for sequence scanning when no other sequence is being executed.

Once enabled (**SSJ1**), the SX will run the sequence number that the active sequence-select inputs and their respective BCD weightings represent. After executing and completing the selected sequence, the SX will scan the inputs again and run the selected sequence. If a sequence is selected that has not been defined via RS-232C, no sequence will be executed.

If it is not desirable for the SX to immediately execute another sequence after running the currently selected sequence, the Sequence Interrupted Run mode (**XQ1**) can be enabled. In this mode, after executing a sequence, all sequence-select lines must be placed in an inactive state before a new sequence can be selected. The active state of the inputs is determined by the **INL** command.

The Scan (**SN**) command determines how long the sequence-select inputs must be maintained before the SXexecutes the program. This delay is referred to as **debounce time**.

The **SN** value also determines how long the inputs must remain inactive if in **XQ1** mode. *Increasing the* **SN** *value can help with bouncy switches and electrically noisy environments.* The **SN** value can also help when running higher sequence numbers and combinations of inputs need to be synchronized. The higher the **SN** value, the more time allowed for all of the inputs desired to be activated.

Step ①

The following example demonstrates how to use Sequence Scan mode with Sequence Interrupted mode and the **SN** command.

Define a power-up sequence. (Sequence #100 is always the power-up sequence.)

Command	Definition
> XE1ØØ	Erases sequence #100
> XD1ØØ	Defines sequence #100
SSJ1	Enables Sequence Scan mode
SN2Ø	Sets scan time to 20 msec
XQ1	Sets SX to Interrupted Run mode
A1Ø	Sets acceleration to 10 rps^2
AD1Ø	Sets deceleration to 10 rps^2
V2	Sets velocity to 2 rps
IN1B	Sets Input 1 as a sequence-select input
IN2B	Sets Input 2 as a sequence-select input
IN3B	Sets Input 3 as a sequence-select input
IN4B	Sets Input 4 as a sequence-select input
IN5B	Sets Input 5 as a sequence-select input
IN6B	Sets Input 6 as a sequence-select input
IN7B	Sets Input 7 as a sequence-select input
IN8B	Sets Input 8 as a sequence-select input
OUT1C	Sets Output 1 as sequence-in-progress output
LD3	Disables the limits
XT	Ends the sequence definition

Every time you power up the SX Indexer, it executes Sequence #100 and enables the SX to read up to 100 sequences from the sequence-select inputs.

Step ② Define any sequences that your application may require.

Command	Description
> XE1	Erases sequence #1
> XD1	Defines sequence #1
D2ØØØ	Sets distance to 2,000 steps
G	Executes the move (Go)
XT	Ends sequence #1 definition

Command	Description
> XE2	Erases sequence #2
> XD2	Defines sequence #2
D4ØØØ	Sets distance to 4,000 steps
G	Executes the move (Go)
XT	Ends sequence #2 definition

Command	Description
> XE3	Erases sequence #3
> XD3	Defines sequence #3
D8ØØØ	Sets distance to 8,000 steps
G	Executes the move (Go)
XT	Ends sequence #3 definition

Command	Description
> XE99	Erases sequence #99
> XD99	Defines sequence #99
D-14ØØØ	Sets distance to -14,000 steps
G	Executes the move (Go)
XT	Ends sequence #99 definition

Step ③ Verify that your programs were stored properly by uploading each entered sequence using the **XU** command (**1XU1 1XU2** etc.). If you receive responses that differ from what you programmed, re-enter those sequences.

Step ④ Run each program from the RS-232C interface with the Run Sequence (**XR**) command (**XR1**, **XR2**, etc.). Make sure that the motor moves the distance that you specify.

Step ⑤ Wire normally open switches to the inputs. Refer to *Chapter 3, Installation* for more information on wiring the inputs to other equipment.

Step ⑥ To execute sequences, cycle power to the SX. The system will execute sequence #100.

Step ⑦ You can now execute sequences by closing the corresponding switch or combination of switches.

❏ Close switch 1 to execute sequence #1

❏ Close switch 2 to execute sequence #2

❏ Close switches 1 & 2 to execute sequence #3

❏ Close switches 1, 4, 5, and 8 to execute sequence #99

Thumbwheel Interface

With the SX, you can use up to 16 digits of thumbwheels. The SX uses a multiplexed BCD input scheme to read thumbwheel data. Therefore, a decode circuit must be used for thumbwheels. Compumotor recommends that you purchase Compumotor’s TM8 Module if you want to use a thumbwheel interface. ***The following section assumes that you are using Compumotor’s TM8 module with the SX***.

Reading Parallel Data

The SX has seven parallel read commands that allow data to be read on inputs defined as Data Inputs. These commands are listed below:

Command	Description
> DRD	Read distance via thumbwheels
> VRD	Read velocity via thumbwheels
> LRD	Read loop count via thumbwheels
> TRD	Read time delay via thumbwheels
> VARDn	Read variables via thumbwheels
> XRD	Read sequence count via thumbwheels
> FRD	Read following ratio via thumbwheels (SX-F only)

The following section describes the method used to read parallel data and the different modes available (for a description of the individual commands, see the ***SX Software Reference Guide***). The software description of the parallel read commands assumes the SX is in **TW1** mode. The parallel read commands can be used in one of three modes selected by the **TW** command. A description of each mode is given below.

TWØ **TW0** mode reads two BCD digits at a time over the inputs. It requires that eight inputs be defined as Data Inputs, and at least one output be defined as a Strobe Output. Inputs 5-8 have a higher significance than inputs 1-4 in **TW0** mode. The number of times it reads a pair of BCD digits on the inputs is equal to the number of outputs defined as Strobe Outputs. The table below shows the output patterns during the strobing when all four outputs are Strobe Outputs. If fewer than four are used as Strobe Outputs, disregard the extra columns and rows in the table. (i.e. 2 outputs are Strobe Outputs   Refer to the first two columns and rows in the table). The time delay between the changes are specified by the Strobe Output Delay (**STR**) command.

During each strobe time, the proper BCD data should be put on the SX's Data Inputs. If only one digit of information is desired, only four inputs need to be Data Inputs and only one Output as a Strobe Output. In this configuration, another input could then be defined as a Data Valid Input to trigger the data read instead of the **STR** value.

Strobe Out #1	Strobe Out #2	Strobe Out #3	Strobe Out #4
0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

Ø = Inactive, 1 = Active

TW1 **TW1** mode is compatible with Compumotor's Thumbwheel Interface Module (TM8). It reads one BCD digit at a time over the data inputs. Four of the eight inputs must be defined as Data Inputs with the lowest numbered input having the least significance. Three of the four outputs must be defined as Strobe Outputs.

When a parallel read command is issued, the Strobe Outputs toggle in a binary pattern as shown in the table below. One digit is read in from the four lowest numbered Data Inputs for each binary pattern toggled on the outputs. The Strobe Outputs remain in each state for the amount of time specified by the Strobe Output Delay (**STR**) command and the inputs are read at the end of each strobe delay time. To allow for external or varying strobe rates, an input can be defined as a Data Valid Input. The SX will not read the inputs until it sees a data valid signal for each digit being read. In this case the **STR** value determines how long the data valid input must remain active to be seen.

TW1 mode also allows you to select a range of digits to be read and to provide a scale factor for the data. Three values can be added to the end of each command such as **DRDcde**. Variables c and d select the range of digits to be read from the inputs and may range from 0-7 to represent the desired thumbwheel digits. The TM8 Module's left most digit is 0 and the right most is 7. The variables c and d must satisfy the equation 0 c d 7.   The variable e scales the thumbwheel read value by 10^e. If any of the extra digits are to be used, all three must be specified.

Strobe #1

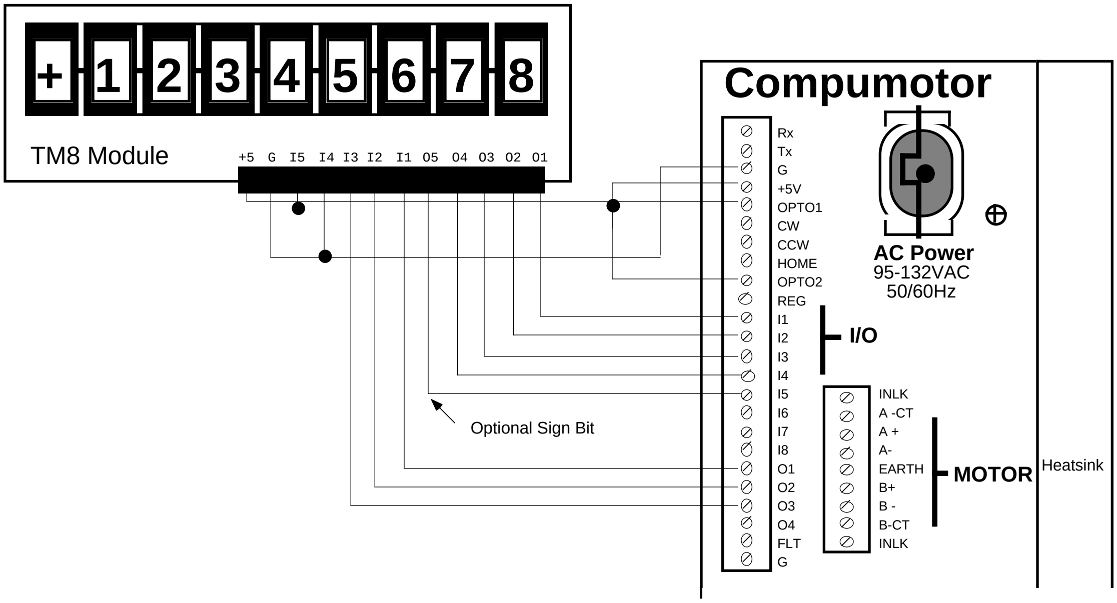
Strobe #2	Strobe #3	
1	1	1
0	1	1
1	0	1
0	0	1
1	1	0
0	1	0
1	0	0
0	0	0

Ø = Inactive, 1 = Active

TW2 **TW2** mode is identical to **TWØ** except that only one BCD digit is read at a time rather than two. Hence, only four inputs must be defined as Data Inputs and a maximum of four digits can be read, one for each output defined as a strobe output.

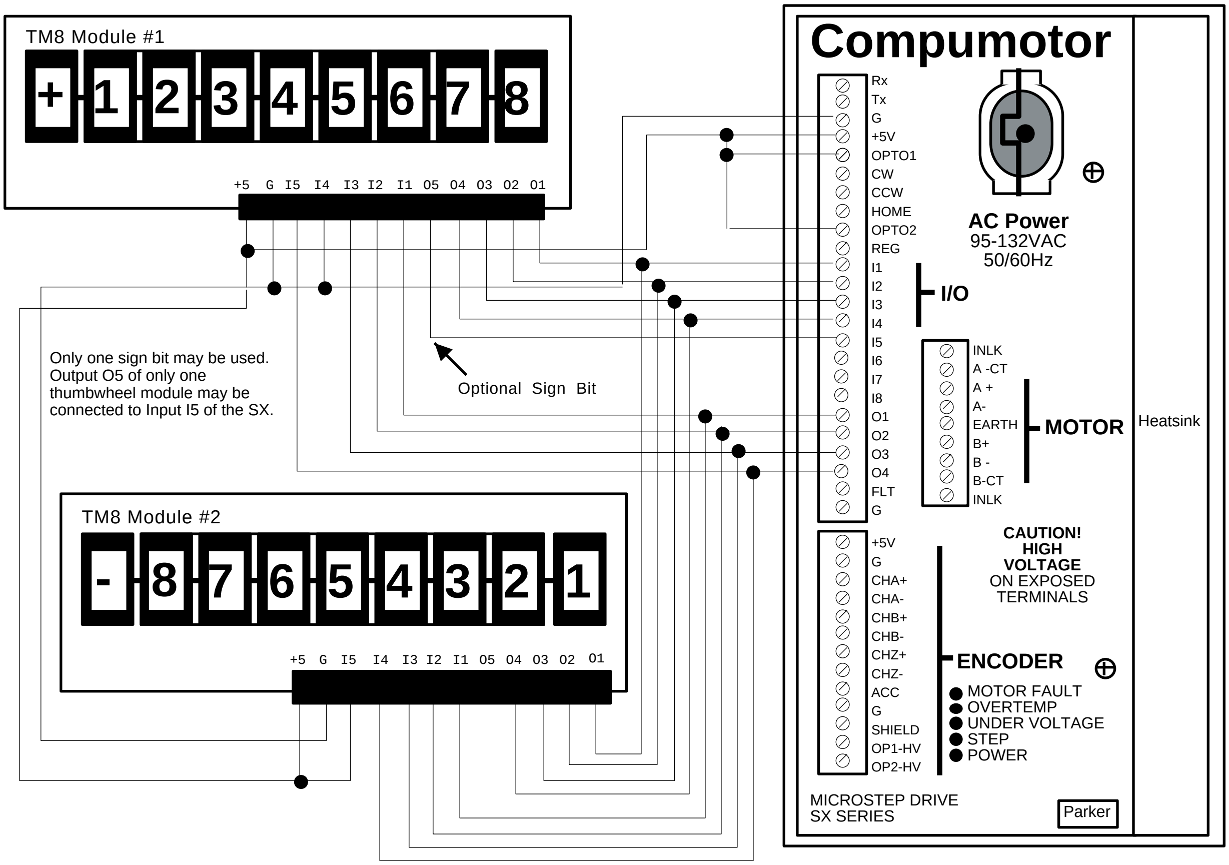
When reading digits from the Thumbwheel the most significant number (digit Ø) is read in first. The output strobe pattern is the same as for **TWØ** mode. Refer to the previous table for the outputs you are using. The following is an example using one or two TM8 Modules.

Step ① If you are using one TM8 Module, wire your module to the SX as shown in the first figure. If you are using two TM8 Modules (the maximum allowed per **each** SX), wire the modules as in the second figure.



Wiring 1 Thumbwheel Unit (TM8) to the SX

If you are powering an encoder with 5V from the SX along with one or two TM8 Modules, an external 5V supply must be used to power OPTO1 and OPTO2 or the encoder.



Only one sign bit may be used. Output O5 of only one thumbwheel module may be connected to Input I5 of the SX.

Optional Sign Bit

Wiring 2 Thumbwheel Units (TM8) to the SX

Step ② Configure your SX as follows:

Command	Description
> OUT1J	**O1** configured as a strobe
> OUT2J	**O2** configured as a strobe
> OUT3J	**O3** configured as a strobe
> IN1N	**I1** configured as a data input
> IN2N	**I2** configured as a data input
> IN3N	**I3** configured as a data input
> IN4N	**I4** configured as a data input
> IN5W	**I5** configured as a sign input (optional)
> INLØ	Inputs configured active low
> STR5Ø	Data strobe time of 50 ms per digit read. If using one TM8 Module, you should now be ready to read in thumbwheel data. If using two TM8 Modules, enter the additional set-up commands. **Minimum recommended strobe time for the TM8 module is 10 ms.**
> OUT4A	**O4** configured as a programmable output
> OUTLØ	Outputs set active low
> O1	Set output 4 high—this enables TM8 Module #1

Step ③ Set the thumbwheel digits on your TM8 Module to **+12345678**. If using two TM8 Modules, set the second module to **-87654321**. To verify that you have wired your TM8 Module(s) correctly and configured your SX I/O properly, enter the following commands:

Command	Description
> DRD	Request distance data from all 8 thumbwheel digits
> 1D	Displays the distance read—**D+0012345678.** If you do not receive the response shown, return to step #1 and retry.

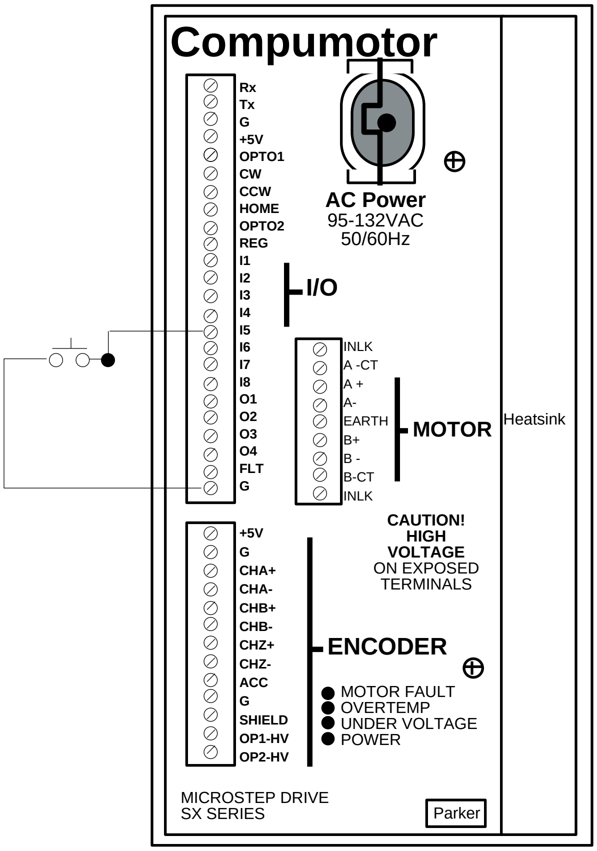
If you are using two TM8 Modules, enter the following commands:

Command	Description
> OØ	O4 becomes 0V—this disables Module #1 and enables Module #2.
> DRD	Request distance data from all 8 thumbwheel digits
> 1D	Displays the distance read**–D-0087654321** *The sign is positive.*—only one sign digit may be used when two TM8 Modules are used. If you do not receive the response shown, return to step #1 and retry.
> O1	Re-enables the first TM8 Module.

Selecting Sequences with Thumbwheel Module

The following example shows how the SX is often used with thumbwheels. In the example, only three sequences are entered. As many as 100 sequences may be defined and up to 100 may be executed with the TM8 Module. Sequence #100 is automatically executed during power-up, reset, or by **XR1ØØ**. Refer to the Reset (**Z**) command in the ***SX Software Reference Guide***.

Ensure that the thumbwheel module is properly installed (as shown in the previous figures). Wire input 5 as shown in the following figure. The switch shown in this configuration is a data valid switch. Note that the sign bit is not being used.



Sequence Start Configuration

Step ① Define a power-up sequence. Set inputs **I1** - **I4** as data inputs and **I5** as a data valid input. Enter the following program.

Command	Description
> XE1ØØ	Erases sequence #100
> XD1ØØ	Begins definition of sequence #100
IN1N	Sets I1 as a data input
IN2N	Sets I2 as a data input
IN3N	Sets I3 as a data input
IN4N	Sets I4 as a data input
IN5V	Sets I5 as a data valid input
INLØ	Sets ØV as active level
STR1Ø	Sets strobe time of 10 ms per digit
OUT1J	Sets O1 as a strobe output
OUT2J	Sets O2 as a strobe output
OUT3J	Sets O3 as a strobe output
OUTLØ	Outputs set active low
L	Start a continuous loop
XRD67Ø	Run the sequence displayed on thumbwheel digits 6 & 7
N	Ends loop
> XT	Ends definition of sequence #100

Step ② Define any sequences that your application may need.

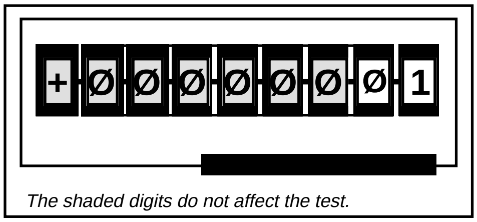
Command	Description
> XE1	Erases sequence #1
> XD1	Begins definition of sequence #1
MN	Sets to Normal mode
A25	Sets acceleration to 25 rps^2
AD25	Sets deceleration to 25 rps^2
V5	Sets velocity to 5 rps
D25ØØØ	Sets distance to 25000 steps
G	Executes the move (Go)
> XT	Ends definition of Sequence #1
> XE2	Erases sequence #2
> XD2	Begins definition of sequence #2
MN	Sets to Normal mode
A25	Sets acceleration to 25 rps^2
AD25	Sets deceleration to 25 rps^2
V5	Sets velocity to 5 rps
D1ØØØØ	Sets distance to 10000 steps
G	Executes the move (Go)
> XT	Ends definition of sequence #2
> XE99	Erases sequence #99
> XD99	Begins definition of sequence #99
MN	Sets to Normal mode
A25	Sets acceleration to 25 rps^2
AD25	Sets deceleration to 25 rps^2
V5	Sets velocity to 5 rps
D-35ØØØ	Sets distance to 1000 steps
G	Executes the move (Go)
> XT	Ends definition of sequence #99

Step ③ Reset the SX. This will execute the power-up sequence (Sequence #100).

Command	Description
`> Z`	Resets the SX

Step ④ Set thumbwheel digits 7 and 8 to **Ø1** and activate the Data Valid Input (#5) to move 25,000 steps CW.

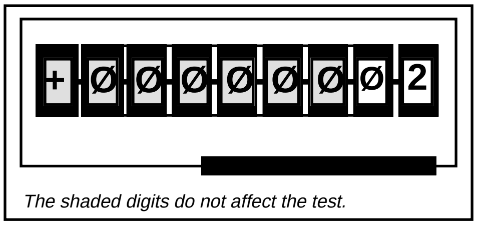
(Execute Sequence #1)



The shaded digits do not affect the test.

Step⑤ Set thumbwheel digits 7 and 8 to **Ø2** and activate the data valid input (#5) to move 10,000 steps CW.

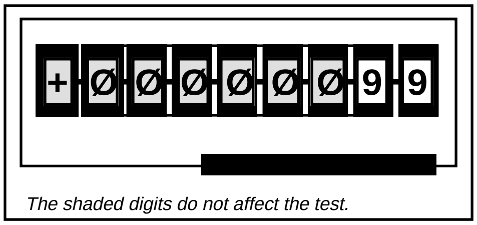
(Executes Sequence #2)



The shaded digits do not affect the test.

Step ⑥ Set thumbwheel digits 7 and 8 to **99** and activate the data valid input (#5) to move 10,000 steps CCW.

(Executes Sequence #99)



The shaded digits do not affect the test.

If you select an invalid or unprogrammed sequence with the thumbwheels, no motion will occur and if the **SSN** command is enabled you will get the error message:

```
*UNDEFINED_SEQUENCE
```

PLC Operation

This section explains and provides examples of how to use a PLC with the SX.

Interfacing with a PLC

In many applications, it is desirable to interface to a PLC. The SX performs the motion segment of a more involved process controlled by a PLC. In these applications, the PLC will start sequences, load data, manipulate inputs, and perform other specific input functions to control the SX and the motion segment of a process. This section assumes the SX is in **TW1** mode. If **TWØ** or **TW2** mode is desired, refer to the Thumbwheel section for more details.

Parallel Data Read with a PLC

As in the thumbwheel case, the PLC can be used to enter data for sequence-select (**XRD**), distance (**DRD**), velocity (**VRD**), loop count (**LRD**), time delay (**TRD**), variable data (**VARD**), and Following Ratio (**FRD**). Refer to the ***SX Software Reference Guide*** for more details on these commands.

To read data from the PLC, four of the SX's inputs must be configured as data inputs. If a sign digit is required, an input should be configured as a sign input. Configure 3 outputs as data strobe outputs. The active level of the inputs can be changed using the **INL** command. The active level of the outputs can be changed using the **OUTL** command.

When the SX executes a data read command, it will cycle its outputs and read BCD data as shown in the following table. The strobe outputs tell the PLC what digit of data to put on the SX inputs for each step in the process.

The Strobe Output Delay Time (**STR**) command sets the maximum rate that the SX will cycle through the output levels. The SX synchronously cycles through the states at the **STR** time if no data valid line is used. If a data valid line is used, the SX maintains its current state until the data valid input is activated. This allows the PLC to control the SX's data strobe rate.

The following table shows inputs 1-5 and outputs 1-3 being used. Any combination of inputs/outputs may be used. Their significance is always from the lowest numbered one to highest numbered one.

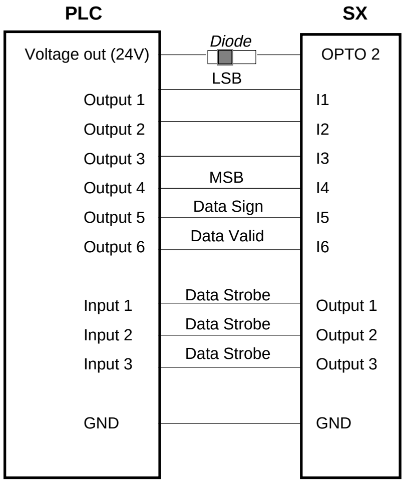
SX Outputs			Corresponding Digits Read	SX Inputs				
01	**02**	**03**	**Data In (Active Low)**	**l1**	**l2**	**l3**	**l4**	**l5**
1	1	1	MSD (Digit 1)	lsb		msb	± sign bit	
0	1	1	Digit 2	“		“	“	
1	0	1	Digit 3	“		“	“	
0	0	1	Digit 4	“		“	“	
1	1	0	Digit 5	“		“	“	
0	1	0	Digit 6	“		“	“	
1	0	0	Digit 7	“		“	“	
0	0	0	LSD (Digit 8)	“		“	“	

Ø = Inactive, 1 = Active (Connection to GND)

Data Strobing Table

Simply put, the SX executes a parallel read command and if there is no data valid input defined, places the first strobe pattern on its outputs and waits a time specified by the **STR** value. At the end of this time it reads the four lowest numbered Data Inputs, places the next strobe pattern on its outputs, and again waits for the **STR** value before reading its Data Inputs. This pattern repeats until all of the strobe patterns have been executed. If there is a Data Valid Input, the SX sees the Data Valid Input go active before each input read and strobe pattern change.

This figure shows a possible PLC-to-SX connection that would allow the SX to input data from the PLC. A data valid line controls the SX's strobe rate. A data sign line is also used.



PLC/SX Connnection

CAUTION

If voltages of 13VDC - 24VDC are used on OPTOs 1 & 2, a zener diode should be placed in series with the DC voltage supply and the OPTOs. If your PLC outputs are at a 13-24 Volt level a zener diode should be placed on each SX input with the same polarity as the OPTO2 diode shown in the previous figure. This will prevent reverse biasing and possibly damaging the SX inputs. Refer to Chapter 6, Hardware Reference for diode specifications.

If the SX executes a Read Distance Via Parallel I/O (**`DRD`**) command, the following events must occur to transfer distance data from the PLC to the SX. This is assuming **`INL`**, **`OUTLØ`**, and the same inputs/outputs as used in the previous figure.

① The SX executes a **`DRD`** command and places its outputs #1 - #3 at ØV.

② The PLC places and holds a BCD digit at the SX's inputs #1 - #4. (This value will be the most significant distance digit). The PLC places a sign value at SX input #5. (This sign bit **must be the same for each digit read**.)

③ When the data is valid, the PLC should activate the data valid line for (**`STR`**) milliseconds. The SX will read the digit and sign values.

④ The PLC must deactivate the data valid line.

⑤ After reading a data valid, the SX will place its output #1 in an inactive state and outputs #2 and #3 at ØV.

⑥ The PLC must place and hold its second BCD digit at the SX's inputs #1 - #4. The PLC should place the same sign value as that given for input #5.

⑦ When the data is valid, the PLC must activate the data valid line for (STR) milliseconds. The SX will read this second BCD digit as the second significant distance digit.

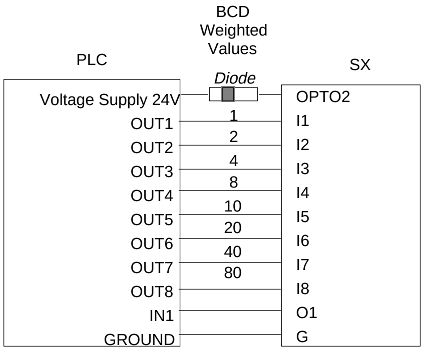
⑧ The PLC must deactivate the data valid line.

This process continues until the SX reads the eighth digit (LSD). At this point, the SX enters the eight digits read into its distance register and proceeds with the execution of subsequent commands.

Sequence Select With a PLC

The PLC can execute sequences through two different methods. First, the sequences may be selected by using the inputs defined as Sequence-Select Inputs. In this case, the input lines are BCD weighted (refer to the following figure). The SX must be in Sequence Scan mode (**`SSJ1`**) and may either operate in Interrupted mode (**`XQ1`**) or Continuous Scan mode (**`XQØ`**). Refer to the earlier switch and thumbwheel discussions for more information on Sequence Select methods.

The PLC activates the lines that will execute the desired sequence. It may be desirable to have the SX indicate to the PLC when it has completed a sequence or to indicate to the PLC when it should select another sequence. A programmable output can be used for this handshake to the PLC.



PLC Connection

CAUTION

If voltages of 13VDC - 24VDC are used on OPTOs 1 & 2, a zener diode should be placed in series with the DC voltage supply and the OPTOs. If your PLC outputs are at a 13-24 Volt level a zener diode should be placed on each SX input with the same polarity as the OPTO2 diode shown in the previous figure. This will prevent reverse biasing and possibly damage the SX inputs. Refer to Chapter 6, Hardware Reference for diode specifications.

Miscellaneous Control by a PLC

You can use a PLC to control the activation of the inputs for many of the input functions that the SX supports (see the **Programmable Input** section). For example, you can use the PLC to stop, kill, go, go home, or reset the SX.

RP240 Operator Panel

Refer to the ***RP240 User Guide*** for information on using it with the SX.

Rotary vs. Linear Indexers

Most Compumotor Indexers are used for rotary motor systems. Hence, velocities and accelerations are selected in rps and rps^2 respectively. The default is often 25,000 steps per revolution. For linear motors, acceleration and velocities are usually defined in g's and inches per second (ips) respectively. Use the following equation to convert rps^2 to g's (1g = 386 ips^2).

$$A[g] = \frac{A[rps^2] \bullet \text{Rotary Resolution [steps/rev]}}{\text{Linear Resolution [steps/in]} \bullet 386\ ips^2}$$

For example, if the rotary resolution is 25,000 steps/rev, the acceleration value is 100 rps^2, and the linear resolution is 10,000 steps/in. The equation is as follows:

$$\frac{100\ [rps^2] \bullet 25000\ \text{[steps/rev]}}{10000\ \text{[steps/in]} \bullet 386\ ips^2} = 0.648\ g$$

Use the following equation to convert rps to ips:

$$V[ips] = \frac{V[rps] \bullet \text{Rotary Resolution [steps/rev]}}{\text{Linear Resolution [steps/in]}}$$

For example, if the resolutions are the same as defined above, and the velocity value is 1 rps, the equation would be as follows:

$$\frac{1\ [rps] \bullet 25000\ \text{[steps/rev]}}{10000\ \text{[steps/in]}} = 2.5\ [ips]$$

☞ ***Helpful Hint:***

Rotary *vs* Linear Indexer Example

① Set the unit with the following move parameters:

- ❏ Acceleration = 1000 rps^2
- ❏ Velocity = 1 rps
- ❏ Distance = 10,000 steps

② Execute the **G** (Go) command:

If the resolution is 25,000 steps/rev, the forcer should move 1 inch at a velocity of 2.5 ips.

CHAPTER ⑤

SXF Follower

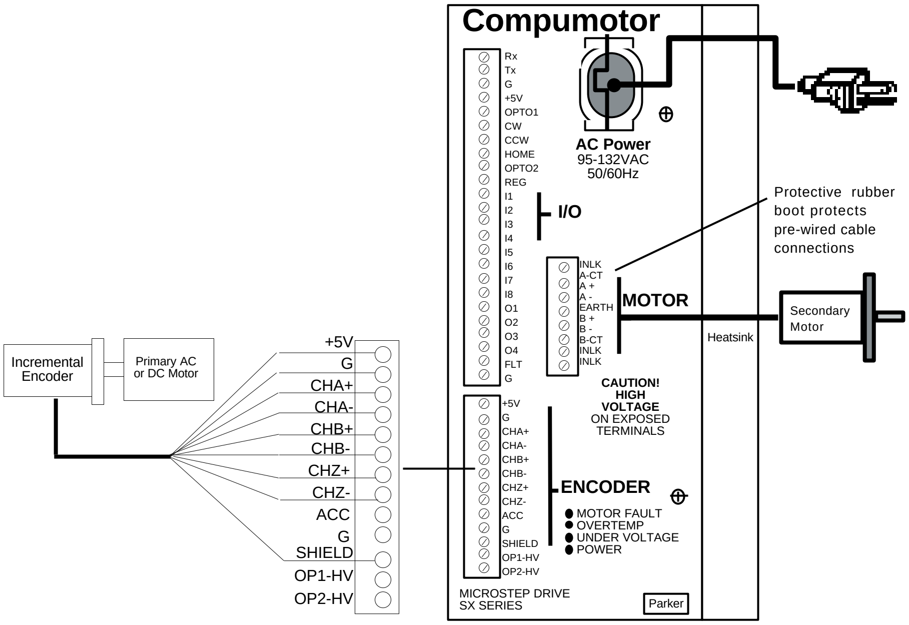
Chapter Objectives

The information in this chapter will enable you to:

- ❏ Understand basic following concepts
- ❏ Understand the basic types of following and their common applications
- ❏ Become familiar with the SXF commands associated with following

What is Following?

The SXF can perform velocity following and distance following moves. The SXF can follow from an incremental or absolute encoder input. *Unless otherwise noted,* ***all the features that will be presented or have been discussed are also valid in Following mode****.* ***The only difference is that you replace the velocity command with a speed ratio and the acceleration with a following acceleration for distance following.*** *Instead of specifying the speed with the* **`V`** *command, you will specify the speed with respect to the primary using the* **`FOL`** *command.* The following figure shows the basic following system configuration.



Typical Following System Configuration

In the previous figure, the primary axis is an AC or DC motor. The encoder is mounted on the shaft of the primary motor. The encoder provides the SXF with data on the primary motor's position and velocity. The SXF uses the position and velocity data to move the secondary axis. Therefore, the secondary motor is *following* the motion of the primary motor. The SXF controls the motion of the secondary motor according to the motion of the primary. This concept of following can be used in different forms to satisfy many different applications.

Types of Following

There are four types of Following application motion.

❏ Velocity following
❏ Velocity and position following
❏ Recede and advance while following (includes electronic cam)
❏ Phase error correction (synchronization)

The order in which the categories are presented are roughly in order of increasing complexity. By identifying how the secondary axis motion relates to the primary axis, you will be able to determine your application's type and the applicable programming commands. The SXF also provides several addition features that have been added to address more specific attributes of the various applications. The SXF has several additional features that enhance the functionality for an application.

❏ Registration while following
❏ Jogging in the following mode
❏ Following a pulse train and a direction
❏ Following encoder pulses and the encoder direction
❏ Entering following ratio via thumbwheels
❏ Capability of loading the primary encoder position into a variable

The SXF can perform all of the SX's programming functions. The only difference is that the motion profiles are now following moves and motion is based on a primary axis. The SXF can also be used in its Indexer mode. You can easily switch back and forth from follower and Indexer functionality and create motion programs in which the SXF serves as both a follower and an Indexer. The SXF has the following programming capabilities:

❏ Variables, general-purpose & read-only (position, primary encoder position, etc.)
❏ Math
❏ Complex branching—**`IF ELSE`**, **`REPEAT UNTIL`**, **`WHILE`**
❏ On-the-fly changes—**`MPP`** mode
❏ Flexible I/O

For an explanation of the features that are common to both the SX and the SXF, refer to *Chapter 4, Application Design*. You should understand how to set the SXF for following. You can use one command (**`FSI`**) to enter and exit the Indexer and Following modes.

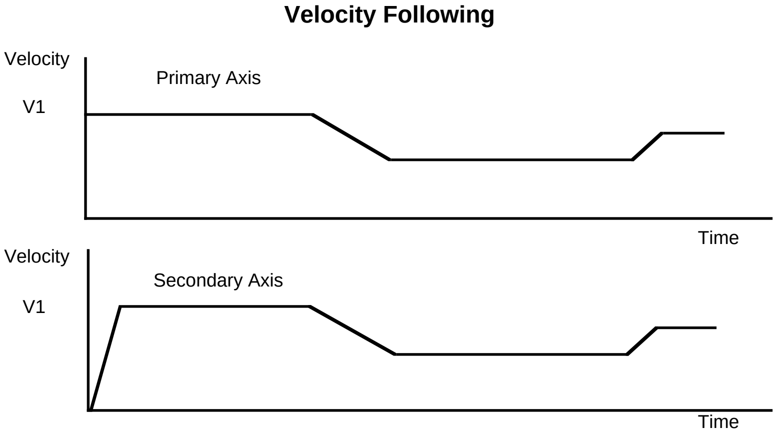
Command	Description
`FSI1`	Enters the following mode
`FSIØ`	Exits the Following mode(Indexer mode)

The rest of this chapter explains the four following types. Additional features of SXF following will also be covered.

Velocity Following

In ***velocity following***, the secondary axis is concerned only with the primary axis' speed. The relationship of the primary axis position with respect to the secondary axis is irrelevant. In this type of application, the secondary axis accelerates at a specified acceleration up to a ratio of the primary axis' speed. Preset or continuous moves are performed in the same manner as in Indexer mode. Exact distances on the secondary axis can be moved in Preset mode. However, instead of moving at a velocity specified by the **`V`** command, the secondary axis moves at a ratio of the primary axis' velocity specified by the **`FOR`** and **`FOL`** commands. The acceleration is independent of the primary axis encoder and is specified by **`A`** and **`AD`** (as with an Indexer).

The following figure shows velocity following. Once the secondary axis accelerates to the specified following ratio, it tracks the primary axis' speed and position at the specified ratio. During acceleration, the primary encoder speed and position are not followed. The acceleration ramp is independent of the primary axis. Once the secondary axis has accelerated to the specified following ratio, it will be following the primary axis (if the primary axis slows down, the secondary axis slows down).

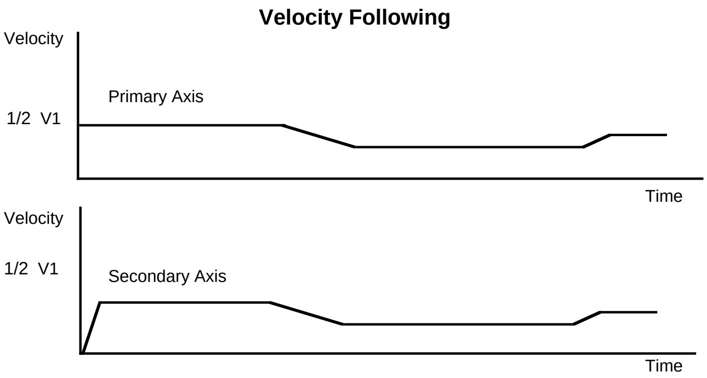


Velocity Following

Velocity following is used in dispensing and on-the-fly cutting applications.

In each of the following categories described, the secondary axis only moves in the direction specified by either the **D** command, the **V** command, or the **H** command. If the primary axis changes direction, the secondary axis will still move in the same direction. Only the number of pulses and the rate of the pulses determines the secondary axis' motion. With position and direction tracking enabled, the secondary axis will follow the primary axis' direction. ***To illustrate velocity following, detach the encoder from the primary axis and manually move the encoder***.

In the previous figure, if the primary velocity changed after the following ratio was achieved, position and velocity were tracked exactly. If the primary axis' velocity were half as fast as the example in next figure, a different phase or positional relationship would result during acceleration from rest. This is why it is velocity following. ***Again, once the secondary axis accelerates to the specified following percentage, it will follow velocity and position exactly***.



Primary Axis as Half Speed with Velocity Following

Setting Up Velocity Following

To perform velocity following, you must use two commands.

Command	Description
FOR	Relates the number of secondary motor steps per unit of travel to the corresponding primary encoder steps per unit of travel.
FOL	Relates the speed of the secondary axis to the speed of the primary axis as long as **FOR** is set correctly. The value is entered as a percentage of the primary axis speed.

The SXF uses the equation below to determine the number of motor steps.

$$\text{Motor Steps} = \mathbf{FOR} * \frac{\mathbf{FOL}}{100} * \text{Encoder Steps}$$

These two commands, (**FOR** and **FOL**) remove the complication of having a different resolution for measurement of distance on the primary and secondary axes. Once you relate the number of primary encoder pulses per unit of travel to the number of secondary motor pulses per that same unit of travel, you can relate their speeds. You can specify the secondary axis to move at 50% of the speed of the primary axis (1:2 ratio) or 100% of the speed (1:1 ratio). The encoder is usually mounted on the primary axis or through gearing and will measure a certain number of pulses per inch. The secondary axis would also have some certain number of pulses per inch. The ratio of the secondary axis steps to the primary axis' steps is entered in the **FOR** command. If the primary axis moves at 10 inches per second (ips) and the secondary axis is also set to move at 10 ips, a speed percentage of 100 is all that needs to be programmed (**FOL1ØØ**). ***The remaining commands are standard SX commands.*** Use the following commands to implement the velocity following feature:

Step ① Enter the number of secondary motor steps per unit of travel per number of primary encoder step per the same unit of travel. In this example, the motor step resolution is 25,000 steps/rev and the encoder resolution is 4,000 steps/rev. The unit of travel is 1 revolution—the ratio is 25,000/4,000 or 6.25.

Command	Description
> FOR6.25	Sets the motor step to encoder step ratio of 6.25
> OFF	Turns the SXF off
> CMR25ØØØ	Sets motor resolution to 25000 steps/rev
> ON	Turns the SXF on

Step ② Enter the following mode with the FSI command.

Command	Description
> FSI1	Enters the Following mode

Step ③ Set the speed ratio. If the secondary axis is to move at the same speed as the primary axis (100% of the primary axis' speed), enter **FOL1ØØ.**

Command	Description
> FOL1ØØ	The secondary axis will move at 100% of the primary axis' speed

Step ④ Set the SXF to .i. continuous mode and begin motion with the G command.

Command	Description
> MC	Enters Continuous mode
> A5ØØ	Sets acceleration
> AD5ØØ	Sets deceleration
> G	Initiates motion

Turn the encoder. The SXF should begin moving at the same speed that you are turning the encoder. Change the direction that you turn the encoder and note the motion of the secondary axis. Follow the steps below.

Step ① Execute the following commands.

Command	Description
> S	Stops the continuous motion command issued above
> 1FSP1	Enables the Position and Direction Tracking mode
> G	Initiates motion

Step ② Now turn the encoder or move the primary so that the encoder moves. The SXF will begin moving. Change the direction that the encoder is moving. The SXF will change direction. To change the relative direction between the SXF and the encoder, use the .i.change direction; command (H±). Now turn the encoder and note that the SXF's relative direction has changed. Tracking the direction as well as pulses can only be used in continuous mode. Preset moves can also be performed in velocity following.

Preset Moves In Velocity Following

A preset move is performed as in the standard Indexer mode. Issue the mode normal command (**`MN`**) and specify a distance in terms of secondary motor steps. In a preset move:

① The secondary axis accelerates (**`A`**) to the desired speed percentage.
② It decelerates at the rate set with the **`AD`** command.
③ It moves the specified distance (**`D`**) and stop.

If the primary axis' speed varies, the acceleration ramp will be the same, but the distance that the secondary axis travels to reach its following ratio will be different. The positional relationship or phase relationship is not maintained during acceleration. Once the secondary axis achieves the following percentage speed, it will track both velocity and position exactly.

Step ① Attach the encoder to the primary axis. Start the primary axis moving.

Step ② Enter the following set of commands.

Command	Description
`> MN`	Enters Normal mode
`> FSPØ`	Exits the direction Tracking mode
`> A5ØØ`	Sets acceleration
`> AD5ØØ`	Sets deceleration
`> FOL1ØØ`	Sets the following speed percentage to 100
`> D125ØØØ`	Sets the distance to 125000 steps
`> G`	Initiates motion

The secondary axis will move 125000 motor steps at the same speed as the primary axis. Repeat the example, but vary the speed of the primary axis. ***The secondary still moves 125000 steps, and its speed varies with the primary axis***.

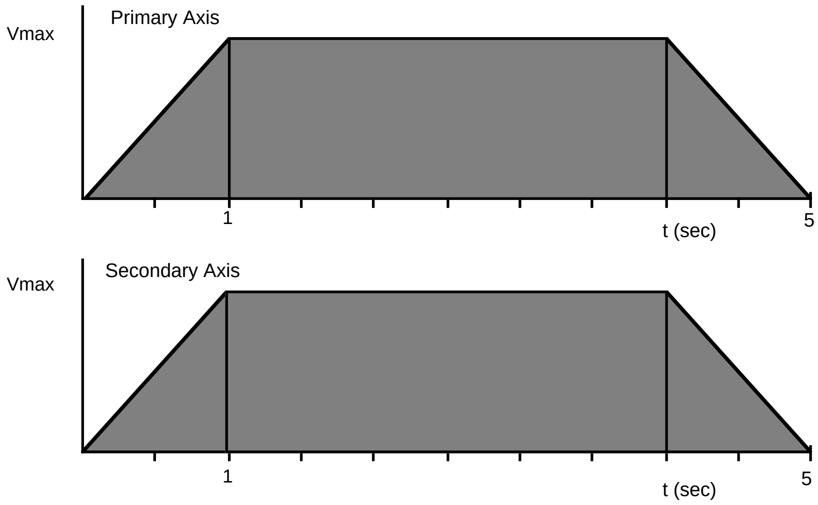
Velocity following is the simplest form of following. All motion in the SXF is programmed in the same manner as the SX. The only exception is that the velocity command is replaced by a following percentage.

Position and Velocity Following

Position and velocity following is the most common form of following. In this case, the secondary axis must maintain a specific positional relationship with the primary axis. The primary axis moves a specific number of primary encoder pulses while the secondary axis must move a specified number of secondary motor steps. The secondary axis moves at the same speed as the primary axis (a 1:1 speed ratio) in most of these applications. ***Coil winding applications, however, are an exception***. In coil winding applications, the relationship between the primary (usually the spindle axis) and secondary axes (usually the traverse axis) is based on a desired pitch. This pitch defines a positional and velocity relationship that will have an arbitrary speed relationship based on the particular application. The positional relationship is usually defined by a single traverse corresponding to a specific number of spindle axis turns, thus defining a specific number of primary encoder pulses while a specific number of secondary pulses must be traveled.

Primary Axis at Rest

There are two variations of ***position and velocity following***. The difference is in the motion of the primary axis. The less common form starts the primary axis motion from rest. The secondary axis must follow the primary axis pulse for pulse. In this type of application, the secondary axis will be in a continuous move and will track the motion of the primary axis exactly. A *web positioning system* where two axes are guiding the web (one edge is the primary axis and the other edge is the secondary axis), is an example of such an application. The web can be positioned based on the motion relationship between the primary and secondary axes. The following figure shows the profiles of a secondary axis following a primary axis pulse for pulse.



Pulse-for-Pulse Following

If the primary axis starts from rest, the secondary axis must track it pulse for pulse. **FSA** enables this capability. If position tracking is disabled, the SXF will follow the pulse count only and will not change direction if the primary axis changes direction. This is acceptable in many applications. In some applications, however, the primary axis may overshoot when it comes to a rest and the encoder will change directions.

Since the SXF is following the primary axis' pulse count only, it will actually move the secondary axis too many pulses. By enabling Position Tracking (**FSP1**), you can you can track the primary encoder's direction and pulse count and therefore not accumulate excess pulses caused by overshoot.

Command	Description
FSA1	Enables Pulse Tracking mode where secondary instantaneously accelerates between commanded velocities
FSP1	Secondary tracks both the direction & pulse count from the primary encoder

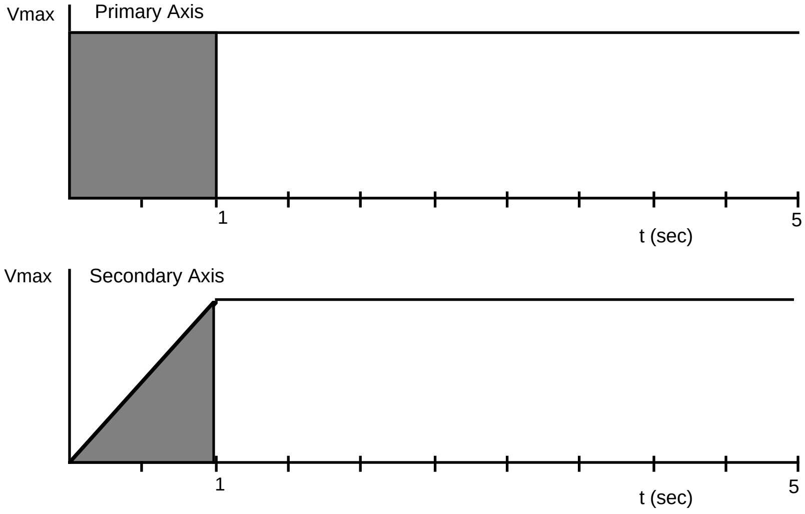
You must disable the Following Synchronized Acceleration mode (**FSFØ**) and activate Continuous Move mode to use Pulse Tracking mode (**FSA1**). **FSAØ** disables Pulse Tracking mode.

Primary Axis in Motion

The most common form of ***position and velocity following*** begins with the primary axis already in motion. The secondary axis must accelerate up to the specified speed ratio of the primary axis. In this type of application, the secondary axis must accelerate to a known positional or phase relationship with the primary axis. The primary axis is usually a conveyer or web and the secondary axis is performing an operation on the web or parts on the conveyer. The primary axis is always moving so the secondary axis must be moving at the same speed and with the correct orientation to perform some operation on the moving primary axis. To maintain positional and velocity relationships, the secondary axis must be able to accelerate over a known distance with respect to the primary axis (i.e., following acceleration is needed). The Set Following Synchronization Rate (**FAC**) and Set Following Synchronization Count (**FEN**) commands are used in conjunction with the velocity following commands.

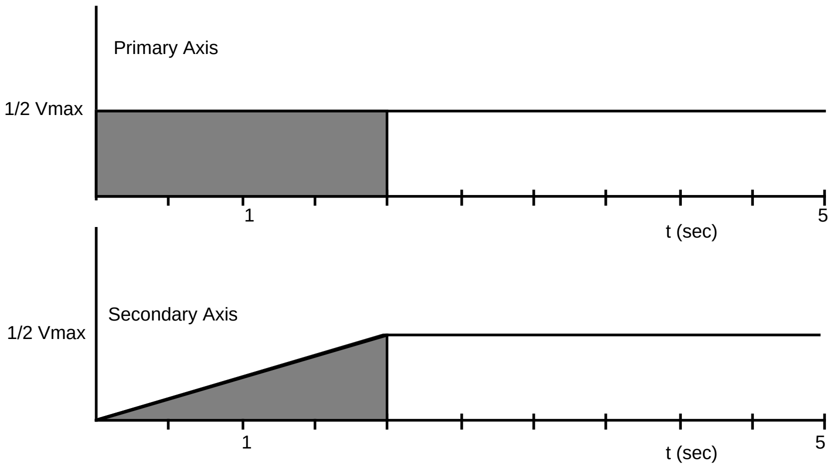
The **A** and **AD** commands used in velocity following are replaced by a following acceleration. A following acceleration is accomplished by stepping through subsequent ratios from the present following ratio to the final following ratio (**FOL**). The increment between ratios is set by the **FAC** command. Incrementing from one ratio to the next is based on the primary encoder changing by a set number of primary encoder pulses. This number of pulses is set by the **FEN** command. For example, if the value for **FAC** is 1, the value for **FEN** is 10, and the final following speed percentage is 100 (**FOL**), the secondary axis must accelerate from zero speed to 100% of the primary speed. From rest, every 10 primary encoder steps (**FEN**) the following percentage will change by an increment of 1% (**FAC**) until it is equal to the final following percentage of 100% (**FOL**). Therefore, the secondary axis will accelerate over 1000 primary encoder steps. It does not matter if the speed of the primary axis varies, the secondary axis' acceleration is based on primary encoder steps (not time).

The following figures illustrate position and velocity following. The shaded area indicates the distance moved by both the primary and secondary axes while the secondary axis is accelerating. ***The secondary axis moves 1/2 the distance that the primary axis moves***. This will always be the case when the secondary axis accelerates from rest to the same speed as the primary axis, when the primary axis is already moving.



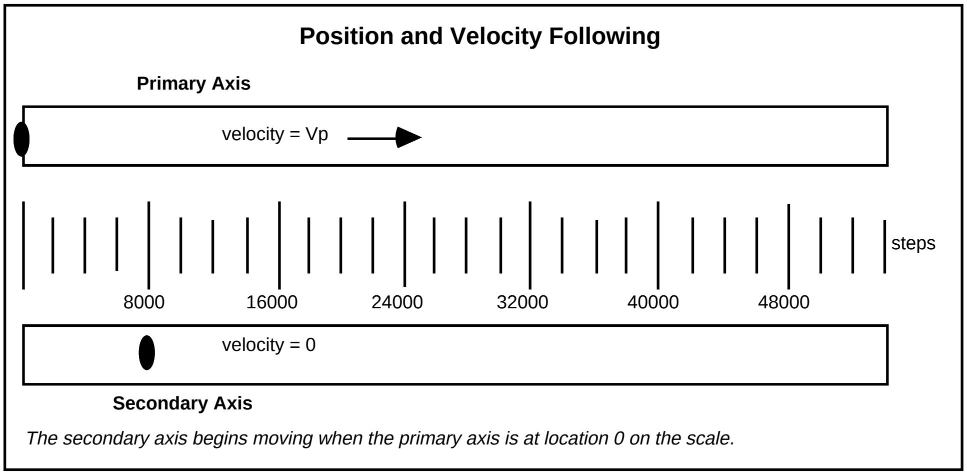
Following Acceleration

This figure shows how the acceleration ramp varies if the primary axis' speed varies. If the primary axis' speed is 1/2 the maximum velocity, the acceleration ramp will be twice as long. The shaded portions in the following figure are equivalent to the shaded portions in the previous figure. ***The primary axis' velocity can change at any time, even during the secondary axis' acceleration ramp, without changing the positional relationship***.



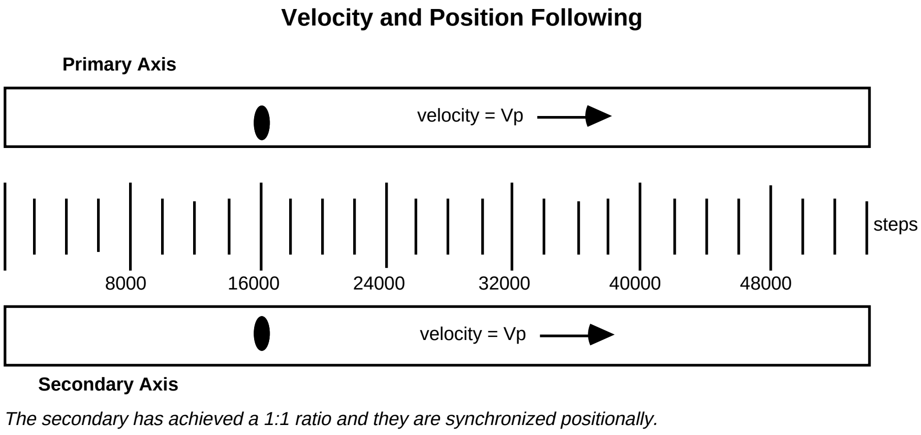
Following Acceleration With Primary Axis Velocity Change

The shaded area for the secondary axis is 1/2 of the shaded area under the primary axis curve. When the primary axis is moving and the secondary axis must start from rest and accelerate to the primary axis' velocity, the secondary axis will always move 1/2 the distance. Using following acceleration, the secondary axis adjusts its acceleration according to the primary axis' velocity so that it will always accelerate over the same distance while the primary axis moves a specified distance. To synchronize secondary and primary axes' positions, the secondary axis must start ahead of the primary axis (Refer to the figure titled *Starting a Velocity & Position Following Move—1:1 Ratio)* to compensate for the fact that the primary axis is already moving (velocity = **Vp**).


Position and Velocity Following
Primary Axis
velocity = Vp
steps
8000
16000
24000
32000
40000
48000
velocity = 0
Secondary Axis
The secondary axis begins moving when the primary axis is at location 0 on the scale.

Starting a Velocity & Position Following Move—1:1 Ratio

The key to position and velocity following is that the **Vp** value does not matter (previous figure). Assuming the application in the previous figure is programmed with following acceleration, accelerating over 8000 primary encoder steps, the spots will always be at a 1:1 speed ratio after the primary axis moves the 16000 steps (see the following figure).


Velocity and Position Following
Primary Axis
velocity = Vp
steps
8000
16000
24000
32000
40000
48000
velocity = Vp
Secondary Axis
The secondary has achieved a 1:1 ratio and they are synchronized positionally.

Velocity & Position Following After Acceleration Is Done

Following Acceleration

Accelerating over a known distance with respect to a known primary axis distance allows you to synchronize the exact phase relationship you want between the primary and secondary axes. Following acceleration enables this synchronization.

Command	Description
FAC	The increment of following ratio by which the following ratio changes during acceleration
FEN	The number of encoder pulses that cause the following ratio to increment to the next value
FSF1	Enables Following Acceleration mode

Determining FAC and FEN with Primary Axis Data

To determine what **FAC** *and* **FEN** *are, you must know*:

- ❑ The maximum velocity that the primary axis can travel (the velocity must be in units of primary encoder steps per second)
- ❑ The distance in primary encoder steps that the primary axis will move during which time the secondary axis will accelerate ***or*** the maximum acceleration that the secondary axis can accelerate

These parameters will be denoted as follows:

V_{Pmax} = Maximum primary velocity in encoder steps per second

D_{Pacc} = Distance primary axis travels while secondary axis accelerates in primary encoder steps

OR

As_{MAX} = Maximum acceleration of the secondary axis

Based on these equations, the values for **FAC** and **FEN** can be determined:

Equation 5-1. FEN

$$FEN = V_{PMAX} * \frac{TF}{1000} \qquad V_{Pmax} = \frac{\text{Primary Encoder Counts}}{\text{Second}}$$

TF = Primary Encoder Sample Period in ms

Equation 5-2. FAC

$$FAC = \frac{FOL * V_{Pmax}}{D_{Pacc}} * \frac{TF}{1000} \qquad V_{Pmax} = \frac{\text{Primary Encoder Counts}}{\text{Second}}$$

TF = Primary Encoder Sample Period in ms

FOL = Following percentage in units of percent

Dp_{acc} = Distance primary moves during secondary axis accel in units of primary axis enc counts

Typically, the application will have the secondary axis start from rest and accelerate up to an **FOL** value of 100 (1:1 ratio). However, the **FOL** value can be any value that is within the limits of the motor/drive system. ***D_{acc} is in units of primary encoder steps***. Determining the **FAC** and **FEN** values sets the number of primary axis encoder steps over which the secondary axis will accelerate (independent of the primary axis' speed). The secondary axis always travels the same number of motor steps during acceleration while the primary encoder moves Dp_{acc}.

The **TF** command allows you to set the sample period of the primary axis' encoder. It is programmable from 1 - 32 ms. ***The default is 4 ms***. **TF** simply scales the **FAC** and **FEN** values. In Equations 5.1 and 5.2, **TF** is used in units of ms—the constant of 1000 converts it to seconds so that the units cancel properly. Typically you will want **TF** to be as fast as your system will allow (1 ms). If the primary encoder is moving slowly, you may need to increase the sample rate to more than 1 ms because the actual encoder count does not change by much in a sample period and thus you have coarser resolution on the changes in encoder counts. For example, if your sample period is 1 ms and the maximum speed is 1 rps, the encoder count only changes by 4 counts each sample period. If there is a slight variation in speed and you read a change of 3 counts, there is a 25% variation. This may cause choppier secondary axis motion. Changing the encoder sample period can have a smoothing effect.

Determining FAC and FEN with Secondary Axis Data

Instead of knowing the distance that you want the secondary axis to accelerate over, you may know the maximum acceleration that your secondary axis can accelerate at. You can determine Dp_{acc} using As_{max} and the following speed percentage that you are accelerating to. Use Equation 5-3 to determine Dp_{acc} from As_{max}.

Equation 5-3.

Dpacc

$$DP_{acc} = \frac{V_{Pmax2}}{2AS_{max}} * FOR * \frac{FOL}{100} = \frac{V_{P2_{MAX}}}{2AS_{MAX}} * FOR * \frac{FOL}{100}$$

The acceleration is in units of secondary motor steps/sec^2. The maximum velocity of the primary axis is in primary encoder steps/sec.

Using this value for Dp$_{acc}$, you can use Equations 5-1 and 5-2 to determine **`FAC`** and **`FEN`**. Remember to enable the Following Synchronized Acceleration mode (**`FSF1`**) to enable following acceleration.

How Following Acceleration Works

The concept of accelerating the secondary axis over a known distance with respect to a known primary axis distance (independent of the primary axis' speed) is developed on the analogies drawn between *following* and *time-based motion*. In ***time-based motion***, the velocity describes the rate of change of the position with respect to a change in time. In ***following-based motion***, the secondary axis moves at a ratio of the primary axis' velocity. This following ratio is of the same units as the velocity it is following, but is simply scaled. Therefore, the following ratio is analogous to a velocity. In the same manner, an acceleration in the time domain is defined as the rate of velocity change. The analogy in following would be to have a following acceleration that is a rate of change of the following ratio. ***Time-based motion is based on sampled time whereas following is based on the sampled primary axis encoder pulses (for digital systems)***. Examine the following example.

❑ Primary axis encoder → 4000 counts/revolution

❑ Secondary axis → 4000 steps/revolution.

❑ Primary axis speed = 1 rps

❑ Secondary axis following speed percentage = 100%

❑ Distance that secondary axis accelerates = 2000 primary axis encoder steps

❑ Primary axis encoder sample period = 1 ms

Therefore, the secondary axis must now accelerate over 2000 primary encoder steps to a following percentage of 100% or a speed of 1 rps. Specifying the number of primary encoder pulses and the final speed that the secondary axis must attain after acceleration defines the ***acceleration ramp***. If the secondary axis' acceleration is based on time, you can calculate an acceleration ramp that would accelerate the secondary axis in the desired fashion.

$$\text{Acceleration time} = \frac{2000\ \text{steps}}{4000\ \frac{\text{steps}}{\text{sec}}} = 0.5\ \text{seconds}$$

$$\text{Change in velocity} = 4000\ \text{steps/sec} — 0\ \text{steps/sec} = 4000\ \text{steps/sec}$$

$$\text{Acceleration} = \frac{4000\ \frac{\text{steps}}{\text{sec}}}{0.5\ \text{sec}} = 8000\,\frac{\text{steps}}{\text{sec}^2} = \frac{8\ \frac{\text{steps}}{\text{sec}}}{\text{ms}}$$

If you changed the velocity by 8 steps/sec every sample period (**`TF`** - 1 ms), you would achieve the desired acceleration ramp. The problem is that application is dependent on time. If the primary axis' speed changes, the secondary axis would no longer be accelerating over 2000 primary encoder steps. Therefore, the application requires a following acceleration that is based on encoder pulses rather than time.

$$\text{Change in Following Percentage} = 100\% - 0\% = 100\%$$

$$\text{Following Acceleration time} = \frac{2000 \text{ steps}}{4000 \frac{\text{steps}}{\text{sec}}} = 500 \text{ ms} = 500 \text{ sample periods}$$

$$\text{Following Acceleration} = \frac{100}{500} = 0.2\,(\textbf{FAC})$$

The application is still time dependent. To remove the time dependency and make the acceleration dependent on the encoder pulses ***replace the time sample period by an encoder period***. In the above example, following percentage was based on changing the following percentage by 0.2 every sample period. It will take 500 sample periods to achieve a 100% following percentage. At 4000 steps/sec, the primary encoder is changing at a rate of 4 steps per ms or 4 steps/sample period.

1 sample period = 4 steps (encoder period—**FEN**)

Instead of changing the following percentage every sample period, change it every time the encoder count changes by 4 steps. If the primary encoder is moving at the maximum velocity, the acceleration ramp will be equal to the maximum acceleration provided. However, if the primary encoder velocity is less than the maximum velocity, the acceleration will also be reduced. The distance that the secondary axis accelerates and travels (with respect to the primary axis' moves during this acceleration) will remain unchanged. If the primary axis exceeds the maximum velocity, the acceleration ramp would also increase and exceed the maximum acceleration, which may cause a motor stall.

In summary, following acceleration uses the analogy that a change in velocity per change in time (normal acceleration) is the same concept as a following acceleration being a change in following percentage per change in encoder steps. In the SXF, the following percentage changes by the **FAC** value for each change in encoder steps of **FEN** steps. Several different combinations of **FAC** and **FEN** can achieve the same acceleration ramp. However, only unique **FAC** and **FEN** values will satisfy a specific maximum velocity and maximum acceleration. **FAC** and **FEN** can easily be determined with the equations and examples discussed earlier.

Decelerating

The SXF decelerates to zero speed using the **AD** value. Although you may expect that this will diminish the positional relationship, this is usually not a concern at the endpoint in a profile. The following move is usually started by a trigger, which indicates that the primary axis is at a particular location. The move could also be started based on the primary axis' encoder position. The move is typically of a preset distance. ***The important point is that the secondary axis is at a known position with respect to the primary axis when the profile begins***. This positional relationship is maintained during acceleration. After the secondary axis moves the appropriate distance, it will normally return the same distance it just traveled at a high speed to prepare for a repeat move. This is why the deceleration is not important. However, in cases where the SXF must perform electronic cam profiles, a deceleration's positional relationship may be necessary. In this case, the SXF can decelerate to a stop using the **FAC** and **FEN** values by setting the following ratio to 0 (**FOLØ**) while in Position Profile mode. To terminate this move, a stop command must be issued after the secondary axis reaches zero speed.

Position and Velocity Following Example

In this example, you will perform a preset move using following acceleration (with the parameters and motion requirements listed below).

❏ Primary axis encoder resolution = 4000 counts/rev

❏ Secondary axis motor resolution = 25000 steps/rev

❏ Maximum primary encoder speed = 2 rps

❏ Distance in primary encoder steps that the secondary axis must accelerate over = 2000 steps

❏ Desired speed ratio—**FOL** = 100% (1:1 ratio)

❏ Preset secondary axis move distance = 3 motor revolutions

❏ Encoder sample period—**TF** = 4 ms

A preset move of 75000 steps will be made. The secondary axis will accelerate over 2000 primary encoder steps up to the same speed as the primary encoder. A trigger will initiate motion. Follow these steps to perform the move profile. The encoder sampling period is set to the default of 4 ms.

Step ①

Determine the values for **FAC** and **FEN** from Equations 5-1 and 5-2.

$$\mathbf{FEN} = Vp_{max} * \frac{\mathbf{TF}}{1000} = 2\ \frac{revs}{sec} * 4000\ \frac{counts}{rev} * \frac{4}{1000}\ seconds = 32\ encoder\ counts$$

$$\mathbf{FAC} = \frac{\mathbf{FOL}\ *\ Vp_{max}}{Dp_{acc}} * \frac{\mathbf{TF}}{1000} = \frac{100\ *\ 8000\ \frac{counts}{sec}}{2000\ \ counts} * \frac{4}{1000}\ seconds = 1.6\ percent$$

Step ②

Enter the values for **FEN** and **FAC**.

Command	Description
`> FEN32`	Number of encoder counts of change required to increment the following percentage by **FAC**
`> FAC1.6`	The amount the following percentage increments for each **FEN** change in encoder counts

If you want to change the sample period of the primary encoder to 1 ms, scale both **FEN** and **FAC** by the change in the **TF** value. For example, if you go from 4 ms to 1 ms, divide **FAC** and **FEN** by 4 to get the following values.

FEN = 8

FAC = 0.4

If we went from **TF4** to **TF8**, multiply **FAC** and **FEN** by 2. ***Remember to change* FAC *and* FEN *if you change* TF**.

Step ③

Enable Following Acceleration mode.

Command	Description
`> FSF1`	Enables following acceleration

Step ④

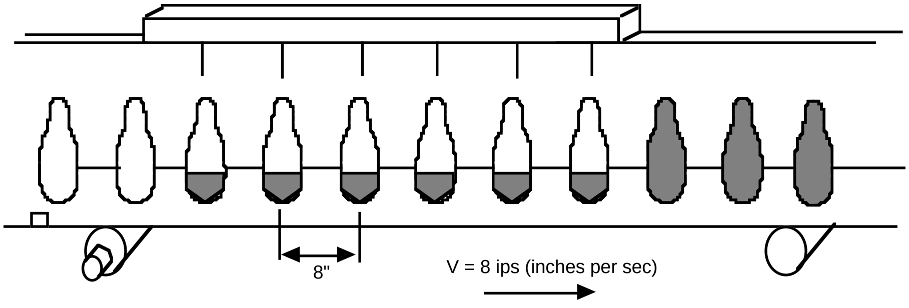
Start the primary axis into motion, then enter the commands below to perform the following acceleration move. If the primary axis' speed exceeds 2 rps, the following acceleration will not work properly.

Command	Description
`> MN`	Sets SXF to Normal mode
`> FSI1`	Enables following
`> D75ØØØ`	Sets the preset move distance to 75000 steps
`> FOL1ØØ`	Sets the following percentage to 100%
`> G`	Starts the following move

The secondary axis will accelerate over 2000 primary encoder steps. The secondary axis will move 1000 * **FOR** or 6250 motor steps over this acceleration ramp.

Following Acceleration Example: Bottle Filling

Typically, an application that requires position and velocity following will start the secondary axis from rest and accelerate it to 100% of the primary axis' speed (a 1:1 speed ratio). A trigger initiates motion on the secondary axis when a part or product is in a particular location on the primary axis. In this example, a conveyer belt moves bottles on a production line. The secondary axis is a filler that accelerates up to the conveyer line speed and fills the bottles. It fills six bottles at a time and then returns to the start point to fill six more.



Position & Velocity Following—Bottle Filling Application

One cycle of operation consists of the following steps.

① The secondary axis accelerates to the conveyer line speed.

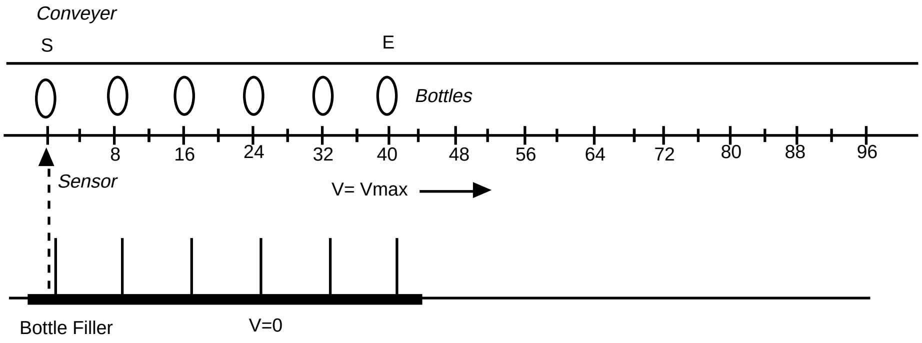
② The secondary axis enables an output that tells the dispenser to begin filling the bottles.

③ The secondary axis decelerates to a stop and returns to the starting point (at a high speed) to begin filling the next set of bottles.

In this application, the rate at which the bottles can be filled determines the maximum rate of the entire dispensing cycle. *Note the following information about the application.*

Maximum conveyer speed: 8 ips
Time to bill bottles: 2.5 seconds
Primary encoder resolution: 4000 counts/rev
Primary encoder linear resolution: 2 revs/inch = 8000 counts/inch
Distance between bottles: 8 inches
Bottle filler motor resolution: 25000 steps/revolution
Bottle filler linear resolution: 1 revolution/inch = 25000 steps/inch
Distance over which bottle filler accelerates: 2 inches

An output must be activated at the point that the bottle filler is moving at the same speed as the conveyer to initiate the dispensing of fluid into the bottles. A photoelectric sensor detects a bottle and begins the filling cycle. The SXF waits for the sensor as a trigger. When a bottle is detected, the SXF accelerates to the line speed, turns on an output, fills the bottle stops, and returns to the starting point to wait for the next trigger. The following figure depicts the conveyer and the filler axis at the start of a filling cycle.



Start of A Filling Cycle

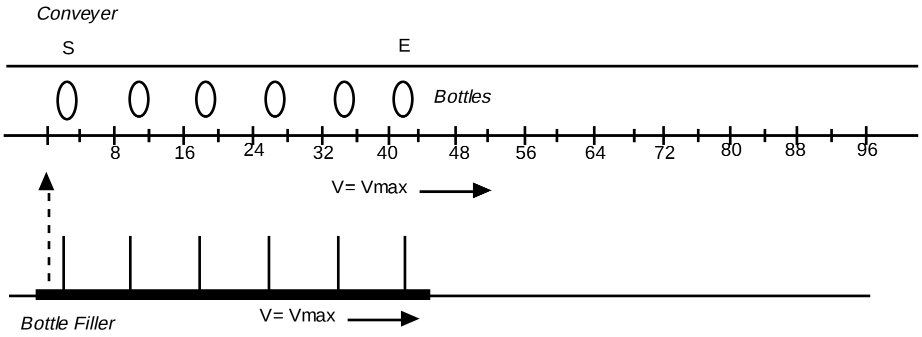
When the bottle marked **S** crosses the trigger sensor, the cycle begins.

① The secondary axis accelerates to the conveyer line speed.

② The secondary axis enables an output that tells the dispenser to begin filling the bottles.

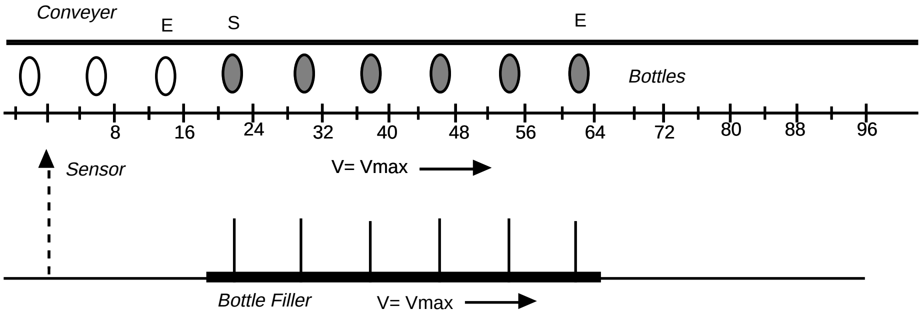
③ The secondary axis decelerates to a stop and returns to the starting point (at a high speed) to begin filling the next set of bottles.

This figure shows the two axis after the bottle filler reaches line speed and is ready to begin dispensing.



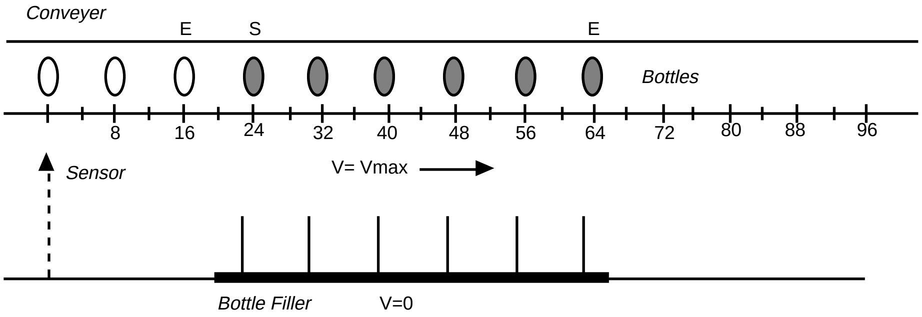
Bottle Filler Has Accelerated to Line Speed

A bottle can be filled in 2.5 seconds. At a maximum conveyer speed of 8 ips, the conveyer will move the bottles 20 inches. The next figure shows the bottle's location after the first six bottles have been filled. ***Note the location of the next set of bottles that will be filled***.



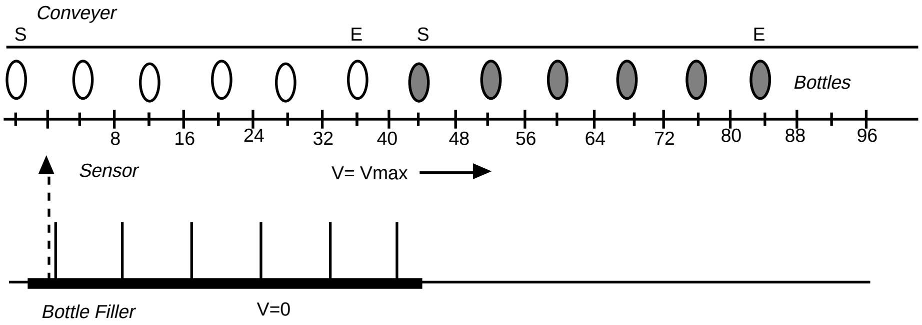
Dispensing is Completed

The bottle filler must now stop and return to the start location before the bottle marked **S** crosses the trigger point. The following figure shows the location of the bottle filler after it has stopped. It must now return 22 inches to the start before the bottles have moved 20 inches. The bottle filler will be in place, ready for the next trigger from the next set of bottles. It will arrive in place 4 inches before the next bottle. At 8 ips, 500 ms will elapse before the next bottle.



Bottle Filler Stops

To get back to the starting point before the conveyer has moved 20 inches, the bottle filler must return at a speed faster than the conveyer. It will accelerate at the same following acceleration set for the first part of the cycle. The following figure shows the bottle filler after it returns to the starting point. After the conveyer travels 4 more inches, the cycle will resume.



Bottle Filler Ready to Start a New Cycle

To program the application, use the following steps.

Step ①

Determine the **FOR** value.

Primary conveyer axis steps per inch = 8000

Secondary bottle filler axis steps per inch = 25000

$$\mathbf{FOR} = \frac{25000}{8000} = 3.125$$

Step ②

Determine the **FAC** and **FEN** following acceleration parameters. The encoder sample period time (**TF**) is 1 ms.

$$\mathbf{FEN} = 8000\,\frac{\text{steps}}{\text{inch}} * 8\,\frac{\text{inches}}{\text{second}} * \frac{1}{1000} = 64\,\text{steps}$$

$$\mathbf{FAC} = \frac{100\% \; * \; 8000\,\dfrac{\text{steps}}{\text{inch}} \; * \; 8\,\dfrac{\text{inches}}{\text{second}}}{16000\;\;\text{steps}} * \frac{1}{1000} = 0.4$$

Step ③

Determine how far the primary and secondary axes will move during the dispensing part of the cycle. This includes the acceleration and deceleration parts of the move for the bottle filler when no fluid is dispensed.

$$Vp_{max} = 64000\,\frac{\text{steps}}{\text{sec}}$$

The dispensing takes 2.5 seconds. During the time that a constant following percentage occurs, the conveyer will have moved:

$$Dp_{con} = 64000\,\frac{\text{steps}}{\text{sec}} \;\; * \; 2.5\text{ sec} = 160000$$

The secondary axis will move this distance * **FOR**:

$$Ds_{con} = 160000\text{ primary encoder steps} * 3.125\,\frac{\text{secondary motor steps}}{\text{primary encoder steps}} = 500000\text{ secondary motor steps}$$

The secondary axis or bottle filler axis will accelerate over 2 inches of the conveyer at maximum speed and decelerate over two inches. If the conveyer moves slower, the dispensing part of the process will become a smaller percentage of the total cycle. The distance the secondary axis travels during acceleration will be one half of the distance the conveyer or primary axis travels. The secondary axis starts from rest and accelerates to match the primary axis' speed. The secondary axis will move 1 inch during acceleration and 1 inch during deceleration. The total distance of the bottle filler move is:

$$D_{sec} = 500000 + 2 * (1\,\text{inch} * 25000\,\frac{\text{steps}}{\text{inch}}) = 550000\,\text{steps (secondary motor steps)}$$

The value to be entered for the **AD** command for deceleration is:

$$\mathbf{AD} = \frac{V^2}{2{*}D} = \frac{\left(8\,\frac{\text{inches}}{\text{sec}}\right)^2}{2{*}1\,\text{inch}} = 32\,\frac{\text{inches}}{\text{sec}^2} * 1\,\frac{\text{revolution}}{\text{inch}} = 32\,\frac{\text{revs}}{\text{sec}^2}$$

If the primary axis' speed changes, the SXF will still decelerate at this rate. The distance that the bottle filler moves will be identical, regardless of the primary axis' speed, so the bottle filler will still have to make a 22-inch move ***back*** while the primary axis moves 20 inches.

Step ④

You have determined the parameters necessary for the first part of the move. Next, you must determine the following percentage required to move the secondary axis back 22 inches while the primary axis moves 20 inches. Use Equation 5-4 to determine **FOL**.

Equation 5-4. FOL

$$\mathbf{FOL} = \frac{D_{prim}}{200{*}K} - \sqrt{\left(\frac{D_{prim}}{200{*}K}\right)^2 - \frac{D_{sec}}{\mathbf{FOR}{*}K}} \quad \text{Where } K = \frac{\mathbf{FEN}}{100{*}\mathbf{FAC}}$$

$$\mathbf{FOL} = \frac{160000}{200{*}1.6} - \sqrt{\left(\frac{160000}{200{*}1.6}\right)^2 - \frac{550000}{3.125{*}1.6}} = 500 - 374.2 = 125.8\%$$

125.8% is the return following percentage.

Step ⑤

Enter the sequence below to implement the motion.

Command	Description
> XE1	Erases sequence #1
> XD1	Defines sequence #1
FOR3.125	Sets motor to encoder steps per unit travel ratio
FAC.4	Sets the following acceleration increment to .4 % per encoder period
FEN64	Sets the encoder period, which increases the percentage to 64 steps
D55ØØØØ	Sets the secondary axis move for the cycle to 550000 motor steps
FSI1	Enables following mode
AD32	Sets the deceleration to 32 rps^2
IN1A	Defines input 1 as a trigger input
IN2D	Defines input 2 as a stop input
L	Starts a continuous loop
FOL1ØØ	Sets the initial following percentage to 100%
TR1	Waits on the input trigger
G	Starts motion
FOL125.8	Sets return move following percentage to 125.8%
H	Changes the direction
G	Starts the return following move
H	Changes direction again
N	Ends the loop—the following cycle will repeat
>XT	Ends the sequence

To decelerate using the **FAC** and **FEN** values, modify the program as follows:

Command	Description
> XE1	Erases sequence #1
> XD1	Defines sequence #1
SSH1	Sets save buffer on stop
FOR3.125	Sets motor to encoder steps per unit travel ratio
FAC.4	Sets the following acceleration increment to 0.4 % per encoder period
FEN64	Sets the encoder period, which increases the percentage to 64 steps
TF1	Sets the following encoder sample period to 1 ms
D55ØØØØ	Sets the secondary axis move for the cycle to 550000 motor steps
FSI1	Enables Following mode
AD32	Sets the deceleration to 32 rps^2
IN1A	Defines input 1 as a trigger input
IN2D	Defines input 2 as a stop input
L	Starts a continuous loop
FOL1ØØ	Sets the initial following percentage to 100%
TR1	Waits on the input trigger
MPP	Enters the Motion Profiling mode
G	Starts motion
FP176ØØØ	Waits until 176000 encoder pulses have passed
FOLØ	Stops the motion of the secondary
FP16ØØØ	Waits for the decel ramp distance
STOP	Stops the move itself
FOL125.8	Sets return move following percentage to 125.8%
H	Changes the direction
G	Starts motion
FP139782	Waits until 139782 encoder pulses have passed
FOLØ	Stops the motion of the secondary
FP2Ø218	Waits for the decel ramp distance
STOP	Stops the move itself
H	Changes direction again
N	Ends the loop—the following cycle will repeat
> XT	Ends the sequence

The value for **FP** during the return move is determined by calculating the distance the primary axis will move during acceleration and the constant following percentage portion and determining the distance it moves during the deceleration portion. This is determined from:

$$Dp_{acc} = \mathbf{FOL} * \frac{\mathbf{FEN}}{\mathbf{FAC}} = 125.8\% * \frac{64}{0.4} = 20218 \text{ encoder steps}$$

$$Dp_{dec} = Dp_{acc} = 20218 \text{ inches}$$

$$Dp_{con} = D_{prim} - Dp_{dec} - Dp_{acc} = 119564$$

$$D_{prim} = 20 \text{ inches} = 160000$$

The *first* **FP** = Dp_{con} + Dp_{acc} = 119564 + 20218 = 139782. The *second* **FP** = Dp_{dec} = 20218. The second **FP** (20218 steps) measures the deceleration ramp. After deceleration, the move stops. The acceleration and deceleration ramps are based on the primary axis' speed.

Recede and Advance While Following

Receding and advancing while following requires position and velocity following. In this type of application, the secondary axis follows the primary encoder at a 1:1 ratio or at the same speed. The secondary axis has a specific positional or phase relationship with the primary encoder. This type of application is used when multiple operations (such as welds) must be performed on one moving part. The operations are performed at various places on the part, requiring the secondary axis to advance or recede.

In an ***advance application***, the secondary axis must accelerate and move a specific distance beyond the primary axis, then decelerate to a 1:1 ratio. The secondary axis moves a specific distance with respect to the primary axis while both axes are moving. In a ***recede application***, the secondary axis decelerates until it recedes a specific distance behind the primary axis and then resumes a 1:1 speed ratio with the primary axis.

The point at which the advance move or recede move occurs is based on a specific position on the primary axis or on an input trigger. This type of application requires that the following ratio be changed on-the-fly while based on either an input or the primary encoder's position. It also requires that the secondary axis be able to move a specific distance while the primary axis moves a corresponding specific distance. In this manner, the secondary can advance or recede a specific distance with respect to the primary axis. *To change the following ratio on-the-fly*, you must use Motion Profiling (**MPP**) mode. You will need to measure the distance traveled by the primary encoder. Use the set of commands below.

Command	Description
> FPn	Delays processing for n primary encoder steps
> FPAn	Delays processing until the absolute count of the primary encoder reaches n
> VAR1=FEP	Allows you to read the value of the **F**ollowing **E**ncoder **P**osition into variable 1.

Advance Following Example

In this example, the primary axis has a 4000 count per revolution encoder. The secondary motor has a 4000 step per revolution motor. Therefore, the value for **FOR** is 1. The application requires that the secondary axis:

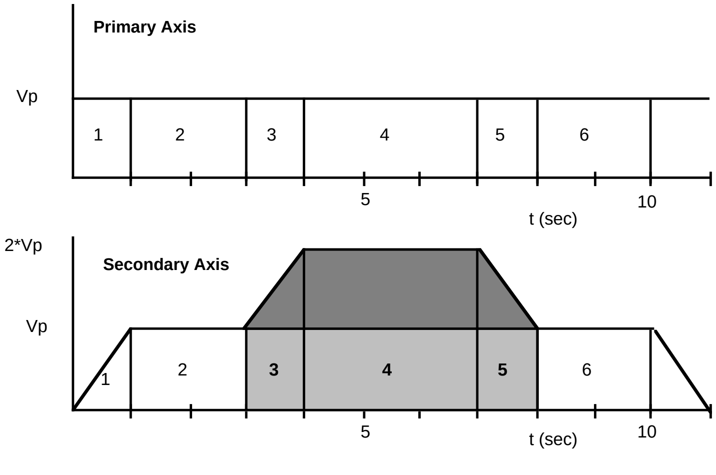
① Accelerate over 4000 primary encoder steps

② Move at a 1:1 ratio for 2 primary encoder revolutions

③ Advance (with respect to primary axis) 12000 primary encoder steps

④ Move 2 more primary encoder revolutions at 1:1 after advancing

⑤ Stop. The move profiles are shown below.



Advancing with Respect to the Primary While Following

The darkly shaded area in the previous figure represents the distance that the secondary axis advances with respect to the primary axis. The lightly shaded area represents the distance that both the primary and the secondary axes move during the ***advance portion*** of the profile. In these types of applications, the phase relationship or positional relationship is set with the **FAC** and **FEN** values (for as long as the following acceleration is performed). When the secondary axis accelerates from the 1:1 ratio to the 2:1 ratio, it will again accelerate at the following acceleration set. When it decelerates back to the 1:1 ratio, it decelerates in the same manner as it accelerates (using the **FAC** and **FEN** following acceleration rate). When the secondary axis decelerates back to *rest or zero speed*, it will decelerate at the **AD** rate. If the secondary axis must decelerate to zero speed at the FAC and FEN values, you must use **FOLØ** and set the following ratio to zero to make it stop.

Before programming this move profile, we will completely analyze the motion of the secondary and primary axes and then describe the sequence of commands necessary for performing the move.

In the previous figure, assume that the primary axis is moving at a maximum speed of 4000 steps/second. The area of each 1-second block represents 4000 primary encoder counts. Starting with Section #1, the primary encoder begins the section at a speed of $Vp_{max} = V_p$ = 4000 steps/sec. Therefore, the primary encoder moves 4000 counts during this section. From Section #1 of the secondary profile's plot, you can see that the secondary motor starts from rest and accelerates to a 1:1 speed ratio over 4000 counts of the primary encoder, Dp_{acc} = 4000 steps. You can calculate the desired **FAC** and **FEN** values to create such an acceleration ramp. This following acceleration will be used throughout the profile when changing from one following ratio to another. Set the encoder sample rate, **TF**, to 1 ms.

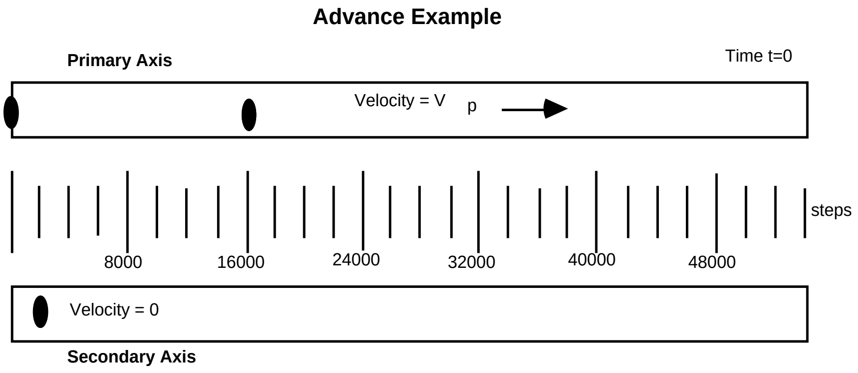
$$Vp_{max} = 4000 \text{ steps/sec}$$

$$Dp_{acc} = 4000 \text{ steps}$$

$$\mathbf{FEN} = Vp_{max} * \frac{\mathbf{TF}}{1000} = 4000 \frac{\text{steps}}{\text{sec}} * \frac{1}{1000} \text{ sec} = 4 \text{ encoder counts}$$

$$\mathbf{FAC} = \frac{\text{FOL} * Vp_{max}}{Dp_{acc}} * \frac{\mathbf{TF}}{1000} = \frac{100\,\% * 4000 \frac{\text{steps}}{\text{sec}}}{4000 \text{ steps}} * \frac{1}{1000} \text{ sec} = 0.1 \text{ percent}$$

By calculating the area under the secondary axis profile curve, you can determine that the secondary motor has moved 2000 motor steps. If the secondary axis is accelerating to the same speed as the primary axis, it will always travel half of the primary axis' distance. In this manner, the secondary axis will be physically aligned with this point when it reaches a 1:1 speed ratio. The next figure shows two conveyer belts—primary and secondary axes. The primary and secondary axes' spots are at different locations at different times in the profile.



Advance Example at the Start of Section # 1

Usually, you will use an equation, not a graphic, to determine the distance traveled. Equation 5-5 calculates the distance the secondary axis travels for any acceleration, even if the secondary axis starts acceleration while it is already moving at a given ratio to the primary axis.

Equation 5-5 Ds_{acc}

$$Ds_{acc} = \mathbf{FOR} * \left(\frac{1}{2} * \Delta\mathbf{FOL}^2 * \frac{\mathbf{FEN}}{100*\text{FAC}} + \Delta\mathbf{FOL}*\mathbf{FOL}_I * \frac{\mathbf{FEN}}{100*\mathbf{FAC}} \right)$$

FOL = The change in following percentage

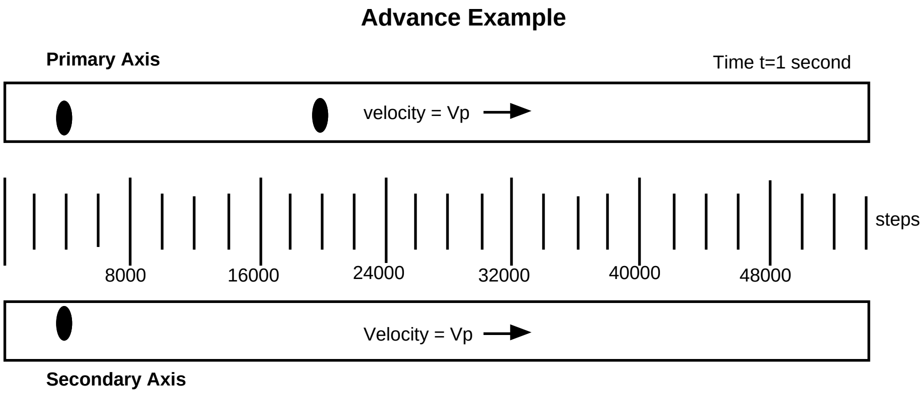
$\mathbf{FOL}_I$ = The initial following percentage

In Equation 5-5, the change in following percentage is the difference between the final following percentage that you are accelerating to and the following percentage you are starting from. If you are starting from rest, the initial following percentage is Ø and the change in following percentage is **FOLØ** or **FOL**. If you are at **FOL1ØØ** and want to accelerate to a following percentage of **FOL2ØØ**, the change in following percentage is 100 and the initial following percentage is 100. **FAC** and **FEN** are the values calculated from the equations for **FAC** and **FEN** using the V_{max} and primary encoder acceleration distance.

The equation has two parts. The first part has the square of the following percentage change and the second part has a single following percentage change term. The first term determines the distance that the secondary axis travels due to the acceleration ramp portion of the curve. The second term gives the distance that the secondary axis travels due to its initial velocity. In this case, the secondary axis starts from rest (the second term contributes zero to the distance traveled). From the plot of the profile (Section # 1), the initial following percentage is Ø and the final percentage is 100, or the same speed as the primary axis. Using the general equation above, you should get 2000 steps, which is the area under the curve.

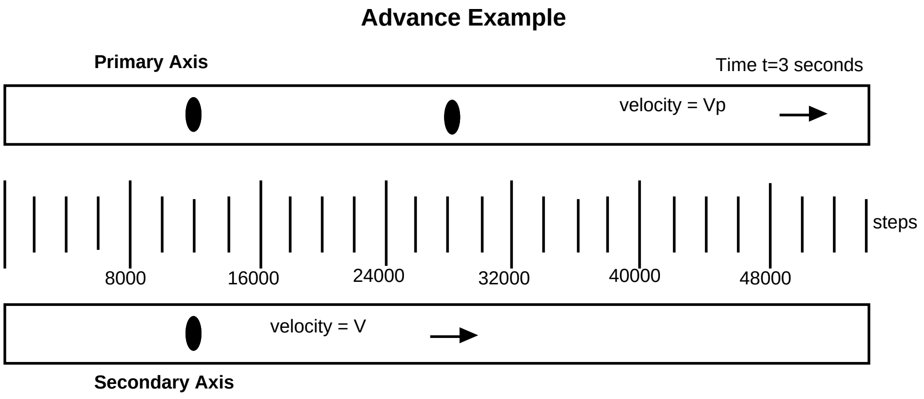
$$Ds_{acc} = 1* \left(\frac{100^2}{2} * \frac{4}{100*0.1} + 100 * 0 * \frac{4}{100*0.1} \right) = 2000 \text{ secondary motor steps}$$

Therefore, in Section #1, the primary axis moved 4000 encoder counts and the secondary axis moved 2000 motor steps. We have determined the values for **FAC** and **FEN** based on the plots of the primary and secondary axes' profiles and the fact that V_{max} is 4000 steps/sec. At the end of Section #1, the spots will be in the locations shown in the following figure (t = 1 second).



Advance Example—End of Section #1

In Section #2, the primary axis is moving at V_{max} and the secondary axis is moving at the same speed because the ratio is 1:1 (**FOL1ØØ**). Section #2 lasts for 2 seconds. The primary axis travels 8000 steps during this section. The secondary axis travels 8000 steps too. This can be determined from the profile plot by calculating the area beneath the curve for the section. The primary and secondary axes are lined up at the start of Section #2 and they travel at a 1:1 ratio for the duration of the section. At the end of Section #2, the primary axis has moved 12000 steps and the secondary axis has moved 10000 steps (thru Sections #1 and #2). The next figure shows the relative location of the spots at the end of Section #2 (t = 3 seconds).

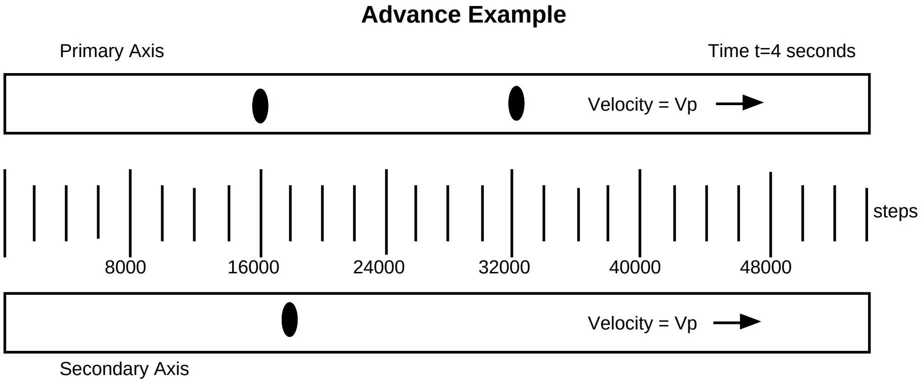


Advance Example—End of Section #2

In Section #3, the advance portion of the secondary profile begins. The secondary axis accelerates from a following percentage of 100% to 200% (a 2:1 ratio). Look at the plot of the profiles to graphically determine the distance that the primary and secondary axes have traveled. The primary axis moves 4000 steps and the secondary axis moves 6000 steps. The secondary axis' distance can also be determined from the previous equation..

$$Ds_{acc} = 1* \left(\frac{1}{2} * 100^2 * \frac{4}{100*0.1} + 100 * 100 * \frac{4}{100*0.1}\right) = 6000 \text{ steps}$$

At the end of Section #3, the primary axis has moved 4000 steps since the beginning of the advance portion of the profile. The secondary axis has advanced 2000 steps with respect to the primary axis. The following figure shows the location of the spots at the end of Section #3.



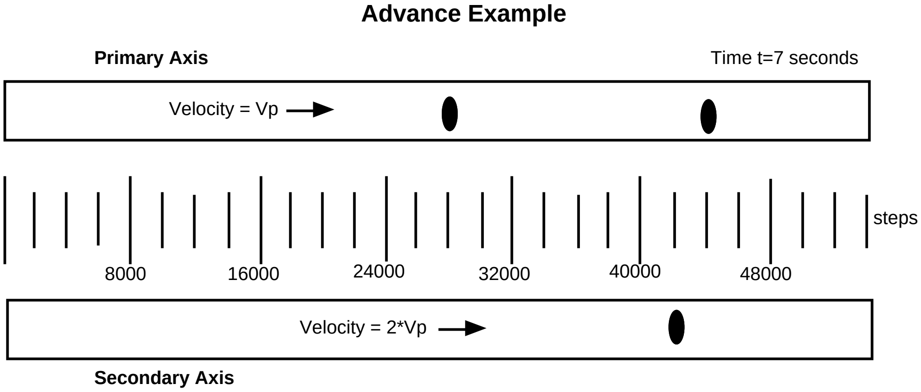
Advance Example—End of Section #3

At the start of Section #4, the secondary axis is at a following percentage of 200% and is moving at 2 * Vp (a 2:1 ratio). Section #4 lasts 3 seconds. The primary axis moves 12000 steps, while the secondary axis moves 24000 steps. The distance the secondary axis traveled can be determined from the equation below.

Equation 5-6. Ds_{con}

$$Ds_{con} = \frac{\mathbf{FOL}}{100} * D_{prim} = 1 * \frac{200}{100} * 12000 = 24000$$

Since the start of the advance portion, the primary axis has moved a total of 16000 steps and the secondary axis has moved 30000 steps. The secondary axis has advanced 14000 steps with respect to the primary axis. The next figure shows the location of the spots at the end of Section #4.



Advance Example—End of Section #4

During Section #5, the secondary axis decelerates to a following percentage of 100% (a 1:1 ratio). After it decelerates, it will have completed the advance portion of the profile. The primary axis travels 4000 steps in Section #5. The secondary axis travels 6000 steps. When the secondary axis accelerates from one following percentage to another, then decelerates to the original following percentage (as in this example), ***the secondary axis' acceleration distance will always equal the deceleration distance***. However, the deceleration distance can also be calculated from the following equation.

Equation 5-7. Ds_{dec}

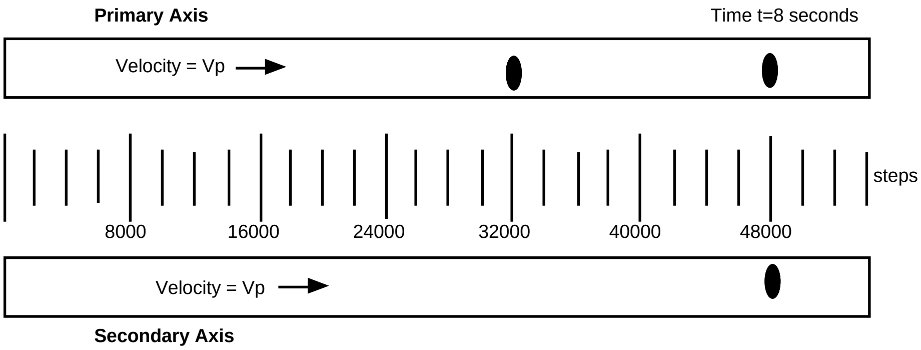
$$Ds_{dec} = \mathbf{FOR} * \left(\frac{1}{2} * \Delta\mathbf{FOL}^2 * \frac{\mathbf{FEN}}{100 * \mathbf{FAC}} + \mathbf{FOL} * \mathbf{FOL}_I * \frac{\mathbf{FEN}}{100 * \mathbf{FAC}}\right)$$

When decelerating, the initial following percentage is the percentage you are at when you begin deceleration. In this example, it is 200%. The final following percentage is 100%. Therefore, the change in following percentage is a negative number (-100%).

$$Ds_{dec} = 1 * \left(\frac{1}{2} * (-100)^2 * \frac{4}{100*0.1} + -100 * 200 * \frac{4}{100*0.1}\right) = -2000 + 8000 = 6000 \text{ steps}$$

At the end of Section #5, the advance portion of the move profile is complete. The secondary axis is moving at a 1:1 speed ratio with the primary axis. The secondary axis has moved 36000 steps and the primary axis has moved 20000 steps during the advance portion. The secondary axis has advanced 16000 steps with respect to the primary axis. The following figure shows the locations of the axes at the end of Section #5.

Advance Example



Advance Example—End of Section #5

During Section #5, the secondary axis travels at a 1:1 ratio until the deceleration ramp begins. If the move is a preset move, it will begin the deceleration ramp at the appropriate time based on the **AD** command. It will be at rest at the exact distance of the preset move. If the move is a continuous move, it will decelerate according to the **AD** value when a Stop (**S**) or Kill (**K**) command is encountered. From within a sequence, you can use the buffered Stop command. If you want the deceleration ramp to use the following acceleration value to decelerate, use the **FOLØ** command. After the secondary axis rests, a Stop command must be used to terminate the move. As depicted in the move profiles, the secondary axis moves at a 1:1 ratio for 8000 more steps, then decelerates to zero. During this time, the primary axis moves 8000 steps too.

Programming this profile requires the Following Encoder Distance Point (**FP**) or Following Encoder Absolute Point (**FPA**) command. **FP** is a delay-based on an incremental encoder distance. **FPA** is a delay-based on an absolute encoder distance.

FP Delay Example

To perform the advance move profile, breakpoints are needed to indicate when the secondary axis should accelerate to new following percentages. The **FP** and **FPA** commands define these breakpoints. These changes are performed on-the-fly and require the Motion Profiling mode. The breakpoints are the points at which acceleration or deceleration begin. This example will show the program using the **FP** command.

Step ① Set up the SXF with the appropriate encoder interface and enable the Following mode.

Command	Description
> FSI1	Enables SXF's Following mode

Step ② Set up the velocity following portion of the application. The number of secondary motor steps per unit of travel is 4000. The number of primary encoder steps per unit of travel is also 4000. Therefore, the **FOR** value is 1. The **FOL** command will be set up in the sequence for running the profile.

Command	Description
> FOR1	Relates the number of secondary motor steps for a distance to the number of primary encoder pulses for the same distance

Step ③ Set-up the following acceleration value and enable Following Acceleration mode. The values chosen are the same as were calculated in the explanation of the example above.

Command	Description
> FAC.1	Increases the following percentage to 0.1 for every change in encoder pulses by FEN
> FEN4	Sets the number of encoder pulses required before the following percentage is incremented by FAC.
> FSF1	Enables the use of following acceleration

Step ④ The breakpoints in the desired profile occur at the end of Section #2, the end of Section #4, and if **FAC** and **FEN** are used for deceleration, a breakpoint is set at the end of Section #6. If your **FOR** command differs from this example, the distance command is entered in terms of your secondary motor. The following sequence performs the desired profile. Each step of the sequence is explained. Enter the sequence.

Command	Description
> XE1	Erases Sequence #1
> XD1	Begins the definition of Sequence #1
D56ØØØ	The total distance the secondary moves is 56000 steps.
FOL1ØØ	The first following percentage to accelerate to is 100%.
MPP	Enters Motion Profiling mode so changes can be made on the fly.
G	The secondary motion begins.
FP12ØØØ	The first breakpoint occurs after the **primary axis** moves 12000 steps. **FP** causes command processing in the sequence to delay until 12000 primary encoder steps have been counted.
FOL2ØØ	After 12000 primary encoder steps the following percentage is changed to 200%. The **secondary axis** begins to accelerate at the following acceleration. Distance is known from previous analysis.
FP16ØØØ	Right after the command to begin acceleration to an **FOL** of 200%, the command processing is delayed until the **primary axis** has moved 16000 more pulses (from when **FP** is encountered and thus is an incremental encoder distance).
FOL1ØØ	After 16000 more primary encoder pulses, the following percentage changes to 100% and the **secondary axis** decelerates to a 1:1 ratio.
FP12ØØØ	Command processing is delayed 12000 more steps.
FOLØ	The secondary axis decelerates to an FOL of 0.
FP4ØØØ	It takes 4000 primary encoder steps to decelerate.
STOP	A Stop command is needed because the SXF would still think it was in a move and that the current following ratio was set to zero.
NG	Exits the Motion Profiling mode.
XT	Ends the definition of Sequence #1

Step ⑤ Typically, in an application that requires velocity and position following, it does not matter how you decelerate to a stop (you are moving from a synchronized state to rest). At the end of such a move, you will need to reverse direction and return to the starting location to repeat the profile. In this case, the optimal profile is as follows:

① Accelerate to a known positional relationship

② Perform the operation required at the synchronized speed

③ When it is complete, decelerate as fast as possible to repeat the cycle

You must decelerate at a rate unrelated to the primary encoder speed. The SXF allows you to do this. This will also simplify the programming. The same profile is programmed below using the **AD** deceleration value.

Command	Description
> XE1	Erases sequence #1
> XD1	Defines sequence #1
D56000	The total distance the secondary moves is 56000 steps
FOL100	The first following percentage to accelerate to is 100%
MPP	Enters Motion Profiling mode so changes can be made on the fly
G	The secondary axis' motion begins
FP12000	Delays processing until primary axis moves 12000 encoder pulses
FOL200	Change to 200% following percentage
FPA16000	Delays processing until primary axis moves 16000 encoder pulses
FOL100	Change ratio back to 100%
NG	Exits the Motion Profiling mode
XT	Ends sequence #1 definition

In this sequence, only two breakpoints are needed, the breakpoint to accelerate to 200% and then to decelerate back to 100%. Since the SXF will decelerate at the value in **AD**, it will automatically decelerate to a distance of exactly 56000 secondary motor steps at the appropriate time.

Step ⑥

This step uses **FPA** instead of **FP**. **FPA** delays processing until the absolute value of the following encoder counter exceeds the **FPA** value.

Command	Description
> XE1	Erases Sequence #1
> XD1	Defines Sequence #1
D56000	The total distance the secondary moves is 56000 steps.
PFZ	Zero the following encoder counter.
FOL100	The first following percentage to accelerate to is 100%.
MPP	Enters Motion Profiling mode so changes can be made on the fly.
G	The secondary motion begins.
FPA12000	Delay command processing until primary encoder count exceeds 12000 encoder pulses.
FOL200	Change to 200% following percentage.
FPA28000	Delay command processing until primary encoder counter exceeds 28000 encoder pulses or an incremental change of 16000 pulses.
FOL100	Change ratio back to 100%.
NG	Exits the profiling mode.
XT	Ends Sequence #1 definition

In Steps 5 and 6, the encoder is counting in the positive direction. If the encoder is counting in the negative direction, a negative sign is required for both the **FP** and **FPA** commands. Use either the **FP** or the **FPA** command, depending on your application.

*Use **FPA** when repetitive cycles of the same move profile are done without a trigger to start each cycle*. By making the delays dependent on an absolute encoder position, there is no accumulative error. In many cases, a trigger input from a sensor is used to start the move profile that is repeated.

*If you use a trigger to start the move each time, use **FP** and the trigger will remove any* accumulative *error*. The following sequence illustrates the uses of the **FPA** command and a variable to perform a repetitive move that does not use a trigger to start it. This case is more like a cam cycle and the position relationship must be maintained while the cycle repeats.

Command	Description
> XE1	Erases sequence #1
> XD1	Defines sequence #1
VAR1=12ØØØ	Sets variable 1 equal to the first breakpoint.
VAR2=28ØØØ	Sets variable 2 equal to the second breakpoint
VAR3=Ø	Sets variable 3 equal to the primary reference point
D56ØØØ	Sets distance of secondary axis move to 56000 steps
PFZ	Zeroes the following encoder counter
L	Begins the continuous loop of the profile cycle
FOL1ØØ	The first following percentage to accelerate to is 100%
MPP	Enters Motion Profiling mode so changes can be made on-the-fly
FPA(VAR3)	Variable 3 synchronizes the move start with the primary axis. For the first 40000 primary steps, the secondary axis is moving, then it moves back during the next 40000 steps of the primary then it repeats.
G	Begins secondary axis motion.
VAR3=VAR3+8ØØØØ	Sets variable 3 equal to the start of the next cycle
FPA(VAR1)	Delays command processing until primary encoder count exceeds 12000 + 56000*n* encoder pulses, *n* = # of times through the loop.
FOL2ØØ	Changes following percentage to 200%.
VAR1=VAR1+8ØØØØ	Sets variable 1 equal to the first breakpoint for the next cycle
FPA(VAR2)	Delays command processing until primary encoder count exceeds 28000 + 56000*n* encoder pulses, *n* = the # of times through the loop.
FOL1ØØ	Changes ratio back to 100%.
VAR2=VAR2+8ØØØØ	Sets variable 2 equal to the second breakpoint for the next cycle
NG	Exits the profiling mode and complete the 56000 step move
FOL2ØØ	Sets the following ratio to a higher speed to move back to the starting point at a fast speed
H	Changes direction
G	Moves back to the starting point.
H	Changes direction
N	Repeats the cycle
XT	Ends sequence #1 definition

In this example, the move profile is repeated. One cycle consists of the following events:

① Secondary axis moves 56000 steps while primary axis moves 40000 steps.

② The secondary axis retreats to the start and after another 40000 primary encoder steps the cycle is repeated.

③ No operation during the secondary axis' retreat.

④ The retreat is set to a high following ratio to get the secondary axis back to the start before the primary axis moves 40000 steps.

This cycle is very similar to a cam cycle (which will be described in the next section). You can load the **FP** command with a variable (like the **FPA** command). You can check the following encoder counter value at any time by loading it into a variable.

Command	Description
> VAR1=FEP	Loads variable 1 with the value of the following encoder counter

Calculating FOL, FP, or FPA For An Advance or Recede Application

The advance example explained how an advance move is made and how the different commands (**FOL**$_{I}$, **FAC** and **FEN**) contribute to the move. This section provides some simple formulas that you can use to set up such a profile. To do position and velocity following, you must be able to accelerate the secondary axis to a known position with respect to the primary axis. This is what determines the **FAC** and **FEN** following acceleration values. Once you determine these values, you will use them to calculate the acceleration ratio that you must use to make the advance move. The following information will help you understand the move profile in the example.

- □ **FAC**: Following speed percentage increment
- □ **FEN**: Change in primary encoder pulses to cause an increment of **FAC**
  - **D**$_{\mathbf{prim}}$: The distance the primary axis will travel during the advance portion of the secondary move profile. This is 20000 primary encoder steps in the example above.
  - **D**$_{\mathbf{sec}}$: The distance the secondary must advance with respect to the moving primary, measured in primary encoder steps. In the example above this is 16000 primary encoder steps.
  - **FOL**$_{\mathbf{I}}$: The initial following percentage that you will accelerate from to the new following percentage.

In applications that require an advance move, you will usually know the distance that you want to advance with respect to the primary axis and the distance the primary axis will move during the advance. The distance that you want the secondary axis to advance with respect to the primary axis is given in terms of primary encoder steps. The distance can be converted from secondary motor steps to primary motor steps (and vice versa) with the **FOR** command. After determining the parameters listed above, you can use the following formula to determine the following percentage you must accelerate to. First, determine a following acceleration constant (**K**) to simplify the equations.

Equation 5-8. Following Constant

$$\mathbf{K} = \frac{\mathbf{FEN}}{100*\mathbf{FAC}}$$

The constant **K** is used in Equation 5-9 to determine **FOL**.

Equation 5-9. FOL

$$\mathbf{FOL} = \mathbf{FOL_I} + \frac{D_{prim}}{200*K} - \sqrt{\left(\frac{D_{prim}}{200*K}\right)^2 - \left(\frac{D_{sec}}{K}\right)}$$

With an advance move, the value of **FOL**$_{\mathbf{I}}$ will always be 100. Apply the formula to the example above (the following percentage should be 200%). This is the following percentage that you must attain to advance 16000 steps with respect to the primary axis, while the primary axis moves 20000 steps.

D_{prim} = 20000 primary encoder steps

D_{sec} = 16000 primary encoder steps

FAC = 0.1

FEN = 4

FOL$_{\mathbf{I}}$ = 100

We will first determine the following acceleration constant K.

$$K = \frac{4}{100*0.1} = 0.4$$

We will now determine **FOL**.

FOL is the same as $\textbf{FOL}_F$ in the equations used earlier for determining the distances traveled by the primary and secondary axes.

$$\textbf{FOL} = 100 + \frac{20000}{200*0.4} - \sqrt{\left(\frac{20000}{200*0.4}\right)^2 - \left(\frac{16000}{0.4}\right)}$$

$$= 100 + 250 - \sqrt{(250)^2 - 40000} = 350 - \sqrt{22500} = 350 - 150 = 200 = \textbf{FOL}_F$$

The following percentage that must be accelerated to is 200%. Now calculate what the breakpoint is for decelerating back to a 1:1 ratio or 100%. The **FP** value will be determined from the Equation 5-10.

Equation 5-10. Following Breakpoint

$$\textbf{FP} = D_{prim} - \frac{\textbf{FEN}}{\textbf{FAC}} * \left(\textbf{FOL}_F - \textbf{FOL}_I\right) = 20000 - \frac{4}{0.1} * (200 - 100) = 16000 \text{ steps}$$

The value you would use for **FP** is 16000. The breakpoint at which you begin the advance portion of the move profile was not calculated. This value varies from application to application and you may want to use a trigger to begin the advance move rather than **FP**. An example of using a trigger to begin an advance move is described below.

Configure an input as a trigger input with the **IN** command.

Command	Description
> IN1A	Configures input #1 as a trigger input.
> IN2D	Configures input #2 as a stop input.
> XD1	Defines sequence #1.
MC	We will make this a continuous move.
FOL1ØØ	The first following percentage to accelerate to is 100%.
MPP	Enters Motion Profiling mode so changes can be made on-the-fly.
G	Initiates motion.
TR1	Command processing pauses until input #1 (trigger input) is activated. The secondary axis will move continuously at a speed percentage of 100% (with respect to the primary axis).
FOL2ØØ	The following percentage is changed to 200%. Acceleration begins.
FP16ØØØØ	Command processing will delay 16000 primary encoder steps.
FOL1ØØ	The following percentage is changed to 100%. Deceleration begins.
NG	Exits Motion Profiling mode.
XT	Ends sequence #1 definition.

In this sequence, the secondary axis will begin moving at a 100% speed percentage. When trigger input #1 is activated, the secondary axis will advance 16000 steps with respect to the primary axis. It will then decelerate to a 100% or 1:1 ratio and continue until the stop input (input #2), is activated (or a Stop [**S**] command is issued).

Recede vs Advance

Recede moves are similar to advance moves. In the illustration of the spots for the advance example, the secondary axis synchronized with the first primary axis, then receded while the primary axis moved. This motion can be analyzed in the same manner as the advance move with the exception that a different equation is used to determine the required value for **FOL**. Again, you will have to provide the distance that the primary axis will move during the recede move and the distance with respect to the primary axis that the secondary axis must recede (measured in terms of primary encoder steps). For example:

❏ A primary encoder has a resolution of 4000 steps/rev (1 rev = 1 inch)

❏ The secondary motor has 25000 steps/revolution and also has 1 rev = 1 inch

❏ The **FOR** command is set to 6.25.

Typically you will know what distance you want the secondary axis to recede. If the secondary is to recede 1.5 inches with respect to the primary axis while the primary axis moves 3.5 inches, set D_{prim} and D_{sec} equal to:

$$D_{prim} = 3.5" = 3.5" * 4000 \frac{steps}{inch} = 14000 \text{ primary steps}$$

$$D_{sec} = 1.5" = 1.5" * 25000 \frac{steps}{inch} = 37500 \text{ secondary motor steps} = \frac{37500}{6.25} = 6000 \text{ primary encoder steps}$$

Both distances are provided in primary encoder steps. The two terms are used to determine the required **FOL** in Equation 5-11.

Equation 5-11. FOL

$$FOL = FOL_I - \frac{D_{prim}}{200*K} + \sqrt{\left(\frac{D_{prim}}{200*K}\right)^2 - \frac{D_{sec}}{K}}$$

Where K is the following acceleration constant (as in Equation 5-8).

$$K = \frac{FEN}{100 * FAC}$$

To calculate the **FOL** value, determine a value for **FAC** and **FEN**. This will depend on your application's maximum velocity and either the maximum acceleration for the secondary axis or the distance the primary axis travels while the secondary axis must accelerate. Use the same **FAC** and **FEN** values from the advance example:

FAC = 0.1

FEN = 4

In an advance or recede application, the initial following percentage FOL_I will always be 100.

The value for **FOL** produced by the equation above is:

$$\mathbf{K} = \frac{4}{100*0.1} = 0.4$$

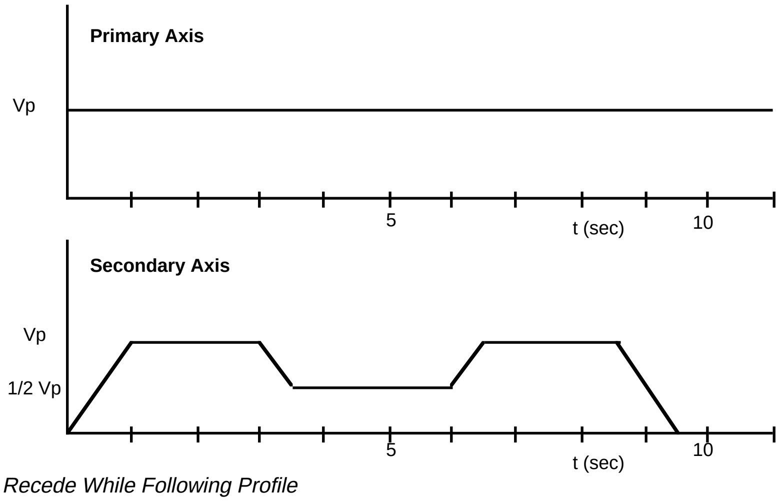
$$\mathbf{FOL} = 100 - \frac{14000}{200*0.4} + \sqrt{\left(\frac{14000}{200*0.4}\right)^2 - \frac{6000}{0.4}} = 100 - 175 + \sqrt{(175)^2 - 15000} = 50\%$$

The value needed for **FP** can be determined from Equation 5-12.

Equation 5-12. FP

$$\mathbf{FP} = D_{prim} + \frac{\mathbf{FEN}}{\mathbf{FAC}} * \left(\mathbf{FOL}_F - \mathbf{FOL}_I\right) = 14000 + \frac{4}{0.1} * (50 - 100) = 12000 \text{ primary enc. steps}$$

The move profile is shown below.



Recede While Following Profile

The sequence that will execute this profile is provided below:

Command	Description
`> FACØ.1`	Sets the percentage increment to 0.1
`> FEN4`	Sets the number of encoder counts for an increment to 4
`> FSF1`	Enables following acceleration
`> FOR6.25`	Sets the secondary motor steps per unit distance to primary encoder steps per unit distance ratio
`> XD1`	Defines sequence #1
`MN`	Enters Normal mode
`D175ØØØ`	Sets distance to 175000 steps
`FOL1ØØ`	The first following percentage to accelerate to is 100%
`MPP`	Enters Motion Profiling mode so changes can be made on-the-fly
`G`	Initiates motion
`FP12ØØØ`	The command processing will pause here until the primary encoder has moved 12000 steps. The secondary will then decelerate to 50%
`FOL5Ø`	Changes following percentage to 50%—the recede portion begins
`FP12ØØØ`	Command processing will delay 12000 primary encoder steps.
`FOL1ØØ`	The following percentage is changed to 100%—deceleration begins
`NG`	Ends the Motion Profiling mode
`XT`	End sequence definition

Cam Following

A common application that requires velocity and position following is the simulation of a cam or an electronic cam. To simulate the motion produced by a cam, you must satisfy the following requirements:

❏ Follow both the position and the velocity of a primary encoder.

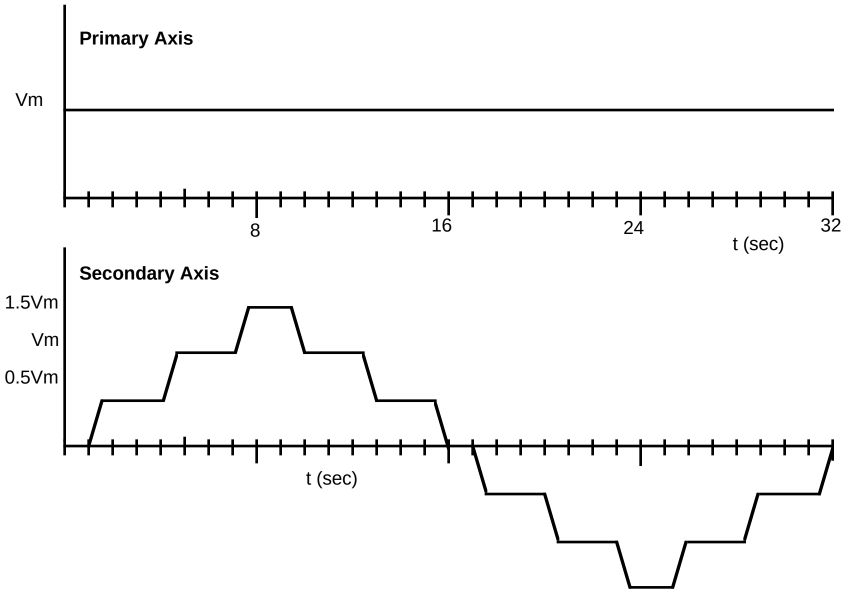
❏ You must also be able to change following ratios during motion and still maintain a positional relationship.

❏ Change ratios based on primary encoder distance.

❏ Must be able to keep track of the primary encoder position even if the secondary axis is not moving.

❏ Must be able to continuously repeat a cam cycle without developing accumulative error.

You can simulate a cam profile electronically using the commands and equations developed earlier. Motion Profiling mode (**`MPP`**) is required for cam following. For more information on Motion Profiling mode, refer to *Chapter 4, Application Design*. The next figure shows a typical cam profile.



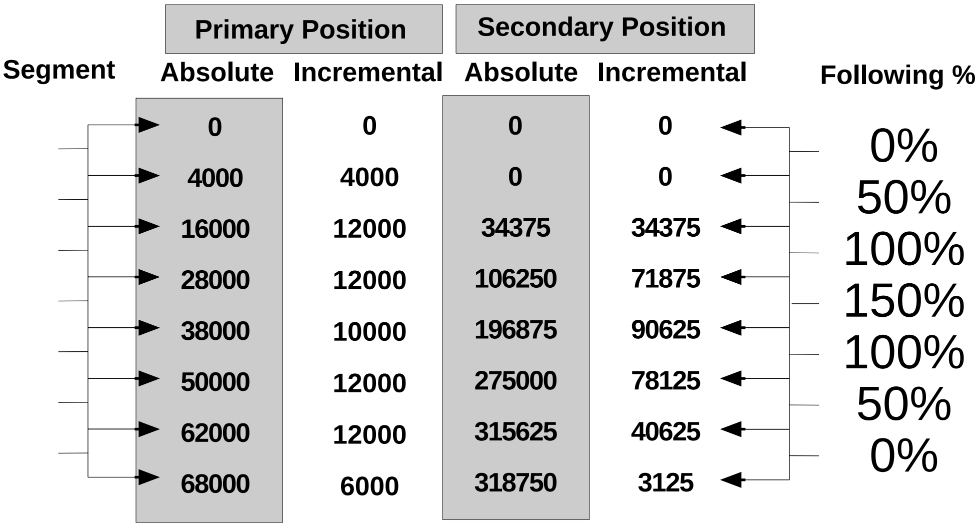
Cam Following

In this example, the encoder is a 4000 pulse per revolution encoder and it is mounted on the primary axis. The ***secondary axis*** will perform an electronic cam cycle, which consists of the following steps:

① Sitting at rest for one primary encoder revolution, then executing a step profile of 3 following percentages from 50% to 100%, then to 150%, and back down to 0%.

② It will then delay 1 primary encoder revolution and perform the same profile in the opposite direction.

③ This cycle is to be repeated until a stop is issued. The secondary axis will accelerate at its maximum rate when the primary axis is at its maximum velocity. Table 5-1 defines the cam cycle for this profile.



Segment	Primary Position Absolute	Primary Position Incremental	Secondary Position Absolute	Secondary Position Incremental	Following %
	0	0	0	0	0%
	4000	4000	0	0	50%
	16000	12000	34375	34375	100%
	28000	12000	106250	71875	150%
	38000	10000	196875	90625	100%
	50000	12000	275000	78125	50%
	62000	12000	315625	40625	0%
	68000	6000	318750	3125	

Cam Cycle

The following percentage is given for each segment. Each of the distance points is a breakpoint where the following percentage changes. Use the following acceleration to change from one following percentage to the next. The data needed to program this type of profile is listed below.

Vp_{max}: Maximum velocity of the primary axis

As_{max}: Maximum acceleration of the secondary axis

FOR: Relative resolutions per unit of distance for the primary and secondary axes

FAC: Following acceleration value, percentage increments

FEN: Following acceleration value, encoder counts for an increment

For each ratio segment, you must know the distance the primary axis will travel and the corresponding distance the secondary axis will travel. The breakpoints can be determined from the graph. Usually, you will not be able to graphically describe the motion relationship, and will simply know that you want the primary axis to move ***x*** steps and the secondary axis to move ***y*** steps in the same time frame.

After you determine **FAC** and **FEN** from your maximum acceleration and maximum velocity, or from the distance you want the secondary axis to accelerate over and the maximum velocity, you can use **FAC** and **FEN** and the primary and secondary axes' travel distances to determine the following percentages and the values for the breakpoints to change to new following percentages. When accelerating to a higher following percentage, you can use Equation 5-13 to determine the required **FOL** percentage that you must accelerate to using the **FAC** and **FEN** values you have determined.

Equation 5-13. FOL ACCEL

$$\mathbf{FOL} = \mathbf{FOL}_I + \frac{D_{prim}}{100*K} — \sqrt{\left(\frac{D_{prim}}{100*K}\right)^2 + \frac{2*\mathbf{FOL}_I*D_{prim}}{100*K} — \frac{2*D_{sec}}{\mathbf{FOR}*K}}$$

Where K is the following acceleration constant determined by **FAC** and **FEN** in Equation 5-8.

$$K = \frac{\mathbf{FEN}}{100*\mathbf{FAC}}$$

If you are decelerating to a lower following percentage, use Equation 5-14 to determine the following percentage you must use to move the secondary axis the specified number of steps for the corresponding motor steps.

Equation 5-14. FOL DECEL

$$\mathbf{FOL} = \mathbf{FOL}_I + \frac{D_{prim}}{100*K} + \sqrt{\left(\frac{D_{prim}}{100*K}\right)^2 — \frac{2*\mathbf{FOL}_I*D_{prim}}{100*K} + \frac{2*D_{sec}}{\mathbf{FOR}*K}}$$

In both the accelerating and decelerating equations, the terms listed below for the primary encoder steps and the secondary motor steps are required.

D_{prim} = The number of primary encoder steps that the motor will move in the segment.

D_{sec} = The number of corresponding secondary motor steps that the secondary will move during which time the primary encoder will move D_{prim}.

To illustrate the programming of the profile above, we will assign values to the application's requirements. The maximum velocity, acceleration and the **FOR** value are to be determined by the application. The **FAC** and **FEN** values are calculated. The maximum velocity and acceleration are given below. The example has the following parameters:

Primary encoder resolution	4000 steps/revolution
Secondary motor resolution	25000 steps/revolution
1 encoder rev	1 motor rev

FOR 6.25

V_{pmax} = 4000 steps/second

As_{max} = 1 rev/second ^{2}

Encoder sample period TF = 1 ms

From the equations in the *Velocity and Position Following* section **FAC** and **FEN** are determined:

$$\mathbf{FEN} = V_{pmax} * \frac{\mathbf{TF}}{1000}$$

$$\mathbf{FAC} = \frac{\mathbf{FOL}*Vp_{max}}{Dp_{acc}} * \frac{\mathbf{TF}}{1000}$$

Since we are starting with Vp_{max} and As_{max}, we must determine D_{acc} for the equations above.

$$Dp_{acc} = \frac{Vp_{max}^2 * \mathbf{FOR} * \mathbf{FOL}}{As_{max} * 100} = \frac{\left(4000 \frac{steps}{sec}\right)^2 * 6.25 * 100}{25000 \frac{steps}{sec^2} * 100} = 4000 \text{ primary encoder steps}$$

Vp_{max} is in primary enc. steps/sec

As_{max} is in secondary motor steps/sec^2

The **FOR** term converts the acceleration units to primary encoder steps units. **FAC** and **FEN** can now be calculated.

$$\textbf{FEN} = 4000\ \frac{\text{steps}}{\text{sec}} * \frac{1\ \text{ms}}{1000} = 4\ \text{steps}$$

$$\textbf{FAC} = \frac{100\ *\ 4000\frac{\text{steps}}{\text{sec}}}{4000\ \text{steps}} * \frac{1\ \text{ms}}{1000} = 0.1$$

You can now use these equations to determine the **FOL** value for each segment of primary encoder distance and secondary motor distance. The **FOL** values are already given in the *Cam Cycle* table, but it will be illustrative to determine the **FOL** required for some of the segments. Evaluate **FOL** for Segments #4 and #6. Typically, you will know the distance you want the primary axis to move and the corresponding distance that you want the secondary axis to move. You will have to enter the **FOL** values for the table from the equations given in this chapter.

Segment 4

D_{prim} = 10000 primary encoder steps

D_{sec} = 90625 secondary motor steps

Using the equation for accelerating, we can evaluate for **FOL** in Segment #4.

$$\text{The acceleration constant K} = \frac{4}{100*0.1} = 0.4$$

$$\textbf{FOL} = 100 + \frac{10000}{100\ *\ 0.4} - \sqrt{\frac{2*100*10000}{100*0.4} + \left(\frac{10000}{100\ *\ 0.4}\right)^2 - \frac{2*90625}{6.25*0.4}}$$

$$= 100 + 250 - \sqrt{50000 + (250)^2 - 72500} = 150\%$$

The breakpoint is given automatically by the table and is 10000 for FP and is 38000n for **FPA** where n is the number of cycles completed thus far.

Segment 6

D_{prim} = 12000 primary encoder steps

D_{sec} = 40625 secondary motor steps

Using the equation for decelerating to a lower percentage we can calculate the value for **FOL** for Segment 6.

$$\textbf{FOL} = 100 - \frac{12000}{100*0.4} + \sqrt{\left(\frac{12000}{100*0.4}\right)^2 - \frac{2*12000*100}{100*0.4} + \frac{2*40625}{6.25*0.4}}$$

$$= 100 - 300 + \sqrt{(300)^2 - 60000 + 32500} = 50\%$$

The breakpoints for **FP** and **FPA** are 12000 and 62000***n*** steps respectively. In many cases involving a cam cycle, a trigger is not used to start each cycle and the repetition of the cycle is based on the primary encoder. In these situations, use the **FPA** command (it is based on the following encoder's absolute count). The absolute count comes from a hardware counter that can be accessed by assigning it to a variable:

```
> VAR1=FEP           FEP is the value in the hardware counter, it is a read only value
```

The following sequence will perform the cam profile. The secondary axis will be put in a continuous move. Two parts occur in a cycle. The first part moves the stepped profile in one direction. The second part reverses direction and returns to the start to repeat the cycle.

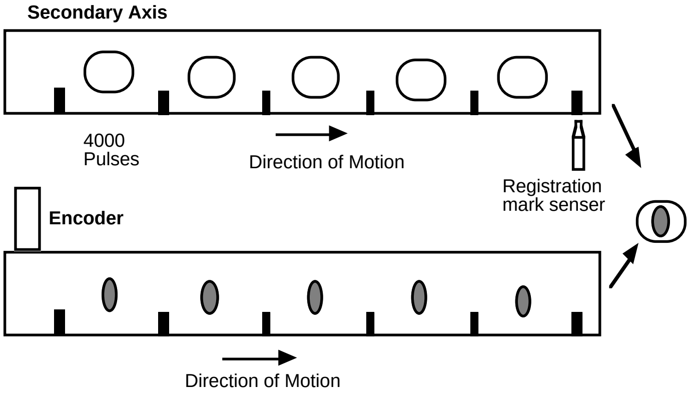
Command	Description
> **FOR6.25**	Sets the secondary motor steps to primary encoder steps ratio
> **FAC0.1**	Sets the change in following percentage for following acceleration
> **FEN4**	Sets the number of encoder pulses required to change by **FAC**
> **FSF1**	Enables following acceleration
> **FSI1**	Enables following
> **SSH1**	Saves buffer on stop
> **VAR1=64ØØØ**	Variable for the incrementing the cycle
> **VAR2=4ØØØ**	Breakpoint 1
> **VAR3=16ØØØ**	Breakpoint 2
> **VAR4=28ØØØ**	Breakpoint 3
> **VAR5=38ØØØ**	Breakpoint 4
> **VAR6=5ØØØØ**	Breakpoint 5
> **VAR7=62ØØØ**	Breakpoint 6
> **VAR8=64ØØØ**	Breakpoint 7
> **1XE1**	Erases sequence #1
> **1XD1**	Defines sequence #1
MC	Enables continuous mode
FOLØ	Sets the current following percentage to 0%
L	Begins the loop cycle
MPP	Enters the Motion Profiling Mode
G	Initiates motion
FPA(VAR2)	Pauses execution until absolute value of the primary encoder counter exceeds breakpoint 1
FOL5Ø	Following % is changed to 50% or 1/2 as fast as the primary motor
VAR2=VAR2+VAR1	Set variable 2 to the breakpoint value for the next cycle
FPA(VAR3)	Pauses execution until absolute value of the primary encoder counter exceeds breakpoint 2
FOL1ØØ	Speed ratio is changed to 1:1
VAR3=VAR3+VAR1	Set variable 3 to the breakpoint value for the next cycle
FPA(VAR4)	Pauses execution until absolute value of the primary encoder counter exceeds breakpoint 3
FOL15Ø	Speed ratio is changed to 1.5:1
VAR4=VAR4+VAR1	Set variable 4 to the breakpoint value for the next cycle
FPA(VAR5)	Pauses execution until absolute value of the primary encoder counter exceeds breakpoint 4
FOL1ØØ	Speed ratio is changed to 1:1
VAR5=VAR5+VAR1	Set variable 5 to the breakpoint value for the next cycle
FPA(VAR6)	Pauses execution until absolute value of the primary encoder counter exceeds breakpoint 5
FOL5Ø	Speed ratio is changed to 5:1
VAR6=VAR6+VAR1	Set variable 6 to the breakpoint value for the next cycle
FPA(VAR7)	Pauses execution until absolute value of the primary encoder counter exceeds breakpoint 6
FOLØ	Speed ratio is changed to 0
VAR7=VAR7+VAR1	Set variable 7 to the breakpoint value for the next cycle
FPA(VAR8)	Pauses execution the absolute value of the primary encoder counter exceeds breakpoint 7
STOP	Ends the move (this is required)
NG	Ends Motion Profiling mode
VAR8=VAR8+VAR1	Set variable 8 to the breakpoint value for the next cycle
H	change the direction
N	Ends the loop cycle
XT	Ends sequence #1 definition

This sequence is an example of a complex following profile. Position and velocity are synchronized and the positional relationship is maintained.

Synchronization

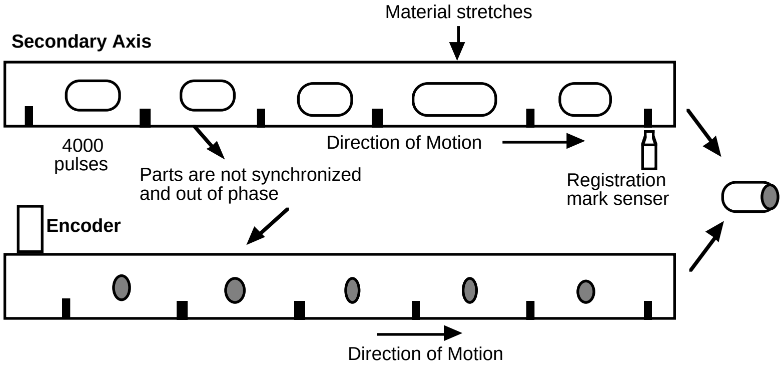
The SXF can synchronize its speed and phase with respect to a primary axis. In many applications, it is necessary to have the position and speed of a secondary axis synchronized with the speed and position of the primary axis with registration marks on the secondary axis parts or material. These marks must be evenly spaced so that at a constant speed (with respect to the primary axis) the number of primary axis encoder steps recorded between registration marks is an expected constant number. If these marks should come further apart (e.g., the material stretches) the SXF will adjust the speed ratio to correct for the error between the registration marks. The following figures illustrate this process.

In the figure below, a secondary axis has parts that are to be synchronized to the primary axis' parts. The registration sensor detects the location of the parts with respect to the primary axis. This sensor goes to the SXF. It indicates the start of the part. The SXF then counts the encoder pulses from the primary axis that occur between registration marks.



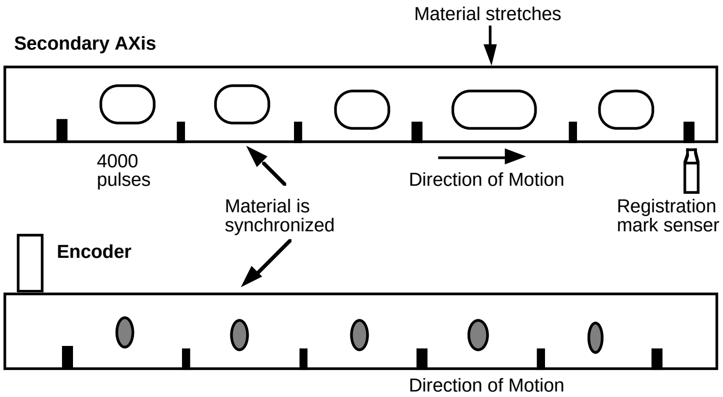
Secondary Parts Synchronized With the Primary Axis

If the material on the secondary axis stretches, as indicated in the following figure, all parts after the stretched material will no longer be synchronized with the primary axis' parts. The next figure depicts the result of not using Synchronization mode.



Material Stretches—Parts Out of Sync

With the SXF's Synchronization mode, the secondary axis accelerates to re-synchronize with the primary axis. It removes the phase shift between the two axes. The SXF detects that the number of pulses between the registration marks has increased (due to stretching). The speed ratio is increased, so the secondary axis speeds up. The material after the stretched portion is good material, so the pulses between the next two registration marks will be slightly less than 4000 because the speed ratio is now higher. The speed will now be reduced. The secondary and primary axes will be synchronized again. Every time a registration mark is encountered, a new actual count is latched and the speed ratio is adjusted to synchronize the axis. Corrections will continue until the secondary axis again has the expected number of pulses between registration marks.



SXF Synchronizes Parts (After Stretching)

You can determine and program the amount of correction that is applied to the speed ratio between each registration mark to fit your application. Use the following commands to program a synchronization application.

Command	Description
> FC	Expected encoder count between registration marks
> FBS	Normalizing count for determining new speed ratio to move at
> FIN	Increment used to determine the new following speed
> FSL	Enables the Synchronization mode
> FSK	Enables the expected encoder count teach mode
> INnI	Defines an input as the registration mark synchronization input.
> TF	Sets the sample rate of the encoder input
> FOR	Motor to encoder count ratio
> FOL	Primary to secondary axis speed ratio
> FSI	Enables Following mode

The **FOL** and **FOR** commands determine the number of secondary motor steps that will be commanded for the encoder pulses that are received. The **FBS** and **FIN** commands determine the amount of correction that will be applied to the **FOL** and **FOR** motor to encoder ratio. **FC** is the number of encoder pulses that are expected between the registration marks if the speeds are synchronized properly. This number is compared to the actual number of pulses that are counted between each registration mark. The difference between these two values represents the error. The **FIN** and **FBS** numbers are applied to this error to determine the new speed ratio.

The **INnI** command configures an input to the SXF for accepting the signal from the sensor that detects the registration marks on the parts or material. The **TF** command is the rate at which the encoder interface is sampled. The **FSL** command enables the Synchronization mode. When Synchronization mode is enabled the first time, the Synchronization input is toggled, the encoder counter is started. The next time it is toggled, the count is latched and the counter is reset to zero. The latched count is the actual number of encoder pulses counted between registration inputs. This is compared to the expected value (**FC**). The difference is multiplied by the correction factor (**FIN** and **FBS**) to determine the speed ratio to run at until the synchronization input is toggled again. This process is continuous as long as the synchronization mode (**FSL**) is enabled.

If you do not know the expected encoder count between registration marks, use the Self Learn mode (**FSK**) to determine the expected count (**FC**). To determine the expected count (**FC**) using the Self Learn mode (**FSK**) you must start the process at the speed ratio that you want to run at, turn on the Self Learn mode. The SXF will count the pulses between the registration marks. When the secondary axis stops, the last recorded number will be placed in the expected count (**FC**) number. This will be used when you are in Synchronization mode.

As an example, the process in the previous figures use an encoder that has 4000 pulses per revolution and the secondary motor has 25000 steps per revolution. The **FOR** command is set to 6.25. The motor is mounted on the secondary conveyer belt so that one revolution is 4 inches. The encoder is mounted on the primary axis so that one encoder revolution is 4 inches. The **FOL** command must be set to 100% (**FOL1ØØ**) for the secondary axis to move at the same speed as the primary axis. If the primary axis moves at 1 rps, the secondary axis must move at 1 rps. In one Figure, the two axes start moving at the same speed. The registration marks are 4 inches apart on the secondary axis. Use the Self Learn mode to determine how many encoder pulses are between the registration marks on the material on the secondary axis. The following steps show how to program the application.

Step ① Set up Self-Learn mode.

Command	Description
> **FOR6.25**	Set the motor to encoder ratio to 6.25
> **FOL1ØØ**	Set the motor to encoder ratio speed percentage to 100%
> **FSI1**	Enable the Following mode
> **FSK1**	Enable the Self Learn mode
> **TF1**	Set the encoder sample rate to 1 ms
> **A5ØØ**	Set the acceleration of 500 rps^2
> **AD5ØØ**	Set the deceleration of 500 rps^2
> **MC**	Place the SXF in the continuous mode
> **IN1I**	Defines input #1 as a synchronization input

Step ② The registration sensor should be wired to the synchronization input. The primary and secondary axes should now be started. After the SXF has passed more than 3 registration marks, it can be stopped. The number of encoder pulses between registration marks can be checked with the **FC** command. In this example, the number that is determined is 4000 counts.

Command	Response
> **1FC**	*4000

If you know the number of encoder pulses you expect to record between registration marks, this number can be entered directly for the **FC** command and will override the number determined in Self Learn mode.

Command	Description
> **1FC4ØØØ**	Manually entering the expected count

Step ③

The SXF now has the number of counts expected between registration marks. The next step is to determine the correction gain desired. The correction will be applied to the difference between the expected count that was just determined and the actual counts that will be counted during actual operation. The equation for the determination of the correction is;

Equation 5-15. Correction

$$\text{Correction} = \frac{(\text{Actual encoder count - Expected encoder count}) * \mathbf{FIN}}{\mathbf{FBS}}$$

This motor-to-encoder step ratio is determined by the following equation.

Equation 5-16. Speed Ratio

$$\text{New Speed Ratio} = \mathbf{FOR} * \mathbf{FOL} + \text{Correction}$$

To determine the number of motor steps that will be commanded for the number of encoder pulses received, use the following equation.

Equation 5-17.
Motor Step
Correction

Motor Steps = Encoder Steps * (**FOR*FOL** + Correction)

You must determine the amount of correction you want to have for a given amount of error. Once this has been determined, you can enter the **FIN** and **FOR** commands.

Command	Description
> **1FIN3.12**	The following increment is 3.12
> **1FBS1ØØ**	The following base number is 100

Step ④ Disable Self Learn mode and enable Synchronization mode.

Command	Description
> **1FSKØ**	Disable Self Learn mode
> **1FSL1**	Enable Synchronization mode

Step ⑤ Orient the primary and secondary axes to attain the desired phase relationship. Start both axes at the same time, or start the SXF first and the primary axis second.

The SXF will now correct any errors in the phase relationship between the two axes and maintain a synchronized speed. The new speed ratio that is determined will be applied for the entire period between registration marks. The time between the registration marks is effectively the sample period. A correction is made for each sample period.

Another method for synchronization is to use the inputs to the SXF for increasing and decreasing the following speed ratio. The SXF inputs can be defined to increase or decrease the following ratio. By setting one input for increasing the following ratio and one input for decreasing the following ratio, synchronization can be achieved. In this case, use an external circuit to determine whether the secondary axis should accelerate (increase ratio) or decelerate (decrease ratio) the secondary axis.

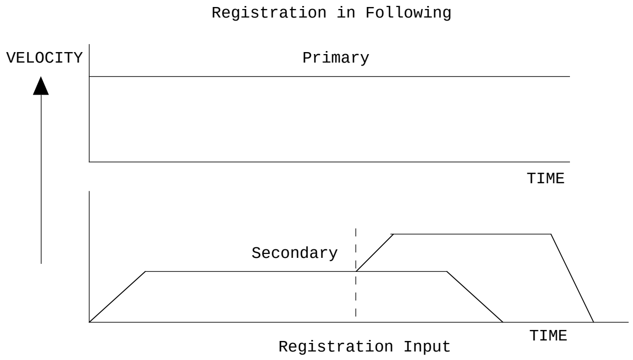
Define the input with the **IN** command. **INnX** defines the input for increasing following ratio, **INnY** defines the input for decreasing the following ratio. The following ratio will be increased or decreased while the input is active. During the SXF's sample periods, the ratio will increase or decrease while the inputs are active. The inputs have a 2 ms debounce time. If the input remains active for 4 ms, the following ratio will be increased or decreased twice. The amount that the following ratio is increased or decreased is determined by the **FIN** command. If **FIN** is 1, **FOL** is increased or decreased by 1 during each sample period.

Other Following Features

This section discusses following features that the SXF provides for special following requirements.

Registration in Following Mode

With the SXF, registration can be performed in Following mode. It is like programming registration in the Indexer version, *but the velocity term is replaced by* **FOL** *for the desired speed*. This figure illustrates registration in Following mode.



Registration in the Following Mode

Registration can only be performed if the Following Synchronized Acceleration mode is disabled (**FSFØ**). For this example, re-attach the encoder to the primary axis.

Step ① Repeat steps 1 - 5 of the *Velocity Following Example.*

Step ② Use the dedicated registration input to start a registration move

Step ③ Define registration move #1 as follows:

```
> REG1,A1Ø,AD1Ø,FOL2ØØ,D25ØØØ
```

Step ④ Begin motion on the primary axis, then begin motion with the SXF.

Command	Description
> MC	Change to continuous mode
> G	Initiate motion

Step ⑤ Toggle the registration input. The motor will begin following at a speed ratio of 2:1 for 25,000 motor steps.

Jogging in Following Mode

In some applications, you may want to move the motor manually while in Following mode. This allows you to follow the primary axis at a following ratio by toggling a switch. You can configure the SXF to allow you to follow the primary motor manually with the Configure Input (**IN**) command. Jogging in Following mode does not require the **JHV** or **JVL** commands. In this case, you will jog at whatever speed ratio the **FOL** command is set to. To use the inputs, you can either configure the input as a CW or a CCW jog as with the preset Indexer jog. However, you cannot use the high-speed/low-speed jog input because you can only jog at the speed set by the **FOL** speed ratio.

Therefore, you use the two jog input functions: CW Jog input (**IN#J**) and CCW Jog Input (**IN#K**). You must also enable the jogging feature with the **OSE1** command. Once you set these parameters, you can attach a switch to the jog inputs (predefined) and jog the motor(s). The **#** character represents digits 1 - 8, which you enter. You must have the SXF in Following mode to jog at a speed ratio of the primary. The following example shows you how to define power-up sequence #100 to jog.

Step ① Define a power-up sequence. ***Position Tracking mode must be disabled* (FSPØ) *to enable direction jogging.***

Command	Description
> XE1ØØ	Erases sequence #100
> XD1ØØ	Defines sequence #100
> LD3	Disables the limits(*not needed if you have limit switches installed*)
> JA25	Sets Jog Acceleration to 25 rps^2
> JAD25	Sets Jog Deceleration to 25 rps^2
> OSE1	Enables Jog function
> IN1J	Sets **IN 1** as a CW jog input
> IN2K	Sets **IN 2** as a CCW jog input
> FOR6.25	Sets the motor to encoder steps
> FOL75	Secondary moves at 75% of the primary speed
> FSI1	Enables Following mode
> XT	Ends sequence definition

Step ②

Command	Description
> Z	Resets the SXF

Reset the SXF. Move the primary or primary axis.

Step ③ Turn on **IN 1** to move the motor CW at 75% of the primary axis' speed (until you turn **IN 1** off).

Step ④ Turn on **IN 2** to move the motor CCW at 75% of the primary axis' speed (until you turn **IN 2** off).

Following a Step and/or Direction Signal

The SXF can follow a step (or pulse) and a direction signal rather than quadrature encoder pulses. The same incremental encoder interfaces used for quadrature following are used for the step and direction following. The **Phase A+** and **Phase A-** inputs are now the **Step+** and **Step-** inputs. The **Phase B+** and **Phase B-** inputs are now the **Direction+** and **Direction-** inputs. The only other requirement is to put the SXF in the pulse and direction mode. This is accomplished with the **FSN** command. By typing **FSN1** the pulse and direction capability is enabled and the **Phase A** and **Phase B** inputs are now step and direction inputs. Once the pulse and direction capability is added, any following applications are performed exactly as if the input were quadrature signals. If your application requires pulse and direction, enable the Pulse and Direction mode and repeat the procedures in this following section.

Following Equation and Command Summary

This section provides a reference for the following equations and the SXF software commands that are associated with following.

Following Command Summary

For a complete explanation of these following commands, refer to the ***SX Software Reference Guide***. *Set-up commands are required for any following application.*

FSA Followed by a 1—enables instant acceleration between commanded velocities for each resulting velocity change after sampling the encoder and determining the ratio.

FSF Followed by a 1—enables the use of following acceleration as determined by **FAC** and **FEN**.

FSI Followed by a 1—enables following mode versus Indexer mode

FSK Followed by a 1—enables the calculation of **FC** for Synchronization mode.

FSL Followed by a 1—enables Synchronization mode

FSN Followed by a 1—enables a step & direction signal to be followed or just a pulse train if direction is not used in conjunction with **FSP**.

FSP Followed by a 1—enables position tracking.

FOR The number of secondary motor pulses per unit of travel divided by the primary encoder pulses per the same unit of travel.

FOL The % of the primary encoder speed that the secondary axis moves at.

FAC The change in following % for each change in encoder pulse count of **FEN** during following acceleration.

FEN The number of encoder pulses that the encoder count must change by to increment the following percentage by **FAC.**

FP In Motion Profiling (MPP) mode, command execution is paused for the number of following encoder steps entered in the **FP** command.

FPA In Motion Profiling (MPP) mode, the execution of commands pauses until the value in the encoder counter exceeds the **FPA** value.

FIN The amount by which the following % changes when changed by inputs.

FBS In synchronization, used with **FIN** to determine the amount of following percentage correction.

FC In synchronization, the expected number of encoder counts between registration marks.

TF The following encoder sample period

VARn=FEP **FEP** is a read only variable of the actual encoder count. Set it equal to the variable to get the current value of the encoder counter.

PF Gives you a report back of the encoder count.

PFZ Clears the encoder counter.

INnX An input function that lets you increase the following percentage by **FIN**.

INnY An input function that lets you to decrease the following percentage by **FIN**.

INnI An input function used for the registration mark sensor of the synchronization mode.

The commands are categorized according to the applications the support.

Velocity Following	***Velocity and Position Following***	
FOR	FOR	FSF
FOL	FOL	FEN
FSI	FSI	FAC
TF	TF	FSA
		FSP

Advance and Recede		***Synchronization***		***Special Function***	
FOR	FSF	FOR	FSF	FBS	FSN
FOL	FAC	FOL	FAC	FSK	FSM
FSI	FEN	FSI	FEN	FSL	PF
TF	FP	TF	FC	NnI	PFZ
	FPA	INnY	FIN	INnX	

Following Equation Summary

The following equations were discussed throughout this chapter. They are provided again here for reference and convenience

Velocity Following

$$\text{Secondary Motor steps} = \text{Primary encoder count} * \textbf{FOR} * \frac{\textbf{FOL}}{100}$$

Velocity and Position Following

Given V_{max} and D_{acc}:

$$\textbf{FEN} = V_{max} * \frac{\textbf{TF}}{1000} \qquad V_{max} = \frac{\text{Primary encoder counts}}{\text{second}}$$

TF = Primary Encoder Sample Period in ms

$$\textbf{FAC} = \frac{\textbf{FOL} * V_{max}}{D_{acc}} * \frac{\textbf{TF}}{1000} \quad V_{max} = \frac{\text{Primary encoder counts}}{\text{second}}$$

TF = Primary Encoder Sample Period in units of ms

FOL = Following percentage in units of percent

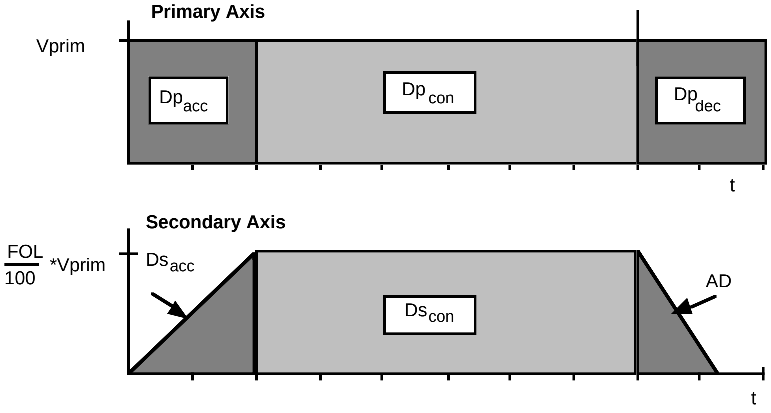
$\mathbf{D_{acc}}$ = Distance primary axis moves during secondary axis acceleration in units of primary encoder counts

Given V_{max} and A_{max} Determine D_{acc} from $D_{acc} = \frac{{V_{max}}^2}{A_{max}} * \textbf{FOR} * \frac{\textbf{FOL}}{100}$

The acceleration is in units of secondary motor steps/sec2. The maximum velocity of the primary is in primary motor steps/sec.

The following figure illustrates the motion profiles of a secondary and the primary. The different parameters are shown on the profiles and the equations to determine the parameters are given below. **AD** is used for the deceleration.

Velocity and Position Following With AD Decel



$$Ds_{acc} = \text{FOR} * \left(\frac{1}{2} * \Delta\textbf{FOL}^2 * \frac{\textbf{FEN}}{100*\textbf{FAC}} + \Delta\textbf{FOL}*\textbf{FOL}_I * \frac{\textbf{FEN}}{100*\textbf{FAC}} \right)$$

FOL= The change in following percentage

$\mathbf{FOL_I}$ = The initial following percentage

The equations above can be simplified by defining a following acceleration constant determined by **FAC** and **FEN**.

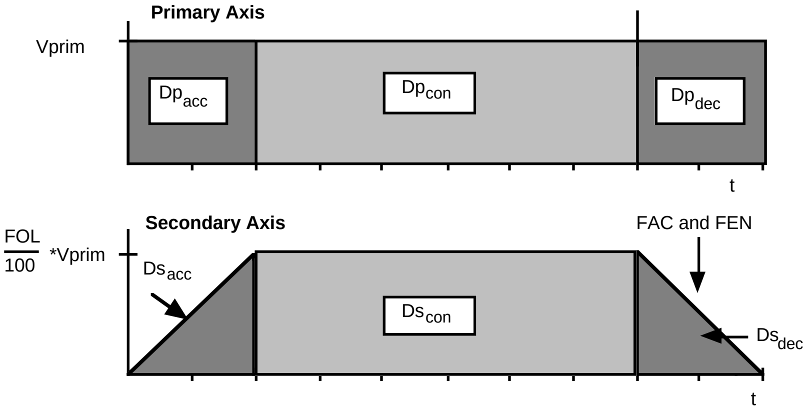
$$K = \frac{\mathbf{FEN}}{100*\mathbf{FAC}}$$

The equations can now be written as:

$$D_{sec} = \mathbf{FOR} * K * \left(\frac{1}{2} * \Delta\mathbf{FOL}^2 + \Delta\mathbf{FOL}*FOL_I\right)$$

$$D_{sec}\ \text{(Deceleration)} = -\mathbf{FOR} * K * \left(\frac{1}{2} * \Delta\mathbf{FOL}^2 + \Delta\mathbf{FOL}*FOL_I\right)$$

Use the equations and parameters in the next figure To make a trapezoidal move (deceleration is done according to the **FAC** and **FEN** commands).



Velocity and Position Following With Following Decel

$$D_{prim} = Dp_{acc} + Dp_{con} + Dp_{dec}$$

$$D_{sec} = Ds_{acc} + Ds_{con} + Ds_{dec}$$

$$Ds_{acc} = \mathbf{FOR} * K * \left(\frac{1}{2} * \Delta\mathbf{FOL}^2 + \Delta\mathbf{FOL}*FOL_I\right)$$

$$Ds_{dec} = -\mathbf{FOR} * K * \left(\frac{1}{2} * \Delta\mathbf{FOL}^2 + \Delta\mathbf{FOL}*\mathbf{FOL}_I\right)$$

$$Dp_{acc} = \mathbf{FOL} * \frac{\mathbf{FEN}}{\mathbf{FAC}}$$

$$Ds_{con} = Dp_{con} * \mathbf{FOR} * \frac{\mathbf{FOL}}{100}$$

$$\mathbf{FOL} = \frac{D_{prim}}{200*K} - \sqrt{\left(\frac{D_{prim}}{200*K}\right)^2 - \left(\frac{D_{sec}}{K}\right)}$$

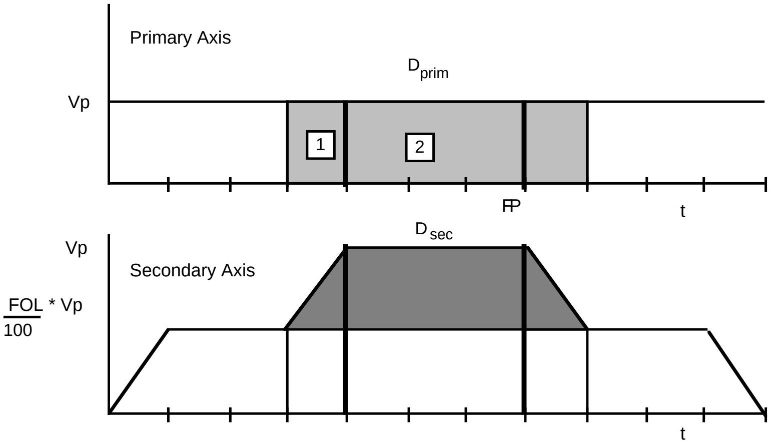
K is the following acceleration constant, D_{prim} and D_{sec} are the distances that the primary axis and secondary axes will move, respectively. $\mathbf{FOL}_I$ is the initial following percentage. In the case of a trapezoidal move, it will always be 0. The required sequence is:

Command	Description
`> XE1`	Erases sequence #1
`> XD1`	Defines sequence #1
`FSI1`	Enables Following mode
`FORn`	Sets motor to encoder steps per unit travel ratio
`FACn`	Sets the following acceleration increment
`FENn`	Sets the encoder period
`Dn`	Sets the secondary axis distance to n motor steps
`FOLn`	Sets the initial following percentage to n
`TR1`	Waits on the input trigger
`MPP`	Enters the Motion Profiling mode
`G`	Starts motion
`FPa`	Waits until a encoder pulses have passed
`FOLØ`	Stops the motion of the secondary
`FPb`	Waits for the decel ramp distance
`STOP`	Stops the move itself
`> XT`	Ends the sequence #1 definition

The value for a in the first **FP** = $Dp_{acc} + Dp_{con}$

The value for b in the second **FP** = Dp_{dec}

Advance and Recede



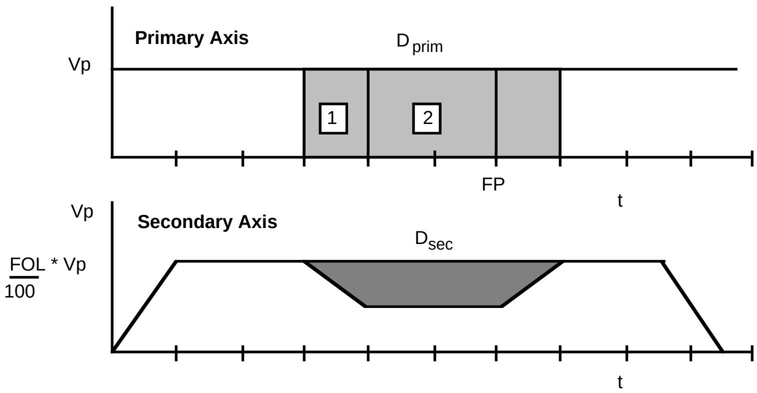
Advance Profile Mode

In the figure shown below, D_{prim} is the distance that the primary axis moves in encoder steps during the advance portion of the profile (lightly shaded region). D_{sec} is the distance in secondary motor steps that the secondary axis advances with respect to the moving primary axis (darkly shaded region).

$$\mathbf{FOL} = \mathbf{FOL}_I + \frac{D_{prim}}{200*K} - \sqrt{\left(\frac{D_{prim}}{200*K}\right)^2 - \left(\frac{D_{sec}}{\mathbf{FOR}*K}\right)}$$

The .i.breakpoint; to decelerate to FOL1ØØ is entered for the FP value in the advance sequence. FP is a distance equal to the sum of areas 1 and 2 in the primary profile.

$$\mathbf{FP} = D_{prim} - \frac{\mathbf{FEN}}{\mathbf{FAC}} * \left(\mathbf{FOL}_F - \mathbf{FOL}_I\right)$$



Recede Profile

The dark shaded area is the distance that the secondary will recede with respect to the moving primary. The lightly shaded are is the distance the primary will move while the secondary recedes.

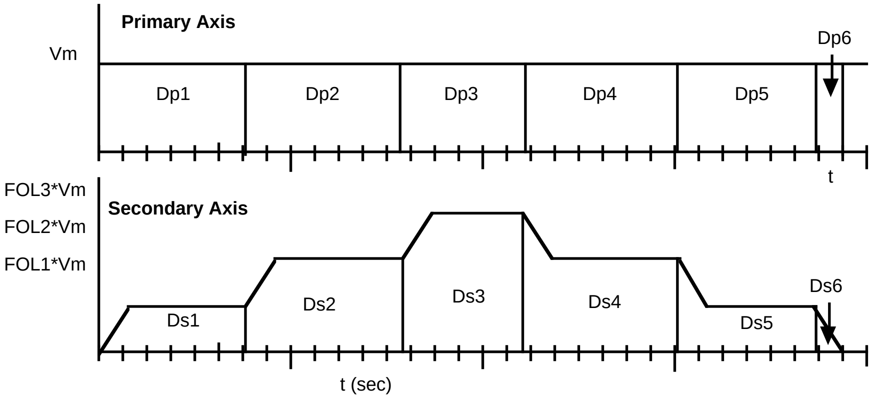
$$\mathtt{FOL} = \mathbf{FOL_I} - \frac{D_{prim}}{200*K} + \sqrt{\left(\frac{D_{prim}}{200*K}\right)^2 - \frac{D_{sec}}{\mathtt{FOR}*K}}$$

K is the following acceleration constant and is equal to: $\frac{\mathbf{FEN}}{100*\mathbf{FAC}}$

FP is equal to: $D_{prim} + \frac{\mathbf{FEN}}{\mathbf{FAC}} * (\mathbf{FOL}_F - \mathbf{FOL}_I)$

Cam Following

The following figure is a cam profile.



Cam Profile

Each segment of the secondary and primary move profile is marked by a primary distance Dpn and a secondary distance Dsn. The following percentage required for each segment is determined from the equations below. D_{prim} and D_{sec} are equal to Dpn and Dsn for each segment. When accelerating to a higher percentage use.

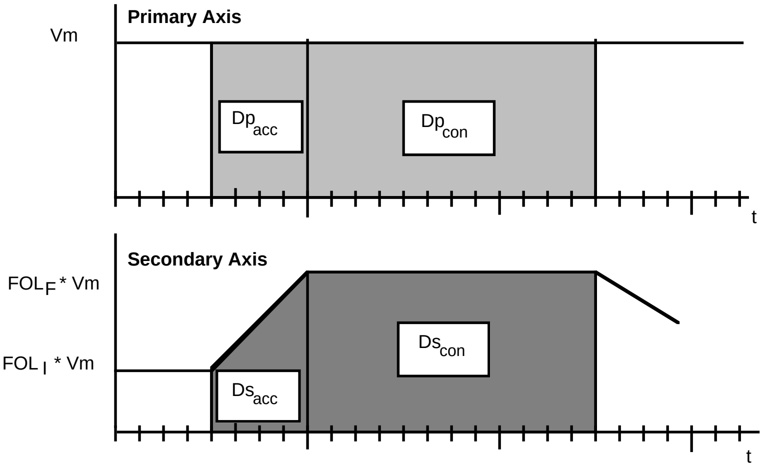
$$\mathbf{FOL} = \mathbf{FOL}_I + \frac{D_{prim}}{100*K} - \sqrt{\left(\frac{D_{prim}}{100*K}\right)^2 + \frac{2*FOL_I*D_{prim}}{100*K} - \frac{2*D_{sec}}{\mathbf{FOR}*K}}$$

When decelerating to a lower percentage, use:

$$\mathbf{FOL} = \mathbf{FOL}_I - \frac{D_{prim}}{100*K} + \sqrt{\left(\frac{D_{prim}}{100*K}\right)^2 - \frac{2*\mathbf{FOL}_I*D_{prim}}{100*K} + \frac{2*D_{sec}}{\mathbf{FOL}*K}}$$

FP or FPA are equal to Dpn or the accumulative Dpn respectively.

Each segment of the cam can be broken down as shown in below and described the equations below.



Cam Profile Segment

$$D_{prim} = Dp_{acc} + Dp_{con}$$

$$D_{sec} = Ds_{acc} + Ds_{con}$$

$$Ds_{acc} = FOR * K * \left(\frac{1}{2} * (FOL_F - FOL_I)^2 + (FOL_F - FOL_I) * FOL_I \right)$$

$$Dp_{acc} = FOL_F - FOL_I * \frac{FEN}{FAC}$$

$$Ds_{con} = Dp_{con} * FOR * \frac{FOL}{100}$$

The distance for FP is $Dp_{acc} + Dp_{con}$, for FPA, is the absolute value of the primary distance since the start of the cam cycle.

CHAPTER ⑥

Hardware Reference

Chapter Objectives

The information in this chapter will enable you to:

- ❏ Use this chapter as a quick-reference tool for most system specifications (dimensions and performance)
- ❏ Using this chapter as a quick-reference tool for DIP switch settings

Environmental Specifications

Drive Temperature

122°F (50°C) ambient air temperature, measured at the heatsink fins. An internal thermostat will shut down the drive if the unit reaches 158°F (70°C) internally. Current settings in excess of 4A in high ambient temperature environments, above 113°F (45°C), may require fan cooling to keep drive temperature within allowable limits and to keep the drive from shutting itself down due to overtemperature. Low temperature of 32°F (0°C).

Motor Temperature

212°F (100°C) maximum allowable motor case temperature. Actual temperature rise is duty cycle dependent.

Humidity

0 - 95%, non-condensing

Drive Electrical Specifications

Input Power

90VAC to 132VAC @50/60Hz, Low voltage fault below 85VAC

Output Power

- ❏ Low power: 0.1 to 6 amps per phase at 170 VDC (PWM)
- ❏ High power: 0.2 to 8 amps per phase at 170 VDC (PWM)

Motor Output

- ❏ Two phase Mosfet bipolar (H-bridge) switching at 20kHz (nominal), recirculating current, pulse width modulated.

I/O Electrical Specifications

Rx & Tx (RS-232C)

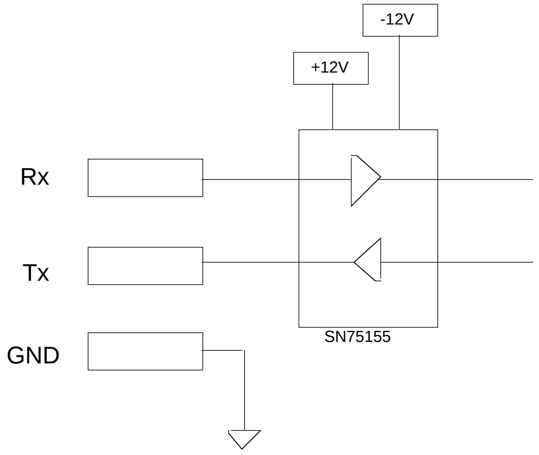
❑ Rx± 24VDC maximum input voltage

❑ High-level input 2VDC minimum

❑ Low-level input 0.8VDC maximum

❑ Tx± 11VDC output voltage typical

❑ 10 mA current limited output.



Schematic:RS-232C Input

OPTO1 and OPTO2

The **OPTO1** and **OPTO2** terminals are the (5-12VCD) source inputs for the limit, home, registration, and general-purpose inputs. The voltage range is from +5 VDC to +12 VDC. A diode must be used for 13VDC - 24VDC voltage supplies, or OP1-HV/OP2-HV may be used without the zener diode.

☞ **Helpful Hint: Zener Diode Specifications**

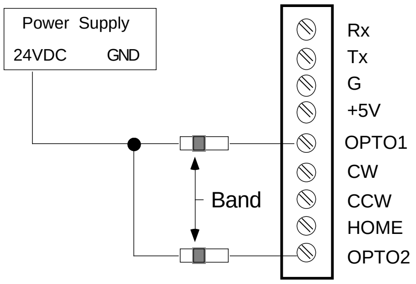
If a voltage source from 13VDC - 24VDC is used, a Zener Diode must be placed in series with the voltage source. Voltages from 13VDC - 24VDC cannot be wired directly to **OPTO1** or **OPTO2** (this voltage would overdrive the internal components and damage the SX Indexer/Drive). The Zener Diodes limit the maximum voltage seen by the **OPTO1** and **OPTO2** inputs. The following Zener Diode values are recommended:

❑ Supply = 13VDC - 17VDC—Use a 5-watt Zener Diode with nominal Zener voltage of 6.8VDC

- Motorola 1N5342, General Semiconductor 1N5342, Microsemi Corp., 1N5342, Diodes Inc. 1N5342, SGS Thompson 1N5342

❑ Supply = 17VDC - 24VDC—Use a 5-watt Zener Diode with nominal Zener voltage of 12VDC

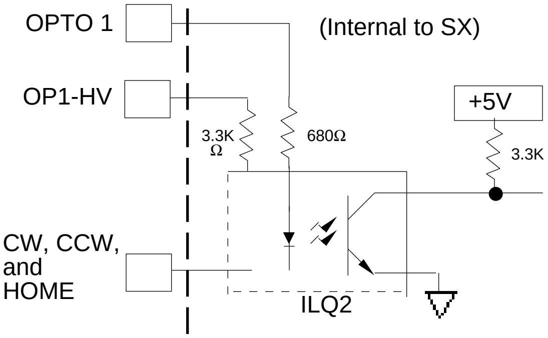
- Motorola 1N5349, General Semiconductor 1N5349, Microsemi Corp., 1N5349, Diodes Inc. 1N5349, SGS Thompson 1N5349



Zener Diode

CW, CCW, and HOME Inputs

The inputs are optically isolated and may be driven by providing a negative signal with respect to the **OPTO1** input. The input driver must be capable of providing a minimum sink current of 2 mA to ensure proper operation. The maximum reverse voltage on these terminals is -3VDC with respect to the **OPTO1** input (**OPTO1** +3 VDC).

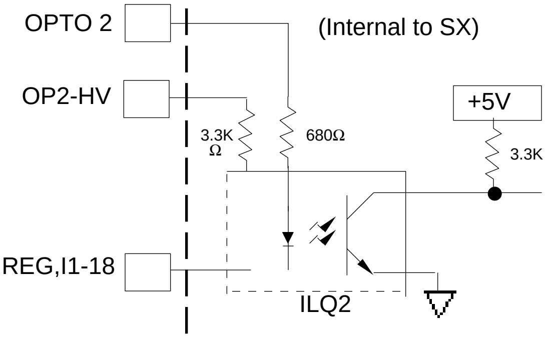


Note: OPTO1 is for use with (5-12VDC) power supplies and OP1-HV is for use with (12-24VDC) power supplies.
They should not be used together.

CW, CCW, and Home Inputs

Registration General-Purpose Inputs (I1-I8)

The inputs are optically isolated and may be driven by providing a negative signal with respect to the **OPTO2** input. The input driver must be capable of providing a minimum sink current of 2 mA to ensure proper operation. The maximum reverse voltage on these terminals is -3 VDC with respect to the **OPTO2** input (**OPTO2** +3 VDC).



Note: OPTO2 is for use with (5-12VDC) power supplies and OP2-HV is for use with (12-24VDC) power supplies.
They should not be used together.

REG and I1 - I8 Inputs

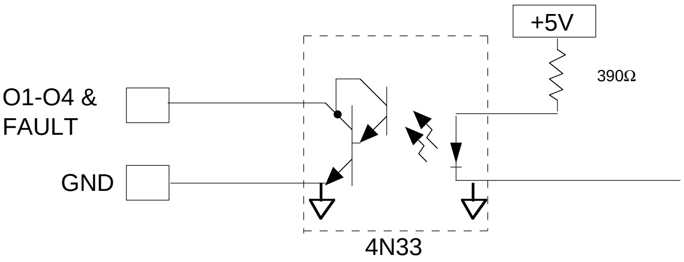
CAUTION

The maximum reverse voltage across OPTO1 and OPTO2 and their corresponding inputs is 3VDC. A zener diode or blocking diode may be required on the input as well if applying 13-24VDC to the inputs from a PLC output or other source.

General-Purpose Outputs (O1-O4) and Fault Output

The general-purpose outputs and the fault output are optically isolated darlington drive transistors. The maximum sink current is 35 mA with respect to the GND terminal. The maximum pull-up voltage at these terminals is 24VDC. The maximum reverse voltage at these terminals is -5VDC (GND - 5VDC). To provide a stable output signal, a maximum pull-up resistor of 1K is recommended.

$R_L \geq \frac{V_P}{0.035} < 47k\Omega$ *R_L = Pull-up Resistor (Ohms), V_P = Pull-up Voltage (VCD)*



Outputs and Fault

Refer to *Chapter 3, Installation*, for more information on the inputs, outputs, and fault output.

GND

The GND terminal is the ground reference for the general-purpose outputs, the fault output, the Rx input, and the Tx output.

+5V

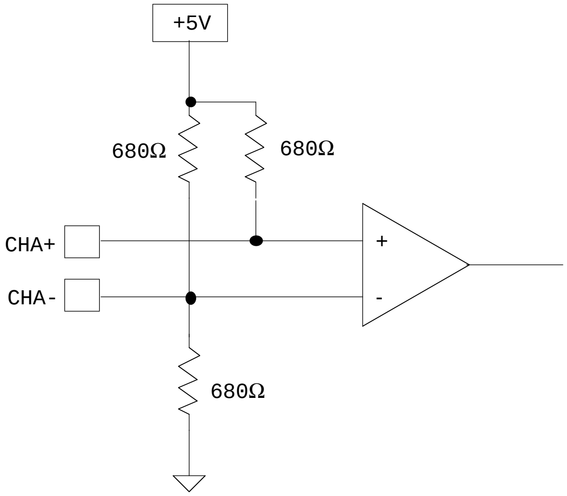
The +5V terminal is a +5VDC internal supply designed to provide +5VDC power at a maximum of 250 mA to run an optical encoder. The +5V supply may be used to power the I/O only if an encoder is not used.

Encoder Inputs CHA+, CHB+, and CHZ+

The plus inputs to the encoder are pulled up to +5 VDC with a 680 resistor . The input driver connected to this input must be capable of sinking 6.3 mA of current (minimum).

Encoder Inputs CHA-, CHB-, and CHZ-

The minus inputs to the encoder are biased at +2.5VDC with a 680 pull-up and a 680 pull-down. The input driver connected to this input must be capable of sinking and sourcing 6.3 mA of current minimum.



Incremental Encoder Schematic

The maximum encoder input frequency is 160 KHz (pre-quadrature) with a minimum pulse width of 500 nsec.

ACC

Reserved for expansion of Compumotor product features.

OP1-HV and OP2-HV

Formerly, OP1-HV and OP2-HV were labelled RSV+ and RSV-. If your unit is marked RSV+/RSV- these terminals have no function. If labeled OP1-HV/OP2-HV, these terminals can be used in place of OPT01/OPT02 when using a +12-24VDC power supply to pull up the I/O. They should not be used at the same time as OPT01/OPT02, power supply damage could occur.

Motor Electrical Specifications

Minimum Motor Winding Inductance

0.5 mH—Compumotor strongly recommends 2mH measured in series or parallel.

Maximum Motor Winding Inductance

None—Compumotor recommends 50mH measured in series or parallel. Use of motors with a winding inductance greater than 50mH may result in a significant reduction in system performance.

Minimum Motor Hipot

500VDC

Operational Specifications

Accuracy

±5 arcminutes typical (unloaded, bidirectional) with Compumotor motors.

Repeatability

±5 arcseconds typical (unloaded, unidirectional).

Hysteresis

Less than 2 arcminutes (0.0334°) unloaded, bidirectional.

Rotor Inertia

Size	**Rotor Inertia oz-in^2**	**Rotor Inertia (Kg-m^{2}X10^{-6})**
S57-51	0.546	9.998
S57-83	1.1	20.1
S57-102	1.69	30.9
Size 34	**Rotor Inertia oz-in^2**	**Rotor Inertia (Kg-m^{2}X10^{-6})**
S83-62	3.47	63.4
S83-93	6.76	124.0
S83-135	10.47	191.0
Size 42	**Rotor Inertia oz-in^2**	**Rotor Inertia Kg-cm^2**
SX106-178	44.0	8.05
SX106-205	52.0	9.51
SX106-250	63.0	12.14

Rotor Inertia (Compumotor Motors)

Motor Current & Torque

Speed/torque curves for the SX are provided later in this chapter.

Motor Size	Current	Static Torque (in-oz)
S57-51 **S**	1.18	65
S57-51 **P**	2.28	65
S57-83 **S**	1.52	100
S57-83 **P**	3.09	100
S57-102 **S**	1.71	125
S57-102 **P**	3.47	125
S83-62 **S**	2.19	160
S83-62 **P**	4.42	160
S83-93 **S**	2.85	300
S83-93 **P**	5.62	300
S83-135 **S**	3.47	400
S83-135 **P**	6.00	343

S: Series Configuration P: Parallel Configuration

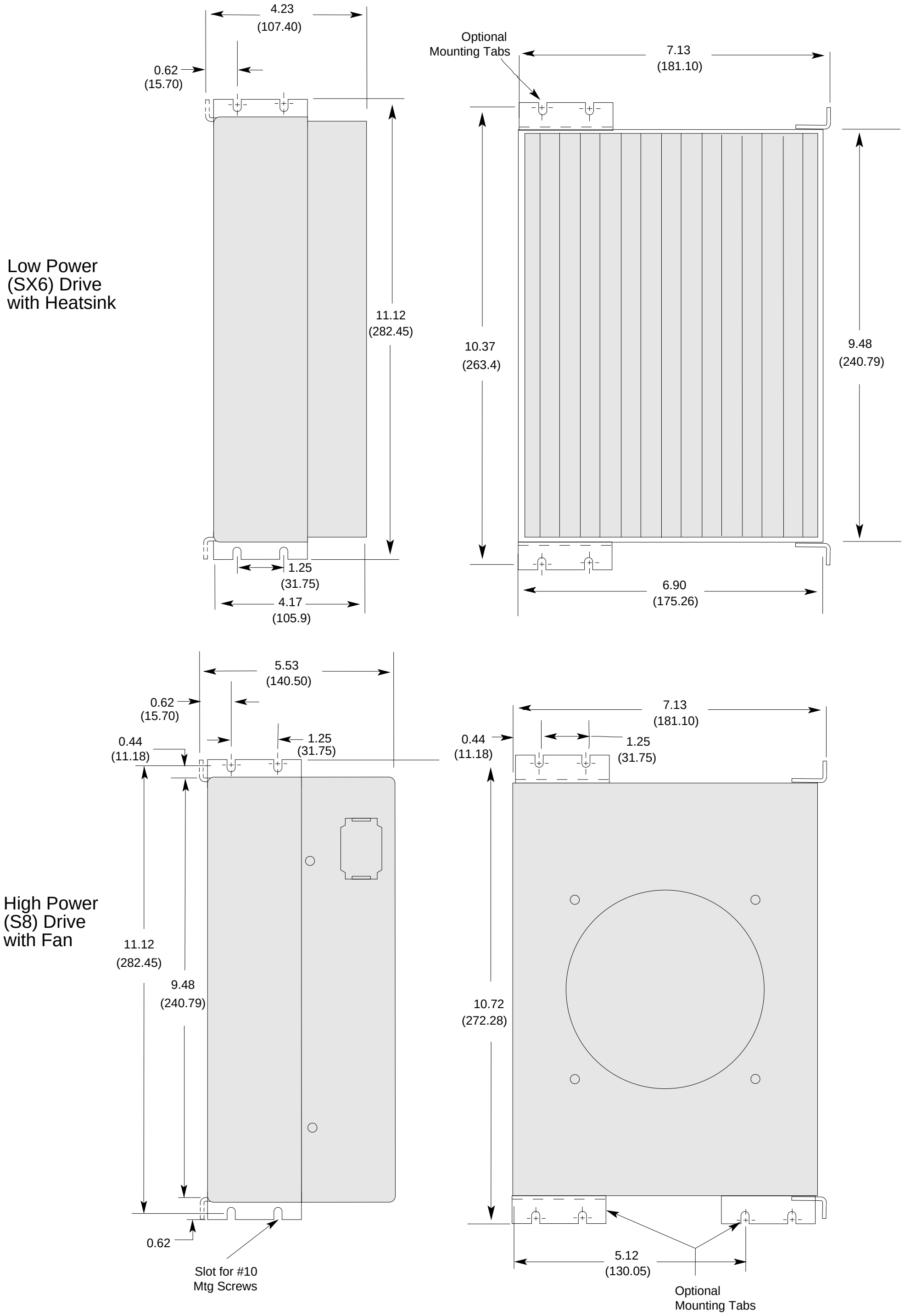
Motor Specifications (57 & 83 Series Motors)

Motor Size	Current	Static Torque (in-oz)
SX106-178 **S**	6.02	1000
SX106-178 **P**	8.0	667
SX106-205 **S**	3.55	1900
SX106-205 **P**	6.99	1900
SX106-250 **S**	6.23	1450
SX106-250 **P**	8.0	967

S: Series Configuration P: Parallel Configuration

Motor Specifications (106 Series Motors)

Drive Dimensions



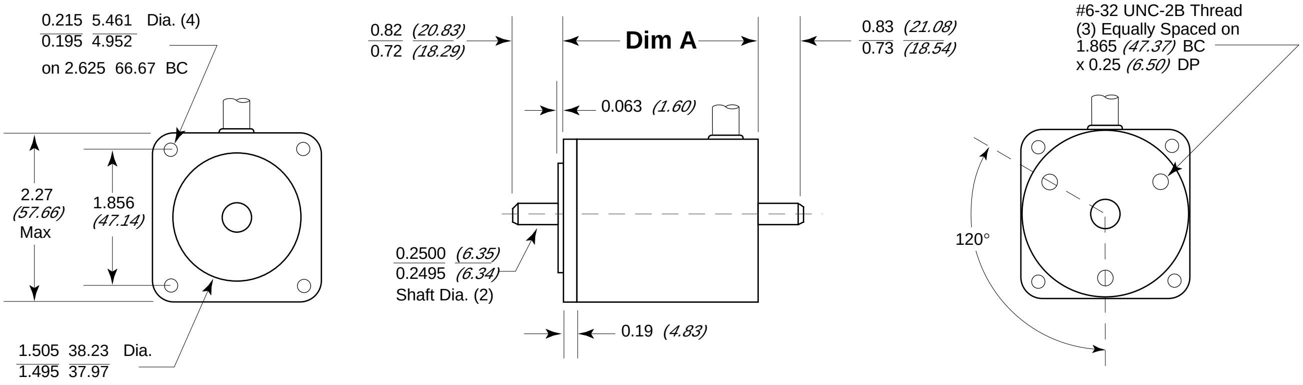
SX Dimensions

The fan kit is optional with the low-power version of the SX Drive.

Motor Dimensions

Size 23 frame

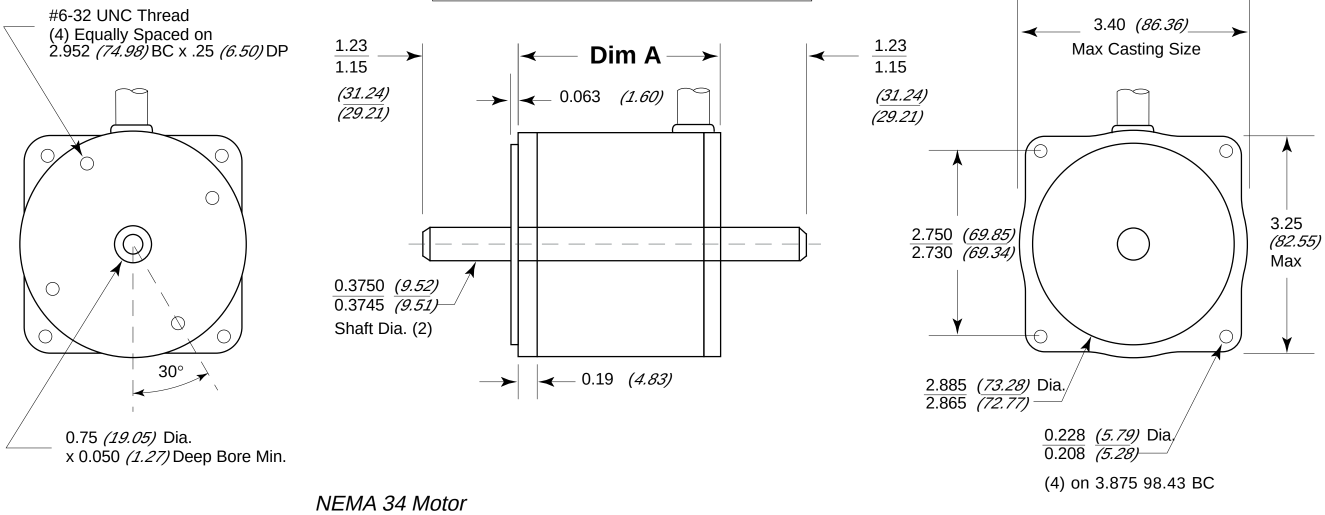
Model #	Dim A	
SX57-51-MO	2.0	50.23
SX57-83-MO	3.1	75.23
SX57-102-MO	4.0	101.6



NEMA 23 Motor

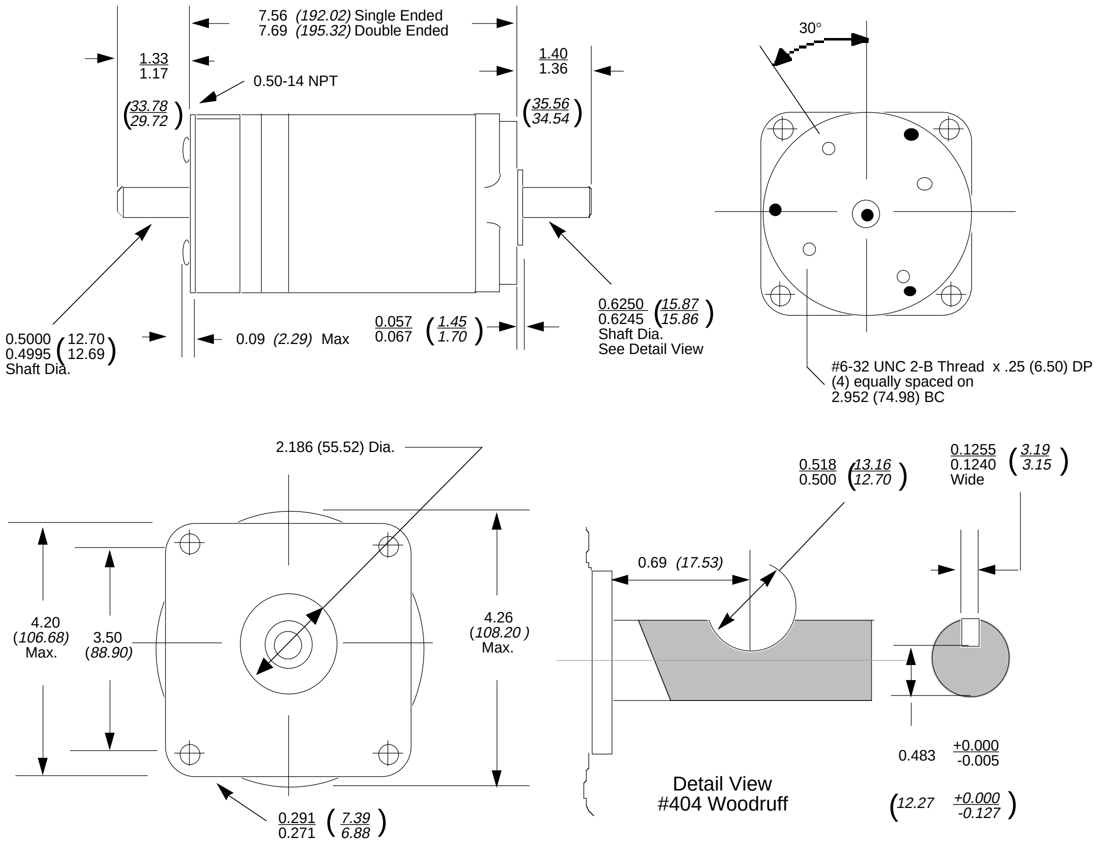
Size 34 frame

Model #	Dim A	
SX83-62-MO	2.5	62.0
SX83-93-MO	3.7	93.98
SX83-135-MO	5.2	129.0

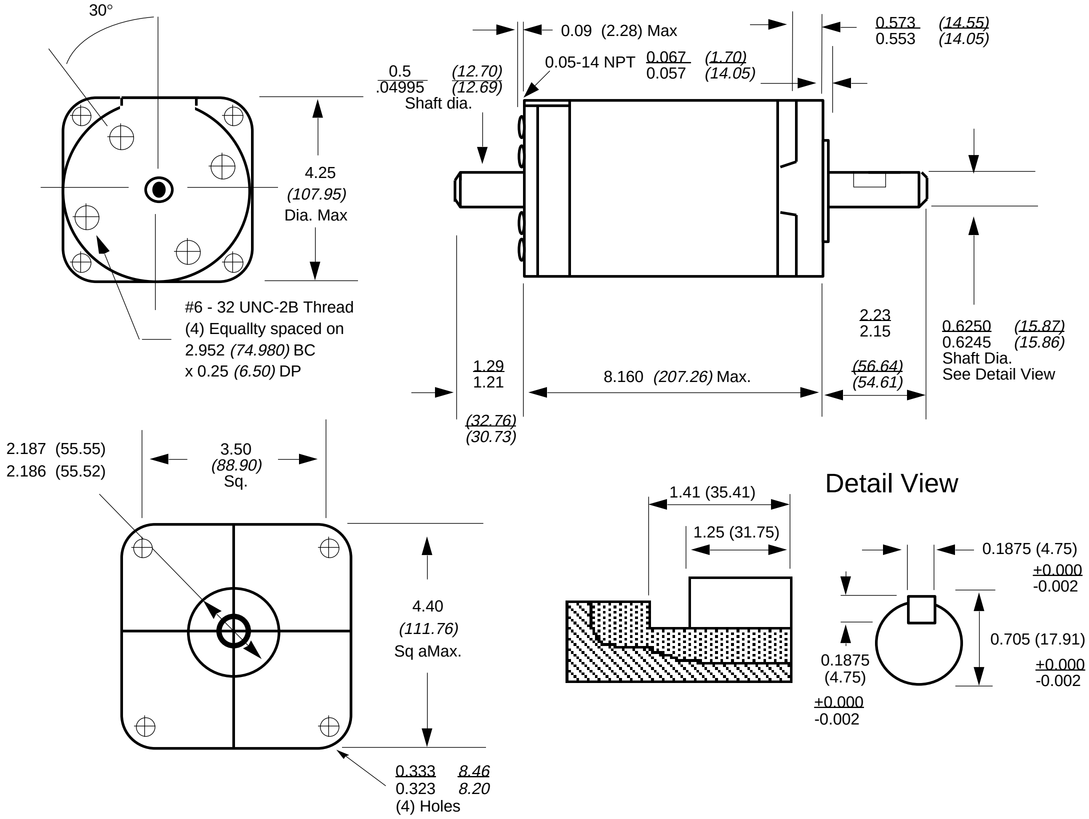


NEMA 34 Motor

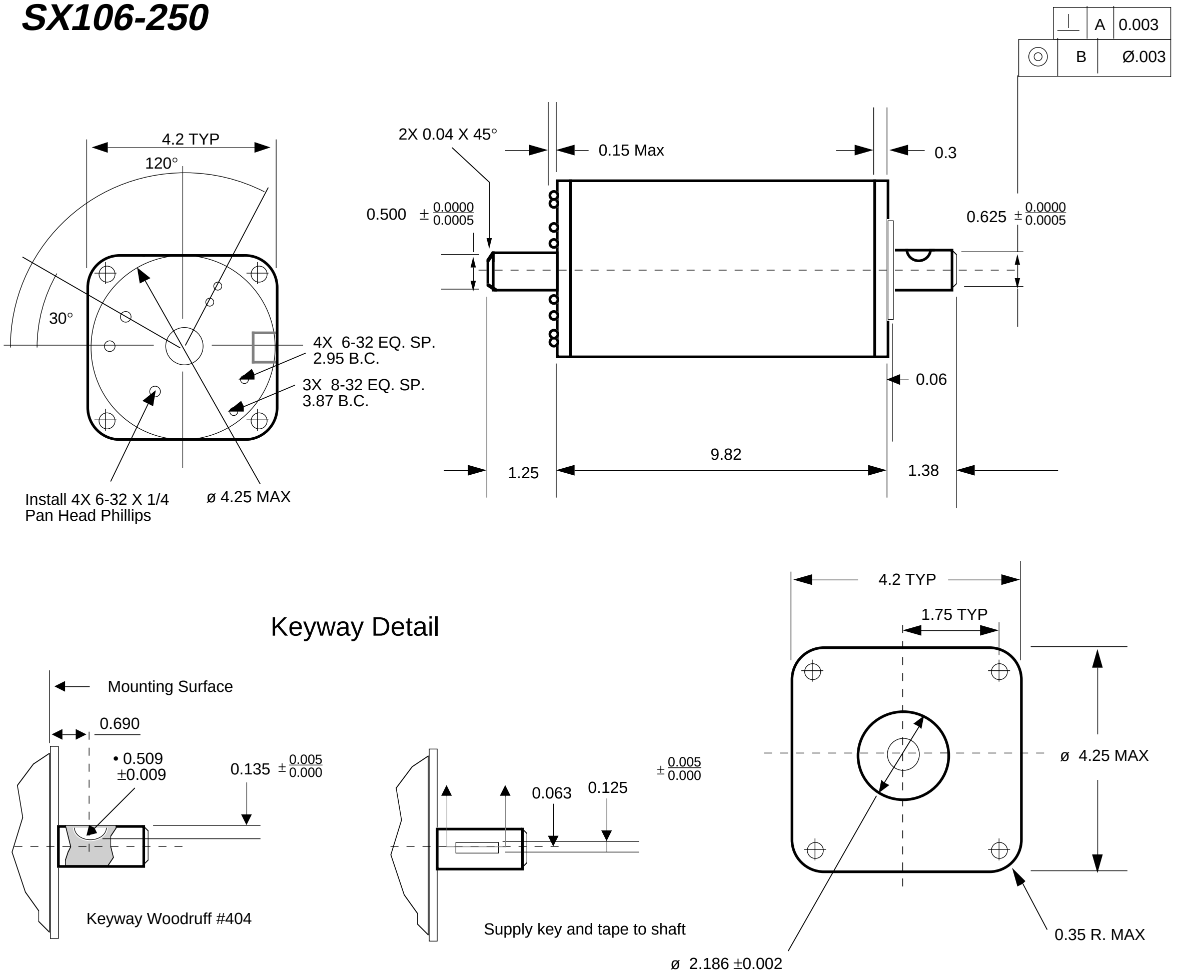
SX106-178



SX106-205



SX106-250



DIP Switch Summary

The SX has two sets of DIP switches (refer to *Chapter 2, Getting Started*). Each set of DIP switches has eight individual switches. The first set of switches is referred to as **SW1** and the second set as **SW2**. The individual switch will be preceded by the **#** symbol. Hence, the third switch on **SW1** is referred to as **SW1-#3**, while the third switch on **SW2** is referred to as **SW2-#3.**

Switch #	Function	Default
SW1-#1	Current - most significant bit	*off
SW1-#2	Current	*off
SW1-#3	Current	*off
SW1-#4	Current	*off
SW1-#5	Current	*off
SW1-#6	Current - least significant bit	*off
SW1-#7	No function	*off
SW1-#8	No function	*off
SW2-#1	Address - least significant bit	*off
SW2-#2	Address	*off
SW2-#3	Address	*off
SW2-#4	Address - most significant bit	*off
SW2-#5	Baud Rate - least significant bit	*off
SW2-#6	Baud Rate	*off
SW2-#7	Baud Rate - most significant bit	*off
SW2-#8	Auto Test *off	

DIP Switch Summary

Motor Current

Motor Size	Current	SW1-#1	SW1-#2	SW1-#3	SW1-#4	SW1-#5	SW1-#6
S57-51**S**	1.18	off	off	on	on	off	off
S57-51**P**	2.28	off	on	on	off	off	off
S57-83**S**	1.52	off	on	off	off	off	off
S57-83**P**	3.09	on	off	off	off	off	off
S57-102**S**	1.71	off	on	off	off	on	off
S57-102**P**	3.47	on	off	off	on	off	off
S83-62**S**	2.19	off	on	off	on	on	on
S83-62**P**	4.42	on	off	on	on	on	off
S83-93**S**	2.85	off	on	on	on	on	off
S83-93**P**	5.62	on	on	on	off	on	on
S83-135**S**	3.47	on	off	off	on	off	off
S83-135**P**	6.00	on	on	on	on	on	on

S: Series Configuration P: Parallel Configuration

SX6 Drive Motor Current Settings (Compumotor Motors)

Motor Size	Current	SW1-#1	SW1-#2	SW1-#3	SW1-#4	SW1-#5	SW1-#6
S106-178**S**	6.02	on	off	on	on	on	on
S106-178**P**	8.0	on	on	on	on	on	on
S106-205**S**	3.55	off	on	on	on	off	off
S106-205**P**	6.99	on	on	off	on	on	on
S106-250**S**	6.23	on	on	off	off	off	on
S106-250**P**	8.0	on	on	on	on	on	on

S: Series Configuration P: Parallel Configuration

SX8 Drive Motor Current Settings (Compumotor Motors)

Low-Power SX6 and High Power SX8 Drives

Current	SW1 #1	SW1 #2	SW1 #3	SW1 #4	SW1 #5	SW1 #6	Current	SW1 #1	SW1 #2	SW1 #3	SW1 #4	SW1 #5	SW1 #6
0.04	off	off	off	off	off	off	**3.09**	on	off	off	off	off	off
0.13	off	off	off	off	off	on	**3.19**	on	off	off	off	off	on
0.23	off	off	off	off	on	off	**3.28**	on	off	off	off	on	off
0.32	off	off	off	off	on	on	**3.38**	on	off	off	off	on	on
0.42	off	off	off	on	off	off	**3.47**	on	off	off	on	off	off
0.51	off	off	off	on	off	on	**3.57**	on	off	off	on	off	on
0.61	off	off	off	on	on	off	**3.66**	on	off	off	on	on	off
0.70	off	off	off	on	on	on	**3.76**	on	off	off	on	on	on
0.80	off	off	on	off	off	off	**3.85**	on	off	on	off	off	off
0.89	off	off	on	off	off	on	**3.95**	on	off	on	off	off	on
0.99	off	off	on	off	on	off	**4.04**	on	off	on	off	on	off
1.08	off	off	on	off	on	on	**4.14**	on	off	on	off	on	on
1.18	off	off	on	on	off	off	**4.23**	on	off	on	on	off	off
1.27	off	off	on	on	off	on	**4.33**	on	off	on	on	off	on
1.37	off	off	on	on	on	off	**4.42**	on	off	on	on	on	off
1.46	off	off	on	on	on	on	**4.51**	on	off	on	on	on	on
1.52	off	on	off	off	off	off	**4.58**	on	on	off	off	off	off
1.62	off	on	off	off	off	on	**4.68**	on	on	off	off	off	on
1.71	off	on	off	off	on	off	**4.77**	on	on	off	off	on	off
1.81	off	on	off	off	on	on	**4.86**	on	on	off	off	on	on
1.90	off	on	off	on	off	off	**4.96**	on	on	off	on	off	off
2.00	off	on	off	on	off	on	**5.05**	on	on	off	on	off	on
2.09	off	on	off	on	on	off	**5.15**	on	on	off	on	on	off
2.19	off	on	off	on	on	on	**5.24**	on	on	off	on	on	on
2.28	off	on	on	off	off	off	**5.34**	on	on	on	off	off	off
2.38	off	on	on	off	off	on	**5.43**	on	on	on	off	off	on
2.47	off	on	on	off	on	off	**5.53**	on	on	on	off	on	off
2.57	off	on	on	off	on	on	**5.62**	on	on	on	off	on	on
2.66	off	on	on	on	off	off	**5.72**	on	on	on	on	off	off
2.76	off	on	on	on	off	on	**5.81**	on	on	on	on	off	on
2.85	off	on	on	on	on	off	**5.91**	on	on	on	on	on	off
2.95	off	on	on	on	on	on	**6.00**	on	on	on	on	on	on

Setting SX6 Drive Motor Current (Non-Compumotor Motors)

Current	SW1 #1	SW1 #2	SW1 #3	SW1 #4	SW1 #5	SW1 #6	Current	SW1 #1	SW1 #2	SW1 #3	SW1 #4	SW1 #5	SW1 #6
0.05	off	off	off	off	off	off	**4.12**	on	off	off	off	off	off
0.18	off	off	off	off	off	on	**4.25**	on	off	off	off	off	on
0.30	off	off	off	off	on	off	**4.38**	on	off	off	off	on	off
0.43	off	off	off	off	on	on	**4.50**	on	off	off	off	on	on
0.56	off	off	off	on	off	off	**4.63**	on	off	off	on	off	off
0.69	off	off	off	on	off	on	**4.75**	on	off	off	on	off	on
0.81	off	off	off	on	on	off	**4.89**	on	off	off	on	on	off
0.93	off	off	off	on	on	on	**5.01**	on	off	off	on	on	on
1.06	off	off	on	off	off	off	**5.14**	on	off	on	off	off	off
1.19	off	off	on	off	off	on	**5.26**	on	off	on	off	off	on
1.31	off	off	on	off	on	off	**5.39**	on	off	on	off	on	off
1.44	off	off	on	off	on	on	**5.51**	on	off	on	off	on	on
1.59	off	off	on	on	off	off	**5.64**	on	off	on	on	off	off
1.69	off	off	on	on	off	on	**5.77**	on	off	on	on	off	on
1.82	off	off	on	on	on	off	**5.90**	on	off	on	on	on	off
1.94	off	off	on	on	on	on	**6.02**	on	off	on	on	on	on
2.03	off	on	off	off	off	off	**6.11**	on	on	off	off	off	off
2.16	off	on	off	off	off	on	**6.23**	on	on	off	off	off	on
2.28	off	on	off	off	on	off	**6.36**	on	on	off	off	on	off
2.41	off	on	off	off	on	on	**6.48**	on	on	off	off	on	on
2.54	off	on	off	on	off	off	**6.61**	on	on	off	on	off	off
2.66	off	on	off	on	off	on	**6.73**	on	on	off	on	off	on
2.79	off	on	off	on	on	off	**6.87**	on	on	off	on	on	off
2.91	off	on	off	on	on	on	**6.99**	on	on	off	on	on	on
3.04	off	on	on	off	off	off	**7.12**	on	on	on	off	off	off
3.17	off	on	on	off	off	on	**7.24**	on	on	on	off	off	on
3.297	off	on	on	off	on	off	**7.37**	on	on	on	off	on	off
3.42	off	on	on	off	on	on	**7.49**	on	on	on	off	on	on
3.55	off	on	on	on	off	off	**7.62**	on	on	on	on	off	off
3.67	off	on	on	on	off	on	**7.75**	on	on	on	on	off	on
3.80	off	on	on	on	on	off	**7.87**	on	on	on	on	on	off
3.93	off	on	on	on	on	on	**8.00**	on	on	on	on	on	on

Setting SX8 Drive Motor Current (Non-Compumotor Motors)

Address Settings

Address	SW2-1	SW2-2	SW2-3	SW2-4
***** 1	off	off	off	off
2	on	off	off	off
3	off	on	off	off
4	on	on	off	off
5	off	off	on	off
6	on	off	on	off
7	off	on	on	off
8	on	on	on	off
9	off	off	off	on
10	on	off	off	on
11	off	on	off	on
12	on	on	off	on
13	off	off	on	on
14	on	off	on	on
15	off	on	on	on
16	on	on	on	on

* Default Setting

Address Settings

RS-232C Baud Rate

Baud Rate	SW2-5	SW2-6	SW2-7
***** 9600	off	off	off
Reserved	on	off	off
9600	off	on	off
4800	on	on	off
2400	off	off	on
1200	on	off	on
600	off	on	on
300	on	on	on

* Default Setting

RS-232C Baud Rate

Automatic Test

*** SW2-#8 OFF Disables Auto Test**
SW2-#8 ON Enables Auto Test

* Default Setting

Non-Compumotor—Drive/Motor Connection

Compumotor does not recommend that you use non-Compumotor motors with the SX. If you do use a non-Compumotor motor, it must meet the following requirements:

① A minimum inductance of 0.5 mH, series or parallel, may be used (Compumotor strongly recommends a minimum inductance of 2 mH).

② A minimum of 500VDC high-pot insulation rating from phase-to-phase and phase-to-ground.

③ The motor must not have riveted rotors or stators.

④ Do not use solid rotor motors.

⑤ Test all motors carefully. Verify that the motor temperature in your application is within the system limitations. *The motor manufacturer's maximum allowable motor case temperature must not be exceeded*. You should test the motor over a 2- to 3-hour period. Motors tend to have a long thermal time constant, but can still overheat, which results in motor damage.

CAUTION

Consult a Compumotor Applications Engineer if you have any questions regarding the use of a non-Compumotor motor.

Wiring Configurations

You can determine the motor's wiring configuration by referencing the manufacturer's motor specification document supplied with the motor. You can also determine the wiring configuration with an ohmmeter using the procedures below (*4-Lead Motor, 6-Lead Motor, 8 Lead Motor*). Once you determine the correct motor wiring configuration, use the terminal connection diagram that applies to your configuration (refer to the following figure).

4-Lead Motor

① Label one motor lead **A+**.

② Connect one lead of an ohmmeter to the **A+** lead and touch the other lead of the ohmmeter to the three remaining motor leads until you find the lead that creates continuity. Label this lead **A-**.

③ Label the two remaining leads **B+** and **B-**. *Verify that there is continuity between the B+ and B- leads.*

④ Proceed to the *Terminal Connections* section.

6-Lead Motor

① Determine, with an ohmmeter, which three of the six motor leads are common (one phase).

② Label each one of these three motor leads **A**.

③ Using the ohmmeter, verify that the remaining three leads are common.

④ Label the other three leads **B**.

⑤ Set the ohmmeter range to approximately the 100 ohm scale.

⑥ Connect the ohmmeter's negative lead to one of the motor leads labeled **A**. Alternately measure the resistance to the two remaining motor leads also labeled **A**. The resistance measurements will reflect one of the following scenarios.

Scenario #1

The resistance measurements to the two remaining motor leads are virtually identical. Label the two remaining motor leads **A+** and **A-**. Label the motor lead connected to the negative lead of the ohmmeter **A-CT** (this is the center tap lead for Phase A of the motor).

Scenario #2

The resistance measurement to the second of the three motor leads measures 50% of the resistance measurement to the third of the three motor leads. Label the second motor lead **A-CT** (this is the center tap lead for Phase A of the motor). Label the third motor lead **A-**. Label the motor lead connected to the ohmmeter **A+**.

⑦ Repeat the procedure as outlined in step 6 for the three leads labeled **B** (**B-CT** is the center tap lead for Phase B of the motor).

⑧ Connect the **A-CT** motor lead to the **A-CT** pin on the **MOTOR** connector. Connect the **B-CT** motor lead to the **B-CT** pin on the **MOTOR** connector.

⑨ Proceed to the *Terminal Connections* section.

8-Lead Motor

Because of the complexity involved in phasing an 8-lead motor, you must refer to the manufacturer's motor specification document. You can configure the 8-lead motor in parallel or series. Using the manufacturer's specifications, label the motor leads, as shown in chapter 3, on page 12.

Parallel Configuration

Use the following procedures for parallel configurations.

① Connect motor leads A1 & A3 together and relabel this common point **A+**.

② Connect motor leads A2 & A4 together and relabel this common point **A-**.

③ Connect motor leads B1 & B3 together and relabel this common point **B+**.

④ Connect motor leads B2 & B4 together and relabel this common point **B-**.

Series Configuration

Use the following procedures for series configurations.

① Connect A2 & A3 to **A-CT**. You may also connect B2 & B3 to **B-CT**.

② Relabel the A1 lead to **A+**.

③ Relabel the A4 lead to **A-**.

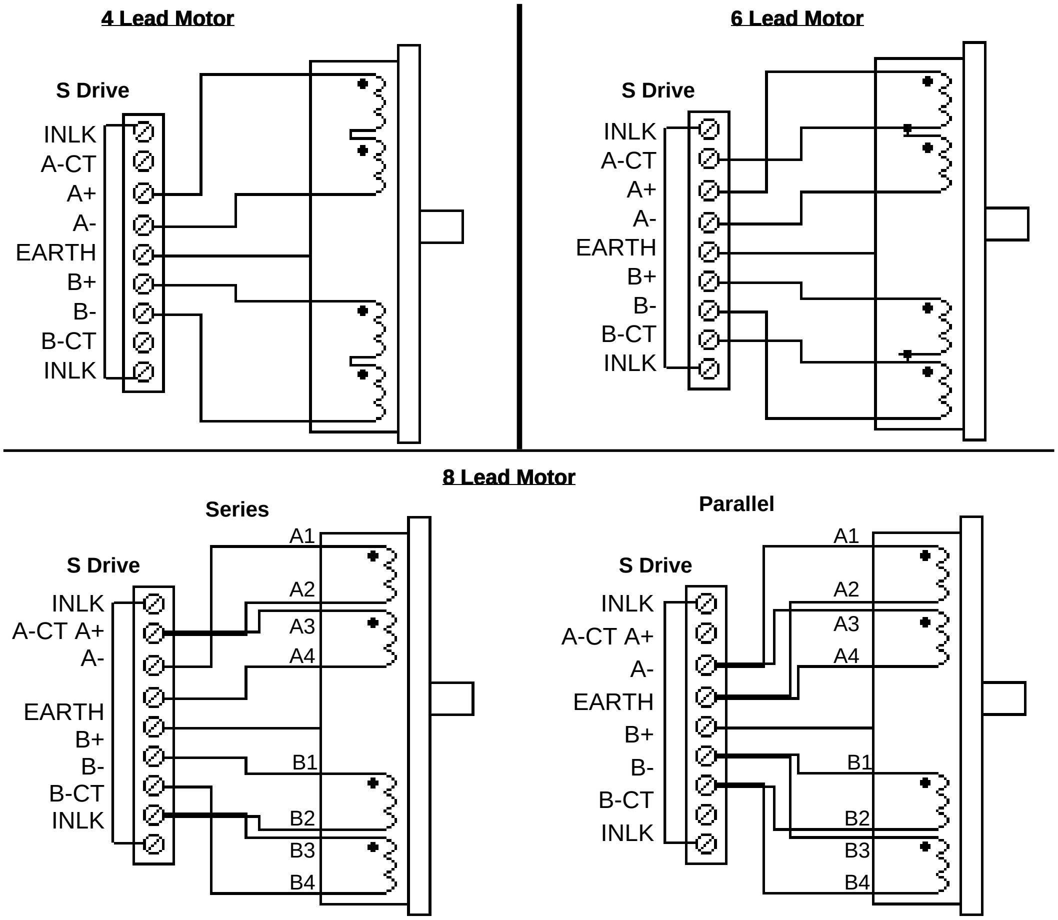
④ Relabel the B1 lead to **B+**.

⑤ Relabel the B4 lead to **B-**.

⑥ Proceed to the Terminal Connections section below.

Terminal Connections

After you determine the motor's wiring configuration, connect the motor leads to the 9-pin **MOTOR** connector according to the figure below.



9-Pin Motor Connector (Non-Compumotor Motors)

CAUTION

Do not connect or disconnect the motor with the power on. This will damage the contacts of the motor connector and may cause personal injury.

Extended Motor Cables

The following table contains the recommended motor cables for various motor types and the minimum recommended motor/driver wire size (AWG).

Maximum Current Per Winding (Amps)	Less than100 ft. (20.5M)	100 - 200 ft.(30.5M - 71M)
3	22 AWG	20 AWG
6	20 AWG	18 AWG
8	16 AWG	14 AWG

Recommended Motor Cables

Cable runs of more than 200 feet (71M) are not recommended. Cable runs greater than 50 feet may degrade system performance.

Non-Compumotor Motors—Setting Motor Current

Compumotor does not recommend that you use non-Compumotor motors with the SX. If you do, refer to the formulas below that correspond to your motor (4-lead, 6-lead, or 8-lead) and use the previous tables titled *Setting SX6 Drive Motor Current* and *Setting SX8 Drive Motor Current* to set the motor's current. **Never increase current more than 10% above the specified rating**.

4-Lead Motors

If you use a 4-lead motor, the manufacturer's current setting will translate directly to the values shown in the previous tables titled *Setting SX6 Drive Motor Current* and *Setting SX8 Drive Motor Current*.

6-Lead Motors

If you use a 6-lead motor, and the manufacturer specifies the motor current as a unipolar rating, you must use the following formula to convert the unipolar current rating to the correct bipolar rating.

Unipolar Current X .707 = Bipolar Current

After you make the conversion, use the Motor Current tables to set the motor current. If the manufacturer specifies the motor current as a bipolar rating, you can use the Motor Current tables directly (no conversion) to set motor current.

8-Lead Motors

If you are using an 8-lead motor, manufacturers generally rate the motor current in one of two ways:

❑ If the motor current is listed as a unipolar rating, use the following formula to convert the unipolar current rating to the correct bipolar current rating.

Unipolar Current X .707 = Bipolar Series Current

If you are wiring the motor in ***series***, use the table *Setting SX6 Drive Motor Current (Non-Compumotor motors)*, previously in this chapter, the converted value to set the motor current.

If you wire the motor in ***parallel***, you must **double** the converted value and use the tables titled *Setting SX6 and SX8 Drive Motor Current (Non-Compumotor motors)*, previously in this chapter, to set the motor current.

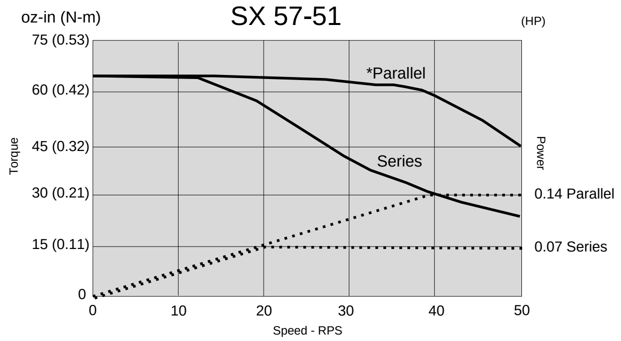
❑ If the motor current is listed as a bipolar series rating, you can wire the motor in ***series*** and use the tables titled *Setting SX6 Drive Motor Current (Non-Compumotor motors)*, previously in this chapter, and *Setting SX8 Drive Motor Current (Non-Compumotor motors)*, previously in this chapter, directly (no conversion) to set motor current.

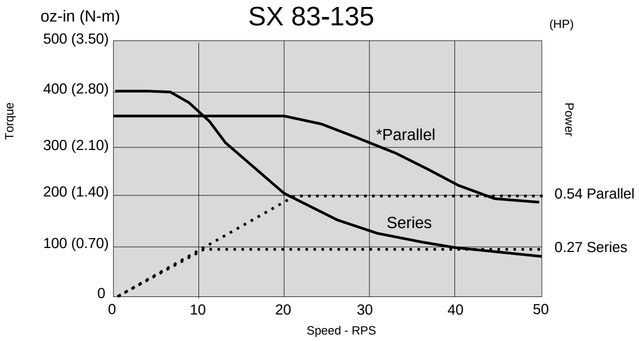
❑ If the motor current is listed as a bipolar series rating and you wire the motor in ***parallel***, you must **double** the manufacturer's rating and then use table *Setting SX6 Drive Motor Current (Non-Compumotor motors)*, previously in this chapter, to set the motor current.

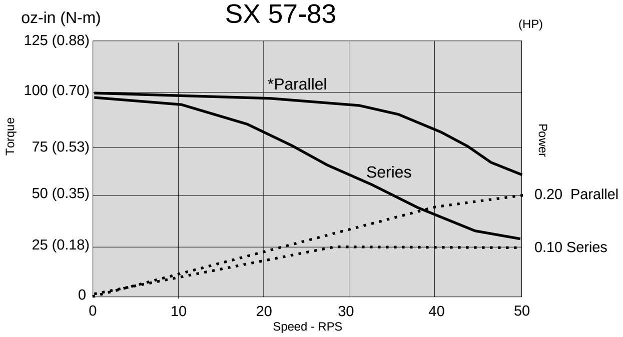
If you have any questions with regard to the configurations, please call Compumotor's Applications Engineering Department at 800-358-9070.

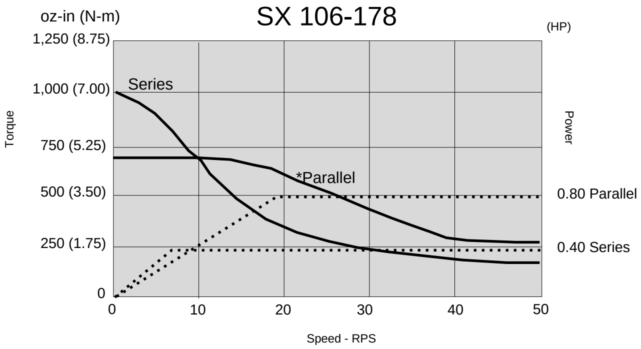
Motor Performance Specifications

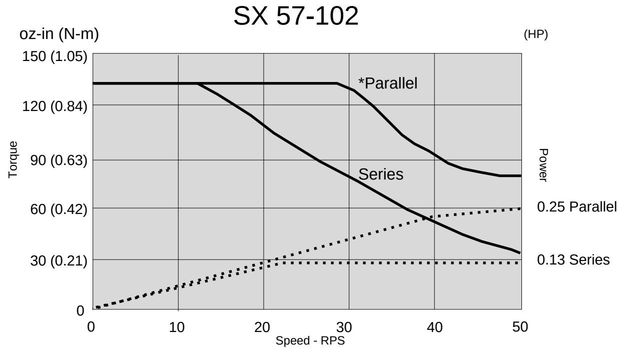
SX Series motors are designed to allow you to change the motor winding configuration easily. The performance curves shown below indicate that different levels of performance can be obtained by connecting the step motor windings in series or in parallel. You must exercise caution when you run motors in a parallel configuration. ***Sustained operation at high speeds may cause the motor to overheat due to electrical pole heating.***

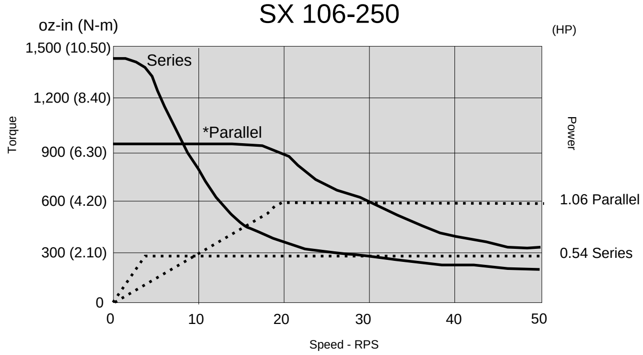


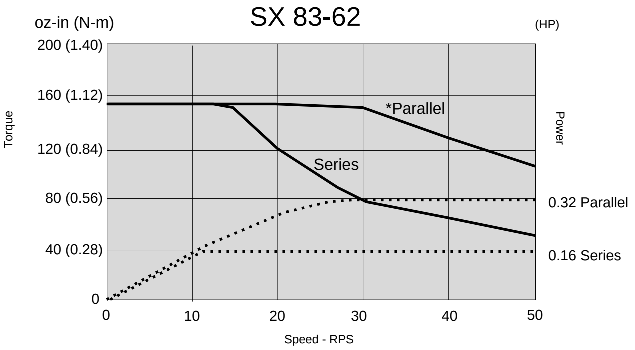


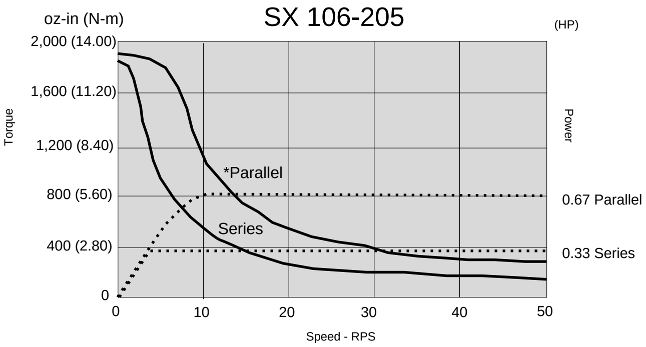


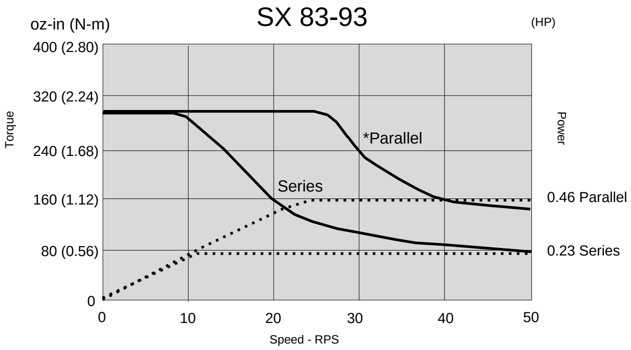












*Parallel connected motors are limited to 50% duty cycle when operated above 5 rps. For greater than 50% duty cycle above 5 rps, you must connect the motor in series. Fan cooling the motor will increase duty cycles above 5 rps.

CHAPTER ⑦

Maintenance & Troubleshooting

Chapter Objectives

The information in this chapter will enable you to:

- ❑ Maintain the system’s components to ensure smooth, efficient operation
- ❑ Isolate and resolve system hardware problems
- ❑ Isolate and resolve system software problems

Maintenance

The following items are included with the SX Indexer/Drive.

Part	Part Number
13-Pin Phoenix Connector	43-011796-01
7-Pin for SX8	43-013575-01
8-Pin Phoenix (3)	43-007483-01
AC Power Cord	44-000054-01
Battery	47-011743-01
Mounting Bracket	53-006007-01

Spare Parts List

Part	Part Number
ESD Service Strap	58-011184-01

Optional Equipment List

Battery Maintenance

The battery provided with the SX is a lithium battery which should last approximately 4-5 years. If a battery failure occurs, the battery may be replaced with a commonly available 3V, 500MAH, 24.5 x 5 mm lithium battery. Ensure that the battery clip maintains good contact with the battery after replacement. Some manufacturers and part numbers are:

Duracell - DL2450

Sanyo - CR2450

Drive Maintenance

Ensure that the SX’s heatsink is free of particles and has a free flow of air over its entire surface. Enclosures must be connected to earth ground through a grounding electrode conductor to provide a low-impedance path for ground-fault or noise-induced currents. All earth ground connections must be continuous and permanent.

Motor Maintenance

You should inspect all mechanical parts of the motor regularly to ensure that no bolts or couplings have become loose during normal operation. This will prevent some minor problems from developing into more serious problems.

You should inspect the motor cable periodically for signs of wear. This inspection interval is duty-cycle, environment, and travel-length dependent. The cable should not have excessive force applied to it and should not be bent beyond a one-inch radius of curvature during normal operation. Tighten all cable connectors.

Reducing Electrical Noise

For detailed information on reducing electrical noise in your system, refer to the current Compumotor Catalog.

Problem Isolation

When your system does not function properly (or as you expect it to operate), the first thing that you must do is identify and isolate the problem. When you accomplish this, you can effectively begin to resolve and eradicate the problem.

The first step is to isolate each system component and ensure that each component functions properly when it is run independently. You may have to dismantle your system and put it back together piece by piece to detect the problem. If you have additional units available, you may want to use them to replace existing components in your system to help identify the source of the problem.

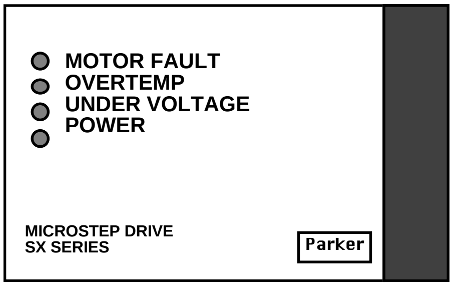
Determine if the problem is mechanical, electrical, or software-related. Can you repeat or re-create the problem? Do not make quick rationalizations about the problems. Random events may appear to be related, but they may not be contributing factors to your problem. Carefully investigate and decipher the events that occur before the subsequent system problem.

You may be experiencing more than one problem. You must solve one problem at a time. Document all testing and problem isolation procedures. You may need to review and consult these notes later. This will also prevent you from duplicating your testing efforts.

Once you have isolated the problem, take the necessary steps to resolve it. Refer to the problem solutions contained in this chapter. If your system's problem persists, contact Parker Compumotor at 800-358-9070.

Front Panel LEDs

There are four LEDs on the front panel of the SX (refer to the following figure).



Bottom of SX Drive Front Panel

❑ Motor short-circuit

❑ The interlock is broken (opened)

The **OVERTEMP** LED is **red** and turns on when the internal drive temperature exceeds 70°C.

The **UNDERVOLTAGE** LED is **red** and turns on when AC line voltage is below 85VAC.

The **POWER** LED is **green** and turns on when the internal bias supply is operating and providing +5V.

If all LED's are on, the board monitor alarm will be activated.

Common Problems and Solutions

The following table contains common problems, probable causes, and solutions to the problems. It should help you eradicate most problems you might encounter.

Symptoms	Probable Causes	Solutions
The power LED is not on (illuminated).	• The drive is not receiving AC voltage.	• Verify that the connector on the SX is fully seated • Verify that there is AC voltage at the AC outlet that the drive is plugged into • Verify that there is AC voltage at the drive at the AC power connector
The power LED is flashing.	• AC Line voltage is too low.	• Check AC line voltage (90VAC minimum).
	• There is insufficient load regulation on the AC line.	• Increase the AC line wire size. Increase the isolation transformer size (if used).
Little or no holding torque. Power LED is on, motor fault LED is off.	• The motor current is set too low.	• Check the current select switches and verify that the current is set correctly.
	• The motor winding or cable is open.	• Check motor and cable with an ohmmeter.
	• The Auto Standby function is enabled.	• Disable Auto Standby function if this function does not allow enough holding torque.
	• Amplifier shutdown is enabled.	• Use the RSE command to inform you if the drive has been disabled and the reason.
The motor fault LED is on.	• The motor cable is disconnected or not fully seated at the drive.	• Check the motor cable.
	• The motor connector interlock jumper is missing or is disconnected.	• Check the interlock jumper.
	• The SX has detected a motor/wiring short circuit.	• Check the motor and cable wiring.
Overtemperature LED is on.	• The internal drive temperature is greater than 70 C.	• Remove fin cooling obstructions or add fan cooling to the SX (Compumotor offers a fan kit).
The undervoltage LED is on.	• AC line voltage is less than 85VAC.	• Provide a minimum of 90VAC underload.
Motor fault, overtemperature, and undervoltage LEDs are on.	• The internal Indexer monitor has reset the Indexer.	• Excessive electrical noise. Verify the drive case is properly grounded.
The motor moves erratically at low speeds.	• Motor current is set incorrectly.	• Check the current select switches and verify that the current is set correctly.
	• One motor phase open.	• With motor connector removed from the SX, use an ohmmeter to measure continuity of motor windings.
	• Motor resolution is set for 200 or 400 steps per revolution.	• Full and half step modes will cause the motor to run roughly at low speeds.
The motor stalls at high speeds.	• Motor current is not set correctly.	• Check DIP switches verify that motor current is set correctly.
	• Motor is undersized for the application.	• Verify that the motor is sized correctly.

Symptoms	Probable Causes	Solutions
The motor stalls during acceleration.	• Motor current is not set correctly.	• Check the current select switches and verify that the current is set correctly.
	• The acceleration is set too high.	• Decrease the acceleration.
	• There is insufficient rotor inertia.	• Add inertia to the motor shaft.
	• Motor is undersized for the application.	• Verify that the motor is sized correctly.
The motor (unloaded) stalls at nominal speed.	• There is insufficient rotor inertia.	• Ad inertia to the motor shaft.
The motor does not move the commanded distance.	• The motor resolution is set incorrectly.	• Ensure that the SX's indexer and drive resolution settings are the same.
The drive moves the motor in the wrong direction.	• The motor is not wired to the drive properly.	• Verify motor connections. Swapping motor leads A+ and A- at the drive connector to change direction if necessary.
A SX move is commanded and no motion occurs.	• Following mode may be enabled when you are trying to make a move as an Indexer.	• Check your configuration and motion settings.
	• A limit may be enabled and active.	
	• You may be in Absolute mode and are already at the position you are commanding the motor to move to.	
A following move is attempted and no motion occurs.	• The SX may not be in the Following mode (FSI).	• Set the SX to Following mode. Check your configuration and motion settings.
	• Limits are enabled, you may be in the absolute mode and are already at the position you are commanding the motor to move to.	
The unit may appear to not be responding to commands.	• If you defined a sequence and never issued XT, the SX still thinks you are defining a sequence.	• Issue an XT command at the end of the sequence to end sequence definition.

Problems & Solutions Table

Motor

If the motor fails to move, you should test the motor with an ***ohmmeter*** to examine the resistance between the motor connections. If the motor is not malfunctioning, the source of the problem is probably within the drive. If you operate a faulty drive with a reliable motor, you may damage the motor. If you find that the motor is not faulty, remove power, and remove the motor from the drive. Use the following steps to test the motor.

① Remove power from the system. Detach the motor from the drive.

② With the motor detached from the system, use an ohmmeter to check the resistance across Phase A. **It should be approximately 2 ohms**.

③ Now use the ohmmeter to check the resistance across Phase B. **It should be approximately 2 ohms too (*the resistance across Phase A and Phase B should be nearly identical*).**

④ Use the ohmmeter to check the resistance between Phase A and Phase B. **It should be infinite.**

⑤ Use the ohmmeter to check the resistance between Phase A and Earth (the motor case shaft). **It should be infinite.**

⑥ Use the ohmmeter to check the resistance between Phase B and Earth (the motor case shaft). **It should be infinite.**

⑦ Turn the shaft manually. **There should not be any torque**.

If the motor responds as described to each of these steps, it is probably functioning properly (it may still fail when connected to the drive because of the high motor voltage). The source of the problem is probably within the drive.

RS-232C Problems

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

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.

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

③ If you receive double characters, for instance typing **A** and receiving **AA**, the computer is set for half duplex mode. Change the setup to full duplex mode.

④ **Use DC common or signal ground as a reference, not earth ground.**

⑤ Cable lengths should not exceed 50 ft. 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.

⑥ To test the terminal or terminal emulation software and the RS-232C cable for proper three-wire communication, unhook the SX 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.

Software Debugging Tips

This section offers some helpful tips for debugging your programs or to understand why something may be happening. The SX has several tools that can be used to aid in the determination of a problem in the system design. The software tools are listed below:

R	Report Indexer Status
XTR	Trace Mode
DIN,DOUT	I/O Simulation
XST	Single Step
XS	Sequence Execution Status
DF	Displays Indexer Status
X	Displays state of the Indexer
DR	Displays Interface Option Status
FS	Displays Indexer Status Options
OS	Displays Homing/Jog Status
	Report backs
IS	Display inputs

Trace Mode

Trace mode is used to display what is occurring as you execute your sequence. By running the trace mode you can see what commands are being executed and if the program stops running you can see what command was last executed. The trace mode along with the interactive mode (**SSI**) will help you to find commands that the Indexer may not recognize. Trace mode is enabled and disabled with the **XTR** command. When enabled, you will execute sequences as you normally would using **XR**. As the sequence is running, the commands are displayed on the screen. **XTR1** enables the Trace mode, **XTRØ** disables it.

I/O Simulation

I/O simulation can be done without actually physically toggling the inputs or outputs using the **DIN** and **DOUT** commands. These commands can be used to your sequences and program.

Displaying SX Status

There are several commands that you can use to check the SX's status. You can report back the value or setting of almost all of the SX commands by typing the device address followed by the command then a carriage return or a space bar. In this way, you can find out what values you have entered in different commands. The **DR** command reports the current state of the SX. Use this to verify that the SX is configured as you want it. You can use four other report back commands to determine the unit's state. These commands report a binary number. Each bit of the report corresponds to different functions or modes that the SX could be in.

DFX Command Report Back

The **DFX** command reports the SX 's current states and conditions.

```
32  31  3Ø  29  23  27  26  25  24  23  22  21  20  19  18  17  16  15  14  13  12  11  10  9  8  7  6  5  4  3  2  1  Ø
*Ø   Ø   Ø   Ø__Ø   Ø   Ø   Ø    Ø__Ø   Ø   Ø    Ø__Ø   Ø   Ø    Ø__Ø   Ø   Ø    Ø__Ø   Ø   Ø  Ø__Ø  Ø  Ø  Ø__Ø  Ø  Ø  Ø
```

Bit representations —25-32,13 reserved

Bit

24 Mode Profile: no = Ø, yes = 1

23 Mode Alternate: no = Ø, yes = 1

21 Hit a software CCW limit: no = Ø, yes = 1

20 Hit a software CW limit: no = Ø, yes= 1

19 Home limit not found = Ø, found = 1

18 Jogging: no = Ø, yes = 1

17 Queued for **RM** mode: no = Ø, yes = 1

16 Run sequence on power up: no = Ø, yes = 1

15 **U** command: not waiting =, waiting = 1

14 Waiting for a trigger: no = Ø, yes = 1

12 Back up to home limit: no = Ø; yes = 1

11 Home move—High-speed: no = Ø; in process = 1

10 Execute a sequence: no = Ø; yes = 1

9 Wait on a timer: no = Ø, yes = 1

8 Hit a CCW limit : no = Ø, yes = 1

7 Hit a CW limit : no = Ø, yes = 1

6 **PS** (Pause): not waiting = Ø, waiting = 1

5 Abs. move direction: CW = Ø, CCW = 1

4 Incremental/absolute: **MPI =** Ø, **MPA** = 1

3 Mode preset = Ø; Continuous = 1

2 Move direction: CW = Ø, CCW = 1

1 Preset move: not moving = Ø, moving = 1

Ø Continuous move: not moving = Ø, moving = 1

FS reports a binary word that has various interface options associated with each bit. **OS** reports the homing options and the Jog enable option. **SS** reports various Indexer software options. If **1** is in the bit location, the feature or mode is enabled. Refer to the example below.

```
      A B C D_E F G H_I J K L_MNOP_QRST
SS    *Ø Ø Ø Ø  _ Ø Ø Ø Ø _Ø Ø Ø Ø_Ø Ø Ø Ø _Ø Ø Ø Ø
FS    *Ø Ø Ø Ø  _ Ø Ø Ø Ø _Ø Ø Ø Ø_Ø Ø Ø Ø
OS    *Ø Ø Ø Ø  _ Ø Ø Ø Ø
```

SSA RS232 Echo: Echo on = Ø, Echo off = 1
SSD Alternate Mode Stop: end of cycle = Ø, immediately = 1
SSG Clear/Save buffer on limit : clear = Ø, Save = 1
SSH Clear/Save buffer on stop: clear = Ø, Save = 1
SSI Enable/Disable Interactive Mode
SSJ Enable/Disable Continuous scan mode
SSL Resume execution enable
SSN Enable Error Message mode
SSQ Enable Drive Fault indicator

FSD Enable/Disable Stop on Stall
FSI Enable/Disable Following
FSK Enable Following Learn mode
FSL Enable following Self Correction mode
FSN Enable Pulse and Direction following
FSP Enable Position Tracking

OSA	Define Active Level of Limit Switches 1 = Active Low (N.O.)
OSB	Enable back-up to home switch
OSC	Define Active Level of Home switch 1=Active high signal
OSD	Enable Encoder Z channel Input for Homing
OSE	Enable Jogging
OSG	Define Final Home approach direction 1=CCW
OSH	Define Active edge of home switch to stop on 1=CCW

Returning the System

If your SX system is faulty, you must return the drive and motor for replacement or repair. A failed drive can damage motors. If you must return your SX to effect repairs or upgrades, use the following steps:

Step ①

Get the serial number and the model number of the defective unit(s), and a purchase order number to cover repair costs in the event the unit is determined by Parker Compumotor to be out of warranty.

Step ②

Before you ship the drive to Parker Compumotor, have someone from your organization with a technical understanding of the SX Indexer/Drive and its application include answers to the following questions:

❑ What is the extent of the failure/reason for return?

❑ How long did it operate?

❑ How many units are still working?

❑ How many units failed?

❑ What was happening when the unit failed (i.e., installing the unit, cycling power, starting other equipment, etc)?

❑ How was the product configured (in detail)?

❑ What, if any, cables were modified and how?

❑ With what equipment is the unit interfaced?

❑ What was the application?

❑ What was the system sizing (speed, acceleration, duty cycle, inertia, torque, friction, etc.)?

❑ What was the system environment (temperature, enclosure, spacing, unit orientation, contaminants, etc.)?

❑ What upgrades, if any, are required (hardware, software, user guide)?

Step ③

Call Parker Compumotor’s Applications Engineering Department [(800) 358-9070] for a .i.Return Material Authorization (RMA); number. Returned products cannot be accepted without an RMA number.

Step ④

Ship the unit to:

Parker Compumotor Corporation
5500 Business Park Drive, Suite D
Rohnert Park, CA 94928
Attn: RMA # xxxxxxx

INDEX

APPENDICES

Command Listing

"—Quote Command

#—Step Sequence

;—Comment Field

A—Acceleration

[ABS]—Absolute Encoder Comparison

AD—Deceleration

[AND]—Boolean AND Operator

B—Buffer Status Report

BCPE—Buffered Configure Position Error

BCPG—Buffered Configure Proportional Gain

BCPM—Buffered Configure Proportional Max.

BL—Backlash

BRK—Break Command

BS—Buffer Status Report

C—Continue

CEW—Configure Error Window

CIT—Configure In Position Time

CPE—Configure Position Error

CPG—Configure Proportional Gain

CPM—Configure Proportional Maximum

CR—Carriage Return

D—Distance

DCLR—Clear Display

DCNT—Enable/Disable Pause and Continue

DFS—Display Flags for Drive Parameters

DFX—Display Flags for Indexer Status

DIN—Disable Inputs

DLED—Turn RP240 LEDs On/Off

DOUT—Disable Outputs

DP—Distance Point

DPA—Display Actual Position

DPC—Position Cursor

DPE—Display Position Error

DPI—Display Position Indexer

DR—Display Parameters

DRD—Read Distance Via Parallel I/O

DSTP—Enable/Disable Stop

DTXT—Display Text Data on RP240 LCD

DVA—Display Actual Velocity

DVO—Display Variable Data on RP240 LCD

DVS—Display Velocity Setpoint

DW—Dead Band Window

E—Enable Communication Interface

ELSE—Else

ER—Configure Encoder Resolution

[ER]—Error Flag Operator

F—Disable Communication Interface

FAC—Set Following Synchronization Rate

FBS—Following Base

FC—Following Learn Count

FEN—Set Following Synchronization Count

[FEP]—Following Encoder Position Comparison

FIN—Following Increment

[FL]—User Flag Operator

FOL—Following Percent

FOR—Following Ratio

FP—Following Encoder Point

FPA—Following Encoder Absolute Point

FRD—Read Following Via Parallel I/O

FS—Encoder Function Report

FSA—Enable Following Mimic Mode

FSB—Enable/Disable Encoder Step Mode

FSC—Enable/Disable Position Maintenance

FSD—Stop on Stall

FSF—Enable Following Synchronized Acceleration

FSG—Stop Position Maintenance Move on Limit Encountered

FSH—Abort Position Maintenance on Limit Encountered

FSI—Enable/Disable Following Mode

FSK—Set Following Learn Mode

FSL—Enable/Disable Self Correction Mode

FSM—Set Absolute Encoder

FSN—Set Pulse Following

FSP—Set Tracking Mode

G—Go

GD—Go Defined

GDEF—Move Definition

GH—Go Home

GHA—Go Home Acceleration

GHAD—Go Home Deceleration

GHF—Go Home Final Velocity

GHV—Go Home Velocity

GOSUB—GOSUB Sequence

GOTO—GOTO Sequence

H—Set Direction

^H—Backspace

HALT—Halt

ID—Immediate Distance

IF—If

IN—Set Input Functions

[IN]—Input Flag Operator

INA—CW Limit Status

INB—CCW Limit Status

INC—Home Limit Status

INL—Set Active Input Level

INR—Enable/Disable Registration Input

IO—Immediate Output

IS—Input Status Report

IV—Immediate Velocity

JA—Jog Acceleration
JAD—Jog Deceleration
JVH—Jog Velocity (High)
JVL—Jog Velocity (Low)

K—Kill

L—Loop
LAD—Limit Deceleration
LD—Limit Disable
LF—Line Feed
LRD—Read Loop Count via Parallel I/O

MC—Mode Continuous
MN—Mode Normal
MPA—Mode Position Absolute
MPI—Mode Position Incremental
MPP—Mode Position Profile
MR—Configure Motor Resolution
MSP—Maximum Synchronization Percent
MV—Maximum Correction Velocity
MW—Set Motor Waveform

N—End of Loop
NG—End Position Profile
NIF—End of IF
NWHILE—End of While

O—Output
OFF—Off
ON—On
[OR]—Boolean OR Operator
OS—Function Set-Up Report
OSA—Define Active State of Limit Switch/Sensor
OSB—Backup to Home Switch
OSC—Define Active State of Home Switch
OSD—Enable Encoder Z Channel Input
OSE—Jog Enable
OSF—Acknowledge STOP & KILL Inputs On Power-up
OSG—Final Homing Direction
OSH—Reference Edge of Home Switch
OSI—Save Sequence Scan Mode on Stop
OSJ—Configure Z-Channel Search Mode
OUT—Output Functions
OUTL—Set Active Output Level
OUTP—Output on Position

PF—Follower Position Report
PFZ—Set Follower Counter to Zero
PHZ—Zero Motor Phase
[POS]—Position Counter Comparison
PR—Absolute Position Report
PS—Pause
PU—Configure Square Wave Output
PUL—Activate Square Wave Output
PX—Report Encoder Position
PZ—Set Absolute Counter to Zero

Q0—Exit Velocity Profiling Mode
Q1—Enter Velocity Profiling Mode

R—Request SX Status
RA—Limit Switch Status Report
RB—Loop, Pause, Shutdown, Trigger Status Report
REG—Configure Registration Move
REPEAT—Repeat
RG—Go Home Status Report
RIFS—Return Indexer to Factory Settings
RM—Rate Multiplier in Velocity Streaming Mode
RS—Report Status Sequence Execution
RSE—Report Servo Errors
RSIN—Set Variables Interactively
RV—Revision Level
RVV—Report RevisionVerbose

S—Stop
SCR—Set Standby Current Reduction
SFL—Set User Flag
SL—Software Limits
SLD—Software Limits Disable
SN—Scan Delay Time
SP—Set Position Absolute
SPA—Set Position Zero
SS—Function Set-Up Report
SSA—RS-232 Echo Control
SSC—Enable End of Move In-Position Window
SSG—Clear/Save the Command Buffer on Limit
SSH—Clear/Save the Command Buffer on Stop
SSI—Enable/Disable Interactive Mode
SSJ—Enable/Disable Continuous Scan Mode
SSL—Enable Resume Execution
SSN—Set Message Mode
SSP—Clear All Position Offsets with PZ, PFZ
SSR—Enable/Disable Fault On Shutdown
ST—Shutdown
STOP—Stop
STR—Set Strobe Output Delay Time

T—Time
TD—Set Input Debounce Time
TDR—Set Registration Input Debounce Time
TEST—Test
TF—Set Following Time
TM—Move Time Report
TR—Wait for Trigger
TRD—Read Timer from Parallel I/O
TS—Trigger Input Status
TW—Thumbwheel Input Mode
TX—Transmit Variable and String

U—Pause and Wait for Continue
UNTIL—Until

V—Velocity
VAR—Variables
VARD—Read Variables via Parallel I/O
VARN=FUN—Enable and Read Function Keys
VARN=NUM—Enable and Read Numeric Keypad
VRD—Read Velocity via Parallel I/O
VS—Start/Stop Velocity

W1—Signed Binary Position Report
W2—Hexadecimal Position Report
W3—Signed Hexadecimal Position Report
WHEN—Set When Condition
WHILE—While

XBS—Sequence Memory Available Report
XC—Sequence Checksum Report
XD—Sequence Definition
XDIR—Sequence Directory
XE—Sequence Erase
XEALL—Erase All Sequences
XFK—Set Fault or Kill Sequence
XG—GOTO Sequence
XQ—Sequence Interrupted Run Mode
XR—Run a Sequence
XRD—Read Sequence via Parallel I/O
XRP—Sequence Run with Pause
XS—Sequence Execution Status
XST—Sequence Step Mode
XT—Sequence Termination
XTR—Set Trace Mode
XU—Upload Sequence
XWHEN—Set When Sequence

Y—Stop Loop

Z—Reset

SX Example Programs

The following programs are example sequences to show general programming flow for the SX Indexer/Drive. These are not intended to be an actual application solution. For more example programs call the Compumotor BBS number at 707-584-4059 or look on the Xware diskette.

```
;REFER_TO_THE_SX_SOFTWARE_REFERENCE_GUIDE_OR_EARLIER_IN
;THIS_USER_GUIDE_FOR_MORE_INFORMATION_CONCERNING_SPECIFIC
;COMMANDS_AND_WHAT_THEIR_FUNCTIONALITY_IS_IN_THESE_EXAMPLE
;PROGRAMS.

;This_program_is_the_power_up_sequence.__Sequence_#100_is_always_the_power
;up_sequence.__All_of_the_applicable_set_up_parameters_for_the_other_sequences
;are_typically_set_in_the_power_up_sequence.__Anything_that_needs_to_be_set
;or_enabled_but_isn't_changed_again_should_be_placed_in_the_power_up_
;sequence.__This_example_also_uses_the_sequence_selecting_feature_of_the
;SX.__Anytime_it_is_not_in_the_middle_of_a_sequence_already,_a_new_
;sequence_can_be_run_by_activating_the_appropriate_inputs_that_are_defined
;as_sequence_select_inputs.
1XE100          ;erase_sequence_#100_(power_up_sequence)
1XD100          ;begin_definition_of_sequence_#100

1XFK0           ;disable_fault_or_kill_sequence_NOTE:_this_allows_
                ;someone_to_be_able_to_kill_the_program_execution
                ;immediately_after_powering_up_so_they_can_edit
                ;the_programs

1OSA0           ;set_hard_limit_inputs_for_normally_closed_switches
1LAD100         ;set_limit_deceleration_to_100_rps
1LD0            ;enable_both_hard_limits
1GHA50          ;set_go_home_acceleration_to_50_rps

1GHAD50         ;set_go_home_deceleration_to_50_rps
1GHF.5          ;set_final_homing_velocity_to_0.5_rps
1GHV2           ;set_initial_homing_velocity_to_2_rps
1OSC0           ;set_home_input_active_level_for_normally_open_switch
1OSB1           ;enable_backup_to_home_function
1OSG1           ;set_final_homing_approach_direction_to_CCW
1OSH0           ;set_home_switch_reference_edge_to_CW(nearest_to_CW_limit)

1INL0           ;set_inputs_active_level_for_normally_open_switches
1IN1A           ;set_input_#1_to_a_trigger_input_(default_type)
1IN2C           ;set_input_#2_to_a_kill_input
1IN3J           ;set_input_#3_to_a_jog_CW_input
1IN4K           ;set_input_#4_to_a_jog_CCW_input
1IN5L           ;set_input_#5_to_a_jog_velocity_select_input
1IN6B           ;set_input_#6_to_a_sequence_select_input_(weighting=1)
1IN7B           ;set_input_#7_to_a_sequence_select_input_(weighting=2)
1IN8B           ;set_input_#8_to_a_sequence_select_input_(weighting=4)
1SN50           ;set_sequence_scan_input_debounce_time_to_50_milliseconds

1TD10           ;set_general_input_debounce_time_to_10_milliseconds
1JA100          ;set_jog_acceleration_to_100_rps

1JAD100         ;set_jog_deceleration_to_100_rps
1JVH10          ;set_jog_velocity_high_to_10_rps
1JVL0.5         ;set_jog_velocity_low_to_0.5_rps

1OUT1A          ;set_output_#1_to_a_programmable_output_(default_type)
1OUT2J          ;set_output_#2_to_a_strobe_output
1OUT3J          ;set_output_#3_to_a_strobe_output
1OUT4J          ;set_output_#4_to_a_strobe_output

1STR100         ;set_strobe_output_delay_time_to_100_milliseconds

1XQ1            ;set_sequence_interrupted_run_mode

1SSJ1           ;enable_sequence_scan_mode

1XFK99          ;set_fault_or_kill_sequence_to_be_sequence_99

1SFL0           ;initialize_user_flag_to_zero

1XT             ;end_definition_of_sequence_#100

;This_sequence_performs_a_simple_preset_move_(MN)_at_5_rps.__It_will
;cause_the_motor_to_travel_a_distance_of_50000_steps_(2_revolutions_with
;the_default_drive_resolution_setting)._This_sequence_can_be_run_by_
;activating_input_#6_(sequence_select_weighting_of_1)
1XE1                                    ;erase_sequence_#1
1XD1                                    ;begin_definition_of_sequence_#1
1A50                                    ;set_acceleration_to_50_rps
1AD50                                   ;set_deceleration_to_50_rps
1V5                                     ;set_velocity_to_5_rps
1D50000                                 ;set_distance_to_50000_steps
1MPI                                    ;set_SX_to_incremental_positioning_mode
1MN                                     ;set_SX_to_preset_distance_mode
1G                                      ;start_move_(GO)
1XT                                     ;end_definition_of_sequence_#1
;This_sequence_is_a_simple_example_of_a_registration_move.__The_SX_is_put_into
;continuous_mode_and_the_continuous_move_is_started_with_the_G_command.__
;It_will_move_at_3_rps_until_the_registration_input_goes_active_(or_a_limit_is_
;encountered).__After_the_registration_input_goes_active,_the_motor_will
;decelerate_at_10_rpss_to_a_velocity_of_0.1_rps_and_travel_25000_steps_from_
;the_point_at_which_the_registration_input_was_seen_as_active._(within_50_
;seconds).__If_it_never_sees_the_registration_input,_the_motion_will_stop
;at_the_software_or_hardware_limits._This_sequence_can_be_run_by_
;activating_input_#7_(sequence_select_weighting_of_2)
1XE2                                    ;erase_sequence_#2
1XD2                                    ;begin_definition_of_sequence_#2
1MC                                     ;set_SX_to_continuous_positioning_mode
1A100                                   ;set_acceleration_to_100_rps
1AD100                                  ;set_deceleration_to_100_rps
1V3                                     ;set_velocity_to_5_rps
1REG1,A100,AD10,V0.1,D25000             ;define_the_registration_move
1G                                      ;start_continuous_move_(GO)
1XT                                     ;end_definition_of_sequence_#2
;This_sequence_performs_a_more_complex_move_profile.__The_motion
;begins_at_velocity_5_rps_and_then_runs_at_this_velocity_for_2_seconds.
;After_the_2_seconds,_the_velocity_is_changed_to_1_rps.__The_program
;then_waits_for_a_trigger_input_(trigger_#1).__After_the_trigger_is
;activated_the_velocity_is_changed_to_0.1_rps_for_the_rest_of
;the_move.__The_MPP_mode_is_used_to_allow_the_SX_to_
;process_other_commands_while_a_move_is_in_progress._
;If_not_in_MPP_mode_the_SX_will_wait_until_the_current_move
;is_completed_before_processing_the_next_command._This_sequence_can_be_run_by_
;activating_input_#6_&_input_#7_together_(sequence_select_weighting_of_1_and_2_=_3)
1XE3                                    ;erase_sequence_#3
1XD3                                    ;begin_definition_of_sequence_#3
1MN                                     ;set_SX_to_preset_distance_mode
1MPP                                    ;enable_mode_position_profile
1A50                                    ;set_acceleration_to_50_rps
1AD50                                    ;set_deceleration_to_50_rps
1V5                                     ;set_velocity_to_5_rps
1D500000                                ;set_distance_to_500000_steps
1G                                      ;start_move_(GO)
1T2                                     ;wait_2_seconds_before_continuing
1V1                                     ;reduce_velocity_to_1_rps
1TR1                                    ;wait_for_trigger_#1
1V0.1                                   ;reduce_velocity_to_0.1_rps
1NG                                     ;exit_MPP_mode
1XT                                     ;end_definition_of_sequence_#3
;This_sequence_performs_a_homing_function.__It_starts_the_
;homing_move_with_the_GH_command_and_will_stay_on_this
;command_until_the_home_switch_is_found_or_both_end_of
;travel_limits_are_hit.__The_position_counter_is_then_zeroed
;after_the_successful_home.__If_both_limits_are_hit_before_the
;home_switch_is_found_the_fault_sequence_is_run._This_sequence_
;can_be_run_by_activating_input_#8_(sequence_select_weighting_of_4)
```

```
1XE4                                  ;erase_sequence_#4
1XD4                                  ;begin_definition_of_sequence_#4
1GH                                   ;start_a_go_home_move
1PZ                                   ;zero_the_position_counter
1XT                                   ;end_definition_of_sequence_#4
;This_sequence_puts_the_SX_into_jogging_mode_so_the_motor’s
;motion_can_be_controlled_with_the_defined_jog_inputs_and
;jog_velocity_select_input.__The_sequence_waits_for_the_
;trigger_input_to_go_active_to_signal_an_end_to_the_jog
;mode._This_sequence_can_be_run_by_activating_input_#6__
;&_input_#8_together_(sequence_select_weighting_of_1_and_4_=_5)
1XE5                                  ;erase_sequence_#5
1XD5                                  ;begin_definition_of_sequence_#5
1OSE1                                 ;enable_jog_mode
1TR1                                  ;wait_for_trigger_#1
1OSE0                                 ;disable_jog_mode
1XT                                   ;end_definition_of_sequence_#5
;This_is_the_fault_or_kill_sequence_(error_handling).__The_SX_will_run_this
;sequence_if_a_kill_is_commanded_or_a_general_fault_occurs.__The_SX_User
;Guide_lists_the_faults_that_will_call_the_fault_sequence.__The_sequence_first
;disables_any_mode_or_condition_that_may_still_be_active_to_prevent_further
;motion_from_occuring.__It_then_checks_the_error_flag_to_see_what_error_
;occurred.__This_fault_sequence_is_not_comprehensive_of_all_the_errors_that
;are_flagged_by_the_error_flag.__The_sequence_finally_prompts_the_operator
;if_they_want_to_rehome_and_continue_or_just_continue.__If_a_system_fault
;occurs_the_program_flow_will_stop.
1XE99                                 ;erase_sequence_#99_(fault_or_kill_sequence)
1XD99                                 ;begin_definition_of_sequence_#99
1XFK0                                 ;disable_the_fault_or_kill_sequence
1OSE0                                 ;disable_jog_mode_if_active
1SSJ0                                 ;disable_sequence_select_mode
1NG                                   ;exit_MPP mode_if_in_it
1IF(ER1)                              ;check_if_CCW_limit_was_hit
1"CCW_Limit_Hit                       ;print_error_message
1SFL1                                 ;set_user_flag
1NIF                                  ;end_of_if_statement
1IF(ERX1)                             ;check_if_CW_limit_was_hit
1"CW_Limit_Hit
1SFL1
1NIF
1IF(ERXX1)                            ;check_if_it_is_a_system_fault
1"System_Fault_Occured
1ST1                                  ;shutdown_the_drive
1SFL1
1XG98                                 ;goto_non-existant_sequence_to_stop_everything
1NIF
1IF(FL0)                              ;check_if_user_flag_has_been_set
1"Kill_Commanded                      ;if_it_hasn’t_then_kill_caused_the_jump_to_#99
1NIF
1CR                                   ;print_carriage_return
1"Activate_input_#1_to_resume_sequence_select_mode.
1CR
1"Activate_input_#8_to_rehome_and_then
1"resume_sequence_select_mode.
1CR
1CR
1REPEAT                               ;wait_for_input_to_be_activated
1UNTIL(INXXXX1_OR_INXXXXXXXXXXX1)
1IF(INXXXX1)                          ;check_if_input_#1_is_active
1XG100                                ;goto_sequence_#100
1NIF
1IF(INXXXXXXXXXXX1)                   ;check_if_input_#8_is_active
1XR4                                  ;gosub_sequence_#4
1XG100                                ;goto_sequence_#100
1NIF
1XT
```

APPENDIX C

LVD Installation Instructions

Complying with the Low Voltage Directive (LVD)

The SX Drive, 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 Economic Community. To install the SX Drive so that it complies with LVD, you must follow the additional procedures described in this appendix. If this is not done, the protection of the product may be impaired.

For more information about LVD, see 73/23/EEC and 93/68/EEC, published by the European Economic Community (EEC).

Additional Installation Procedures for LVD Compliance

Environmental Conditions

Pollution Degree	The SX Drive is designed for pollution degree 2.
Installation Category	The SX Drive is designed for installation category II.

Electrical

Connecting and Disconnecting Power Mains

The SX Drive's protective earth connection is provided through its make first/break last earth terminal on the power mains connector. You must reliably earth the SX Drive's protective earth connection. Attach or remove the SX Drive's power plug only while input power is OFF.

Isolation Transformer

The SX Drive's mains voltage is limited to 120 VAC nominal, single phase. If your mains voltage is higher, use an isolation transformer located between the power mains and the SX Drive. Your isolation transformer should be insulated to ~2300V rms.

CAUTION
Do not use an autotransformer.

Line Fuses

Line fuses need to be added to protect the transformer and associated wiring. If the live wire cannot be readily identified, fuse both phase conductors. The value of fuse required is given by:

(1.5 x VA)/(supply volts) [amps]

Fuse types should be anti-surge HBC.

Providing a Protective Earth Connection for Motors

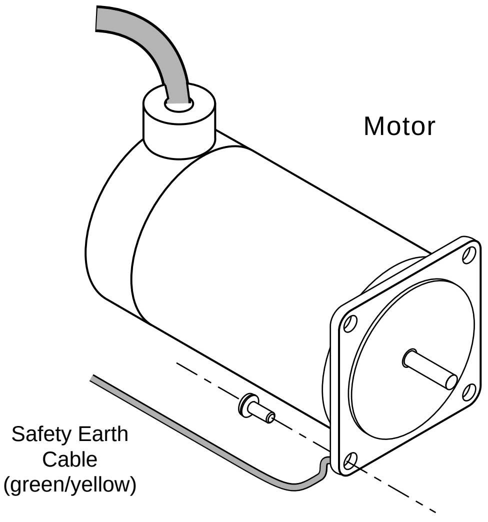
You must provide a connection from the motor to a reliable system earth. This connection provides a protective earth for the motor contact point. The motor's protective earth connection is important for safety reasons, and *must not be omitted*.

Make connections according to the following instructions and diagram:

① Use a spade lug 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 recognized green/yellow safety conductor 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 W. Use thicker gauge wire if the resistance is too high.



Providing Protective Earth Connection for Motor

Mechanical

Installing in an Enclosure

The SX Drive must be installed within an enclosure. The enclosure's interior must not be accessible to the machine operator. The enclosure should be opened only by skilled or trained service personnel.

Servicing the SX Drive

Changing Firmware

Only skilled or trained personnel should change firmware.

Changing Batteries

The SX Drive contains a replaceable lithium battery, of type Duracell DL2450, or Sanyo CR2450, or equivalent. Only skilled or trained personnel should change batteries.

CAUTION

Danger of explosion if battery is incorrectly replaced.

Replace only with the same or equivalent type recommended by the manufacturer.

Disposal of Batteries

Dispose of batteries in accordance with local regulations.

Do Not Replace Fuses

The SX Drive has no fuses designed to be replaced by the user. Fuse failure indicates that other components have also failed. Fuses and other components should only be replaced by Compumotor or its designated repair facilities.

Thermal Safety

The SX Drive May Be Hot

The SX Drive may reach high temperatures during normal operations, and may remain hot after power is removed.

The Motor May Be Hot

The motor may reach high temperatures during normal operations, and may remain hot after power is removed.

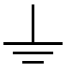
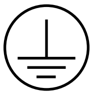
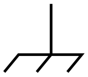
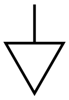
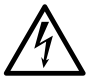
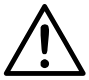
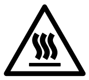
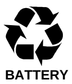
Sonic Pressure

High Sound Level

The sound level from some large frame step motors (NEMA 34, NEMA 42, and larger) may exceed 85 dBA. Actual sound level is application dependent, and varies with motor loads and mounting conditions. Measure the sound level in your application; if it exceeds 85 dBA, install the motor in an enclosure to provide sound baffling, or provide ear protection for personnel.

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
	Recycle Battery